



UNIVERSIDADE DOS AÇORES
DEPARTAMENTO DE MATEMÁTICA

Conjetura de Goldbach - Uma visão Aritmética

JOSÉ EMANUEL SOUSA

PONTA DELGADA
ABRIL DE 2013



UNIVERSIDADE DOS AÇORES
DEPARTAMENTO DE MATEMÁTICA

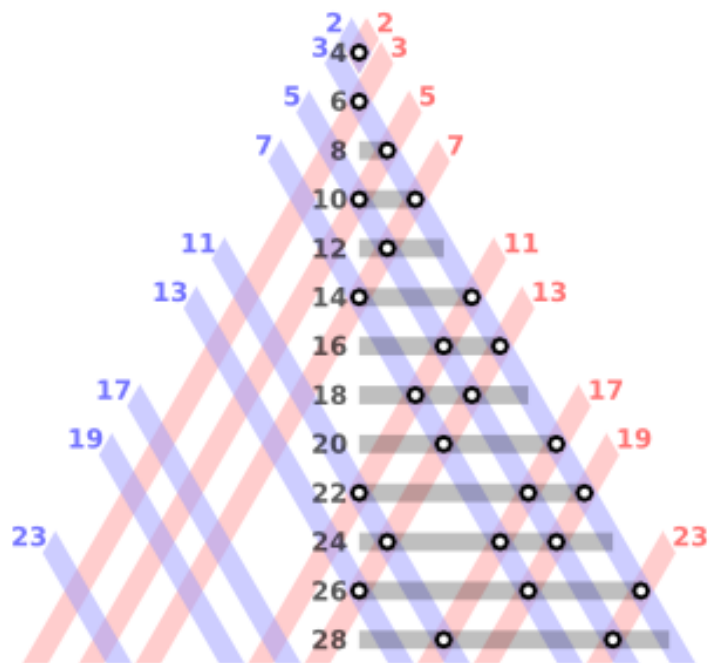
MESTRADO EM MATEMÁTICA PARA PROFESSORES

Conjetura de Goldbach - Uma visão Aritmética

JOSÉ EMANUEL SOUSA

Dissertação apresentada para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Matemática para Professores, sob a orientação da Professora Doutora Ana Paula de Ornelas Garrão e da Professora Doutora Margarida de Jesus Silva Raposo Dias, do Departamento de Matemática da Universidade dos Açores.

PONTA DELGADA
ABRIL DE 2013



Partições de Goldbach dos números pares menores que 30.

Resumo

Esta dissertação insere-se no Mestrado em Matemática para Professores, da Universidade dos Açores, e centra-se no estudo da Conjetura de Goldbach visando apresentar e aprofundar de forma sistemática e sintetizada este problema da Matemática, ainda em aberto, do ponto de vista da Teoria Aritmética dos Números.

Sendo a Conjetura de Goldbach considerada a mais famosa conjectura, ainda por resolver, sobre os números primos, uma parte considerável do capítulo 1 é dedicada aos mesmos, bem como a outros conceitos essenciais da Teoria dos Números. Está reunido neste documento os principais resultados relacionados com a Conjetura de Goldbach, o seu enquadramento histórico, bem como algumas das diferentes abordagens desenvolvidas, tendo por base a Teoria Aritmética dos Números.

No capítulo 3 é apresentada e explorada uma nova abordagem à conjectura, que tem na sua génese um método que permite determinar diferentes partições de Goldbach, por vezes até todas, de uma forma prática e eficaz, sem recorrer a algoritmos complexos, tendo por base apenas propriedades entre os números, nomeadamente a divisibilidade.

Abstract

This work is part of the Master of Mathematics for Teachers, University of the Azores, and focuses on the study of Goldbach's Conjecture seeking to present and develop in a systematic and synthesized form, based in Arithmetic Number Theory point of view, this Math problem still open.

Being the Goldbach Conjecture considered the most famous conjecture, unsolved, about prime numbers, a considerable part of Chapter 1 is devoted to them, as well as other essential concepts of Number Theory. In this document are gathered the main results related to the Goldbach Conjecture, its historical background, as well as some of the different approaches developed based on Arithmetic Number Theory.

In Chapter 3 is presented and explored a new approach to the conjecture, which has its genesis in a method that allows to find different partitions of Goldbach, sometimes even all, in a practical and effective way, without using complex algorithms, based on properties of numbers, including the divisibility.

Agradecimentos

Uma dissertação resulta de um esforço pessoal e colectivo. Nesse sentido manifesto o meu sincero agradecimento: à Professora Doutora Ana Paula de Ornelas Garrão e Professora Doutora Margarida Raposo Dias, do Departamento de Matemática da Universidade dos Açores, pela imprescindível orientação, partilha de saber, permanente disponibilidade, e valiosas contribuições para o trabalho; à Carla, pelo apoio, incentivo, compreensão e encorajamento, durante todo este período, além de toda a ajuda no desenvolvimento das aplicações.

à curiosidade matemática

Considerações iniciais

“Primeiro estranha-se, depois entranha-se.”

Fernando Pessoa

A Conjetura de Goldbach é um tema que sempre despertou a curiosidade ao longo dos séculos pela sua aparente simplicidade. Contudo, ainda não é conhecida uma demonstração aceite pela Comunidade Matemática. Pessoalmente, o primeiro contacto com a Conjetura de Goldbach ocorreu numa aula de Teoria dos Números. De início estranhei, mas depois entranhou-se em mim, e ao longo dos últimos anos foi crescendo uma vontade de tentar, como tantos outros o sentiram, provar a Conjetura de Goldbach. Essa procura, infrutífera, culminou com a oportunidade de desenvolver esta dissertação, como um tributo a todos aqueles que, tal como eu, ficaram fascinados, e possivelmente intrigados, pelo facto da conjectura ainda ser um problema em aberto.

O capítulo 1, denominado “Conceitos e Teoremas Fundamentais”, é dedicado à revisão de conceitos fundamentais da Matemática relacionados com a Teoria dos Números, nomeadamente os conceitos de divisibilidade e congruência, bem como o conceito de número primo, para que seja possível ao leitor ter uma base de conhecimento que permita acompanhar a exploração dos conteúdos nos capítulos subsequentes.

No capítulo 2, designado por “Conjetura de Goldbach: Enquadramento histórico e evolução”, reunimos informação relevante sobre a Conjetura de Goldbach desde a sua primeira formulação até à atualidade, no contexto da Teoria Aritmética dos Números. Apresentamos uma pesquisa bibliográfica sobre o tema, nomeadamente

o enquadramento histórico, a descrição da conjectura e o trabalho de diferentes investigadores.

Numa última fase, no capítulo 3, intitulado “Conjetura de Goldbach: uma abordagem aritmética” apresentamos uma nova abordagem que visa uma contribuição pessoal sobre a Conjetura de Goldbach, suportada por proposições demonstráveis e reforçada recorrendo a exemplos significativos.

Índice

1	Conceitos e Teoremas Fundamentais	1
1.1	Divisibilidade e Congruências	1
1.1.1	Conceito de Divisibilidade	1
1.1.2	Máximo divisor comum e mínimo múltiplo comum	3
1.1.3	Algoritmo da divisão e Algoritmo de Euclides para o mdc	5
1.1.4	Congruências	7
1.2	Números Primos	14
1.3	Teoremas Subjacentes aos Números Primos	20
1.4	Testes de Primalidade e Aplicações dos números primos	28
1.4.1	Testes determinísticos versus testes probabilísticos	29
1.4.2	Aplicações dos números primos	32
2	Conjetura de Goldbach: Enquadramento histórico e evolução	39
2.1	Vida e obra de Goldbach	40
2.2	Enquadramento histórico	42
2.3	Explorando a Conjetura de Goldbach	45
2.4	Evolução da Teoria dos Números no contexto da Conjetura de Goldbach	49
2.5	Diferentes abordagens à conjetura	50
2.5.1	Roger Ellman	51

2.5.2	Kent Slinker	52
2.5.3	Bernard Farley	55
2.6	Verificação da Conjetura	59
3	Conjetura de Goldbach: uma abordagem aritmética	61
3.1	Método para determinar partições de Goldbach	61
3.2	Número de partições existentes versus determinadas por este método	72
3.3	Propriedades dos inteiros e relação com a Conjetura de Goldbach	77
	Referências Bibliográficas	85
A	Lista de números primos menores que 10000	87
B	Código fonte	91

Capítulo 1

Conceitos e Teoremas Fundamentais

1.1 Divisibilidade e Congruências

1.1.1 Conceito de Divisibilidade

Um dos conceitos mais básicos do estudo da Teoria dos Números é o conceito de divisibilidade.

Definição 1.1.1. *Diz-se que $a \neq 0$ divide b , ou a é um divisor de b , ou ainda que b é um múltiplo de a , se existe algum inteiro k , para o qual $b = ak$.*

Em notação, representa-se por $a|b$. Se a não é um divisor de b então escreve-se $a \nmid b$. Assim, as seguintes expressões são equivalentes a $a|b$:

- a divide b ;
- a é um divisor de b ;
- a é um fator de b ;
- b é divisível por a ;
- b é um múltiplo de a .

Exemplo 1.1.2. *Desta forma,*

$2|10$, porque $10 = 2 \cdot 5$;

$-3|15$, porque $15 = (-3) \cdot (-5)$;

$5|-20$, porque $-20 = 5 \cdot (-4)$;

$7 \nmid 20$, porque não existe k inteiro tal que $20 = 7k$.

Proposição 1.1.3. *Quaisquer que sejam os inteiros a , b e c , tem-se:*

- I. $a|a$, $1|a$ e $a|0$;
- II. se $a|b$ então $-a|b$ e $a|-b$;
- III. se $a|b$ e $c|d$, então $ac|bd$;
- IV. se $a|b$ e $b|c$, então $a|c$;
- V. se $a|b$ e $b|a$, então $a = \pm b$;
- VI. se $a|b$ e $a|c$, então $a|(bx + cy)$, para quaisquer inteiros x e y (combinação linear).

Demonstração.

I. É evidente que:

$$a = a \cdot 1, a = 1 \cdot a, 0 = 0 \cdot a$$

□

II. Se $a|b$, então existe k inteiro tal que $b = ak$. Assim:

$$b = ak = (-a) \cdot (-k)$$

Logo, $-a|b$. Por outro lado,

$$-b = -ak = a(-k)$$

Logo, $a|-b$.

□

III. Se $a|b$ e $c|d$ então existem k e k' inteiros tais que $b = ak$ e $d = ck'$.

Portanto:

$$bd = (ak)(ck') = (ac)(kk')$$

Logo, $ac|bd$.

□

IV. Se $a|b$ e $b|c$ então existem k e k' inteiros tais que $b = ak$ e $c = bk'$.

Portanto:

$$c = bk' = (ak)k' = a(kk')$$

Logo, $a|c$. □

V. Se $a|b$ e $b|a$ então existem k e k' inteiros tais que $b = ak$ e $a = bk'$.

Portanto:

$$a = bk' = (ak)k' = a(kk')$$

Assim, temos que $kk' = 1$, o que implica que $k = k' = \pm 1$. Logo, $a = \pm b$. □

VI. Se $a|b$ e $a|c$ então existem k e k' inteiros tais que $b = ak$ e $c = ak'$.

Portanto, quaisquer que sejam os inteiros x e y , temos que:

$$bx + cy = akx + ak'y = a(kx + k'y)$$

Logo, $a|(bx + cy)$. □

1.1.2 Máximo divisor comum e mínimo múltiplo comum

O máximo divisor comum e o mínimo múltiplo comum são conceitos subjacentes à divisibilidade em \mathbb{Z} .

Definição 1.1.4. *O máximo divisor comum entre dois números é o inteiro positivo d que satisfaz as condições:*

1. $d|a$ e $d|b$ (d é divisor de a e de b);
2. Se $c|a$ e $c|b$, então $c \leq d$ (d é o maior dos divisores comuns de a e de b).

Por outras palavras, dados dois números inteiros a e b e o conjunto dos divisores de cada um, o maior elemento comum aos dois conjuntos é o máximo divisor comum entre a e b .

O máximo divisor comum entre a e b representa-se por $\text{mdc}(a, b)$.

Assim, por exemplo, dados os inteiros 42 e 70, verifica-se que os divisores comuns a ambos são 1, 2, 7 e 14, logo o $\text{mdc}(42, 70) = 14$.

Teorema 1.1.5. *(Existência e unicidade do mdc) O máximo divisor comum entre dois inteiros existe e é único.*

Teorema 1.1.6. *O $\text{mdc}(a, b)$ é o menor inteiro positivo d que se pode escrever como combinação linear de a e b , isto é, existem inteiros x e y tais que:*

1. $\text{mdc}(a, b) = ax + by$;
2. Se $c = ax + by$, então $d \leq c$.

Teorema 1.1.7. *Seja d o máximo divisor comum entre os inteiros a e b . Então o $\text{mdc}(\frac{a}{d}, \frac{b}{d}) = 1$.*

Definição 1.1.8. *(Inteiros primos entre si) Sejam a e b dois inteiros. Diz-se que a e b são primos entre si se $\text{mdc}(a, b) = 1$.*

Teorema 1.1.9. *Dois inteiros são primos entre si se e só se existirem inteiros x e y tais que $ax + by = 1$.*

Definição 1.1.10. *(Mínimo múltiplo comum) O mínimo múltiplo comum entre dois números é o inteiro positivo m que satisfaz as condições:*

1. $a|m$ e $b|m$ (m é múltiplo de a e b);
2. Se $a|c$ e se $b|c$, com $c > 0$, então $m \leq c$. (m é o menor dos múltiplos comuns a a e b).

Por outras palavras, dados dois números inteiros não nulos a e b e o conjunto dos múltiplos de cada um, o menor elemento comum aos dois conjuntos é o mínimo múltiplo entre a e b .

O mínimo múltiplo comum de a e b representa-se por $\text{mmc}(a, b)$.

Assim, por exemplo, dados os inteiros 15 e 35, verifica-se que os múltiplos comuns a ambos são 105, 210, \dots , logo o $\text{mmc}(15, 35) = 105$.

Teorema 1.1.11. (*Relação entre o mdc e o mmc*) Para a e b , inteiros positivos, tem-se que:

$$\text{mdc}(a, b) \cdot \text{mmc}(a, b) = ab$$

As demonstrações dos teoremas anteriores podem ser consultadas em [Filho (1981)].

1.1.3 Algoritmo da divisão e Algoritmo de Euclides para o *mdc*

Na relação de divisibilidade definida no conjunto dos inteiros nem todos os elementos de \mathbb{Z} estão em relação entre si, pois dados dois números inteiros a e b , pode não existir k inteiro, tal que $b = ak$, o que significa que b não divide a . Nesse caso, a divisão de a por b não é exata, isto é, deixa um resto não nulo.

Teorema 1.1.12. *Sejam a e b dois inteiros, com $b \neq 0$. Existem e são únicos os inteiros q e r que satisfazem as condições:*

$$a = bq + r, \quad 0 \leq r < |b|$$

Corolário 1.1.13. (*De Euclides*) Se $a = bq + r$, então $\text{mdc}(a, b) = \text{mdc}(b, r)$.

Demonstração.

Seja $d_1 = \text{mdc}(a, b)$ e $d_2 = \text{mdc}(b, r)$. Vamos provar que $d_1 = d_2$.

Ora,

$$d_1 | a \text{ e } d_1 | b \Rightarrow d_1 | (a - bq) \Rightarrow d_1 | r$$

Por outro lado,

$$d_2 | b \text{ e } d_2 | r \Rightarrow d_2 | (r + bq) \Rightarrow d_2 | a$$

Deste modo,

$$d_2 | a \text{ e } d_2 | b \Rightarrow d_2 \leq d_1 \quad ; \quad d_1 | r \text{ e } d_1 | b \Rightarrow d_1 \leq d_2$$

Logo, tem-se que $d_1 = d_2$. □

Observação 1.1.14.

a. Se $b \nmid a$ e $a > 0$, então o resto da divisão de b por a é diferente de 0. O resto da divisão é um inteiro r que satisfaz:

$$a = bq + r, \quad 0 < r < a$$

b. Se $b|a$ e $a > 0$, então o resto da divisão de a por b é 0.

Algoritmo de Euclides para o mdc

O máximo divisor comum entre dois inteiros pode ser determinado utilizando o algoritmo de Euclides, que consiste em preencher o seguinte quadro:

mdc	q_1	q_2	\dots	\dots	\dots
a	b	r_1	r_2	\dots	\dots
r_1	r_2	\dots	\dots	d	0

O máximo divisor comum é o último resto diferente de zero.

Exemplo 1.1.15. Determinar o $\text{mdc}(315, 44)$. O quadro associado é:

mdc	7	6	3	2
315	44	7	2	1
7	2	1	0	

Ora,

$$315 \equiv 7(\text{mod } 44) \Leftrightarrow \text{mdc}(315, 44) = \text{mdc}(44, 7)$$

$$44 \equiv 2(\text{mod } 7) \Leftrightarrow \text{mdc}(44, 7) = \text{mdc}(7, 2)$$

$$7 \equiv 1(\text{mod } 2) \Leftrightarrow \text{mdc}(7, 2) = \text{mdc}(2, 1)$$

$$2 \equiv 0(\text{mod } 1) \Leftrightarrow \text{mdc}(2, 1) = \text{mdc}(1, 1)$$

$$1 \equiv 0(\text{mod } 1) \Leftrightarrow \text{mdc}(1, 1) = \text{mdc}(0, 1) = 1$$

Assim, temos que $\text{mdc}(315, 44) = 1$.

1.1.4 Congruências

Definição 1.1.16. *Dados os inteiros a , b e m , diz-se que a e b são congruentes módulo m se e só se existir um inteiro k tal que $a - b = km$.*

Por outras palavras, a e b são congruentes módulo m se e só se m divide $a - b$.

A relação de congruência representa-se por:

$$a \equiv b(\text{mod } m)$$

Desta forma,

$$a \equiv b(\text{mod } m) \Leftrightarrow m|(a - b) \Leftrightarrow \exists k \in \mathbb{Z} : a - b = km$$

Quando m não divide $a - b$, então diz-se que a é incongruente a b módulo m , e representa-se por:

$$a \not\equiv b(\text{mod } m)$$

Exemplo 1.1.17. *Deste modo:*

$$18 \equiv 3(\text{mod } 5), \text{ porque } 5|(18 - 3)$$

$$21 \equiv 5(\text{mod } 4), \text{ porque } 4|(21 - 5)$$

$$16 \not\equiv 11(\text{mod } 10), \text{ porque } 10 \nmid (16 - 11)$$

$$35 \not\equiv 14(\text{mod } 11), \text{ porque } 11 \nmid (35 - 14)$$

Teorema 1.1.18. *(Inteiros congruentes) Dois inteiros a e b são congruentes módulo m se e só se os restos das divisões de a e b , por m , forem iguais.*

Demonstração.

Vamos supor que a e b são congruentes módulo m , isto é, $a \equiv b(\text{mod } m)$. Vamos ainda supor que $a \equiv x(\text{mod } m)$ e $b \equiv y(\text{mod } m)$, com x e y inteiros.

Assim,

$$a \equiv b(\text{mod } m) \Leftrightarrow m|(a - b),$$

$$a \equiv x(\text{mod } m) \Leftrightarrow m|(a - x),$$

$$b \equiv y(\text{mod } m) \Leftrightarrow m|(b - y).$$

Deste modo, temos que,

$$m|(a-b) - (a-x) + (b-y) \Leftrightarrow m|(x-y)$$

Mas, por definição, x e y são menores que m , logo, $x-y < m$.

De, $m|(x-y)$ e $x-y < m$, concluimos que $x-y = 0$, ou seja, $x = y$.

Assim, os restos das divisões de a e b por m são iguais.

Vamos agora supor que os restos das divisões de a e b por m são iguais, e provar que a e b são congruentes módulo m . Assim,

$$a \equiv x(\text{mod } m) \Leftrightarrow m|(a-x), b \equiv x(\text{mod } m) \Leftrightarrow m|(b-x)$$

Então, temos que,

$$m|(a-x) - (b-x) \Leftrightarrow m|(a-b) \Leftrightarrow a \equiv b(\text{mod } m)$$

□

Teorema 1.1.19. (*Propriedades das congruências*) Sejam a , b e $m > 1$ números inteiros. Então:

- I. $a \equiv a(\text{mod } m)$;
- II. Se $a \equiv b(\text{mod } m)$, então $b \equiv a(\text{mod } m)$;
- III. Se $a \equiv b(\text{mod } m)$ e se $b \equiv c(\text{mod } m)$, então $a \equiv c(\text{mod } m)$;
- IV. Se $a \equiv b(\text{mod } m)$ e se $c \equiv d(\text{mod } m)$, então $(a+c) \equiv (b+d)(\text{mod } m)$ e $ac \equiv bd(\text{mod } m)$;
- V. Se $a \equiv b(\text{mod } m)$, então $(a+c) \equiv (b+c)(\text{mod } m)$ e $ac \equiv bc(\text{mod } m)$;
- VI. Se $a \equiv b(\text{mod } m)$, então $a^n \equiv b^n(\text{mod } m)$, para todo o inteiro positivo n ;
- VII. Se $ab \equiv 0(\text{mod } m)$ e $\text{mdc}(b, m) = 1$, então $a \equiv 0(\text{mod } m)$.

Demonstração.

- I. É evidente, pois $m|0 \Rightarrow m|(a-a) \Rightarrow a \equiv a(\text{mod } m)$.

□

II. Se $a \equiv b \pmod{m}$, então $a - b = km$, com k inteiro.

Logo, $b - a = (-k)m$, ou seja, $b \equiv a \pmod{m}$. \square

III. Se $a \equiv b \pmod{m}$ e se $b \equiv c \pmod{m}$, então existem inteiros h e k tais que $a - b = hm$ e $b - c = km$. Logo, $(a - b) + (b - c) = hm + km$, ou seja, $a - c = (h + k)m$, o que significa que $a \equiv c \pmod{m}$. \square

IV. Se $a \equiv b \pmod{m}$ e se $c \equiv d \pmod{m}$, então existem inteiros h e k tais que $a - b = hm$ e $c - d = km$. Logo, $(a - b) + (c - d) = hm + km$, ou ainda $(a + c) - (b + d) = (h + k)m$. Assim, $a + c \equiv b + d \pmod{m}$.

Por outro lado, como $a - b = hm$, então $ac - bc = hcm$, e dado que $c = km + d$, temos que $ac - b(km + d) = hcm$, ou ainda, $ac - bd = hcm + bkm$, ou seja, $ac - bd = (hc + bk)m$. Assim, $ac \equiv bd \pmod{m}$. \square

V. Se $a \equiv b \pmod{m}$ e como $c \equiv c \pmod{m}$, temos que, pela proposição anterior, $a + c \equiv b + c \pmod{m}$ e $ac \equiv bc \pmod{m}$. \square

VI. Como $a \equiv b \pmod{m}$, então pela proposição 1.1.19(IV), temos que:

$$a.a \equiv b.b \pmod{m}$$

Isto é, $a^2 \equiv b^2 \pmod{m}$.

Pelo mesmo raciocínio, $a^2.a \equiv b^2.b \pmod{m}$, ou seja, $a^3 \equiv b^3 \pmod{m}$.

Utilizando indução matemática, podemos generalizar:

$$a^n \equiv b^n \pmod{m}$$

\square

VII. Se $\text{mdc}(b, m) = 1$, então existem x e y , inteiros, tais que $bx + my = 1$. Multiplicando ambos os membros por a , obtemos $abx + amy = a$, ou seja, $abx = a - amy$. Como $ab \equiv 0 \pmod{m}$, temos ainda que, $abx \equiv 0 \pmod{m}$, e por conseguinte, $a - amy \equiv 0 \pmod{m}$, ou seja, $a \equiv amy \pmod{m}$. Uma vez que $amy \equiv 0 \pmod{m}$, temos que $a \equiv 0 \pmod{m}$. \square

Proposição 1.1.20. *Sejam, a um inteiro positivo, x e y inteiros positivos, tais que $x + y = a$. Então,*

$$x^2 \equiv y^2 \pmod{a}$$

Demonstração.

Seja $a = x + y$, com a , x e y inteiros positivos. Então,

$$x^2 - y^2 = (x - y)(x + y) = a(x - y) \equiv 0 \pmod{a}$$

Deste modo,

$$x^2 - y^2 \equiv 0 \pmod{a} \Leftrightarrow x^2 \equiv y^2 \pmod{a}$$

□

Exemplo 1.1.21. *O número 40 pode ser escrito coma a soma de 10 com 30, ou ainda, 3 com 37.*

Ora, $10^2 \equiv 20 \pmod{40}$ e $30^2 \equiv 20 \pmod{40}$, ou seja, $10^2 \equiv 30^2 \pmod{40}$.

Do mesmo modo, $3^2 \equiv 9 \pmod{40}$ e $37^2 \equiv 9 \pmod{40}$, ou seja, $3^2 \equiv 37^2 \pmod{40}$.

Proposição 1.1.22. *Sejam, a um inteiro positivo, x e y inteiros positivos inferiores ao inteiro a e primos com a . Sejam ainda r_1 e r_2 tais que:*

$$xy \equiv r_1 \pmod{a} \text{ e } x^2 \equiv r_2 \pmod{a}$$

Então $x + y = a$ se e só se $r_1 + r_2 = a$.

Demonstração.

Como $x^2 \equiv r_2 \pmod{a}$ e $xy \equiv r_1 \pmod{a}$, temos que:

$$x^2 + xy \equiv (r_1 + r_2) \pmod{a}$$

Ou seja,

$$x(x + y) \equiv (r_1 + r_2) \pmod{a}$$

Ou ainda,

$$ax \equiv (r_1 + r_2) \pmod{a}$$

Como $ax \equiv 0(\text{mod } a)$, temos então que $r_1 + r_2 \equiv 0(\text{mod } a)$ e, dado que r_1 e r_2 são restos módulo a , e por conseguinte, inferiores ao inteiro a , concluímos que $r_1 + r_2 = a$. Vamos agora supor que $r_1 + r_2 = a$. Como,

$$xy \equiv r_1(\text{mod } a) \text{ e } x^2 \equiv r_2(\text{mod } a)$$

temos, como já vimos, que,

$$x(x + y) \equiv r_1 + r_2(\text{mod } a)$$

ou seja,

$$x(x + y) \equiv a(\text{mod } a)$$

ou ainda,

$$x(x + y) \equiv 0(\text{mod } a)$$

Como x e a são primos entre si, então pela proposição 1.1.19(VII), $(x + y) \equiv 0(\text{mod } a)$. Dado que x e y são menores que a , temos que $x + y = a$. \square

Exemplo 1.1.23. *O número 132 pode ser escrito como a soma de 49 com 83, ou ainda, 23 com 109.*

Ora, $49 \times 83 \equiv 107(\text{mod } 132)$ e $49^2 \equiv 25(\text{mod } 132)$.

Do mesmo modo, $23 \times 109 \equiv 131(\text{mod } 132)$ e $23^2 \equiv 1(\text{mod } 132)$.

Com efeito, $107 + 25 = 131 + 1 = 132$.

Proposição 1.1.24. *(Solução de uma congruência linear) Sejam a , b e n inteiros, com $n > 1$. A Congruência $ax \equiv b(\text{mod } n)$ tem solução em \mathbb{Z} se e só se b é múltiplo de $d = \text{mdc}(a, n)$.*

É evidente que se $d = 1$ então a congruência tem sempre solução. Nesse caso, utilizando o algoritmo de Euclides, é necessário determinar s e t , inteiros, tais que: $1 = sa + tn$. A solução da congruência é dada por:

$$x = bs + nk, \quad k \in \mathbb{Z}$$

Caso $d > 1$, e como b é múltiplo de d , de acordo com o teorema 1.1.7, a congruência $ax \equiv b(\text{mod } n)$ é equivalente a:

$$\frac{a}{d} x \equiv \frac{b}{d}(\text{mod } \frac{n}{d})$$

Exemplo 1.1.25. *Determinar a solução da congruência $315x \equiv 37 \pmod{44}$.*

Como o $\text{mdc}(315, 44) = 1$, é evidente que a congruência tem solução. O quadro associado é:

mdc	7	6	3	2
315	44	7	2	1
7	2	1	0	

Assim, temos que:

$$1 = 7 - 3 \times 2 = 7 - 3 \times (44 - 7 \times 6) = 19 \times 7 - 3 \times 44$$

Como $7 = 315 - 44 \times 7$,

$$1 = 19 \times (315 - 44 \times 7) - 3 \times 44 = 19 \times 315 - 136 \times 44$$

Deste modo $s = 19$ e $t = 136$. A solução da congruência dada é:

$$x = 37 \times 19 + 44k = 703 + 44k, k \in \mathbb{Z}$$

Teorema 1.1.26. *(Teorema do Resto Chinês) Sejam m_1, m_2, \dots, m_r inteiros positivos primos entre si dois a dois. Nestas condições, o sistema de congruências lineares:*

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \dots \\ x \equiv a_r \pmod{m_r} \end{cases}$$

tem uma única solução módulo $m = m_1 m_2 \dots m_r$. Essa solução é dada por:

$$X = a_1 M_1 x_1 + a_2 M_2 x_2 + \dots + a_r M_r x_r \pmod{m}$$

Em que,

$$M_1 = \frac{m}{m_1} ; M_2 = \frac{m}{m_2} ; \dots ; M_r = \frac{m}{m_r}$$

e x_1, x_2, \dots, x_r são as soluções das seguintes congruências, respetivamente

$$M_1 \equiv 1 \pmod{m_1} ; M_2 \equiv 1 \pmod{m_2} ; \dots ; M_r x \equiv r \pmod{m_r}$$

Exemplo 1.1.27. *Determinar a solução do seguinte sistema de congruências:*

$$\begin{cases} x \equiv 0(mod\ 2) \\ x \equiv 1(mod\ 3) \\ x \equiv 2(mod\ 5) \end{cases}$$

Como 2, 3 e 5 são primos entre si. Logo, pelo Teorema do Resto Chinês, o sistema tem uma única solução módulo $m = 2 \times 3 \times 5 = 30$. Assim, temos que:

$$M_1 = \frac{m}{2} = 15 ; M_2 = \frac{m}{3} = 10 \text{ e } M_3 = \frac{m}{5} = 6$$

As congruências:

$$15x \equiv 1(mod\ 2) ; 10x \equiv 1(mod\ 3) \text{ e } 6x \equiv 1(mod\ 5)$$

têm como soluções respectivas: $x_1 = 1$; $x_2 = 1$ e $x_3 = 1$. Obtemos assim,

$$X = 0 \times 15 \times 1 + 1 \times 10 \times 1 + 2 \times 6 \times 1 = 22$$

Então, $X \equiv 22(mod\ 30)$ é a única solução do sistema de congruências dado.

1.2 Números Primos

Os números primos e as suas propriedades foram pela primeira vez estudados pelos matemáticos da Antiga Grécia, como Pitágoras, Euclides e Eratóstenes. Mais tarde, a partir do século XVII, com Marin Mersenne e Pierre de Fermat, os números primos revelaram-se imprescindíveis à Matemática, pois representam a base, os alicerces, da Teoria dos Números.

Definição 1.2.1. (*Número primo*) Diz-se que um inteiro positivo é primo quando os seus únicos divisores são 1 e o próprio número, ou seja, um número primo é um inteiro positivo com apenas dois divisores distintos.

O número 1 já foi considerado primo, mas, entretanto, alterou-se a definição, considerando-se que os dois divisores teriam de ser distintos e, assim, como 1 tem apenas um divisor, não é, atualmente, considerado primo. Desta forma o menor primo é 2, já que tem apenas dois divisores, 1 e 2. Inclusivamente, 2 é o único par primo, já que os restantes pares são múltiplos de 2, e por conseguinte teriam pelo menos três divisores, 1, 2 e o próprio número. Os primos seguintes são 3, 5, 7 e 11. Um número inteiro positivo maior que 1 e que não é primo, diz-se composto. Assim, por exemplo, 4, 6, 8 e 9 dizem-se compostos.

Como veremos mais tarde, os números primos são a base da Teoria Aritmética dos Números, pois é a partir deles que se consegue obter, por multiplicação, todos os números compostos. Por exemplo, o número composto 10 pode obter-se por 2×5 .

É frequente chamar-se aos números primos os “geradores”. Essa denominação assenta no facto de que, com os números primos podemos gerar qualquer número composto. Como sabemos, um número pode ser decomposto num produto de fatores. Assim, por exemplo, o número 18 pode decompor-se em 2×9 , ou seja, com os números 2 e 9 podemos gerar o 18. Poderíamos também, gerar 18 com outros números:

$$18 = 2 \times 9 = 3 \times 6 = 2 \times 3 \times 3$$

Todos eles são fatores de 18. De todas as decomposições do número 18 a única que é formada por números primos é a última:

$$18 = 2 \times 3 \times 3$$

Crivo de Eratóstenes

Gerar números primos é, desde a antiga Grécia até aos dias de hoje, um tema verdadeiramente desafiante. Um dos primeiros métodos para gerar números primos deve a sua origem a Eratóstenes de Cirene (273 - 194 a.C), matemático, astrónomo e geógrafo grego. O método ficou conhecido como “crivo de Eratóstenes” e baseia-se no conceito de múltiplos de um número. É um algoritmo simples e permite encontrar todos os primos existentes até um determinado inteiro positivo. Em primeiro lugar constrói-se uma tabela com os números de 1 até ao número escolhido, por exemplo 100, e de seguida segue-se os seguintes passos:

1. Cortar o número 1, porque não é considerado primo;
2. Selecionar o menor número ainda não cortado;
3. Cortar todos os múltiplos desse número;
4. Se todos os números da lista já tiverem sido selecionados ou cortados, então o processo terminou. Caso contrário é necessário regressar ao passo 2.

1	(2)	(3)	4	(5)	6	(7)	8	9	10
(11)	12	(13)	14	15	16	(17)	18	(19)	20
21	22	(23)	24	25	26	27	28	(29)	30
(31)	32	33	34	35	36	(37)	38	39	40
(41)	42	(43)	44	45	46	(47)	48	49	50
51	52	(53)	54	55	56	57	58	(59)	60
(61)	62	63	64	65	66	(67)	68	69	70
(71)	72	(73)	74	75	76	77	78	(79)	80
81	82	(83)	84	85	86	87	88	(89)	90
91	92	93	94	95	96	(97)	98	99	100

Os números que ficaram por eliminar são primos, pois não são múltiplos de nenhum número, ou seja, apenas são divisíveis por 1 e por si próprios. Assim, os primos até 100 são:

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	

Também se verifica que a crivagem termina quando se cortam os múltiplos de 7, o maior primo inferior a 10, raiz quadrada de 100.

Como veremos adiante, para encontrar todos os números primos menores ou iguais a um número n dado, basta crivar os números inferiores ou iguais a raiz de n . Este método continua ainda hoje a ser utilizado para encontrar números primos relativamente pequenos, inferiores a 10^9 .

Como vimos, com o crivo de Eratóstenes podemos obter todos os números primos até um determinado número natural. Mas surge logo uma pergunta: será que o número de primos é infinito? Ou por outro lado, a partir de um determinado número natural, os números são todos compostos? Será que existe alguma fórmula aritmética que gere os primos? Para tentar responder a estas perguntas, é preciso, em primeiro lugar, dispor de uma lista de primos e, em seguida, examinar se existe relação entre os números primos ou alguma regra que permita perceber quando surgirá o próximo. Observemos a lista dos números primos até 500:

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167
173	179	181	191	193	197	199	211	223	227	229	233	239
241	251	257	263	269	271	277	281	283	293	307	311	313
317	331	337	347	349	353	359	367	373	379	383	389	397
401	409	419	421	431	433	439	443	449	457	461	463	467
479	487	491	499									

De acordo com a lista, entre os números 1 e 500 existem 95 primos. Um olhar mais atento permite constatar que os números primos parecem ser imprevisíveis e que dificilmente existirá alguma regularidade. Observando a lista de primos em anexo, verifica-se que entre 500 e 1000 existem apenas 73, o que leva a crer que, de facto, os números primos não seguem qualquer lei ou regra, mas por outro lado, nos coloca outra questão pertinente: a quantidade de números primos em intervalos com a mesma amplitude diminui?

Recorrendo novamente à lista de primos, verifica-se que entre 1000 e 1500 existem 71 primos, entre 1500 e 2000, 64 primos, entre 2000 e 2500, também 64 primos, entre 2500 e 3000, 63 primos, e entre 3000 e 3500, apenas 59 primos. Assim, aparentemente o número de primos parece mesmo diminuir por intervalo de números naturais, o que também é suportado pelo facto de, por exemplo, entre 1 e 100 existirem 25 números primos, enquanto que entre 100 e 200 existem 21, e entre 200 e 300, apenas 16. Ao verificar essa característica dos números primos, surgem ainda outras questões:

Será que, como o número de primos diminui por intervalo, eventualmente se encontrará um intervalo sem qualquer primo?

Ora, a resposta a esta pergunta é obviamente sim, pois, por exemplo, entre 55000 e 55010 existem 2 primos, mas entre 55010 e 55020 não se encontra nenhum primo. Também se constata que quanto menor a amplitude do intervalo utilizado, mais rapidamente se encontra um intervalo que não contenha números primos.

Pelo facto de encontrarmos um intervalo sem números primos, conclui-se que nos intervalos seguintes não surgirá algum número primo?

Ora, a resposta é neste caso negativa, já que utilizando os intervalos anteriores, se verifica que entre 55010 e 55020 não existe qualquer primo, mas que entre 55020 e 55030 existe 1 primo. Mais ainda, entre 55030 e 55040, volta a não existir qualquer primo, para no intervalo seguinte, 55040 a 55050, se encontrar um primo, 55049. Portanto, pode-se conjecturar que os números primos são cada mais raros, visto que, à medida que utilizamos intervalos de números inteiros cada vez maiores, verifica-se que o número de primos presentes nesse mesmo intervalo tem tendência para decrescer, ou seja, que entre dois primos arbitrariamente grandes, pode existir um salto grande, isto é, existir uma grande quantidade de números compostos entre os dois primos.

Se se verificam grandes lacunas entre dois primos arbitrariamente grandes e se essa lacuna tem tendência a aumentar, será que a partir de certo número natural já não se encontrarão números primos?

Euclides conjecturou e provou, como veremos mais tarde, que o número de primos é infinito.

Estas perguntas e respostas levam a acreditar que a distribuição dos primos é bastante irregular, ou por outras palavras, que os números primos não têm um sentido de ritmo, são imprevisíveis e arbitrários, não se regem por qualquer lei, nem permitem que uma fórmula aritmética simples os gere a todos.

Gerar números primos

A irregularidade e arbitrariedade dos números primos não permite a existência de uma fórmula que gere todos os números primos, no entanto, existem algumas fórmulas, ou leis, que permitem gerar números primos:

1. $x^2 + x + 41$ é primo para x inteiro, com $0 \leq x \leq 39$; (Polinómio de Euler)
2. $x^2 - 79x + 1601$ é primo para x inteiro, com $0 \leq x \leq 79$;
3. $2x^2 + 29$ é primo para x inteiro, com $0 \leq x \leq 28$;
4. $x^2 + x + 17$ é primo para x inteiro, com $0 \leq x \leq 16$;
5. $3x^2 + 3x + 23$ é primo para x inteiro, com $0 \leq x \leq 21$.

Definição 1.2.2. (*Números de Mersenne*) Chama-se número de Mersenne a todo o inteiro positivo da forma: $M_n = 2^n - 1$.

Se M_n é primo, diz-se que é um primo de Mersenne.

Sabe-se que M_n é primo para $n = 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107$ e 127. Nas últimas décadas, recorrendo a computadores, têm sido descobertos mais primos de Mersenne, sendo que, atualmente são conhecidos 48 primos de Mersenne. O maior primo de Mersenne conhecido tem mais de 17 milhões de dígitos.

Definição 1.2.3. (*Números de Fermat*) Chama-se número de Fermat a todo o inteiro positivo da forma: $F_n = 2^{2^n} + 1$, com $n \geq 0$.

Se F_n é primo, diz-se que é um primo de Fermat.

Atualmente, conhecem-se apenas cinco primos de Fermat:

$$F_0 = 3, F_1 = 5, F_2 = 17, F_3 = 257; F_4 = 65537$$

Não se sabe, até à data, se o número de primos de Fermat é finito.

1.3 Teoremas Subjacentes aos Números Primos

Teorema 1.3.1. *Se um primo p não divide um inteiro a , então a e p são primos entre si.*

Demonstração.

Seja d o $\text{mdc}(a, p)$. Então $d|a$ e $d|p$. Como $d|p$ e p é primo, temos que $d = 1$ ou $d = p$. Como a segunda igualdade é falsa, porque p não divide a , segue-se que $d = 1$. Deste modo, a e p são primos entre si. \square

Deste teorema resultam resultados importantes:

Corolário 1.3.2. *Se p é primo tal que $p|bc$, então $p|b$ ou $p|c$;*

Corolário 1.3.3. *Se p é um primo tal que $p|b_1b_2 \dots b_n$, então existe um índice k , com $1 \leq k \leq n$, tal que $p|b_k$;*

Corolário 1.3.4. *Se os inteiros p, q_1, q_2, \dots, q_n são todos primos e se $p|q_1q_2 \dots q_n$, então existe um índice k , com $1 \leq k \leq n$, tal que $p = q_k$.*

Demonstração.

1. Vamos supor, sem perda de generalidade, que $p \nmid b$. Pelo teorema 1.3.1, p e b são primos entre si. Assim, pelo teorema 1.1.9,

$$\text{mdc}(p, b) = 1 \Rightarrow 1 = px + by, \text{ para algum } x \text{ e } y \text{ inteiros.}$$

Multiplicando ambos os membros por c , obtemos $c = cpx + cby$. Como,

$$p|bc \Rightarrow bc = pk, \text{ para algum } k \text{ inteiro}$$

Temos que,

$$c = cpx + pky \Rightarrow c = p(cx + ky) \Rightarrow p|c$$

\square

2. Como $p|b_1b_2\ldots b_n$, então pelo corolário 1.3.1 e recorrendo ao método de indução matemática, $p|b_1$ ou $p|b_2$ ou \ldots ou $p|b_n$, e assim, existe um índice k tal que $p|b_k$.
3. Se $p|q_1q_2\ldots q_n$, pelo corolário 1.3.2, existe k inteiro tal que $p|q_k$. Desta forma, existe também m inteiro tal que $q_k = pm$.
Como q_k é primo, temos que:

$$q_k = pm \Rightarrow q_k = p \text{ ou } q_k = m$$

Se $q_k = m$, então $p = 1$, o que é impossível, logo conclui-se que $p = q_k$. □

Teorema 1.3.5. *Todo o número composto possui um divisor primo.*

Demonstração.

Seja a um número composto e $p > 1$ o menor dos divisores positivos de a .

Vamos supor que p é composto. Então p admitiria pelo menos um divisor d tal que $1 \leq d \leq p$, e por conseguinte $d|p$ e $p|a$, o que implica que $d|a$, isto é, p não seria o menor divisor não trivial de a , logo p é primo. □

Teorema 1.3.6. *(Teorema fundamental da Aritmética) Todo o inteiro positivo $n > 1$ pode ser decomposto num produto de fatores primos:*

$$n = p_1^{h_1} p_2^{h_2} \ldots p_s^{h_s} \quad (1)$$

onde p_1, p_2, \ldots, p_s são primos distintos e h_1, h_2, \ldots, h_s são inteiros positivos.

A fatorização (1) é única a menos da ordem dos fatores.

Demonstração.

Vamos provar separadamente a existência e a unicidade da fatorização. Em primeiro lugar vamos provar a existência por indução.

É evidente que a igualdade se verifica para $n = 2$. Suponhamos que para todo o $2 < k < n$ se verifica a igualdade (1) e vamos provar que também é satisfeita para n . Se n é um número primo, é evidente que a igualdade (1) é verificada. Se n é composto,

então existem inteiros $2 < a < n$ e $2 < b < n$ tais que $n = ab$.

Por hipótese de indução existem q_i, p_i , números primos distintos e h_i, m_i inteiros positivos, tal que:

$$a = q_1^{h_1} q_2^{h_2} \dots q_s^{h_s}, b = q_1^{m_1} q_2^{m_2} \dots q_k^{m_k}$$

Então,

$$n = q_1^{h_1} q_2^{h_2} \dots q_s^{h_s} p_1^{j_1} p_2^{m_2} \dots p_s^{m_s}$$

Agrupando os primos iguais, obtemos a fatorização de n na forma pretendida.

Relativamente à unicidade da fatorização, vamos também efetuar a prova por indução no número de elementos da fatorização de n .

Dada a decomposição de n :

$$n = p_1^{h_1} p_2^{h_2} \dots p_s^{h_s} (1)$$

O número de fatores da decomposição de n é dado por $m = h_1 + h_2 + \dots + h_s$.

Se $m = 1$, n é primo e possui uma única decomposição.

Vamos supor agora que a decomposição de um inteiro positivo em $m - 1$ fatores é única. Seja n um inteiro positivo que possui duas fatorizações em m fatores primos.

Então,

$$n = p_1^{h_1} p_2^{h_2} \dots p_s^{h_s} = q_1^{k_1} q_2^{k_2} \dots q_t^{k_t}$$

com, $h_1 + h_2 + \dots + h_s = m$ e $k_1 + k_2 + \dots + k_t = m$.

Como $p_1 | n$, então $p_1 | q_1 q_2 \dots q_t$. Assim, pelo corolário 1.3.4, existe q_i primo, com $1 < i \leq t$ tal que $p_1 = q_i$.

Desta forma, assumindo, sem perda de generalidade, $p_1 = p_1$,

$$p_1^{h_1-1} p_2^{h_2} \dots p_s^{h_s} = q_1^{k_1-1} q_2^{k_2} \dots q_t^{k_t},$$

ou seja, temos duas decomposições de um número inteiro positivo com $m - 1$ fatores, logo, pela hipótese de indução, temos que a fatorização de n em primos é única, a menos da ordem dos fatores. \square

Teorema 1.3.7. (*Teorema de Euclides*) *Existe um número infinito de primos.*

Demonstração.

Suponhamos que existe um número finito de número primos, $p_1 < p_2 < \dots < p_n$ e consideremos o inteiro positivo,

$$n = p_1 p_2 \dots p_n + 1$$

Pelo Teorema Fundamental da Aritmética, n tem pelo menos um divisor primo p . Pela hipótese, p_1, p_2, \dots, p_n são os únicos primos, de modo que p deve ser, necessariamente, um desses primos. Assim sendo, temos que:

$$p|n \text{ e } p|p_1 p_2 \dots p_n$$

Assim, pela proposição 1.1.3(VI),

$$p|(n - p_1 p_2 \dots p_n)$$

ou seja, $p|1$, o que é um absurdo, porque $p > 1$.

Desta forma, concluímos que o número de primos é infinito. \square

Observação 1.3.8. (*Demonstração alternativa*)

Em 1878, o matemático alemão Kumer (1810-1893) apresentou outra demonstração, ainda mais simples e elegante, para o teorema de Euclides, supondo, tal como Euclides, que o número de primos é finito, e considerando P o produto de todos os primos conhecidos. Assim, $P = 2 \times 3 \times 5 \times 7 \times \dots \times p$, em que p é o maior primo conhecido. Ora, $P - 1$ ou é primo ou é um produto de primos, logo, existe um primo q que divide $P - 1$. Dado que q também divide P , então divide a sua diferença, $P - (P - 1) = 1$, logo q divide 1, o que é absurdo. \square

Observação 1.3.9. *Se $n = p_1 p_2 \dots p_n + 1$, verifica-se que n é um novo número primo, ou um número composto cuja decomposição contém números primos superiores a p_n .*

Por exemplo:

$$2 \times 3 \times 5 \times 7 \times 11 + 1 = 2311$$

$$2 \times 3 \times 5 \times 7 \times 11 \times 13 + 1 = 30031$$

Verifica-se que 2311 é primo, enquanto 30031 é composto, cujos fatores primos são 59 e 509.

Teorema 1.3.10. (*Critério primal*) *Se um número inteiro $n > 1$ não possuir nenhum divisor primo menor ou igual a \sqrt{n} , então n é primo.*

Demonstração.

Pelo método da contra-recíproca, é necessário provar que se um inteiro positivo $a > 1$ é composto, então possui um divisor primo $p \leq \sqrt{a}$.

Com efeito, se o inteiro positivo a é composto, então $a = bc$, com $1 < b < a$ e $1 < c < a$.

Sem perda de generalidade, suponhamos que $b < c$. Então,

$$b^2 \leq bc = a \Rightarrow b \leq \sqrt{a}$$

Dado que $b > 1$, o Teorema fundamental da Aritmética assegura que b tem pelo menos um divisor primo p , com $p \leq b \leq \sqrt{a}$.

Como $p|b$ e $b|a$, segue-se, pela proposição 1.1.3(IV), que $p|a$, isto é, o inteiro primo $p \leq \sqrt{a}$ é um divisor de a .

Fica, deste modo, provado que dado a inteiro, se não existir p primo, com $p \leq \sqrt{a}$, em que p divida a , então a é primo. \square

Observação 1.3.11.

Para verificar se um número é primo é necessário verificar, como sabemos, se esse número é divisível por algum inteiro menor que si próprio, exceto a unidade. Pelo teorema anterior, é necessário e suficiente verificar se esse número tem algum divisor primo inferior ou igual à sua raiz quadrada, o que, evidentemente, simplifica o processo de verificar a primalidade de um dado número.

Por exemplo, para determinar se 101 é primo, temos de verificar se 101 tem algum divisor primo inferior a $\sqrt{101} \approx 10,05$, ou seja, se 2, 3, 5 ou 7 dividem 101.

Teorema 1.3.12. (*Teorema de Fermat*) Se p é primo e não divide o inteiro a , então:

$$a^{p-1} \equiv 1(\text{mod } p)$$

Corolário 1.3.13. (*Pequeno Teorema de Fermat*) Se p é um primo, então qualquer que seja o inteiro a , temos que:

$$a^p \equiv a(\text{mod } p)$$

Demonstração.

Vamos utilizar indução em a , pois verificando a equivalência para a natural, facilmente se estende aos inteiros.

Para $a = 1$, é trivial, pois, $1^p \equiv 1(\text{mod } p)$.

Supondo agora que $a \geq 1$ e $a^p \equiv a(\text{mod } p)$. Pelo Binómio de Newton, temos que:

$$(a+1)^p = a^p + \binom{p}{p-1} a^{p-1} + \dots + \binom{p}{i} a^i + \dots + \binom{p}{1} a + 1$$

Assim,

$$(a+1)^p \equiv a^p + \binom{p}{p-1} a^{p-1} + \dots + \binom{p}{i} a^i + \dots + \binom{p}{1} a + 1(\text{mod } p)$$

E dado que:

$$\binom{p}{p-1} \equiv 0(\text{mod } p); \dots \binom{p}{i} \equiv 0(\text{mod } p); \dots \binom{p}{1} \equiv 0(\text{mod } p);$$

Temos que,

$$(a+1)^p \equiv a^p + 1(\text{mod } p) \Leftrightarrow (a+1)^p \equiv a + 1(\text{mod } p)$$

□

Este corolário é, como veremos mais tarde, um critério para reconhecer se um inteiro é primo. De qualquer forma, para inteiros grandes este critério é impraticável.

Teorema 1.3.14. (*Teorema de Wilson*) Um número natural n maior que 1 é primo se e só se $(n-1)! \equiv -1(\text{mod } n)$.

Definição 1.3.15. (*Função de Euler*) Chama-se função de Euler a $\phi(n)$ que a cada inteiro positivo n , faz corresponder o número de inteiros positivos não superiores a n e que são primos com n .

Doutra forma, $\phi(n)$ é o número de elementos do conjunto:

$$\{x \in N : 1 \leq x \leq n \text{ e } \text{mdc}(x, n) = 1\}$$

Assim, por exemplo, $\phi(1) = 1$, porque $\text{mdc}(1, 1) = 1$. Para $n > 1$, verifica-se que $\text{mdc}(n, n) \neq 1$, logo, $\phi(n)$ representa o número de inteiros positivos menores que n que são primos com n . Outros exemplos:

$\phi(20) = 8$, pois os inteiros positivos menores que 20 e primos com 20 são 1, 3, 7, 9, 11, 13, 17 e 19.

$\phi(13) = 12$, pois os inteiros positivos menores que 13 e primos com 13 são 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 e 12.

Com este último exemplo, observa-se que, como 13 é primo, $\phi(13)$ é dado por $13-1$, pois 13 é primo com todos os inteiros positivos inferiores a 13.

Teorema 1.3.16. *Se o inteiro $n > 1$, então $\phi(n) = n - 1$ se e só se n é primo.*

Demonstração.

Se n é primo, então n é primo com qualquer um dos $n - 1$ inteiros positivos menores que n , o que permite concluir que $\phi(n) = n - 1$.

Reciprocamente, vamos supor que n é composto. Sendo n composto, teria pelo menos um divisor d , com $1 < d < n$, em que $\text{mdc}(d, n) = d > 1$, de modo que $\phi(n) \leq n - 2$, o que contradiz a hipótese inicial. Assim conclui-se que n é primo. \square

Teorema 1.3.17. (*Teorema de Gauss*) *Para todo o inteiro positivo n tem-se que:*

$$\sum_{d|n} \phi(d) = n$$

Teorema 1.3.18. (*Teorema de Euler*) *Se a e n são inteiros, com $n \geq 1$ e se o $\text{mdc}(a, n) = 1$, então:*

$$a^{\phi(n)} \equiv 1(\text{mod } n)$$

Note-se que, se n for primo e $\text{mdc}(a, n) = 1$, temos que,

$$a^{n-1} \equiv 1 \pmod{n}$$

que é o teorema de Fermat. Por outras palavras, o teorema de Euler é uma generalização do Teorema de Fermat.

Teorema 1.3.19. *(Teorema de Euler) Seja a um número inteiro, e p e q , primos distintos. Se $\text{mdc}(a, p) = \text{mdc}(a, q) = 1$, então*

$$a^{(p-1)(q-1)} \equiv 1 \pmod{pq}$$

Apesar da irregularidade na distribuição dos números primos, desde cedo os matemáticos tentaram abordar o problema através da Teoria Analítica dos Números.

Definição 1.3.20. *Seja x um número inteiro positivo. A função $\pi(x)$ representa o número de primos inferiores ou iguais a x .*

Por exemplo:

$$\pi(1) = 0 \text{ (não existem nenhum primo inferior ou igual a 1)}$$

$$\pi(2) = 1 \text{ (2 é o único primo inferior ou igual a 2)}$$

$$\pi(6) = 3 \text{ (2, 3 e 5 são os únicos primos inferiores ou iguais a 3)}$$

$$\pi(19) = 8 \text{ (existem 8 primos inferiores ou iguais a 19)}$$

$$\pi(100) = 25 \text{ (existem 25 primos inferiores ou iguais a 100)}$$

Teorema 1.3.21. *A função $\pi(x)$ cresce mais lentamente que qualquer função linear, ou seja,*

$$\lim_{x \rightarrow \infty} \frac{\pi(x)}{x} = 0$$

Teorema 1.3.22. *(Teorema dos números primos)*

$$\lim_{x \rightarrow \infty} \frac{\pi(x)}{\frac{x}{\log(x)}} = 1$$

As demonstrações dos teoremas anteriores, bem como de mais resultados obtidos pela Teoria Analítica dos Números, com vista a melhorar conhecimentos sobre o tema, podem ser consultados em [Filho (1981)] e [Ribenoim (1995)].

1.4 Testes de Primalidade e Aplicações dos números primos

O crivo de Eratóstenes permite-nos gerar números primos. No entanto, esse método é moroso se pretendermos verificar se um determinado número é primo, pois teríamos de construir uma tabela com os números naturais até esse número e posteriormente aplicar o método descrito, até verificarmos se o número é cortado, o que significaria que era composto, ou caso contrário, que levar-nos-ia a concluir que o número era primo.

Uma forma de aplicar o método mais rapidamente é verificar apenas se esse número é divisível por todos os números primos até à sua raiz quadrada, já que se um número é composto, ao decompor-se num produto, em que, se um dos fatores é maior que a raiz quadrada, os outros fatores têm de ser inferiores à raiz, pois o produto de pelo menos dois fatores maior que a raiz quadrada, tem um resultado maior que o número a verificar. Por exemplo, se pretendermos verificar se 227 é um número primo, teríamos de verificar apenas se 227 é divisível por todos os primos até $\sqrt{227} \approx 15.1$, ou seja, por 2, 3, 5, 7, 11 e 13.

Os testes de primalidade são algoritmos que permitem verificar se um dado número é primo. Existem vários tipos de teste de primalidade cujos algoritmos se baseiam nas propriedades dos números primos, que podem ser classificados de acordo com vários critérios, sendo os mais relevantes: o tipo de número, a justificação do teste e a certeza do teste.

Classificação dos testes de primalidade:

Tipo de número	Testes para números de forma particular
	Testes para números genéricos
Justificação do teste	Testes justificados por teoremas
	Testes justificados por conjecturas
Certeza do teste	Testes determinísticos
	Testes probabilísticos (ou de Monte Carlo)

1.4.1 Testes determinísticos versus testes probabilísticos

Os testes determinísticos são testes que asseguram que a conclusão obtida está correta, ou seja, se um teste determinístico afirmar que um determinado número é primo (ou composto), temos a garantia que de facto esse número é primo (ou composto). Estes testes, por um lado, produzem uma resposta fiável, mas por outro, podem ser bastante lentos, sendo que, regra geral, são mais lentos que qualquer teste probabilístico.

Crivo de Eratóstenes e Método da Força Bruta

O crivo de Eratóstenes que permite, como já vimos numa secção anterior, determinar todos os primos até um determinado número inteiro seleccionado previamente, também pode ser utilizado para verificar se um número é primo. No método é criada uma lista de números inteiros até o número que se pretende testar, e depois vão sendo removidos os múltiplos de números primos. Quando forem removidos os múltiplos de todos os primos inferiores ou iguais à raiz quadrada do número que se pretende testar, é suficiente verificar se esse número foi ou não removido durante o processo. Desta forma, se o número tiver sido removido, concluimos que estamos perante um número composto, já que é múltiplo de algum número primo, caso contrário, estamos perante um número primo.

Este algoritmo, embora lento mas eficiente, funciona para qualquer número e é justificado pela própria definição de número primo. No entanto, para testar números muito grandes, é completamente inviável.

O Método da Força Bruta é um dos testes de primalidade mais simples que se conhecem e baseia-se no conceito de divisibilidade. O algoritmo consiste em, dado um número inteiro que se pretende testar, verificar se esse número é divisível por qualquer número inteiro positivo inferior a si próprio. Obviamente que é possível melhorar o algoritmo, sendo que, podemos apenas verificar se o número é divisível por qualquer primo inferior ou igual à sua raiz quadrada. Este algoritmo apresenta as mesmas características que o Crivo de Eratóstenes, apenas difere na forma como aplicamos o algoritmo.

Os testes anteriores, considerados determinísticos, têm na sua gênese o conceito de número primo. Por um lado, o Crivo de Eratóstenes explora o conceito de número primo como número inteiro que não é múltiplo de nenhum primo, enquanto o Método da Força Bruta explora o conceito como um número inteiro que não possui divisores inferiores a si próprio, exceto, a unidade.

Teste de Primalidade de Lucas-Lehmer

O teste de primalidade de Lucas-Lehmer é baseado no teorema de Fermat e afirma que um inteiro positivo n maior que 2, e primo com um inteiro positivo a , é primo se e só se:

$$a^{n-1} \equiv 1 \pmod{n} \text{ e } a^{\frac{n-1}{q}} \pmod{n} \neq 1$$

para todo o q , fator primo de $n - 1$.

Exemplo 1.4.1. *Consideremos os números $n = 101$ e $a = 2$, primos entre si.*

Os fatores primos de $n - 1 = 100$ são 2 e 5. Como,

$$2^{100} \equiv 1 \pmod{101} ; 2^{\frac{100}{2}} \pmod{101} \neq 1 \text{ e } 2^{\frac{100}{5}} \pmod{101} \neq 1$$

concluimos que 101 é um número primo.

Exemplo 1.4.2. *Consideremos os números $n = 120$ e $a = 7$, primos entre si.*

Os fatores primos de $n - 1 = 119$ são 7 e 17. Como,

$$7^{119} \equiv 103 \pmod{120}$$

concluimos que 120 não é um número primo.

Testes probabilísticos

Os testes probabilísticos, regra geral, nunca afirmam que um número primo é composto, mas podem concluir que um número composto é primo, ou seja, num teste probabilístico existe a possibilidade de ocorrer um erro, pode ser reduzida ao se repetir o teste várias vezes, alterando sempre os parâmetros iniciais. De qualquer forma, este tipo de teste é muito utilizado, pois são testes cujo algoritmo permite obter uma resposta rápida sobre a primalidade de um dado número, sendo que,

por vezes alguns números podem ser declarados como primos prováveis, quando na realidade são compostos. Assim, quando comparados com os testes determinísticos, os testes probabilísticos são incomparavelmente mais rápidos, mas menos fiáveis.

Teste de Primalidade de Fermat

O teste de primalidade de Fermat é baseado no pequeno teorema de Fermat, já apresentado anteriormente. O teorema afirma que se p é primo e não divide o inteiro a , então $a^{p-1} \equiv 1 \pmod{p}$. Assim, o teste consiste em escolher valores aleatórios para a e verificar se a congruência permanece verdadeira. Se a congruência permanecer verdadeira, então p é provavelmente primo, caso contrário, estamos perante um número composto. O teste de Fermat não é utilizado frequentemente, já que existem números compostos que independentemente dos valores de a escolhidos, o teste afirma serem números primos. Este tipo de número é chamado número de Carmichael.

Teste de Primalidade de Miller-Rabin

O teste de Miller-Rabin é composto por duas condições a verificar, sendo que, se pelo menos uma dessas condições for verdadeira, estamos perante um primo provável, e, caso ambas não se verifiquem, perante um número composto. Dado um inteiro positivo n , as condições são:

$$a^d \equiv 1 \pmod{n} \text{ ou } a^{2^r d} \equiv -1 \pmod{n} \text{ para algum } r \in \{0, 1, \dots, s-1\}$$

para a um inteiro positivo primo com n , $1 < a < n$, em que a decomposição de $n-1$ é da forma $2^s \times d$.

Exemplo 1.4.3. *Consideremos os números $n = 181$ e $a = 3$, primos entre si.*

A decomposição de $n-1 = 180$ é dada por $2^2 \times 45$. Como,

$$3^4 \times 5 \equiv 1 \pmod{181}$$

concluimos que 181 é um primo provável.

Exemplo 1.4.4. *Consideremos os números $n = 221$ e $a = 3$, primos entre si.*

A decomposição de $n-1 = 220$ é dada por $2^2 \times 55$. Como,

$$3^5 \times 5 \equiv 198 \pmod{221} \text{ e } 3^1 \times 10 \equiv 87 \pmod{221}$$

nenhuma das condições se verifica, logo 221 não é um número primo.

1.4.2 Aplicações dos números primos

Como já pudemos perceber, a temática dos números primos é um assunto que tem permitido melhorar os conhecimentos nas diferentes áreas da Matemática, com aplicações nomeadamente na Álgebra, mas a questão que podemos colocar é: será que os números primos têm alguma utilidade de ordem prática? Os testes de primalidade, do ponto de vista analítico, permitem encontrar mais e mais primos, o que permite aumentar o conhecimento que temos relativamente à forma como estes se distribuem, mesmo sem uma fórmula, ou algoritmo, que consiga gerar todos os números primos, mas, por outro lado, os primos serviram de inspiração para a criação de algoritmos que permitem criar sistemas seguros e fiáveis, nomeadamente, os sistemas de transmissão de dados e os sistemas de identificação modulares.

Sistema RSA

A Criptografia é a arte de codificar e decodificar uma mensagem do modo a que apenas o destinatário pretendido a possa ler. São conhecidas muitas cifras, métodos de codificar e decodificar mensagens, criadas ao longo dos séculos, no entanto, foi apenas em 1978 que surgiu um método em que decodificar pode ser apenas feito pelo destinatário. Esse método, criado por Rivest, Shamir e Adleman, e denominado por sistema RSA, é considerado um método de chave pública. O sistema RSA é fundamentado na teoria dos números primos e é mesmo uma das mais importantes aplicações dos números primos. Este sistema baseia-se no teorema 1.3.19 (Euler).

O Recetor cria um chave privada (que apenas ele conhece) e uma chave pública, conhecida por todos, do seguinte modo:

1. Selecionar dois números primos, p e q (grandes);
2. Determinar $n = pq$;
3. Escolher $e \in [0, n[$, de forma a que $\text{mdc}(e, (p-1)(q-1)) = 1$, criando assim a chave pública (n, e) ;
4. Determinar $d \in [0, n[$, de forma a que $de \equiv 1 \pmod{(p-1)(q-1)}$, criando a chave privada (n, d) .

Exemplo 1.4.5. (*criação de chave pública e privada*)

1. $p = 53$ e $q = 59$

2. $n = 53 \times 59 = 3127$

3. $(p-1).(q-1) = 52 \times 58 = 3016$; $e = 11$; $\text{mdc}(11, 3016) = 1$;

Chave pública: $(3127, 11)$

4. $11d \equiv 1 \pmod{3016}$

Pelo Algoritmo de Euclides:

$$1 = 11 - 5 \times 2 = 11 - 5(3016 - 274 \times 11) = -5 \times 3016 + 1371 \times 11$$

Ou seja,

$$1 = -5 \times 3016 + 1371 \times 11$$

Deste modo, $d = 1371 + 3127k$, $k \in \mathbb{Z}$.

Para $k = 0$, *temos* $d = 1371$. *Logo, a chave privada é* $(3127, 1371)$.

Vamos agora simular o envio da seguinte mensagem: “conjetura de goldbach”. Para esse efeito vamos considerar a seguinte tabela de correspondência:

letra	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>	<i>i</i>	<i>j</i>	<i>l</i>	<i>m</i>
número	00	01	02	03	04	05	06	07	08	09	10	11
letra	<i>n</i>	<i>o</i>	<i>p</i>	<i>q</i>	<i>r</i>	<i>s</i>	<i>t</i>	<i>u</i>	<i>v</i>	<i>x</i>	<i>z</i>	
número	12	13	14	15	16	17	18	19	20	21	22	

Utilizando a tabela anterior obtemos a seguinte correspondência:

c o n j e t u r a d e g o l d b a c h x
 02 13 12 09 04 18 19 16 00 03 04 06 13 10 03 01 00 02 07 21

Agrupando em blocos com 4 algarismos:

0213 1209 0418 1916 0003 0406 1310 0301 0002 0721

A letra *x* foi adicionada para que o último bloco também tivesse 4 algarismos.

Para codificar a mensagem, utiliza-se a chave pública do destinatário da seguinte forma:

$$0213^{11}(\text{mod } 3127) = 1326$$

$$1209^{11}(\text{mod } 3127) = 1069$$

$$0418^{11}(\text{mod } 3127) = 1141$$

$$1916^{11}(\text{mod } 3127) = 2681$$

$$0003^{11}(\text{mod } 3127) = 2035$$

$$0406^{11}(\text{mod } 3127) = 2443$$

$$1310^{11}(\text{mod } 3127) = 0534$$

$$0301^{11}(\text{mod } 3127) = 2162$$

$$0002^{11}(\text{mod } 3127) = 2048$$

$$0721^{11}(\text{mod } 3127) = 0114$$

Assim, a mensagem codificada a enviar ao destinatário é:

1326 1069 1141 2681 2035 2443 0534 2162 2048 0114

Para o destinatário (recetor) decodificar a mensagem, utiliza a sua chave privada da seguinte forma:

$$1316^{1371}(\text{mod } 3127) = 0213$$

$$1069^{1371}(\text{mod } 3127) = 1209$$

$$1141^{1371}(\text{mod } 3127) = 0418$$

$$2681^{1371}(\text{mod } 3127) = 1916$$

$$2035^{1371}(\text{mod } 3127) = 0003$$

$$2443^{1371}(\text{mod } 3127) = 0406$$

$$0534^{1371}(\text{mod } 3127) = 1310$$

$$2162^{1371}(\text{mod } 3127) = 0301$$

$$2048^{1371}(\text{mod } 3127) = 0002$$

$$0114^{1371}(\text{mod } 3127) = 0721$$

Desta forma, o destinatário obtém a seguinte sequência de blocos:

0213 1209 0418 1916 0003 0406 1310 0301 0002 0721

Agrupando em blocos de 2 algarismos e efetuando a correspondência com as letras do alfabeto:

c o n j e t u r a d e g o l d b a c h x

A segurança do RSA baseia-se no facto de não existir nenhum algoritmo eficiente para fatorizar n , pois mesmo sendo n e e públicos, é necessário determinar p e q , ambos primos muito grandes, que não podem ser descobertos em tempo útil pelos métodos conhecidos atualmente.

Sistemas de Identificação Modulares

Os sistemas de identificação modulares são sistemas que utilizam números de identificação que permitem identificar um livro, pessoa, artigo ou produto comercializável. Estes números permitem criar uma identificação única e possibilitam ainda, dependendo da forma como é implementado o sistema, detectar possíveis erros na transmissão dos números de identificação, sendo assim uma ferramenta essencial para garantir uma maior segurança e eficácia no processo de identificação e transmissão de informação. A segurança é avaliada, normalmente, pelo tipo de erros que o sistema consegue detetar. No quadro seguinte podemos consultar quais os tipos de erros mais comuns:

Tipo de erro	Exemplo	Frequência relativa
<i>Singulares</i>	$2345 \Rightarrow 2945$	79,1%
<i>Troca de algarismos adjacentes</i>	$2345 \Rightarrow 2435$	10,2%
<i>Troca de algarismos intercalados</i>	$2345 \Rightarrow 2543$	0,8%
<i>Gémeos</i>	$2335 \Rightarrow 2995$	0,5%
<i>Gémeos intercalados</i>	$2343 \Rightarrow 2646$	0,3%
<i>Aleatórios e Fonéticos</i>		9,1%

Os diversos sistemas de identificação modulares apresentam, na sua base de conceção, símbolos de controlo ou algarismos de teste que permitem, em caso de erro, que o utilizador seja alertado para esse facto. Estes sistemas utilizam para esse efeito um algoritmo pré-definido, em que o algarismo de teste é determinado de acordo com esse algoritmo, e baseiam-se, regra geral, em congruências lineares módulo k , em que k é um inteiro positivo definido pelo algoritmo. Cada sistema de identificação módulo k apresenta um vector de verificação, (v_1, v_2, \dots, v_n) . A eficácia do sistema é determinada, como já referi, pelo tipo de erros que consegue detetar, erros esses, que são detetados segundo determinadas condições inerentes ao algoritmo. No quadro

seguinte podemos consultar essas condições:

Tipo de erro	Condição
<i>Singulares</i>	$mdc(v_i, k) = 1$
<i>Troca de algarismos adjacentes</i>	$mdc(v_i - v_j, k) = 1$
<i>Troca de algarismos intercalados</i>	$mdc(v_i - v_{i+1}, k) = 1$
<i>Gêmeos</i>	$mdc(v_i - v_{i+2}, k) = 1$
<i>Gêmeos intercalados</i>	$mdc(v_i + v_j, k) = 1$

Como facilmente se percebe, cada sistema tem por base um objetivo que se prende com a deteção de determinado tipo de erros, ou, por outro lado, a deteção da maior percentagem possível de erros. Assim, o vetor de verificação, bem como, o valor de k , são selecionados com esse intuito. Deste modo, para satisfazer as condições inerentes a cada erro, o ideal é escolher para k um número primo, já que assim seria fácil encontrar números primos com k , para constituírem o vetor, o que facilitaria a deteção dos erros. Os sistemas módulo k , com k primo, são, por essa razão os mais utilizados, já que permitem detetar a totalidade dos erros singulares, transposições e erros gêmeos. Um dos sistemas de identificação mais conhecidos é o ISBN (*International Standard Book Number*), que é utilizado para identificar livros ou publicações. O ISBN-10 é número de identificação de dez algarismos $(a_1 a_2 \dots a_{10})$, em que os primeiros nove identificam o produto e o último número é o algarismo de teste. O vetor de verificação associado ao ISBN-10 é $(10, 9, 8, 7, 6, 5, 4, 3, 2, 1)$. O algarismo de teste, a_{10} é determinado de forma a que:

$$S = (a_1 a_2 \dots a_{10}) \cdot (10, 9, \dots, 1) \equiv 0 \pmod{11}$$

Exemplo 1.4.6. (*Cálculo do algarismo de teste*) O livro “O tio Petros e a Conjetura de Goldbach”, de Apostolos Doriadis, apresenta o seguinte ISBN-10:

$$972 - 1 - 04857 - 7$$

Vejamos como é determinado o algarismo de teste:

$$(9, 7, 2, 1, 0, 4, 8, 5, 7, a_{10}) \cdot (10, 9, 8, 7, 6, 5, 4, 3, 2, 1) = 257 + a_{10} \equiv 0 \pmod{11}$$

Como $264 \equiv 0 \pmod{11}$, temos que $a_{10} = 264 - 257 = 7$.

O sistema ISBN-13 é constituído por doze algarismos que identificam o produto e um último, a_{13} , algarismo de teste, que é determinado de forma a que:

$$S = (a_1 a_2 \dots a_{13}) \cdot (1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1) \equiv 0 \pmod{11}$$

Exemplo 1.4.7. (*Cálculo do algarismo de teste*) O livro “Os números primos” de Enrique Gracián, apresenta o seguinte ISBN-13:

$$978 - 84 - 473 - 70 - 22 - 1$$

Vejamos como é determinado o algarismo de teste:

$$(9, 7, 8, 8, 4, 4, 7, 3, 7, 0, 2, 2, a_{13}) \cdot (1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1) = 109 + a_{13} \equiv 0 \pmod{11}$$

Como $110 \equiv 0 \pmod{11}$, temos que $a_{13} = 110 - 109 = 1$.

Ao compararmos os dois sistemas anteriores, podemos observar que o sistema ISBN-10 é mais seguro do que o sistema ISBN-13, visto que, o sistema ISBN-13 não detecta trocas de algarismos intercalados, já que o vetor de verificação apresenta números repetidos intercaladamente. Por exemplo, no caso anterior, se em vez de 978 colocássemos 879, a soma de verificação ainda seria 109.

Observação 1.4.8. *Como estamos perante um sistema módulo 11, pode ocorrer que o algarismo de teste seja 10, sendo que nesse caso, é utilizado o carácter X.*

Outro sistema de identificação muito utilizado no dia-a-dia é o sistema utilizado no NIB (*Número de identificação Bancário*), que é um sistema módulo 97, constituído por vinte e um algarismos, sendo os últimos dois os algarismos de teste. Os algarismos de testes são determinados de forma a que:

$$(a_1, a_2, a_3, a_4, a_5, \dots, a_{19}, t)$$

×

$$(73, 17, 89, 38, 62, 45, 53, 15, 50, 5, 49, 34, 81, 76, 27, 90, 9, 30, 3, 1) \equiv 1 \pmod{97}$$

Exemplo 1.4.9. *Consideremos o seguinte NIB : 0035 071400004758300 94.*

Vejamos como são determinados os algarismos de teste:

a_i	0	0	3	5	0	7	1	4	...	4	7	5	8	3	0	0	t
v_i	73	17	89	38	62	45	53	15	...	81	76	27	90	30	30	3	1
$a_i \times v_i$	0	0	267	190	0	315	53	60	...	324	532	135	720	27	0	0	t

Assim, a soma de controlo é 2623. Deste modo, $2623 + t \equiv 1(\text{mod } 97)$, mas, como $2717 \equiv 1(\text{mod } 97)$, temos que $t = 2717 - 2623 = 94$.

Observação 1.4.10. *Como estamos perante um sistema módulo 97, pode ocorrer que t tenha apenas um algarismo, sendo que nesse caso é acrescentado um zero. Por exemplo, se $t = 8$, teríamos $a_{20} = 0$ e $a_{21} = 8$.*

Capítulo 2

Conjetura de Goldbach: Enquadramento histórico e evolução

A Conjetura de Goldbach é um dos mais antigos problemas de Teoria dos Números por resolver e até mesmo da Matemática. Nos últimos 250 anos, a conjectura fascinou muitos matemáticos e curiosos. Em 1900, no 2.º Congresso Internacional de Matemática, realizado em Paris, David Hilbert, matemático alemão, propôs 23 problemas em aberto [Yandell (2002)], para os matemáticos do século XX. Muitos dos problemas apresentados já foram resolvidos, ou pelos menos, parcialmente, mas a Conjetura de Goldbach, que fazia parte do problema 8, é um dos poucos que permanece em aberto. No último século, principalmente, muito trabalho tem sido feito no sentido de provar a conjectura, embora sem resultado. No entanto, essa procura, tem contribuído para o desenvolvimento da própria Teoria dos Números, na medida em que têm surgido outros resultados, menos importantes que a conjectura, mas que podem permitir, quem sabe, comprovar a conjectura. A investigação da conjectura tem, por outro lado, permitido o desenvolvimento de métodos úteis à Teoria dos Números e até mesmo noutras áreas da Matemática. A Conjetura de Goldbach já serviu de inspiração para as mais diversas áreas, como por exemplo, o Cinema e o Literatura. O filme espanhol “La Habitación de Fermat” conta a história de dois matemáticos, de gerações distintas, que afirmavam ter a demonstração da conjectura. Na Literatura,

destaca-se o livro “O tio Petros e a Conjetura de Goldbach” de Apostolos Doxiadis, que conta a história de Petros, um famoso matemático da comunidade científica, que passou a vida a tentar provar a conjectura, e que nos últimos momentos de loucura, o conseguiu, utilizando feijões.

2.1 Vida e obra de Goldbach

Christian Goldbach nasceu a 18 de Março de 1690, em Königsberg, Prússia, atualmente Kaliningrado, Rússia, e morreu a 20 de Novembro de 1764, em Moscovo. Filho de um Pastor da Igreja Protestante, Goldbach estudou Matemática, mas principalmente Direito e Medicina. Em 1710, iniciou uma das várias viagens pela Europa, estabelecendo contacto com muitos dos intelectuais da época. Durante essa viagem, em 1711, conheceu Leibniz, na cidade de Leipzig. Durante os dois anos seguintes, os dois trocaram correspondência em latim. Em 1712, em Londres, Goldbach encontrou-se com De Moivre e Nikolaus Bernoulli que, tal como Goldbach, também viajava pela Europa. Os dois voltaram a encontrar-se em Oxford, onde Bernoulli, apercebendo-se do fascínio de Goldbach pela Matemática, tentou discutir o tema das séries infinitas, mas Goldbach nada sabia sobre o assunto, confessando mesmo que os seus conhecimentos matemáticos eram muito limitados. Sabe-se mesmo que Bernoulli terá entregue a Goldbach um texto sobre as séries infinitas, mas que Goldbach terá afirmado que o assunto era demasiado difícil e não tinha compreendido o texto. Mais tarde, em 1721, em Veneza, os dois voltaram a encontrar-se, e por sugestão de Nikolaus Bernoulli, Goldbach começou, em 1723, e durante os sete anos seguintes, a trocar correspondência com Daniel Bernoulli, irmão mais novo de Nikolaus. Entre 1712 e 1724, Goldbach publicou alguns textos, um dos quais sobre as séries infinitas. Por essa altura, já era um matemático estabelecido e reconhecido, mas sem que o seu contributo fosse considerado muito relevante para o conhecimento matemático.

Goldbach regressou a Königsberg em 1724 e no ano seguinte foi nomeado professor de Matemática e História da recém criada Academia Imperial de Ciências, mais tarde denominada Academia de Ciências de São Petersburgo, e desempenhou ainda o cargo

de secretário de gravação até 1728, quando foi enviado para Moscovo para ser tutor do Czar Peter II. Leonhard Euler, por sua vez, tinha chegado a São Petersburgo em Maio de 1727, e algum tempo depois de Goldbach se ter mudado para Moscovo, estes iniciaram uma troca de correspondência que duraria 25 anos, entre 1729 e 1754. Dessa correspondência, 196 cartas sobreviveram. Muitas destas cartas tratam variados problemas de Teoria dos Números, alguns deles apresentados anteriormente por Pierre de Fermat. A extensiva correspondência entre Goldbach e Euler é uma fonte de informação sobre a história da Matemática no século XVIII, nomeadamente fornecendo um registo fundamental do legado de Euler na Teoria dos Números, mais até dos que as próprias publicações de Euler. Numa dessas cartas, entretanto perdida, Goldbach apresentou, segundo Euler, a demonstração de um teorema, que ficou conhecido como o teorema Goldbach-Euler:

$$\sum_{m,n \geq 2} \frac{1}{m^n - 1} = 1 \text{ (Eliminando termos repetidos) [Yuan (2002)]}$$

Em 1732, Goldbach regressou a São Petersburgo, onde voltou a representar um papel ativo na Academia, bem como no governo Russo. Foi-lhe atribuído o cargo de secretário correspondente da Academia e em 1737 tornou-se corresponsável pela administração da Academia. Devido às crescentes responsabilidades no Governo Russo, em 1740, e a pedido de Goldbach, as suas obrigações com a Academia foram sendo reduzidas, cessando completamente quando foi nomeado para uma posição relevante no Ministério dos Negócios Estrangeiros. Goldbach foi-se tornando, ao longo dos anos, uma figura importante da sociedade, e em 1760 tornou-se conselheiro privado e foi responsável por linhas orientadoras da educação para as crianças da realeza que perduraram por um século.

Goldbach desenvolveu um trabalho importante na Teoria Dos Números, conhecido principalmente através da correspondência entre Goldbach e Euler, no entanto aparentemente Goldbach olhava para a Matemática como uma atividade lúdica, não desenvolvendo muitos esforços, não obstante, era atribuída a Goldbach uma intuição matemática notável.

2.2 Enquadramento histórico

Goldbach, numa das cartas enviada a Euler, em 7 de Junho de 1742, propôs:

Todo o inteiro que pode ser escrito como a soma de dois primos, também pode ser escrito como a soma de tantos primos quanto necessário, até todos os termos serem unitários.

Na mesma carta, Goldbach propôs ainda, na margem da mesma:

Todo o inteiro superior a 2 pode ser escrito como a soma de três primos.

Esta conjectura ficou conhecida como a conjectura marginal de Goldbach que considerava, tal como outros matemáticos da época, que 1 era um número primo, convenção esta que mais tarde foi abandonada. Atualmente, as conjecturas são consideradas equivalentes, mas a versão atual da conjectura marginal de Goldbach é:

Todo o inteiro maior que 5 pode ser escrito como a soma de três primos.

Goldbach ainda nessa carta, na sequência da sua conjectura inicial, referiu:

Todo o inteiro par maior que 2 pode ser escrito como a soma de dois primos.

Esta última conjectura é atualmente conhecida como Conjetura de Goldbach. Em resposta a Goldbach, Euler escreveu numa carta, em 30 Junho de 1742, que a conjectura marginal a ser verdadeira se decompunha em duas:

Todo o número par maior que 2 é a soma de dois primos.
(Já enunciada antes por Goldbach)

Todo o número ímpar superior a 6 é a soma de três primos.

Euler afirmou que estava absolutamente convicto que a conjectura era verdadeira, contudo não era capaz de a provar. As versões modernas das conjecturas anteriores são:

Todo o inteiro par maior que 2 pode ser escrito como a soma de dois números primos. (Conjetura forte)

Todo o inteiro ímpar maior que 6 pode ser escrito como a soma de três números primos.

A primeira conjetura, conhecida como a conjetura “forte”, “par” ou “binária” de Goldbach, ou simplesmente por Conjetura de Goldbach, permanece por demonstrar, enquanto que a segunda foi demonstrada, no século XX, para números suficientemente grandes. Esta última deu origem a outra conjetura, conhecida por conjetura “fraca”, “ímpar” ou “ternária” de Goldbach, que afirma:

Todo o inteiro ímpar maior que 7 pode ser escrito como a soma de três números primos ímpares. (Conjetura fraca)

Esta conjetura também permanece por demonstrar, mas no entanto, tem permitido obter outros resultados relevantes, como veremos mais tarde na secção 2.4. Estas duas últimas conjeturas parecem semelhantes, mas não o são. A conjetura fraca de Goldbach é mais específica pois impõe que os três números primos sejam ímpares, enquanto que a formulada por Euler não refere se algum dos números primos pode ser par. Um exemplo simples é o número ímpar 21. Uma decomposição possível é $21=2+2+17$, logo é possível decompor o número 21 numa soma de três primos. No entanto, para confirmar a Conjetura Fraca de Goldbach para 21, é necessário encontrar três números primos ímpares. Obviamente que também não é uma tarefa difícil, pois por exemplo, $21=3+7+11$. Contudo, se utilizarmos um número ímpar elevado, que se decomponha utilizando duas vezes o primo 2, pode tornar-se moroso encontrar três primos ímpares.

Por outro lado, se a Conjetura Forte de Goldbach for verdadeira então a Conjetura Fraca de Goldbach também será verdadeira. Com efeito, se qualquer número par superior a 2 pode ser escrito como a soma de dois números primos, então qualquer par superior a 4 pode ser escrito como a soma de dois primos ímpares, e assim, adicionando 3 a qualquer número par superior a 4, obtemos um número ímpar superior a 7 escrito como a soma de três números primos ímpares.

Por exemplo, dado o número par 22, que pode ser escrito como a soma de 5 com 17, ambos primos ímpares, temos que o número ímpar 25 pode ser escrito como a soma de 7 com 17 e com 3, sendo os três números, primos ímpares.

2.3 Explorando a Conjetura de Goldbach

Na secção anterior foi apresentada a Conjetura de Goldbach, contudo, sem a concretizar. Nesta secção vamos explorar a conjectura:

Todo o número par superior a 2 pode ser escrito como a soma de dois números primos.

Por exemplo,

$$4 = 2 + 2$$

$$6 = 3 + 3$$

$$8 = 3 + 5$$

$$10 = 5 + 5$$

$$12 = 5 + 7$$

$$14 = 7 + 7$$

Ao primeiro contacto com a conjectura ficamos com a impressão que a conjectura refere-se a algo simples, como sendo uma propriedade, facilmente verificável, dos números pares. De facto, para números pares relativamente pequenos, rapidamente se encontram dois primos cuja soma é esse par, com recurso a uma lista de primos ou a um computador. A questão fulcral que se coloca é: se para qualquer par é possível sempre encontrar esses dois primos? Esta pergunta, evidentemente, ainda não tem resposta, mas é aceite pela Comunidade Matemática que, provavelmente, a conjectura é verdadeira.

Definição 2.3.1. *Seja a um número par superior ou igual a 4. Se existirem p e q , primos, tal que $a = p + q$, então diz-se que p e q formam uma partição de Goldbach.*

Por exemplo, 5 e 11 formam uma partição de Goldbach para 16, pois são ambos primos e $16 = 5 + 11$.

Sobre a Conjetura de Goldbach também se pode questionar se, a ser verdadeira, o número de formas de representar um número par como soma de dois primos, isto é, o número de partições de Goldbach, tende a aumentar, ou se tal como a distribuição dos

números primos esse número de partições não apresenta uma regra ou lei definida. Por exemplo, para os primeiros nove números pares, verifica-se que o número de partições de Goldbach tende a aumentar:

$$\begin{aligned}
 4 &= 2 + 2 \\
 6 &= 3 + 3 \\
 8 &= 3 + 5 \\
 10 &= 5 + 5 = 3 + 7 \\
 12 &= 5 + 7 \\
 14 &= 7 + 7 = 3 + 11 \\
 16 &= 3 + 13 = 5 + 11 \\
 18 &= 5 + 13 = 7 + 11 \\
 20 &= 3 + 17 = 7 + 13 \\
 22 &= 3 + 19 = 5 + 17 = 11 + 11
 \end{aligned}$$

Definição 2.3.2. *Seja a um inteiro positivo par superior a 2. Define-se como $G(a)$, o número de partições de Goldbach de a , isto é, G é a função que a cada inteiro positivo par a superior a 2 faz corresponder o número de partições de Goldbach associadas:*

$$G(a) = \#\{(p, q), \text{ primos: } a = p + q, p \leq q\}$$

Deste modo, temos:

$$G(8) = 1, \text{ pois } 8 = 3 + 5.$$

$$G(10) = 2, \text{ pois } 10 = 3 + 7 = 5 + 5.$$

$$G(22) = 3, \text{ pois } 22 = 3 + 19 = 5 + 17 = 11 + 11.$$

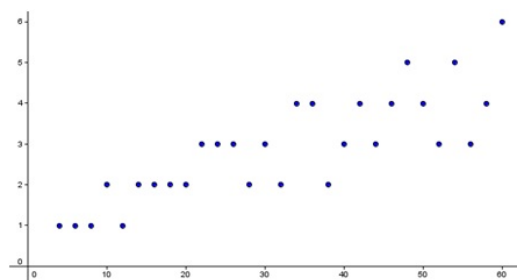


Figura 2.3.1: Gráfico - Número de partições de Goldbach para os pares até 60.

Ao observarmos o gráfico verifica-se que o número de partições de Goldbach aumenta à medida que o número par aumenta. Esta observação, como também podemos notar no gráfico, é a regra, mas em alguns casos verifica-se o contrário. Por exemplo,

1. 10 tem duas partições de Goldbach ($5 + 5$; $3 + 7$), enquanto 12 tem apenas uma ($5 + 7$);
2. 26 tem três partições de Goldbach ($13 + 13$; $7 + 19$; $3 + 23$), enquanto 28 tem apenas duas ($5 + 23$; $11 + 17$);
3. 36 tem quatro partições de Goldbach ($5 + 31$; $7 + 29$; $13 + 23$; $17 + 19$), enquanto 38 tem apenas duas ($7 + 31$; $19 + 19$).

Em 1989, Henry Fliegel e Douglas Robertson [Fliegel (1989)] obtiveram, por computação, um gráfico que relaciona a com $G(a)$:

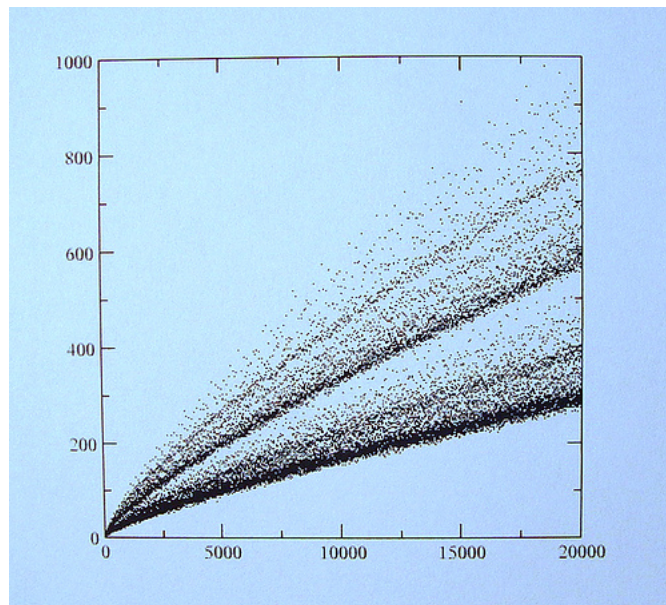


Figura 2.3.2: Gráfico - Cometa de Goldbach

O gráfico anterior foi denominado de Cometa de Goldbach, obviamente porque se assemelha à cauda de um cometa. A partir da visualização podemos conjecturar que o número de partições de Goldbach tem tendência a aumentar à medida que a aumenta.

Hardy e Littlewood [Crandall (2005)] conjecturaram que,

$$\lim_{a \rightarrow \infty} G(a) \approx \frac{2c.a}{\ln(a)} \prod \frac{p-1}{p-2}, \text{ para todo } p \text{ fator primo de } a$$

em que c , designada por constante de Hardy-Littlewood, é dada por $c \approx 0,6602$.

Esta conjectura ainda não foi provada.

Max S.C. Woon [Woon (2000)], conseguiu uma expressão para aproximar $G(a)$,

$$G(a) \approx \sum_{3 \leq k \leq \frac{a}{2}} \frac{1}{\ln(k) \cdot \ln(a-k)}, a \geq 6$$

A conjectura de Hardy-Littlewood obtém um resultado mais aproximado do que a expressão anterior, se bem que requer, ao contrário desta, que sejam conhecidos os fatores primos de a .

2.4 Evolução da Teoria dos Números no contexto da Conjetura de Goldbach

O primeiro grande resultado obtido relacionado com a Conjetura de Goldbach ocorreu em 1923, quando os matemáticos britânicos Hardy e Littlewood mostraram que, assumindo a Hipótese generalizada de Riemann [Fine (2007)], todo o número ímpar suficientemente grande é a soma de três números primos, e quase todos os números pares são a soma de dois primos. Hardy afirmou mesmo que a Conjetura de Goldbach não era só um dos problemas mais difíceis da Teoria dos Números, mas também de toda a Matemática. Já antes, em 1919, o matemático norueguês Brun tinha demonstrado que qualquer número par suficientemente grande é a soma de dois números, cada um tendo no máximo nove fatores primos. Em 1930, e com base no resultado de Brun, o russo Lev Schnirelman provou que todo o inteiro par igual ou superior a 2 é a soma de, no máximo, vinte primos. Este resultado tem sido melhorado, sendo que o melhor resultado conhecido deve-se a Olivier Ramaré que, em 1995, conseguiu reduzir o número máximo de primos para seis.

A investigação da conjectura obteve também um grande resultado em 1937, pelo Russo I.M. Vinogradov, que conseguiu remover a dependência na hipótese de Riemann, originando assim a prova incondicional das conclusões de Hardy e Littlewood, mas não conseguiu determinar o que significava “suficientemente grande”. Atualmente, assume-se que números “suficientemente grandes” são não inferiores a 2×10^{1346} , o que, mesmo com verificação computacional, torna inviável a verificação de todos os ímpares inferiores. As pesquisas mais recentes têm apenas conseguido verificar até 10^{20} .

Quase três décadas mais tarde, em 1966, o matemático chinês Chen Jingrun, baseado no resultado de Brun, demonstrou que todo o número par suficientemente grande é a soma de um número primo e um produto de, no máximo, dois primos. Em 1975, Hugh Montgomery and Robert Charles Vaughan provaram que a quantidade de números pares inferiores ou iguais a um determinado inteiro x , que não podem ser escritos como a soma de dois números primos é no máximo Cx^{1-c} , em que c e C são

constante positivas.

Todos estes desenvolvimentos, relacionados com a Conjetura de Goldbach, deram-se devido aos grandes desenvolvimentos na teoria analítica dos números, no século XIX, em particular, a teoria de Riemann e Dirichlet, na distribuição dos números primos, que é, aparentemente, um pré-requisito na investigação atual. Em 1997, Deshouillers, Effinger, Te Riele e Zinoviev mostraram que a Hipótese de Riemann implica a Conjetura fraca de Goldbach, para todos os números ímpares.

2.5 Diferentes abordagens à conjectura

Nos últimos anos, e devido à emergente fonte de informação e divulgação que é a *World Wide Web*, têm surgido muitos textos sobre a Conjetura de Goldbach, alguns dos quais com supostas demonstrações da mesma. Até à data, nenhuma foi aceite pela comunidade matemática, todavia, é possível retirar de algumas informação relevante, como, por exemplo, diferentes formas de abordar a conjectura. Alguns autores procuram, em primeiro lugar, criar conjecturas equivalentes à Conjetura de Goldbach e, de seguida, abordar o problema utilizando uma nova conjectura, em moldes diferentes. Outros procuram, por outro lado, explorar as propriedades dos números, tentando desenvolver um método que permita depois provar a conjectura.

As seguintes conjecturas são equivalentes:

- i. (Conjetura de Goldbach) Seja a um número par maior que 2. Existem p e q , primos, de forma a que: $a = p + q$;
- ii. A combinação de todos os primos ímpares, somados dois de cada vez, gera todos os números pares superiores a 4;
- iii. Seja a um número par maior que 2. Existe p , primo, com $3 \leq p \leq a - 3$, de forma a que $a - p$ seja primo;
- iv. Seja a um número par maior que 2. Existe j , inteiro, de forma a que:

$$a = 2n = (n + j) + (n - j) \text{ com } n + j \text{ e } n - j \text{ primos.}$$

2.5.1 Roger Ellman

Como sabemos, todos os números primos, exceto 2, são ímpares. Por outro lado, também sabemos que a soma de dois ímpares é um número par. Assim, se provássemos que com as combinações de todos os números primos ímpares, somados dois de cada vez, obteríamos todos os números pares superiores a 4, a Conjetura de Goldbach seria verdadeira.

Vejamos, por exemplo, como podemos verificar a conjectura para todos os pares menores ou iguais a 100: A partir do conjunto dos números primos $P = \{3, 5, 7, 11, \dots\}$, a primeira combinação será $3 + p_i$, com $p_i \in P$.

Subconjunto $\{3 + p_i\}$, com $p_i \in P$:

6	8	10	12	14	16	18	20	22	24	26	28
30	32	34	36	38	40	42	44	46	48	50	52
54	56	58	60	62	64	66	68	70	72	74	76
78	80	82	84	86	88	90	92	94	96	98	100

Subconjuntos $\{3 + p_i\}$ e $\{5 + p_i\}$ com $p_i \in P$:

6	8	10	12	14	16	18	20	22	24	26	28
30	32	34	36	38	40	42	44	46	48	50	52
54	56	58	60	62	64	66	68	70	72	74	76
78	80	82	84	86	88	90	92	94	96	98	100

Subconjuntos $\{3 + p_i\}$, $\{5 + p_i\}$ e $\{7 + p_i\}$ com $p_i \in P$:

6	8	10	12	14	16	18	20	22	24	26	28
30	32	34	36	38	40	42	44	46	48	50	52
54	56	58	60	62	64	66	68	70	72	74	76
78	80	82	84	86	88	90	92	94	96	98	100

Como podemos observar, combinando apenas 3, 5 e 7 com os demais primos, o único par inferior a 100 que não é combinação de um dos subconjuntos anteriores é o 98, cuja partição de Golbach possível é $98 = 19 + 79$. Ao analisarmos as sucessivas etapas, torna-se evidente que, de facto, é possível gerar todos os números pares de uma lista, combinando os primos dois a dois. O problema que se coloca é perceber quantos subconjuntos são necessários para o conseguir. Por exemplo, para verificar

os números pares até 300, os últimos pares a serem gerados seriam 98, 128 e 208, que resultam de:

$$98 = 19 + 79; 128 = 19 + 109; 208 = 19 + 191$$

Assim, observa-se que, independentemente de começarmos com uma lista de pares até 100, até 200 ou até 300, teríamos de combinar todos os primos até 19, o que nos leva a concluir que dificilmente se sabe à partida quantos subconjuntos serão necessários para gerar todos os pares até um determinado par.

2.5.2 Kent Slinker

A abordagem é baseada na seguinte equação:

$$a = (a - p) + p, \text{ com } 3 \leq p \leq (a - 3)$$

Assim, o objetivo é encontrar algum p primo, nas condições anteriores, de forma a que $a - p$ também seja primo, o que provaria a conjectura.

O método consiste na construção de um bloco de números primos e a partir deste determinar as partições do número par dado. Cada bloco parte inicialmente da escolha de um número par $m = p_n + 3$, em que p_n é o maior primo inferior ou igual a $a - 3$, e consiste em duas imagens refletidas de uma lista de todos os primos $p_0, p_1, p_2, \dots, p_n$, em que $p_0 = 3$ e $p_n = m - 3$.

Método: Dado a par superior a 4,

1. Se $a - 3$ for primo, então $a = (a - 3) + 3$ e terminou o método;
2. Caso contrário, determinar o maior primo inferior ou igual a $a - 3$ (p_n);
3. Considerar $m = p_n + 3$ e todos os primos até m ($p_0 = 3, p_1 = 5, \dots, p_n$);
4. Construir um bloco com os números naturais até $3, 5, \dots, p_n$;
5. Inverter e espelhar o bloco;
6. Mover a parte inferior do bloco $a - m$ unidades para a direita e encontrar as partições para a .

Vejamos um exemplo: Vamos determinar as partições de 30.

Como $a - 3 = 27$ é composto, o maior primo inferior a $a - 3$ é 23, ou seja, temos que:

$p_n = 23$ e assim $m = 26$.

Temos que considerar então os primos:

$p_0 = 3, p_1 = 5, p_2 = 7, p_3 = 11, p_4 = 13, p_5 = 17, p_6 = 19$ e $p_7 = 23$.

Vamos construir um bloco com todos os naturais até p_0, p_1, \dots, p_n :

p_7	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
p_6	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19				
p_5	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17						
p_4	1	2	3	4	5	6	7	8	9	10	11	12	13										
p_3	1	2	3	4	5	6	7	8	9	10	11												
p_2	1	2	3	4	5	6	7																
p_1	1	2	3	4	5																		
p_0	1	2	3																				

Figura 2.5.1: Bloco 1

Invertendo e espelhando o bloco:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
p_7	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	3	2	1
p_6	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19					5	4	3
p_5	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17			7	6	5	4	3	2	1
p_4	1	2	3	4	5	6	7	8	9	10	11	12	13			11	10	9	8	7	6	5	4	3	2	1
p_3	1	2	3	4	5	6	7	8	9	10	11			13	12	11	10	9	8	7	6	5	4	3	2	1
p_2	1	2	3	4	5	6	7				17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
p_1	1	2	3	4	5			19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
p_0	1	2	3	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Figura 2.5.2: Bloco 2

Como $a - m = 4$ ($30 - 26$), temos de deslocar a parte inferior do bloco 4 unidades para a direita e procurar as partições para 30:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
p_7	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23					3	2	1
p_6	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19							5	4	3	2	1
p_5	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17							7	6	5	4	3	2	1
p_4	1	2	3	4	5	6	7	8	9	10	11	12	13					11	10	9	8	7	6	5	4	3	2	1		
p_3	1	2	3	4	5	6	7	8	9	10	11			13	12	11	10	9	8	7	6	5	4	3	2	1				
p_2	1	2	3	4	5	6	7				17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1			
p_1	1	2	3	4	5			19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1				
p_0	1	2	3			23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		

Figura 2.5.3: Bloco 3

Deste modo, temos:

$$30 = 7 + 23 ; 30 = 11 + 19 \text{ ou } 30 = 13 + 17$$

Com o bloco criado inicialmente, também seria possível determinar as partições para 28, já que 23 ainda seria o maior primo inferior ou igual a 25 ($28-3$), bastando neste caso mover a parte inferior do bloco apenas 2 unidades para a direita:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28		
p_7	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23			3	2	1		
p_6	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19					5	4	3	2	1		
p_5	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17					7	6	5	4	3	2	1		
p_4	1	2	3	4	5	6	7	8	9	10	11	12	13				11		10	9	8	7	6	5	4	3	2	1		
p_3	1	2	3	4	5	6	7	8	9	10	11					13	12	11	10	9	8	7	6	5	4	3	2	1		
p_2	1	2	3	4	5	6	7					17		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
p_1	1	2	3	4	5							19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
p_0	1	2	3			23	22	21	20	19	18	17	16	15	14	13	12	1	10	9	8	7	6	5	4	3	2	1		

Figura 2.5.4: Bloco 4

Por outras palavras, dado o bloco com os primos até 23, é possível determinar as decomposições para os pares 26, 28 e 30. Se pensarmos no próximo par, 32, verifica-se que não é possível utilizar o bloco já construído, já que o maior primo inferior ou igual a 29 ($32-3$) é o próprio 29, e assim o bloco teria de conter mais um linha com os números naturais até 29. Com este novo bloco seria possível determinar as decomposições para 32, contudo, para 34, precisaríamos de criar mais um linha, pois 31 ($34-3$) é primo.

Posto isto, se nos restringirmos a mover a parte da direita do bloco, a partir de uma posição inicial, apenas enquanto não surgir um novo primo, poderíamos tentar provar que cada movimento produz partições de Goldbach.

Este método, se bem que visualmente apelativo e de simples desenvolvimento, é útil se pretendermos determinar as partições de $m+2$, $m+4$, ..., a partir do bloco de partições de m . Assim, por exemplo, considerando $m = p_n + 3$, podemos determinar as decomposições de todos os pares $m+2$, $m+4$, ..., enquanto não surgir um primo da forma $a-3$.

Dado a par, $p_0 = 3$, $m = p_0 + p_n$, em que p_n é o maior primo inferior ou igual a $a-3$, o valor de m na posição inicial do bloco é $p_0 + p_n$. Designamos por r_i o espaço entre cada primo p_i e a sua imagem refletida p_{n-i} , ou seja, a diferença entre m e $(p_{n-i} - p_i)$, com $0 \leq i \leq n$:

$$r_a = m - (p_{n-i} - p_i) \Leftrightarrow m = r_i + p_{n-i} + p_i$$

Desta forma, na partição inicial do bloco, $r_0 = r_n = 0$, pois $m = p_0 + p_n$.

Definimos r_t como sendo a diferença entre a e m , ou seja, o número de movimentos

para a direita que é necessário a parte refletida do bloco efetuar a partir da partição inicial de m para obter as partições de a . Desta forma, temos que $a = m + r_t$.

Uma partição de Goldbach ocorre quando a soma de um primo na parte inferior do bloco com um primo da parte superior é a . Assim, se $p_i + r_i + r_t$ for um primo mais acima no bloco, então $p_i + r_i + r_t$ e p_{n-i} representam uma partição de Goldbach.

Teorema 2.5.1. *(Kent Slinker (2005)) Seja a um número par superior ou igual a 4. Se existir um índice i , com $0 \leq i \leq n$, tal que $p = p_i + r_i + r_t < p_n$ for primo, então $a - p$ é primo.*

Demonstração.

Ora, como $a = m + r_t$ e $m = r_i + p_{n-i} + p_i$, temos que, $a = (r_i + p_{n-i} + p_i) + r_t$. Reagrupando, obtemos:

$$a = (p_i + r_i + r_t) + p_{n-i} \Leftrightarrow a = p + p_{n-i} \Leftrightarrow a - p = p_{n-i}$$

Assim, $a - p$ é primo. □

2.5.3 Bernard Farley

A abordagem é baseada na seguinte equação:

$$a = 2n = (n - j) + (n + j), \text{ com } n - j \text{ e } n + j \text{ primos.}$$

O objetivo é encontrar j inteiro, de forma a que $n - j$ e $n + j$ sejam primos.

Teorema 2.5.2. *(Bernard Farley (2005)) Seja n um número inteiro superior ou igual a 2. Então, existe j inteiro tal que $n - j$ e $n + j$ não são divisíveis por $2, 3, 5, \dots, p_k$, em que p_k é o maior primo inferior ou igual a $\sqrt{2n}$.*

Demonstração.

Seja n um número inteiro superior ou igual a 2, e consideremos p_1, p_2, \dots, p_k , os primos inferiores ou iguais a $\sqrt{2n}$.

Consideremos $a_i \not\equiv \pm n \pmod{p_i}$, com $1 \leq i \leq k$. Pelo Teorema do Resto Chinês,

existe uma solução módulo m , onde $m = p_1.p_2.\dots.p_k$, para o seguinte sistema de congruências:

$$\begin{cases} j \equiv a_1(\text{mod } p_1) \\ j \equiv a_2(\text{mod } p_2) \\ \dots \\ j \equiv a_k(\text{mod } p_k) \end{cases}$$

Então, $j \equiv a_i \not\equiv \pm n(\text{mod } p_i)$, e logo, $n \pm j \not\equiv 0(\text{mod } p_i)$ para $1 \leq i \leq k$. Assim, $n + j$ e $n - j$ não são divisíveis por p_1, p_2, \dots, p_k . \square

Se $n \pm j \leq 2n$, então $n \pm j$ é primo já que $\sqrt{n \pm j} \leq \sqrt{2n}$ e $n \pm j$ não é divisível por todos os números primos inferiores ou iguais a $\sqrt{2n}$.

Contudo, j é uma solução módulo m , logo, j pode ser superior a n , o que implicaria que $n + j > 2n$ e $n - j < 0$. Assim, se $|j| \leq n - 2$, então $n + j$ e $n - j$ são primos.

Como o valor de j depende dos valores de a_i escolhidos, é necessário que estes sejam escolhidos de forma a que o valor de j encontrado torne $n + j$ e $n - j$ em números primos.

Exemplo 2.5.3. Consideremos $a = 70$. Assim temos que $n = 35$ e $\sqrt{70} \approx 8.4$, logo temos que:

$$\pm 35 \equiv 1(\text{mod } 2)$$

$$35 \equiv 2(\text{mod } 3) \text{ e } -35 \equiv 1(\text{mod } 3)$$

$$\pm 35 \equiv 0(\text{mod } 5)$$

$$\pm 35 \equiv 0(\text{mod } 7)$$

Assim, $a_1 \neq 1; a_2 \neq \{1, 2\}; a_3 \neq 0; a_4 \neq 0$.

Por exemplo, $a_1 = 0; a_2 = 0; a_3 = 1; a_4 = 1$. Temos que,

$$\begin{cases} j \equiv 0(\text{mod } 2) \\ j \equiv 0(\text{mod } 3) \\ j \equiv 1(\text{mod } 5) \\ j \equiv 1(\text{mod } 7) \end{cases}$$

Utilizando o Teorema do Resto Chinês, obtemos $j = 36$, que não podemos utilizar pois é superior a $33(n - 2)$, mas se fizermos $a_1 = 0; a_2 = 0; a_3 = 1; a_4 = 6$, obtemos $j = 6$. Assim, $n + j = 35 + 6 = 41$ e $n - j = 35 - 6 = 29$, ambos números primos. É possível encontrar mais partições de Goldbach de 70 utilizando este processo:

$n - j$	3	11	17	23	29	31
$n + j$	67	59	53	47	41	39
j	32	24	18	12	6	4

Como $j \equiv 0(\text{mod } 2)$ e $j \equiv 0(\text{mod } 3)$, j é múltiplo de 2 e 3 e, por consequência, múltiplo de 6. Logo, não é possível encontrar todas as partições de 70, já que 4 e 32 não são múltiplos de 6. No entanto, as restantes partições podem ser encontradas, pois 6, 12, 18 e 24 são múltiplos de 6.

Mais precisamente, se fizermos:

$a_1 = a_2 = 0; a_3 = 2; a_4 = 5$, obtemos $j = 12$;

$a_1 = a_2 = 0; a_3 = 3; a_4 = 4$, obtemos $j = 18$;

$a_1 = a_2 = 0; a_3 = 4; a_4 = 3$, obtemos $j = 24$.

Exemplo 2.5.4. Consideremos $a = 132$. Assim temos que $n = 66$ e $\sqrt{132} \approx 11.5$, logo temos que:

$$\pm 66 \equiv 0(\text{mod } 2)$$

$$\pm 66 \equiv 0(\text{mod } 3)$$

$$66 \equiv 1(\text{mod } 5) \text{ e } -66 \equiv 4(\text{mod } 5)$$

$$66 \equiv 3(\text{mod } 7) \text{ e } -66 \equiv 4(\text{mod } 7)$$

$$\pm 66 \equiv 0(\text{mod } 11)$$

Assim, $a_1 \neq 0; a_2 \neq 0; a_3 \neq \{1; 4\}; a_4 \neq \{3 : 4\}; a_5 \neq 0$.

Por exemplo, $a_1 = 1; a_2 = 1; a_3 = 0; a_4 = 0; a_5 = 2$. Temos que,

$$\left\{ \begin{array}{l} j \equiv 1(\text{mod } 2) \\ j \equiv 1(\text{mod } 3) \\ j \equiv 0(\text{mod } 5) \\ j \equiv 0(\text{mod } 7) \\ j \equiv 2(\text{mod } 11) \end{array} \right.$$

Utilizando o Teorema do Resto Chinês obtemos $j = 415$ que não podemos utilizar pois é superior a $65(n - 2)$, mas se fizermos $a_1 = 1; a_2 = 2; a_3 = 0; a_4 = 0; a_5 = 2$, obtemos $j = 35$. Assim, $n + j = 66 + 35 = 101$ e $n - j = 66 - 35 = 31$, ambos números primos.

É possível encontrar mais partições de Goldbach utilizando este processo. As partições de Goldbach de 132:

$n - j$	5	19	23	29	31	43	53	59	61
$n + j$	127	113	109	103	101	89	79	73	71
j	61	49	43	37	35	23	13	7	5

Como $j \not\equiv 0(mod\ 2)$, $j \not\equiv 0(mod\ 3)$ e $j \not\equiv 0(mod\ 11)$, j não é múltiplo de 2, 3 e 11, por consequência, j é primo ou múltiplo de 5 ou 7. Se verificarmos, de facto, j é primo (5, 7, 13, 23, 37, 43 e 61) ou múltiplo de 5 ou 7 (35 e 49).

Observação 2.5.5. *Esta abordagem à Conjetura de Goldbach tenta provar que é possível encontrar uma combinação $(a_1; a_2; \dots; p_k)$ cuja solução j , do sistema de congruências:*

$$\begin{cases} j \equiv a_1(mod\ p_1) \\ j \equiv a_2(mod\ p_2) \\ \dots \\ j \equiv a_k(mod\ p_k) \end{cases}$$

é um inteiro tal que $n - j$ e $n + j$ são primos, o que provaria a Conjetura de Goldbach.

2.6 Verificação da Conjetura

A Conjetura de Goldbach, embora sem demonstração, tem levado a que diversos investigadores, ao longo do tempo, verifiquem se a conjectura é, de facto, verdadeira, procurando, até hoje e sem sucesso, encontrar um número par que não admita uma partição de Goldbach.

Até ao final do século XIX, a conjectura tinha sido apenas verificada até 10000, sendo que, durante o século XX e até os dias de hoje, a conjectura já foi verificada até 4×10^{18} , recorrendo principalmente à utilização do computador. Os últimos progressos na verificação da conjectura foram feitos pelo investigador português Tomás Oliveira e Silva, professor na Universidade de Aveiro.

No seguinte quadro podemos aferir como evoluiu a verificação da Conjetura de Goldbach:

$n < \dots$	Referência
1000	Georg Cantor, século XIX
2000	A. Aubry, século XIX
5000	R. Haussner, século XIX
1×10^4	Desboves, século XIX
1×10^5	Pipping, 1938
1×10^8	Stein and Stein, 1965
2×10^{10}	Granville, 1989
4×10^{11}	Sinisalo, 1993
1×10^{14}	Deshouillers e Saouter, 1998
4×10^{14}	Richstein, 1999
2×10^{16}	Oliveira e Silva (Março 24, 2003)
6×10^{16}	Oliveira e Silva (Outubro 3, 2003)
2×10^{17}	Oliveira e Silva (Fevereiro 5, 2005)
3×10^{17}	Oliveira e Silva (Dezembro 30, 2005)
12×10^{17}	Oliveira e Silva (Julho 14, 2008)
15×10^{17}	Oliveira e Silva (Julho 24, 2009)
16×10^{17}	Oliveira e Silva (Dezembro 23, 2009)
2×10^{18}	Oliveira e Silva (Novembro 6, 2010)
4×10^{18}	Oliveira e Silva (Abril 4, 2012)

Capítulo 3

Conjetura de Goldbach: uma abordagem aritmética

Neste capítulo final é apresentado o resultado de meses de trabalho, que culminou com a criação de um método para determinar partições de Goldbach, bem como de uma aplicação em linguagem Java que traduz esse método, permitindo encontrar partições de um dado número par.

3.1 Método para determinar partições de Goldbach

A Conjetura de Goldbach enuncia que:

”Todo o inteiro par maior que 2 pode ser escrito como a soma de dois números primos.”

No secção 2.5 foram apresentadas várias conjeturas equivalentes ao enunciado anterior. Neste capítulo final será abordada a seguinte versão:

Seja a um número par maior que 2. Então existem p e q , primos, tais que:

$$a = p + q$$

Observação 3.1.1. *É imediato que se $a = 2p$, com p primo, então $a = p + p$, ou seja, a conjectura verifica-se para número pares da forma $2p$, com p primo. Caso contrário, é necessário encontrar p e q , primos distintos.*

Exemplo 3.1.2. *O número par 26 pode ser escrito como $26 = 3 + 23$, porém, outra partição possível é $26 = 13 + 13 = 2 \times 13$, isto é, 26 é da forma $2p$.*

Os pares inferiores a 100, da forma $2p$ são:

6 10 14 22 26 34 38 46 58 62 74 82 86 94

Proposição 3.1.3. *Seja a um inteiro positivo, cuja decomposição em fatores primos é dada por,*

$$a = p_1^{s_1} p_2^{s_2} \dots p_n^{s_n}, \text{ com } (p_1 < p_2 < \dots < p_n) \text{ e } (s_i \in \mathbb{N})$$

e seja $p < a$ um número primo tal que $p \neq p_i$, com $i = 1, 2, \dots, n$.

Então $a - p$ não é divisível por p_i , com $i = 1, 2, \dots, n$.

Demonstração.

Sejam p e p_i primos distintos, com $i = 1, 2, \dots, n$. Vamos supor, por absurdo, que existe um índice i tal que p_i divide $(a - p)$. Dado que p_i também divide a , pois é um fator da decomposição de a , temos que $p_i | a$ e $p_i | (a - p)$ logo, pela proposição 1.1.3 (VI), $p_i | (a - (a - p))$, ou seja, $p_i | p$. Ora, se $p_i | p$, ambos primos, pelo corolário 1.3.4, $p_i = p$, o que é um absurdo, porque p_i e p são primos distintos. Assim concluímos que $a - p$ não é divisível por p_i . \square

Exemplo 3.1.4. *Consideremos o número par 30. A sua decomposição em fatores primos é dada por $30 = 2 \times 3 \times 5$.*

Então,

$$30 - 7 = 23$$

$$30 - 11 = 19$$

$$30 - 13 = 17$$

$$30 - 17 = 13$$

$$30 - 19 = 11$$

$$30 - 23 = 7$$

não são divisíveis por 2, 3 ou 5. Observamos ainda que $a - p$ é primo, para qualquer p .

Exemplo 3.1.5. *Consideremos agora o número par 70. A sua decomposição em fatores primos é dada por $70 = 2 \times 5 \times 7$.*

Então,

$$70 - 3 = 67$$

$$70 - 11 = 59$$

$$70 - 17 = 53$$

$$70 - 19 = 51$$

$$70 - 23 = 47$$

...

$$70 - 67 = 3$$

não são divisíveis por 2, 5 ou 7.

Observamos ainda que $a - p$ é primo ou então múltiplo de 3.

Definição 3.1.6. *Seja a um inteiro positivo, cuja decomposição em fatores primos é dada por,*

$$a = p_1^{s_1} p_2^{s_2} \dots p_n^{s_n}, \text{ com } (p_1 < p_2 < \dots < p_n) \text{ e } (s_i \in \mathbb{N})$$

A partir da decomposição de a , definimos os conjuntos A e B do seguinte modo:

$$A = \{p_1, p_2, \dots, p_n\} ; B = \{t \text{ primo} : p_1 < t < p_n \wedge t \notin A\}$$

O conjunto A engloba todos os primos que fazem parte da decomposição de a , enquanto que B engloba os primos menores que p_n que não pertencem a A , isto é, não fazem parte da decomposição de a . Assim, qualquer primo inferior ou igual a p_n pertence necessariamente a um e só um dos dois conjuntos definidos.

Exemplo 3.1.7. Consideremos o número par 350, cuja decomposição em fatores primos é dada por:

$$350 = 2 \times 5^2 \times 7$$

Deste modo, temos que:

$$A = \{2, 5, 7\} \text{ e } B = \{3\}$$

Proposição 3.1.8. Seja a um inteiro positivo par, cuja decomposição em fatores primos é dada por:

$$a = p_1^{s_1} p_2^{s_2} \dots p_n^{s_n}, \text{ com } (p_1 < p_2 < \dots < p_n) \text{ e } (s_i \in \mathbb{N})$$

Se existir p primo com $p \neq p_i$ tal que $p \in]a - p_{n+1}^2, a - 3]$ e para qualquer primo $t \in B$, t não divide $a - p$, então $a - p$ é primo.

Demonstração.

Seja a um número par cuja decomposição em fatores primos é dada por $a = p_1^{s_1} p_2^{s_2} \dots p_n^{s_n}$, com $p_1 < p_2 < \dots < p_n$, e consideremos o conjunto B associado. Seja também, p primo com $p \neq p_i$ e $p \in]a - p_{n+1}^2, a - 3]$.

Em relação ao limite superior do intervalo, $a - 3$, apenas limita superiormente o valor de p , pois se $p = a - 3$, temos que q assume o menor valor possível, 3. Relativamente ao limite inferior do intervalo, temos que:

$$a - p < p_{n+1}^2 \Leftrightarrow \sqrt{a - p} < p_{n+1}$$

Ora, $a - p$ será primo se não possuir nenhum divisor primo inferior ou igual a $\sqrt{a - p}$. Como $\sqrt{a - p} < p_{n+1}$, os únicos divisores primos possíveis são p_1, p_2, \dots, p_n , isto é, $p_i \in A$ ou $t \in B$, mas como para qualquer $t \in B$, por hipótese, t não divide $a - p$, e pela proposição 3.1.3, $a - p$ não é divisível por p_i , concluímos que $a - p$ não tem nenhum divisor primo inferior ou igual a $\sqrt{a - p}$ e, sendo assim, pelo teorema 1.3.10, $a - p$ é necessariamente primo. \square

Exemplo 3.1.9. Consideremos o número par 220, cuja decomposição em fatores primos é dada por:

$$220 = 2^2 \times 5 \times 11$$

Assim, temos que $A = \{2, 5, 11\}$, $B = \{3, 7\}$, $p_n = 11$ e $p_{n+1} = 13$. Então,

$$a - p_{n+1}^2 = 220 - 13^2 = 51 \text{ e } a - 3 = 220 - 3 = 217$$

Consultando uma lista de primos, encontramos os seguintes primos no intervalo $]51, 217]$:

$$\begin{pmatrix} 53 & 59 & 61 & 67 & 71 & 73 & 79 & 83 & 89 & 97 & 101 \\ 103 & 107 & 109 & 113 & 127 & 131 & 137 & 139 & 149 & 151 & 157 \\ 163 & 167 & 173 & 179 & 181 & 191 & 193 & 197 & 199 & 211 \end{pmatrix}$$

Assim, as possíveis partições para 220 são:

p	53	59	61	67	71	73	79	83	89	97	101
$a - p$	167	161	159	153	149	147	141	137	131	123	119
p	103	107	109	113	127	131	137	139	149	151	157
$a - p$	117	113	111	107	93	89	83	81	71	69	63
p	163	167	173	179	181	191	193	197	199	211	
$a - p$	57	53	47	41	39	29	27	23	21	9	

Das possíveis partições anteriores, retirando as partições em que $a - p$ é múltiplo de 3 ou 7, neste caso, 161, 159, 153, 147, 141, 123, 119, 117, 111, 93, 81, 69, 63, 57, 39, 27, 21 e 9, obtemos as seguintes partições de Goldbach para o número par 220:

p	57	71	83	89	107	173	179	191	197
$q = a - p$	167	149	137	131	113	47	41	29	23

Recorrendo a uma folha de cálculo, verificamos que todas as partições de 220 foram encontradas.

Exemplo 3.1.10. Consideremos o número par 252, cuja decomposição em fatores primos é dada por:

$$252 = 2^2 \times 3^2 \times 7$$

Assim, temos que $A = \{2, 3, 7\}$, $B = \{5\}$, $p_n = 7$ e $p_{n+1} = 11$. Então,

$$a - p_{n+1}^2 = 252 - 11^2 = 131 \text{ e } a - 3 = 252 - 3 = 249$$

Obtemos assim o intervalo $]131; 249]$, que contém os seguintes números primos:

$$\left(\begin{array}{cccccccccc} 137 & 139 & 149 & 151 & 157 & 163 & 173 & 179 & 181 & 191 \\ 193 & 197 & 199 & 211 & 223 & 227 & 233 & 239 & 241 & \end{array} \right)$$

Assim as possíveis partições para 252 são:

p	137	139	149	151	157	163	173	179	181	191	193
$a - p$	115	113	103	101	95	89	79	73	71	61	59
p	197	199	221	227	233	239	241				
$a - p$	55	53	31	25	19	13	11				

Das possíveis partições anteriores, retirando as partições em que $a - p$ é múltiplo de 5, neste caso, 115, 95, 55 e 25, obtemos as seguintes partições de Goldbach para o número par 252:

p	139	149	151	163	173	179	181	191	193	199	211
$q = a - p$	113	103	101	89	79	73	71	61	59	53	41
p	223	233	239	241							
$q = a - p$	29	19	13	11							

Recorrendo novamente a uma folha de cálculo, verificamos que todas as partições de 252 foram encontradas.

Exemplo 3.1.11. *Consideremos o número par 700, cuja decomposição é dada por:*

$$700 = 2^2 \times 5^2 \times 7$$

Assim, temos que $A = \{2, 5, 7\}$, $B = \{3\}$, $p_n = 7$ e $p_{n+1} = 11$. Então,

$$a - p_{n+1}^2 = 700 - 11^2 = 579 \text{ e } a - 3 = 700 - 3 = 697$$

Obtemos então o intervalo $]579; 697]$, que contém os seguintes números primos:

$$\begin{pmatrix} 587 & 593 & 599 & 601 & 607 & 613 & 617 & 619 & 631 & 641 \\ 643 & 647 & 653 & 659 & 661 & 673 & 677 & 683 & 691 \end{pmatrix}$$

Assim as possíveis partições para 700 são:

p	587	593	599	601	607	613	617	619	631	641
$a - p$	113	107	101	99	93	87	83	81	69	59
p	643	647	653	659	661	673	677	683	691	
$a - p$	57	53	47	41	39	27	23	17	9	

Das possíveis partições anteriores, retirando as partições em que $a - p$ é múltiplo de 3, neste caso, 99, 93, 87, 81, 69, 57, 39, 27 e 9, obtemos as seguintes partições de Goldbach para o número par 700:

p	587	593	599	617	641	647	653	659	677	683
$q = a - p$	113	107	101	83	59	53	47	41	23	11

Neste caso, não foram determinadas todas as partições de 700. Por exemplo, 700 pode ser escrito como a soma de 347 com 353, ambos números primos.

Mais tarde veremos em que condições este método garante que foram encontradas todas as partições de um dado número par.

Exemplo 3.1.12. *Consideremos o número par 720, cuja decomposição em fatores primos é dada por:*

$$720 = 2^4 \times 3^2 \times 5$$

Assim, temos que $A = \{2, 3, 5\}$, $B = \emptyset$, $p_n = 5$ e $p_{n+1} = 7$. Então,

$$a - p_{n+1}^2 = 720 - 7^2 = 671 \text{ e } a - 3 = 720 - 3 = 717$$

Obtemos então o intervalo $]671; 717]$, que contém os seguintes números primos:

p	673	677	683	691	701	709
$q = a - p$	47	43	37	29	19	11

Neste exemplo, verificou-se que $a - p$ originou sempre um número primo, já que a decomposição de 720 contém todos os primos entre 2 e 5, inclusive, o que resulta

do facto do conjunto B não ter elementos. Por outro lado, não foram determinadas todas as decomposições do número par 720, pois, por exemplo, 101 e 619, ambos números primos, formam uma partição de Goldbach.

No caso da decomposição de a conter todos os primos inferiores ou iguais a p_n , isto é $B = \emptyset$, então $a - 3, a - 5, \dots, a - p_n$ são divisíveis por 3, 5, ..., p_n , respetivamente, logo, basta definir $a - p_{n+1}$ como limite superior do intervalo definido na proposição 1.3.8. Nestas condições, $a - p$ não é divisível por p_i , com $i = 1, 2, \dots, n$, de acordo a proposição 3.1.3, e como $B = \emptyset$, temos que $a - p$ não tem fatores primos inferiores a $\sqrt{a - p} < p_{n+1}$, ou seja, $a - p$ é primo. Assim temos:

Corolário 3.1.13. *Dado um inteiro positivo a , cuja decomposição em fatores primos é dada por*

$$a = p_1^{s_1} p_2^{s_2} \dots p_n^{s_n}, \text{ com } (p_1 < p_2 < \dots < p_n) \text{ e } (s_i \in \mathbb{N})$$

se existir p primo com $p > p_n$ tal que $p \in]a - p_{n+1}^2, a - p_{n+1}]$ e $B = \emptyset$, então $a - p$ é primo.

Exemplo 3.1.14. *Consideremos o número par 3150, cuja decomposição em fatores primos é dada por:*

$$3150 = 2 \times 3^2 \times 5^2 \times 7$$

Assim, temos que $A = \{2, 3, 5, 7\}$, $B = \emptyset$, $p_n = 7$ e $p_{n+1} = 11$. Então,

$$a - p_{n+1}^2 = 3150 - 11^2 = 3029 \text{ e } a - p_{n+1} = 3150 - 11 = 3139$$

Obtemos então o intervalo $]3029; 3139]$, que contém os seguintes números primos:

$$\left(\begin{array}{cccccccccccc} 3037 & 3041 & 3049 & 3061 & 3067 & 3079 & 3083 & 3089 & 3109 & 3119 & 3121 & 3137 \end{array} \right)$$

Como $B = \emptyset$, temos que, com cada um dos primos anteriores é possível encontrar uma partição de Goldbach:

p	3037	3041	3049	3061	3067	3079	3083	3089	3109	3119	3121	3137
$q = a - p$	113	109	101	89	83	71	67	61	41	31	29	13

Também neste caso não foram encontradas todas as partições de Goldbach para 3150. Por exemplo, 1571 e 1579 formam uma partição de Goldbach para 3150.

Exemplo 3.1.15. *Consideremos o número par 390390, cuja decomposição em fatores primos é dada por:*

$$390390 = 2 \times 3 \times 5 \times 7 \times 11 \times 13^2$$

Assim, temos que $A = \{2, 3, 5, 7, 11, 13\}$, $B = \emptyset$, $p_n = 13$ e $p_{n+1} = 17$. Então,

$$a - p_{n+1}^2 = 390390 - 17^2 = 390101 \text{ e } a - p_{n+1} = 390390 - 17 = 390373$$

Obtemos assim o intervalo $]390101; 390373]$, em que, com cada primo desse intervalo é possível encontrar uma partição de Goldbach para 390390:

p	390107	390109	390113	390119	390157	390161	390191	390193
$q = a - p$	283	281	277	271	239	233	229	199
p	390199	390209	390211	390223	390263	390281	390289	390289
$q = a - p$	197	191	181	179	167	127	109	101
p	390307	390323	390343	390347	390353	390359	390367	390373
$q = a - p$	83	67	47	43	37	31	23	17

Novamente, não foram encontradas todas as partições de Goldbach para 390390. Por exemplo, 195193 e 195197 formam uma partição de Goldbach para 390390.

Nos exemplos apresentados anteriormente, o intervalo encontrado continha sempre números primos. Caso o intervalo não contenha números primos, torna-se necessário determinar outro intervalo, com maior amplitude, que contenha pelo menos um número primo p , tal que $a - p$ também seja primo.

Proposição 3.1.16. *Seja a um inteiro positivo par, cuja decomposição em fatores primos é dada por,*

$$a = p_1^{s_1} p_2^{s_2} \dots p_n^{s_n}, \text{ com } (p_1 < p_2 < \dots < p_n) \text{ e } (s_i \in \mathbb{N})$$

se existir p primo com $p > p_n$ tal que $p \in]a - p_{n+1+k}^2, a - p_{n+1+k}]$, $B = \emptyset$ e $p_{n+1}, p_{n+2} \dots p_{n+k}$ não dividem $a - p$, então $a - p$ é primo.

Demonstração.

Seja a um número par cuja decomposição em fatores primos é dada por $a = p_1^{s_1} p_2^{s_2} \dots p_n^{s_n}$, com $p_1 < p_2 < \dots < p_n$, em que a contém na sua decomposição todos os primos entre p_1 e p_n , inclusive. Seja também p primo com $p > p_n$ e $p \in]a - p_{n+1+k}^2, a - p_{n+1+k}]$. Assim, temos que:

$$p < a - p_{n+1+k} \Leftrightarrow a - p \geq p_{n+1+k}$$

Ou seja, $a - p$ não é nenhum dos p_i da decomposição de a . Por outro lado,

$$a - p < p_{n+1+k}^2 \Leftrightarrow \sqrt{a - p} < p_{n+1+k}$$

Ora, $a - p$ será primo se não possuir nenhum divisor primo inferior ou igual a $\sqrt{a - p}$. Como $\sqrt{a - p} < p_{n+1+k}$, os únicos divisores possíveis são $p_1, p_2, \dots, p_n, p_{n+1}, \dots, p_{n+k}$, mas $a - p$ não é divisível por p_1, p_2, \dots, p_n , e dado que $p_{n+1}, p_{n+2} \dots p_{n+k}$ não dividem $a - p$, conclui-se, que $a - p$ não possui nenhum divisor primo inferior a $\sqrt{a - p}$, logo, de acordo com a proposição 1.3.10, $a - p$ é necessariamente primo. \square

Exemplo 3.1.17. Consideremos o número par 2048, cuja decomposição em fatores primos é dada por:

$$2048 = 2^{11}$$

Assim, temos que $p_n = 2$ e $p_{n+1} = 3$. Então,

$$a - p_{n+1}^2 = 2048 - 3^2 = 2039 \text{ e } a - p_{n+1} = 2048 - 3 = 2045$$

Obtemos desta forma, o intervalo $]2039; 2045]$. Consultando a lista de primos, verifica-se que não existem primos no intervalo. Então, de acordo com a proposição anterior, definimos outro intervalo com maior amplitude que contenha números primos.

Por exemplo, façamos $k = 1$. Deste modo, obtemos o seguinte intervalo:

$$]a - p_{n+2}^2, a - p_{n+2}]$$

em que, $p_{n+1} = 3$ e $p_{n+2} = 5$. Deste modo, temos que:

$$2048 - 5^2 = 2023 \text{ e } 2048 - 5 = 2043$$

Então, o novo intervalo a considerar é $]2023, 2043]$. Neste intervalo já existem números primos: 2029 e 2039. Deste modo, as possíveis partições de 2048 são:

a	2029	2039
$a - p$	19	9

Das duas possíveis partições anteriores, retirando a partição em que $a - p$ é múltiplo de 3 (p_{n+1}), neste caso, 9, obtemos uma das partições de Goldbach para o número par 2048: (2029,19).

Exemplo 3.1.18. *Outros exemplos:*

$$a = 128 = 2^7,]119, 125]$$

$$a = 8192 = 2^{13},]8183, 8189]$$

$$a = 32768 = 2^{15},]32759, 32765]$$

$$a = 131072 = 2^{17},]131063, 131069]$$

$$a = 524288 = 2^{19},]524278, 524285]$$

Nenhum dos intervalos anteriores contém números primos, logo é necessário definir intervalos com maior amplitude até encontrar p , primo, que defina uma partição.

3.2 Número de partições existentes versus determinadas por este método

Como vimos nos exemplos da secção anterior, por vezes são determinadas todas as partições de Goldbach de um dado número par e, noutros casos, apenas algumas partições são determinadas.

Definição 3.2.1. *Seja a um inteiro positivo par, cuja decomposição em fatores primos é dada por:*

$$a = p_1^{s_1} p_2^{s_2} \dots p_n^{s_n}, \text{ com } (p_1 < p_2 < \dots < p_n) \text{ e } (s_i \in \mathbb{N})$$

Define-se como $g(a)$, o número de partições de Goldbach de a determinadas a partir do intervalo:

$$]a - p_{n+1}^2, a - 3]$$

Deste modo, recorrendo a exemplos anteriores, temos:

$$g(252) = 15 ; g(378) = 13 ; g(720) = 6 ; g(3150) = 12$$

Exemplo 3.2.2. *Consideremos o número par 300, cuja decomposição em fatores primos é dada por:*

$$300 = 2^2 \times 3 \times 5^2$$

Assim, $A = \{2, 3, 5\}$, $B = \emptyset$, $p_n = 5$ e $p_{n+1} = 7$.

O intervalo a considerar é $]251, 293]$. Desta forma, as partições encontradas pelo método são:

p	257	263	269	271	277	281	283	293
$q = a - p$	43	37	31	29	23	19	17	7

As partições em falta são:

p	151	173	191	193	197	199	211	227	229	239	241
q	149	127	109	107	103	101	89	73	71	61	59

Neste caso, o método apenas determina 8 das 19 partições, ou seja, $G(300) = 19$ e $g(300) = 8$, estando em falta as correspondentes ao intervalo $[150, 251]$.

É evidente, que se $a - p_{n+1}^2 > \frac{a}{2}$, não há a garantia de todas as partições de Goldbach serem encontradas, ficando em falta as correspondentes a $p \in]\frac{a}{2}, a - p_{n+1}^2]$, em que $a - p$ é primo. Assim, no caso de $a - p_{n+1}^2 < \frac{a}{2}$, basta neste método considerar o intervalo $]\frac{a}{2}, a - p_{n+1}^2]$, se $B = \emptyset$, e o intervalo $[\frac{a}{2}, a - 3]$, se $B \neq \emptyset$.

Exemplo 3.2.3. *Consideremos agora o número par 60, cuja decomposição em fatores primos é dada por:*

$$60 = 2^2 \times 3 \times 5$$

Assim, temos que $p_n = 5$ e $p_{n+1} = 7$. Então,

$$a - p_{n+1}^2 = 60 - 7^2 = 11 ; \frac{a}{2} = \frac{60}{2} = 30 \text{ e } a - 3 = 60 - 3 = 57$$

Como $B = \emptyset$ e $11 < 30$, os limites inferior e superior são dados por $\frac{a}{2} = 30$ e $a - p_{n+1} = 53$. Desta forma, o intervalo a considerar é $[30, 53]$, que determina todas as partições de Goldbach do número par 60:

p	31	37	41	43	47	53
$q = a - p$	29	23	19	17	13	7

Exemplo 3.2.4. *Consideremos agora o par 308, cuja decomposição em fatores primos é dada por:*

$$308 = 2^2 \times 7 \times 11$$

Assim, temos que $A = \{2, 7, 11\}$, $B = \{3, 5\}$, $p_n = 11$ e $p_{n+1} = 13$. Então,

$$a - p_{n+1}^2 = 308 - 13^2 = 139 ; \frac{a}{2} = \frac{308}{2} = 154 \text{ e } a - 3 = 308 - 3 = 305$$

Obtemos então o intervalo $[139, 305]$ e, como $139 < 154$, o intervalo a considerar é $[154, 305]$. Assim, recorrendo a uma lista de primos, as possíveis partições de 308 são:

p	157	163	167	173	179	181	191	193	197	199	211	223	227
$a - p$	151	145	141	135	129	127	117	115	111	109	97	85	81
p	229	233	239	241	251	257	263	269	271	277	281	283	293
$a - p$	79	75	69	67	57	51	45	39	37	31	27	25	15

Das possíveis partições de 308, retirando as partições em que $a - p$ é divisível por 3 ou 5, obtemos as partições de Goldbach do número par 308:

p	157	181	199	211	229	241	271	277
$q = a - p$	151	127	109	97	79	67	37	31

Exemplo 3.2.5. Consideremos o número par 30, cuja decomposição em fatores primos é dada por:

$$30 = 2 \times 3 \times 5$$

Assim, temos que $A = \{2, 3, 5\}$, $B = \emptyset$, $p_n = 5$ e $p_{n+1} = 7$. Então:

$$a - p_{n+1}^2 = 30 - 7^2 = -19 ; \frac{a}{2} = 15 \text{ e } a - p_{n+1} = 30 - 7 = 23$$

Como $a - p_{n+1}^2 < 15$, o intervalo a considerar é $[15, 23]$, que origina as seguintes partições:

p	17	19	23
$q = a - p$	13	11	7

Exemplo 3.2.6. Utilizando novamente como exemplo o número par 70, cuja decomposição é dada por:

$$70 = 2 \times 5 \times 7$$

Temos que $A = \{2, 5, 7\}$, $B = \{3\}$, $p_n = 7$ e $p_{n+1} = 11$. Então,

$$a - p_{n+1}^2 = 70 - 11^2 = -51 ; \frac{a}{2} = 35 \text{ e } a - 3 = 70 - 3 = 67$$

Como $a - p_{n+1}^2 < \frac{a}{2} = 35$, o intervalo a considerar é $[35, 67]$, que determina as seguintes possíveis partições:

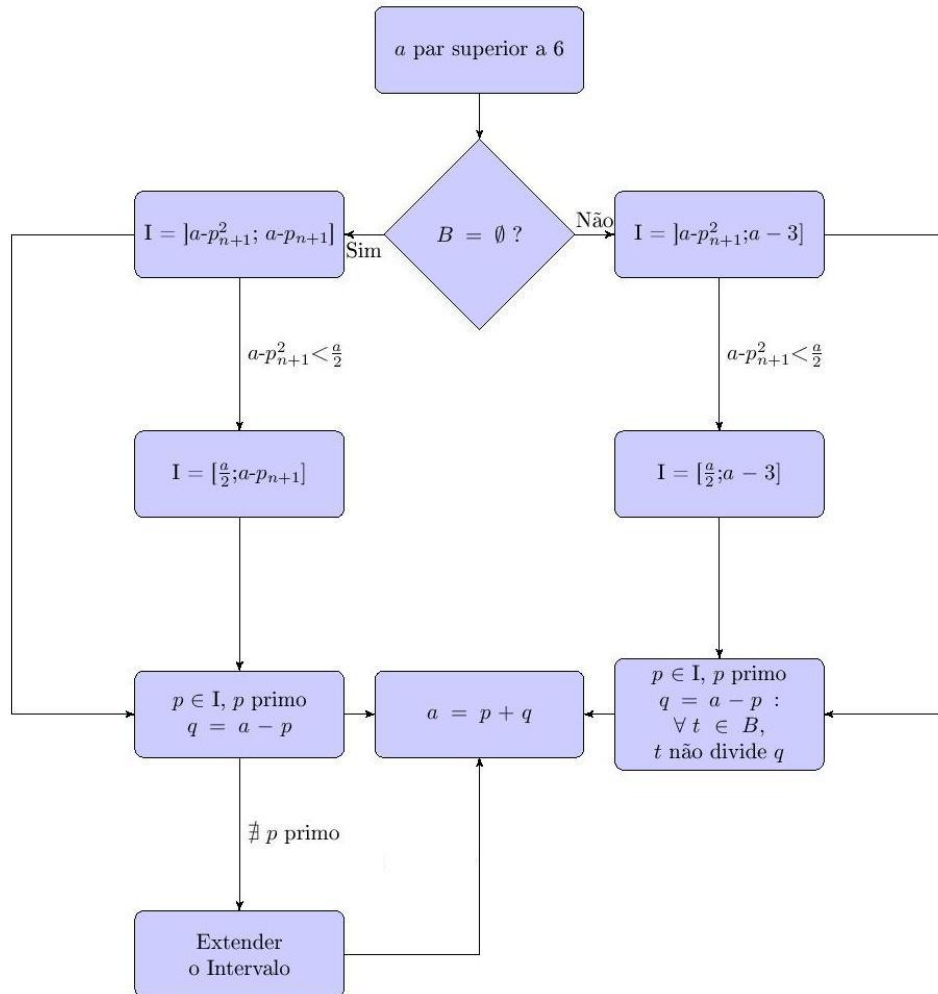
p	37	41	43	47	53	59	61	67
$a - p$	33	29	27	23	17	11	9	3

Das possíveis partições de 70, retirando as partições em que $a - p$ é divisível por 3, obtemos as partições de Goldbach do número par 70:

p	41	47	53	59	67
$q = a - p$	29	23	17	11	3

Como a Conjetura de Goldbach não foi provada, não é possível garantir que qualquer número par se pode escrever como soma de dois primos, no entanto, é possível indicar o número máximo de partições diferentes que um número par pode admitir. Com efeito, sendo $\pi(a)$ o número de primos inferiores ao inteiro a , temos que $\pi(a) - \#A$ representa o número de primos inferiores ao inteiro a e que não pertencem à sua decomposição, isto é, existem $\pi(a) - \#A$ primos candidatos a formarem uma partição de Goldbach de a , que podem formar, no máximo $\frac{\pi(a) - \#A}{2}$ diferentes partições de a .

Para finalizar a apresentação e exploração do método para determinar partições de Goldbach de um dado número par, desenvolvemos um fluxograma que permite sintetizar o algoritmo subjacente ao método:



Foi criada uma aplicação em Java, recorrendo ao algoritmo anterior, que permite determinar partições de Goldbach de um dado número par. O código fonte associado pode ser consultado no Apêndice B, enquanto a aplicação pode ser executada a partir do *cd* incluído.

3.3 Propriedades dos inteiros e relação com a Conjetura de Goldbach

Nesta secção final apresentamos algumas propriedades dos pares de primos que formam partições de Goldbach de um dado número par.

Exemplo 3.3.1. *Consideremos o número par 132 cuja decomposição em fatores é dada por:*

$$132 = 2^2 \times 3 \times 11$$

Temos que $A = \{2, 3, 11\}$ e $B = \{5, 7\}$. Consideremos, por exemplo, $p = 5$. Assim, temos que:

$$5^2 \equiv 25 \pmod{132}$$

Recorrendo a uma folha de cálculo, obtém-se também,

$$17^2 \equiv 25 \pmod{132} ; 61^2 \equiv 25 \pmod{132} ; 71^2 \equiv 25 \pmod{132}$$

$$83^2 \equiv 25 \pmod{132} ; 127^2 \equiv 25 \pmod{132}$$

Deste modo, o conjunto P_{25} , de todos os primos menores que 132, cujo quadrado tem resto 25 módulo 132 é:

$$P_{25} = \{5, 17, 61, 71, 83, 127\}$$

Pelo mesmo raciocínio anterior, e dado que $7^2 \equiv 49 \pmod{132}$ obtemos:

$$29^2 \equiv 49 \pmod{132} ; 37^2 \equiv 49 \pmod{132} ; 59^2 \equiv 49 \pmod{132}$$

$$73^2 \equiv 49 \pmod{132} ; 103^2 \equiv 49 \pmod{132}$$

ou seja,

$$P_{49} = \{7, 29, 37, 59, 73, 103\}$$

Analogamente,

$$P_{37} = \{13, 31, 53, 79, 97, 101\}$$

$$P_{97} = \{19, 41, 47, 107, 113\}$$

$$P_1 = \{23, 43, 67, 89, 109, 131\}$$

Vamos recordar as partições de Goldbach de 132, encontradas anteriormente:

p	5	19	23	29	31	43	53	59	61
$q = a - p$	127	113	109	103	101	89	79	73	71

Podemos observar que todos os primos menores que 132 estão incluídos num dos conjuntos anteriores, mas mais relevante, que os pares de primos que compõem qualquer uma das nove partições de Goldbach pertencem ao mesmo conjunto P_r , com r inteiro positivo.

Definição 3.3.2. *Sejam a um número par superior a 4 e p primo, com $p < a$ e tal que $p \notin A$.*

Seja ainda r o resto da divisão de p^2 por a , ou seja, $p^2 \equiv r \pmod{a}$.

Define-se por P_r o conjunto:

$$P_r = \{x \text{ primo: } x < a \wedge x^2 \equiv r \pmod{a}\}$$

Exemplo 3.3.3. *Consideremos o número par 90 cuja decomposição em fatores é dada por:*

$$90 = 2 \times 3^2 \times 5$$

Temos assim que $A = \{2, 3, 5\}$. Recorrendo a uma folha de cálculo, obtemos:

$$P_{49} = \{7, 43, 47, 83\} ; P_{31} = \{11, 29, 61, 79\} ; P_{79} = \{13, 23, 67\}$$

$$P_{19} = \{17, 37, 53, 73\} ; P_1 = \{19, 71, 89\} ; P_{61} = \{31, 41, 59\}$$

Apartir dos conjuntos anteriores podemos determinar as partições de Goldbach de 90. Com efeito, de:

- P_{49} , determinamos duas partições: (7,83) e (43,47);
- P_{31} , determinamos duas partições: (11,79) e (29,61);
- P_{79} , determinamos uma partição: (23,67);
- P_{19} , determinamos duas partições: (17,73) e (37,53);
- P_1 , determinamos uma partição: (19,71);

- P_{61} , determinamos uma partição: (31,59).

Podemos agora refletir, em primeiro lugar, se é possível criar uma partição com dois primos de conjuntos diferentes, ou se, pelo contrário, é apenas possível criar partições com primos do mesmo conjunto.

Com efeito, se p e q formam uma partição de Goldbach de a , isto é, $a = p + q$, pela proposição 1.1.20, temos que $p^2 \equiv q^2 \pmod{a}$, ou seja, p e q pertencem ao mesmo conjunto P_r , para algum inteiro positivo r . Reciprocamente, se $p^2 \not\equiv q^2 \pmod{a}$, então p e q não formam uma partição de Goldbach de a .

Mas mais importante, podemos refletir se será sempre possível encontrar, em algum dos conjuntos P_r , dois primos cuja soma seja a , ou mesmo, encontrar uma ou mais partições de Goldbach em todos os conjuntos P_r . Dificilmente será possível dar uma resposta afirmativa, por mais exemplos que o comprovem, simplesmente porque, seria o mesmo que provar a Conjetura de Goldbach. Deste modo, podemos formular a Conjetura de Goldbach de uma forma equivalente:

Seja a um número par superior a 6.

Então existe um inteiro $r < a$, tal que P_r , definido anteriormente, contém pelo menos uma partição de Goldbach.

Como vimos no capítulo 1, uma das aplicações dos números primos são os sistemas de identificação modulares. Nesse sentido, e dado que a Conjetura de Goldbach tem na sua génese os números primos, desenvolvemos uma tentativa de criação de um sistema que permite criar um número de identificação, utilizando a Conjetura de Goldbach, cujo algoritmo é apresentado a seguir:

1. Escolher a par, superior a 6;
2. Determinar uma das partições de Goldbach de a , dada por $a = p + q$;
3. Determinar j , algarismo de teste, dado por $j = \frac{a}{2} - p = q - \frac{a}{2}$.

O número de identificação criado é da forma:

$$a - p - q - j$$

Exemplo 3.3.4. *Dado o número par 100, uma das partições é $47+53$. Assim sendo, temos que $j = 50 - 47 = 53 - 50 = 3$.*

Então, o número de identificação criado é:

$$100 - 47 - 53 - 3$$

Note-se que se o valor de a for alterado, o algarismo de teste também teria de ser alterado. Mais ainda, se o valor de p ou q forem alterados, a soma já não seria a .

Em relação a este sistema podemos também determinar quais são os tipos de erros que são detetados. Facilmente se verifica que os erros singulares são detetados, já que ao alteramos um algarismo, o valor de j também teria de ser alterado. Mesmo as trocas de algarismos, quer adjacentes ou intercaladas, e os erros gémeos, são detetados pela mesma razão. Por exemplo, recorrendo ao número de identificação já criado,

$$100 - 47 - 53 - 3$$

podemos verificar que se um algarimo for alterado,

$$100 - 47 - 63 - 3$$

temos que,

$$47 + 63 \neq 100 \text{ e } 63 - 50 \neq 3$$

ou seja, o erro é detetado. Relativamente a uma troca de algarismos adjacentes,

$$100 - 74 - 53 - 3 \text{ ou } 104 - 07 - 53 - 3$$

temos que,

$$74 + 53 \neq 100 \text{ e } 50 - 74 \neq 3$$

ou

$$07 + 53 \neq 104 \text{ e } 52 - 07 \neq 3$$

Em relação aos erros gémeos,

$$100 - 47 - 56 - 6 \text{ ou } 122 - 47 - 53 - 3$$

temos que,

$$47 + 56 \neq 100 \text{ e } 50 - 47 \neq 6$$

ou

$$47 + 53 \neq 122 \text{ e } 61 - 47 \neq 3$$

Por fim, em relação aos erros aleatórios e fonéticos, que podem ocorrer das mais diversas formas, também estes serão, na sua generalidade, detetados pelo algarismo de teste. A única forma de os erros não serem detetados, seria alterar vários algarismos, o que, supondo que se pretende transmitir corretamente o número de identificação, é muito pouco provável.

Exemplo 3.3.5. *Dado a número par 120, uma das partições possíveis é $59 + 61$.*

Assim sendo, temos que $j = 60 - 59 = 61 - 60 = 1$. Então, o número de identificação criado é:

$$120 - 59 - 61 - 1$$

Uma das formas de criar um erro não detetável é:

$$122 - 61 - 61 - 0$$

o que envolve cometer 4 erros singulares, sendo assim, um caso com uma probabilidade de ocorrer quase nula.

Paralelamente ao desenvolvimento deste trabalho foi criado uma aplicação que possibilita a criação de um número de identificação. O código fonte associado pode ser consultado no Apêndice B, enquanto a aplicação em Java pode ser executada a partir do *cd* incluído.

Considerações finais

“Deus criou os números, o resto é obra do homem.”

Kronecker

Concluimos analisando alguns aspetos relevantes que decorrem do trabalho desenvolvido durante a elaboração da dissertação e que evidenciam a simplicidade, bem como a complexidade deste problema em aberto.

Simplicidade

Como pudemos observar durante a exploração da Conjetura de Goldbach, esta surge como um problema facilmente verificável, até para alguém pouco experiente, ou mesmo sem grandes conhecimentos matemáticos. Euler, em resposta a Goldbach, afirmava que a conjectura era seguramente verdadeira, no entanto os seus esforços tinham sido infrutíferos. Foi certamente o primeiro, mesmo antes de Goldbach, a aperceber-se da simplicidade inerente à conjectura, e provavelmente terá pensado que Goldbach ou outro matemático conseguiria, mais cedo ou mais tarde, provar a conjectura, devido à aparente facilidade em encontrar uma partição de Goldbach para um dado número par.

Complexidade

A Conjetura de Goldbach, apresentada há 271 anos, permanece por demonstrar, na minha opinião, e certamente na opinião de quem já contactou com a conjectura, por várias razões, sendo a mais relevante a forma como os números primos se distribuem, pela sua imprevisibilidade e irregularidade. Por outro lado, cada número par, e principalmente os seus fatores primos, também dificultam qualquer tentativa de criar um método, um algoritmo, ou forma de abordar o problema,

já que, por exemplo, pares consecutivos têm apenas o fator primo 2 em comum e, por outro lado, números pares muito afastados podem ter os mesmos fatores primos.

A Teoria dos Números é talvez o maior exemplo da interdisciplinaridade na Matemática. A existência de questões em aberto representa um acréscimo de conhecimentos resultante das investigações desses problemas. Os resultados acabam, eventualmente, por ter aplicação na resolução de problemas matemáticos que diariamente apresentamos aos nossos alunos, por mais simples que seja o enunciado. Deste modo, os fundamentos da Teoria dos Números, alguns dos quais abordados nesta dissertação, como a teoria da divisibilidade, as congruências, a demonstração por indução, entre outros não abordados mas também importantes, como a lógica ou a definição por recursividade, representam uma parte fundamental de toda a Matemática que permite justificar as demonstrações, os métodos, as fórmulas, e em fim último, formar a base de toda a Matemática.

Porque fazer Matemática não é só resolver problemas, é muito mais do que isso. Fazer Matemática, envolve um processo de criação, execução, justificação, e *feedback* que permite depois generalizar e desenvolver ferramentas que possibilitam resolver outros problemas mais complexos.

Referências Bibliográficas

- Bernard Farley (2005). *Two Approaches to Proving Goldbach's Conjecture*. EUA. <http://www.math.vt.edu/people/plinnell/Ugresearch/farley.pdf>, consultado em: 27, Abril, 2013.
- Books, H. (2011). *Conjectures about Prime Numbers*. Hephaestus Books, Nova Iorque, EUA.
- Bressoud, D. M. (1989). *Factorization and Primality Testing*. Springer-Verlag, New York, EUA.
- Chris K. Caldwell (2012). *The Prime Pages*. Universidade de Tennessee, EUA. <http://primes.utm.edu/>, consultado em: 27, Abril, 2013.
- Crandall, Richard; Pomerance, C. (2005). *Prime Numbers, A Computational Perspective*. Springer Science, Nova Iorque, EUA.
- E. Robertson (2006). *Christian's Goldbach biographie*. Universidade de St Andrews, Escócia. <http://www-history.mcs.st-andrews.ac.uk/Biographies/Goldbach.html>, consultado em: 27, Abril, 2013.
- Filho, E. d. A. (1981). *Teoria Elementar dos Números*. Livraria Nobel, São Paulo, Brasil.
- Filho, E. d. A. (1986). *Aritmética dos Inteiros*. Livraria Nobel, São Paulo, Brasil.
- Fine, Benjamin; Rosenberger, G. (2007). *Number Theory, An Introduction via the Distribution of Primes*. Birkhauser Publishing, Boston, EUA.
- Fliegel, Henry F.; Robertson, D. S. (1989). *Goldbach's Comet: the numbers related to Goldbach's Conjecture*. Journal of Recreational Mathematics, volume 21, EUA.

- Gracián, E. (2010). *Os números primos, Um longo caminho para o infinito*. RBA Coleccionables, Espanha.
- Kent Slinker (2005). *A model of Goldbach's conjecture that all even numbers greater than 4 are the sum of two odd primes*. Pima Community College, Tucson, EUA. <http://arxiv.org/vc/arxiv/papers/0712/0712.2381v2.pdf>, consultado em: 27, Abril, 2013.
- Nogueira, J. E. & et. al. (2004). *Contar e Fazer Contas, Uma Introdução à teoria Dos Números*. SPM/Grádiva-publicações, Lisboa, Portugal.
- Ribenboim, P. (1995). *The New Book of Prime Number Records*. Springer-Verlag, Nova Iorque, EUA.
- Roger Ellman (2000). *A Proof of Goldbach's conjecture*. Santa Rosa, EUA. <http://arxiv.org/ftp/math/papers/0005/0005185.pdf>, consultado em: 27, Abril, 2013.
- Woon, M. S. (2000). *On Partitions of Goldbach's Conjecture*. Cambridge University, Reino Unido. <http://arxiv.org/pdf/math/0010027v1.pdf>, consultado em: 27, Abril, 2013.
- Yandell, B. H. (2002). *The Honors Class: Hilbert's Problems and Their Solvers*. A. K. Peters Ltd, Oxfordshire, Reino Unido.
- Yuan, W. (2002). *The Goldbach Conjecture*. World scientific, Singapore.

Apêndice A

Lista de números primos menores que 10000

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103
107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211
223 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307 311 313 317 331
337 347 349 353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449
457 461 463 467 479 487 491 499 503 509 521 523 541 547 557 563 569 571 577 587
593 599 601 607 613 617 619 631 641 643 647 653 659 661 673 677 683 691 701 709
719 727 733 739 743 751 757 761 769 773 787 797 809 811 821 823 827 829 839 853
857 859 863 877 881 883 887 907 911 919 929 937 941 947 953 967 971 977 983 991
997 1009 1013 1019 1021 1031 1033 1039 1049 1051 1061 1063 1069 1087 1091 1093
1097 1103 1109 1117 1123 1129 1151 1153 1163 1171 1181 1187 1193 1201 1213 1217
1223 1229 1231 1237 1249 1259 1277 1279 1283 1289 1291 1297 1301 1303 1307 1319
1321 1327 1361 1367 1373 1381 1399 1409 1423 1427 1429 1433 1439 1447 1451 1453
1459 1471 1481 1483 1487 1489 1493 1499 1511 1523 1531 1543 1549 1553 1559 1567
1571 1579 1583 1597 1601 1607 1609 1613 1619 1621 1627 1637 1657 1663 1667 1669
1693 1697 1699 1709 1721 1723 1733 1741 1747 1753 1759 1777 1783 1787 1789 1801
1811 1823 1831 1847 1861 1867 1871 1873 1877 1879 1889 1901 1907 1913 1931 1933
1949 1951 1973 1979 1987 1993 1997 1999 2003 2011 2017 2027 2029 2039 2053 2063
2069 2081 2083 2087 2089 2099 2111 2113 2129 2131 2137 2141 2143 2153 2161 2179
2203 2207 2213 2221 2237 2239 2243 2251 2267 2269 2273 2281 2287 2293 2297 2309

2311 2333 2339 2341 2347 2351 2357 2371 2377 2381 2383 2389 2393 2399 2411 2417
2423 2437 2441 2447 2459 2467 2473 2477 2503 2521 2531 2539 2543 2549 2551 2557
2579 2591 2593 2609 2617 2621 2633 2647 2657 2659 2663 2671 2677 2683 2687 2689
2693 2699 2707 2711 2713 2719 2729 2731 2741 2749 2753 2767 2777 2789 2791 2797
2801 2803 2819 2833 2837 2843 2851 2857 2861 2879 2887 2897 2903 2909 2917 2927
2939 2953 2957 2963 2969 2971 2999 3001 3011 3019 3023 3037 3041 3049 3061 3067
3079 3083 3089 3109 3119 3121 3137 3163 3167 3169 3181 3187 3191 3203 3209 3217
3221 3229 3251 3253 3257 3259 3271 3299 3301 3307 3313 3319 3323 3329 3331 3343
3347 3359 3361 3371 3373 3389 3391 3407 3413 3433 3449 3457 3461 3463 3467 3469
3491 3499 3511 3517 3527 3529 3533 3539 3541 3547 3557 3559 3571 3581 3583 3593
3607 3613 3617 3623 3631 3637 3643 3659 3671 3673 3677 3691 3697 3701 3709 3719
3727 3733 3739 3761 3767 3769 3779 3793 3797 3803 3821 3823 3833 3847 3851 3853
3863 3877 3881 3889 3907 3911 3917 3919 3923 3929 3931 3943 3947 3967 3989 4001
4003 4007 4013 4019 4021 4027 4049 4051 4057 4073 4079 4091 4093 4099 4111 4127
4129 4133 4139 4153 4157 4159 4177 4201 4211 4217 4219 4229 4231 4241 4243 4253
4259 4261 4271 4273 4283 4289 4297 4327 4337 4339 4349 4357 4363 4373 4391 4397
4409 4421 4423 4441 4447 4451 4457 4463 4481 4483 4493 4507 4513 4517 4519 4523
4547 4549 4561 4567 4583 4591 4597 4603 4621 4637 4639 4643 4649 4651 4657 4663
4673 4679 4691 4703 4721 4723 4729 4733 4751 4759 4783 4787 4789 4793 4799 4801
4813 4817 4831 4861 4871 4877 4889 4903 4909 4919 4931 4933 4937 4943 4951 4957
4967 4969 4973 4987 4993 4999 5003 5009 5011 5021 5023 5039 5051 5059 5077 5081
5087 5099 5101 5107 5113 5119 5147 5153 5167 5171 5179 5189 5197 5209 5227 5231
5233 5237 5261 5273 5279 5281 5297 5303 5309 5323 5333 5347 5351 5381 5387 5393
5399 5407 5413 5417 5419 5431 5437 5441 5443 5449 5471 5477 5479 5483 5501 5503
5507 5519 5521 5527 5531 5557 5563 5569 5573 5581 5591 5623 5639 5641 5647 5651
5653 5657 5659 5669 5683 5689 5693 5701 5711 5717 5737 5741 5743 5749 5779 5783
5791 5801 5807 5813 5821 5827 5839 5843 5849 5851 5857 5861 5867 5869 5879 5881
5897 5903 5923 5927 5939 5953 5981 5987 6007 6011 6029 6037 6043 6047 6053 6067
6073 6079 6089 6091 6101 6113 6121 6131 6133 6143 6151 6163 6173 6197 6199 6203
6211 6217 6221 6229 6247 6257 6263 6269 6271 6277 6287 6299 6301 6311 6317 6323
6329 6337 6343 6353 6359 6361 6367 6373 6379 6389 6397 6421 6427 6449 6451 6469
6473 6481 6491 6521 6529 6547 6551 6553 6563 6569 6571 6577 6581 6599 6607 6619
6637 6653 6659 6661 6673 6679 6689 6691 6701 6703 6709 6719 6733 6737 6761 6763

6779 6781 6791 6793 6803 6823 6827 6829 6833 6841 6857 6863 6869 6871 6883 6899
6907 6911 6917 6947 6949 6959 6961 6967 6971 6977 6983 6991 6997 7001 7013 7019
7027 7039 7043 7057 7069 7079 7103 7109 7121 7127 7129 7151 7159 7177 7187 7193
7207 7211 7213 7219 7229 7237 7243 7247 7253 7283 7297 7307 7309 7321 7331 7333
7349 7351 7369 7393 7411 7417 7433 7451 7457 7459 7477 7481 7487 7489 7499 7507
7517 7523 7529 7537 7541 7547 7549 7559 7561 7573 7577 7583 7589 7591 7603 7607
7621 7639 7643 7649 7669 7673 7681 7687 7691 7699 7703 7717 7723 7727 7741 7753
7757 7759 7789 7793 7817 7823 7829 7841 7853 7867 7873 7877 7879 7883 7901 7907
7919 7927 7933 7937 7949 7951 7963 7993 8009 8011 8017 8039 8053 8059 8069 8081
8087 8089 8093 8101 8111 8117 8123 8147 8161 8167 8171 8179 8191 8209 8219 8221
8231 8233 8237 8243 8263 8269 8273 8287 8291 8293 8297 8311 8317 8329 8353 8363
8369 8377 8387 8389 8419 8423 8429 8431 8443 8447 8461 8467 8501 8513 8521 8527
8537 8539 8543 8563 8573 8581 8597 8599 8609 8623 8627 8629 8641 8647 8663 8669
8677 8681 8689 8693 8699 8707 8713 8719 8731 8737 8741 8747 8753 8761 8779 8783
8803 8807 8819 8821 8831 8837 8839 8849 8861 8863 8867 8887 8893 8923 8929 8933
8941 8951 8963 8969 8971 8999 9001 9007 9011 9013 9029 9041 9043 9049 9059 9067
9091 9103 9109 9127 9133 9137 9151 9157 9161 9173 9181 9187 9199 9203 9209 9221
9227 9239 9241 9257 9277 9281 9283 9293 9311 9319 9323 9337 9341 9343 9349 9371
9377 9391 9397 9403 9413 9419 9421 9431 9433 9437 9439 9461 9463 9467 9473 9479
9491 9497 9511 9521 9533 9539 9547 9551 9587 9601 9613 9619 9623 9629 9631 9643
9649 9661 9677 9679 9689 9697 9719 9721 9733 9739 9743 9749 9767 9769 9781 9787
9791 9803 9811 9817 9829 9833 9839 9851 9857 9859 9871 9883 9887 9901 9907 9923
9929 9931 9941 9949 9967 9973

Apêndice B

Código fonte

Applet - Conjetura de Goldbach

```
1 package Conjetura_Goldbach;
2
3 import java.awt.BorderLayout;
4 import java.awt.CardLayout;
5 import java.awt.Color;
6 import java.awt.Container;
7 import java.awt.Dimension;
8 import java.awt.Font;
9 import java.awt.event.ActionEvent;
10 import java.awt.event.ActionListener;
11 import java.io.File;
12 import java.io.FileNotFoundException;
13 import java.util.ArrayList;
14 import java.util.Scanner;
15 import javax.swing.ImageIcon;
16 import javax.swing.JButton;
17 import javax.swing.JFileChooser;
18 import javax.swing.JFrame;
19 import javax.swing.JLabel;
20 import javax.swing.JMenu;
21 import javax.swing.JMenuBar;
22 import javax.swing.JMenuItem;
23 import javax.swing.JOptionPane;
24 import javax.swing.JPanel;
25 import javax.swing.JScrollPane;
26 import javax.swing.JTextArea;
27 import javax.swing.JTextField;
28
29 /**
30  * Created on Abril 2013
31  *
32  */
33
34 /**
35  * @author José Emanuel Sousa
36  * Mestrado em Matemática para Professores
37  */
38
39 public class Conjetura_Goldbach extends JFrame {
40
41     private Container cp;
42     private TestePrimoPanel testePanel;
```

```

43     private DecomposicaoPanel decomporPanel;
44     private ParticaoPanel particaoPanel;
45     private SobrePanel sobrePanel;
46     private JPanel cardPanel;
47     private JMenu[] menus = {
48         new JMenu("Testar Primo"), new JMenu("Fatorização"), new JMenu("Partição"), new JMenu("Sobre")
49     };
50     private JMenuItem[] items = {
51         new JMenuItem("Testar"), new JMenuItem("Fatorizar"), new JMenuItem("Particionar"), new JMenuItem("Sobre")
52     };
53     private ActionListener action = new ActionListener() {
54         @Override
55         public void actionPerformed(ActionEvent e) {
56             JMenuItem target = (JMenuItem) e.getSource();
57             if (target.getActionCommand().equals("Testar")) {
58                 update("teste");
59             } else if (target.getActionCommand().equals("Fatorizar")) {
60                 update("fatorizar");
61             } else if (target.getActionCommand().equals("Sobre")) {
62                 update("sobre");
63             } else {
64                 update("particao");
65             }
66         }
67     };
68
69     public Conjetura_Goldbach() {
70         super("Conjetura de Goldbach");
71         for (int i = 0; i < items.length; i++) {
72             items[i].addActionListener(action);
73             menus[i % 4].add(items[i]);
74         }
75         JMenuBar mb = new JMenuBar();
76         for (int i = 0; i < menus.length; i++) {
77             mb.add(menus[i]);
78         }
79         setJMenuBar(mb);
80
81         testePanel = new TestePrimoPanel();
82         decomporPanel = new DecomposicaoPanel();
83         particaoPanel = new ParticaoPanel();
84         sobrePanel = new SobrePanel();
85
86         testePanel.setBackground(Color.getHSBColor(0.0f, 0.0f, 0.95f));
87         decomporPanel.setBackground(Color.getHSBColor(0.0f, 0.0f, 0.95f));
88         particaoPanel.setBackground(Color.getHSBColor(0.0f, 0.0f, 0.95f));
89         sobrePanel.setBackground(Color.getHSBColor(0.0f, 0.0f, 0.95f));
90
91         cardPanel = new JPanel();
92         cardPanel.setLayout(new CardLayout());
93
94         cp = getContentPane();
95         cp.setLayout(new BorderLayout());
96         cardPanel.add(testePanel, "teste");
97         cardPanel.add(decomporPanel, "fatorizar");
98         cardPanel.add(particaoPanel, "particao");
99         cardPanel.add(sobrePanel, "sobre");
100
101         cp.add(cardPanel, BorderLayout.CENTER);
102         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
103         setSize(new Dimension(500, 700));
104         setResizable(false);
105         setVisible(true);
106     }
107
108     public void update(String txt) {
109         CardLayout card = (CardLayout) cardPanel.getLayout();
110         card.show(cardPanel, txt);
111     }

```



```

112
113 class TestePrimoPanel extends JPanel implements ActionListener {
114     JLabel numero = new JLabel("Introduza um número a testar:");
115     JTextArea texto = new JTextArea(6, 20);
116     JTextField caixaTexto = new JTextField();
117     JButton botao = new JButton("Testar");
118     JButton Limpar = new JButton("Limpar");
119
120     public TestePrimoPanel() {
121         setLayout(null);
122         numero.setBounds(10, 10, 450, 20);
123         caixaTexto.setBounds(10, 40, 350, 20);
124         botao.setBounds(380, 40, 90, 20);
125         Limpar.setBounds(380, 600, 90, 20);
126         texto.setBounds(10, 65, 550, 20);
127         botao.addActionListener(this);
128         texto.setEditable(false);
129         Limpar.setToolTipText("Limpar");
130         texto.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
131         Font font = new Font("Arial", Font.BOLD, 12);
132         texto.setFont(font);
133
134         add(numero);
135         add(caixaTexto);
136         add(botao);
137         add(Limpar);
138         add(texto);
139         setVisible(true);
140
141         Limpar.addActionListener(new ActionListener(){
142
143             public void actionPerformed ( ActionEvent event){
144                 caixaTexto.setText("");
145                 texto.setText("");
146             }
147         });
148     }
149
150     @Override
151     public void actionPerformed(ActionEvent e) {
152         ArrayList<Integer> Primos = new ArrayList<Integer>();
153         int num, i=0, k;
154         boolean flag = false;
155
156         criarListaPrimos(Primos);
157
158         try {
159             String inputNum = caixaTexto.getText();
160             num = (int) Integer.parseInt(inputNum);
161
162             if(num<=1 || num>1299830)
163                 JOptionPane.showMessageDialog(this, "Introduziu um valor inválido");
164             else {
165                 k = (int) (Math.sqrt(num));
166                 while(Primos.get(i)<=k && flag==false)
167                 {
168                     if(num%Primos.get(i) == 0)
169                     {
170                         texto.setText(num+" não é primo porque é divisível por "+Primos.get(i));
171                         flag = true;
172                     } i++;
173                 }
174                 if(flag==false)
175                     texto.setText(num+" é primo.");
176             }
177         } catch (Exception ex) {
178             JOptionPane.showMessageDialog(this, "Introduziu um valor inválido");
179         }
180     }

```

```

181     }
182
183     class DecomposicaoPanel extends JPanel implements ActionListener {
184         JLabel numero = new JLabel("Introduza um número:");
185         JTextArea texto = new JTextArea(6, 20);
186         JTextField caixaTexto = new JTextField();
187         JButton botao = new JButton("Fatorizar");
188         JButton Limpar = new JButton("Limpar");
189
190         public DecomposicaoPanel() {
191             setLayout(null);
192             numero.setBounds(10, 10, 240, 20);
193             caixaTexto.setBounds(10, 40, 350, 20);
194             botao.setBounds(380, 40, 100, 20);
195             Limpar.setBounds(380, 600, 90, 20);
196             Limpar.setBounds(380, 600, 90, 20);
197             texto.setBounds(10, 65, 500, 500);
198             texto.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
199             botao.addActionListener(this);
200             Limpar.setToolTipText("Limpar");
201             texto.setEditable(false);
202             Font font = new Font("Arial", Font.BOLD, 12);
203             texto.setFont(font);
204
205             add(numero);
206             add(caixaTexto);
207             add(botao);
208             add(Limpar);
209             add(texto);
210             setVisible(true);
211
212             Limpar.addActionListener(new ActionListener(){
213                 public void actionPerformed ( ActionEvent event){
214                     caixaTexto.setText("");
215                     texto.setText("");
216                 }
217             });
218         }
219
220         @Override
221         public void actionPerformed(ActionEvent e) {
222             int num, k, i=0;
223             ArrayList<Integer> fatores = new ArrayList<Integer>();
224             ArrayList<Integer> Primos = new ArrayList<Integer>();
225             criarListaPrimos(Primos);
226
227             try {
228                 String inputNum = caixaTexto.getText();
229                 int aux = (int) Integer.parseInt(inputNum);
230                 num = aux;
231                 k = (int) num/2;
232
233                 while(Primos.get(i)<=k && num!=1)
234                 {
235                     if((num%Primos.get(i))==0)
236                     {
237                         if(!fatores.contains(Primos.get(i)))
238                             fatores.add(Primos.get(i));
239                         num = num/Primos.get(i);
240                     }
241                     if(num%Primos.get(i)!=0)
242                     {
243                         i++;
244                     }
245                 }
246                 if (fatores.size() == 0)
247                 {
248                     texto.append("\n 0 número introduzido é primo");
249                 }else

```

```

250         {
251             texto.setText("Os fatores de "+aux+" são\n");
252             for(Integer b : fatores)
253                 texto.append(b+"\n");
254         }
255     } catch (Exception ex) {
256         JOptionPane.showMessageDialog(this, "Introduziu um valor inválido");
257     }
258 }
259 }
260
261 class ParticaoPanel extends JPanel implements ActionListener {
262
263     JLabel numero = new JLabel("Introduza um número par superior a 6 (seis):");
264     JTextArea texto = new JTextArea(6, 50);
265     JTextField caixaTexto = new JTextField();
266     JButton botao = new JButton("Particionar");
267     JButton Limpar = new JButton("Limpar");
268     JScrollPane sp;
269
270     public ParticaoPanel() {
271         setLayout(null);
272         numero.setBounds(10, 10, 300, 20);
273         caixaTexto.setBounds(10, 40, 350, 20);
274         botao.setBounds(380, 40, 100, 20);
275         Limpar.setBounds(380, 600, 90, 20);
276
277         Limpar.setToolTipText("Limpar");
278         botao.addActionListener(this);
279         texto.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
280         texto.setEditable(false);
281         Font font = new Font("Arial", Font.BOLD, 12);
282         texto.setFont(font);
283
284         sp = new JScrollPane(texto);
285         sp.setBounds(10, 75, 480, 480);
286         sp.setBorder(javax.swing.BorderFactory.createEmptyBorder());
287         sp.setHorizontalScrollBar(null);
288         sp.setHorizontalScrollBarPolicy(sp.HORIZONTAL_SCROLLBAR_NEVER);
289
290         add(numero);
291         add(caixaTexto);
292         add(botao);
293         add(Limpar);
294         add(sp);
295         setVisible(true);
296
297         Limpar.addActionListener(new ActionListener(){
298             public void actionPerformed ( ActionEvent event){
299                 caixaTexto.setText("");
300                 texto.setText("");
301             }
302         });
303     }
304
305     @Override
306     public void actionPerformed(ActionEvent e) {
307         ArrayList<Integer> A = new ArrayList<Integer>();
308         ArrayList<Integer> B = new ArrayList<Integer>();
309         ArrayList<Integer> Primos = new ArrayList<Integer>();
310         ArrayList<Integer> Intervalo = new ArrayList<Integer>();
311         ArrayList<Integer> Duplicados = new ArrayList<Integer>();
312
313         int ultimoPrimoA, primoSuperior=0, primeiroPrimoIntervalo=0, limiteInferior = 0, limiteSuperior;
314         int posAux, aux, inputNumber = 0, count=0;
315         boolean neprimos = true, stop;
316
317         try {
318             String inputNum = caixaTexto.getText();

```

```

319         inputNumber = (int) Integer.parseInt(inputNum);
320
321     if(inputNumber<=6 || inputNumber%2!=0 || inputNumber>1299830)
322         JOptionPane.showMessageDialog(this, "Introduziu um valor inválido");
323
324     else {
325         criarListaPrimos(Primos);
326         criarListaA(inputNumber, A, Primos);
327         criarListaB(A, B, Primos);
328
329         if((A.size() == 2) && (A.get(A.size()-1)*2 == inputNumber))
330         {
331             texto.setText("Como "+inputNumber+" é da forma 2p, então: ");
332             texto.append("\n"+inputNumber+" = "+A.get(A.size()-1)+" + "+A.get(A.size()-1));
333         }
334         ultimoPrimoA = A.get(A.size()-1);
335
336         for(int i=0; i<Primos.size(); i++)
337         {
338             if(Primos.get(i) == ultimoPrimoA)
339             {
340                 primoSuperior = Primos.get(i+1);
341                 i = Primos.size()-1;
342             }
343         }
344
345         limiteInferior = 1+inputNumber-(primoSuperior*primoSuperior);
346
347         if(B.size()==0)
348             limiteSuperior = inputNumber-primoSuperior;
349         else
350             limiteSuperior = inputNumber-3;
351
352         if(B.size()==0 && limiteInferior<=2)
353             limiteInferior=ultimoPrimoA+1;
354
355         if(B.size()!=0 && limiteInferior<=2)
356             limiteInferior=B.get(0);
357
358         for(int i=limiteInferior; i<(limiteSuperior+1); i++)
359         {
360             if(!determinaPrimo(i))
361             {
362                 primeiroPrimoIntervalo=i;
363                 i=limiteSuperior;
364                 neprimos = false;
365             }
366         }
367         if(neprimos==true)
368         {
369             texto.append("\nNão Existem Primos no Intervalo ["+limiteInferior+","+limiteSuperior+"].");
370             texto.append("\nExtendendo o Intervalo encontramos as seguintes \nPartições de Goldbach:");
371
372             posAux=determinaPosicao(primoSuperior, Primos);
373             primoSuperior = Primos.get(posAux+1);
374             limiteInferior = 1+inputNumber-(primoSuperior*primoSuperior);
375             limiteSuperior = inputNumber-primoSuperior;
376
377             for(int i=limiteInferior; i<(limiteSuperior+1); i++)
378             {
379                 if(!determinaPrimo(i))
380                 {
381                     primeiroPrimoIntervalo=i;
382                     i=limiteSuperior;
383                     neprimos = false;
384                 }
385             }
386
387             if(primeiroPrimoIntervalo == limiteSuperior)

```

```

388         Intervalo.add(limiteSuperior);
389     else
390     {
391         posAux=determinaPosicao(primeiroPrimoIntervalo, Primos);
392         aux = Primos.get(posAux);
393         for(int i=posAux; aux<limiteSuperior; i++)
394         {
395             if(Primos.get(i)==Primos.get(Primos.size()-1))
396                 aux = limiteSuperior;
397             else
398             {
399                 aux = Primos.get(i);
400                 if(Primos.get(i)<=limiteSuperior && ((inputNumber-Primos.get(i))%3!=0))
401                     Intervalo.add(Primos.get(i));
402             }
403         }
404     }
405
406     if(B.size()==0)
407     {
408         for(Integer i : Intervalo)
409         {
410             if(!Duplicados.contains(inputNumber-i)){
411                 Duplicados.add(i);
412                 texto.append("\n"+inputNumber+" = "+i+" + "+(inputNumber-i));
413             }
414         }
415     }
416     if(B.size()!=0)
417     {
418         for(Integer c : Intervalo)
419         {
420             count=0; stop = false;
421             for (Integer j : B)
422             {
423                 if(((inputNumber-c)%j)!=0)
424                     count++;
425             }
426             if(count==B.size() && !A.contains(c))
427             {
428                 if(!Duplicados.contains(inputNumber-c)){
429                     Duplicados.add(c);
430                     texto.append("\n"+inputNumber+" = "+c+" + "+(inputNumber-c));
431                 }
432             }
433         }
434     }
435
436     else
437     {
438         if(primeiroPrimoIntervalo == limiteSuperior)
439             Intervalo.add(limiteSuperior);
440         else
441         {
442             posAux=determinaPosicao(primeiroPrimoIntervalo, Primos);
443             aux = Primos.get(posAux);
444
445             for(int i=posAux; aux<limiteSuperior; i++)
446             {
447                 if(Primos.get(i)==Primos.get(Primos.size()-1))
448                     aux = limiteSuperior;
449                 else
450                 {
451                     aux = Primos.get(i);
452                     if(Primos.get(i)<=limiteSuperior)
453                         Intervalo.add(Primos.get(i));
454                 }
455             }
456         }

```

```

457
458         if (B.size()==0)
459         {
460             texto.append("\nPartições de Goldbach de "+inputNumber+" determinadas pelo intervalo");
461
462             texto.append("[ "+limiteInferior+", "+limiteSuperior+" ] são:");
463             for (Integer b : Intervalo)
464             {
465                 if (!Duplicados.contains(inputNumber-b)){
466                     Duplicados.add(b);
467                     texto.append("\n"+inputNumber+" = "+b+" + "+(inputNumber-b));
468                 }
469             }
470         }
471
472         if (B.size()!=0)
473         {
474             texto.append("\nPartições de Goldbach de "+inputNumber+" determinadas pelo intervalo");
475
476             texto.append("[ "+limiteInferior+", "+limiteSuperior+" ] são:");
477
478             for (Integer c : Intervalo)
479             {
480                 count=0; stop = false;
481                 for (Integer j : B)
482                 {
483                     if (((inputNumber-c)%j)!=0)
484                         count++;
485                 }
486                 if (count==B.size() && !A.contains(c))
487                 {
488                     if (!Duplicados.contains(inputNumber-c)){
489                         Duplicados.add(c);
490                         texto.append("\n"+inputNumber+" = "+c+" + "+(inputNumber-c));
491                     }
492                 }
493             }
494         }
495     }
496
497     if ((B.size()==0 && limiteInferior<=inputNumber/2) || (B.size()!=0 && limiteInferior<=B.get(0)))
498         texto.append("\n\nTodas as partições foram encontradas");
499 }
500 } catch (Exception ex) {
501     JOptionPane.showMessageDialog(this, "Introduziu um valor inválido");
502 }
503 }
504 }
505
506 class SobrePanel extends JPanel implements ActionListener {
507
508     JTextField msg1 = new JTextField("Universidade dos Açores");
509     JTextField msg2 = new JTextField("Departamento de Matemática");
510     JTextField msg3 = new JTextField("Mestrado em Matemática para Professores");
511     JTextField msg4 = new JTextField("Conjetura de Goldbach - Uma visão Aritmética");
512     JTextField msg6 = new JTextField("José Emanuel Sousa");
513     JTextField msg7 = new JTextField("Ponta Delgada");
514     JTextField msg8 = new JTextField("Abril de 2013");
515     JFileChooser fc = new JFileChooser();
516     ImageIcon img;
517     JLabel L1;
518
519     public SobrePanel() {
520         setLayout(null);
521
522         String aux = "c:\\Goldbach\\UAC.png";
523         img = new ImageIcon(aux);
524
525         L1 = new JLabel(img);

```

```

526         L1.setVisible(true);
527
528         L1.setBounds(10, 5, 450, 180);
529         msg1.setBounds(10, 190, 450, 20);
530         msg2.setBounds(10, 210, 450, 20);
531         msg3.setBounds(10, 240, 450, 20);
532         msg4.setBounds(10, 270, 450, 20);
533         msg6.setBounds(10, 300, 450, 20);
534         msg7.setBounds(10, 330, 450, 20);
535         msg8.setBounds(10, 360, 450, 20);
536
537         msg1.setBorder(javax.swing.BorderFactory.createEmptyBorder());
538         msg2.setBorder(javax.swing.BorderFactory.createEmptyBorder());
539         msg3.setBorder(javax.swing.BorderFactory.createEmptyBorder());
540         msg4.setBorder(javax.swing.BorderFactory.createEmptyBorder());
541         msg6.setBorder(javax.swing.BorderFactory.createEmptyBorder());
542         msg7.setBorder(javax.swing.BorderFactory.createEmptyBorder());
543         msg8.setBorder(javax.swing.BorderFactory.createEmptyBorder());
544
545         msg1.setEditable(false);
546         msg2.setEditable(false);
547         msg3.setEditable(false);
548         msg4.setEditable(false);
549         msg6.setEditable(false);
550         msg7.setEditable(false);
551         msg8.setEditable(false);
552
553         msg1.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
554         msg2.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
555         msg3.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
556         msg4.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
557         msg6.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
558         msg7.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
559         msg8.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
560         Font font = new Font("Arial", Font.BOLD, 13);
561         msg1.setFont(font);
562         msg2.setFont(font);
563         msg3.setFont(font);
564         msg4.setFont(font);
565         msg6.setFont(font);
566         msg7.setFont(font);
567         msg8.setFont(font);
568
569         msg1.setHorizontalAlignment(JTextField.CENTER);
570         msg2.setHorizontalAlignment(JTextField.CENTER);
571         msg3.setHorizontalAlignment(JTextField.CENTER);
572         msg4.setHorizontalAlignment(JTextField.CENTER);
573         msg6.setHorizontalAlignment(JTextField.CENTER);
574         msg7.setHorizontalAlignment(JTextField.CENTER);
575         msg8.setHorizontalAlignment(JTextField.CENTER);
576
577         add(L1);
578         add(msg1);
579         add(msg2);
580         add(msg3);
581         add(msg4);
582         add(msg6);
583         add(msg7);
584         add(msg8);
585
586         setVisible(true);
587     }
588
589     @Override
590     public void actionPerformed(ActionEvent e) {
591     }
592 }
593
594 public static void main(String args[]) {

```

```
595     new Conjetura_Goldbach();
596 }
597
598 public static Integer determinaPosicao(int p, ArrayList<Integer> Primos)
599 {
600     for(int i=0; i<Primos.size(); i++)
601     {
602         if(Primos.get(i) == p)
603         {
604             p = i;
605             i = Primos.size()-1;
606         }
607     }
608     return p;
609 }
610
611 public static Boolean determinaPrimo(int p)
612 {
613     boolean flag=false;
614
615     for(int k=2;k<p;k++)
616     {
617         if(p%k==0)
618             flag=true;
619     }
620     return flag;
621 }
622
623 private static void criarListaPrimos(ArrayList<Integer> Primos)
624 {
625     File f = new File("C:/Goldbach/primos.txt");
626
627     int a;
628
629     if(f.exists())
630     {
631         Scanner scanner = null;
632         try {
633             scanner = new Scanner(f);
634             while(scanner.hasNextInt()){
635                 a = scanner.nextInt();
636                 Primos.add(a);
637             }
638         } catch (FileNotFoundException e)
639         {
640             e.printStackTrace();
641         }
642     }else{
643         System.out.println("File not found!");
644     }
645 }
646
647 private static void criarListaA(int inputNumber, ArrayList<Integer> A, ArrayList<Integer> Primos)
648 {
649     int num, k, i=0;
650
651     num = inputNumber;
652     k = (int) num/2;
653
654     while(Primos.get(i)<=k && num!=1)
655     {
656         if((num%Primos.get(i))==0)
657         {
658             if(!A.contains(Primos.get(i)))
659                 A.add(Primos.get(i));
660             num = num/Primos.get(i);
661         }
662         if(num%Primos.get(i)!=0)
663     {
```



```

664         i++;
665     }
666 }
667 }
668
669 private static void criarListaB(ArrayList<Integer> A, ArrayList<Integer> B, ArrayList<Integer> Primos)
670 {
671     int j = A.get(A.size()-1);
672     for(int i=0; i<Primos.size(); i++)
673     {
674         if((Primos.get(i)!= j) && (!A.contains(Primos.get(i))))
675         {
676             B.add(Primos.get(i));
677         }
678         else
679             if (Primos.get(i)== j)
680             {
681                 i = Primos.size()-1;
682             }
683     }
684 }
685 }

```

Applet - Sistema de Identificação

```

1 package Sistema_Identificacao;
2
3 import java.awt.BorderLayout;
4 import java.awt.CardLayout;
5 import java.awt.Color;
6 import java.awt.Container;
7 import java.awt.Dimension;
8 import java.awt.Font;
9 import java.awt.event.ActionEvent;
10 import java.awt.event.ActionListener;
11 import java.io.File;
12 import java.io.FileNotFoundException;
13 import java.util.ArrayList;
14 import java.util.Random;
15 import java.util.Scanner;
16 import javax.swing.ImageIcon;
17 import javax.swing.JButton;
18 import javax.swing.JFileChooser;
19 import javax.swing.JFrame;
20 import javax.swing.JLabel;
21 import javax.swing.JMenu;
22 import javax.swing.JMenuBar;
23 import javax.swing.JMenuItem;
24 import javax.swing.JOptionPane;
25 import javax.swing.JPanel;
26 import javax.swing.JTextArea;
27 import javax.swing.JTextField;
28
29 /**
30  * Created on Abril 2013
31  *
32  */
33
34 /**
35  * @author José Emanuel Sousa
36  * Mestrado em Matemática para Professores
37  */
38
39 public class Sistema_Identificacao extends JFrame {
40
41     private Container cp;
42     private SisIdentPanel sisIdentPanel;
43     private TransmissaoPanel transmissaoPanel;

```

```

44     private SobrePanel sobrePanel;
45     private JTextArea sistIdentificacao = new JTextArea(6, 20);
46
47     private JPanel cardPanel;
48     private JMenu[] menus = {
49         new JMenu("Sistema de Identificação"), new JMenu("Transmissão"), new JMenu("Sobre")
50     };
51     private JMenuItem[] items = {
52         new JMenuItem("Executar"), new JMenuItem("Transmitir"), new JMenuItem("Sobre")
53     };
54     private ActionListener action = new ActionListener() {
55         @Override
56         public void actionPerformed(ActionEvent e) {
57             JMenuItem target = (JMenuItem) e.getSource();
58
59             if (target.getActionCommand().equals("Executar")) {
60                 update("executar");
61             } else if (target.getActionCommand().equals("Sobre")) {
62                 update("sobre");
63             } else
64             {
65                 update("transmitir");
66             }
67         }
68     };
69
70     public Sistema_Identificacao() {
71         super("Sistema de Identificação");
72         for (int i = 0; i < items.length; i++) {
73             items[i].addActionListener(action);
74             menus[i % 4].add(items[i]);
75         }
76         JMenuBar mb = new JMenuBar();
77         for (int i = 0; i < menus.length; i++) {
78             mb.add(menus[i]);
79         }
80         setJMenuBar(mb);
81
82         sisIdentPanel = new SisIdentPanel();
83         sobrePanel = new SobrePanel();
84         transmissaoPanel = new TransmissaoPanel();
85
86         sisIdentPanel.setBackground(Color.getHSBColor(0.0f, 0.0f, 0.95f));
87         transmissaoPanel.setBackground(Color.getHSBColor(0.0f, 0.0f, 0.95f));
88         sobrePanel.setBackground(Color.getHSBColor(0.0f, 0.0f, 0.95f));
89
90         cardPanel = new JPanel();
91         cardPanel.setLayout(new CardLayout());
92
93         cp = getContentPane();
94         cp.setLayout(new BorderLayout());
95         cardPanel.add(sisIdentPanel, "executar");
96         cardPanel.add(transmissaoPanel, "transmitir");
97         cardPanel.add(sobrePanel, "sobre");
98
99         cp.add(cardPanel, BorderLayout.CENTER);
100         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
101         setSize(new Dimension(500, 500));
102         setResizable(false);
103         setVisible(true);
104     }
105
106     public void update(String txt) {
107         CardLayout card = (CardLayout) cardPanel.getLayout();
108         card.show(cardPanel, txt);
109     }
110
111     class SisIdentPanel extends JPanel implements ActionListener {
112         JLabel numero = new JLabel("Introduza um número par maior ou igual a 8:");

```

```

113     JTextArea texto = new JTextArea(6, 20);
114     JTextField caixaTexto = new JTextField();
115     JButton botao = new JButton("Executar");
116     JButton Limpar = new JButton("Limpar");
117
118     public SisIdentPanel() {
119         setLayout(null);
120         numero.setBounds(10, 10, 450, 20);
121         caixaTexto.setBounds(10, 40, 350, 20);
122         botao.setBounds(380, 40, 90, 20);
123         Limpar.setBounds(380, 400, 90, 20);
124         texto.setBounds(10, 65, 450, 190);
125         texto.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
126         Font font = new Font("Arial", Font.BOLD, 12);
127
128         botao.addActionListener(this);
129         Limpar.setToolTipText("Limpar");
130         texto.setEditable(false);
131         texto.setFont(font);
132
133         add(numero);
134         add(caixaTexto);
135         add(botao);
136         add(Limpar);
137         add(texto);
138         setVisible(true);
139
140         Limpar.addActionListener(new ActionListener(){
141
142             public void actionPerformed ( ActionEvent event){
143                 caixaTexto.setText("");
144                 texto.setText("");
145             }
146         });
147     }
148
149     @Override
150     public void actionPerformed(ActionEvent e) {
151         ArrayList<Integer> A = new ArrayList<Integer>();
152         ArrayList<Integer> B = new ArrayList<Integer>();
153         ArrayList<Integer> Primos = new ArrayList<Integer>();
154         ArrayList<Integer> Intervalo = new ArrayList<Integer>();
155         ArrayList<Integer> Particoes = new ArrayList<Integer>();
156         ArrayList<Integer> Duplicados = new ArrayList<Integer>();
157
158         Scanner input = new Scanner(System.in);
159         int inputNumber = 0, ultimoPrimoA, primoSuperior=0;
160         int primeiroPrimoIntervalo=0, limiteInferior = 0, limiteSuperior, count=0;
161         boolean stop, neprimos = true;
162         int posAux, aux, p, a, q, range;
163         Random rn;
164
165
166         try {
167             String inputNum = caixaTexto.getText();
168             inputNumber = (int) Integer.parseInt(inputNum);
169
170             if(inputNumber<=6 || inputNumber%2!=0 || inputNumber>1299830)
171                 JOptionPane.showMessageDialog(this, "Introduziu um valor inválido");
172
173             else {
174                 criarListaPrimos(Primos);
175                 criarListaA(inputNumber, A, Primos);
176                 criarListaB(A, B, Primos);
177
178                 if((A.size() == 2) && (A.get(A.size()-1)*2 == inputNumber))
179                 {
180                     Particoes.add(A.get(A.size()-1));
181

```

```

182
183         ultimoPrimoA = A.get(A.size()-1);
184
185     for(int i=0; i<Primos.size(); i++)
186     {
187         if(Primos.get(i) == ultimoPrimoA)
188         {
189             primoSuperior = Primos.get(i+1);
190             i = Primos.size()-1;
191         }
192     }
193
194     limiteInferior = 1+inputNumber-(primoSuperior*primoSuperior);
195
196     if(B.size()==0)
197         limiteSuperior = inputNumber-primoSuperior;
198     else
199         limiteSuperior = inputNumber-3;
200
201     if(B.size()==0 && limiteInferior<=2)
202         limiteInferior=ultimoPrimoA+1;
203
204     if(B.size()!=0 && limiteInferior<=2)
205         limiteInferior=B.get(0);
206
207     for(int i=limiteInferior; i<limiteSuperior; i++)
208     {
209         if(!determinaPrimo(i))
210         {
211             primeiroPrimoIntervalo=i;
212             i=limiteSuperior-1;
213             neprimos = false;
214         }
215     }
216
217     if(neprimos==true)
218     {
219         posAux=determinaPosicao(primoSuperior, Primos);
220         primoSuperior = Primos.get(posAux+1);
221         limiteInferior = 1+inputNumber-(primoSuperior*primoSuperior);
222         limiteSuperior = inputNumber-primoSuperior;
223
224         for(int i=limiteInferior; i<limiteSuperior; i++)
225         {
226             if(!determinaPrimo(i))
227             {
228                 primeiroPrimoIntervalo=i;
229                 i=limiteSuperior-1;
230                 neprimos = false;
231             }
232         }
233
234         posAux=determinaPosicao(primeiroPrimoIntervalo, Primos);
235         aux = Primos.get(posAux);
236         for(int i=posAux; aux<limiteSuperior; i++)
237         {
238             if(Primos.get(i)==Primos.get(Primos.size()-1))
239                 aux = limiteSuperior;
240             else
241             {
242                 aux = Primos.get(i);
243                 if(Primos.get(i)<=limiteSuperior && ((inputNumber-Primos.get(i))%3!=0))
244                     Intervalo.add(Primos.get(i));
245             }
246         }
247     if(B.size()==0)
248     {
249         for(Integer i : Intervalo)
250         {

```

```

251         if(!Duplicados.contains(inputNumber-i)){
252             Duplicados.add(i);
253             Particoes.add(i);
254         }
255     }
256 }
257 if(B.size()!=0)
258 {
259     for(Integer c : Intervalo)
260     {
261         count=0; stop = false;
262         for (Integer g : B)
263         {
264             if(((inputNumber-c)%g)!=0)
265                 count++;
266         }
267         if(count==B.size() && !A.contains(c))
268         {
269             if(!Duplicados.contains(inputNumber-c)){
270                 Duplicados.add(c);
271                 Particoes.add(c);
272             }
273         }
274     }
275 }
276
277 else
278 {
279     posAux=determinaPosicao(primeiroPrimoIntervalo, Primos);
280     aux = Primos.get(posAux);
281
282     for(int i=posAux; aux<limiteSuperior; i++)
283     {
284         if(Primos.get(i)==Primos.get(Primos.size()-1))
285             aux = limiteSuperior;
286         else
287         {
288             aux = Primos.get(i);
289             if(Primos.get(i)<=limiteSuperior)
290                 Intervalo.add(Primos.get(i));
291         }
292     }
293
294     if(B.size()==0)
295     {
296         for(Integer b : Intervalo)
297         {
298             if(!Duplicados.contains(inputNumber-b)){
299                 Duplicados.add(b);
300                 Particoes.add(b);
301             }
302         }
303     }
304
305     if(B.size()!=0)
306     {
307         for(Integer c : Intervalo)
308         {
309             count=0; stop = false;
310             for (Integer j : B)
311             {
312                 if(((inputNumber-c)%j)!=0)
313                     count++;
314             }
315             if(count==B.size() && !A.contains(c))
316             {
317                 if(!Duplicados.contains(inputNumber-c)){
318                     Duplicados.add(c);
319                     Particoes.add(c);

```

```

320         }
321     }
322 }
323 }
324 }
325 }
326
327     rn = new Random();
328     range = Particoes.size();
329     p = Particoes.get(rn.nextInt(range));
330     a = inputNumber;
331     q = a-p;
332
333     if(p>q)
334     {
335         p = q;
336         q = a-p;
337     }
338     int j = a/2 - p;
339
340     texto.append("O número de identificação criado é :\n\n"+a+" - "+p+" - "+q+" - "+j);
341     sistIdentificacao.setText("");
342     sistIdentificacao.append("Número de identificação criado é : "+a+" - "+p+" - "+q+" - "+j);
343
344     } catch (Exception ex) {
345         JOptionPane.showMessageDialog(this, "Introduziu um valor inválido");
346     }
347 }
348 }
349
350 class SobrePanel extends JPanel implements ActionListener {
351
352     JTextField msg1 = new JTextField("Universidade dos Açores");
353     JTextField msg2 = new JTextField("Departamento de Matemática");
354     JTextField msg3 = new JTextField("Mestrado em Matemática para Professores");
355     JTextField msg4 = new JTextField("Conjetura de Goldbach - Uma visão Aritmética");
356     JTextField msg6 = new JTextField("José Emanuel Sousa");
357     JTextField msg7 = new JTextField("Ponta Delgada");
358     JTextField msg8 = new JTextField("Abril de 2013");
359     JFileChooser fc = new JFileChooser();
360     ImageIcon img;
361     JLabel L1;
362
363     public SobrePanel() {
364         setLayout(null);
365
366         String aux = "c:\\Goldbach\\UAC.png";
367         img = new ImageIcon(aux);
368
369         L1 = new JLabel(img);
370         L1.setVisible(true);
371
372         L1.setBounds(10, 5, 450, 180);
373         msg1.setBounds(10, 190, 450, 20);
374         msg2.setBounds(10, 210, 450, 20);
375         msg3.setBounds(10, 240, 450, 20);
376         msg4.setBounds(10, 270, 450, 20);
377         msg6.setBounds(10, 300, 450, 20);
378         msg7.setBounds(10, 330, 450, 20);
379         msg8.setBounds(10, 360, 450, 20);
380
381         msg1.setBorder(javax.swing.BorderFactory.createEmptyBorder());
382         msg2.setBorder(javax.swing.BorderFactory.createEmptyBorder());
383         msg3.setBorder(javax.swing.BorderFactory.createEmptyBorder());
384         msg4.setBorder(javax.swing.BorderFactory.createEmptyBorder());
385         msg6.setBorder(javax.swing.BorderFactory.createEmptyBorder());
386         msg7.setBorder(javax.swing.BorderFactory.createEmptyBorder());
387         msg8.setBorder(javax.swing.BorderFactory.createEmptyBorder());
388

```

```

389         msg1.setEditable(false);
390         msg2.setEditable(false);
391         msg3.setEditable(false);
392         msg4.setEditable(false);
393         msg6.setEditable(false);
394         msg7.setEditable(false);
395         msg8.setEditable(false);
396
397         msg1.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
398         msg2.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
399         msg3.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
400         msg4.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
401         msg6.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
402         msg7.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
403         msg8.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
404         Font font = new Font("Arial", Font.BOLD, 13);
405         msg1.setFont(font);
406         msg2.setFont(font);
407         msg3.setFont(font);
408         msg4.setFont(font);
409         msg6.setFont(font);
410         msg7.setFont(font);
411         msg8.setFont(font);
412
413         msg1.setHorizontalAlignment(JTextField.CENTER);
414         msg2.setHorizontalAlignment(JTextField.CENTER);
415         msg3.setHorizontalAlignment(JTextField.CENTER);
416         msg4.setHorizontalAlignment(JTextField.CENTER);
417         msg6.setHorizontalAlignment(JTextField.CENTER);
418         msg7.setHorizontalAlignment(JTextField.CENTER);
419         msg8.setHorizontalAlignment(JTextField.CENTER);
420
421         add(L1);
422         add(msg1);
423         add(msg2);
424         add(msg3);
425         add(msg4);
426         add(msg6);
427         add(msg7);
428         add(msg8);
429
430         setVisible(true);
431     }
432     @Override
433     public void actionPerformed(ActionEvent e) {
434     }
435 }
436 class TransmissaoPanel extends JPanel implements ActionListener
437 {
438     JLabel numero = new JLabel("Indique o número de identificação a transmitir:");
439     JTextField a = new JTextField();
440     JTextField p = new JTextField();
441     JTextField q = new JTextField();
442     JTextField j = new JTextField();
443     JTextArea texto = new JTextArea(6, 20);
444     JButton botao = new JButton("Executar");
445     JButton Limpar = new JButton("Limpar");
446
447
448     public TransmissaoPanel() {
449
450         setLayout(null);
451         sistIdentificacao.setBounds(10, 10, 450, 20);
452         numero.setBounds(10, 40, 450, 20);
453         a.setBounds(10, 70, 67, 20);
454         p.setBounds(80, 70, 67, 20);
455         q.setBounds(150, 70, 67, 20);
456         j.setBounds(220, 70, 67, 20);
457         botao.setBounds(380, 70, 90, 20);

```

```

458         Limpar.setBounds(380, 600, 90, 20);
459         texto.setBounds(10, 100, 550, 190);
460
461         sistIdentificacao.setEditable(false);
462         sistIdentificacao.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
463         texto.setBackground(Color.getHSBColor(0.0f,0.0f,0.95f));
464         Font font = new Font("Arial", Font.BOLD, 12);
465         sistIdentificacao.setFont(font);
466
467         botao.addActionListener(this);
468         Limpar.setToolTipText("Limpar");
469         texto.setEditable(false);
470         texto.setFont(font);
471
472         add(sistIdentificacao);
473         add(numero);
474         add(a);
475         add(p);
476         add(q);
477         add(j);
478         add(botao);
479         add(Limpar);
480         add(texto);
481         setVisible(true);
482
483         Limpar.addActionListener(new ActionListener(){
484
485             public void actionPerformed ( ActionEvent event){
486                 a.setText("");
487                 p.setText("");
488                 q.setText("");
489                 j.setText("");
490                 sistIdentificacao.setText("");
491                 texto.setText("");
492             }
493         });
494     }
495
496     @Override
497     public void actionPerformed(ActionEvent e) {
498
499         try {
500             String inputA = a.getText();
501             String inputP = p.getText();
502             String inputQ = q.getText();
503             String inputJ = j.getText();
504
505             int numA = (int) Integer.parseInt(inputA);
506             int numP = (int) Integer.parseInt(inputP);
507             int numQ = (int) Integer.parseInt(inputQ);
508             int numJ = (int) Integer.parseInt(inputJ);
509
510             if(numA<=1 || numP<=1 || numQ<=1 || numJ<=1 || numA>1299830 || numP>1299830 || numQ>1299830 || numJ>1299830)
511                 JOptionPane.showMessageDialog(this, "Introduziu um valor inválido");
512             else {
513                 if ((numP+numQ == numA) && (numA/2-numP == numJ) && (numQ-numA/2 == numJ))
514                 {
515                     texto.setText("Não foi detectado nenhum erro");
516                 }
517                 else
518                 {
519                     texto.setText("Erro detectado.\nIntroduza novamente.");
520                 }
521             }
522         } catch (Exception ex) {
523             JOptionPane.showMessageDialog(this, "Introduziu um valor inválido");
524         }
525     }
526 }

```



```
527
528 public static void main(String args[]) {
529     new Sistema_Identificacao();
530 }
531
532 public static Integer determinaPosicao(int p, ArrayList<Integer> Primos)
533 {
534     for(int i=0; i<Primos.size(); i++)
535     {
536         if(Primos.get(i) == p)
537         {
538             p = i;
539             i = Primos.size()-1;
540         }
541     }
542     return p;
543 }
544
545 public static Boolean determinaPrimo(int p)
546 {
547     boolean flag=false;
548
549     for(int k=2;k<p;k++)
550     {
551         if(p%k==0)
552             flag=true;
553     }
554     return flag;
555 }
556
557 private static void criarListaPrimos(ArrayList<Integer> Primos)
558 {
559     File f = new File("C:/Goldbach/primos.txt");
560
561     int a;
562
563     if(f.exists())
564     {
565         Scanner scanner = null;
566         try {
567             scanner = new Scanner(f);
568             while(scanner.hasNextInt()){
569                 a = scanner.nextInt();
570                 Primos.add(a);
571             }
572         } catch (FileNotFoundException e)
573         {
574             e.printStackTrace();
575         }
576     }else{
577         System.out.println("File not found!");
578     }
579 }
580
581 private static void criarListaA(int inputNumber, ArrayList<Integer> A, ArrayList<Integer> Primos)
582 {
583     int num, k, i=0;
584
585     num = inputNumber;
586     k = (int) num/2;
587
588     while(Primos.get(i)<=k && num!=1)
589     {
590         if((num%Primos.get(i))==0)
591         {
592             if(!A.contains(Primos.get(i)))
593                 A.add(Primos.get(i));
594             num = num/Primos.get(i);
595         }
596     }
```

```
596         if(num%Primos.get(i)!=0)
597         {
598             i++;
599         }
600     }
601 }
602
603 private static void criarListaB(ArrayList<Integer> A, ArrayList<Integer> B, ArrayList<Integer> Primos)
604 {
605     int j = A.get(A.size()-1);
606     for(int i=0; i<Primos.size(); i++)
607     {
608         if((Primos.get(i)!= j) && (!A.contains(Primos.get(i))))
609         {
610             B.add(Primos.get(i));
611         }
612         else
613             if (Primos.get(i)== j)
614             {
615                 i = Primos.size()-1;
616             }
617     }
618 }
619 }
```