UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA
DEPARTAMENTO DE MATEMÁTICA APLICADA

THIAGO FELIPE CASTRO CARRENHO

Novas aplicações da representação matricial de partições para casos restritos e irrestritos

Campinas
07/07/2021

THIAGO FELIPE CASTRO CARRENHO

Novas aplicações da representação matricial de partições para casos restritos e irrestritos*

Monografia apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos para obtenção de créditos na disciplina Projeto Supervisionado II, sob a orientação do(a) Prof. José Plinio de Oliveira Santos.

*Este trabalho foi financiado pelo CNPq, projeto 135119/2020.

Resumo

A primeira parte deste projeto tem por finalidade apresentar a implementação de um algoritmo que calcule o número de partições irrestritas de $n \in \mathbb{N}$ a partir de uma bijeção entre o conjunto de partições irrestritas de n e um conjunto de matrizes de duas linhas de inteiros não negativos. Em seguida mostramos o mesmo processo de construção de árvore por meio da representação matricial de uma partição para o caso em que a partição tem parte mínima sendo maior do que ou igual a $c \in \mathbb{N}$, com $c < n - 1$ e diferença mínima entre as partes $\lambda \in \mathbb{N}$

Palavras-Chave: Partições irrestritas. Partições. Matrizes de duas linhas.

Abstract

The first part of this project aims to present an implementation of an algorithm that computes the number of unrestricted partitions of $n \in \mathbb{N}$ from a bijection between the set of unrestricted partitions of n and a set of two-line matrices of non-negative integers. Then we show the same process of constructing the tree by the matrix represention of a partition with the restrictions of minimum part greater than or equal to $c \in \mathbb{N}$, $c < n-1$, and minimum difference between parts being $\lambda \in \mathbb{N}$

Keywords: Unrestricted partitions. Partitions. Two-line matrices.

Conteúdo

1 Introdução

Partindo de algumas definições, vamos encontrar a bijeção entre o conjunto de partições irrestritas de n e um conjunto de matrizes de duas linhas de inteiros não negativos.

Definição 1.1 (Partição)**.** *Uma partição de $n \in \mathbb{N}$ é uma coleção de inteiros positivos cuja soma é n, isto é, $\Omega = \{c_1, c_2, \cdots, c_s\}$ é uma partição de n se $c_1 + c_2 + \cdots + c_s = n$.*

A ordem das partes $c_1, c_2, \cdots, c_s$ é irrelevante, por isso podemos pedir uma ordenação específica. Por questão de facilidade, vamos pedir que $c_1 \geq c_2 \geq \cdots \geq c_s \geq 1$.

*A definição acima é de **partição irrestrita**, se colocarmos algumas restrições para as partições, definiremos as **partições restritas**, como por exemplo, pedir que todas as partes sejam números pares, todas as partes sejam maiores do que 5, todas as partes sejam distintas, entre outras.*

Em especial neste projeto, trabalharemos apenas com dois tipos de restrições.

Tamanho mínimo: todas as partes devem ser no mínimo $c \in \mathbb{N}$, com $c < n-1$, isto é, para $\Omega = \{c_1, c_2, \cdots, c_s\}$, $c_1 \geq c_2 \geq \cdots \geq c_s \geq c \in \mathbb{N}$.

Diferença mínima entre as partes: as partes devem ser distintas entre si por, no mínimo, $\lambda \in \mathbb{N}$, isto é $c_i \geq \lambda + c_{i+1}$, $\forall i = 1, \cdots, s-1$.

Perceba que, na definição de partição, todas as partes são números inteiros positivos, então se $c = 1$, a restrição de tamanho mínimo se torna redundante, além disso, representamos as partes numa sequência não decrescente, então se $\lambda = 0$, a restrição de diferença mínima entre as partes é redundante, logo, não temos nenhuma restrição adicional, voltando para o caso irrestrito.

Definição 1.2 (Conjunto de partições)**.** *Unindo todas as partições de n com tamanho mínimo c e diferença mínima entre as partes λ, temos o conjunto de partições $\mathbb{P}(n, c, \lambda)$.*

Dessa forma, temos que $\mathbb{P}(n, 1, 0)$ é o conjunto de partições irrestritas, e $p(n) = \#\mathbb{P}(n, 1, 0)$.

Definição 1.3. *$\mathbb{M}(n, c, \lambda)$ é o conjunto de todas matrizes de duas linhas do tipo*

$$M = \begin{pmatrix} a_1 & a_2 & \cdots & a_s \\ b_1 & b_2 & \cdots & b_s \end{pmatrix}$$

onde $a_j, b_j \in \mathbb{N} \cup \{0\}$ *e*

$$a_s = c,\ a_j = a_{j+1} + b_{j+1} + \lambda \text{ e } \sum_{i=1}^{s}(a_i + b_i) = n$$

Tendo uma matriz de duas linhas desta forma, tomemos $c_j = a_j + b_j$, como definido $a_j, b_j \in \mathbb{N} \cup \{0\}$ e $a_j = a_{j+1} + b_{j+1} + \lambda\ \forall j = 1, \cdots, s$, temos que $c_j \geq c_{j+1}$, isto é, $\{a_1 + b_1, a_2 + b_2, \cdots, a_s + b_s\}$ é uma partição de n. Dessa forma, vemos que cada matriz se relaciona a uma única partição.

Agora, tendo uma partição $\{c_1, c_2, \cdots, c_s\}$, tomemos $a_s = c$ e $b_s = c_s - 1$ para a última coluna, e as outras colunas com $a_j = a_{j+1} + b_{j+1} + \lambda$ e $b_j = c_j - a_j$, para $j = 1, \cdots, s-1$, obtendo uma matriz pertencente a $\mathbb{M}(n, c, \lambda)$.

Além disso, é fácil perceber que, por estes dois processos, uma matriz M sempre será ligada a uma partição Ω, e esta partição sempre será ligada à mesma matriz M. Assim, temos uma bijeção entre o conjunto de partições $\mathbb{P}(n, c, \lambda)$ e o conjunto $\mathbb{M}(n, c, \lambda)$.

Ao definir partições definimos $c < n - 1$, pois

$$|\mathbb{M}(n, n, \lambda)| = |\mathbb{M}(n, n-1, \lambda)| = 1 \text{ e } \mathbb{M}(n, n+t, \lambda) = \emptyset,\ t \in \mathbb{N}.$$

Inicialmente, trabalharemos apenas com partições irrestritas, com o propósito de implementar um algoritmo que calcule $p(n)$. Abaixo, temos um exemplo das partições irrestritas de 5, e sua respectiva forma matricial.

Exemplo 1.1. *Tomando* $n = 5$, $c = 1$ *e* $\lambda = 0$, *isto é, estamos tratando do caso irrestrito, listamos todas as partições e as respectivas matrizes abaixo*

$c_1 + c_2 + \cdots + c_s$	$\{c_1, c_2, \cdots, c_s\}$	$\begin{pmatrix} a_1 & a_2 & \cdots & a_s \\ b_1 & b_2 & \cdots & b_s \end{pmatrix}$
5	$\{5\}$	$\begin{pmatrix} 1 \\ 4 \end{pmatrix}$
$4+1$	$\{4,1\}$	$\begin{pmatrix} 1 & 1 \\ 3 & 0 \end{pmatrix}$
$3+2$	$\{3,2\}$	$\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$
$3+1+1$	$\{3,1,1\}$	$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 0 & 0 \end{pmatrix}$
$2+2+1$	$\{2,2,1\}$	$\begin{pmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$
$2+1+1+1$	$\{2,1,1,1\}$	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$
$1+1+1+1+1$	$\{1,1,1,1,1\}$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Portanto, $p(5) = \#\mathbb{P}(5,1,0) = \#\mathbb{M}(5,1,0) = 7$.

Uma explicação mais detalhada sobre partições pode ser encontrada em Santos et al. [2007], para saber mais sobre a representação de partições em matrizes de duas linhas, veja Santos et al. [2011], e uma visão de um procedimento por caminhos, e uma aproximação obtida a partir deste procedimento são abordados em Santos and Matte [2018].

Obtendo a bijeção, temos que a cardinalidade dos dois conjuntos é a mesma, e este algoritmo calcula quantas matrizes existem no conjunto $\mathbb{M}(n,1,0)$, que, por haver bijeção com $\mathbb{P}(n)$, é o mesmo que o número de partições irrestritas de n.

Partindo disso, a dedução do algoritmo depende da análise feita no artigo Godinho and Santos [2020], que possui um caso especial para partições irrestritas, uma fórmula fechada para este caso, e é este algoritmo que buscamos implementar.

2 Desenvolvimento

Dadas duas matrizes de duas linhas quaisquer A e B, definimos a operação de justaposição $A \uplus B$ como

$$A \uplus B = \begin{pmatrix} a_1 & \cdots & a_s \\ b_1 & \cdots & b_s \end{pmatrix} \uplus \begin{pmatrix} c_1 & \cdots & c_t \\ d_1 & \cdots & d_t \end{pmatrix} = \begin{pmatrix} a_1 & \cdots & a_s & c_1 & \cdots & c_t \\ b_1 & \cdots & b_s & d_1 & \cdots & d_t \end{pmatrix}.$$

Agora vamos definir a matriz $A(t_1, t_2)$, que é importante para a contagem do número de zeros no final da segunda linha, e a matriz $D(t_1, \cdots, t_\ell)$:

$$A(t_1, t_2) = \begin{pmatrix} c + t_1\lambda & c + (t_1 - 1)\lambda & \cdots & c + \lambda & c \\ 1 + t_2 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

$$D(t_1, t_2, t_3) = \begin{pmatrix} c + (t_1 + 1)\lambda + 1 + t_2 \\ t_3 \end{pmatrix}.$$

E recursivamente, para $\ell \geq 4$

$$D(t_1, \ldots, t_\ell) = \begin{pmatrix} c + (t_1 + 1)\lambda + 1 + (\ell - 3)\lambda + \sum_{i=2}^{\ell-1} t_i \\ t_\ell \end{pmatrix} \uplus D(t_1, \ldots, t_{\ell-1}).$$

Precisaremos também das seguintes funções:

$$L(x) = c + (x + 1)\lambda + 1$$

$$Q(x) = c(x + 1) + \lambda\frac{x(x + 1)}{2}$$

$$m(r) = n - Q(r).$$

A partir destas funções vamos definir a primeira geração da nossa árvore de matrizes: os blocos, definimos os blocos como abaixo, e cada um deles vai gerar uma série

de descendentes, e assim, montamos a árvore.

$$B(0) = \begin{pmatrix} c \\ m(0) \end{pmatrix}, B(1) = \begin{pmatrix} c+\lambda & c \\ m(1) & 0 \end{pmatrix}, \cdots, B(r) = \begin{pmatrix} c+r\lambda & \cdots & c+\lambda & c \\ m(r) & \cdots & 0 & 0 \end{pmatrix}.$$

Perceba que o bloco é um tipo de matriz $A(t_1, t_2)$, mais especificamente, $B(j) = A(j, m(j)-1)$, assim, vamos ter que os descendentes de $B(j)$ serão matrizes concatenada a uma $A(j, t_2)$, isto é, a descendência conserva o número de zeros no fim da segunda linha.

Além disso, precisamos que $m(r)$ seja não negativo, então $Q(r) \leq n$. Assim, tomamos a equação

$$Q(x) - n = \frac{1}{2}(\lambda x^2 + x(2c+\lambda) + 2(c-n)) = 0 \tag{1}$$

que possui duas raízes distintas de sinal distinto se $\lambda \neq 0$, e uma raiz positiva de $\lambda = 0$. Tomamos então r_1 como o piso da raiz positiva, e, como quisto, $m(r) \geq 0$ para $r = 0, 1 \cdots, r_1$, dessa forma, sabemos também que teremos $r_1 + 1$ blocos.

Para encontrar os primeiros descendentes de $B(t_1)$, definimos

$$m(t_1, t_2) = m(t_1) - 2t_2 - L(t_1) - 1,$$

e os descendentes serão da forma

$$F(t_1, t_2) = \begin{pmatrix} L(t_1) + t_2 \\ m(t_1, t_2) \end{pmatrix} \uplus A(t_1, t_2)$$

para $t_2 = 0, 1, \cdots, \lfloor \frac{m(t_1, 0)}{2} \rfloor$.

Para encontrar os descendentes de $F(t_1, t_2)$, isto é, a segunda geração de $B(t_1)$, definimos

$$m(t_1, t_2, t_3) = m(t_1, t_2) - L(t_1) - \lambda - t_2 - 2t_3$$

e os descendentes serão da forma

$$F(t_1,t_2,t_3)=\begin{pmatrix} L(t_1)+\lambda+t_2+t_3 \\ m(t_1,t_2,t_3)\end{pmatrix}\uplus D(t_1,t_2,t_3)\uplus A(t_1,t_2)$$

para $t_3=0,1,\cdots,\lfloor\frac{m(t_1,t_2,0)}{2}\rfloor$.

Por fim, recursivamente, para encontrar a ℓ-ésima geração de $B(t_1)$ (descendentes de $F(t_1,\cdots,t_{\ell-1})$) para $\ell\geq 4$, definimos

$$m(t_1,\cdots,t_\ell)=m(t_1,\cdots,t_{\ell-1})-L(t_1)-(\ell-2)\lambda-\sum_{i=2}^{\ell-1}t_i-2t_\ell$$

e os descendentes serão da forma

$$F(t_1,\cdots,t_\ell)=\begin{pmatrix} L(t_1)+(\ell-2)\lambda+\sum_{i=2}^{\ell-1}t_i \\ m(t_1,\cdots,t_\ell)\end{pmatrix}\uplus D(t_1,\cdots,t_\ell)\uplus A(t_1,t_2)$$

para $t_\ell=0,1,\cdots,\lfloor\frac{m(t_1,\cdots,t_{\ell-1},0)}{2}\rfloor$.

Assim temos definidas todas as matrizes de nossa árvore de acordo com o definido em Godinho and Santos [2020].

Vale a pena destacar que todas as matrizes descendentes de $B(t_1)$ são justapostas com $A(t_1,t_2)$, sendo assim, elas têm t_1 zeros no final da segunda linha, isso torna rápida a dedução de qual bloco uma matriz qualquer é descendente, além disso, o número de colunas de $F(t_1,\cdots,t_\ell)$ é o número de colunas de $A(t_1,t_2)$, t_1+1, somado ao número de colunas de $D(t_1,\cdots,t_\ell)$, $\ell-2$, somado a 1, isto é, $t_1+\ell$ colunas. Assim, sabendo t_1 contando os zeros à direita, sabemos também facilmente a geração da matriz.

Outro ponto interessante é que t_ℓ varia de 0 a $\lfloor\frac{m(t_1,\cdots,t_{\ell-1},0)}{2}\rfloor$, ou seja, se $m(t_1,\cdots,t_{\ell-1},0)<0$, o nó $F(t_1,\cdots,t_\ell$ é uma folha, não tem descendentes. Unindo essa informação com a definição recursiva da função m, percebemos que quanto maior t_i, menor é $m(t_1,\cdots,t_\ell)$, para todo $i=1,\cdots,\ell$.

2.1 Caso irrestrito

No caso irrestrito temos $c = 1$ e $\lambda = 0$, como definido na **Definição 1.2**, e temos $p(n) = \#\mathbb{M}(n, 1, 0)$.

Dado $n \in \mathbb{N}$, queremos que o algoritmo nos devolva $p(n)$. Para tal, vamos trazer as definições acima para esse caso simplificado:

$$A(t_1, t_2) = \underbrace{\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 + t_2 & 0 & \cdots & 0 & 0 \end{pmatrix}}_{t_i+1 \text{ colunas}}$$

$$D(t_1, t_2, t_3) = \begin{pmatrix} t_2 + 2 \\ t_3 \end{pmatrix}.$$

E recursivamente, para $\ell \geq 4$

$$D(t_1, \ldots, t_\ell) = \begin{pmatrix} 2 + \sum_{i=2}^{\ell-1} t_i \\ t_\ell \end{pmatrix} \uplus D(t_1, \ldots, t_{\ell-1}).$$

Adaptando as funções, temos que agora a função $L(x)$ é constante:

$$L(x) = 2$$

$$Q(x) = x + 1$$

$$m(r) = n - Q(r) = n - r - 1.$$

Então, adaptamos os blocos:

$$B(0) = \begin{pmatrix} 1 \\ n-1 \end{pmatrix}; B(1) = \begin{pmatrix} 1 & 1 \\ n-2 & 0 \end{pmatrix}; \cdots; B(n-1) = \underbrace{\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{pmatrix}}_{n \text{ colunas}}$$

Vale ressaltar que $D(t_1, \ldots, t_\ell)$ não depende de t_1, pois $\lambda = 0$, mas vamos

manter como uma variável de D para manter a conformidade o que já foi definido. Além disso, vale notar que os blocos podem ser escritos como a matriz A, de forma que $B(j) = A(j, n-j-2)$.

Agora, como $\lambda = 0$, a equação (1) tem uma única solução: $x = n-1$, isto é, nesse caso, teremos $r = n-1$, logo, $r+1 = n$ blocos.

A maneira que o algoritmo fará para calcular é por geração, isto é, na visão de árvore, contar por linha, o número de matrizes numa mesma linha, este número denotaremos por $\mathbb{D}_\ell$.

Para definir estes valores, precisamos do seguinte polinômio:

$$\mathcal{P}_0(x) = n - 2(x+1)$$

Cuja raiz é $x_0 = \dfrac{n}{2} - 1$, e com esta obtemos $\ell_0 = \left\lfloor \dfrac{n-2}{2} \right\rfloor + 1 = \left\lfloor \dfrac{n}{2} \right\rfloor$, que é o número de gerações da família de matrizes, a profundidade da árvore.

Com ℓ_0 podemos definir os seguintes polinômios:

$$\mathcal{P}_1(x) = n - 1 - x \text{ e } \mathcal{P}_j(x) = n - 2j - x \text{ para } j = 2, \cdots, \ell_0,$$

Cujas raízes são:

$$r_1 = n - 1 \text{ e } r_j = n - 2j \text{ para } j = 2, \cdots, \ell_0.$$

Agora, para encontrar os primeiros descendentes de $B(t_1)$, tomamos

$$m(t_1, t_2) = m(t_1) - 2t_2 - L(t_1) - 1 = n - t_1 - 2t_2 - 4$$

e os descendentes serão da forma

$$F(t_1, t_2) = \begin{pmatrix} 2 + t_2 \\ m(t_1, t_2) \end{pmatrix} \uplus A(t_1, t_2)$$

para $t_2 = 0, 1, \cdots, \lfloor \dfrac{m(t_1, 0)}{2} \rfloor$, com $m(t_1, 0) = n - t_1 - 4$.

Para encontrar os descendentes de $F(t_1, t_2)$, isto é, a segunda geração de $B(t_1)$,

definimos

$$m(t_1, t_2, t_3) = m(t_1, t_2) - L(t_1) - \lambda - t_2 - 2t_3 = m(t_1, t_2) - 2 - t_2 - 2t_3$$

e os descendentes serão da forma

$$F(t_1, t_2, t_3) = \begin{pmatrix} 2 + t_2 + t_3 \\ m(t_1, t_2, t_3) \end{pmatrix} \uplus D(t_1, t_2, t_3) \uplus A(t_1, t_2)$$

para $t_3 = 0, 1, \cdots, \lfloor \frac{m(t_1, t_2, 0)}{2} \rfloor$.

Por fim, recursivamente, para encontrar a ℓ-ésima geração de $B(t_1)$ (descendentes de $F(t_1, \cdots, t_{\ell-1})$) para $\ell \geq 4$, definimos

$$m(t_1, \cdots, t_\ell) = m(t_1, \cdots, t_{\ell-1}) - 2 - \sum_{i=2}^{\ell-1} t_i - 2t_\ell$$

e os descendentes serão da forma

$$F(t_1, \cdots, t_\ell) = \begin{pmatrix} 2 + \sum_{i=2}^{\ell-1} t_i \\ m(t_1, \cdots, t_\ell) \end{pmatrix} \uplus D(t_1, \cdots, t_\ell) \uplus A(t_1, t_2)$$

para $t_\ell = 0, 1, \cdots, \lfloor \frac{m(t_1, \cdots, t_{\ell-1}, 0)}{2} \rfloor$.

Exemplo 2.1. *Vamos descobrir a denominação de uma matriz genérica, tomemos a representação matricial de uma partição de 32 a seguir.*

$$\begin{pmatrix} 6 & 5 & 5 & 3 & 1 & 1 & 1 & 1 & 1 \\ 3 & 1 & 0 & 2 & 2 & 0 & 0 & 0 & 0 \end{pmatrix}$$

A sequência de zeros no final da linha inferior é feita de 4 zeros, isto é, $t_1 = 4$*, e sabemos que essa matriz é descendente de* $B(4)$*, como temos 9 colunas, e* $t_1 = 4$*, sabemos que essa matriz pertence à* $5^{\underline{a}}$ *geração (*$\ell + t_1$ *é o número de colunas), isto é, ela é da forma* $F(t_1, t_2, t_3, t_4, t_5)$*. Já sabemos* t_1*, como o primeiro elemento não nula na linha inferior é 2, temos* $t_2 = 2 - 1 = 1$*, e os elementos anteriores a este na linha inferior, lidos da direita*

para a esquerda, com exceção do primeiro elemento da linha, formam os outros valores: $t_3 = 2$, $t_4 = 0$, $t_5 = 1$.

Dessa forma, a matriz acima é $F(4, 1, 2, 0, 1)$, *e representa a partição* $9 + 6 + 5 + 5 + 3 + 1 + 1 + 1 + 1$ *de 32.*

Exemplo 2.2. *Agora tomemos uma denominação e vamos retornar à matriz que ela se refere.representa. Vamos supor que* $F(3, 2, 0, 4)$ *é uma representação matricial de uma partição de 30.*

Primeiro, vejamos que ela depende de t_1, t_2, t_3 *e* t_4, *assim sendo uma matriz pertencente à* $4^{\underline{a}}$ *geração. Como* $t_1 = 3$, *ela tem as 3 últimas colunas de 1 na linha superior 0 na linha inferior, sendo um total de* $3 + 4 = 7$ *colunas.*

$$\begin{pmatrix} a_1 & a_2 & a_3 & 1 & 1 & 1 & 1 \\ b_1 & b_2 & b_3 & b_4 & 0 & 0 & 0 \end{pmatrix}$$

Por construção, $b_4 = 1 + t_2$, $b_3 = t_3$ *e* $b_2 = t_4$*:*

$$\begin{pmatrix} a_1 & a_2 & a_3 & 1 & 1 & 1 & 1 \\ b_1 & 4 & 0 & 3 & 0 & 0 & 0 \end{pmatrix}$$

logo,

$$\begin{pmatrix} 8 & 4 & 4 & 1 & 1 & 1 & 1 \\ b_1 & 4 & 0 & 3 & 0 & 0 & 0 \end{pmatrix}$$

Os elementos somam $27 + b_1$, *que deve ser 30, logo,* $b_1 = 3$, *obtendo:*

$$F(3, 2, 0, 4) = \begin{pmatrix} 8 & 4 & 4 & 1 & 1 & 1 & 1 \\ b_1 & 4 & 0 & 3 & 0 & 0 & 0 \end{pmatrix}$$

Se o último elemento encontrado (b_1*) fosse negativo, concluiríamos que não há uma matriz com essa denominação dentre as representações matriciais de 30, e nossa suposição inicial estaria errada.*

Partindo para o cálculo dos valores de quantas matrizes existem em cada geração, $\mathbb{D}_\ell$, definamos $T_j^{(\ell)}$, para $j = 2, \cdots, \ell - 1$, que é um valor necessário para o cálculo de $\mathbb{D}_\ell$, da seguinte forma:

$$T_j^{(\ell)} = T_j^{(\ell)}(t_1, t_2, \cdots, t_{j-1}) = \left\lfloor \frac{\mathcal{P}_\ell(t_1) - \sum_{i=\ell-j+3}^{\ell} i \cdot t_{\ell-i+2}}{\ell - (j-2)} \right\rfloor. \tag{2}$$

Agora podemos definir $\mathbb{D}_1 = r_1 + 1 = n$ (número de blocos, isto é, matrizes da primeira geração), e, usando $T_j^{(\ell)}$ calculamos $\mathbb{D}_\ell$, para $\ell = 2, \cdots, \ell_0$, definido como:

$$\mathbb{D}_\ell = \sum_{t_1=0}^{r_\ell} \sum_{t_2=0}^{T_2^{(\ell)}} \cdots \sum_{t_{\ell-1}=0}^{T_{\ell-1}^{(\ell)}} \left(\left\lfloor \frac{\mathcal{P}_\ell(t_1) - \sum_{i=3}^{\ell} i \cdot t_{\ell-i+2}}{2} \right\rfloor + 1 \right) \tag{3}$$

Por fim, obtido todo $\mathbb{D}_\ell$ para $\ell = 2, \cdots, \ell_0$, temos o número de matrizes em cada geração, logo, o total de matrizes na família é, simplesmente, a soma desses valores:

$$p(n) = \sum_{\ell=1}^{\ell_0} \mathbb{D}_\ell = (r_1 + 1) + \sum_{\ell=2}^{\ell_0} \mathbb{D}_\ell$$

A dedução destas fórmulas está em Godinho and Santos [2020]. Tomando-as definidas como acima, passamos agora a implementar esses cálculos, e o faremos em forma de funções aninhadas.

2.1.1 Cálculo de $T_j^{(\ell)}$

Partindo de (2), queremos implementar uma função que receba, de entrada, valores de j, ℓ e um vetor $\vec{t} = (t_1, \cdots, t_{j-1})$ e devolva $T_j^{(\ell)}(t_1, \cdots, t_{j-1})$, que deve ser natural.

Para o caso irrestrito, já sabemos a forma explícita de $\mathcal{P}_\ell(x)$, em especial neste caso podemos substituir na fórmula, o que nos deixa com

$$T_j^{(\ell)} = T_j^{(\ell)}(t_1, t_2, \cdots, t_{j-1}) = \left\lfloor \frac{n - 2\ell - t_1 - \sum_{i=\ell-j+3}^{\ell} i \cdot t_{\ell-i+2}}{\ell - (j-2)} \right\rfloor.$$

Para a implementação da função, podemos perceber que t_i é equivalente a $\vec{t}(i)$, para $i = 1, \cdots, j-1$.

Algoritmo 1: Função $T_j^{(\ell)}(t_1, \cdots, t_{j-1})$

Entrada: $\ell \in \mathbb{N}$, $\ell \geq 2$, $j \in \{2, 3, \cdots, \ell - 1\}$ e $\vec{t} = (t_1, \cdots, t_{j-1})$, vetor de tamanho $j-1$, n

1 $s = 0$;

2 **Para** $i = \ell - j + 3$ *até* ℓ **faça:**

3 $\quad s = s + i \cdot t(\ell - i + 2)$;

4 $T = \dfrac{n - 2\ell - t(1) - s}{\ell - (j-2)}$;

5 $T = floor(T)$ (toma o maior inteiro menor do que ou igual a T);

Saída: $T_j^{(\ell)}(t_1, \cdots, t_{j-1}) = T$

2.1.2 Cálculo de $\mathbb{D}_\ell$

Partindo de (3), queremos implementar uma função que receba, de entrada, valores de ℓ e r_ℓ (raiz de $\mathcal{P}_\ell(x)$), e nos devolva o valor de $\mathbb{D}_\ell$.

Para o caso irrestrito, já sabemos a forma explícita de $\mathcal{P}_\ell(x)$, em especial neste caso podemos substituir na fórmula, o que nos deixa com

$$\mathbb{D}_\ell = \sum_{t_1=0}^{r_\ell} \sum_{t_2=0}^{T_2^{(\ell)}} \cdots \sum_{t_{\ell-1}=0}^{T_{\ell-1}^{(\ell)}} \left(\left\lfloor \frac{n - 2\ell - t_1 - \sum_{i=3}^{\ell} i \cdot t_{\ell-i+2}}{2} \right\rfloor + 1 \right).$$

Façamos o somatório por recursão, pois o número de somatórios é variável, para isso, vamos dar como entrada o vetor, que carrega os valores atuais de $t_1, t_2, \cdots, t_\ell$.

Então vamos criar uma função 'soma', que recebe ...

Algoritmo 2: Função soma

Entrada: $\ell \in \mathbb{N}$, $\ell \geq 2$, r_ℓ, $\vec{t} = (t_1, \cdots, t_{\ell-1})$, j, $somatorio$, n

Se $j = 1$ **então**
    $T = r_\ell$;
Senão
    $T = T_j^{(\ell)}(\vec{t})$;
Para $t(j) = 0$ *a* T **faça:**
    Se $j = \ell - 1$ **então**
        $somainterna = 0$;
        Para $i = 3$ *até* ℓ **faça:**
            $somainterna = somainterna + i \cdot t(\ell - i + 2)$;
        $a = \dfrac{n - 2 \cdot \ell - t(1) - somainterna}{2}$;
        $somatorio = somatorio + floor(a) + 1$;
    Senão
        $somatorio = soma(\vec{t}, \ell, r_\ell, somatorio, j + 1, n)$;

Saída: $somatorio$, valor a ser utilizado na função $\mathbb{D}_\ell$

E então a função que calcula $\mathbb{D}_\ell$ fica:

Algoritmo 3: Função $\mathbb{D}_\ell$

Entrada: $\ell \in \mathbb{N}$, $\ell \geq 2$, n, r_ℓ

$j = 1$;
$somatorio = 0$;
$t = (0, 0, \cdots, 0)$ de $\ell - 1$ dimensões;
$somatorio = \text{soma}(\vec{t}, \ell, r_\ell, somatorio, j)$;

Saída: $somatorio = \mathbb{D}_\ell$

2.1.3 Algoritmo final

Com a função que calcula $\mathbb{D}_\ell$, basta somarmos os valores de $\mathbb{D}_\ell$ para $\ell = 2, \cdots, \ell_0$.

Algoritmo 4: Cálculo de $p(n)$

Entrada: n

$\ell_0 = \left\lfloor \frac{n}{2} \right\rfloor$;

Defina r vetor de ℓ_0 zeros;

$r(1) = n - 1$

$p = r(1) + 1$

Para $\ell = 2$ *até* ℓ_0 **faça:**

  $r(\ell) = n - 2 \cdot \ell$;

  $d = \mathbb{D}_\ell(n, r_\ell)$ usando a função;

  $p = p + d$;

Saída: $p = p(n)$

2.1.4 Exemplos

Para entender melhor o que significam esses resultados na prática, vamos montar dois casos de exemplo, um primeiro, pequeno, para $n = 5$, mostrando todas as matrizes de acordo com o Exemplo 1.1, e a árvore formada.

Exemplo 2.3. *Tomando $n = 5$, como no Exemplo 1.1, vamos identificar as matrizes de $\mathbb{M}(5)$ que são blocos:*

$$B(0) = \begin{pmatrix} 1 \\ 4 \end{pmatrix}; B(1) = \begin{pmatrix} 1 & 1 \\ 3 & 0 \end{pmatrix}; B(2) = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 0 & 0 \end{pmatrix};$$
$$B(3) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}; B(4) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Os primeiros descendentes de $B(0)$ são, para $t_2 \geq 0$ da forma:

$$F(0, t_2) = \begin{pmatrix} 2 + t_2 & 1 \\ n - 2t_2 - 4 & t_2 + 1 \end{pmatrix}.$$

Cuja única matriz que se encaixa é:

$$F(0,0) = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}.$$

Fazemos de maneira semelhante para descobrir os descendentes de $B(1)$*, vendo quais valores* t_2 *se encaixam na matriz de forma que o segundo elemento da linha inferior seja* $t_2 + 1$ *e a matriz ainda pertença a* $\mathbb{M}(5)$*:*

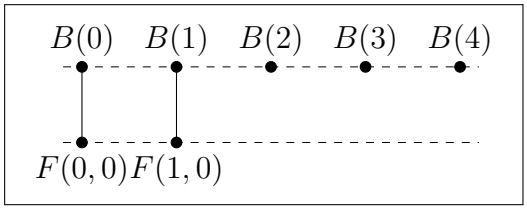
$$F(1,t_2) = \begin{pmatrix} 2+t_2 & 1 & 1 \\ n-2t_2-5 & t_2+1 & 0 \end{pmatrix}.$$

Cuja única matriz que se encaixa é:

$$F(1,0) = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Os blocos $B(3)$*,* $B(4)$ *e* $B(5)$ *não tem primeiros descendentes, isto é, não tem valores* t_2 *que, ao adicionar* $t_2 + 1$ *no segundo elemento de linha inferior desses blocos e fazendo as alterações necessárias, as matrizes resultantes ainda pertençam ao grupo* $\mathbb{M}(5)$*.*

Assim, temos listados todos as matrizes do grupo, e podemos montar a árvore:

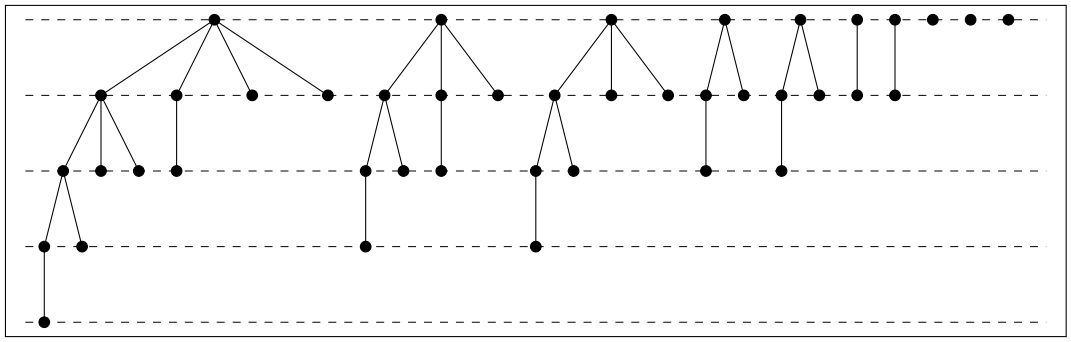


Assim, podemos ver as gerações, a primeira geração, a geração dos blocos $B(t_1)$*, tem 5 matrizes (*$n = 5$*), assim,* $\mathbb{D}_1 = 5$*, a segunda geração, dos descendentes diretos dos blocos,* $F(t_1, t_2)$*, tem 2 matrizes, assim,* $\mathbb{D}_2 = 2$*.*

Logo, $p(n = 5) = \mathbb{D}_1 + \mathbb{D}_2 = n + \mathbb{D}_2 = 5 + 2 = 7$*.*

Este exemplo simples conseguimos mostrar todas as matrizes e a maneira que são escolhidos os descendentes, entretanto, por terem poucos descendentes, a árvore não fica muito visível, para tal, vamos pegar um exemplo maior, $n = 10$, $p(10) = 42$.

Exemplo 2.4. *Neste exemplo, maior, o caso* $n = 10$*, vamos apenas apresentar as matrizes, mas os descendentes foram encontrados da maneira descrita no texto.*



Estas são as árvores dos descendentes de cada bloco, inclusive os casos de $B(7)$*,* $B(8)$ *e* $B(9)$ *que não tem descendentes, são árvores. A primeira linha contém os blocos, as linhas seguintes seus descendentes, os nomes das matrizes são listados na tabela abaixo:*

Geração	*Matrizes*	$\mathbb{D}_\ell$
1	*B(0), B(1), B(2), B(3), B(4), B(5), B(6), B(7), B(8), B(9)*	$\mathbb{D}_1 = n = 10$
2	*F(0,0), F(0,1), F(0,2), F(0,3), F(1,0), F(1,1), F(1,2), F(2,0), F(2,1), F(2,2), F(3,0), F(3,1), F(4,0), F(4,1), F(5,0), F(6,0)*	$\mathbb{D}_2 = 16$
3	*F(0,0,0), F(0,0,1), F(0,0,2), F(0,1,0), F(1,0,0), F(1,0,1), F(1,1,0), F(2,0,0), F(2,0,1), F(3,0,0), F(4,0,0)*	$\mathbb{D}_3 = 11$
4	*F(0,0,0,0), F(0,0,0,1), F(1,0,0,0), F(2,0,0,0)*	$\mathbb{D}_4 = 4$
5	*F(0,0,0,0,0)*	$\mathbb{D}_5 = 1$

Que completam, na soma, as 42 partições irrestritas de 10.

2.2 Caso Restrito

No caso restrito para as restrições que estamos trabalhando, isto é $\mathbb{P}(n, c, \lambda)$, com $n, c, \lambda \in \mathbb{N}$, com $c < n - 1$, já vimos como encontrar os blocos e seus descendentes. Vamos então ver um exemplo de como encontrar uma matriz na árvore, e montar uma árvore para demonstrar.

Exemplo 2.5. *Seja a matriz*

$$\begin{pmatrix} 26 & 23 & 18 & 14 & 11 & 8 & 5 \\ 5 & 0 & 2 & 1 & 0 & 0 & 0 \end{pmatrix},$$

seu último elemento da primeira linha é 5, isto é, $c = 5$ *é a menor parte, além disso, a diferença entre as partes é* $\lambda = 3$*, pois* $8 = 5 + \lambda + 0$*. Somando todas as entradas da matriz temos* $n = 113$*, portanto, temos que essa matriz pertence à* $\mathbb{M}(113, 5, 3)$*.*

Como a matriz tem três zeros no fim da segunda linha, isto é, $t_1 = 3$*, ela é uma descendente do bloco* $B(3)$*, como temos 7 colunas, sabemos que essa matriz pertence à* $4^{\underline{a}}$ *geração da árvore, isto é, ela é da forma* $F(3, t_2, t_3, t_4)$*.*

O último elemento não nulo é $t_2 + 1$*, para confirmar que o fim da matriz seja* $A(t_1, t_2)$*, logo,* $t_2 = 0$*. Agora, os elementos* b_2 *e* b_3 *são justamente* t_4 *e* t_3*. Assim, concluímos que a matriz dada é denominada* $F(3, 0, 2, 0) \in \mathbb{M}(113, 5, 3)$*.*

Observação: *Essa matriz representa a partição* $\{31, 23, 20, 15, 11, 8, 5\}$*, que é uma partição que também pertence ao conjunto de partições irrestritas de 113, e também pertence a* $\mathbb{P}(113, 1, 1)$ *(conjunto de partições com partes distintas), e alguns outros, porém, em cada um desses conjuntos que ela seja referenciada, ela terá uma representação matricial distinta. A partição é a mesma, mas a representação matricial depende das restrições.*

Agora que sabemos identificar onde uma determinada matriz se encontra na árvore, e sabemos montar as matrizes da árvore, vamos dar um exemplo de uma árvore.

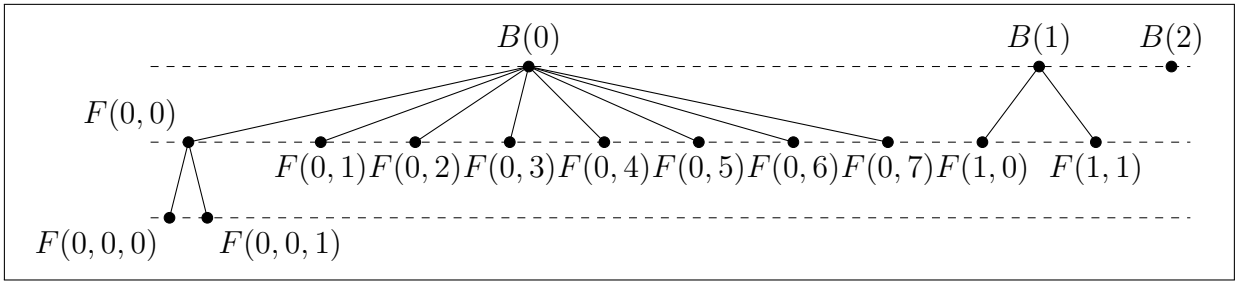
Exemplo 2.6. *Tomemos* $\mathbb{M}(32, 7, 2)$*, o primeiro passo é montar os blocos, que são os três abaixo, perceba que ao tentar montar um quarto, precisaríamos de um termo* -8*, o que não pode ocorrer (essa é uma maneira prática de montar a árvore, aumentar até aparecer um valor negativo).*

$$B(0) = \begin{pmatrix} 7 \\ 25 \end{pmatrix}; B(1) = \begin{pmatrix} 9 & 7 \\ 16 & 0 \end{pmatrix}; B(2) = \begin{pmatrix} 11 & 9 & 7 \\ 5 & 0 & 0 \end{pmatrix};$$

$$''B(3)'' = \begin{pmatrix} 13 & 11 & 9 & 7 \\ -8 & 0 & 0 & 0 \end{pmatrix}.$$

$B(2)$ não tem descendentes, pois $m(2,0) = -10 < 0$ (a maneira analítica de verificar). $B(1)$ tem dois descendentes, pois $m(1,1) = 1 > 0$, mas $m(1,2) = -1 < 0$. Por fim, checamos também que $B(0)$ tem 8 descendentes, pois $m(0,8) = -2 < 0$ é o primeiro $m(0,t_2)$ negativo.

Da mesma maneira, vemos que, da 2ª geração, apenas $F(0,0)$ tem descendentes, e são dois, obtemos assim a árvore de $\mathbb{M}(32,7,2)$:



E, portanto, $\#\mathbb{M}(32,7,2) = 15$.

3 Resultados

3.1 Caso irrestrito

O principal resultado é o próprio algoritmo mostrado acima, implementado em funções em MATLAB, as implementações estão no anexo, para não poluir o texto. Fazendo um laço em n dessa função, conseguimos analisar melhor o comportamento do algoritmo.

Além do valor de $p(n)$, coletamos o tempo de processamento do algoritmo para os valores calculados, que foram de $n = 1$ até $n = 167$.

Primeiro de tudo, o gráfico de $n \times p(n)$ comprova que a função $p(n)$ realmente explode para n crescente:

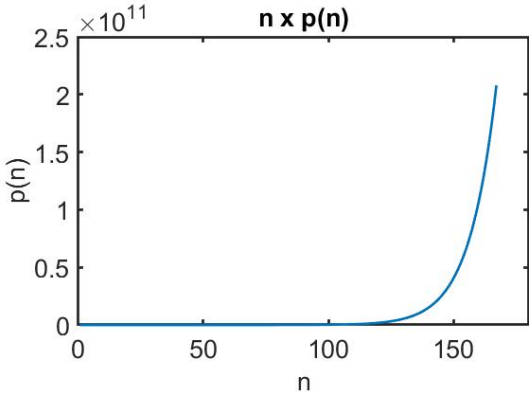
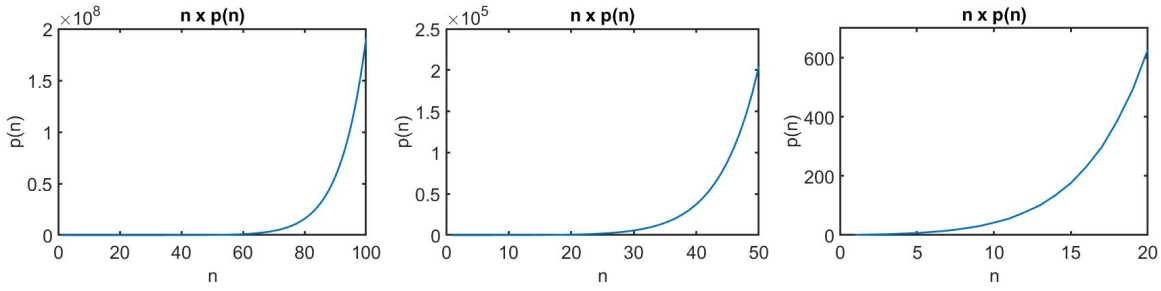


Figura 1: $n \times p(n),\ n = 1 : 167$

Porém, é necessário dividir o gráfico em três menores, pois no Gráfico 1, $p(100)$ aparenta ser um número próximo a 0, quando, na verdade, é da ordem de 10^8. Tomamos, então, os sub-intervalos $1 : 100$, $1 : 50$ e $1 : 20$ e obtivemos os seguintes gráficos:



(a) $n \times p(n),\ n = 1 : 100$ (b) $n \times p(n),\ n = 1 : 50$ (c) $n \times p(n),\ n = 1 : 20$

Figura 2: $n \times p(n)$ para sub-intervalos

Tomando o gráfico log-log com estes mesmos valores, temos o gráfico 3 abaixo, que mostra que o comportamento da função é, não só extremamente crescente, mas mais crescente que exponencial.

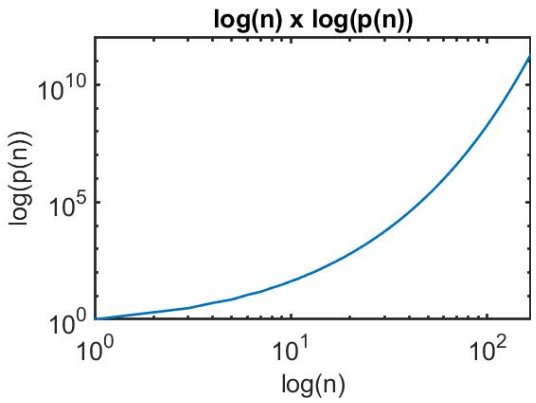


Figura 3: $\log n \times \log p(n),\ n = 1 : 167$

Analisando agora, a velocidade do algoritmo, temos que o tempo gasto para achar cada resultado é aproximadamente proporcional ao próprio resultado $p(n)$, como o gráfico abaixo mostra (aproximadamente uma reta).

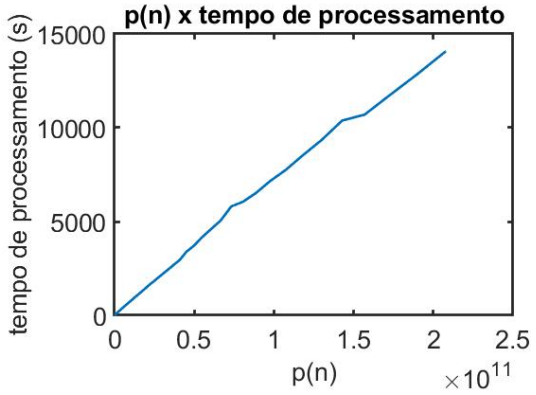


Figura 4: $p(n)\times$tempo de processamento, $n = 1 : 167$

Isto significa que, assim como a função $p(n)$, o tempo de processamento também cresce muito rapidamente, o gráfico abaixo, de n em função do tempo, nos dá uma boa noção deste crescimento.

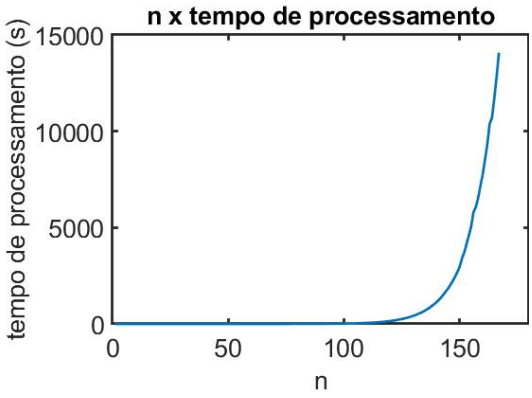


Figura 5: $n\times$tempo de processamento, $n = 1 : 167$

Para simplificar a compreensão, a tabela abaixo mostra intervalos de tempo, e os valores de n cujo processamento se encaixa no intervalo.

Tempo de Processamento	Valores de n
Até 1 segundo	$n = 1 : 76$
De 1 segundo a 1 minuto	$n = 77 : 111$
De 1 minuto até 1 hora	$n = 112 : 151$
Mais de 1 hora	$n = 152 : 167$

Tabela 1: Intervalos de Tempo de Processamento

Com ela, percebe-se que os cálculos se tornam cada vez menos praticáveis para um computador simples.

4 Conclusão

Conclui-se que o algoritmo acima calcula o número de partições irrestritas de n, como era seu objetivo. Conclui-se também que a função $p(n)$ tem um comportamento explosivo, cresce muito rapidamente.

Além disso, é possível concluir que o algoritmo é eficiente e viável, pois calcula valores até $p(111)$ em menos de um minuto. Entretanto, o algoritmo perde valor para grandes valores de n, já que o tempo se torna impraticável, isto é, o algoritmo tem um 'limite', que depende de quanto tempo o computador pode estar alocado apenas para cálculo desta função.

Já para o caso restrito, o calculo número de partições de n com menor parte

no mínimo c e diferença mínima entre partes λ por meio de sua representação matricial alcançou o que foi proposto.

5 Anexos

5.1 Implementação do cálculo de $T_j^{(\ell)}$

Implementação em MATLAB referente ao algoritmo 1.

Listing 1: Cálculo de $T_j^{(\ell)}$

```
function T = Tj(j,l,t,n)
    s = 0;
    for i = (l-j+3):l
        s = s+i*t(l-i+2);
    end
    T = (n-2*l-t(1)-s)/(l-j+2);
    T = floor(T);
end
```

5.2 Implementação da função soma

Implementação em MATLAB referente ao algoritmo 2.

Listing 2: Função Soma

```
function somatorio = soma(t,l,rl,somatorio,j,n)
    if j == 1
        T = rl;
    else
        T = Tj(j,l,t,n);
    end
    for tj = 0:T
        t(j) = tj;
        if j == l-1
            somainterna = 0;
            for i = 3:l
                somainterna = somainterna + i*t(l-i+2);
            end
```

```
14              a = (n-2*l-t(1)-somainterna)/2;
15              somatorio = somatorio + floor(a) + 1;
16          else
17              somatorio = soma(t,l,rl,somatorio,j+1,n);
18          end
19      end
20  end
```

5.3 Implementação do cálculo de $\mathbb{D}_\ell$

Implementação em MATLAB referente ao algoritmo 3.

Listing 3: Cálculo de $\mathbb{D}_\ell$

```
1  function somatorio = D(l,rl,n)
2      j = 1;
3      somatorio = 0;
4      t = zeros(l-1,1);
5      somatorio = soma(t,l,rl,somatorio,j,n);
6  end
```

5.4 Implementação da função principal: cálculo de $p(n)$

Implementação em MATLAB referente ao algoritmo 4.

Listing 4: Cálculo de $p(n)$

```
1  function p = particao(n)
2      %Valores iniciais
3      lzero = floor(n/2);
4      r = zeros(lzero,1);
5      r(1) = n-1;
6      p = r(1)+1;
7
8      %Laco para somar D2, D3, ..., Dlzero
9      for l=2:lzero
```

```
10          r(l) = n-2*l;
11          d=D(l,r(l),n);
12          p = p+d;
13      end
14  end
```

Bibliografia

Hemar Godinho and José Plínio O. Santos. A family of partitions equinumerous with the set of nodes of a family of trees. *INTEGERS*, 20, 2020. URL http://math.colgate.edu/~integers/u101/u101.pdf.

José Plínio O. Santos and Marilia L. Matte. A new approach to integer partition. *Bulletin of the Brazilian Mathematical Society*, 2018. doi: https://doi.org/10.1007/s00574-018-0082-z.

José Plínio O. Santos, Margarida P. Mello, and Idani T.C. Murari. *Introdução à Análise Combinatória*. Editora Ciência Moderna Ltda., Rio de Janeiro, 2007.

José Plínio O. Santos, Paulo Mondek, and Andréia C. Ribeiro. New two-line arrays representing partitions. *Annals of Combinatorics*, 15(341), 2011. doi: https://doi.org/10.1007/s00026-011-0099-0.