

## 1. INTRODUÇÃO

1.1. **Sobre o Trabalho.**

1.2. **Notação.**

1.3. **Matrizes.** Explicar o que são matrizes online e offline.

1.4. **Implementações.** Explicar os padrões que estou usando pra implementar os programas. Por exemplo: Funções como argumentos, 0-index (em contraste com o 1-index do pseudo-código) e wrappers.

## 2. MONOTONICIDADE, CONVEXIDADE E MATRIZES MONGE

Aqui serão apresentados e explorados os conceitos de monotonicidade, convexidade e matrizes Monge, além disso, alguns resultados referentes a estes conceitos serão demonstrados. Estes conceitos são fundamentais para o desenvolvimento do restante do trabalho.

**Definição 2.1** (Vetor monótono). *Seja  $a \in \mathbb{Q}^n$  um vetor,  $a$  é dito monótono quando vale uma das propriedades abaixo.*

- Se para todo  $i, j \in [n]$ ,  $i < j \Rightarrow a_i \leq a_j$ ,  $a$  é dito monótono crescente (ou só crescente).
- Se para todo  $i, j \in [n]$ ,  $i < j \Rightarrow a_i \geq a_j$ ,  $a$  é dito monótono decrescente (ou só decrescente).

Sabemos que a monotonicidade de vetores pode ser aproveitada para agilizar alguns algoritmos importantes, por exemplo, a busca binária pode ser interpretada como uma otimização da busca linear para vetores monótonos.

**Definição 2.2** (Função convexa). *Seja  $g : \mathbb{Q} \rightarrow \mathbb{Q}$  uma função,*

- se para todo par de pontos  $x, y \in \mathbb{Q}$  e  $\lambda \in \mathbb{Q}$  que respeita  $0 \leq \lambda \leq 1$ , vale  $g(\lambda x + (1 - \lambda)y) \leq \lambda g(x) + (1 - \lambda)g(y)$ ,  $g$  é dita convexa e
- se para todo par de pontos  $x, y \in \mathbb{Q}$  e  $\lambda \in \mathbb{Q}$  que respeita  $0 \leq \lambda \leq 1$ , vale  $g(\lambda x + (1 - \lambda)y) \geq \lambda g(x) + (1 - \lambda)g(y)$ ,  $g$  é dita côncava.

**Proposição 2.3.** *A função  $g(x) = x^2$  é convexa.*

*Demonstração.* Sejam  $x, y, \lambda \in \mathbb{Q}$  onde vale  $0 \leq \lambda \leq 1$ . Queremos provar  $(\lambda x + (1 - \lambda)y)^2 \leq \lambda x^2 + (1 - \lambda)y^2$ , isso equivale a

$$\begin{aligned} \lambda^2 x^2 + (1 - \lambda)^2 y^2 + 2\lambda(1 - \lambda)xy &\leq \lambda x^2 + (1 - \lambda)y^2, \text{ ou seja} \\ (\lambda^2 - \lambda)(x^2) + ((1 - \lambda)^2 - (1 - \lambda))y^2 + 2(\lambda - \lambda^2)xy &\leq 0, \text{ que é} \\ (\lambda^2 - \lambda)(x^2 + y^2 - 2xy) = (\lambda^2 - \lambda)(x + y)^2 &\leq 0. \end{aligned}$$

□

É interessante definir convexidade também em termos de vetores.

**Definição 2.4** (Vetor convexo). *Seja  $a \in \mathbb{Q}^n$  um vetor,*

- se para todo  $i, j, k \in [n]$ ,  $i < j < k \Rightarrow a_j \leq \frac{(j-k)a_i + (i-j)a_k}{i-k}$ ,  $a$  é dito convexo e
- se para todo  $i, j, k \in [n]$ ,  $i < j < k \Rightarrow a_j \geq \frac{(j-k)a_i + (i-j)a_k}{i-k}$ ,  $a$  é dito côncavo.

Assim como a monotonicidade, a convexidade também é usualmente explorada para agilizar algoritmos, por exemplo, se um vetor é convexo podemos definir o valor mínimo do vetor com uma busca ternária ao invés de percorrer todo o vetor.

**Definição 2.5.** *Seja  $A \in \mathbb{Q}^{n \times m}$ , definimos quatro vetores a seguir.*

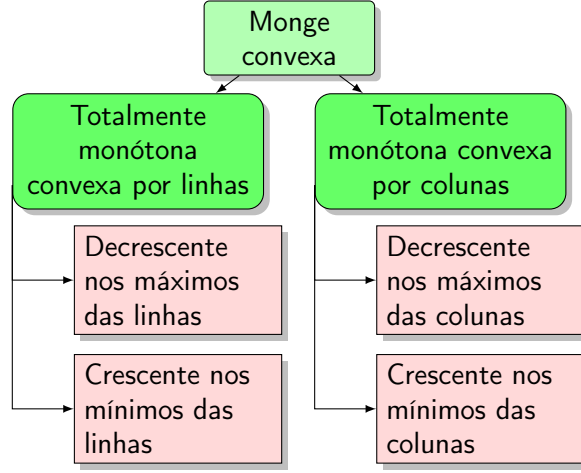


FIGURA 2.6. Comportamento dos vetores de índices ótimos em relação à convexidade.

- O vetor de índices de máximos das linhas de  $A$  guarda na posição  $i$  o número  $\max\{j \in [m] \mid A[i][j] \geq A[i][j'] \text{ para todo } j' \in [m]\}$ .
- O vetor de índices de mínimos das linhas de  $A$  guarda na posição  $i$  o número  $\min\{j \in [m] \mid A[i][j] \leq A[i][j'] \text{ para todo } j' \in [m]\}$ .
- O vetor de índices de máximos das colunas de  $A$  guarda na posição  $j$  o número  $\max\{i \in [n] \mid A[i][j] \geq A[i'][j] \text{ para todo } i' \in [n]\}$ .
- O vetor de índices de mínimos das colunas de  $A$  guarda na posição  $j$  o número  $\min\{i \in [n] \mid A[i][j] \leq A[i'][j] \text{ para todo } i' \in [n]\}$ .

Note que o máximo de uma linha (ou coluna) foi definido como o maior índice que atinge o máximo e o de mínimo foi definido como o menor índice que atinge o mínimo. Esta escolha foi feita para simplificar o Lema 2.9.

Dada uma matriz, encontrar estes vetores é um problema central para este trabalho. Neste momento é interessante classificar algumas matrizes de acordo com propriedades que vão nos ajudar a calcular os vetores de mínimos e máximos de maneira especialmente eficiente.

A Figura 2.6 resume as relações de implicação da classificação que será realizada. Os conceitos ilustrados nela serão apresentados a seguir.

**Definição 2.7** (Matriz monótona). *Seja  $A \in \mathbb{Q}^{n \times m}$  uma matriz. Se  $A$  tiver o vetor de índices de máximos das linhas monótono,  $A$  é dita monótona nos máximos das linhas.*

*Valem também as definições análogas para mínimos ou colunas e pode-se especificar monotonicidade crescente ou decrescente.*

**Definição 2.8** (Matriz totalmente monótona). *Seja  $A \in \mathbb{Q}^{n \times m}$  uma matriz.*

- *Se  $A[i'][j] \leq A[i'][j']$  implica  $A[i][j] \leq A[i][j']$  para todo  $1 \leq i < i' \leq n$  e  $1 \leq j < j' \leq m$ ,  $A$  é monótona convexa nas linhas.*

- Se  $A[i][j'] \leq A[i'][j']$  implica  $A[i][j] \leq A[i'][j]$  para todo  $1 \leq i < i' \leq n$  e  $1 \leq j < j' \leq m$ ,  $A$  é monótona convexa nas colunas.
- Se  $A[i'][j] > A[i][j]$  implica  $A[i][j'] > A[i'][j']$  para todo  $1 \leq i < i' \leq n$  e  $1 \leq j < j' \leq m$ ,  $A$  é monótona côncava nas linhas.
- Se  $A[i][j'] > A[i'][j']$  implica  $A[i][j] > A[i'][j]$  para todo  $1 \leq i < i' \leq n$  e  $1 \leq j < j' \leq m$ ,  $A$  é monótona côncava nas colunas.

O motivo do uso dos termos “convexa” e “côncava” em relação a matrizes durante o texto são justificados pelo Teorema 2.15. Note que se uma matriz é totalmente monótona, todas as suas submatrizes são totalmente monótonas no mesmo sentido.

**Lema 2.9.** *Se  $A \in \mathbb{Q}^{n \times m}$  é uma matriz totalmente monótona convexa nas linhas, toda submatriz de  $A$  é monótona decrescente nos máximos das linhas e monótona crescente nos mínimos das linhas.*

*Se  $A$  é totalmente monótona côncava nas linhas, toda submatriz de  $A$  é monótona decrescente nos mínimos das linhas e monótona crescente nos máximos das linhas.*

*As afirmações valem identicamente em termos de colunas.*

*Demonstração.* Considere uma matriz  $A$  totalmente monótona convexa nas linhas. Sejam  $i$  e  $i'$  índices de linhas de  $A$  onde  $i < i'$ . Chamamos de  $j$  o índice de máximo da linha  $i$  e de  $j'$  o índice de máximo da linha  $i'$ . Queremos provar que os máximos são decrescentes, portanto, vamos supor por absurdo que  $j < j'$ . Com isso, teremos  $A[i][j'] < A[i][j]$  e  $A[i'][j] \leq A[i'][j']$ . Porém, já que  $A$  é monótona convexa nas linhas, a segunda desigualdade implica em  $A[i][j] \leq A[i][j']$ , que contradiz a primeira. Portanto, os índices de máximos são decrescentes.

Agora, considere novamente dois índices  $i$  e  $i'$  quaisquer de linhas de  $A$  onde  $i < i'$ . Denotamos por  $j$  o índice de mínimo da linha  $i'$  e por  $j'$  o índice de mínimo da linha  $i$  (note e a inversão no uso de  $'$ ). Vamos supor por absurdo que  $j < j'$  e teremos  $A[i'][j] \leq A[i'][j']$  e  $A[i][j'] < A[i][j]$ . E, novamente, usando o fato de que  $A$  é monótona convexa nas linhas, obtivemos uma contradição.

Finalmente, se  $A'$  é uma submatriz de  $A$ , então  $A'$  é totalmente monótona convexa nas linhas, portanto monótona crescente nos máximos das linhas e monótona decrescente nos mínimos das linhas.

As demonstrações no caso côncavo e nos casos relacionados a colunas são análogas.  $\square$

**Definição 2.10** (Monge Convexidade). *Seja  $A \in \mathbb{Q}^{n \times m}$ .*

- (1) *Se vale  $A[i][j] + A[i'][j'] \leq A[i][j'] + A[i'][j]$  para todo  $1 \leq i < i' \leq n$  e  $1 \leq j < j' \leq m$ ,  $A$  é dita Monge convexa.*
- (2) *Se vale  $A[i][j] + A[i'][j'] \geq A[i][j'] + A[i'][j]$  para todo  $1 \leq i < i' \leq n$  e  $1 \leq j < j' \leq m$ ,  $A$  é dita Monge côncava.*

A desigualdade que define as matrizes Monge é conhecida também por “Condição de Monge” ou “Desigualdade Quadrangular”[4][2].

**Lema 2.11.** *Se  $A$  é Monge convexa,  $A$  é totalmente monótona convexa tanto nas linhas quanto nas colunas.*

*Se  $A$  é Monge côncava,  $A$  é totalmente monótona côncava tanto nas linhas quanto nas colunas.*

*Demonstração.* Seja  $A$  uma matriz Monge convexa. Suponha que vale, para certos  $i, i' \in [n]$  e  $j, j' \in [m]$  onde  $i < i'$  e  $j < j'$ ,  $A[i][j] \leq A[i'][j']$ , então, somamos esta desigualdade à definição de Monge convexa e obtemos  $A[i][j] \leq A[i][j']$ , ou seja,  $A$  é totalmente monótona convexa nas linhas.

Por outro lado, se vale, para certos  $i, i' \in [n]$  e  $j, j' \in [m]$  com  $i < i'$  e  $j < j'$ ,  $A[i][j'] \leq A[i'][j']$ , somamos esta desigualdade à definição de Monge convexa e obtemos  $A[i][j] \leq A[i'][j]$ , assim,  $A$  é totalmente monótona convexa nas colunas.

A prova para o caso côncavo é análoga.  $\square$

As matrizes Monge são usadas para resolver uma série de problemas que serão explorados aqui. A condição de Monge é a mais forte apresentada aqui. Alguns dos algoritmos apresentados não dependem dela, apenas da monotonicidade ou total monotonicidade, ainda assim, ela leva a resultados úteis que nos permitem provar a pertinência dos algoritmos a alguns problemas, mesmo que o algoritmo usado não se utilize da condição diretamente.

Como consequência desta utilidade, iremos discutir um problema que será resolvido com um algoritmo apresentado somente na Seção 4, o algoritmo SMAWK. Ele não será explicado neste momento, utilizamos ele como caixa preta. Isto é motivado pelo fato de que o pensamento apresentado aqui não é útil somente para o algoritmo SMAWK, ele é útil também em vários dos outros momentos deste trabalho.

**Problema 2.12.** *Dados dois inteiros  $k$  e  $n$  com  $k \leq n$ , um vetor de pesos  $a \in \mathbb{Q}_+^n$  e uma matriz de custos  $A[i][j] = \left( \sum_{k=i+1}^j a_k \right)^2$ . Queremos particionar o vetor  $a$  em  $k$  partes não-vazias de forma a maximizar a soma dos custos das partes, isto é, queremos escolher um vetor  $r \in \mathbb{N}^{[0..k]}$  de índices tal que  $1 = r_0 < r_1 < r_2 < \dots < r_{k-1} < r_k = n$  de forma que  $\sum_{i=1}^k A[r_{i-1}][r_i]$  seja máximo.*

Podemos resolver este problema com programação dinâmica. Vamos preencher a matriz  $E \in \mathbb{Q}^{k \times n}$  definida recursivamente para todo  $k' \in [k]$  e  $n' \in [n]$ :

$$E[k'][n'] = \begin{cases} A[1][n'] & , \text{ se } k' = 1, \\ \max_{i=k'-1}^{n'-1} E[k'-1][i] + A[i][n'] & , \text{ se } k' \leq n', \\ \text{indefinida} & , \text{ caso contrário.} \end{cases}$$

A recorrência acima nos dá em cada entrada  $E[k'][n']$  o maior valor possível alcançado particionando o vetor  $a[1..n']$  em  $k'$  partes não vazias. Podemos

preencher esta tabela trivialmente em tempo  $O(kn^2)$ , basta iterarar primeiro pelos índices  $k'$  crescentemente. O caso onde  $k' = 1$  é resolvido trivialmente e os casos maiores podem ser resolvidos, um por vez, testando todas as possibilidades de máximo para todo  $n'$ .

Fixamos um  $k' > 1$ . Vamos utilizar o algoritmo SMAWK para agilizar a solução deste subproblema. Este algoritmo é capaz de resolver o Problema 2.13 (descrito abaixo) em tempo  $O(n)$ . Precisamos provar, então, que o subproblema resolvido para cada  $k'$  é equivalente ao Problema 2.13.

**Problema 2.13.** *Dada uma matriz  $A \in \mathbb{A}^{n \times n}$  totalmente monótona convexa nas colunas, encontrar o vetor de máximos das colunas de  $A$ .*

Definimos a matriz  $B_{k'}$  para todo  $i, n' \in \mathbb{N}$  onde  $k' - 1 \leq i < n' \leq n$  como  $B_{k'}[i][n'] = E[k' - 1][i] + A[i][n']$ . Note que já que  $k'$  é fixo, já descobrimos os valores da entrada da matriz  $E$  na linha  $k' - 1$ . Encontrar o índice  $i$  que atinge o máximo em  $E[k'] [n']$  é exatamente encontrar o índice de máximo da coluna  $n'$  na matriz  $B_{k'}$ , formalmente,

$$\max_{i=k'-1}^{n'-1} B_{k'}[i][n'] = \max_{i=k'-1}^{n'-1} E[k' - 1][i] + A[i][n'].$$

Portanto, basta mostrar que  $B_{k'}$  é monótona convexa nas colunas. Para isso, vamos mostrar, com a ajuda dos resultados abaixo, que  $B_{k'}$  é Monge convexa.

**Lema 2.14.** *Sejam  $A, B \in \mathbb{Q}^{n \times m}$  matrizes e  $c \in \mathbb{Q}^n$  um vetor tais que para todo  $i \in [n]$  e  $j \in [m]$ ,  $B[i][j] = A[i][j] + c[i]$ . Se  $A$  é Monge convexa,  $B$  é Monge convexa.*

*O mesmo resultado vale se  $c \in \mathbb{Q}^m$  e  $B[i][j] = A[i][j] + c[j]$ .*

*Demonstração.* Sejam  $A, B$  e  $b$  definidos como no enunciado do teorema. Suponha que  $A$  é Monge convexa. Vale, para quaisquer  $1 \leq i < i' \leq n$  e  $1 \leq j < j' \leq m$ ,  $A[i][j] + A[i'][j'] \leq A[i'][j] + A[i][j']$ , logo, vale  $A[i][j] + b[i] + A[i'][j'] + b[i'] \leq A[i'][j] + b[i'] + A[i][j'] + b[i]$  que é  $B[i][j] + B[i'][j'] \leq B[i'][j] + B[i][j']$ . A prova para o caso onde  $c \in \mathbb{Q}^m$  e  $B[i][j] = A[i][j] + c[j]$  é análoga.  $\square$

Com este resultado, é fácil ver que, desde que  $A$  seja Monge convexa,  $B_1$  será Monge convexa e a Monge convexidade será mantida para todo  $B_{k'}$  com  $1 \leq k' \leq k$ . O teorema a seguir nos ajuda a mostrar que  $A$  é Monge convexa.

**Teorema 2.15.** *Sejam  $A \in \mathbb{Q}^{n \times n}$  uma matriz,  $w \in \mathbb{Q}_+^n$  um vetor e  $g : \mathbb{Q} \rightarrow \mathbb{Q}$  uma função tais que para todo  $i, j \in [n]$  vale  $A[i][j] = g\left(\sum_{k=1}^j w_k - \sum_{k=1}^i w_k\right)$ . Se  $g$  é convexa,  $A$  é Monge convexa.*

*Similarmente, se  $g$  é côncava,  $A$  é Monge côncava.*

Antes de apresentar uma prova para o teorema acima, vamos mostrar a utilidade dele no nosso problema atual, o que deve ajudar na compreensão de seu enunciado.

Queremos mostrar que  $A$  é Monge convexa, porém, para um certo vetor  $a \in \mathbb{Q}_+^n$ ,  $A[i][j] = \left( \sum_{k=(i+1)}^j a_k \right)^2 = \left( \sum_{k=1}^j a_k - \sum_{k=1}^i a_k \right)^2$ , por definição. Assim, precisamos mostrar apenas que a função  $g(x) = x^2$  é convexa, o que segue da Proposição 2.3. Agora, como explicado acima, todas as matrizes  $B_{k'}$  são Monge convexas e podemos aplicar o algoritmo SMAWK para todo  $k'$ , resolvendo o Problema 2.12 em tempo  $O(kn)$ .

Agora, nos resta provar o Teorema 2.15.

*Demonstração.* Sejam  $A$  e  $g$  quaisquer que respeitem as condições do enunciado. Sejam ainda  $i, i', j, j' \in [n]$  onde  $i < i'$  e  $j < j'$ . Escrevemos  $a = \sum_{k=1}^{i'} w_k - \sum_{k=1}^i w_k$ ,  $b = \sum_{k=1}^{j'} w_k - \sum_{k=1}^j w_k$  e  $z = \sum_{k=1}^j w_k - \sum_{k=1}^{i'} w_k$ . Desta forma, temos  $g(z) = A[i'][j]$ ,  $g(z+a+b) = A[i][j']$ ,  $g(z+a) = A[i][j]$  e  $g(z+b) = A[i'][j']$ , portanto,  $g(z+a) + g(z+b) \leq g(z) + g(z+a+b)$  se e somente se  $A[i][j] + A[i'][j'] \leq A[i][j'] + A[i'][j]$  ( $A$  é Monge convexa).

Com isso, vamos provar que se  $g$  é convexa,  $A$  é Monge convexa. Sejam  $i, i', j, j' \in [n]$  onde  $i < i'$  e  $j < j'$ . Definimos  $a$ ,  $b$  e  $z$  como acima. Sabemos  $0 \leq a$  e  $0 \leq b$ . Consideramos o caso onde  $0 < a \leq b$ . Temos  $0 < a \leq b < a+b$ , ou seja,  $z < z+a \leq z+b < z+a+b$ . Definimos  $\lambda = \frac{a}{a+b}$ . Já que  $z+a = \lambda z + (1-\lambda)(z+a+b)$  e  $z+b = (1-\lambda)z + \lambda(z+a+b)$ , por convexidade de  $g$ , obtemos  $g(z+a) \leq \lambda g(z) + (1-\lambda)g(z+a+b)$  e  $g(z+b) \leq \lambda g(z+a+b) + (1-\lambda)g(z)$ . Somando, obtemos  $g(z+a) + g(z+b) \leq g(z) + g(z+a+b)$ .

Se considerarmos o caso onde  $0 < b \leq a$ , seguimos o mesmo raciocínio e obtemos, novamente,  $g(z+a) + g(z+b) \leq g(z) + g(z+a+b)$ . Falta considerar o caso onde  $0 = a = b$ , neste caso,  $g(z) = g(z+a) = g(z+b) = g(z+a+b)$  e vale  $(z+a) + g(z+b) \leq g(z) + g(z+a+b)$ . Portanto,  $A$  é Monge convexa.  $\square$

### 3. DIVISÃO E CONQUISTA

Nesta seção será apresentada uma técnica que chamamos de Divisão e Conquista. A ideia é brevemente citada em [?] e é um tópico recorrente em competições de programação, sendo conhecida como “Divide and Conquer Optimization” [?] e geralmente aplicada a problemas de programação dinâmica. Além disso, as hipóteses deste algoritmo são mais fracas do que as do algoritmo SMAWK, apresentado na Seção 4, portanto, todo problema para o qual aquela solução pode ser aplicada, esta também pode. Ao final desta seção, apresentamos exemplos de aplicações em programação dinâmica.

Dada uma matriz  $A \in \mathbb{Q}^{n \times m}$ , listamos os casos de uso deste algoritmo:

- Se  $A$  é monótona crescente nos mínimos das linhas podemos encontrar os índices de mínimos das linhas em tempo  $\mathcal{O}((n+m) \lg(n))$ ,
- se  $A$  é monótona decrescente nos mínimos das linhas podemos encontrar os índices de mínimos das linhas em tempo  $\mathcal{O}((n+m) \lg(n))$ ,
- se  $A$  é monótona crescente nos máximos das linhas podemos encontrar os índices de máximos das linhas em tempo  $\mathcal{O}((n+m) \lg(n))$ ,
- se  $A$  é monótona decrescente nos máximos das linhas podemos encontrar os índices de máximos das linhas em tempo  $\mathcal{O}((n+m) \lg(n))$ ,
- se  $A$  é monótona crescente nos mínimos das colunas podemos encontrar os índices de mínimos das colunas em tempo  $\mathcal{O}((n+m) \lg(m))$ ,
- se  $A$  é monótona decrescente nos mínimos das colunas podemos encontrar os índices de mínimos das colunas em tempo  $\mathcal{O}((n+m) \lg(m))$ ,
- se  $A$  é monótona crescente nos máximos das colunas podemos encontrar os índices de máximos das colunas em tempo  $\mathcal{O}((n+m) \lg(m))$  e
- se  $A$  é monótona decrescente nos máximos das colunas podemos encontrar os índices de máximos das colunas em tempo  $\mathcal{O}((n+m) \lg(m))$ .

Apresentaremos o caso em que  $A$  é crescente nos mínimos das linhas. Na Subseção 3.4 explicamos como reduzir os problemas elencados para este caso.

**3.1. Técnica.** Dada uma matriz  $A \in \mathbb{Q}^{n \times m}$  monótona crescente nos máximos das linhas, queremos encontrar o vetor de índices de máximos das linhas de  $A$ . Isto é, para todo  $i \in [n]$ , queremos encontrar

$$R[i] = \min\{j \mid A[i][j] \geq A[i][j'] \text{ para todo } j' \in [n]\}.$$

Se, para alguma linha  $i$ , encontrarmos o valor  $R[i]$ , sabemos, já que  $A$  é monótona convexa, que para todo  $i' < i$ ,  $R[i'] \leq R[i]$  e, para todo  $i' > i$ ,  $R[i'] \geq R[i]$ , isto é, sabemos que os máximos de menor índice das outras linhas se encontram nas submatrizes  $A[1..i-1][1..R[i]]$  e  $A[i+1..n][R[i]..m]$ . Basta, agora, seguindo o paradigma de divisão e conquista, resolver o mesmo problema para estas submatrizes e conseguimos resolver o problema original.



---

**Algoritmo 3.1** Máximos das linhas com divisão e conquista

---

```

1: função FINDROWMAX_DC( $A, r_s, r_t, c_s, c_t$ )
2:    $\ell \leftarrow \lceil (r_s + r_t)/2 \rceil$ 
3:    $R[\ell] \leftarrow \min\{j \mid A[i][j] \geq A[i][j'] \text{ para todo } j' \in [c_s \dots c_t]\}$ 
4:   se  $i > r_s$  então
5:      $R[r_s \dots \ell - 1] \leftarrow \text{FINDROWMAX\_DC}(A, r_s, \ell - 1, c_s, R[\ell])$ 
6:   se  $i < r_t$  então
7:      $R[\ell + 1 \dots r_t] \leftarrow \text{FINDROWMAX\_DC}(A, \ell + 1, r_t, R[\ell], c_t)$ 
8:   devolve  $R$ 

```

---

**3.2. Análise.** Será feita uma análise do tempo de execução do algoritmo acima no pior caso assumindo que as atribuições feitas nas linhas 5 e 7 custam tempo constante, futuramente, em 3.3, iremos apresentar uma implementação em C++ que está de acordo com a análise realizada.

Se  $A$  é uma matriz e  $r_s, r_t, c_s$  e  $c_t$  são índices tais que  $r_t - r_s = n > 0$  e  $c_t - c_s = m > 0$ , o tempo gasto por  $\text{FINDROWMAX\_DC}(A, r_s, r_t, c_s, c_t)$  pode ser expresso pela seguinte recorrência:

$$T(n, m) = \begin{cases} m & , \text{ se } n = 1, \\ m + \max_{j \in [m]} T(1, j) & , \text{ se } n = 2, \\ m + \max_{j \in [m]} \left\{ T(\lceil n/2 \rceil - 1, m - j + 1) + T(\lfloor n/2 \rfloor, j) \right\} & , \text{ caso contrário.} \end{cases}$$

**Proposição 3.2.** Para todo  $n, m \geq 1$ ,  $T(n, m) \leq (m + n) \lg(2n)$  e, portanto, a técnica da divisão e conquista consegue encontrar o máximo de todas as linhas em tempo  $\mathcal{O}((m + n) \lg(n))$

*Demonstração.* Vamos usar indução em  $n$  para provar a tese. Se  $n = 1$  e  $m \geq 1$ ,  $T(1, m) = m \leq (m + 1) \lg(2)$ . Se  $n = 2$  e  $m \geq 1$ , existe  $j \in [m]$  tal que  $T(2, m) = m + r \leq 2m \leq (m + 2) \lg(4)$ . Agora, se  $n \geq 3$  e  $m \geq 1$ , existe um  $j \in [m]$  tal que

$$T(n, m) = m + T(\lceil n/2 \rceil - 1, j) + T(\lfloor n/2 \rfloor, m - j + 1).$$

Assumimos para  $1 \leq n' < n$  e  $m' \geq 1$  que  $T(n', m') \leq (m' + n') \lg(2n')$ . Com isso, já que  $1 \leq \lceil n/2 \rceil - 1 < n$ ,  $1 \leq \lfloor n/2 \rfloor < n$ ,  $j \geq 1$  e  $m - j + 1 \geq 1$ , temos, com a equação acima e o fato de que  $\lceil n/2 \rceil - 1 \leq \lfloor n/2 \rfloor \leq n/2$ ,

$$\begin{aligned} T(n, m) &\leq m + (j + \lceil n/2 \rceil - 1 + m - j + 1 + \lfloor n/2 \rfloor) \lg(n) \\ &= m + (m + n) \lg(n) < (m + n)(\lg(n) + 1) = (m + n) \lg(2n). \end{aligned}$$

□

**3.3. Implementação.** Para implementar o Algoritmo 3.1 com a complexidade desejada, devemos tomar cuidado com as atribuições feitas nas linhas 5 e 7. A forma como elas foram apresentadas sugere que os vetores  $R$  recebidos pelas funções sejam recebidos e copiados para o vetor  $R$ . Ao invés de fazer isso, passaremos o endereço do vetor  $R$  recursivamente e garantir

que cada chamada só complete o subvetor  $R[r_c \dots r_t]$ , referente a seu subproblema. Além disso, como explicado na Seção 1.4, a matriz  $A$  será passada como uma função e não como uma matriz.

A implementação em C++ do algoritmo apresentado, levando em conta as considerações acima, pode ser encontrada em `implementacao/FindRowMax_DC.cpp`.

Isso é chato de ser escrito, não basta transformar a matriz e usar as coisas como caixa preta por causa da forma como índices de máximos e mínimos foram definidos

### 3.4. Generalizações.

## 4. SMAWK

Nesta seção discutiremos o algoritmo SMAWK. Ele é conhecido pela sua aplicação no problema de encontrar o vértice mais distante de cada vértice num polígono convexo em tempo linear [1]. Ao final desta seção serão explicadas esta e outras aplicações deste algoritmo.

Dada uma matriz  $A \in \mathbb{Q}^{n \times m}$ , listamos os casos de uso deste algoritmo:

- Se  $A$  é totalmente monótona convexa ou côncava nas linhas podemos encontrar os índices de mínimos e máximos das linhas em tempo  $\mathcal{O}(n + m)$  e
- se  $A$  é totalmente monótona convexa ou côncava nas colunas podemos encontrar os índices de mínimos e máximos das colunas em tempo  $\mathcal{O}(n + m)$ .

Apresentaremos o caso onde  $A$  é totalmente monótona convexa nas linhas e estamos interessados nos índices de mínimos. Na Subseção ?? explicamos como reduzir os problemas elencados para este caso.

**4.1. Técnica Primordial.** Para facilitar a compreensão do algoritmo SMAWK, iremos apresentar uma técnica parecida com a Divisão e Conquista apresentada na Seção ?? e mostrar uma otimização desta técnica que leva ao algoritmo SMAWK.

Dada uma matriz  $A \in \mathbb{Q}^{n \times m}$  totalmente monótona convexa por linhas, queremos encontrar o menor índice de mínimo de cada uma das linhas de  $A$ . Se para uma dada linha  $i$  onde  $i > 0$  e  $i < n$  conhecermos os índices  $l$  e  $r$  de mínimos das linhas  $i - 1$  e  $i + 1$ , respectivamente, já que  $A$  tem os índices de mínimos das linhas crescente (por ser totalmente monótona) basta buscar o índice de mínimo da linha  $i$  no intervalo entre  $l$  e  $r$  (inclusive). Além disso, se  $i$  é a primeira linha da matriz podemos considerar  $l = 1$  ou se  $i$  é a última linha da matriz podemos considerar  $r = n$  sem perder a validade do fato de que basta buscar entre  $l$  e  $r$ .

Após realizar as observações acima note que, já que  $A$  é totalmente monótona, remover qualquer linha de  $A$  mantém a total monotonicidade e não altera o índice de mínimo de outra linha. Com esta observação, concluímos que podemos remover todas as linhas pares da matriz, resolver o problema recursivamente para a matriz resultante e utilizar este resultado para calcular os índices de interesse para as linhas pares da matriz.

Com uma análise similar à realizada para a técnica da Divisão e Conquista é fácil concluir que uma implementação desta técnica que consiga remover as linhas pares da matriz (e adicionar elas de volta) em tempo  $\mathcal{O}(1)$  resolve o problema em tempo  $\mathcal{O}((n + m) \lg(n))$ , assim como a técnica da divisão e conquista.

## 4.2. Reduce.

## REFERÊNCIAS

- [1] Alok Aggarwal, Maria M. Klawe, Shlomo Moran, Peter Shor, and Robert Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1):195–208, 1987.
- [2] Wolfgang Bein, Mordecai J. Golin, Lawrence L. Larmore, and Yan Zhang. The knuth-yao quadrangle-inequality speedup is a consequence of total monotonicity. *ACM Trans. Algorithms*, 6(1):17:1–17:22, December 2009.
- [3] Zvi Galil and Kunsoo Park. Dynamic programming with convexity, concavity and sparsity. *Theoretical Computer Science*, 92(1):49 – 76, 1992.
- [4] F. Frances Yao. Efficient dynamic programming using quadrangle inequalities. In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, STOC '80, pages 429–435, New York, NY, USA, 1980. ACM.