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 (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.*

Vale notar que a definição dada acima é específica para vetores porém é compatível com a definição usual de funções convexas, isto é, a é convexa se e somente se existe uma função f convexa tal que para todo $i \in [n]$, $a_i = f(i)$. A afirmação análoga para a concavidade também é verdadeira.

Assim como a monotonicidade, a convexidade também é usualmente explorada para agilizar algoritmos, por exemplo, se um vetor é estritamente convexo (basta substituir $\leq$ por $<$ na definição) podemos encontrar o mínimo (que é único) com uma busca ternária ao invés de percorrer todo o vetor.

Talvez essa definição vá para a introdução. Prometo não usar essa notação estranha sem aviso.

Definição 2.3. *Seja $A \in \mathbb{Q}^{n \times m}$, definimos quatro vetores a seguir.*

- *O vetor de índices de máximos das linhas de A guarda na posição i o número $\min\{j \in [m] \mid A[i][j] \geq A[i][j']$ para todo $j' \in [m]\}$.*
- *$\check{j_A} \in \mathbb{Q}^n : i \mapsto \min\{j \in [m] \mid A[i][j] \leq A[i][j']$ para todo $j' \in [m]\}$, o vetor de índices de mínimos das linhas de A,*
- *$\hat{i_A} \in \mathbb{Q}^m : j \mapsto \min\{i \in [n] \mid A[i][j] \geq A[i'][j]$ para todo $i' \in [n]\}$, o vetor de índices de máximos das colunas de A e*
- *$\check{i_A} \in \mathbb{Q}^m : j \mapsto \min\{i \in [n] \mid A[i][j] \leq A[i'][j]$ para todo $i' \in [n]\}$, o vetor de índices de mínimos das colunas de A.*

Note que o índice de máximo de uma linha é definido como o menor índice que contém o valor máximo daquela linha. O mesmo vale para colunas e mínimos.

Calcular estes vetores para dadas matrizes é um problema interessante. Algumas propriedades das matrizes podem ser exploradas a fim de agilizar este cálculo. A monotonicidade, convexidade ou concavidade de matrizes indicam propriedades interessantes sobre estes problemas, portanto, vamos explorar estas definições.

Definição 2.4 (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 2.5 (Matriz totalmente monótona)**.** *Seja $A \in \mathbb{Q}^{n\times m}$ uma matriz.*

- *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 convexa 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 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 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 colunas.*

O motivo do uso dos termos "convexa" e "côncava" neste contexto são justificados pelo Teorema **??**. Note que se uma matriz é totalmente monótona, todas as suas submatrizes são totalmente monótonas no mesmo sentido. O seguinte lema caracteriza uma propriedade importante sobre matrizes totalmente monótonas.

Lema 2.6. *Se $A \in \mathbb{Q}^{n\times m}$ é uma matriz totalmente monótona convexa nas linhas.*

(1) *Toda submatriz de A é monótona crescente nos máximos das linhas.*
(2) *Toda submatriz de A é monótona decrescente nos mínimos das linhas.*

Simetricamente, se A é côncava nas linhas, suas submatrizes são decrescentes nos máximos e crescentes nos mínimos das linhas. O lema funciona análogamente em termos de colunas, também.

Demonstração. Seja $A \in \mathbb{Q}^{n\times m}$ uma matriz totalmente monótona convexa nas linhas. Vamos provar que A é monótona crescente nos máximos das linhas. Sejam $i < i'$ índices de linhas de A. Seja j' o índice de máximo da linha i e j o índice de máximo da linha i'. Suponha, por absurdo, $j < j'$. Assim, teremos, já que j' é o máximo na i-ésima linha, $A[i][j'] > A[i][j]$ e, por outro lado, já que j é o máximo na i'-ésima, $A[i'][j] \geq A[i'][j']$, porém, A é totalmente monótona, portanto, $A[i'][j'] > A[i'][j]$, uma contradição.

Agora, vamos mostrar que A é monótona decrescente nos mínimos das linhas. Sejam $i < i'$ índices de linhas de A. Seja j o índice de mínimo da linha i e j' o índice de mínimo da linha i'. Suponha, por absurdo, $j < j'$. Assim, teremos $A[i][j] \leq A[i][j']$ e $A[i'][j'] < A[i'][j]$ □

3. Divisão e Conquista

Nesta seção será apresentada uma técnica que pode ser usada para encontrar máximos de linhas em matrizes monótonas crescentes nos máximos das linhas em tempo $\mathcal{O}((m+n)\lg(n))$, onde m é a quantidade de colunas e n a quantidade de linhas da matriz. Ao final desta seção, apresentamos exemplos de aplicações desta ideia em programação dinâmica e elencamos generalizações diretas do que foi explicado.

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:     ℓ ← ⌈(r_s + r_t)/2⌉
3:     R[ℓ] ← min{j | A[i][j] ≥ A[i][j'] para todo j' ∈ [c_s .. c_t]}
4:     se i > r_s então
5:         R[r_s .. ℓ − 1] ← FindRowMax_DC(A, r_s, ℓ − 1, c_s, R[ℓ])
6:     se i < r_t então
7:         R[ℓ + 1 .. r_t] ← FindRowMax_DC(A, ℓ + 1, r_t, R[ℓ], 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 FindRowMax_DC(A, r_s, r_t, c_s, c_t) pode ser expresso pela seguinte recorrência:

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

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 $R[r_c \,..\, r_t]$, 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`.

3.4. **Generalizações.** Podemos manipular a matriz dada e usar o Algoritmo 3.1 como caixa preta para resolver vários problemas parecidos. Se a matriz $A \in \mathbb{Q}^{n \times m}$ recebida tiver os índices de máximos decrescentes, podemos espelhar a matriz e aplicar o algoritmo nesta nova matriz, ou seja, na matriz B onde $B[i][j] = A[i][m - j + 1]$ para todo $i \in [n]$ e $j \in [m]$.

Se estivermos interessados no vetor de mínimos das linhas em uma matriz A onde este é monótono, podemos trabalhar sobre a matriz $-A$ e teremos um problema que sabemos resolver usando a técnica original ou a transformação já apresentada. Ainda mais, se estivermos interessados nos vetores relativos às colunas, basta transpor a matriz recebida e aplicar os conhecimentos já discutidos, desde que as hipóteses necessárias tornem-se válidas.

Todas as adaptações exemplificadas aqui podem ser atingidas facilmente com adaptações realizadas diretamente no algoritmo ao invés de transformações na matriz de entrada.

Referências

[1] 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.