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*

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

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

O motivo do uso dos termos "convexa"e "côncava"neste contexto são justificados pelo teorema **??**. O seguinte lema caracteriza uma propriedade importante sobre matrizes totalmente monótonas.

Lema 2.6. *Seja $A \in \mathbb{Q}^{n \times m}$. As seguintes afirmações são equivalentes.*

(1) *Toda submatriz de A é monótona crescente no máximo das linhas.*
(2) *Toda submatriz 2×2 de A é monótona crescente no máximo das linhas. Isto é, vale $A[i'][j'] < A[i'][j] \Rightarrow A[i][j'] < A[i][j]$ para todo $1 \leq i < i' \leq n$ e $1 \leq j < j' \leq m$.*
(3) *Vale $A[i+1][j+1] < A[i+1][j] \Rightarrow A[i][j+1] < a[i][j]$.*

Vale notar que o lema é análogo em termos de colunas, mínimos e índices decrescentes.

Demonstração. Trivialmente, 1 implica em 2 que, por sua vez, implica em 3. Queremos provar que 3 implica em 1. $\square$

3. Divisão e Conquista

eu acho que tem que falar algo como "monótona convexa por linhas". Eu quero dizer que o row-maxima tem índice crescente no índice da linha. Quando estiver escrevedo a seção 2 vou pensar melhor sobre isso.

Aqui será apresentada uma técnica que pode ser usada para encontrar máximos ou mínimos de linhas em matrizes monótonas convexasou côncavas em tempo $O((m+n)\lg(n))$, onde m é a quantidade de colunas e n a quantidade de linhas da matriz. Ao final, apresentamos exemplos de aplicações desta ideia em programação dinâmica.

3.1. **Técnica.** Dada uma matriz $A \in \mathbb{Q}^{n\times m}$ monótona convexa, queremos encontrar, para toda linha i da matriz, o menor índice j tal que $A[i][j]$ é o valor máximo da linha i. Formalmente, queremos encontrar o vetor $R \in \mathbb{N}^n$ onde, para todo $i \in [n]$,

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

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, resolver o mesmo problema para estas submatrizes e conseguimos resolver o problema original.

Algoritmo 3.1 Máximo de linhas com divisão e conquista

1: **função** DivConq(A)
2: $\quad n \leftarrow$ quantidade de linhas de A
3: $\quad m \leftarrow$ quantidade de colunas de A
4: $\quad i \leftarrow \lceil n/2 \rceil$
5: $\quad R[i] \leftarrow \min\{j \mid A[i][j] \geq A[i][j'] \text{ para todo } j' \in [n]\}$
6: $\quad$ **se** $i > 1$ **então**
7: $\quad\quad R[1..i-1] \leftarrow$ DivConq($A[1..i-1][1..R[i]]$)
8: $\quad$ **se** $i < n$ **então**
9: $\quad\quad R[i+1..n] \leftarrow R[i] +$ DivConq($A[i+1..n][R[i]..m]$)
10: $\quad$ **devolve** R

3.2. **Análise.** Será feita uma análise do tempo de execução do algoritmo acima assumindo que as atribuições feitas nas linhas 7 e 9 custam tempo constante, futuramente, em 3.3, iremos apresentar uma implementação em C++ que está de acordo com a análise realizada.

Se $A \in \mathbb{Q}^{n\times m}$, o tempo gasto por FindRowMax_DivConq(A) pode ser expresso pela seguinte recorrência:

$$T(n,m) = \begin{cases} m & \text{, se } n = 1, \\ m + T(1,r) \text{ para algum } r \in [m] & \text{, se } n = 2, \\ \begin{aligned} m \;&+\; T(\lceil n/2 \rceil - 1, m - r + 1) \\ &+\; T(\lfloor n/2 \rfloor, r) \text{ para algum } r \in [m] \end{aligned} & \text{, caso contrário.} \end{cases}$$

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 $O((m+n)\lg(n))$*

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 $r \in [m]$ tal que $T(2,m) = m + r \leq 2m \leq (m+2)\lg(4)$. Agora, se $n > 2$ e $m \geq 1$, existe um $r \in [m]$ tal que

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

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

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

□

3.3. **Implementação.** Aqui serão discutidos os detalhes de implementação do algoritmo 3.1.

Primeiramente a matriz A deve ser facilmente manipulada, não podemos gerar novas matrizes para alimentar as chamadas recursivas da função DivConq. Além disso, como explicado em 1.4, o parâmetro a ser passado será uma função e não uma matriz. Note que a matriz nas chamadas recursivas é sempre formada por um intervalo das linhas e um intervalo das colunas da matriz A original, portanto, podemos guardar 4 inteiros, ℓs, ℓt, cs e ct, rempresentando, respectivamente, a primeira linha, a última linha, a primeira coluna e a última coluna acessíveis na chamada atual da recursão.

Agora, devemos tomar cuidado com as atribuições feitas nas linhas 7 e 9. 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, iremos passar o endereço do vetor R recursivamente e garantir que cada chamada só complete o subvetor $R[\ell s..\ell t]$, referente a seu subproblema. Além disso, a linha 9 possui uma adição, o que sugere que o valor $R[i]$ seja adicionado elemento-a-elemento, o que não será feito, os vetor R já será preenchido com o índice correto da coluna na matriz original A da primeira vez que for acessado.

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.