Aula 2

Busca Binária

Maratona de Programação

FEI

November 4, 2025

Fundamento: Busca em Lista Ordenada

Problema: dado um vetor A de tamanho N ($N \leq 10^6, 0 \leq A_i \leq 10^9$) , responda Q queries ($Q \leq 10^6$) do tipo:

O valor X ($X \leq 10^9$) está presente no vetor?

Fundamento: Busca em Lista Ordenada

Ideia de solução 1: busca completa

Fundamento: Busca em Lista Ordenada

Ideia de solução 1: busca completa

- Ideia: Para cada query, olhar para o vetor A e verificar se existe algum i tal que $X == A_i$ é verdade.

Fundamento: Busca em Lista Ordenada

Ideia de solução 1: busca completa

- Ideia: Para cada query, olhar para o vetor A e verificar se existe algum i tal que $X == A_i$ é verdade.
- Isso com certeza da o resultado certo, já que toda posição é explorada.

Fundamento: Busca em Lista Ordenada

Ideia de solução 1: busca completa

- Ideia: Para cada query, olhar para o vetor A e verificar se existe algum i tal que $X == A_i$ é verdade.
- Isso com certeza da o resultado certo, já que toda posição é explorada.
- Por que essa solução **não** é ótima?

Fundamento: Busca em Lista Ordenada

Ideia de solução 1: busca completa

- Ideia: Para cada query, olhar para o vetor A e verificar se existe algum i tal que $X == A_i$ é verdade.
- Isso com certeza da o resultado certo, já que toda posição é explorada.
- Por que essa solução **não** é ótima?
- Complexidade de tempo: $\mathcal{O}(N * Q)$.

Fundamento: Busca em Lista Ordenada

Ideia de solução 1: busca completa

- Ideia: Para cada query, olhar para o vetor A e verificar se existe algum i tal que $X == A_i$ é verdade.
- Isso com certeza da o resultado certo, já que toda posição é explorada.
- Por que essa solução **não** é ótima?
- Complexidade de tempo: $\mathcal{O}(N * Q)$.
- Quantidade de operacoes no pior caso: $10^6 * 10^6 = 10^{12}$.

Fundamento: Busca em Lista Ordenada

Ideia de solução 1: busca completa

- Ideia: Para cada query, olhar para o vetor A e verificar se existe algum i tal que $X == A_i$ é verdade.
- Isso com certeza da o resultado certo, já que toda posição é explorada.
- Por que essa solução **não** é ótima?
- Complexidade de tempo: $\mathcal{O}(N * Q)$.
- Quantidade de operacoes no pior caso: $10^6 * 10^6 = 10^{12}$.
- Resultado: **Time Limit Exceeded (TLE)**.

Fundamento: Busca em Lista Ordenada

Ideia de solução 2: Pré-processamento

Fundamento: Busca em Lista Ordenada

Ideia de solução 2: Pré-processamento

- Ideia: Antes das queries, criar um vetor V de tamanho $\max_{i=1}^{n} A_i$, preenchido com 0.

Fundamento: Busca em Lista Ordenada

Ideia de solução 2: Pré-processamento

- Ideia: Antes das queries, criar um vetor V de tamanho $\max_{i=1}^{n} A_i$, preenchido com 0.
- Para cada $A_i \in A$, fazer $V_{A_i} := 1$.

Fundamento: Busca em Lista Ordenada

Ideia de solução 2: Pré-processamento

- Ideia: Antes das queries, criar um vetor V de tamanho $\max_{i=1}^{n} A_i$, preenchido com 0.
- Para cada $A_i \in A$, fazer $V_{A_i} := 1$.

A :

1	3	5	6	8

Fundamento: Busca em Lista Ordenada

Ideia de solução 2: Pré-processamento

- Ideia: Antes das queries, criar um vetor V de tamanho $\max_{i=1}^{n} A_i$, preenchido com 0.
- Para cada $A_i \in A$, fazer $V_{A_i} := 1$.

A :

1	3	5	6	8

V :

0	1	0	1	0	1	1	0	1
0	1	2	3	4	5	6	7	8

Fundamento: Busca em Lista Ordenada

Ideia de solução 2: Pré-processamento

- Ideia: Antes das queries, criar um vetor V de tamanho $\max_{i=1}^{n} A_i$, preenchido com 0.
- Para cada $A_i \in A$, fazer $V_{A_i} := 1$.

A :

1	3	5	6	8

V :

0	1	0	1	0	1	1	0	1
0	1	2	3	4	5	6	7	8

- Agora, eu consigo responder uma query com uma busca num vetor.

Fundamento: Busca em Lista Ordenada

Ideia de solução 2: Pré-processamento

- Ideia: Antes das queries, criar um vetor V de tamanho $\max_{i=1}^{n} A_i$, preenchido com 0.
- Para cada $A_i \in A$, fazer $V_{A_i} := 1$.

A :

1	3	5	6	8

V :

0	1	0	1	0	1	1	0	1
0	1	2	3	4	5	6	7	8

- Agora, eu consigo responder uma query com uma busca num vetor.
- Por que essa solução **não** é ótima?

Fundamento: Busca em Lista Ordenada

Ideia de solução 2: Pré-processamento

Fundamento: Busca em Lista Ordenada

Ideia de solução 2: Pré-processamento

- Complexidade de tempo: $\mathcal{O}(N + Q)$

Fundamento: Busca em Lista Ordenada

Ideia de solução 2: Pré-processamento

- Complexidade de tempo: $\mathcal{O}(N + Q)$ (OK).

Fundamento: Busca em Lista Ordenada

Ideia de solução 2: Pré-processamento

- Complexidade de tempo: $\mathcal{O}(N + Q)$ (OK).
- Complexidade de espaco: $\mathcal{O}(\max_{i=1}^{n} A_i)$.

Fundamento: Busca em Lista Ordenada

Ideia de solução 2: Pré-processamento

- Complexidade de tempo: $\mathcal{O}(N + Q)$ (OK).
- Complexidade de espaco: $\mathcal{O}(\max_{i=1}^{n} A_i)$.
- Memoria necessaria no pior caso: $10^9/sizeof(bool) \approx 125MB$.

Fundamento: Busca em Lista Ordenada

Ideia de solução 2: Pré-processamento

- Complexidade de tempo: $\mathcal{O}(N + Q)$ (OK).
- Complexidade de espaco: $\mathcal{O}(\max_{i=1}^{n} A_i)$.
- Memoria necessaria no pior caso: $10^9 / sizeof(bool) \approx 125MB$.
- Resultado:
  Memory Limit Exceeded (MLE) ou Runtime Error (RTE).

Fundamento: Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

Fundamento: Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

- Ideia: Para cada query, buscamos o primeiro i tal que $A_i \geq X$.

Fundamento: Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

- Ideia: Para cada query, buscamos o primeiro i tal que $A_i \geq X$.
- Começamos considerando todo o segmento $[1, N]$ como possível resposta.

Fundamento: Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

- Ideia: Para cada query, buscamos o primeiro i tal que $A_i \geq X$.
- Começamos considerando todo o segmento $[1, N]$ como possível resposta.
- Escolhemos o valor no meio do segmento e verificamos se ele é $\geq X$.

Fundamento: Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

- Ideia: Para cada query, buscamos o primeiro i tal que $A_i \geq X$.
- Começamos considerando todo o segmento $[1, N]$ como possível resposta.
- Escolhemos o valor no meio do segmento e verificamos se ele é $\geq X$.
- Se for, ele pode ser uma resposta, mas ainda tentamos encontrar uma posição melhor à esquerda, olhando agora para $[1, meio]$.

Fundamento: Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

- Ideia: Para cada query, buscamos o primeiro i tal que $A_i \geq X$.
- Começamos considerando todo o segmento $[1, N]$ como possível resposta.
- Escolhemos o valor no meio do segmento e verificamos se ele é $\geq X$.
- Se for, ele pode ser uma resposta, mas ainda tentamos encontrar uma posição melhor à esquerda, olhando agora para $[1, meio]$.
- Se não for, descartamos ele e toda a parte à esquerda, olhando agora para $(meio, N]$.

Fundamento: Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

- Ideia: Para cada query, buscamos o primeiro i tal que $A_i \geq X$.
- Começamos considerando todo o segmento $[1, N]$ como possível resposta.
- Escolhemos o valor no meio do segmento e verificamos se ele é $\geq X$.
- Se for, ele pode ser uma resposta, mas ainda tentamos encontrar uma posição melhor à esquerda, olhando agora para $[1, meio]$.
- Se não for, descartamos ele e toda a parte à esquerda, olhando agora para $(meio, N]$.
- Repetimos até o sobrar só um valor. A posição final encontrada (se houver) é a menor com valor $\geq X$.

Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

Visualização Query ($X = 5$):

A :

2	5	9	12	13	17	20	25	26
0	1	2	3	4	5	6	7	8

Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

Visualização Query ($X = 5$):

A :	<u>2</u>	5	9	12	**13**	17	20	25	<u>26</u>
	0	1	2	3	4	5	6	7	8

Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

Visualização Query (X = 5):

$A:$

2	5	**9**	12	13	17	20	25	26
0	1	2	3	4	5	6	7	8

Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

Visualização Query (X = 5):

A :

2	**5**	9	12	13	17	20	25	26
0	1	2	3	4	5	6	7	8

Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

Visualização Query ($X = 5$):

$A:$	<u>2</u>	<u>5</u>	9	12	13	17	20	25	26
	0	1	2	3	4	5	6	7	8

Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

Visualização Query (X = 5):

A :

2	**5**	9	12	13	17	20	25	26
0	1	2	3	4	5	6	7	8

Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

Visualização Query ($X = 21$):

A :	2	5	9	12	13	17	20	25	26
	1	2	3	4	5	6	7	8	9

Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

Visualização Query (X = 21):

A :

2	5	9	12	**13**	17	20	25	26
1	2	3	4	5	6	7	8	9

Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

Visualização Query (X = 21):

A :

2	5	9	12	13	17	**20**	25	26
1	2	3	4	5	6	7	8	9

Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

Visualização Query (X = 21):

A :

2	5	9	12	13	17	20	**25**	26
1	2	3	4	5	6	7	8	9

Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

Visualização Query ($X = 21$):

A :	2	5	9	12	13	17	20	**<u>25</u>**	26
	1	2	3	4	5	6	7	8	9

Busca em Lista Ordenada

Ideia de solução 3: Busca Binária

Implementando: A. Binary Search.

Ponderando

Ponderando

- Qual a complexidade de tempo desse algoritmo?

Ponderando

- Qual a complexidade de tempo desse algoritmo? $\mathcal{O}(logN)$.

Ponderando

- Qual a complexidade de tempo desse algoritmo? $\mathcal{O}(logN)$.
- Qual a complexidade de espaço desse algoritmo?

Ponderando

- Qual a complexidade de tempo desse algoritmo? $\mathcal{O}(logN)$.
- Qual a complexidade de espaço desse algoritmo? $\mathcal{O}(N)$.

Ponderando

- Qual a complexidade de tempo desse algoritmo? $\mathcal{O}(logN)$.
- Qual a complexidade de espaço desse algoritmo? $\mathcal{O}(N)$.
- Por que ele sempre funciona?

Generalizando: **Monotonicidade**

Generalizando: **Monotonicidade**

- Se criassemos um vetor R, de tamanho N, tal que $R_i := A_i \geq X$, teríamos:

Generalizando: **Monotonicidade**

- Se criassemos um vetor R, de tamanho N, tal que $R_i := A_i \geq X$, teríamos:

$A:$

2	5	9	12	13	17	20	25	26
1	2	3	4	5	6	7	8	9

Generalizando: **Monotonicidade**

- Se criassemos um vetor R, de tamanho N, tal que $R_i := A_i \geq X$, teríamos:

A :

2	5	9	12	13	17	20	25	26
1	2	3	4	5	6	7	8	9

$$X = 12$$

R :

0	0	0	1	1	1	1	1	1
1	2	3	4	5	6	7	8	9

Generalizando: **Monotonicidade**

- Se criassemos um vetor R, de tamanho N, tal que $R_i := A_i \geq X$, teríamos:

A :

2	5	9	12	13	17	20	25	26
1	2	3	4	5	6	7	8	9

$$X = 12$$

R :

0	0	0	1	1	1	1	1	1
1	2	3	4	5	6	7	8	9

- E esse padrão para o vetor R (prefixo contendo apenas $0s$, sufixo contendo apenas $1s$) seria encontrado para todo X em qualquer vetor A ordenado de maneira não decrescente.

Generalizando: **Monotonicidade**

- Isso porque, num array ordenado, nenhum valor anterior à mim vai ser maior que um valor que eu não sou maior.

Generalizando: **Monotonicidade**

- Isso porque, num array ordenado, nenhum valor anterior à mim vai ser maior que um valor que eu não sou maior.

Formalmente, basta garantir que:

$$A_i \geq A_{i-1} \forall i \in [2, N]$$

Para que o vetor R se pareça assim

Generalizando: **Monotonicidade**

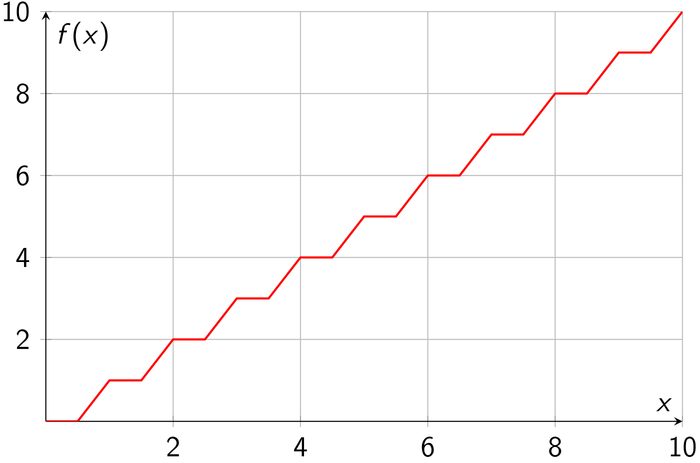
- Isso porque, num array ordenado, nenhum valor anterior à mim vai ser maior que um valor que eu não sou maior.

> Formalmente, basta garantir que:
> $$A_i \geq A_{i-1} \forall i \in [2, N]$$
> Para que o vetor R se pareça assim

- O que é sempre verdade em um vetor ordenado. Mas também é sempre verdade pra qualquer **função monotônica**.

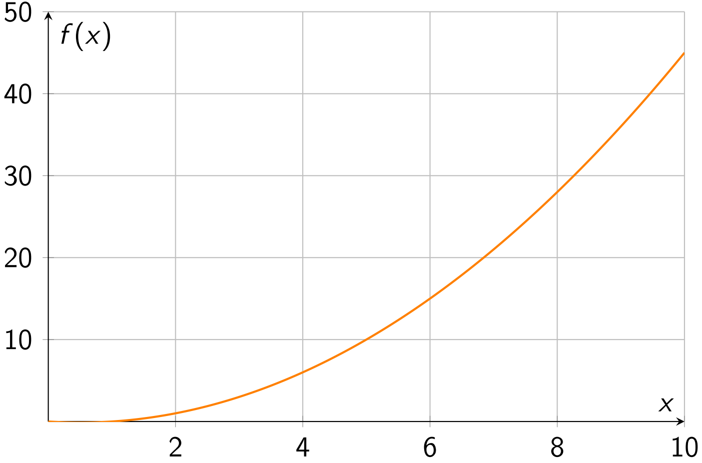
Generalizando: **Monotonicidade**

Exemplo: $\{(x) = \lfloor x \rfloor$



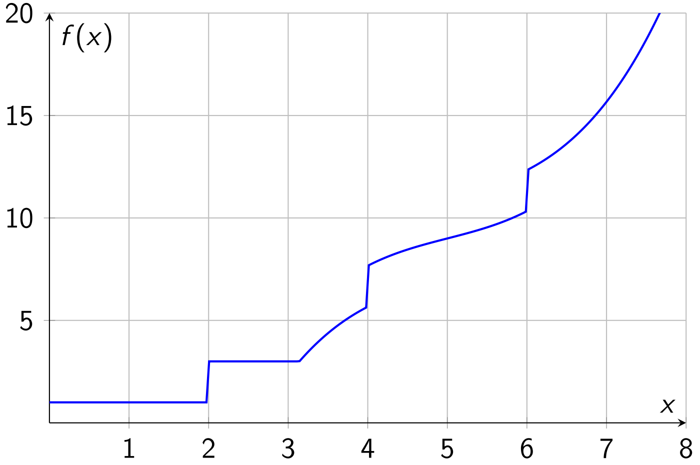
Generalizando: **Monotonicidade**

Exemplo: $\{(x) = \frac{x(x-1)}{2}$



Generalizando: **Monotonicidade**

Exemplo: $\{(x) = \max\left(1, \frac{(x-5)^3}{3} + x\right) + 2(\lfloor \frac{x}{2} \rfloor)$



Busca binária na maratona

- Resumindo:

Busca binária na maratona

- Resumindo:

  Para qualquer problema de maratona, é possível achar:

  - Primeiro ponto tal que $\{(x) \leq Y$ ou,
  - Último ponto tal que $\{(x) > Y$ ou,
  - qualquer outra variação de condição que faça o vetor R parecer com: $R = [0, 0, 0, 1, 1, 1, 1]$.

Busca binária na maratona

- Resumindo:

> Para qualquer problema de maratona, é possível achar:
> - Primeiro ponto tal que $\{(x) \leq Y$ ou,
> - Último ponto tal que $\{(x) > Y$ ou,
> - qualquer outra variação de condição que faça o vetor R parecer com: $R = [0, 0, 0, 1, 1, 1, 1]$.

- Algumas pessoas chamam essa ideia de "busca binária na resposta".

Busca binária na maratona

- Resumindo:

> Para qualquer problema de maratona, é possível achar:
> - Primeiro ponto tal que $\{(x) \leq Y$ ou,
> - Último ponto tal que $\{(x) > Y$ ou,
> - qualquer outra variação de condição que faça o vetor R parecer com: $R = [0, 0, 0, 1, 1, 1, 1]$.

- Algumas pessoas chamam essa ideia de "busca binária na resposta".
- A dificuldade dos problemas usualmente é:

Busca binária na maratona

- Resumindo:

> Para qualquer problema de maratona, é possível achar:
> - Primeiro ponto tal que $\{(x) \leq Y$ ou,
> - Último ponto tal que $\{(x) > Y$ ou,
> - qualquer outra variação de condição que faça o vetor R parecer com: $R = [0, 0, 0, 1, 1, 1, 1]$.

- Algumas pessoas chamam essa ideia de "busca binária na resposta".
- A dificuldade dos problemas usualmente é:
  - Notar que uma $\{(x)$, útil para resolver o problema, é monotônica.

Busca binária na maratona

- Resumindo:

> Para qualquer problema de maratona, é possível achar:
> - Primeiro ponto tal que $\{(x) \leq Y$ ou,
> - Último ponto tal que $\{(x) > Y$ ou,
> - qualquer outra variação de condição que faça o vetor R parecer com: $R = [0, 0, 0, 1, 1, 1, 1]$.

- Algumas pessoas chamam essa ideia de "busca binária na resposta".
- A dificuldade dos problemas usualmente é:
  - Notar que uma $\{(x)$, útil para resolver o problema, é monotônica.
  - Achar um jeito esperto de computar $\{(x)$.

Busca binária na maratona

Exemplo resolvido 1:

BeeCrowd: Torre de Cartas.

Busca binária na maratona

Exemplo resolvido 2:

CSES: Array Division.

Busca binária na maratona

Exemplo comentado 1:

CodeForces: B. The Meeting Place Cannot Be Changed.

Busca binária na maratona

Exemplo comentado 2:

CodeForces: K. Delivery Bears.

Listas de problemas relacionados

Lista conhecidas:

- UFMG: **link**.
    - veja também o vídeo deles **aqui**.
- USACO Guide: **link**.
- UTL: **link**.