ESCOLA DE PRIMAVERA DA MARATONA DE PROGRAMAÇÃO

GCC

PROMOÇÃO:

CEFET-MG

UFMG

UFV

UNIFEI

USP Universidade de São Paulo

UFU

APOIO:

beecrowd

SBC Sociedade Brasileira de Computação

Grupo de Computação Competitiva

BUSCA BINÁRIA

Por: *Vinicius Alves Pereira (CEFET-DIV)*

CONTEÚDOS

01 - EXPLICAÇÃO

- Busca linear
  - Complexidade O(n)

1 3 5 8 12 18 19 21 24 27

01 - EXPLICAÇÃO

- Busca linear
  - Complexidade O(n)

1 3 5 8 12 18 19 21 24 27

01 - EXPLICAÇÃO

- Busca linear
  - Complexidade O(n)

1 3 5 8 12 18 19 21 24 27

- Como utilizar a ordenação para melhorar o algoritmo?

01 - EXPLICAÇÃO

- Busca linear
  - Complexidade O(n)

1 3 5 8 12 18 19 21 24 27

- Como utilizar a ordenação para melhorar o algoritmo?
  - Dicionário

02 - DEFINIÇÃO DO ALGORITMO

A busca binária funciona com uma ideia semelhante. A busca binária é um algoritmo de busca que encontra um item em uma lista ordenada de itens. Ela funciona dividindo a lista em partes cada vez menores até encontrar o item ou não restar mais elementos para pesquisar.

03 - FUNCIONAMENTO DO ALGORITMO

Vamos resolver o seguinte problema: Dado um vetor, com elementos ordenados, encontrar o maior elemento que seja menor ou igual a X.

03 - FUNCIONAMENTO DO ALGORITMO

Vamos resolver o seguinte problema: Dado um vetor, com elementos ordenados, encontrar o maior elemento que seja menor ou igual a X.

X = 12

A 1 3 5 8 12 18 19 21 24 27

03 - FUNCIONAMENTO DO ALGORITMO

Vamos resolver o seguinte problema: Dado um vetor, com elementos ordenados, encontrar o maior elemento que seja menor ou igual a X.

X = 12

```
A       1  3  5  8  12 18 19 21 24 27
i   -1  0  1  2  3  4  5  6  7  8  9  10
     L                                R
```

onde:

A[i] <= X, para todo i <= L

A[i] > X, para todo i >= R

03 - FUNCIONAMENTO DO ALGORITMO

Vamos resolver o seguinte problema: Dado um vetor, com elementos ordenados, encontrar o maior elemento que seja menor ou igual a X.

X = 12

M = 18

A		1	3	5	8	12	18	19	21	24	27
i	-1	0	1	2	3	4	5	6	7	8	9	10
	L										R

03 - FUNCIONAMENTO DO ALGORITMO

Vamos resolver o seguinte problema: Dado um vetor, com elementos ordenados, encontrar o maior elemento que seja menor ou igual a X.

X = 12

M = 18

A		1	3	5	8	12	18	19	21	24	27
i	-1	0	1	2	3	4	5	6	7	8	9	10
	L						R				

03 - FUNCIONAMENTO DO ALGORITMO

Vamos resolver o seguinte problema: Dado um vetor, com elementos ordenados, encontrar o maior elemento que seja menor ou igual a X.

X = 12

M = 8

A		1	3	5	8	12	18	19	21	24	27
i	-1	0	1	2	3	4	5	6	7	8	9	10
	L						R					

03 - FUNCIONAMENTO DO ALGORITMO

Vamos resolver o seguinte problema: Dado um vetor, com elementos ordenados, encontrar o maior elemento que seja menor ou igual a X.

X = 12

M = 8

A		1	3	5	8	12	18	19	21	24	27	
i	-1	0	1	2	3	4	5	6	7	8	9	10
					L		R					

03 - FUNCIONAMENTO DO ALGORITMO

Vamos resolver o seguinte problema: Dado um vetor, com elementos ordenados, encontrar o maior elemento que seja menor ou igual a X.

X = 12

M = 12

A		1	3	5	8	12	18	19	21	24	27
i	-1	0	1	2	3	4	5	6	7	8	9	10
					L		R				

03 - FUNCIONAMENTO DO ALGORITMO

Vamos resolver o seguinte problema: Dado um vetor, com elementos ordenados, encontrar o maior elemento que seja menor ou igual a X.

X = 12

M = 12

A		1	3	5	8	12	18	19	21	24	27
i	-1	0	1	2	3	4	5	6	7	8	9	10
						L	R				

03 - FUNCIONAMENTO DO ALGORITMO

Vamos resolver o seguinte problema: Dado um vetor, com elementos ordenados, encontrar o maior elemento que seja menor ou igual a X.

X = 12

M = 12

A		1	3	5	8	12	18	19	21	24	27
i	-1	0	1	2	3	4	5	6	7	8	9	10
						L	R				

Nossa resposta será o índice L

03 - FUNCIONAMENTO DO ALGORITMO

```cpp
int main() {
  _;

  int x = 12;
  vector<int> a = {1, 3, 5, 8, 12, 18, 19, 21, 24, 27};

  int l = -1;
  int r = a.size();

  while (r - l > 1) {
    int m = (r + l) / 2;

    if (a[m] > x) {
      r = m;
    } else {
      l = m;
    }
  }

  dbg(l);
  dbg(r);

  if (l != -1) dbg(a[l]);
  if (r != a.size()) dbg(a[r]);
  return 0;
}
```

04 - COMPLEXIDADE

- Quantas comparações faremos no pior caso?
- A cada passo diminuímos o tamanho do nosso vetor pela metade (chão ou teto).
- Logo, podemos pensar no número de comparações como o número de vezes em que precisamos dividir N (tamanho do range) por 2 para que N fique igual a 1 (caso base do nosso algoritmo).

Sendo K esse número de comparações:

$$1 = \frac{N}{2^K}$$
$$2^K = N$$
$$K = \log_2(N)$$

04 - COMPLEXIDADE

Nosso código faz O(log(N)) comparações, sendo N o tamanho do range.

Logo, a complexidade do nosso código é O(M. log(n)), sendo M o custo de cada comparação.

E.g. No caso de uma comparação entre inteiros M = O(1).

05 - C++ STD

A função std::binary_search verifica se um elemento está contido em um vetor

A função std::lower_bound retorna o primeiro elemento não menor (maior ou igual)

A função std::upper_bound retorna o primeiro elemento maior.

05 - C++ STD

```cpp
int main() {
  _;

  vector<int> a = {1, 3, 5, 8, 12, 18, 19, 21, 27, 24};

  binary_search(a.begin(), a.end(), 12);  // Output: 1
  binary_search(a.begin(), a.end(), 30);  // Output: 0
}
```

05 - C++ STD

```cpp
int main() {
  _;

  auto it = lower_bound(a.begin(), a.end(), 11);
  int idx = it - a.begin();

  dbg(*it); // *it = 12
  dbg(idx); // idx = 4
}
```

05 - C++ STD

```cpp
int main() {
  _;

  auto it = upper_bound(a.begin(), a.end(), 11);
  int idx = it - a.begin();

  dbg(*it); // *it = 12
  dbg(idx); // idx = 4
}
```

06 - BUSCA BINÁRIA NA RESPOSTA

Na busca binária na resposta nós buscamos um elemento do vetor que seja resposta para nosso problema. Essa técnica pode ser aplicada caso nossa resposta seja transitiva. Ou seja:

Se Ai é válido, Aj (para todo j >= i) também será

Se Ai é inválido, Aj (para todo j > 0 e j <= i) também será

06 - BUSCA BINÁRIA NA RESPOSTA

Na busca binária na resposta nós buscamos um elemento do vetor que seja resposta para nosso problema. Essa técnica pode ser aplicada caso nossa resposta seja transitiva. Ou seja:

Se Ai é válido, Aj (para todo j >= i) também será
Se Ai é inválido, Aj (para todo j > 0 e j <= i) também será

Exemplo: Se uma compra ficou em 49 reais, qual a menor nota que podemos utilizar para pagar?

2 5 10 20 | 50 100 200

07 - REFERÊNCIAS

- CP Algo - Binary search
- Lista UFMG (aula e exercícios)
- Codeforces EDU

OBRIGADO PELA ATENÇÃO

Grupo de Computação Competitiva

GCC.CEFETMG

gcc-cefetmg.github.io/Site-GCC

@gcc.cefetmg