BUSCA BINÁRIA

Algoritmos e Estrutura de Dados II

Prof. André Vignatti

BUSCA BINÁRIA

Busca em Vetor Ordenado

Instância: (x, v, a, b), onde x é um valor e $v[a..b]$ é um vetor ordenado.
Resposta: $m \in [a..b]$ tal que $v[m] = x$ ou **não** se tal m não existir.

BuscaBinaria(x,v,a,b)

Se $a > b$
    Devolva $N\tilde{A}O$
$m \leftarrow \lfloor \frac{a+b}{2} \rfloor$
Se $x = v[m]$
    Devolva m
Se $x < v[m]$
    Devolva *BuscaBinária*$(x, v, a, m-1)$
Devolva *BuscaBinária*$(x, v, m+1, b)$

Exemplo. *Executar* *BuscaBinária*$(8, v, 1, 8)$ *e* *BuscaBinária*$(20, v, 1, 8)$ *para*

i	1	2	3	4	5	6	7	8
$v[i]$	4	7	8	15	16	17	23	42

BuscaBinaria(x,v,a,b)

Se $a > b$
    Devolva $N\tilde{A}O$
$m \leftarrow \lfloor \frac{a+b}{2} \rfloor$
Se $x = v[m]$
    Devolva m
Se $x < v[m]$
    Devolva *BuscaBinária*$(x, v, a, m-1)$
Devolva *BuscaBinária*$(x, v, m+1, b)$

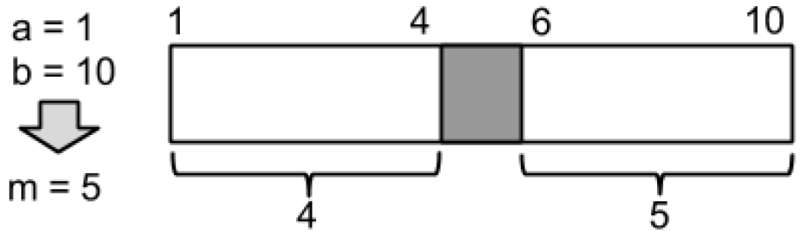
ANÁLISE: PIOR CASO

$C^+(n)$: núm. de comparações com elementos do vetor no **pior caso** para entradas de tamanho n.

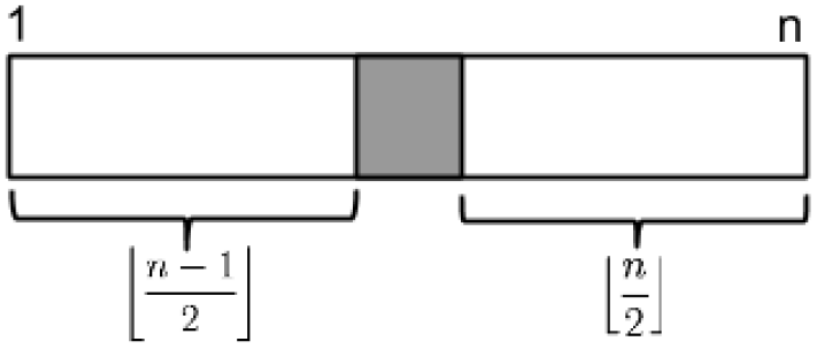
- OBS: O tamanho das chamadas recursivas dependem se n é par ou ímpar.

ANÁLISE: PIOR CASO

Exemplo para $n = 10$ (par):

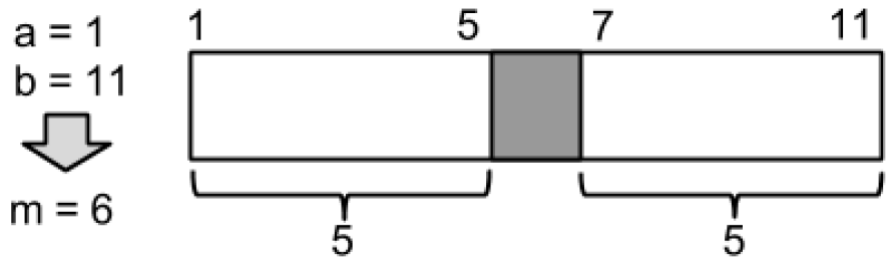


Generalizando, para todo n **par**, o tamanho das chamadas recursivas ficam:

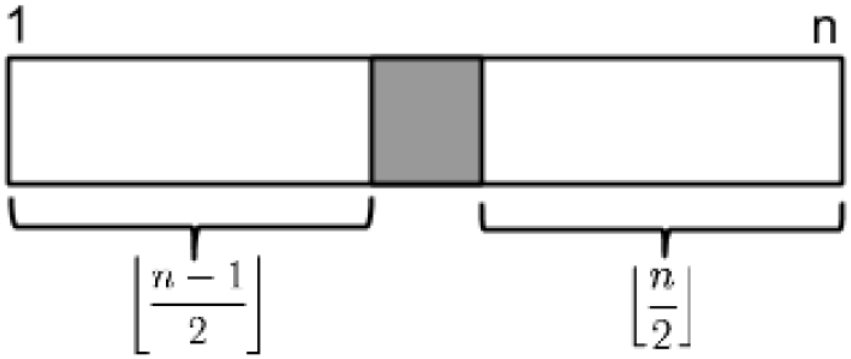


ANÁLISE: PIOR CASO

Exemplo para $n = 11$ (ímpar):

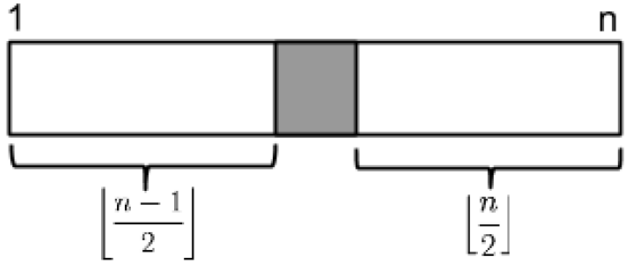


Generalizando, para todo n **ímpar**, o tamanho das chamadas recursivas ficam:



ANÁLISE: PIOR CASO

Portanto, n sendo par ou ímpar, os tamanhos das chamadas recursivas são $\left\lfloor \frac{n-1}{2} \right\rfloor$ e $\left\lfloor \frac{n}{2} \right\rfloor$.



Assim, podemos escrever:

$$C^+(n) = \begin{cases} 0, & \text{se } n \leq 0, \\ 1 + \max\left\{C^+\left(\left\lfloor \frac{n-1}{2} \right\rfloor\right), C^+\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\right\}, & \text{se } n > 0, \end{cases}$$

ANÁLISE: PIOR CASO

Mas, $\max\left\{C^+\left(\left\lfloor\frac{n-1}{2}\right\rfloor\right), C^+\left(\left\lfloor\frac{n}{2}\right\rfloor\right)\right\} = C^+\left(\left\lfloor\frac{n}{2}\right\rfloor\right)$, de forma que

$$C^+(n) = \begin{cases} 0, & \text{se } n = 0, \\ 1 + C^+\left(\left\lfloor\frac{n}{2}\right\rfloor\right), & \text{se } n > 0, \end{cases}$$

Agora podemos resolver a recorrência.

Fato. $\left\lfloor \frac{\left\lfloor \frac{n}{2^k} \right\rfloor}{2} \right\rfloor = \left\lfloor \frac{n}{2^{k+1}} \right\rfloor.$

deve-se provar, mas não iremos

ANÁLISE: PIOR CASO

Abrindo a recorrência,

$$\begin{aligned} C^+(n) &= 1 + C^+\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \\ &= 1 + 1 + C^+\left(\left\lfloor \frac{n}{2^2} \right\rfloor\right) \\ &= 1 + 1 + 1 + C^+\left(\left\lfloor \frac{n}{2^3} \right\rfloor\right) \\ &\dots \\ &= u + C^+\left(\left\lfloor \frac{n}{2^u} \right\rfloor\right) \end{aligned}$$

Qual valor de u para chegar em $C^+(0)$? Ou seja, quando $\left\lfloor \frac{n}{2^u} \right\rfloor = 0$?

ANÁLISE: PIOR CASO

Isso ocorre quando o denominador é maior que o numerador

ocorre para o **menor** u inteiro tal que $2^u > n$

Isolando u, temos que

$$u > \log_2 n$$

Mas $\log_2 n$ pode ser inteiro ou fracionário:

ANÁLISE: PIOR CASO

Mas $\log_2 n$ pode ser inteiro ou fracionário:

- Se $\log_2 n$ é inteiro

$$u > \log_2 n \text{ implica que } u = \log_2 n + 1 = \lfloor \log_2 n \rfloor + 1$$

- Se $\log_2 n$ é fracionário

$$u > \log_2 n \text{ implica que } u = \lceil \log_2 n \rceil = \lfloor \log_2 n \rfloor + 1$$

ANÁLISE: PIOR CASO

Em ambos os casos, o menor u inteiro para chegar na base é

$$u = \lfloor \log_2 n \rfloor + 1$$

Substituindo o valor de u na recorrência:

$$C^+(n) = \lfloor \log_2 n \rfloor + 1 + C^+(0) = \lfloor \log_2 n \rfloor + 1.$$

ANÁLISE: PIOR CASO

conclusão:

Teorema. *Para um vetor de tamanho n, o número de comparações com elementos do vetor do Algoritmo* BuscaBinária *é, no pior caso,* $\lfloor \log_2 n \rfloor + 1$.

e o melhor caso?