Teste 1 - 2019

105

5

1. int maxarray (int v[], int a, int b) {

// pre : $0 \leq a \leq b$

$\boxed{V: b - i + 1}$

int i = a+1, r = a;

// inv: $\boxed{a \leq r \leq i \wedge \text{forall}_{\{a \leq k < i\}}\ v[k] \leq v[r] \wedge i \leq b+1}$

$\to i+1 < b$

while (i <= b) {

if (v[i] > v[r]) { r = i; }

i += 1;

}

// pos: $a \leq r \leq b \wedge \forall k.\ a \leq k \leq b \to v[k] \leq v[r]$

}

$i > b$

$i >= b+1$

$6 > 5$

$6 >= 6$

N = 4

0	1	2	3
10	5	17	9

i = 2, r = 0 ; i = 3, r = 2 ; i = 4, r = 2

Seja a = 0, b = 3

Possível I, olhando para a pos e para o array

I: $a \leq r \leq i \wedge \text{forall}_{\{a \leq k < i\}}\ v[k] \leq v[r]$

→ Provar $\boxed{\text{Init}}$ {P} ... {I}

$\{0 \leq a \leq b\}\ i = a+1;\ r = a;\ \{a \leq r \leq i \wedge \text{forall}_{\{a \leq k < i\}}\ v[k] \leq v[r]\}$

$0 \leq a \leq b \Rightarrow a \leq a \leq a+1 \wedge \text{forall}_{\{a \leq k < a+1\}}\ v[k] \leq v[a]$

$a \leq k < a$

→ Init

→ Provar $\boxed{\text{Util}}$ $I \wedge \neg c \Rightarrow Q$

$\{a \leq r \leq i \wedge \text{forall}_{\{a \leq k < i\}}\ v[k] \leq v[r] \wedge i > b\} \Longrightarrow a \leq r \leq b \wedge \text{forall}_{\{a \leq k \leq b\}}\ v[k] \leq v[r]$

$i \geq b+1$ logo falta acrescentar $i \leq b+1$ para podermos ter $i = b$

Afinal I: $a \leq r \leq i \wedge \text{forall}_{\{a \leq k < i\}}\ v[k] \leq v[r] \wedge i \leq b+1$

→ Provar $\boxed{\text{Pres}}$ $\{I \wedge C\}$ S $\{I\}$

$a \leq \underline{r} \leq i \wedge$ forall_$\{a \leq k \leq i\}$ $v[k] \leq v[\underline{r}] \wedge i \leq b$

$\wedge\, v[i] > v[r]$

$v[k] \leq v[r] < v[i]$

$\Longrightarrow$

$a \leq \underline{i} \leq i+1 \wedge$ forall_$\{a \leq k \leq i+1\}$ $v[k] \leq v[\underline{i}]$

$\wedge\, i \leq b$

k pode ser igual a i e $v[k] \leq v[r]$

$i \leq b$

$\Updownarrow$

$\wedge\, i+1 \leq b+1$

$$\frac{I \wedge C \wedge v[i] > v[r] \Rightarrow R[r \backslash i] \qquad I \wedge C \wedge v[i] \leq v[r] \rightarrow R}{\{I \wedge C\} \text{ if } (v[i] > v[r])\ r = i;\ \{R\}} \text{ if} \qquad \frac{R \Rightarrow a \leq r \leq i+1 \wedge \text{forall\_}\{a \leq k \leq i+1\}\ v[k] \leq v[r]}{\{R\}\quad i = i+1; \quad \{I\}} \text{ Atr}$$

$\{a \leq r \leq i \wedge$ forall_$\{a \leq k < i\}$ $v[k] \leq v[r] \wedge i \leq b+1$
$\wedge\ i \leq b\}$ if $(v[i] > v[r])\ r = i;\ \ i = i+1;\quad \{I\}$

$i \leq b$

(1)

2.

```
Void MaxWindowRec (int v[], int r[], int N, int K) {
    if (N>=K) {
        r[0] = v[maxarray (v, 0, K-1)];
        MaxWindowRec (v+1, r+1, N-1, K);
    }
}
```

✓ pré: $0 \leq K \leq N$

× pós: forall_{0 ≤ i ≤ N-K} r[i] == v[maxarray (v, i, i+K-1)]

(2.5)

3. (i) Recorrência para nº comparações efetuadas entre elems do array em função do N e do K (T(N,K)).

$$T(N,K) = \begin{cases} 0 & \Leftarrow N < K \\ T_{maxarray}(a,b) + T(N-1, K) & \Leftarrow N >= K \end{cases}$$

é errado já são termos a e b mas sabemos que $\left.\begin{matrix} a=0 \\ b=K-1 \end{matrix}\right\}$ no código acima. então

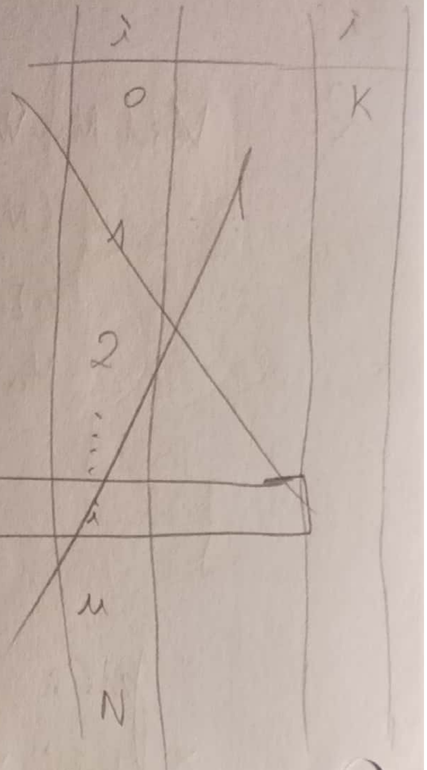
Sabemos que

$$\sum_{i=1}^{K-1} 1 = K-1-1+1 = K-1$$

Logo

$$T_{MaxWindowRec}(N,K) = \begin{cases} 0 & \Leftarrow N < K \\ K-1 + T_{MWR}(N-1, K) & \Leftarrow N >= K \end{cases}$$

$T_{MWR}(N)$ — $K-1$
|
$T_{MWR}(N-1)$ — $K-1$
|
$T_{MWR}(N-2)$ — $K-1$
⋮
$T(1)$ — $K-1$
|
$T(0)$ — 0



Mas o modo de pensar é idêntico, temos a cada i-nível $K-1$ logo

$$T_{MWR}(N,K) = \sum_{i=K}^{N} K-1 + 0$$

$$= (K-1) \sum_{i=K}^{N} 1$$

$$= \underline{(K-1)(N-K+1)} = \Theta(N)$$

é errado pois aqui temos

$$T(N) = \begin{cases} \dots \Leftarrow N < K \\ \dots \Leftarrow N >= K \end{cases}$$

e não

$$T(N) = \begin{cases} \dots \Leftarrow N == 0 \\ \dots \Leftarrow N > 0 \end{cases}$$

(ii) Caso Médio de $T(N)$ de MaxWindowRec X

valores de K tem igual probabilidade de ocorrer.

Sabemos que $0 \leq K \leq N$ e se

$$\overline{T_{MWR}}(N) = \sum_{K=1}^{N} \underbrace{\frac{1}{N}}_{p} * \underbrace{T_{MWR}(N,K)}_{c}$$

entre tem probabilidade $\frac{1}{N}$

Se tivessemos PC e MC temos uma tabela como est

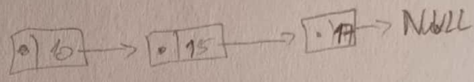
Aqui temos

K	p	Custo
1	$\frac{1}{N}$	$T_{MWR}(N,1)$
2	"	$T_{MWR}(N,2)$
3	"	⋮
4	"	⋮
⋮	"	⋮
N	"	

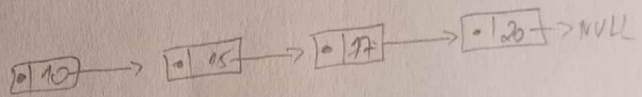
PC	0
⋮	⋮
MC	

4.

```
node *insert_rem (node *p, int x){
    node *new = malloc (sizeof (node)); new->value = x;
    while (p && x > p->value) {
        aux = p;
        p = p->next;
        free (aux);
    }
    new->next = p;
    return new;
}
```



x = 20



(i) Analisar tempo de execução $\boxed{\text{assimptótico}}$ de insert_rem, identificando o MC e PC

Se tivermos em conta nº de comparações de x > p->value

MC -> quando x ≤ p->value e por isso faz 1 comparação $\boxed{\Theta(1)}$

PC -> quando x > p->value a todos os N-1 elems da lista e então é depois inserido na última ou penúltima posição

$$T^{P}(N) = \sum_{i=0}^{N-1} 1 = (N-1) = \boxed{\Theta(N)}$$

$$\therefore \boxed{T(N) = \Omega(1),\ O(N)}$$