RODRIGO ALVES DE ALMEIDA

COMP - T22

01

	$\omega(\log^3 n)$	$\Omega(n^2 \log n)$	$O(n^{7/3})$	$\Theta(n \log^3 n)$	$o(n^2)$
$n.\log^3 n + 3n^{5/3}$	$\in$	$\notin$	$\in$	$\notin$	$\in$
$4n^{7/3} + 7n\log^3 n - \log n$	$\in$	$\in$	$\in$	$\notin$	$\notin$

02

```
bool    arv_completa = TRUE;
int Altura_Nó (p){
    if( p = NULL)
        return 0;
    int altura_dir = Altura_Nó (p→dir);
    int altura_esq = Altura_Nó (p→esq);
    if (altura_dir != altura_esq)
        arv_completa = FALSE;   → var global
    return altura_dir +1;   → retorna altura do nó
                              (atura dos filhos +1)
}
```

Esse algoritmo verifica se algum nó da arvore tem seu filho esquerdo com altura diferente do direito e, caso tenha, indica que a árvore não é completa na variável global arv_completa.

03 Essa conclusão é errada pois, no momento de intercalar os k vetores, será necessário fazer muito mais comparações para formar o vetor intercalado.

Por exemplo, vamos pensar no caso limite onde $n=k$. Aqui, todos os elementos do vetor serão tratados como subvetor e, na hora da intercalação, teremos que comparar entre todos para decidir qual é o menor (ou seja, temos um Selection Sort!), deixando a complexidade $O(n^2)$

(04)

```
void Inverter (v, n){
    Pilha p;
    for (int i=0; i<[n/2]; i++)
        p.push (i);
    while (! p.isEmpty()) {
        int pos_troca = p.top();
        p.pop();
        int pos_simetrica = n - pos_troca - 1;
        aux = v[pos_simetrica],
        v[pos_simetrica] = p[pos_troca],
        v[pos_troca] = aux;
    }
}
```

05

$T(1) = 1$

$T(5/3) = 1 + \frac{5}{3}$

$T((\frac{5}{3})^2) = 1 + \frac{5}{3} + (\frac{5}{3})^2$

sendo $n = (\frac{5}{3})^k$

$$T(n) = \sum_{i=0}^{k} (\frac{5}{3})^i = (\frac{5}{3})^k \sum_{i=0}^{k} (\frac{3}{5})^i$$

Sabemos que:

$$1 < \sum_{i=0}^{k} (\frac{3}{5})^i < \frac{1}{1-3/5}$$

$$\therefore \quad n \leq T(n) \leq \frac{5}{2} n$$

$$T(n) \in \Theta(n)$$

06

1) Para $n = 1$

$$2^{2n} - 1 = 3$$ ✓

2) Para $n = k$

$$2^{2k} - 1 = 3 \cdot m \quad , \; m \text{ inteiro} \qquad (1)$$

3) Para $n = k+1$

$$2^{2k+1} - 1 = 4\,4^k - 1$$

temos de (1):

$$4^k = 3m + 1$$

$$2^{2(k+1)} = 12m + 4 - 1 = 12m + 3 = 3 \cdot (4m + 1)$$ ∎

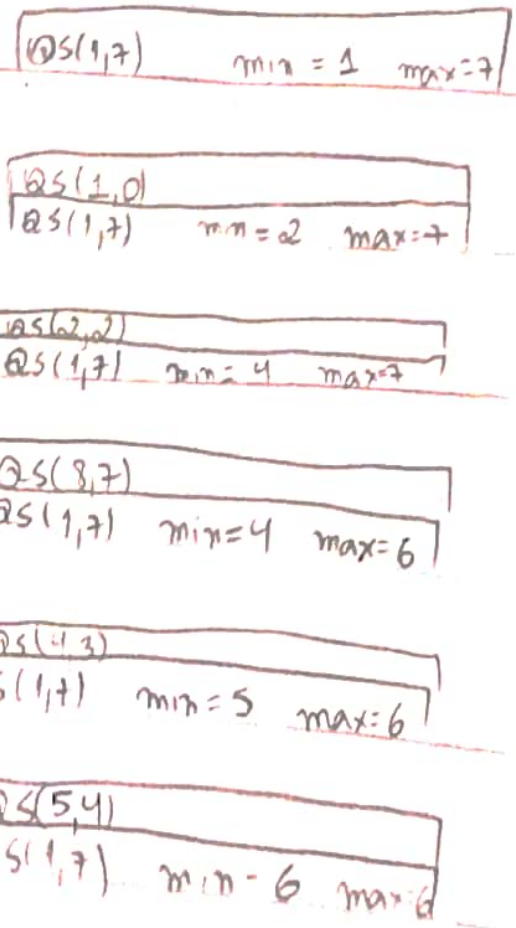
↳ inteiro

07

vetor

	2	3	6	10	5	3	4
Partition(1,7)	|2|	3	6	10	5	3	4
Partition(2,7)	2	3	|3|	10	5	6	4
Partition(4,7)	2	3	3	4	5	6	|10|
Partition(4,6)	2	3	3	|4|	5	6	10
Partition(5,6)	2	3	3	4	|5|	6	10
Partition(6,6)	2	3	3	4	5	6	10

Pilha

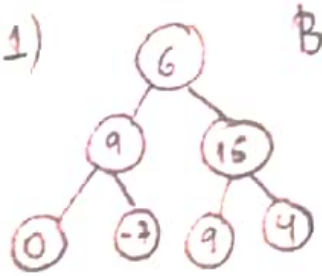


O tamanho maximo da pilha é 2

08) Em cada percorrimento da lista, são realizadas n comparações e metade das pessoas são eliminadas. Portanto, o número total de comparações é da ordem de $n \log_2 n$

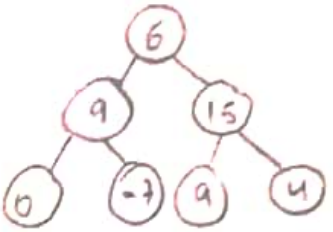
$$T(n) \in n \log n$$

09 1) Build:

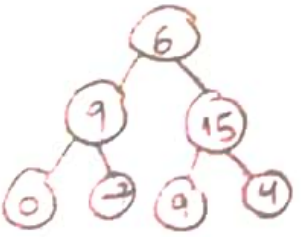


6 9 15 0 -7 9 4
1 2 3 4 5 6 7

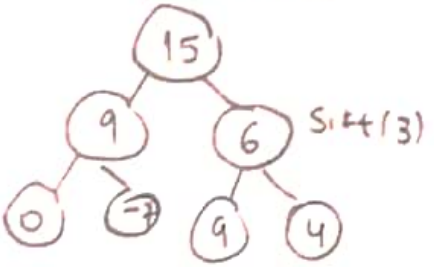
Sift (3)



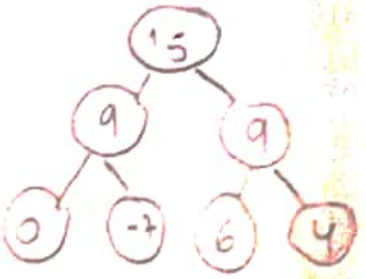
Sift (2)



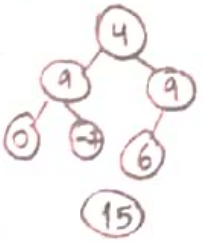
Sift (1)



Sift (3) ⇒

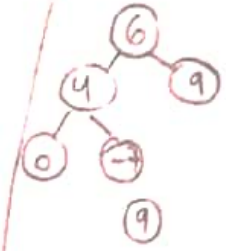


Extrações da raiz:



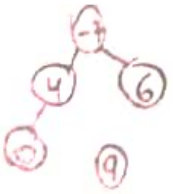
Sift (1)

9 4 9 0 -7 6 15



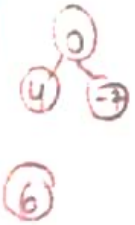
Sift (1)

9 4 6 0 -7 9 15



Sift (1)

6 4 -7 0 9 9 15



Sift (1)

4 0 -7 6 9 9 15

Sift (1)

0 -7 4 6 9 9 15

Sift (1)

-7 0 4 6 9 9 15

(10) a) 1) Procurar o maior disco

2) Realizar um giro do topo até a posição desse elemento, fazendo com que ele vá para o topo

3) Realizar um giro do topo até a base, fazendo com que o elemento vá para a base

4) Repetir o procedimento, considerando agora a subpilha a partir do primeiro elemento (depois a partir do segundo, etc)

b) Em um procedimento numa subpilha de tamanho K, são realizados:

- K comparações, para achar o maior disco
- 2 giros

$T(K) = Ka + 2b$

$$T(n) = \sum_{K=1}^{n} T(K) = a(1+2+3+\dots+n) + 2b(n-1) = a\cdot\frac{(n+1)\cdot n}{2} + 2b(n-1)$$

$\therefore T(n) \in \Theta(n^2)$