



AVALIAÇÃO I

Nome: Moyses Abreu Aguiar

Questões:

1. (1,0) Suponha que o vetor v tenha 511 elementos e que x não está no vetor. Quantas comparações um algoritmo de **busca binária** fará entre x e os elementos do vetor?
2. (1,0) Se preciso de t segundos para fazer uma **busca binária** em um vetor com n elementos, de quanto tempo preciso para fazer uma busca em n^2 elementos?
3. (1,0) Um algoritmo leva 1ms para terminar com uma entrada de tamanho 100. Em quanto **tempo** o algoritmo termina quando a entrada for de tamanho 1000000 e a complexidade for $O(n^3)$?
4. (1,0) Expresse a função $f(n) = 100n^2 + 30n + 6$ em termos da notação *Big-O*, ou seja, prove que $f(n)$ é $O(g(n))$.
5. (1,0) O algoritmo A usa 100n instruções enquanto o algoritmo B usa $3n^2$ instruções. Determine n_0 para o qual A é melhor do que B para $n \geq n_0$.
6. (1,0) Em geral, muitos organismos se reproduzem a uma taxa constante (por exemplo, duplicam a cada 25 minutos ou a cada hora). Podemos supor que temos uma população de bactérias que dobra a cada hora, a partir de uma população inicial x e, a cada hora, colocamos também outras y bactérias no ambiente. Estamos interessados em saber quantas teremos na hora n . Especifique a relação de recorrência para esse problema e use essa relação para determinar o tamanho da população para $x=25$, $n=4$ e $y=15$.
7. (1,0) Mostre que um *heap* de n elementos tem altura $\lfloor \log n \rfloor$.
8. (1,0) Ilustre a operação **MAX-HEAPFY**(A , 3) sobre o vetor de entrada $A=[27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0]$. Determine a complexidade do procedimento (notação *Big-O*).

9. (1,0) No algoritmo **Quicksort**, que valor de q o método *Particionar*(r) retorna quando todos os elementos do vetor $A[p..r]$ têm o mesmo valor? (p e r são respectivamente os valores do primeiro e último índices do vetor A).

10. (1,0) Qual das seguintes afirmações sobre crescimento assintótico de funções não é verdadeira:

- (a) $2n^2 + 3n + 1$ é $O(n^2)$.
- (b) Se $f(n)$ é $O(g(n))$ então $g(n)$ é $O(f(n))$.
- (c) $\log n^2$ é $O(\log n)$.
- (d) Se $f(n)$ é $O(g(n))$ e $g(n)$ é $O(h(n))$ então $f(n)$ é $O(h(n))$.
- (e) 2^{n+1} é $O(2^n)$.

ANEXO

PENSE!

MAX-HEAPFY(A, i) $i \leftarrow \text{LEFT}(i)$ $r \leftarrow \text{RIGHT}(i)$ if $i \leq \text{tamanho_do_heap}$ e $A[i] > A[r]$ then maior $\leftarrow i$ else maior $\leftarrow r$ if $r \leq \text{tamanho_do_heap}$ e $A[r] > A[\text{maior}]$ then maior $\leftarrow r$ if maior $< i$ then trocar $A[i] \leftrightarrow A[\text{maior}]$ MAX-HEAPFY(A, maior)	QUICKSORT(A, p, r) if $p < r$ $q \leftarrow \text{PARTICIONAR}(A, p, r)$ QUICKSORT(A, p, q-1) QUICKSORT(A, q+1, r)
Busca_Binaria(x, e, d, v[]) { if (e == d-1) return d; else { $m = (e + d) / 2$; if (v[m] < x) return Busca_Binaria(x, m, d, v); else return Busca_Binaria(x, e, m, v); } }	PARTICIONAR(A, p, r) $x \leftarrow A[r]$ $i \leftarrow p-1$ for j $\leftarrow p$ to r-1 if $A[j] \leq x$ $i \leftarrow i + 1$ trocar $A[i] \leftrightarrow A[j]$ trocar $A[i+1] \leftrightarrow A[r]$ return i+1

1.0 1) $V \Rightarrow 511$ elementos

busca binária = $O(\log n)$, logo

$\log 511 \approx 9$ comparações

2) Busca binária

n elementos $\Rightarrow \log n \Rightarrow$ exemplo 10 elementos ≈ 4

n^2 elementos $\Rightarrow \log n^2$ $n=10$ 100 elementos ≈ 7

3) $1ms = 0,001s$

$0,001s = 100$

$1s = 100.000$

$$\Rightarrow \frac{n^3}{10^5} = t \Rightarrow \frac{(10^7)^3}{10^5} = t \Rightarrow \frac{10^{21}}{10^5} = t$$

$$t = 10^{16}$$

$$10^{13}$$

4) $100n^2 + 30n + 6$ provando $100 + 30 + 6$

$$100n^2 + 30n + 6 < 136n^2$$

teste $n=2$ $100 \cdot 2^2 + 30 \cdot 2 + 6 < 136 \cdot 2^2$

$$466 < 544$$

5) $A = 100n$

$B = 3n^2$

$$100n < 3n^2$$

tabela

n	$100n$	$3n^2$
31	3100	2883
32	3200	3072
33	3300	3267
34	3400	3468
35	3500	3675

n: o algoritmo A é melhor quando $n \geq 34$

$$6) f(n) = \begin{cases} x, & \text{se } n = 0 \\ 2 \cdot f(n-1) + y, & \text{se } n > 0 \end{cases}$$

ex: $x = 25$ / $n = 4$ / $y = 15$

$$f(4) = 2 \cdot f(3) + 15 = 2 \cdot 305 + 15 = 625$$

$$f(3) = 2 \cdot f(2) + 15 = 2 \cdot 145 + 15 = 305$$

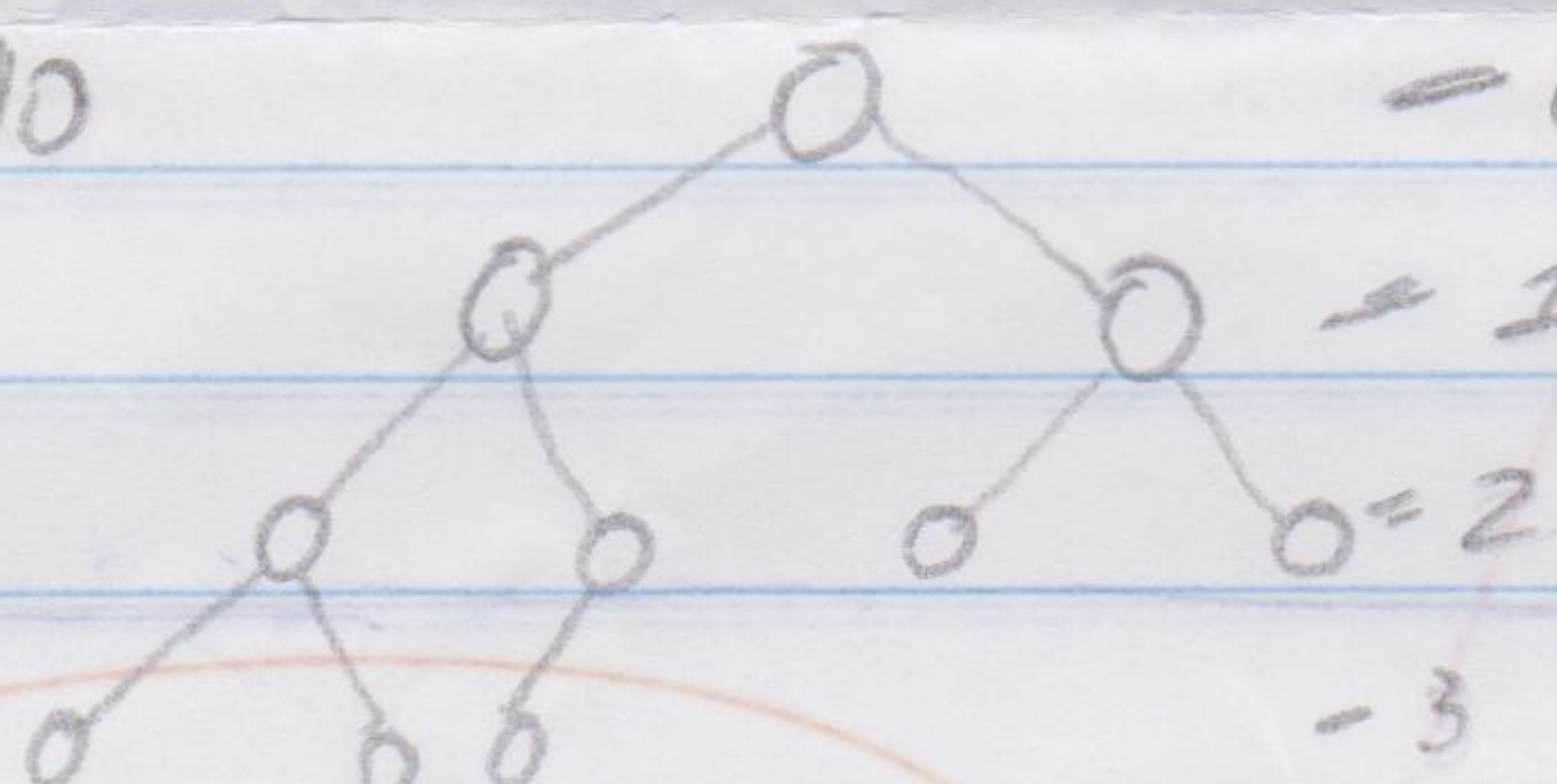
$$f(2) = 2 \cdot f(1) + 15 = 2 \cdot 65 + 15 = 145$$

$$f(1) = 2 \cdot f(0) + 15 = 2 \cdot 25 + 15 = 65$$

$$f(0) = 25$$

7) Heap de n elementos tem altura = $\lfloor \log n \rfloor$

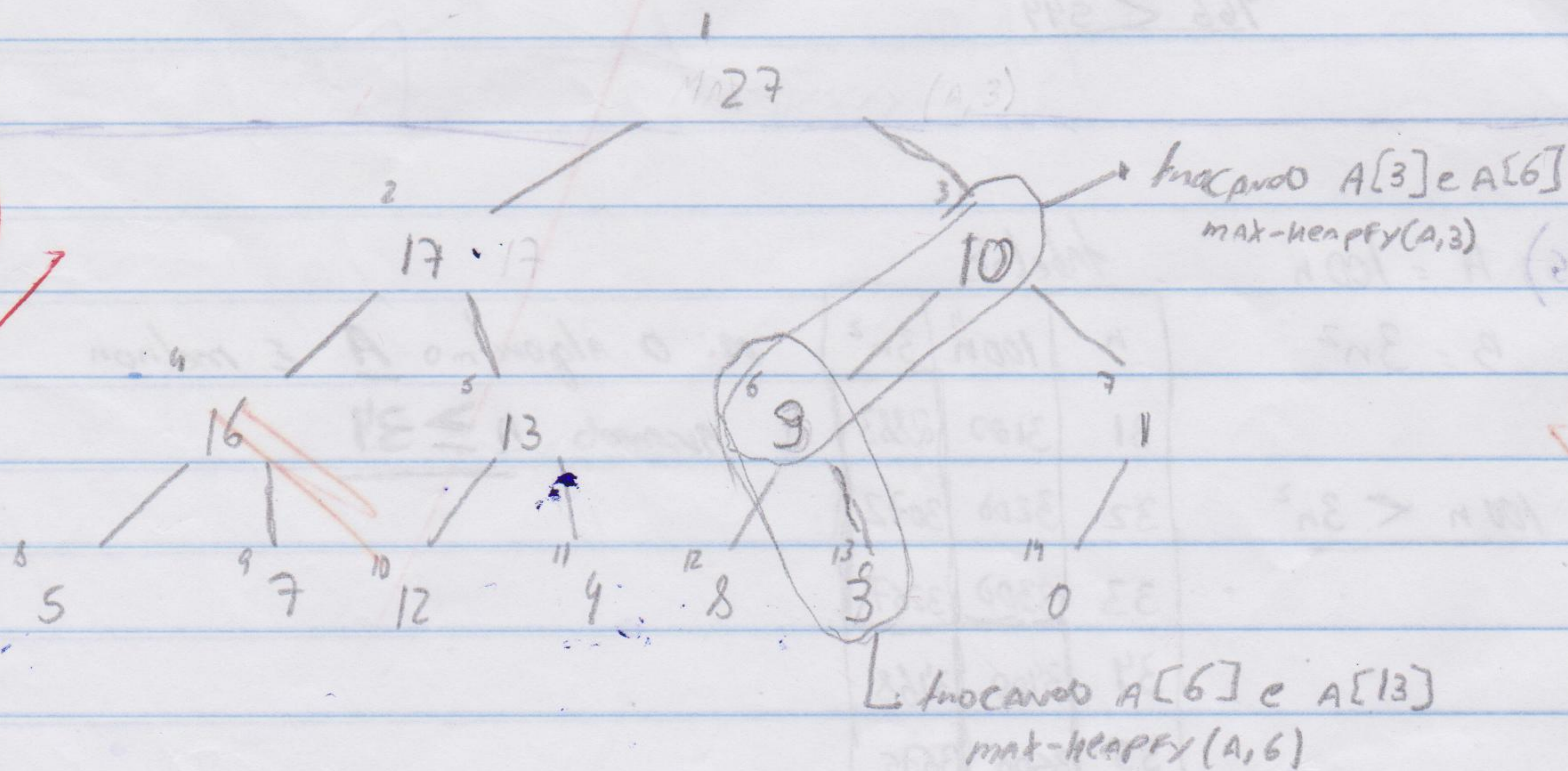
ex: $n = 10$



altura = 4

$\lfloor \log 10 \rfloor = 4$

8) $A = [27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0]$



Algoritmo de heapsort

Algoritmo de heapsort

max-heapify(A, 3)

max = 6

max-heapify(A, 6)

max = 13

max-heapify(A, 13)

Complexidade: $O(\log n)$

9) $A = [2, 2, 2]$

Particionamento(A, 1, 3)

$X = 2$

$i = 0$

For i to 2

If $2 \leq 2$ ($j = 1$)

$I = 1$

$A[1] = A[1]$

If $2 \leq 2$ ($j = 2$)

$I = 2$

$A[2] = A[2]$

$A[3] = A[3]$

return 3

n. o valor de q retornado é 3