PROVA 1999/1

Questão 1 - $2^{n+1} = O(2^n)$? $2^{2n} = O(2^n)$?

Para que $2^{n+1} = O(2^n)$ devem existir constantes positivas c e N_0 para as quais:

$2^{n+1} \leq c \cdot 2^n \quad \forall n \geq N_0$, onde

$2 \leq c$

Portanto, tomamos $c = 2$ e $N_0 = 1$.

Para que $2^{2n} = O(2^n)$ devem existir constantes positivas c e N_0 para as quais:

$2^{2n} \leq c \cdot 2^n \quad \forall n \geq N_0$, onde

$2^n \leq c$

Como 2^n trata-se de uma função monótona crescente, e não pode ser uma constante o que é absurdo. Portanto 2^{2n} não é $O(2^n)$.

Questão 2 -

1. $f(n) = \Omega_1(g(n))$

$f(n) > A\,g(n)$ para $n > n_0$ não implica que vale para valores infinitos de n

contra-exemplo

$f(n) = n \log n$ $\quad f(n) \geq A \cdot g(n) \quad \forall n \geq 2$, não

$g(n) = n$ $\quad$ pq valor de n

$A = 1$

2. $f(n) = \Omega_2(g(n))$

$f(n) > A\,g(n)$ para infinitos valores de n implica que vale para $n \geq N_0$

Questão 4 - Pg 200

Questão 9 -

1- Acha-se máximo de cada linha: $O(n.m)$

Acha-se o máximo do vetor $O(n)$

total $O(n.m + n)$

2- Acha-se o máximo de cada coluna: $O(m.n)$

Acha-se o máximo do vetor $O(m)$

total $O(m.n + m)$

Supondo que $n < m$ então o algoritmo 1 com consumo $O(n.m + n)$ realiza menos comparações

Questão 4 - Exiyam: n números em $\{1, n^3\}$ vamos representá-los na base n na forma:

$a_2 n^2 + a_1 n + a_0$

onde a_2, a_1 e a_0 são algarismos em $\{0..n\}$.

a_0 é o algarismo menos significativo e a_2 o mais significativo

O algoritmo Ordem é equivalente ao RadixSort salvo algumas modificações

Ordem(A, n)

1 Ordene(A, n) ▷ como chave o menos significativo

2 Ordene(A, n) ▷ como chave a_1

3 Ordene(A, n) ▷ como chave o mais significativo

O algoritmo Ordene pode ser implementado por CountingSort que recebe um vetor e um inteiro k e ordena em tempo $O(n+k)$ números em $\{0..k\}$

Como $k = n$ o consumo de cada linha de Ordem é $O(n)$

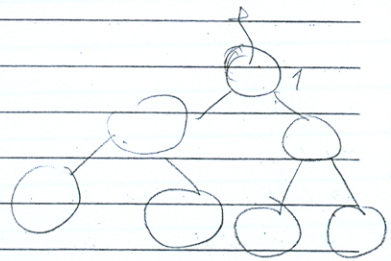
ajar
abacaxia
abacaxi.

Questão 5 - / /

x	x	x	x	y	x	x	x
x	x	x	x	x	x	y	y
y	x	x	x	y		x	y
x	x	x	y	x			y
y	x	x	x	x			y
	x	x	x				y

Questão 6 - 1 - $O(n^3)$
2 - Pg 124

Questão 7 -
Heap mínimo (normal): a) $O(1)$
b) $O(\lg n)$
c) $O(\lg n)$
d) $O(\lg n)$



Criar um heap com k elementos
Inicialização $O(k)$
Inserção $O(\lg k)$
Extração $O(\lg k)$
Reclassificação $O(\lg k)$

Criar um heap mínimo com k elementos e em cada nó possui uma lista de itens de mesma chave. Ao inserir faz-se no último lugar da lista e ao extrair, extrai-se do 1º lugar da lista. Como k é constante $\lg k$ é constante então

/ /

Todas as operações levam tempo constante.

2) AlgoritmoA(G, w) aresta |V|
1 Criar vetor Adj
2 Para cada $v \in V(G)$
3 MakeSet(v)
4 Obter um vetor k-ordenado de acordo com o peso crescente. (countingSort)
5 Para cada $uv \in Adj(v)$
6 se findset(u) ≠ findset(v)
7 $A \leftarrow A \cup \{uv\}$ (union(u,v))
8 Remover(Q, u)
9 devolva A

Consumo de tempo:
1 - $O(|V(G)|)$
2-3 $O(|V(G)|)$
4 $O(|V(G)| \cdot \lg k)$
5-8 $O(|V(G)| \cdot \lg^* |V(G)|)$

Como k é constante então a linha 4 tem consumo $O(|V(G)|)$ e como $\lg^* |V(G)|$ tem um crescimento muito lento podemos considerar como constante
Assim o algoritmo A tem consumo $O(|V(G)|)$

Questão 8 -
1 - $O(n \lg n)$

2 - a) $O(n)$
b) $O(n \lg n)$
c) $O(n \lg n)$
d) $O(n)$

Totalmente geral: 1 - $O(n \lg n)$
2 - $O(n \lg n)$

Cada nome ocorre $\Theta((\lg n)^r)$ p/ $r > 1$
1 - $O(n \lg n)$
2 - $\frac{n}{(\lg n)^2} \lg\left(\frac{n}{\lg^2 n}\right) = \frac{n}{(\lg n)^2} \cdot \lg n - \lg(\lg n)^2$

$\frac{n}{\lg n} \cdot \lg n - 2 \lg \lg n = \frac{n}{\lg^2 n} \cdot \lg n$ /

3- Cada termo ocorre n^ϵ vezes para algum ϵ entre 0 e 1

1- $O(n \lg n)$

2- $O\left(\frac{n}{n^\epsilon} \lg\left(\frac{n}{n^\epsilon}\right)\right)$

$= O\left(\frac{n}{n^\epsilon}(\lg n - \lg n^\epsilon)\right)$

$= O\left(\frac{n}{n^\epsilon} \lg n - \frac{\epsilon \cdot n}{n^\epsilon} \lg n\right)$

$= O\left(\frac{n}{n^\epsilon} \lg n\right)$

$= O(n^{1-\epsilon} \lg n)$