Lista de Exercícios - Projeto e Análise de Algoritmos

Luiz Alberto do Carmo Viana

12 de setembro de 2019

Questão 1

Dado um grafo direcionado $G = (V, A)$, queremos determinar se G possui uma celebridade, isto é, um vértice $v \in G$ que conhece nenhum vértice em $V \setminus v$, mas que é conhecido por todo vértice em $V \setminus v$. Para determinar a existência de uma celebridade em G, podemos verificar, em tempo constante, se um par $(u, v) \in V^2$ pertence a A. Desenvolva, por indução, um algoritmo que, dado um grafo direcionado $G = (V, A)$ como entrada, decide se G possui uma celebridade. Dica: dado um par $(u, v) \in V^2$ qualquer, podemos descartar ou u ou v da lista de candidatos a celebridade. Isso nos ajuda a utilizar a hipótese da indução.

Questão 2

Dados $a, b \in \mathbb{N}$, prove que $(n + a)^b \in \Theta(n^b)$.

Questão 3

$2^{n+1} \in O(2^n)$? $2^{2n} \in O(2^n)$?

Questão 4

Dada uma função $g(n)$, prove que $o(g(n)) \cap \omega(g(n)) = \emptyset$.

Questão 5

Prove que $n! \in \omega(2^n)$ e que $n! \in o(n^n)$.

Questão 6

Dado $d \in \mathbb{N}$, tome o polinômio $p(n) = \sum_{i=0}^{d} a_i n^i$, onde $a_0, \ldots, a_d \in \mathbb{R}$. Para $p(n)$ ter grau d, vamos assumir que $a_d > 0$. Prove que:

- se $k \geq d$, então $p(n) \in O(n^k)$.
- se $k \leq d$, então $p(n) \in \Omega(n^k)$.
- se $k = d$, então $p(n) \in \Theta(n^k)$.
- se $k > d$, então $p(n) \in o(n^k)$.

- se $k < d$, então $p(n) \in \omega(n^k)$.

Questão 7

Tome as funções $(\frac{3}{2})^n, \ln(\ln n), 2^{\log n}, n^3, (\log n)^{\log n}, 2^{\sqrt{2\log n}}, \sqrt{2}^{\log n}, \log^2 n, n2^n, e^n,$ $n, n^2, \log(n!), n^{\log(\log n)}, 4^{\log n}, 2^n, n!, 2^{2^n}, \ln n, (n+1)!, n\log n, n^{\frac{1}{\log n}}, 1, \sqrt{\log n}, 2^{2^{n+1}}$. Ordene-as assintoticamente.

Questão 8

Tome $f(n)$ e $g(n)$ funções assintoticamente positivas. Prove ou dê um contra-exemplo:

- $f(n) \in O(g(n))$ implica que $g(n) \in O(f(n))$.
- $f(n) + g(n) \in \Theta(\min(f(n), g(n)))$.
- $f(n) \in O(g(n))$ implica que $\log(f(n)) \in O(\log(g(n)))$, desde que, para valores de n suficientemente altos, $f(n) \geq 1$ e $\log(g(n)) \geq 1$.
- $f(n) \in O(g(n))$ implica que $2^{f(n)} \in O(2^{g(n)})$.

Questão 9

Forneça um limite superior assintótico (em termos de O) para $T(n)$, onde::

- $$T(1) = 1$$ $$T(n) = T(n-1) + 1$$
- $$T(1) = 1$$ $$T(n) = T(n-1) + n$$
- $$T(1) = 1$$ $$T(n) = T(\frac{n}{2}) + 1$$
- $$T(1) = 1$$ $$T(n) = 3T(\frac{n}{2}) + n$$

- $$\begin{aligned} T(1) &= 1 \\ T(2) &= 1 \\ T(n) &= 4T(\frac{n}{3}) + n \end{aligned}$$

- $$\begin{aligned} T(1) &= 1 \\ T(n) &= 4T(\frac{n}{2}) + n^2 \end{aligned}$$

- $$\begin{aligned} T(2) &= 1 \\ T(n) &= 3T(\sqrt{n}) + \log n \end{aligned}$$

Questão 10
Aplique o Teorema Mestre para:

- $T(n) = 2T(\frac{n}{4}) + 1$
- $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$
- $T(n) = 2T(\frac{n}{4}) + n$
- $T(n) = 2T(\frac{n}{4}) + n^2$
- $T(n) = T(\frac{n}{2}) + 1$

Questão 11
O Teorema Mestre pode ser aplicado a $T(n) = 4T(\frac{n}{2}) + n^2 \log n$? De toda forma, forneça um limite superior assintótico para $T(n)$.

Questão 12
Para resolver um certo problema, existem três algoritmos:

1 resolve o problema dividindo-o em em cinco subproblemas com metade do tamanho da entrada, resolvendo-os recursivamente, e combinando suas soluções em tempo linear;

2 resolve o problema (com tanaho de entrada n) resolvendo recursivamente dois subproblemas de tamanho $n - 1$ e então combinando suas soluções em tempo constante;

3 resolve o problema dividindo-o em nove subproblemas com um terço do tamanho da entrada, resolvendo-os recursivamente, e combinando suas soluções em tempo quadrático.

Qual algoritmo deve ser escolhido para resolver esse problema?

Questão 13
Dado um array de inteiros com n posições, apresente um algoritmo que remova elementos duplicados em tempo $\Theta(n \log n)$. Para fins de esclarecimento, se um array tem, digamos, duas posições com o elemento 3, o array a ser produzido deve ter exatamente uma posição com o elemento 3.

Questão 14
Dado um array a de inteiros com n posições, apresente um algoritmo que determina se existe $1 \leq i \leq n$ tal que $a_i = i$. Seu algoritmo deve ter complexidade $O(\log n)$.

Questão 15
Tome um array infinito de inteiros em que as n primeiras posições contém inteiros ordenados e as demais contém ∞. O valor de n não é conhecido a priori. Descreva um algoritmo que, dado um inteiro x como entrada, determina se o array contém x em alguma de suas entradas. O algoritmo deve ter custo $O(\log n)$.

Questão 16
Dados k arrays ordenados, cada um com n inteiros, gostaríamos de combiná-los em um único array ordenado de kn inteiros.

- Usando o procedimento `merge` do algoritmo `mergesort`, combine os dois primeiros arrays, depois combine o array resultante com o terceiro, com o quarto, e assim por diante. Qual a complexidade dessa abordagem, em termos de k e n?
- Utilizando divisão-e-conquista, desenvolva um algoritmo mais eficiente.

Questão 17
Dados dois arrays ordenados com m e n inteiros, respectivamente, desenvolva um algoritmo $O(\log m + \log n)$ para encontrar o k-ésimo menor elemento na "união" dos dois arrays.

Questão 18
Um array possui um elemento majoritário e se mais da metade de suas posições são iguais a e. Dado um array a de n posições, gostaríamos de determinar se a possui um elemento majoritário e, em caso afirmativo, determinar esse elemento. Não iremos assumir que nossos elementos são ordenáveis, mas apenas que são comparáveis por igualdade.

- Mostre como resolver esse problema em tempo $O(n \log n)$
- Dados os n elementos de a, crie $\frac{n}{2}$ pares. Para cada par: se ambos os elementos são iguais, guarde um deles; caso sejam diferentes, guarde nenhum

deles. Mostre que você guardou no máximo $\frac{n}{2}$ elementos, e eles têm um elemento majoritário sse a também tem.

- Qual a complexidade do algortimo descrito nessa abordagem?