MCTA003-17 – Análise de Algoritmos
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC
Profa. Carla Negri Lintzmayer

Lista 2

Entrega: até 23h55 do dia 09/07/2018

- Submeta ao tidia um único arquivo PDF com as suas soluções escaneadas.
- Seja o mais **formal** possível em todas as respostas.
- Identifique devidamente cada exercício.
- Identifique devidamente os autores da lista (caso você tenha feito em dupla).
- A lista é uma forma de treino para a prova, que não terá consulta. Evite plágio!

1. Em cada situação a seguir, prove se $f(n) = O(g(n))$ ou $f(n) \neq O(g(n))$, e se $f(n) = \Omega(g(n))$ ou $f(n) \neq \Omega(g(n))$. Comente quando $f(n) = \Theta(g(n))$. Considere que a e b são constantes positivas e que log está na base 2:

   (a) $f(n) = \log_a n$ e $g(n) = \log_b n$

   (b) $f(n) = 100^{an}$ e $g(n) = 100^n$

   (c) $f(n) = n^2$ e $g(n) = n \log^2 n$

2. Suponha que $T(1) = c$, onde c é uma constante positiva (você pode assumir que $c = 1$ se preferir). Resolva as seguintes recorrências com notação O (quando não indicado, use o método que lhe for mais conveniente):

   (a) $T(n) = aT(\frac{n}{a}) + n$, onde a é inteiro positivo não nulo (método de iteração)

   (b) $T(n) = 4T(\frac{n}{2}) + n$ (método de substituição, suponha $T(n) = \Theta(n^2)$)

   (c) $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + n$ (árvore de recursão e método de substituição)

   (d) $T(n) = 2T(\frac{n}{2}) + n \log n$ (método mestre)

3. Seja A um vetor ordenado onde os elementos são distintos e estão nas posições 1 a n. Mostre um algoritmo que decide se existe índice i, com $1 \leq i \leq n$, tal que $A[i] = i$ em tempo $O(\log n)$.

4. Vimos em sala o algoritmo Corrige-Descendo que auxilia a manter a propriedade de heap. Outro algoritmo importante que auxilia na manutenção da propriedade de heap é o Corrige-Subindo. Ele recebe um índice i e o vetor A onde até a posição $i-1$ temos uma heap. O objetivo desse algoritmo é garantir que a heap vá até a posição i. Sua ideia é trocar o elemento problemático com seu pai repetidamente (até atingir a raiz ou quando a propriedade de heap for restaurada). Seu pseudocódigo é dado a seguir.

```
function Corrige-Subindo(A, i)
    if i ≥ 2 e A[i] > A[⌊i/2⌋] then
        troque A[i] com A[⌊i/2⌋]
        Corrige-Subindo(A, ⌊i/2⌋)
```

Use esse algoritmo e o Corrige-Descendo para criar um algoritmo chamado Atualiza-Heap que deve receber um heap máximo A, um índice i e um novo valor k. Esse algoritmo deve atualizar o valor de $A[i]$ para k e corretamente restaurar a propriedade de heap, caso ela seja violada. Analise o tempo do seu algoritmo.

5. Escreva um algoritmo que recebe um grafo G e dois vértices s e v e retorna a sequência de vértices de um $s - v$ caminho em tempo $O(n + m)$.

6. Vimos em sala a ideia da busca em largura (BFS). Seu pseudocódigo é dado a seguir.

   function BFS(G, s)
       seja Q uma fila
       marque s como explorado e o coloque em Q (ou seja, no fim)
       while $Q \neq \emptyset$ **do**
           remova v, o primeiro vértice de Q
           for all aresta vx **do**
               if x é não explorado **then**
                   marque x explorado e o coloque em Q (ou seja, no fim)

   Utilize a BFS para criar um algoritmo que verifica se um grafo é conexo ou não em tempo $O(n + m)$.

7. Utilize a DFS para criar um algoritmo que verifica se existe um ciclo em um grafo G em tempo $O(n + m)$.

8. Apresente um algoritmo que encontra componentes fortemente conexas em digrafos e mostre sua execução no digrafo G definido por $V = \{a, b, c, d, e, f, g, h, i\}$ e $E = \{ad$, de, ea, ba, bf, fg, gb, cb, ch, hi, $ic\}$. Você pode procurar por esse algoritmo em algum livro.

Exercícios extras (se você quiser treinar mais – não valem nota e não precisam ser entregues)

1. Simule passo a passo a execução do algoritmo Heapsort visto em sala no vetor $A = [6, 1, 8, 7, 3, 9, 5, 2, 4]$ (isto é, mostre a construção da heap e o que ocorre em cada chamada ao Corrige-Descendo).

2. Prove que o algoritmo Heapsort visto em aula está correto, isto é, que ele corretamente ordena qualquer vetor dado na entrada. *Dica:* primeiro prove que o algoritmo Corrige-Descendo está correto por indução na altura do i-ésimo elemento e em seguida prove por invariante de laço o Heapsort.

3. Mostre que uma árvore binária tem altura $\Omega(\log n)$.

4. Prove que o tempo de execução do algoritmo Constroi-Heap visto em sala é $O(n)$.

5. Dado um vetor $A[1..n]$, uma *inversão* é um par $\{i, j\}$ com $1 \leq i < j \leq n$ tal que $A[i] > A[j]$. Faça um algoritmo que conta a quantidade de inversões em um vetor $A[1..n]$ em tempo $O(n \log n)$.

6. Faça um algoritmo para verificar se um dado grafo G é bipartido em tempo $O(n + m)$.

7. Mostre como modificar a BFS para calcular a distância de s aos vértices alcançáveis a partir de s. Prove que o que o seu algoritmo calcula é de fato a distância mínima entre s e os vértices alcançáveis a partir dele.