CCM-001 – Análise de Algoritmos e Estruturas de Dados
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC
Profs. Carla N. Lintzmayer e Guilherme O. Mota

Lista 2

- Seja o mais **formal** possível em todas as respostas.
- A lista é uma forma de treino para a prova, que não terá consulta. Evite plágio!
- Fique à vontade também para procurar exercícios nos livros recomendados na bibliografia. CLRS é sigla para o livro "Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C.. *Introduction to Algorithms.* 2nd ed. MIT Press. 2002."

1. 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 $O(\log n)$.
2. Mostre que uma árvore binária tem altura $\Omega(\log n)$.
3. Escreva um algoritmo 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.
4. 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 do heap e o que ocorre em cada chamada ao Corrige-heap-para-baixo).
5. 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-heap-para-baixo está correto por indução na altura do i-ésimo elemento e em seguida prove por invariante de laço que o Heapsort está correto.
6. Escreva um algoritmo que recebe um grafo $G = (V, E)$ e dois vértices s e v e retorna a sequência de vértices de um sv-caminho em tempo $O(|V| + |E|)$.
7. Utilize a busca em largura para criar um algoritmo que verifica se um grafo $G = (V, E)$ é conexo ou não em tempo $O(|V| + |E|)$.
8. Utilize a busca em profundidade para criar um algoritmo que verifica se existe um ciclo em um grafo $G = (V, E)$ em tempo $O(|V| + |E|)$.
9. 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 $E = \{ad, de, ea, ba, bf, fg, gb, cb, ch, hi, ic\}$.
10. Faça um algoritmo para verificar se um dado grafo $G = (V, E)$ é bipartido em tempo $O(|V| + |E|)$.
11. Modifique 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.
12. (Desafio) 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 $O(n \log n)$.