SSC0503 - Introdução à Ciência de Computação II
Lista de Exercícios - Tópico 5 Part I
Professor: Claudio Fabiano Motta Toledo (`claudio@icmc.usp.br`)
Estagiário PAE: Jesimar da Silva Arantes (`jesimar.arantes@usp.br`)

---

1. Desenvolva manualmente o algoritmo insertion sort sobre o arranjo $A = [13, 19, 9, 5, 12, 8, 7, 4]$.
2. Desenvolva manualmente o algoritmo merge sort sobre o arranjo $A = [3, 41, 52, 26, 38, 57, 9, 49]$.
3. Reescreva a rotina merge do algoritmo merge sort tal que ela não precise utilizar o valor ∞ nos vetores L e R.
4. Pratique a execução do algoritmo heapsort nos vetores dos exercícios 1, 2, 12 e 13.
5. Quais são os números mínimo e máximo de elementos numa heap de altura h?
6. Mostre que uma heap com n elementos tem altura $\lfloor lg\ n \rfloor$.
7. Mostre que, armazenando uma heap com n elementos em um vetor, os nós folhas são os nós indexados por $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \cdots, n$.
8. Escreva um algoritmo para a rotina Min-Heapify(A,i). Compare o tempo de execução da Min-Heapify com Max-Heapify.
9. Qual o efeito de chamar Max-Heapify(A,i), quando o elemento A[i] é maior que seus filhos? e quando i>A.heap-size/2?
10. O código para Max-Heapify é bastante eficiente, exceto pela chamada recursiva na linha 10. Escreva uma rotina Max-Heapify que usa um controle iterativo ao invés de recursão.
11. Mostre que há no máximo $\lceil n/2^{h+1} \rceil$ nós de altura h em qualquer heap com n elementos.
12. Execute Max-Heap_Insert(A, 10) na heap $A = [15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1]$.
13. Desenvolva manualmente o algoritmo quicksort sobre o arranjo $A = [13, 19, 9, 5, 12, 8, 7, 4, 11, 2, 6, 21]$.
    - Pegue o primeiro elemento como pivô.
    - Pegue o elemento do meio como pivô.
    - Pegue um elemento aleatório como pivô.
    - Pegue a mediana de três como pivô.
14. Desenvolva manualmente o algoritmo quicksort com pivô escolhido aleatoriamente sobre os arranjos:
    - $A = [0, 1, 3, 4, 5, 7, 9]$
    - $A = [1, 0, 4, 3, 5, 9, 7]$
    - $A = [1, 2, 3, 4, 3, 2, 1]$

15. Qual é o tempo de execução de quicksort quando todos os elementos do arranjo A têm o mesmo valor?

16. Suponha que as divisões em todo nível do quicksort são na proporção $1 - \alpha$ para α, onde $0 \leq \alpha \leq 1/2$ é uma constante. Mostre que a profundidade mínima de uma folha na árvore de recursão é aproximadamente $-lg\ n/lg\ \alpha$ e a profundidade máxima é aproximadamente $-lg\ n/lg\ (1 - \alpha)$. Não se preocupe com arredondamentos.

17. Mostre que o quicksort no melhor caso executa em $\Omega(n \cdot lg\ n)$.

18. No algoritmo Randomized-Quicksort, quantas chamadas são feitas ao gerador de números aleatórios Random no pior caso? e no melhor caso? Dê sua resposta em termos da notação Θ.

19. Desenvolva um programa em C que faça uma implementação combinada do quicksort com o insertion sort. O quicksort será executado até que o partição fique menor que um determinado valor k (por exemplo, $k = 10$, $k = 20$) então faça uma chamada para o método insertion sort. Avalie e compare o desempenho isolado do quicksort e insertion sort.

20. Desenvolva manualmente o algoritmo bubblesort sobre o arranjo $A = [13, 19, 9, 5, 12, 8, 7, 4]$. Em seguida, analise a complexidade desse algoritmo.

Algorithm 1 Bubblesort(A)

```
A é um arranjo de inteiros
for i = 1 to A.length − 1) do
  for j = A.length downto i + 1 do
    if A[i] < A[j − 1] then
      exchange A[j] with A[j − 1]
    end if
  end for
end for
```