Centro Universitário Senac
Bacharelado em Ciência da Computação
Análise e projeto de algoritmos

Professor: Leonardo Takuno
{leonardo.takuno@gmail.com}

28 de abril de 2020

1. O algoritmo `Desce-Heap(A, n, i)` visto em aula, foi utilizado para manutenção de um heap de máximo, "*descendo*" um elemento até sua posição correta. Uma outra maneira de obter o mesmo resultado seria "*subindo*" um elemento até a posição adequada. Sendo assim, escreva o pseudocódigo para uma versão recursiva do algoritmo `Sobe-Heap(A, i)`.

2. Execute o algoritmo `particiona` com a entrada $A = \{5, 9, 6, 4, 3, 7, 1\}$ e $A = \{11, 10, 4, 3, 3, 2\}$.

3. Qual o resultado do algoritmo `particiona` quando os elementos de $A[]$ são todos iguais ? E quando os elementos de $A[]$ são crescentes ?

4. O algoritmo `particiona` produz um rearranjo estável a partir do vetor $A[]$, ou seja, preserva a ordem relativa de elementos de mesmo valor ?

5. Escreva uma versão recursiva do algoritmo `particiona`.

6. 5. Analise e discuta a versão abaixo do algoritmo Particiona, ela funciona? Para um mesmo vetor A[] essa versão e a versão anterior tem o mesmo resultado, ou seja, na saída de ambos os algoritmos o vetor A[] fica idêntico e é encontrada a mesma posição para pivô ?

```
algoritmo particiona2( A, p, r )
   x <- A[r]
   i <- p
   para j de p até r - 1 faça
      se A[j] <= x então
         troca(A[i], A[j])
         i <- i + 1
      fim-se
   fim-para
   troca(A[i], A[r])
   retorne i
fim.
```

7. Apresente o diagrama de execução do algoritmo QuickSort para $A = \{4, 6, 7, 1, 3, 4\}$, lembre-se que no seu diagrama de execução as chamadas recursivas devem apresentar os valores das variáveis p, r e q.

8. Escreva o pseudocódigo (ou implementação) de uma versão iterativa do algoritmo de ordenação `QuickSort`. (**Dica:** Utilize a estrutura de dados *pilha*)

9. Analise e discuta a versão abaixo do algoritmo QuickSort, ela funciona? O que acontece se trocar a linha "enquanto $p < r$ faça" por "se $p < r$ então" ?

```
algoritmo QuickSort( A, p, r )
   enquanto p < r faça
      j <- Particiona(A, p, r)
      QuickSort(A, p, j-1)
      p <- j + 1
   fim-enquanto
fim.
```