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

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

3 de novembro de 2019

1. Considere o seguinte algoritmo, cujo argumento n é uma potência de 2. O algoritmo não faz nada útil.

```
ALGO(n)
1  if n = 1
2     then return 1
3  for j ← 1 to 8
4        do z ← ALGO(n/2)
5  for i ← 1 to n^3
6        do z ← 0
```

   (a) (0,5) Seja $T(n)$ o número de vezes que a atribuição $z \leftarrow 0$ é executada. Escreva uma recorrência que define $T(n)$.

   (b) (1,0) Mostre, **sem usar** o Teorema Mestre, que $T(n)$ é $\Omega(n^3 \log n)$.

   (c) (1,0) Troque "8" por "7" no algoritmo e determine, usando o Teorema Mestre, a complexidade do algoritmo.

2. Um algoritmo age sobre que dependem de um parâmetro n. Explique o significado das seguintes expressões:

   (a) (0,5) "O consumo de tempo do algoritmo é $O(n^3 \log n)$"

   (b) (0,5) "O consumo de tempo do algoritmo é $\Omega(n^2)$"

3. (a) (1,5) Escreva um procedimento que recebe um vetor A de n inteiros distintos e devolve o vetor A ordenado, em ordem crescente, usando *Bubble Sort*.

   (b) (2,0) Prove a corretude do procedimento.

   (c) (2,0) Determine a complexidade de pior e de melhor caso. Indique para que tipo de instância o melhor e o pior caso ocorrem.

4. Merge Sort

```
MERGE-SORT(A, p, r)
1  if p < r
2     then q ← ⌊(p + r)/2⌋
3          MERGE-SORT(A, p, q)
4          MERGE-SORT(A, q + 1, r)
5          MERGE(A, p, q, r)
```

5. Inserção em árvore:

```
TREE-INSERT(T, z)
 1  y ← NIL
 2  x ← root[T]
 3  while x ≠ NIL
 4        do y ← x
 5           if key[z] < key[x]
 6              then x ← left[x]
 7              else  x ← right[x]
 8  p[z] ← y
 9  if y = NIL
10     then root[T] ← z                    ▷ Tree T was empty
11     else  if key[z] <  idkey[y]
12              then left[y] getsz
13              else  right[y] ← z
```

6. outro:

```
SEGMENTS-INTERSECT(p1, p2, p3, p4)
 1  d1 ← DIRECTION(p3, p4, p1)
 2  d2 ← DIRECTION(p3, p4, p2)
 3  d3 ← DIRECTION(p1, p2, p3)
 4  d4 ← DIRECTION(p1, p2, p4)
 5  if ((d1 > 0 and d2 < 0) or (d1 < 0 and d2 > 0)) and
          ((d3 > 0 and d4 < 0) or (d3 < 0 and d4 > 0))
 6     then return TRUE
 7  elseif d1 = 0 and ON-SEGMENT(p3, p4, p1)
 8     then return TRUE
 9  elseif d2 = 0 and ON-SEGMENT(p3, p4, p2)
10     then return TRUE
11  elseif d3 = 0 and ON-SEGMENT(p1, p2, p3)
12     then return TRUE
13  elseif d4 = 0 and ON-SEGMENT(p1, p2, p4)
14     then return TRUE
15  else return FALSE
```