Complexidade de Algoritmos – Segunda Prova

Turma 2017 Data: 19/11/2020 Prof. Manoel Ribeiro

1. Utilize uma das técnicas conhecidas de análise de algoritmos recursivos e forneça um limite assintótico θ() para o pior caso para cada algoritmo abaixo.

a) Para o algoritmo a seguir, T(n), o consumo de tempo do algoritmo no pior caso é aplicado a um vetor A de tamanho n = r - p + 1 (1,5)

```
algoritmoCAV (A, p, r)
1 se p = r
2       então devolva A[p]
3       senão q = (p + r)/2
4             x1 = algoritmoCAV (A, p, q)
5             x2 =  algoritmoCAV (A, q + 1, r)
6             y1 = s = A[q]
7             para i = q - 1 decrescendo até p faça
8                   s = A[i] + s
9                   se s > y1 então y1 = s
10            y2 = s =  A[q + 1]
11            para j =  q + 2 até r faça
12                  s = A[j] + s
13          se s > y2 então y2 =  s
14            x =  max (x1, y1 + y2, x2)
15      devolva x
```

b) (1,5)

```
int algoritmoBom(int* A, int n){
 if (n < 80)  return (A[n]);
 int x=0, i, j, k;
for (i = 0; i<2; i++){
    for (j=0; j<4; j++){
          for (k=0; k<n/2; k++){
```

```
                    A[k] = A[j] - A[n-j];
                }
            }
            x += algoritmoBom(A, n/4);
        }
        return x;
    }
```

2. Determine θ() para o pior caso para o algoritmo abaixo, para o vetor A, de tamanho n= r –p +1. O vetor auxiliar S, tem o mesmo tamanho de A. (1,5)

```
algoritmoCA (A, n)
1       S[1] =  A[1]
2       para q =  2 até n faça
3               se S[q - 1] >= 0
4                       então S[q] =  S[q-1] + A[q]
5                       senão S[q]  = A[q]
6       x = S[1]
7       para q =  2 até n faça
8               se S[q] > x então x =  S[q]
9       devolva x
```

3. Calcule a complexidade, no pior caso e no melhor caso em notação θ(), do fragmento de código abaixo: (1,5)

```
1       int i , j , k ;
2       R[ i ] [ j ] = 0;
3       for ( i =0; i < N; i ++){
4               if(m <100) {
```

```
5               for ( j =0; j < N*N; j ++)
6               R[ i ] [ j ] += A[ i ] [ k ] *B[ k ] [ j ] ;
 7               }
 8               else {
 9               for ( k=1; k < N; k*=2)
10                      R[ i ] [ j ] -= A[ i ] [ k ] *B[ k ] [ j ] ;
11              }
12      }
```

5. Escreva dois algoritmos, um interativo e outro usando a técnica da divisão e conquista, que receba um vetor A de n números inteiros e determine o maior e o segundo maior elemento desse vetor. Teste ambos os algoritmos em C ou em Python. Calcule a complexidade, no pior caso em notação θ() para os dois algoritmos e analise qual o que tem melhor desempenho. (3,0)