UNIVERSIDADE FEDERAL DE LAVRAS

DEPARTAMENTO DE CIÊNCIAS DA COMPUTAÇÃO

GCC 253 – Complexidade e Projeto de Algoritmos

Professor: Douglas H. S. Abreu

Exercício de Fixação de Conceitos II

1. Considere o algoritmo a seguir, sendo que a chamada inicial $FIND-MAXIMUM-SUBARRAY(A, 1, A.length)$ encontrará um subarranjo máximo de $A[1\,..\,n]$. O procedimento recursivo $FIND-MAXIMUM-SUBARRAY$ retorna uma tupla que contém os índices que demarcam um subarranjo máximo, juntamente com a soma dos valores em um subarranjo máximo.

   :

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
left-sum = −∞
sum = 0
for i = mid downto low
    sum = sum + A[i]
    if sum > left-sum
        left-sum = sum
        max-left = i
right-sum = −∞
sum = 0
for j = mid + 1 to high
    sum = sum + A[j]
    if sum > right-sum
        right-sum = sum
        max-right = j
return (max-left, max-right, left-sum + right-sum)
```

```
FIND-MAX-CROSSING-SUBARRAY(A; low; high)
if high == low
    return (low; high;A[low])                    // caso base: só um elemento
else mid = ⌊(low + high)/2⌋
    (left-low, left-high, left-sum) =
        FIND-MAXIMUM-SUBADRRAY(A, low, mid)
    (right-low, right-high, right-sum) =
        FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
    (cross-low, cross-high, cross-sum) =
        FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
    if left-sum ≥ right-sum e left-sum ≥ cross-sum
        return (left-low, left-high, left-sum)
    elseif right-sum ≥ left-sum e right-sum ≥ cross-sum
        return (right-low, right-high, right-sum)
    else return (cross-low, cross-high, cross-sum)
```

   a. Apresente a aplicação do algoritmo
   b. O que $FIND-MAXIMUM-SUBARRAY$ retorna quando todos os elementos de A são negativos?

2. Utilizando o Método da substituição.
    a. Mostre que a solução de $T(n) = T(n - 1) + n$ é $O(n^2)$.
    b. Mostre que a solução de $T(n) = T(n/2) + 1$ é $O(lg\ n)$

3. Demonstre que a solução para a recorrência $T(n) = T(n/3) + T(2n/3) + cn$, onde c é uma constante, é $(n\ lgn)$, utilizando árvore de recursão.

4. Resolva as recorrências a seguir

    a. $T(n) = 3T(\frac{n}{2}) + n$
    b. $T(n) = 3T(\frac{n}{3}) + n$
    c. $T(n) = 3T\left(\frac{n}{2}\right) + n^{\log_3 2}$
    d. $T(n) = T\left(\frac{n}{2}\right) + n$
    e. $T(n) = 4T\left(\frac{n}{2}\right) + n^2 . log^3 n$
    f. $T(n) = 2T\left(\frac{n}{5}\right) + n$
    g. $T(n) = 2T\left(\frac{n}{2}\right) + cn$

5. Os algoritmos abaixo são construídos pela técnica:
    a. Prim
    b. Kruskal
    c. Dijkstra
    d. Bellman-Ford
    e. Ordenação por inserção
    f. MergeSort
    g. Quick-sort
    h. Busca linear (encontrar um item em uma tabela, verificando sequencialmente todas as entradas)

6. Qual a diferença entre um algoritmo para calcular a sequência de Fibonacci recursivo e um algoritmo para calcular a sequência de Fibonacci recursivo desenvolvido pela técnica de programação dinâmica?
7. **(EXECÍCIO EXTRA)** programe (linguagem de sua preferência) os 2 modos Fibonacci do exercício 6 e demonstre sua eficiência.