Listex 7

João Pedro Machado Silva, BV3032477

Exercício 1:

a. Implementado.

b. Ao analisar o algoritmo “findMaxCrossingSubarray”, notamos a presença de 2 laços de repetição(for) e que os dois são independentes um do outro, sendo assim, podemos dizer que o algoritmo é linear, com T(n) = O(n).
Ao analisar o algoritmo “findMaxSubarray”, notamos que não há estruturas de repetição, porém há 2 chamadas recursivas e dividindo o array pela metade. Sendo assim, temos 2T(n/2), e temos também a chamada do algoritmo mencionado no último parágrafo, tendo complexidade O(n), então temos a complexidade de “findMaxSubarray”:T(n) = 1, para n=1 e T(n) = 2T(n) + n, para n>1.

c. Quando todos os elementos do array A são negativos, a função “findMaxSubarray” ainda retorna a submatriz com a maior soma. No entanto, como todos os valores são negativos, a submatriz que a função irá retornar será aquela contendo o único elemento com o menor valor absoluto, ou seja, o menor valor negativo.

d. Não faz sentido para arrays com todos os elementos positivos porque a submatriz com a maior soma é o array inteiro, tornando a abordagem recursiva desnecessária e menos eficiente do que simplesmente somar todos os elementos.

e. O Algoritmo encontra o subarray com a maior soma em um array de números inteiros. Ele funciona iterando pelo array e mantendo uma soma corrente do subarray atual. Se a soma corrente se torna negativa, o subarray é reiniciado, pois uma soma negativa não ajuda a encontrar uma soma máxima. O algoritmo atualiza os índices do subarray máximo e a soma máxima encontrada. Ao final, ele retorna os índices do subarray com a maior soma e o valor dessa soma.
O algoritmo possui apenas um laço de repetição, sendo assim, ele é linear e possui complexidade: T(n) = O(n).