Instituto de Computação – UNICAMP
MC202 – Estruturas de Dados – 2º Semestre de 2022
https://www.ic.unicamp.br/~lehilton/cursos/2s2022/mc202e/

Exercícios de fixação - **Divisão e Conquista**

Questão 1. Implemente uma busca binária de maneira **recursiva**.

Questão 2. No algoritmo de intercalação, alocamos um vetor auxiliar com tamanho máximo fixo `MAX`. Vimos que podemos flexibilizar essa restrição usando `malloc` para alocar memória no início da função e `free` para liberar memória no final da função. No entanto, essas funções são bem **caras**, isso é, gastam muito tempo, o que pode fazer com que o algoritmo fique mais lento já que a função `intercalar` pode ser chamada diversas vezes. Como você resolveria esse problema?

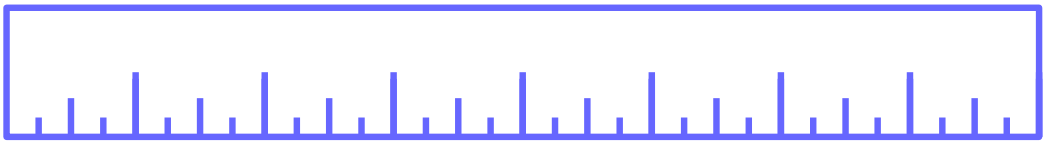
Questão 3. Eficiência do QuickSort Ao contrário do *MergeSort* que tem complexidade de pior caso $O(n \log n)$, o *QuickSort* tem complexidade de pior caso $O(n^2)$ (na verdade, exatamente $\Theta(n^2)$). Surpreendentemente, na prática esse algoritmo é tão rápido quanto o *MergeSort* e (algumas vezes) até mais eficiente. Tente responder as questões a seguir:

(a) Descreva o comportamento do particionamento para vetores: $\{1,2,3,4,5\}$, $\{5,4,3,2,1\}$, $\{1,5,4,2,3\}$. Pode ser útil tentar simular os passos iniciais.

(b) A partição do vetor nem sempre será balanceada, isso é, nem sempre irá dividir o vetor em partes iguais. Você consegue dar um exemplo de vetor com n posições em que toda chamada de particionamento deixa uma parte com apenas um elemento?

(c) Com que frequência você acha que o exemplo do item anterior acontece na prática? Tente pensar em uma maneira que diminua essa probabilidade (é necessário que o pivô sempre seja o primeiro elemento?).

Questão 4. Escreva uma função recursiva que escreva um programa LOGO (relembre o que é um programa LOGO nos exercícios anteriores) para desenhar uma régua, com marcadores de 1cm, 0,5cm, 0,25cm proporcionais, como na figura (considere que cada cm equivale a 100 pixels). Depois, tente implementar uma versão iterativa do problema.



Questão 5. Como é possível implementar o *Merge-Sort* de maneira iterativa (observar o exercício **??** pode ser útil)? Esboce ou implemente o *Merge-Sort* como um algoritmo iterativo.