1. Resumo: Você deverá escrever em C++ diferentes implementações do algoritmo de ordenação Quicksort, bem como avaliar o desempenho por elas obtido. A implementação deverá ser apresentada ao professor em 03/04/2019 ou outra data a combinar com o professor.

IMPORTANTE: lembre que o objetivo da disciplina é cada aluno aprimorar suas habilidades de programação, e também aprender os algoritmos ensinados. Portanto, incluir no trabalho código que não tenha sido escrito por você não faz sentido! Por favor, colabore com o bom andamento da disciplina, e do seu próprio aprendizado, não copiando códigos da internet ou de outros alunos. O professor está à disposição para ajudar.

2. Introdução: As versões do Quicksort baseadas em "escolhas fixas" do pivô (isto é, escolhas determinadas essencialmente pelo tamanho do intervalo a ser ordenado) deverão implementar e utilizar esta função para escolher o pivô:

int escolher_pivo (**int** primeiro, **int** ultimo)

Usando essa função, o professor tentará fornecer instâncias ruins para os Quicksorts nela baseados (observação: "primeiro" e "ultimo" são os índices do primeiro e do último elementos do intervalo a ser ordenado pelo Quicksort).

Você deverá implementar pelo menos as seguintes versões do Quicksort:

(a) **Quicksort via Índices:** escolha fixa do pivô, baseada na função "escolher_pivo", usando índices para percorrer o vetor, e com 2 chamadas recursivas.

(b) **Quicksort via Ponteiros:** semelhante ao anterior, mas usando ponteiros, tanto nos parâmetros quando no percurso do vetor durante o algoritmo de particionamento.

Observação: como é baseada em ponteiros, não precisa fazer chamada explícita à função "escolher_pivo", mas deve fazer a mesma escolha de pivô que a versão baseada em índices. (Opcional: você pode inclusive escrever uma versão baseada em ponteiros da função de escolha do pivô.)

(c) **Quicksort Memória $O(\log n)$:** deve ser baseado em uma das 2 versões anteriores, mas transformando a chamada recursiva de cauda em um laço, e fazendo a chamada recursiva restante sempre sobre a metade de menor tamanho a ser ordenada.

(d) **Quicksort com Mediana das Medianas:** deve escolher o pivô por meio do Algoritmo de Seleção Linear BFPRT. Pode ser baseado em índices ou em ponteiros.

(e) **Quicksort com Pivô Aleatório:** deve ser baseado numa das 2 primeiras versões, mas escolhendo o pivô (pseudo)aleatoriamente, via função "rand" ou semelhante.

3. Requisitos: Segue abaixo uma especificação do restante do trabalho. Em caso de dúvida, por favor entre em contato com o professor rapidamente.

(a) Você deve apresentar um programa que crie um vetor de inteiros aleatórios e então o ordene com as várias versões do Quicksort solicitadas acima. O tamanho do vetor e o intervalo dos números aleatórios nele presentes devem ser solicitados ao usuário no início da execução do programa.

(b) Caso o professor forneça um gerador de instâncias específicas para o Quicksort, o programa deverá fornecer ao usuário a opção de utilizá-las (ao invés das instâncias aleatórias).

(c) Naturalmente, cada função de ordenação deve ser chamada sobre uma cópia do vetor aleatório gerado inicialmente (e não sobre uma versão já ordenada dele). O tempo utilizado pela função de ordenação deve ser exibido na tela. Após a função de ordenação acabar e seu tempo ser registrado, deve ser chamada uma função para checar se o vetor está de fato ordenado.

(d) Após a exibição dos dados de performance de cada versão do Quicksort, deve ser facultado ao usuário executar novo teste, com outro vetor.