IMD0029 - Estrutura de Dados Básicas 1 – 2018.1
Prof. Eiji Adachi M. Barbosa
Atividade Avaliativa Prática em Laboratório – Unidade 1

ANTES DE COMEÇAR, leia atentamente as seguintes instruções:

- Esta é uma atividade de caráter individual e sem consultas a pessoas ou material (impresso ou eletrônico).
- A atividade vale 5,0 pontos na 1ª unidade. O valor de cada questão é informado no seu enunciado.
- Celulares e outros dispositivos eletrônicos devem permanecer desligados durante toda a prova.
- Desvios éticos ou de honestidade levarão à anulação da atividade do candidato (nota igual a zero).
- Junto a este enunciado, você também recebeu uma estrutura de diretórios contendo um diretório para cada questão. Em cada um destes diretórios, já existe uma assinatura de função e uma função main com um pequeno teste executável. A solução da sua questão deverá seguir a assinatura da função já estabelecida. Ou seja, não mude esta assinatura. Se necessário, crie funções auxiliares com outras assinaturas, mas **não mude a assinatura da função original!**

Questão 1 (1,5 ponto): Considere um array A contendo N inteiros, com possíveis repetições, já ordenado em ordem crescente. Implemente uma função que recebe como entrada um inteiro K e retorna um índice i tal que A[i] == K, sendo A[i] o elemento igual a K que está mais a direita em A. Sua solução deverá ser **recursiva** e ter complexidade $\Theta(\lg(n))$.

Questão 2 (1,0 ponto): O problema Soma-3 é definido da seguinte forma: dado um array A e um inteiro X, buscam-se inteiros I, J, K tais que A[I] + A[J] + A[K] == X (admite-se a possibilidade de I, J e K serem iguais). Neste contexto, implemente uma função que apenas checa parte do problema Soma-3. Mais especificamente, implemente uma função que recebe um array de inteiro A e um inteiro X, e retorna verdadeiro caso existam inteiros I, J, K tais que A[I] + A[J] + A[K] == X; caso contrário, ela retorna falso. Sua solução deverá ter **obrigatoriamente** complexidade igual a $\Theta(n^2 * \log_2(n))$. Assuma nesta questão que o array de entrada sempre estará ordenado em ordem crescente.

Questão 3 (1,0 ponto): Considere V um array de n números inteiros distintos e que não está ordenado. Se i < j e V[i] > V[j] então o par (i,j) é chamado uma inversão de V. Por exemplo, o array V[4] = {10, 30, 20, 25} possui duas inversões: (I) (1,2) e (II) (1,3), ou seja, (I) 1 < 2 & V[1] > V[2], (II) 1 < 3 && V[1] > V[3]. Neste contexto, implemente uma função que recebe um array de entrada V e retorna um inteiro X com o número de inversões que existem em V.

Questão 4 (1,5 ponto): Implemente o algoritmo de ordenação QuickSort usando a estratégia de seleção de pivô "mediana de 3". Esta questão requer o arquivo de entrada input.txt que está disponível no SIGAA na aula "Algoritmos de ordenação" do dia 13/03/2018 (o arquivo aparece no SIGAA com o nome "Arquivo de entrada pros algoritmos de ordenação "; baixe-o e copie para dentro do diretório q4) .

ENTREGÁVEL

O entregável desta atividade deverá seguir a mesma estrutura de diretórios do código fonte que você recebeu com este enunciado, obviamente, contendo os arquivos fonte utilizados para construir sua solução nos diretórios de cada questão. Além disso, o diretório pai deverá ter o seu nome e matrícula, seguindo o padrão `<PRIMEIRO_NOME>_<SOBRENOME>-<MATRICULA>` (Dica: basta renomear o diretório /src com o padrão definido anteriormente). Por exemplo:

```
> JOAO_SILVA-200012345
   > q1
   > q2
   > q3
   > q4
```

Toda esta estrutura de diretórios, incluindo os arquivos fonte com sua solução, deverá ser compactada num arquivo .zip que também deverá seguir o padrão `<PRIMEIRO_NOME>_<SOBRENOME>-<MATRICULA>.` Este arquivo compactado deverá ser entregue via SIGAA até as **20:25**. **Este é um prazo fixo que não será estendido**, exceto em casos muito excepcionais (ex.: SIGAA fora do ar). Ou seja, entregas após este horário não serão aceitas. A atividade do SIGAA permite apenas um envio, portanto certifique-se de que está enviando a versão correta antes de anexar ao SIGAA.

CRITÉRIOS DE CORREÇÃO

Para a correção desta atividade, serão levados em consideração, dentre outros, os seguintes pontos:

- Obediência às regras definidas para as assinaturas de função e para o entregável (arquivo .zip), conforme especificado no enunciado desta atividade
- Existência de erros ou warnings de compilação do código fonte[1]
- Programas executam sem apresentar falhas e produzem os resultados esperados
- Soluções atendem critérios de complexidade, caso estabelecido no enunciado
- Apresentação e organização do código fonte entregue (identação, nome das variáveis, modularização do código em funções, etc)

Obs.: Para cada questão, já há uma função main com um pequeno teste executável. Este é um teste simples que **não garante** a corretude da sua implementação. Ou seja, se sua implementação passou no teste executável disponibilizado junto a este enunciado, isto não é garantia de que ela está totalmente correta. Para fins de correção, eu utilizarei outra bateria de testes mais completa, além de analisar manualmente o código produzido.

[1] Compile usando as flags `–Wall –pedantic –std=c++11`