IMD0029 - Estrutura de Dados Básicas 1 – Turma 03 – 2016.1 – Prova 01
Prof. Eiji Adachi M. Barbosa

Nome:_____

Matrícula: _____

ANTES DE COMEÇAR A PROVA, leia atentamente as seguintes instruções:

- Esta é uma prova escrita de caráter individual e sem consultas a pessoas ou material (impresso ou eletrônico).
- A prova vale 10,0 pontos e o valor de cada questão é informado no seu enunciado.
- Preze por respostas legíveis, bem organizadas e simples.
- As respostas devem ser fornecidas preferencialmente em caneta. Respostas fornecidas a lápis serão aceitas, mas eventuais questionamentos sobre a correção não serão aceitos.
- Celulares e outros dispositivos eletrônicos devem permanecer desligados durante toda a prova.
- Desvios éticos ou de honestidade levarão à anulação da prova do candidato (nota igual a zero).
- Considere que o primeiro elemento do vetor ocorre no índice 1 e o último elemento ocorre no índice N.

Questão 1: Um cliente chega até a sua equipe de desenvolvimento com o seguinte problema. Ele possui uma base de dados repleta de registros, em que cada registro é identificado por um valor numérico único (considere como um inteiro positivo). No entanto, devido a um *bug* no sistema que insere registros na base de dados, diversos registros foram inseridos mais de uma vez na base, gerando desnecessariamente dados duplicados. Além disso, o cliente não sabe a princípio quais os registros que estão duplicados. Para identificar os registros duplicados, o cliente está tentando primeiro ordenar os registros de acordo com o identificador numérico. Inicialmente, o cliente tentou ordenar os registros usando o algoritmo de ordenação por partição. No entanto, a versão do algoritmo de ordenação implementada pelo cliente é muito ineficiente. Depois de realizados alguns testes de desempenho, sua equipe identificou que a ineficiência do algoritmo de ordenação deve-se ao algoritmo de partição implementado pelo cliente, o qual não leva em conta que existem elementos repetidos. Analisando este caso, o chefe da sua equipe de desenvolvimento explica-lhe que é possível melhorar a eficiência do algoritmo de ordenação implementando uma nova versão do algoritmo de partição. Em particular, este algoritmo de partição leva em consideração a existência de elementos repetidos e, por isto, particiona o vetor de entrada em três sub-vetores. A figura abaixo mostra como o vetor de entrada deverá estar particionado após a execução deste algoritmo:

< pivô	== pivô	> pivô

Após a execução deste algoritmo de partição, o vetor possui **(i)** um sub-vetor que só contém elementos menores que o pivô, **(ii)** um sub-vetor que só contém elementos iguais ao pivô e **(iii)** um sub-vetor que só contém elementos maiores do que o pivô. Neste contexto:

a) (2,5 pontos) Projete o algoritmo de partição sugerido pelo chefe da sua equipe de desenvolvimento. Este algoritmo deve retorna um par (i,j), em que 'i' indica o índice do primeiro elemento na região em que os elementos são iguais ao pivô e 'j' indica o índice de último elemento desta região. Além disso, você deve fazer o projeto do seu algoritmo seguindo a seguinte estrutura básica definida pelo seu chefe (Obs.: a inicialização da variável pivô não pode ser modificada).

```
Particionar( v[1..N], esquerda, direita ) :
  pivô = v[esquerda]
  // Aqui o restante do algoritmo
  retorne (i,j);
FIM
```

b) (1,0 ponto) De que forma esta versão do algoritmo de partição pode melhorar a eficiência do algoritmo de ordenação por partição quando o vetor de entrada possui muitos elementos repetidos? Justifique sua resposta.

c) (1,0 ponto) Dado um vetor de inteiros de entrada v[1...N], projete um algoritmo que retorna verdadeiro se este vetor está particionado conforme a figura e as condições **(i)**, **(ii)** e **(iii)** acima, e falso caso contrário. Em particular, seu algoritmo deve seguir a seguinte assinatura:

```
Está_Particionado( v[1...N], i, j ) :
  // Aqui o restante do algoritmo
FIM
```

Na assinatura do algoritmo acima, os parâmetros 'i' e 'j' indicam, respectivamente, o índice do primeiro e do último elemento da região que contém apenas elementos iguais ao pivô.

Questão 2: Após resolver o problema da ordenação dos registros repetidos, o cliente recorre à sua equipe de desenvolvimento mais uma vez buscando ajuda para identificar quais registros estão duplicados na base. Desta

forma, dado um vetor de inteiros v[1...N] representando os identificadores numéricos dos registros e uma chave de busca X, o algoritmo pedido pelo cliente deve retornar um par (i,j) em que 'i' é o índice do primeiro elemento igual a X e 'j' o índice do último elemento igual a X. Quando o valor X não é encontrado no vetor, é retornado um par igual a (-1,-1). Neste contexto:

a) (1,0 ponto) Projete uma solução com complexidade $\theta(n)$ para este algoritmo de busca pedido pelo cliente.
b) (2,0 pontos) Projete uma solução com complexidade $\theta(\log_2(n))$ para este algoritmo de busca pedido pelo cliente.

Questão 3: (1,5 ponto) Um algoritmo de ordenação é dito estável se não altera a posição relativa dos elementos que têm um mesmo valor. Em outras palavras, se dois elementos possuem valores iguais e a ordem relativa em que eles aparecem na sequência de dados inicial é mantida após a ordenação, então este algoritmo de ordenação é estável. Considere como exemplo o seguinte vetor v[1..5] = {1', 2', 3´, 1, 2, 3}. Se após a execução de um algoritmo de ordenação o vetor encontrar-se da seguinte forma: v = {1', 1, 2', 2, 3', 3}, então este algoritmo de ordenação é dito estável. Perceba que os números repetidos não alteraram as posições relativas que tinham antes da ordenação (elementos repetidos com aspas simples apareciam primeiro do que os repetidos sem aspas antes da ordenação e continuaram aparecendo primeiro após a ordenação). Neste contexto, considere o seguinte algoritmo de intercalação:

```
Intercalar( v[1...n], inicio1, inicio2, fim2 ) :
     DECLARE tmp[1...n]
     fim1 = inicio2-1, i = inicio1, j=inicio2, k=1
     ENQUANTO i ≤ fim1 E j ≤ fim2 FAÇA:
           SE v[i] < v[j] ENTÃO:
                 tmp[k] = v[i],i = i+1
           SENÃO:
                 tmp[k] = v[j], j = j+1
           FIM_SE
           k = k+1
     FIM_ENQUANTO
     ENQUANTO i ≤ fim1 FAÇA:
           tmp[k] = v[i], i = i+1, k=k+1
     FIM_ENQUANTO
     ENQUANTO j ≤ fim2 FAÇA:
            tmp[k] = v[j], j = j+1, k=k+1
     FIM_ENQUANTO
     COPIE tmp EM v
FIM
```

Suponha que você vai usar o algoritmo acima como parte do seu algoritmo de ordenação por intercalação. O seu algoritmo de ordenação será estável? Se sim, o que garante sua estabilidade? Se não, o que precisa ser mudado para garantir sua estabilidade?

Questão 4: (1,0 ponto) Considerando um vetor V de tamanho N, defina para cada um dos algoritmos abaixo sua ordem de complexidade em termos do tamanho do vetor:

Algoritmo	Pior caso	Melhor caso
Ordenar_por_seleção(V[1...N])		
Ordenar_por_inserção(V[1...N])		
Ordenar_por_flutuação(V[1...N])		
Ordenar_por_intercalação(V[1...N])		
Ordenar_por_partição[1](V[1...N])		

[1]Para o algoritmo de ordenação por partição, considere que é usada a estratégia mais simples de seleção do pivô, isto é, o pivô sempre será o último elemento do vetor (ou, equivalentemente, o primeiro elemento do vetor).