Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação

Disciplina: Estrutura de Dados

AP1 - 2018/2

Nome –

Assinatura –

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

---

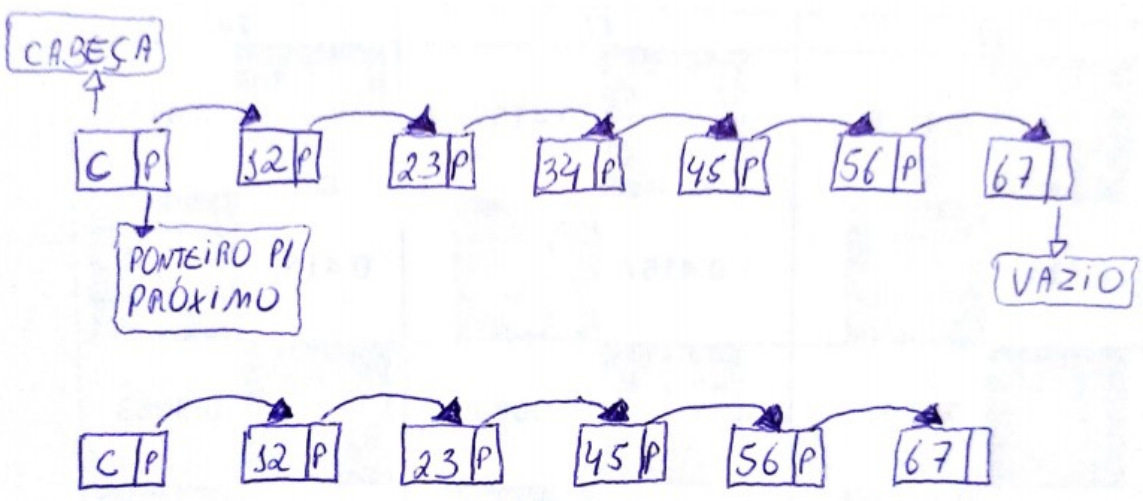
1) (2,5) Dado um vetor *V* com *n* elementos (pode haver elementos repetidos), escreva um algoritmo que determine o elemento de *V* que ocorre o maior número de vezes. Havendo mais de um elemento com esta propriedade, o algoritmo deve determinar todos eles. Exemplo: se *V* é formado pelos elementos **7, 3, 6, 7, 1, 6, 5, 9, 6, 8, 8, 7**, a resposta deve ser: **6** e **7** (pois ocorrem 3 vezes cada). Qual é a complexidade do seu algoritmo?

R: Uma das soluções possíveis para este problema é apresentada abaixo. A complexidade desta solução é de $O(N^2)$.

```python
def encontraRepeticoes(lista, n):

    maior_repeticao = 0 #Armazena a maior quantidade de repetições encontrada
    elementos_maior_repeticao = [] #Armazena os elementos que apresentam a
                                   #maior quantidade de repetições

    for i in range(n):
        elemento_atual = lista[i] #Fixa em um elementoelemento
        qnt_repeticoes = 0

        #Percorre todos os outos elementos contabilizando a quantidade de repetições
        for j in range (i,n):
            if elemento_atual == lista[j]:
                qnt_repeticoes +=1

        #Se este elemento tem maior qnt de repetições atualiza a variavel maior_repeticao
        # A lista contendo os elementos que tem maior repetição é limpa \
        # e atualizada como o novo elemento
        if qnt_repeticoes > maior_repeticao:
            maior_repeticao = qnt_repeticoes #
            elementos_maior_repeticao = [elemento_atual]

        #Se a qnt de repetições é igual atualiza a lista inserindo no final mais um elemento
        elif qnt_repeticoes == maior_repeticao:
            elementos_maior_repeticao.append(elemento_atual)

    print(elementos_maior_repeticao)

encontraRepeticoes([7, 3, 6, 7, 1, 6, 5, 9, 6, 8, 8, 7,8], 13)
```

2) (2,5) Considere uma lista simplesmente encadeada ordenada contendo os nós com os valores: **12, 23, 34, 45, 56, 67**. Desenhe esta lista, representando todos os ponteiros. Redesenhe a lista após a remoção do nó **34**, mostrando as alterações feitas nos ponteiros.
R: O encadeamento da lista pode ser observado na imagem abaixo. Observe que:

- No topo da figura é apresentada a lista com a sequência original, antes da remoção. Note que a lista apresenta um nó cabeça, que aponta para o primeiro elemento (primeiro nó) e que o último elemento aponta para nulo (vazio).
- Já na partir inferior da figura é retratada a mesma lista, porém, após a remoção do nó 34. Observe que o ponteiro do nó 23, que antes apontava para 34, agora aponta para o nó cujo o nó 34 apontava.



3) (2,5) Aplique o método de ordenação por bolhas (Bubblesort) ao vetor abaixo, de modo que ele fique ordenado decrescentemente, isto é, o maior valor fica à esquerda e o menor valor à direita. Mostre todas as trocas de posição entre elementos.

32 33 27 31 29 26 25 30 28

R: O Algoritmo de ordenação bolha consiste em percorrer um vetor e, em cada passagem, empurrar o maior elemento (caso esteja ordenando crescentemente) ou menor elemento (caso esteja ordenando decrescentemente) para o final da lista. A tabela abaixo apresenta as trocas realizadas pelo algoritmo para ordenar de forma decrescente a sequência apresentada no exercício.

32	33	27	31	29	26	25	30	28	Sequência inicial
33	32	27	31	29	26	25	30	28	Trocou 32 e 33
33	32	31	27	29	26	25	30	28	Trocou 27 e 31
33	32	31	29	27	26	25	30	28	Trocou 27 e 29
33	32	31	29	27	26	30	25	28	Trocou 25 e 30
33	32	31	29	27	26	30	28	25	Trocou 25 e 28
33	32	31	29	27	30	26	28	25	Trocou 26 e 30
33	32	31	29	27	30	28	26	25	Trocou 26 e 28
33	32	31	29	30	27	28	26	25	Trocou 27 e 39
33	32	31	29	30	28	27	26	25	Trocou 27 e 28
33	32	31	30	29	28	27	26	25	Trocou 29 e 30 - Sequência final

4) (2,5) Explique como ler uma sequência de números positivos dados (que termina com o marcador "0") e depois imprimi-la em ordem inversa, sem o uso de um vetor. Exemplo: se a sequência lida é: **56 34 23 12 78 45 0**, então a sequência impressa deve ser: **45 78 12 23 34 56** (não é preciso imprimir o marcador "0").
Observações:

(a) não é necessário escrever o código algorítmico, basta explicar o processo com palavras
(b) sugestão: use uma pilha para resolver o problema

R: Cada elemento da sequência pode ser lido e armazenado em uma pilha. Considerando que nesta estrutura as entradas e saídas de dados se dão exclusivamente pelo topo podemos:

- Ler elemento a elemento da sequência, e empilhá-los, isto é, inseri-los na pilha:
  - Ler o elemento;
  - Atualiza a variável que aponta para o topo da pilha;
  - Insere o elemento na pilha (empilha);
  - Repetir enquanto o elemento lido for diferente de "0";
    - Desta forma o marcador não será inserido na pilha
- Inicia-se o processo de impressão e desempilhamento, composto pelas atividades:
  - Imprimir o topo da pilha;
  - Remover o elemento do topo (desempilha);
  - Atualizar a variável que aponta para o topo;
  - Repetir até que a pilha fique vazia (topo = 0).