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?

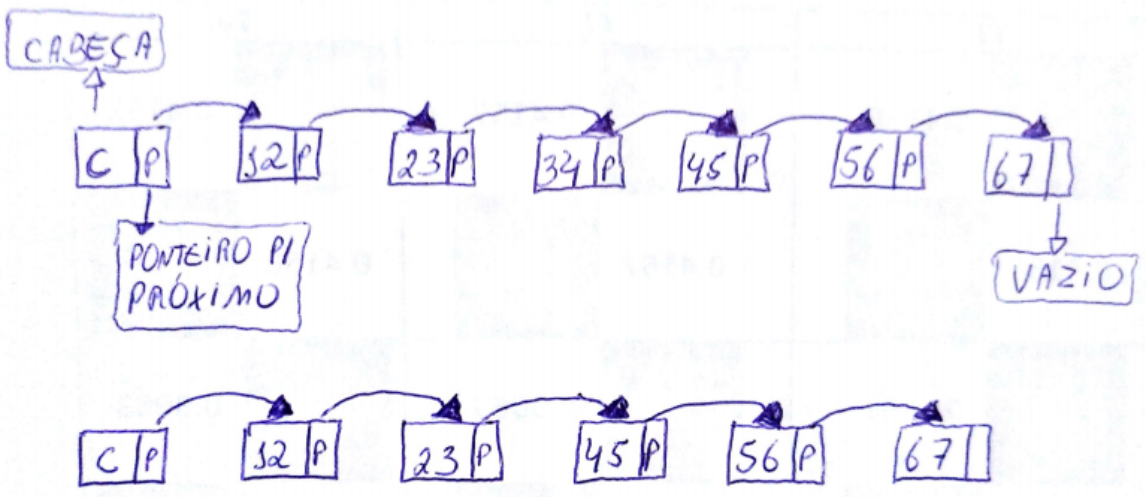
R: Uma das soluções possíveis para este problema é apresentada abaixo. A complexidade desta solução é de $O(N^2)$.

```
Algoritmo 1: encontraRepetições (lista, n):

    Entrada: Lista (vetor) com as sequências
    Saída: Elementos com maior repetição

    maior_repeticao = 0; #Armazena a maior quantidade de repeticoes encontrada
    elementos_maior_repeticao = []; #Armazena os elementos que possuem maior qntd de repetições

    para i de 0 até n:
        elemento_atual = lista[i]
        qnt_repeticoes = 0

       # Percorre todos os demais elementos contabilizando as repetições
        para cada j de i até n:
            se elemento_atual == lista[j]~:
                qnt_repeticoes = qnt_repeticoes + 1

        se qnt_repeticoes > maior_repeticao:
            maior_repeticao = qnt_repeticoes
            elementos_maior_repeticao = [] #A lista de elementos é apagada
            elemento_atual[0] = elemento_atual #A lista é atualizada com o novo elemento

        senão
           se qnt_repeticoes == maior_repeticao:
               # O elemento atual é inserido na lista
               insere_na_lista(elementos_maior_repeticao, elemento_atual)

    retorna elementos_maior_repeticao
```

OBS: * A função *insere_na_lista(lista,elemento)* abstrai o processo de inserir um elemento em uma dada lista/vetor. Como não foi passada nenhuma posição, a função insere no final do vetor.

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 se baseia na ideia de que os elementos mais leves "sobem" e os elementos mais pesados "descem e vão para o fundo". Pode-se comparar a subida de um elemento como uma bolha, que tenta chegar a superfície.

1	2	3	4	5	6	7	8	9	
32	33	27	31	29	26	25	30	28	Sequência inicial
32*	33	27	31	29	26	25	30	28	Bolha: V [1]
32	33*	27	31	29	26	25	30	28	Bolha: V [2]
33*	32	27	31	29	26	25	30	28	troca 33 <-> 32
33	32	27*	31	29	26	25	30	28	Bolha: V [3]
33	32	27	31*	29	26	25	30	28	Bolha: V [4]
33	32	31*	27	29	26	25	30	28	troca 31 <-> 27
33	32	31	27	29*	26	25	30	28	Bolha: V [5]
33	32	31	29*	27	26	25	30	28	troca 29 <-> 27
33	32	31	29	27	26*	25	30	28	Bolha: V [6]
33	32	31	29	27	26	25*	30	28	Bolha: V [7]
33	32	31	29	27	26	25	30*	28	Bolha: V [8]
33	32	31	29	27	26	30*	25	28	troca 30 <-> 25
33	32	31	29	27	30*	26	25	28	troca 30 <-> 26
33	32	31	29	30*	27	26	25	28	troca 30 <-> 27
33	32	31	30*	29	27	26	25	28	troca 30 <-> 29
33	32	31	30	29	27	26	25	28*	Bolha: V [9]
33	32	31	30	29	27	26	28*	25	troca 28 <-> 25
33	32	31	30	29	27	28*	26	25	troca 28 <-> 26
33	32	31	30	29	28*	27	26	25	troca 28 <-> 27

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).