Curso de Tecnologia em Sistemas de Computação
Disciplina: Estrutura de Dados e Algoritmos
AP1 - Primeiro Semestre de 2018

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. Defina:

   (a) (1,0) Complexidade de pior caso de um algoritmo.

   Resposta: Seja A um algoritmo e $E = \{E_1, E_2, \cdots, E_n\}$ o conjunto de todas as entradas possíveis de A. Dada a entrada E_i, seja t_i o número de passos efetuados por A, para $1 \leq i \leq n$. Podemos definir a *complexidade de pior caso* como $\max_{E_i \in E}\{t_i\}$.

   (b) (1,0) Algoritmo ótimo.

   Resposta: Seja g um limite inferior para um problema P. Um algoritmo ótimo A que resolve P é tal que sua complexidade é dada por $f = O(g)$. Dessa forma, o algoritmo A possui complexidade de pior caso $\Omega(g)$ e $O(g)$. Em outras palavras, um algoritmo é ótimo se sua complexidade de pior caso é dada pelo limite inferior para o problema.

2. Considere os algoritmos **ordenação por seleção** e **ordenação pelo método da bolha**, aplicados a uma lista com n elementos em ordem inversa de ordenação (Ex: 8 7 6 5 4 3 2 1).

   (a) (1,0) Qual dos dois algoritmos efetua mais **comparações** entre elementos? Justifique sua resposta.

   Resposta: Os dois algoritmos efetuam $\theta(n^2)$ comparações. No algoritmo por seleção é feita a busca pelo menor elemento do vetor a ser ordenado em cada passo. Logo todo o vetor corrente é percorrido comparando o menor elemento corrente com cada outra posição. Dessa forma o número de comparações independe do vetor de entrada.

   No algoritmo pelo método da bolha é feita a comparação do elemento apontado como bolha com o elemento anterior, a cada passo do algoritmo. Como o vetor de entrada está em ordem inversa, a bolha corrente é menor que todos os elementos em posições menores que aquela ocupada pela bolha. Logo é necessário percorrer todo o vetor, da posição ocupada pela bolha, até o início do vetor, resultando em $\theta(n^2)$ comparações.

   (b) (1,0) Qual dos dois algoritmos efetua mais **trocas** entre elementos? Justifique sua resposta.

Resposta: A ORDENAÇÃO POR SELEÇÃO executa $\lfloor \frac{n}{2} \rfloor$ trocas, uma vez que os elementos trocados ocupam suas posições finais no vetor. Isso pode ser verificado para a primeira troca com n elementos, onde o primeiro e último elementos trocam de lugar. Como o primeiro é o menor elemento do vetor e o n-ésimo o maior, os mesmos não mudam de posição novamente. Podemos aplicar o mesmo raciocínio nos demais passos.

Por outro lado, a ORDENAÇÃO PELO MÉTODO DA BOLHA executa um numero quadrático de trocas, já que em cada iteração a bolha é trocada com todos os elementos que a precedem até o primeiro elemento menor que a mesma. Ou seja, executa-se $(n-1)+(n-2)+\cdots+1 = \frac{n(n-1)}{2}$ trocas.

Portanto a ORDENAÇÃO POR SELEÇÃO executa menos trocas que a ORDENAÇÃO PELO MÉTODO DA BOLHA.

3. (2,0) Escreva um algoritmo que execute a seguinte tarefa: Dado um vetor não ordenado com n elementos ($n \geq 1$), encontre **o maior e o segundo maior elementos** deste vetor. Seu algoritmo deverá percorrer o vetor uma única vez.

   Resposta: O Algoritmo 1 retorna o segundo maior (SM) e o maior (M)

valores, respectivamente.

Algoritmo 1: $Maior\_e\_SegundoMaior(V)$.

Entrada: Vetor V não ordenado composto por $n \geq 1$ valores distintos.

Saída: Os valores do maior, M, e do segundo maior, SM, elementos de V.

```
1  se n = 1 então
2      retorna V[1];
3  senão
4      se V[1] < V[2] então
5          SM ← V[1];
6          M ← V[2];
7      senão
8          SM ← V[2];
9          M ← V[1];
10     para i ← 3, ..., n faça
11         se V[i] > SM então
12             se V[i] > M então
13                 SM ← M;
14                 M ← V[i];
15             senão
16                 SM ← V[i];
17     retorna SM e M;
```

4. Os números 1, 2, 3, 4, 5 são inseridos em uma pilha nesta ordem. Porém, estas cinco operações de inserção são intercaladas com operações de remoção, e a cada remoção é impresso o número desempilhado. Considere os exemplos abaixo, onde I representa uma operação de inserção, e R uma operação de remoção.

   Exemplo 1: Se a sequência de operações na pilha for I R I R I I I R R R, a sequência de números impressos será 1 2 5 4 3.

   Exemplo 2: Se a sequência de operações na pilha for I I I I I R R R R R, a sequência de números impressos será 5 4 3 2 1.

(a) (1,0) Escreva a sequência de números impressos quando a sequência de operações na pilha é I I R R I I R R I R.

Resposta: $2 \cdot 1 \cdot 4 \cdot 3 \cdot 5$.

(b) (1,0) Determine a sequência de operações na pilha que resulta na seguinte sequência de impressão: 3 5 4 2 1.

Resposta: $I \quad I \quad I \quad R \quad I \quad I \quad R \quad R \quad R \quad R$.

5. (2,0) Considere uma lista simplesmente encadeada L com n nós, que armazenam números inteiros. Elabore um algoritmo que crie uma nova lista L' contendo somente nós com os números pares que ocorrem em L. Os números devem aparecer em L' na mesma ordem em que aparecem em L. Por exemplo, se L contiver os números 1, 8, 4, 5, 7, 8, 6, 3, nesta ordem, então L' conterá os números 8, 4, 8, 6, nesta ordem. Qual a complexidade do seu algoritmo?

Resposta: A complexidade do algoritmo a seguir é $\Theta(n)$, pois percorre a lista L apenas uma vez, e para cada nó de L executa um número constante de passos.

```
pt := L          % considerando que L não tem nó cabeça
L' := λ
ultimo := λ
enquanto pt ≠ λ faça
    se pt↑.info mod 2 = 0 então
        ocupar(novo)
        novo↑.info := pt↑.info
        novo↑.prox := λ
        se ultimo ≠ λ então          % L' já contém algum nó
            ultimo↑.prox := novo
        senão
            L' := novo
        ultimo := novo
    pt := pt↑.prox
```