Curso de Tecnologia em Sistemas de Computação
Disciplina: Estrutura de Dados e Algoritmos
AP1 - Segundo Semestre de 2013

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: Sejam A um algoritmo, $E = \{E_1, \cdots, E_n\}$ o conjunto de todas as entradas possíveis de A e t_i o número de passos efetuados por A, quando a entrada for E_i. A complexidade de pior caso de A é definida por $max_{E_i \in E} \{t_i \mid E_i \in E\}$.

   (b) (1,0) Algoritmo ótimo.

   Resposta: Sendo ℓ o limite inferior do problema, um algoritmo é ótimo se sua complexidade é $O(\ell)$.

   (c) (1,0) Árvore estritamente binária.

   Resposta: Uma árvore é estritamente binária se cada nó possui 0 ou 2 filhos.

2. Considere uma lista sequencial ordenada L dada como um vetor de n posições. Forneça as complexidades (em notação O) dos seguintes algoritmos:

   (a) (0,5) Busca sequencial de um elemento x em L (forneça a complexidade de pior caso).

   Resposta: $O(n)$

   (b) (0,5) Busca binária de um elemento x em L (forneça a complexidade de pior caso).

   Resposta: $O(\log n)$

   (c) (0,5) Remoção de um elemento x que se encontra na posição $\lfloor \frac{n}{2} \rfloor$ de L, isto é, $x = L[\ \lfloor \frac{n}{2} \rfloor\ ]$

   Resposta: $O(n)$

   (d) (0,5) Inserção de um elemento x em L tal que $x > L[i]$ para todo $i = 1, \ldots, n$.

   Resposta: $O(1)$

3. Dado um vetor contendo os números $3, 8, 11, 0, 5, 9$, pede-se:

   (a) (1,0) Desenhe todas as trocas de elementos que o *método de ordenação por seleção* efetua. **Exemplo:** se as trocas fossem "3 por 8", "5 por 9", "0 por 3" etc., você desenharia a seguinte sequência de vetores:

$$8, 3, 11, 0, 5, 9$$

$$8, 3, 11, 0, 9, 5$$

$$8, 0, 11, 3, 9, 5$$

etc.

Resposta:

$$3, 8, 11, 0, 5, 9$$

$$0, 8, 11, 3, 5, 9$$

$$0, 3, 11, 8, 5, 9$$

$$0, 3, 5, 8, 11, 9$$

$$0, 3, 5, 8, 9, 11$$

(b) (1,0) Desenhe todas as trocas de elementos que o *método de ordenação da bolha* efetua. Utilize na resposta o mesmo sistema do item anterior.

Resposta:

$$3, 8, 11, 0, 5, 9$$

$$3, 8, 0, 11, 5, 9$$

$$3, 0, 8, 11, 5, 9$$

$$0, 3, 8, 11, 5, 9$$

$$0, 3, 8, 5, 11, 9$$

$$0, 3, 5, 8, 11, 9$$

$$0, 3, 5, 8, 9, 11$$

4. (1,0) Considere uma pilha P contendo 5 posições de 1 a 5. A variável *topo* marca a posição do topo da pilha. No início, a fila P encontra-se vazia, e a variável *topo* vale zero. Usamos a notação R para denotar a operação de remoção de um elemento da pilha P, e a notação $I(X)$ para denotar a operação de inserção de um elemento X na pilha P.

Considere a seguinte sequência de operações em P:

$$I(A), I(B), I(C), R, I(D), R, I(E), I(G), I(H), R, R, R, R, I(J)$$

Desenhe como fica a fila P após a sequência de operações acima, e forneça o valor final da variável *topo*. Use um traço (–) para denotar as posições vazias. Como um exemplo de configuração, poderíamos ter como resposta: $P = [\ C\ D\ H\ -\ -\ ]$, onde *topo* neste caso vale 3.

Resposta: $P = [\ A\ J\ -\ -\ -\ ]$ com $topo = 2$

5. (2,0) Dada uma lista simplesmente encadeada L com n nós, descrever um algoritmo para inverter a direção do encadeamento de L. Isto é, o algoritmo deve transformar L em uma outra lista, contendo exatamente os mesmos nós do que L, porém na ordem invertida. Pede-se:

   (a) Descrever a estratégia geral do algoritmo, em palavras.

   Resposta: O algoritmo percorre a lista uma única vez, com 3 ponteiros que apontam para elementos consecutivos da lista original. Inicialmente, como o primeiro ponteiro aponta para o primeiro elemento válido da lista (descartando o nó cabeça), que passará a ser o último, seu campo *prox* recebe λ. A cada iteração do algoritmo, fazemos com que um nó da lista aponte para o seu anterior, e os 3 ponteiros avançam um elemento na lista. Ao final, o campo *prox* de *ptlista* aponta para o novo primeiro nó da lista, que anteriormente era o último.

   (b) Descrever uma implementação do algoritmo, supondo que a lista L está armazenada com a utilização de ponteiros.

   Resposta:

$pont1 := ptlista \uparrow .prox$

$pont2 := pont1 \uparrow .prox$

$pont1 \uparrow .prox := \lambda$

se $pont2 \neq \lambda$ então

    $pont3 := pont2 \uparrow .prox$

    enquanto $pont3 \neq \lambda$ faça

        $pont2 \uparrow .prox := pont1$

        $pont1 := pont2$

        $pont2 := pont3$

        $pont3 := pont2 \uparrow .prox$

    $pont2 \uparrow .prox := pont1$

    $ptlista \uparrow .prox := pont2$

(c) Determinar e justificar a complexidade do algoritmo.

Resposta: A complexidade do algoritmo é $\theta(n)$, uma vez que percorre a lista L exatamente uma vez, e executa um número constante de passos para cada elemento da lista.