Curso de Tecnologia em Sistemas de Computação
Disciplina: Estrutura de Dados e Algoritmos
Gabarito da AP1 - Segundo Semestre de 2010

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. PRIMEIRA QUESTÃO: (VALOR 2,0)

   Para cada dupla de afirmativas abaixo, assinale uma das seguintes opções, numeradas de (i) a (iv):

   Opção (i): Ambas as afirmativas estão corretas.

   Opção (ii): A afirmativa (a) está correta, mas a afirmativa (b) não está.

   Opção (iii): A afirmativa (b) está correta, mas a afirmativa (a) não está.

   Opção (iv): Ambas as afirmativas estão incorretas.

   Primeira dupla:

   (a) $n^3 + logn + 5 = O(n^3)$

   (b) $n^5 - 200 = O(n^3)$

   RESPOSTA: Opção (ii).

   Segunda dupla:

   (a) Em uma pilha, se um nó x é inserido após o nó y, então a retirada de y só pode ocorrer após a retirada de x.

   (b) Suponha que os nós $x_1, \ldots, x_n$ tenham sido inseridos em uma pilha e também em uma fila, exatamente na mesma ordem, de tal forma que não tenham ocorrido remoções entre as inserções. Então a sequência de inserções coincide com a de remoções, tanto na pilha quanto na fila, se e somente se $n = 1$.

   RESPOSTA: Opção (i).

   Terceira Dupla:

   (a) Duas árvores que possuem o mesmo número de nós e o mesmo número de folhas são sempre isomorfas.

   (b) Sejam x, y dois nós de uma árvore, tais que o $\textit{nível}(x) < \textit{nível}(y)$ e $\textit{altura}(x) > \textit{altura}(y)$. Então x é ancestral de y.

   RESPOSTA: Opção (iv).

Quarta Dupla:

(a) Sejam L_1 uma lista não ordenada e L_2 uma lista ordenada, L_1, L_2 distintas, cada uma delas possuindo n nós, e nas quais é realizada uma operação de busca. Se ambas as buscas efetuarem $m < n$ iterações, pode-se concluir que a chave procurada foi encontrada, nos dois casos.

(b) Em uma lista simplesmente encadeada não ordenada é possivel realizar as operações de remoção em complexidade $O(1)$, desde que seja conhecido um ponteiro que informe a localização do nó a ser removido.

RESPOSTA: Opção (iv).

2. SEGUNDA QUESTÃO: (VALOR 2,0)

Responder CORRETO ou INCORRETO, para cada uma das afirmativas abaixo, justificando a sua resposta. As respostas somente serão consideradas se acompanhadas de justificativas.

(i): Suponha que dois algoritmos A_1 e A_2, que resolvem o mesmo problema, sejam implementados e executados em um computador para uma mesma entrada. Então se A_1 for um algoritmo ótimo e A_2 não o for, concluímos que o tempo de execução de A_2 não pode ser menor do que o de A_1.

CORRETO ( )　　　　INCORRETO (X)

O fato de A_1 ser ótimo garante apenas que sua complexidade de pior caso é a mínima possível, ou seja, nenhum outro algoritmo para o mesmo problema tem complexidade de pior caso menor. Mas outros algoritmos podem ter complexidade de caso médio ou melhor caso menores do que A_1. Logo, pode haver entradas em que A_2 tenha um tempo de execução menor do que A_1.

(ii): Se um certo algoritmo A que resolve um certo problema P possui complexidade de melhor caso igual a f então podemos concluir que f é um limite inferior para P.

CORRETO ( )　　　　INCORRETO (X)

O limite inferior de um problema P é uma função f que limita inferiormente a complexidade de pior caso de qualquer algoritmo que resolva P. Nada podemos afirmar quanto ao limite inferior de problema a partir da complexidade de melhor caso de um algoritmo.

3. TERCEIRA QUESTÃO (VALOR 3,0)

   Responder as seguintes questões relativas à busca binária:

   (a) Explicar o funcionamento do algoritmo de busca binária aplicado a uma lista sequencial L composta por n elementos.

   Resposta: Dada uma lista sequencial ordenada L com n elementos, o algoritmo inicialmente pesquisa o nó que se encontra no meio de L. Se a comparação com o elemento buscado x não é positiva, metade da tabela é abandonada na busca, esta continuando recursivamente na metade inferior (caso x seja menor que o elemento do meio) ou superior (caso x seja maior). A busca segue até que x seja encontrado ou não haja mais elementos para comparação.

   (b) Porque L deve estar ordenada ?

   Resposta: Somente com a lista ordenada podemos garantir que metade da lista considerada pode ser descartada a cada comparação.

   (c) O algoritmo funcionaria se L fosse uma lista encadeada ? Por qual motivo ?

   Resposta: Não. Em uma lista encadeada, não há como acessar os elementos em tempo constante (pois não são indexados). Logo, sempre precisaríamos percorrer metade da lista considerada para acessarmos o elemento do meio.

   (d) Determinar e justificar as complexidades de pior caso e melhor caso, tanto na busca com sucesso, quanto na busca sem sucesso.

   Resposta:

   Busca com sucesso: No pior caso é $O(\log n)$, o que ocorre quando só encontramos o elemento buscado na última comparação, e a lista considerada tem apenas um elemento. No melhor caso é $O(1)$, quando o elemento é encontrado na primeira comparação. Busca sem sucesso: Melhor e pior casos são $O(\log n)$, pois sempre são feitas $O(\log n)$ comparações até reduzir a lista a apenas um elemento, e concluir que este não é o elemento buscado.

4. QUARTA QUESTÃO: (VALOR 3,0)

   Escrever dois algoritmos, o primeiro para efetuar inserção de uma nova chave x em uma lista sequencial ordenada, e o segundo para efetuar a remoção de um elemento y em uma lista sequencial não ordenada. Se for necessário efetuar alguma busca prévia, esta deve ser tambem detalhada no algoritmo. Supor que os elementos da lista sejam $s_1, \ldots s_n$. Determinar tambem a complexidade de cada algoritmo.

   Resposta:

   Inserção em uma lista sequencial ordenada:

   se $n = M$ **então**
       "overflow"
   senão
       $V[n+1] := x$
       $i := 1$
       enquanto $V[i] < x$ **faça**
           $i := i + 1$
       se $i < n + 1$ **e** $V[i] \neq x$ **então**
           para $j := n \cdots i$ **passo** -1
               $V[j+1] := V[j]$
           $V[i] := x$
           $n := n + 1$
       senão
           se $i = n + 1$ **então**
               $n := n + 1$
           senão
               "Elemento já existe na tabela."

   Complexidade: $\Theta(n)$.

Remoção de uma lista sequencial não ordenada:

se $n = 0$ **então**
    "underflow"
senão
    $i := 1$
    enquanto $V[i] \neq x$ **e** $i \leq n$ **faça**
        $i := i + 1$
    se $i > n$ **então**
        "Elemento não se encontra na tabela."
    senão
        $V[i] := V[n]$
        $n := n - 1$

Complexidade: $O(n)$.