Curso de Tecnologia em Sistemas de Computação

Disciplina: Estrutura de Dados e Algoritmos

AP1 - Primeiro Semestre de 2011

GABARITO

1. (Até 1,5) (0,5 pontos cada) Dê as definições de:

   - Complexidade de pior caso de um algoritmo
   - Limite inferior de um problema
   - Algoritmo ótimo

   SOLUÇÃO

   - A complexidade de pior caso é o número de passos que o algoritmo executa para computar a entrada mais desfavorável, isto é, o máximo do número de passos executados, dentre todas as entradas possíveis.
   - O limite inferior de um problema P é uma função ℓ, tal que, a complexidade de pior caso de qualquer algoritmo que resolva P é pelo menos ℓ.
   - Um algoritmo para resolver um problema P é ótimo quando a sua complexidade é igual ao limite inferior de P.

2. (Até 2,0) Descreva o algorimo recursivo de busca binária, onde a entrada é uma lista ordenada L com $n \geq 1$ elementos. Mostre exemplos de execução de pior caso e de melhor caso para $n = 8$, desenhando todos os elementos acessados ao longo da execução do algoritmo.

   SOLUÇÃO

   Seja $L(1), \ldots, L(n)$ a lista ordenada. Representamos por x a chave a ser procurada. O procedimento recursivo BUSCA, abaixo, resolve o problema. O procedimento retorna o resultado da busca. Se a chave for encontrada em L, o valor retornado fornece a posição de x em L, caso contrário este valor é zero.

   proc BUSCA(i, j)
     se $i \leq j$ então
       $m := \lfloor (i + j)/2 \rfloor$
       se $x = L(m)$ então retornar m
       senão se $x < L(m)$ então BUSCA($i, m - 1$) senão BUSCA($m + 1, j$)
     senão retornar 0

   EXEMPLO: Seja a lista seguinte:
   2, 5, 9, 14, 17, 23, 25, 39

Melhor caso: $x = 14$
Elementos acessados: $L(4) = 14$

Pior casi: $x = 39$
Elementos acessados: $L(4) = 14$, $L(6) = 23$, $L(7) = 25$, $L(8) = 39$

3. (Até 2,0) Dada uma lista L, simplesmente encadeada, deseja-se construir uma lista L', também simplesmente encadeada, que seja o reverso de L. Isto é, a lista L' contém os mesmos elementos que L, porém em ordem inversa. Explicar, por meio de palavras, a estratégia empregada para a construção de L' e, em seguida, descrever o processo em uma linguagem algorítmica. Supor que os nós de L sejam $X_1, \ldots, X_n$, sendo que o nó X_i aponta para X_{i+1}, $1 \leq i < n$.

SOLUÇÃO

Assumimos que cada nó da lista L possui dois campos, $info_L$ e $pont_L$, onde o primeiro corresponde à informação propriamente dita e o segundo é um ponteiro para o elemento seguinte na lista. A estratégia consiste em percorrer a lista, do início até o final. Ao se atingir um nó X_i no percurso, deve-se criar o nó X'_i, correspondente a X_i na lista reversa L', o qual deverá apontar para o nó X'_{i-1}, de L'. Este último é o correspondente ao nó X_{i-1} de L', que já deve ter sido criado. O processo assim se desenvolve até a lista L ter sido esgotada.

Para a implementação, assumimos que o último elemento da lista é sinalizado com valor de ponteiro igual a λ. O primeiro nó da lista L é indicado pelo parâmetro $ptlista_L$, enquanto que o da lista L' por $ptlista_{L'}$. O ponteiro $atual$ indica o nó corrente em exame de L. A variável ant indica o nó anterior ao corrente. A função OCUPAR(pt) cria um nó, apontado por pt, contendo dois campos $info_{L'}$ e $pont_{L'}$.

algoritmo: Inversão de uma lista
$ant := \lambda$; $atual := ptlista_L$;
enquanto $atual \neq \lambda$ efetuar
  CRIAR(pt)
  $pt \uparrow .info_{L'} := atual \uparrow .info_L$
  $pt \uparrow .pont_{L'} := ant$
 $ant := atual$
 $atual := atual \uparrow .pont_L$
$ptlista_{L'} := ant$

4. (Até 2,0) Suponha que duas pilhas P_1 e P_2 compartilhem a mesma memória, constituída de um vetor de tamanho n. Isto é, sobre o mesmo vetor X, com elementos $x_1, \ldots, x_n$, são projetadas duas pilhas, a primeira desenvolvendo-se de x_1 para x_n, e a segunda de x_n para x_1. Escrever algoritmos de inclusão e remoção de elementos em P_1 e P_2. Em particular quais seriam as condições de overflow para P_1 e P_2 ?

SOLUÇÃO

Sejam $topo_1$ e $topo_2$ variáveis para apontar para os topos de P_1 e P_2, inicializadas com os valores *zero* e n, respectivamente.

Comecemos pela condição de overflow. Para ambas as pilhas, ela é expressa por:

$$topo_1 = topo_2 - 1.$$

Esta condição significa que ambas as pilhas estão cheias: P_1 não pode crescer em direção a índices maiores, e P_2 não pode crescer em direção a índices menores. Passemos aos algoritmos.

Algoritmo de inclusão em P_1:

se $topo_1 < topo_2 - 1$ então $topo_1 := topo_1 + 1; X[topo_1] := info$ fim-então

Algoritmo de inclusão em P_2:

se $topo_1 < topo_2 - 1$ então $topo_2 := topo_2 - 1; X[topo_2] := info$ fim-então

Algoritmo de exclusão de P_1:

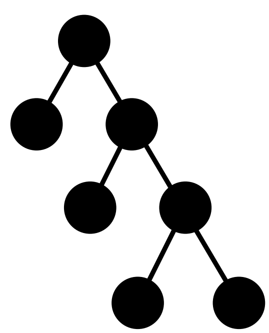
se $topo_1 > 0$ então $info := X[topo_1]; topo_1 := topo_1 - 1$ fim-então

Algoritmo de exclusão de P_2:

se $topo_2 < n$ então $info := X[topo_2]; topo_2 := topo_2 + 1$ fim-então

5. (Até 1,0) Desenhar a menor (em número de nós) árvore estritamente binária de altura 4.

SOLUÇÃO



6. (Até 1,5) Determinar o número de nós e o número de folhas de uma árvore binária cheia que possua k níveis, em função de k.

SOLUÇÃO

Admitindo que a raiz está no nível 1, temos que cada nível j tem exatamente 2^{j-1} nós. Portanto, sendo n o número de nós, temos:

$$n = 2^0 + 2^1 + 2^2 + \cdots + 2^{k-1} = 2^k - 1.$$

O número de folhas f é igual ao número de nós do último nível, isto é, $f = 2^{k-1}$.