Algoritmos e Estruturas de Dados II
Lista de Exercícios de Listas Lineares

Bacharelado em Análise de Sistemas, DCT–UFMS, 20/4/2005

1 Listas lineares em alocação seqüencial

1. Descreva os algoritmos de inserção e remoção de uma lista ordenada em alocação seqüencial.

2. Sejam os seguintes algoritmos de busca em uma lista ordenada de tamanho n em alocação seqüencial:

BUSCA–ORD$^1(\mathcal{L}, n, x)$

$\mathcal{L}[n+1]$.chave $\leftarrow x$
$i \leftarrow 1$
enquanto $\mathcal{L}[i]$.chave $< x$ **faça**
    $i \leftarrow i+1$
    se $i = n+1$ **ou** $\mathcal{L}[i]$.chave $\neq x$ **então**
        devolva -1
    senão
        devolva i

BUSCA–ORD$^2(\mathcal{L}, n, x)$

se $x \leqslant \mathcal{L}[n]$.chave **então**
    $i \leftarrow 1$
    enquanto $\mathcal{L}[i]$.chave $< x$ **faça**
        $i \leftarrow i+1$
    se $\mathcal{L}[i]$.chave $\neq x$ **então**
        devolva -1
    senão
        devolva i
senão
    devolva -1

(a) Compare os algoritmos acima em termos do número total de operações realizadas e de suas complexidades de tempo.
(b) Em que situação o desempenho dos dois é equivalente?
(c) Qual é a restrição que o BUSCA-ORD^2 apresenta em relação ao BUSCA-ORD^1?

3. Um **deque** é uma lista linear particular tal que as operações de inserção e remoção são permitidas em qualquer extremidade da lista. Descreva algoritmos para um deque em alocação seqüencial, considerenado:

   (a) o algoritmo de inserção recebe como entrada o deque, o novo elemento a ser inserido e a extremidade desejada E_1 ou E_2;
   (b) o algoritmo de remoção recebe o deque e a extremidade da remoção.

2 Listas lineares em alocação encadeada

4. Se conhecemos apenas o apontador `apt` para um nó de uma lista linear em alocação encadeada, como na figura 1 e nada mais é conhecido, como podemos modificar a lista linear de modo que passe a conter apenas as chaves $\langle 20, 4, 19, 47\rangle$, isto é, sem aquela do nó apontado por `apt`?

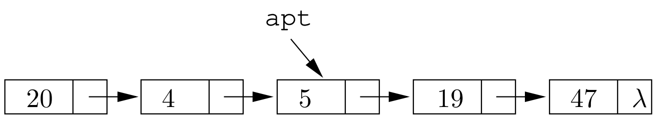

Figura 1:

5. O esquema apresentado na figura 2 permite percorrer uma lista linear simplesmente encadeada nos 2 sentidos, utilizando apenas o atributo `próx` que contém o endereço do próximo elemento da lista. Utilizamos dois apontadores `esq` e `dir`, que apontam para dois elementos vizinhos da lista. A idéia deste esquema é que à medida que os apontadores `esq` e `dir` caminham na lista, os campos `próx` são invertidos de maneira a permitir o tráfego nos dois sentidos.

   Escreva algoritmos para

   (a) mover `esq` e `dir` para a direita de uma posição.
   (b) mover `esq` e `dir` para a esquerda de uma posição.

6. Sejam duas listas, ordenadas, simplesmente encadeadas com nó-cabeça. Apresentar um algoritmo que intercale as duas listas de forma que a lista resultante esteja também ordenada.

7. Seja $\mathcal{L}$ uma lista simplesmente encadeada, com chaves $l_1, l_2, \ldots, l_n$, respectivamente, segundo a ordem de armazenamento. Escreva um algoritmo que, percorrendo $\mathcal{L}$ um única vez, constrói uma outra lista $\mathcal{L}'$, formada dos seguintes elementos:

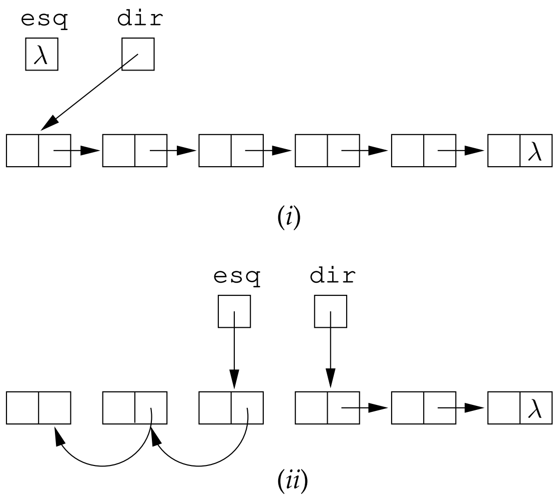

Figura 2: A ilustração (*ii*) foi obtida de (*i*) através da execução do algoritmo em (a) por três vezes.

(a) $l_2, l_3, \ldots, l_n, l_1$.

(b) $l_n, l_{n-1}, \ldots, l_1$.

(c) $l_1 + l_n, l_2 + l_{n-1}, \ldots, l_{n/2} + l_{n/2+1}$, onde n é par.

8. Seja um polinômio da forma $p(x) = a_0x^n + a_1x^{n-1} + \ldots + a_nx^0$. Represente $p(x)$ através de uma lista encadeada conveniente e escreva algoritmos eficientes para efetuar as seguintes operações, onde $q(x)$ é um outro polinômio.

   (a) Calcule $p(x_0)$, onde x_0 é um dado valor para x.

   (b) Calcule $p(q(x_0))$, onde x_0 é um dado valor para x.

   (c) Obtenha $p(x) + q(x)$.

   (d) Obtenha $p(x) \cdot q(x)$.

9. Suponha que uma frase é representada por uma lista linear encadeada, sendo que o campo chave de cada nó da lista contém um único caracter ou símbolo. Escreva um conjunto de procedimentos para manipular uma lista como essa, como segue[1]:

   (a) Coverta a lista seqüencial F em uma lista encadeada, devolvendo o apontador para o nó-cabeça da lista encadeada;

   (b) Converta a lista encadeada apontada por `apt` em uma lista seqüencial F;

   (c) Dados dois apontadores $\texttt{aptlista}_1$ e $\texttt{aptlista}_2$, devolva um inteiro indicando a posição de início da primeira ocorrência da segunda frase dentro da primeira frase. Se a segunda frase não ocorre dentro da primeira, então -1 é devolvido;

[1]Considere $\texttt{aptlista}_1$, $\texttt{aptlista}_2$ e `apt` apontadores para nós-cabeças de listas representando frases, F um vetor de caracteres ou símbolos e i_1 e i_2 inteiros.

(d) Dados dois apontadores $\texttt{aptlista}_1$ e $\texttt{aptlista}_2$, devolva um inteiro indicando a primeira posição da frase apontada por $\texttt{aptlista}_1$ que não está contida na frase apontada por $\texttt{aptlista}_2$;

(e) Dados $\texttt{aptlista}_1$, i_1 e i_2, crie um lista encadeada com nó-cabeça, representando a palavra que inicia em i_1 e termina em i_2 da frase apontada por $\texttt{aptlista}_1$. Devolva o apontador para o nó-cabeça da lista encadeada que representa a palavra. A lista apontada por $\texttt{aptlista}_1$ não deve ser modificada;

(f) Dados $\texttt{aptlista}_1$, i_1, i_2 e $\texttt{aptlista}_2$, os elementos da lista apontada por $\texttt{apt-lista}_2$ devem ser substituídos por i_2 elementos da lista apontada por $\texttt{aptlista}_1$, começando da posição i_1. A lista apontada por $\texttt{aptlista}_2$ não deve ser modificada;

(g) Dados dois apontadores $\texttt{aptlista}_1$ e $\texttt{aptlista}_2$, devolva -1 se a frase apontada por $\texttt{aptlista}_1$ é menor que a frase apontada por $\texttt{aptlista}_2$, 0 se são do mesmo tamanho, e 1 se a frase apontada por $\texttt{aptlista}_1$ é maior.

10. Descreva os algoritmos de inserção e remoção em uma lista não ordenada, em alocação encadeada.

11. Comparar algoritmos de busca, inserção e remoção em uma lista ordenada em alocações seqüencial e encadeada.

3 Pilhas e filas

12. (a) Escreva um algoritmo para "apagar" uma pilha em alocação encadeada, isto é, devolver todos os seus elementos à *Lista de Espaço Disponível*.

    (b) Suponha que a operação descrita acima ocorra com muita freqüência. De que maneira você modificaria a representação da pilha em alocação encadeada para acelerar a operação?

13. Seja $1, 2, \ldots, n$ uma seqüência de elementos que serão inseridos e posteriormente removidos de uma pilha $\mathcal{P}$ uma vez cada. A ordem de inserção dos elementos na pilha é $1, 2, \ldots, n$ enquanto a de remoção depende das operações realizadas.

    Exemplo:

    Com $n = 3$, a seqüência de operações

    incluir em $\mathcal{P}$,
    incluir em $\mathcal{P}$,
    remover de $\mathcal{P}$,
    incluir em $\mathcal{P}$,
    remover de $\mathcal{P}$,
    remover de $\mathcal{P}$,

    produzirá a permutação $2, 3, 1$ a partir da entrada $1, 2, 3$.

Representando por I e R, respectivamente, as operações de inserção e remoção da pilha, a permutação $2, 3, 1$ do exemplo acima pode ser denotada por *IIRIRR*. De modo geral, uma permutação é chamada **admissível** quando puder ser obtida mediante uma sucessão de inserções e remoções em uma pilha a partir da permutação $1, 2, \ldots, n$. Assim, a permutação $2, 3, 1$ do exemplo acima é admissível.

(a) Determine a permutação correspondente a *IIIRRIRR*, com $n = 4$.

(b) Dê um exemplo de uma permutação não admissível.

(c) Escreva a relação de permutações admissíveis de $1, 2, 3, 4$.

14. Repetir o exercício anterior com uma fila.

15. Um estacionamento possui um único corredor que permite dispor 10 carros. Existe somente uma única entrada/saída do estacionamento em um dos extremos do corredor. Se um cliente quer retirar um carro que não está próximo à saída, todos os carros impedindo sua passagem são retirados, o cliente retira seu carro e os outros carros são recolocados na mesma ordem que estavam originalmente. Escreva um algoritmo que processa o fluxo de chegada/saída deste estacionamento. Cada entrada para o algoritmo contém uma letra 'E' para entrada ou 'S' para saída, e o número da placa do carro. Considere que os carros chegam e saem pela ordem especificada na entrada. O algoritmo deve imprimir uma mensagem sempre que um carro chega ou sai. Quando um carro chega, a mensagem deve especificar se existe ou não vaga para o carro no estacionamento. Se não existe vaga, o carro não entra no estacionamento e vai embora. Quando um carro sai do estacionamento, a mensagem deve incluir o número de vezes que o carro foi movimentado para fora da garagem, para permitir que outros carros pudessem sair.

16. Um estacionamento possui um único corredor que permite dispor 10 carros. Os carros chegam pelo sul do estacionamento e saem pelo norte. Se um cliente quer retirar um carro que não está próximo do extremo norte, todos os carros impedindo sua passagem são retirados, o cliente retira seu carro e os outros carros são recolocados na mesma ordem que estavam originalmente. Sempre que um carro sai, todos os carros do sul são movidos para frente, de modo que as vagas fiquem disponíveis sempre no extremo sul do estacionamento. Escreva um algoritmo que processa o fluxo de chegada/saída deste estacionamento. Cada entrada para o algoritmo contém uma letra 'E' para entrada ou 'S' para saída, e o número da placa do carro. Considere que os carros chegam e saem pela ordem especificada na entrada. O algoritmo deve imprimir uma mensagem sempre que um carro chega ou sai. Quando um carro chega, a mensagem deve especificar se existe ou não vaga para o carro no estacionamento. Se não existe vaga, o carro deve esperar até que exista uma vaga, ou até que uma instrução fornecida pelo usuário indique que o carro deve partir sem que entre no estacionamento. Quando uma vaga torna-se disponível, outra mensagem deve ser impressa. Quando um carro sai do estacionamento, a mensagem deve incluir o número de vezes que o carro foi movimentado dentro da garagem, incluindo a saída mas não a chegada. Este número é 0 se o carro partiu da linha de espera, isto é, se o carro esperava uma vaga, mas partiu sem entrar no estacionamento.

4 Listas lineares circulares

17. Considere duas listas circulares encadeadas, não vazias, apontadas por `aptlista1` e `aptlista2` e o seguinte trecho de algoritmo:

```
apt ← aptlista1↑.próx
aptlista1↑.próx ← aptlista2↑.próx
aptlista2↑.próx ← apt
aptlista1 ← aptlista2
aptlista2 ← λ
```

    (a) Qual o resultado da execução deste algoritmo?

    (b) Qual o resultado se `aptlista1` e `aptlista2` apontam para dois elementos distintos da *mesma* lista circular?

18. Descreva o algoritmo de alteração do campo `chave` de uma lista circular encadeada com nó-cabeça.

19. A lenda conta que Josephus[2] não teria sobrevivido para tornar-se famoso se não fosse o seu talento matemático. Durante a guerra entre judeus e romanos, ele estava entre um bando de 41 judeus rebeldes encurralados pelos romanos em uma caverna. Não havia esperança de vencer os inimigos sem reforços e existia um único cavalo para escapar. Os rebeldes fizeram um pacto para determinar qual deles escaparia em busca de ajuda. Eles formaram um círculo e, começando a partir de um deles, começaram a contar no sentido horário em torno do círculo. Quando a contagem alcançava o número 3, aquele rebelde era removido do círculo e a contagem recomeçava a partir do próximo soldado, até que o último soldado restasse no círculo. Este seria o soldado que tomaria o cavalo e fugiria em busca de ajuda. Mas Josephus, com medo da morte iminente, calculou rapidamente onde ele deveria estar neste círculo para que pudesse ser ele o último soldado restante no círculo.

    Descreva um procedimento geral para solucionar o **Problema de Josephus**: dado um número de soldados n e um número $d \leqslant n$, que estabelece o número fixo de remoção de um elementos, determinar a ordem em que os soldados são eliminados do círculo e qual deles escapará.

[2] Flavius Josephus, historiador famoso do primeiro século.

5 Listas lineares duplamente encadeadas

20. Considere uma lista linear duplamente encadeada contendo os elementos $x_1, x_2, \ldots, x_n$, como na figura 3.

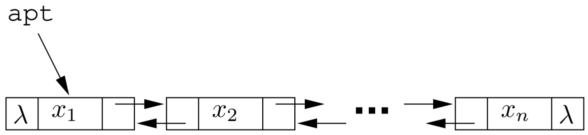

Figura 3:

Escreva um algoritmo que altere os apontadores `ant` e `próx` da lista, *sem* mover suas informações, tal que a lista fique invertida como na figura 4.

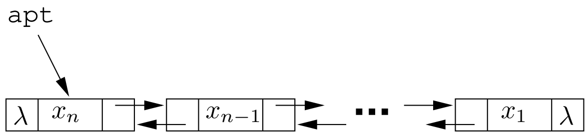

Figura 4: