DCA0214.1 - LABORATÓRIO DE ESTRUTURAS DE DADOS
Aula 6: Listas sequenciais, encadeadas e suas generalizações

Prof. Felipe Fernandes

06 Setembro de 2019

1. Suponha que temos um conjunto $S = \{s_1, ..., s_n\}$ com $n < MAX1$ chaves numéricas distintas, onde $MAX1$ é algum limite superior suficientemente grande. Implemente as seguintes estruturas de dados: lista sequencial ordenada, lista simplesmente encadeada ordenada, lista duplamente encadeada ordenada. Você deve implementar os seguintes procedimentos para manipulação dessas estruturas:
   (a) Busca
   (b) Inserção
   (c) Remoção

2. Dada uma lista encadeada de caracteres formada por uma sequência alternada de letras e dígitos, construa um método que retorne uma lista na qual as letras são mantidas na sequência original e os dígitos são colocados na ordem inversa. Exemplos: $A1E5T7W8G$ deve retornar $AETWG8751$.

3. Imagine que $n > 1$ crianças se organizam em um grande círculo e são numeradas de 1 até n no sentido horário. Seja m um inteiro tal que $1 \leq m < n$. As crianças brincam o seguinte jogo: começando com a criança 1, percorre-se o círculo no sentido horário e retira-se do círculo a m-ésima criança. O procedimento é repetido enquanto o círculo tiver duas ou mais crianças. Ao fim, a criança que sobra é a vencedora da brincadeira. Utilizando uma lista duplamente encadeada circular com cabeça a fim de representar a disposição das crianças, implemente um algoritmo eficiente que recebe o círculo inicial de n crianças e um inteiro positivo m e simula tal brincadeira, retornando, por fim, a criança vencedora.

4. Um palíndromo é uma sequência de caracteres cuja forma é a mesma, quer seja lida da esquerda para a direita ou da direita para a esquerda (exemplo: "arara", "esse"). Faça um algoritmo, utilizando pilhas, que reconheça se uma sequência de caracteres é um palíndromo.

5. Suponha uma estrutura de dados arbitrária chamada **sacola**. Esta estrutura é capaz de se comportar hora como pilha, hora como fila, hora como lista ordenada (crescente ou decrescente). A **sacola** foi implementada numa função, capaz de receber um conjunto de valores numa determinada ordem e retorná-los dispostos em alguma outra ordem. Observando tais ordens de entrada e saída, talvez sejamos capazes de determinar qual o comportamento característico da nossa **sacola**. Por exemplo, se a entrada for 3 10 2 e a saída for 3 10 2, é certo tratar-se de um comportamento do tipo fila. Por outro lado, se a entrada for 2 3 10 e a saída for 2 3 10, dizemos que nossa análise é **indefinida**, pois o comportamento pode ser de uma lista ordenada ou de uma fila. Se a entrada for 20 3 48 e a saída for 3 48 20, então o comportamento não corresponde a nenhum daqueles supracitados, logo dizemos que é **indeterminado**. Escreva um programa que recebe um inteiro n, em seguida recebe duas permutações de n valores. Seu programa deve indicar se o comportamento observado é de pilha, fila, lista ordenada em ordem crescente ou decrescente, indefinido ou indeterminado.

6. O Problema da Torre de Hanoi apresenta três pinos $(P1, P2, P3)$ e n discos. Inicialmente, todos os discos estão empilhados em $P1$, em ordem decrescente de diâmetro, de baixo para cima. Deve-se mover os discos de $P1$ para $P2$, utilizando $P3$ como auxiliar, de tal modo a respeitar as seguintes regras: (1) apenas um disco é movido por vez; (2) pode-se mover apenas o disco do topo da pilha de discos; (3) jamais um disco de maior diâmetro é empilhado sobre um menor. Considere o algoritmo recursivo que soluciona a Torre de Hanoi.

Algoritmo 1: Torre de Hanoi

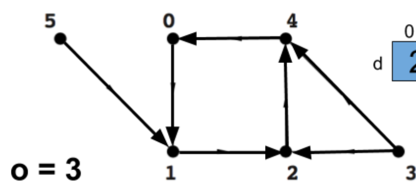
Procedimento `hanoi`$(n, P1, P2, P3)$
    se $n == 1$ **então**
        mover o disco do topo do pino $P1$ para $P2$
    senão
        hanoi$(n-1, P1, P3, P2)$
        mover o disco do topo do pino $P1$ para $P2$
        hanoi$(n-1, P3, P2, P1)$
    fim

Sua tarefa neste exercício consiste em implementar o algoritmo da Torre de Hanoi de modo **iterativo**, utilizando apenas **uma** pilha. Seu algoritmo iterativo deve simular a pilha de recursão que é mantida na versão recursiva. Note que você deve implementar pelo menos três procedimentos: *empilhar* (que insere um elemento na pilha), *desempilhar* (que remove um elemento do topo da pilha) e o *hanoi* (que soluciona a torre de Hanoi de modo iterativo).

7. Suponha que temos n cidades numeradas de 0 a $n-1$ e interligadas por estradas de mão única. As ligações entre as cidades são representadas por

uma matriz A definida da seguinte forma: $A[x][y]$ vale 1 se existe estrada da cidade x para a cidade y e vale 0 em caso contrário. A figura abaixo ilustra um exemplo. Observe que, pela definição acima, **não** há garantias que $A[x][y] = A[y][x]$, pata todo x, y. O problema que queremos resolver é o seguinte: determinar a menor distância de uma dada cidade o a cada uma das outras cidades da rede. As distâncias são armazenadas em um vetor d de tal modo que $d[x]$ seja a menor distância de o a x. Se for impossível chegar de o a x, podemos dizer que $d[x]$ vale -1. Implemente um algoritmo eficiente que, dado a matriz A e uma cidade $0 \leq o < n$, retorne o vetor d de menores distâncias. Dica: use uma fila.

	0	1	2	3	4	5
0	0	1	0	0	0	0
1	0	0	1	0	0	0
2	0	0	0	0	1	0
3	0	0	1	0	1	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0



	0	1	2	3	4	5
d	2	3	1	0	1	-1

8. Seja um tabuleiro com n-por-n posições, modelado por uma matriz $A[n][n]$. As posições "livres" são marcadas com 0 e as posições "bloqueadas" são marcadas com -1. As posições (0, 0) e (n-1, n-1) estão livres. Escreva um algoritmo eficiente que ajude uma formiguinha, que está inicialmente na posição (0, 0), a chegar à posição (n-1, n-1). Em cada posição, a formiguinha só pode se deslocar para uma posição livre que esteja à direita, à esquerda, acima ou abaixo da posição corrente. Seu algoritmo deve imprimir o caminho a ser percorrido pela formiguinha até o destino. Dica: utilize uma fila para achar o caminho de saída e utilize uma pilha para construir (ou recuperar) tal caminho.

	0	1	2	3	4	5
0	0	-1	0	0	0	-1
1	0	0	-1	0	-1	0
2	0	0	0	0	0	-1
3	0	-1	-1	0	0	0
4	-1	0	0	0	-1	-1
5	0	0	-1	0	0	0

9. A notação polonesa reversa é uma maneira de denotar expressões aritméticas. Nesta notação, os operadores aparecem depois dos operandos correspondentes. Por exemplo, a expressão $((A * B) - (C/D))$ tradicional totalmente parentizada pode ser representada pela expressão polonesa reversa $AB * CD/-$. Utilizando uma estrutura de dados do tipo pilha, implemente:

   (a) Um algoritmo eficiente que transforme uma expressão da notação totalmente parentizada para notação polonesa reversa (posfixa).

   (b) Um algoritmo eficiente que transforme uma expressão na notação polonesa reversa numa notação totalmente parentizada.

10. Seja um polinômio da forma $P(x) = a_0x^n + a_1x^{n-1} + ... + a_n$. Representar $P(x)$ através de uma lista encadeada conveniente e escrever algoritmos eficientes para efetuar as seguintes operações, onde $Q(x)$ é um outro polinômio.

    (a) Dado um valor x, calcular $P(x)$

    (b) $P(x) + Q(x)$

    (c) $P(x) * Q(x)$

11. Seja A uma matriz esparsa $n \times m$, isto é, boa parte dos seus elementos são nulos ou irrelevantes. A expressão (1) fornece um exemplo de matriz esparsa. Faça o que se pede:

$$A = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \\ 4 & 0 & 0 & 0 \end{bmatrix} \tag{1}$$

(a) Implementar uma estrutura de dados que represente a matriz A e cujo espaço de armazenamento seja da ordem de $O(n + m + k)$, em vez de $O(nm)$, onde k é o total de elementos não nulos de A.

(b) Implementar um algoritmo eficiente para acessar o elemento $A[i][j]$ da matriz A representada pela estrutura do item (a).

(c) Dadas duas matrizes A e B, representadas pela estrutura do item (a), implementar um algoritmo eficiente que calcule a soma $A + B$.

12. Um deque é uma lista linear que permite a inserção e a remoção de elementos em ambos os seus extremos. Implemente quatro funções para manipular um deque: uma que realiza a inserção de um novo elemento no início do deque, uma que realiza a inserção de um novo elemento no fim do deque, uma que realiza a remoção de um elemento no início do deque e uma que realiza a remoção de um elemento no fim do deque. Utilize uma lista duplamente encadeada.