Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Algoritmos e Estrutura de Dados I – CIC102 / 20091
Professor: David Menotti (menottid@gmail.com)

Trabalho Prático 2

Listas por Cursores e Alocação de Memória

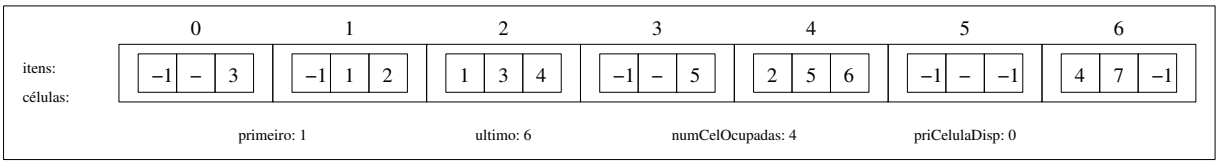
Valor: 0,5 pontos (5% da nota total) **Data de entrega: 21/04/2009**
Documentação em Latex: +0,1 pontos
Interface gráfica: +0,1 pontos

(Extraído e modificado de [2] apud [1]) Implemente o tipo abstrato de dados ***Area***, cuja finalidade é gerenciar uma área interna de memória de forma que o maior e o menor elemento possam ser **removidos** dessa área a um custo O(1). A estrutura de dados que deve ser utilizada é uma lista linear duplamente encadeada implementada por **cursores**. Os cursores são números inteiros que representam posições em um arranjo e são utilizados para simular os apontadores da implementação tradicional das listas lineares duplamente encadeadas. A utilização de cursores evita a alocação e a liberação dinâmica de itens de memória, sendo mais eficiente em aplicações muito dinâmicas em que o número máximo de itens é conhecido.

O tipo abstrato de dados ***Area*** possui as seguintes operações:

a. Criar uma área de memória interna vazia.
b. Obter o número de células ocupadas na área de memória
c. Inserir um item de dado na área interna de memória, mantendo os itens ordenados.
d. Retirar o primeiro item da área de memória.
e. Retirar o último item da área de memória
f. Imprimir o conteúdo da área de memória.

A retirada de um item de uma lista duplamente encadeada pode ser realizada a um custo constante, desde que se conheça previamente o endereço do item na lista. Ao manter a lista ordenada, os elementos de menor e de maior chave estão na primeira e na última posição respectivamente. A figura abaixo ilustra uma lista com capacidade máxima de sete itens.



Após a realização de várias inserções e remoções, os itens contidos na lista apresentada na figura acima possuem as chaves 1, 3, 5 e 7. Os itens de dados da lista linear duplamente encadeada são armazenados em um vetor do tipo *Celula*. Cada entrada do vetor contém uma estrutura que armazena um item de dado, um cursor que aponta para a célula que sucede aquela entrada (*prox*) e um cursor que aponta para a célula que antecede aquela entrada (*ant*). Além disso, são representados dois cursores, *primeiro* e *ultimo*, que apontam para a primeira e para a última célula da lista, respectivamente. Para facilitar o controle de quando a lista se encontra cheia ou vazia, utilize os campo *numCelOcupadas*, que indica quantas células da lista estão ocupadas.

O código abaixo mostra a declaração do tipo abstrato de dados Area. As células armazenados na estrutura *Area* precisam ser mantidas ordenadas. Observe que você deverá implementar uma função que permita comparar dois itens, visto que os itens são inseridos e mantidos ordenados.

```
#define TAMCELS 7
typedef struct {
  int chave;
  /* outros campos */
} Item;
typedef struct {
  Item item;
  int prox, ant;
} Celula;
typedef struct {
  Celula itens[TAMCELS] ;
  int priCelulaDisp, primeiro, ultimo;
  int numCelOcupadas;
} Area;
```

Somente o que foi apresentado até agora não é suficiente para implementar o tipo abstrato de dados *Area*. Isso porque, para incluir um novo item de dado na lista, é necessário haver células disponíveis a fim de que a inserção seja realizada. Assim, para gerenciar a lista de células disponíveis em determinado instante, basta incluir um cursor na representação da estrutura de dados *Area*, o qual irá apontar para a primeira célula disponível. Tal cursor foi denominado *priCelulaDisp*. Como a lista ilustrada pela figura acima possui capacidade para sete itens/células e *numCelOcupadas* = 4, então existem três células disponíveis. A primeira delas é apontada por *priCelulaDisp*, ou seja, o índice zero do vetor de itens, a segunda é indicada pelo cursor *prox* da célula apontada por *priCelulaDisp*, ou seja, o índice três do vetor de itens/células. A terceira e última célula disponível é apontada pelo cursor *prox* da segunda e se encontra no índice cinco do vetor de itens/células. Ela é a última, pois o seu cursor *prox* possui o valor -1, o que indica a falta de um sucessor para ela.

Dessa forma, antes de incluir um novo item de dado em *Area*, remove-se a primeira célula da lista de disponíveis (apontada por *priCelulaDisp*) e a insere ordenamente na lista linear duplamente encadeada que armazena os itens de dados de *Area* (o custo desta inserção não é O(1), pode ser O(*n*) no pior caso). Já ao remover um item de dados de *Area*, a célula que continha tal item deve ser inserida na lista de disponíveis. Para que a inserção e a remoção da **lista de disponíveis** seja realizada a um custo constante, elas devem ser realizadas na posição apontada por *priCelulaDisp*.

Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Algoritmos e Estrutura de Dados I – CIC102 / 20091
Professor: David Menotti (menottid@gmail.com)

O que deve ser entregue

- Código fonte do programa em C ou C++ (bem identada e comentada).
- Documentação do trabalho. Entre outras coisas, a documentação deve conter:
  1. Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.
  2. Implementação: descrição sobre a implementação do programa. Deve ser detalhada a estrutura de dados utilizada (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.
  3. Estudo de Complexidade: estudo da complexidade do tempo de execução dos procedimentos implementados e do programa como um todo (notação O).
  4. Listagem de testes executados: os testes executados devem ser simplesmente apresentados.
  5. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
  6. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso
  7. Em Latex: Caso o trabalho seja elaborado/escrito em latex, ganha-se 0,1 pontos.
  8. Formato: mandatoriamente em PDF (http://www.pdf995.com/).

Obs1: Consulte as dicas do Prof. Nívio Ziviani de como deve ser feita uma boa implementação e documentação de um trabalho prático: http://www.dcc.ufmg.br/~nivio/cursos/aed2/roteiro/

Obs2: Veja modelo de como fazer o trabalho em latex:
http://www.decom.ufop.br/prof/menotti/aedI091/tps/modelo.zip
(caso alguém desenvolva um modelo similar em word, favor me enviar)

Como deve ser feita a entrega:

A entrega DEVE ser feita pelo Moodle na forma de um **único** arquivo zipado, contendo o código, os arquivos e a documentação. Também deve ser entregue a documentação impressa na próxima aula (teórica ou prática) após a data de entrega do trabalho.

Comentários Gerais:

- Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar;
- Clareza, identação e comentários no programa também vão valer pontos;
- O trabalho é individual (grupo de UM aluno);
- Trabalhos copiados (e FONTE) terão nota zero;
- Trabalhos entregue em atraso serão aceitos, todavia a nota atribuída ao trabalho será zero;
- Evite discussões inócuas com o professor em tentar postergar a data de entrega do referido trabalho.

Referências

[1] N. Ziviani, F.C. Botelho, Projeto de Algoritmos com implementações em Java e C++, Editora Thomson, 2006.
[2] F.C. Botelho, Comunicação Pessoal, Belo Horizonte, MG, Brazil, 2003.