Listas Duplamente Encadeadas

Estruturas de Dados e Algoritmos

Listas Duplamente Encadeadas

- A estrutura de lista encadeada vista anteriormente forma um encadeamento simples, onde cada elemento armazena o ponteiro para o próximo elemento da lista

Listas Duplamente Encadeadas

- Uma das limitações dessa estrutura é o percorrimento eficiente da lista em ordem inversa (do final para o início)
- Outra limitação é a retirada de um elemento da lista (sempre exigindo que guardemos a referência para o Nodo anterior ao que está sendo retirado)

Listas Duplamente Encadeadas

- Esses problemas podem ser contornados com a utilização de listas duplamente encadeadas.
- Em uma lista duplamente encadeada, cada elemento possui um ponteiro para o próximo e também para o anterior.
- Com isso, dado um elemento qualquer da lista, é possível acessar ambos os seus elementos adjacentes (próximo e anterior)

Listas Duplamente Encadeadas

- Com a existência de um ponteiro para o último elemento da lista, é possível, por exemplo, percorrer a lista em ordem inversa acessando continuamente o anterior de cada elemento até alcançar o primeiro (cujo anterior é null)

Listas Duplamente Encadeadas

Arranjo da memória de uma lista duplamente encadeada

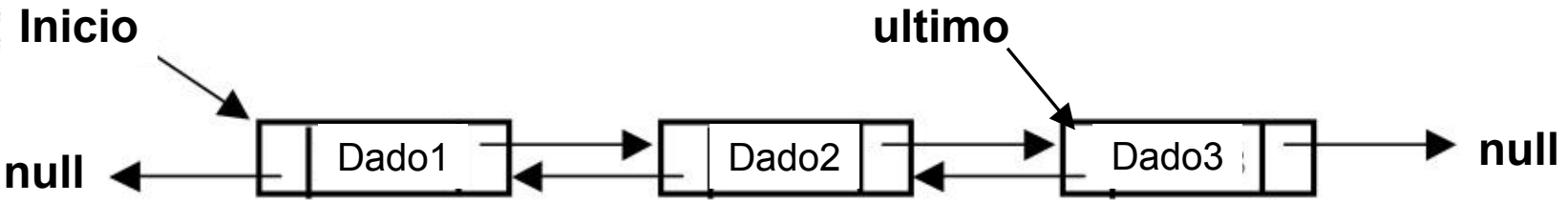

Figura adaptada de (Celes, Serqueira e Rangel, 2004)

Listas Duplamente Encadeadas

- Nessa implementação, o Nodo da lista duplamente encadeada terá um novo atributo que é um apontador para o Nodo anterior

```
public class ListaDuplamenteEncadeada {
    private Nodo inicio;
    private Nodo ultimo;

    private class Nodo {
        int dado;
        Nodo proximo;
        Nodo anterior;
    }

    public ListaDuplamenteEncadeada(){
        inicio = null;
        ultimo = null;
    }
```

Listas Duplamente Encadeadas

- Nessa implementação, o Nodo da lista duplamente encadeada terá um novo atributo que é um apontador para o Nodo anterior

```
public class ListaDuplamenteEncadeada {
    private Nodo inicio;
    private Nodo ultimo;

    private class Nodo {
        int dado;
        Nodo proximo;
        Nodo anterior;
    }

    public ListaDuplamenteEncadeada(){
        inicio = null;
        ultimo = null;
    }
```

Listas Duplamente Encadeadas

Inserção de um novo Nodo no início da lista duplamente encadeada

1. Como o novo elemento é encadeado no início da lista, ele tem como próximo elemento o antigo primeiro elemento da lista e como anterior o valor NULL .
2. Após isso, deve-se testar se a lista não era vazia, nesse caso, o elemento anterior do antigo primeiro elemento passa a ser o novo elmento.
3. O novo elemento passa a ser o primeiro da lista

Listas Duplamente Encadeadas

Arranjo da memória de uma lista duplamente encadeada

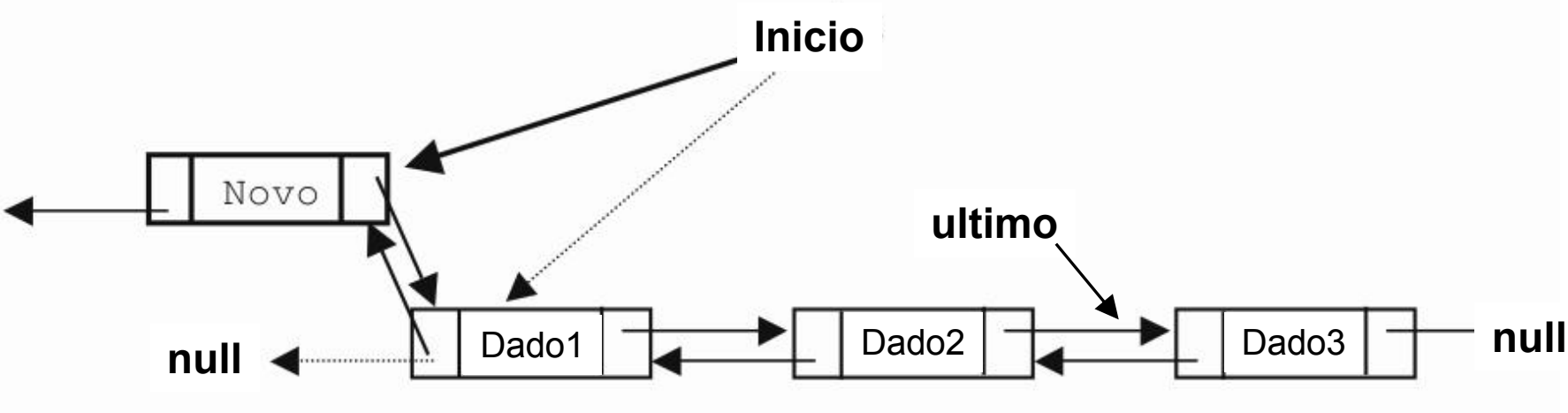

Figura adaptada de (Celes, Serqueira e Rangel, 2004)

Listas Duplamente Encadeadas

Inserção de um novo Nodo no início da lista encadeada

ANTES DE INSERIR O PONTEIRO PARA O ANTERIOR

```
public void insereInicio(int n){
    Nodo novo = new Nodo();
    novo.dado = n;
    novo.proximo = inicio;

    if (this.vazia()){
        inicio = novo;
        ultimo = inicio;
    }
    else {
        inicio = novo;

    }
}
```

Listas Duplamente Encadeadas

Inserção de um novo Nodo no início da lista encadeada

DEPOIS DE INSERIR O PONTEIRO PARA O ANTERIOR

```
public void insereInicio(int n){
    Nodo novo = new Nodo();
    novo.dado = n;
    novo.anterior = null;
    novo.proximo = inicio;

    if (this.vazia()){
        inicio = novo;
        ultimo = inicio;
    }
    else {
        inicio.anterior = novo;
        inicio = novo;

    }
}
```

Listas Duplamente Encadeadas

Inserção de um novo Nodo no início da lista encadeada

DEPOIS DE INSERIR O PONTEIRO PARA O ANTERIOR

```
public void insereInicio(int n){
    Nodo novo = new Nodo();
    novo.dado = n;
    novo.anterior = null;
    novo.proximo = inicio;

    if (this.vazia()){
        inicio = novo;
        ultimo = inicio;
    }
    else {
        inicio.anterior = novo;
        inicio = novo;

    }
}
```

Listas Duplamente Encadeadas

Impressão de elementos em ordem inversa (iniciando pelo final)

1. Utiliza-se o apontador para o último
2. Percorrem-se os elementos utilizando as referências para os nodos anteriores
3. O processo é interrompido após imprimir o primeiro elemento (que possui anterior = NULL)

Listas Duplamente Encadeadas

Impressão de elementos em ordem inversa (iniciando pelo final)

```
public void imprimeReverso(){
    for (Nodo x = ultimo; x!= null; x = x.anterior){
        System.out.print(x.dado+"->");
    }

    System.out.println();
}
```

Listas Duplamente Encadeadas

Remoção de um elemento da lista

1. Se temp representa o ponteiro do elemento que desejamos retirar, para acertar o encadeamento devemos conceitualmente fazer
   - temp.anterior.proximo = temp.proximo
   - temp.proximo.anterior = temp.anterior
2. O anterior passa a apontar para o próximo e o próximo passa a apontar para o anterior
3. Quando temp for um elemento do meio da lista, as duas atribuições acima são suficientes para acertar o encadeamento

Listas Duplamente Encadeadas

Remoção de um elemento da lista

Condições de contorno para remoção nos extremos

Se temp for um elemento no extremo da lista, deve-se considerar as condições de contorno

1. Se temp for o **primeiro** elemento da lista não é possível escrever **temp.anterior.proximo** pois **temp.anterior** é NULL
2. Ainda, é necessário atualizar o início da lista pois o primeiro elemento será removido

Listas Duplamente Encadeadas

Remoção de um elemento da lista

Condições de contorno para remoção nos extremos

Se temp for um elemento no extremo da lista, deve-se considerar as condições de contorno

1. Se temp for o **último** elemento da lista não é possível escrever **temp.proximo.anterior** pois **temp.proximo** é NULL
2. Ainda, é necessário atualizar o ultimo da lista pois o ultimo elemento será removido

Listas Duplamente Encadeadas

Remoção de um elemento da lista

```
public Integer retiraNodo(Integer n){
    Nodo temp = this.inicio;

    while (temp!= null && temp.dado != n){
         temp = temp.proximo;
    }

    if (temp == null){//significa que não achou
        return null;
    }

    int retirado = temp.dado;
    if (temp == inicio) {//primeiro elemento
        inicio = inicio.proximo;
    }
    else {
        temp.anterior.proximo = temp.proximo;
    }

    if (temp.proximo != null) {//verifica se é o último
        temp.proximo.anterior = temp.anterior;
    }
    else {// é o último
        ultimo = temp.anterior;
    }

    return retirado;
}
```

Exercícios

Listas Duplamente Encadeadas

1. Aprimore os métodos de remoção de elementos do início e final da lista considerando o duplo encadeamento.

Listas Duplamente Encadeadas

1. Aprimore os métodos de remoção de elementos do início e final da lista considerando o duplo encadeamento.

```
public Integer retiraInicio(){
    if (vazia()) return null;
    Nodo retirado = inicio;
    if (inicio == ultimo)
        ultimo = inicio = null;
    else {
        inicio.proximo.anterior = null;
        inicio = inicio.proximo;
    }
    return retirado.dado;
}
```

Listas Duplamente Encadeadas

1. Aprimore os métodos de remoção de elementos do início e final da lista considerando o duplo encadeamento.

```
public Integer retiraUltimo(){
    if (vazia()) return null;
    Nodo retirado = ultimo;
    if (inicio == ultimo)
        ultimo = inicio = null;
    else {
        ultimo.anterior.proximo = null;
        ultimo = ultimo.anterior;
    }

    return retirado.dado;
}
```

Listas Duplamente Encadeadas

2. Um DEQUE (Double Ended Queue) é uma fila que contém duas extremidades (início e final), e que permite que as inserções, exclusões e consultas sejam realizadas em qualquer uma dessas extremidades.

Utilize a classe ListaDuplamenteEncadeada para realizar a implementação de uma classe DEQUE que contenha os seguintes métodos:

a) Métodos para inserir elemento no início e no final do DEQUE

b) Métodos para remover elemento do início e do final do DEQUE.

c) Métodos para consultar o primeiro e o último elemento do DEQUE

Referências utilizadas

- CELES, W., R. CERQUEIRA, and JL RANGEL. Introdução a Estruturas de Dados. Rio de Janeiro, RJ, Brasil: Campus, 2004. 294 p. ISBN 85-352-1228-0.
- Sedgewick, Robert, and Kevin Wayne. Algorithms. Addison-Wesley Professional, 2011.
- Deitel, H. M., & Deitel, P. J. (2003). Java, como programar. 4ª Edição. São Paulo.