Newton

Quem se prepara, não para.

Estrutura de Dados

Professora: Michelle Hanne Soares de Andrade
michelle.andrade@newtonpaiva.br

Newton
Quem se prepara, não para.

Sumário:

- ✓ Lista Duplamente Encadeada

Lista duplamente encadeada

Newton
Quem se prepara, não para.

Listas duplamente encadeadas Em listas simplesmente encadeadas, é difícil desenvolver operações de remoção no meio e no final da lista. **Isso porque um nodo específico não tem informações para o nodo anterior da lista, mas apenas para o nodo seguinte.**

A partir dessa dificuldade de implementação, surge a necessidade de expandir a classe nodo para armazenar uma referência também ao nodo anterior da lista (KNUTH, 1997), quando temos uma lista encadeada na qual os nodos apresentam referências tanto para o nodo anterior quanto para o próximo: uma lista duplamente encadeada (CORMEN et al., 2002).

Fonte: Rodrigues, 2021.

Lista duplamente encadeada

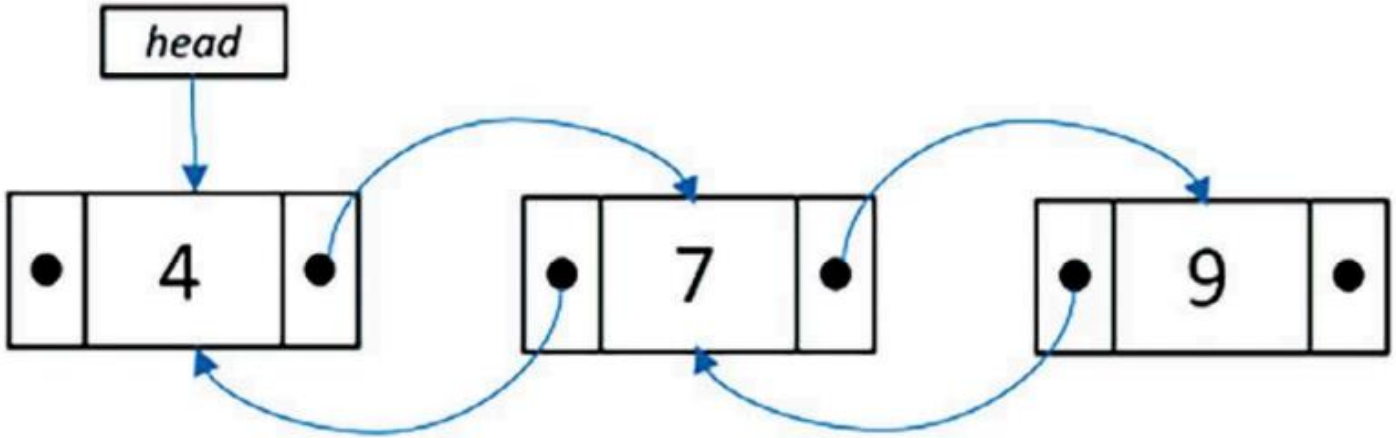

Figura 6. Exemplo de lista duplamente encadeada.
Fonte: Adaptada de Goodrich e Tamassia (2013).

Fonte: Rodrigues, 2021.

Lista duplamente encadeada

Para inserir um elemento após determinado nodo em uma lista duplamente encadeada, é necessário criar um nodo e ajustar as referências anterior e próximo para acomodá-lo na posição correta.

```
        public void adicionarDepois(Nodo nodo, float valor) {
        // É necessário guardar uma referencia para o nodo
posterior
        Nodo nodoPosterior = nodo.getProximo();
        // Criamos o novo nodo que será inserido e
atualizamos o seu valor
        Nodo novoNodo = new Nodo();
        novoNodo.setDado(valor);
        // Ajustamos as referencias do novo nodo primeiro
        novoNodo.setProximo(nodoPosterior);
        novoNodo.setAnterior(nodo);
        // Ajustamos o nodo passado como argumento para que
aponte para o novo nodo
        nodo.setProximo(novoNodo);
        // Caso não seja a última posição da lista, atualiza
a referência
        // do próximo elemento
        if (nodoPosterior != null)
            nodoPosterior.setAnterior(novoNodo);
    }
```

Figura 7. Código 5 — Trecho de código para inserção após um nodo em uma lista duplamente encadeada.

Fonte: Rodrigues, 2021.

Lista duplamente encadeada

De maneira análoga, a remoção de um nodo qualquer de uma **lista duplamente encadeada é facilitada**, pois temos acesso aos elementos anterior e posterior da lista para ajustar as referências, não sendo necessário localizar os nodos específicos. Precisamos tomar cuidado com dois casos específicos: caso o nodo que você está removendo esteja na primeira ou na última posição da lista**, evitando-se, assim, a NullPointerException** (HORSTMANN, 2009).

```
public void removerNodo(Nodo nodo) {
        // Caso o nodo a ser removido seja o primeiro da
lista
        // é necessário ajustar a referência primeiro
        if (this.getPrimeiro() == nodo) {
            this.setPrimeiro(nodo.getProximo());
        } else {
            // Obtemos referências para os nodos anterior e
proximo
            // em relacao ao nodo que queremos excluir
            Nodo anterior = nodo.getAnterior();
            Nodo proximo = nodo.getProximo();
            // Alteramos a referencia proximo do nodo
anterior
            anterior.setProximo(proximo);
            // Caso não seja o ultimo nodo da lista,
atualizamos também
            // o proximo nodo para apontar para o nodo
anterior.
            if (proximo != null) {
                proximo.setAnterior(anterior);
            }
        }
    }
```

Figura 8. Código 6 — Trecho de código para remoção de um nodo em uma lista duplamente encadeada.

Fonte: Rodrigues, 2021.

Desafio:

Newton
Quem se prepara, não para.

Fazer o problema abaixo utilizando uma lista duplamente encadeada

Um casal está fazendo a lista de convidados para o seu casamento. A noiva possui uma lista com 60 convidados e o noivo com 55 convidados.

Implemente um algoritmo que crie uma lista para cada um (noivo e noiva). Em seguida faça a união das listas e imprima o nome de cada convidado por ordem alfabética.

Fonte: Rodrigues, 2021.

Referências

PINTO, Rafael A.; PRESTES, Lucas P.; SERPA, Matheus da S.; et al. Estrutura de dados. Editora SAGAH, 2020. ISBN 9786581492953. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9786581492953

RODRIGUES, Thiago N.; LEOPOLDINO, Fabrício L.; PESSUTTO, Lucas Rafael C.; et al. Estrutura de Dados em Java. Editora SAGAH. 2021. ISBN 9786556901282. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9786556901282