Estruturas de Dados

Listas Duplamente Ligadas

Universidade Federal do Rio Grande do Norte

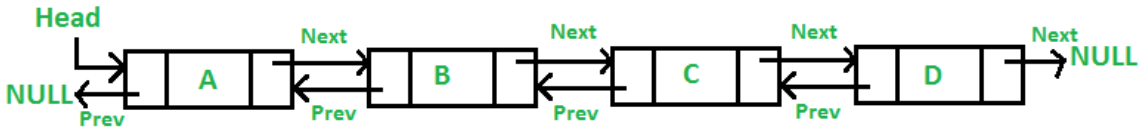
Lista Duplamente Ligada

Lista

Listas são conjuntos de dados em que os objetos estão **organizados de maneira linear**. Diferentemente de arrays, onde a ordem linear é determinada por um índice, nas listas a ordem é determinada por um **ponteiro em cada objeto**.

Lista Duplamente Ligada

São listas que possuem tanto um apontador para o **sucessor** e para o **antecessor**.



FONTE: https://www.geeksforgeeks.org/doubly-linked-list/

Lista Duplamente Ligada

Quantos ponteiros forem necessários

- Um ponteiro externo pode apontar para **qualquer nó da lista**.
- Será possível caminhar para **a direita ou para a esquerda** com igual facilidade.
- Em geral, são guardados os ponteiros para o **primeiro** e para o **último** nó.

Nó de uma lista duplamente ligada

Link para o elemento anterior.	Dado	Link para o próximo elemento.

```
class Node (dado){
        constructor(dado){
                this.dado = dado;
                this.anterior = null;
                this.proximo = null;
        }
}
```

Lista Duplamente Ligada

Outros detalhes

- Uma lista duplamente encadeada pode ser **circular** ou **não**.
- Pode estar em **ordem ou não.**
- Vantagem em relação a lista comum, pois, em alguns casos, **é possível inserir ou remover elementos sem percorrer toda lista.**

Nó de uma lista duplamente ligada

Link para o elemento anterior.	Dado	Link para o próximo elemento.

```
class Node (dado){
        constructor(dado){
                this.dado = dado;
                this.anterior = null;
                this.proximo = null;
        }
}
```

Implementação

Lista Vazia

- Uma lista duplamente encadeada precisa de ponteiros para indicar onde a lista está na memória.
- Em geral são guardados dois ponteiros, um para o **primeiro (head)** e outro para o **último (tail)** elemento da lista.

```
class ListaDuplamenteLigada {
    constructor() {
        this.head = null;
        this.tail = null;
    }
}
```

Principais Operações

add (dado)

Aloca um espaço de memória para um nó e adiciona um dado no início da lista.

append (dado)

Aloca um espaço de memória para um nó e adiciona um dado no final da lista.

addAt (dado, pos)

Aloca um espaço de memória para um nó e adiciona um dado em uma posição específica da lista.

Principais Operações

removeFirst()

Remove um nó do início da lista e libera o espaço de memória usado pelo nó.

removeLast()

Remove um nó do fim da lista e libera o espaço de memória usado pelo nó.

removeAt (pos)

Remove um nó de uma posição específica da lista e libera o espaço de memória usado pelo nó.

Principais Operações

isEmpty()

Verifica se a lista está vazia.

search(dado)

Pesquisa se um dado está presente em algum nó da lista.

Observação

- Os próximos slides detalham a implementação da lista duplamente ligada.
- Perceba que vamos adotar a estratégia de uso de um “nó” especificamente para guardar a cabeça ou a cauda.
- Qualquer um dos nós poderá ser a cabeça ou a cauda.
- Essa implementação é uma alternativa ao uso de um nó específico para guardar a cabeça que foi usada na implementação de Lista Simplesmente Ligada.

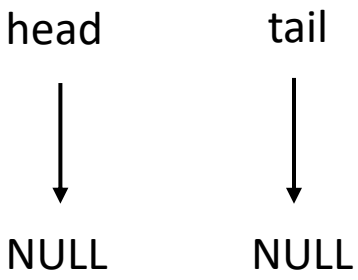
Implementação

Lista vazia

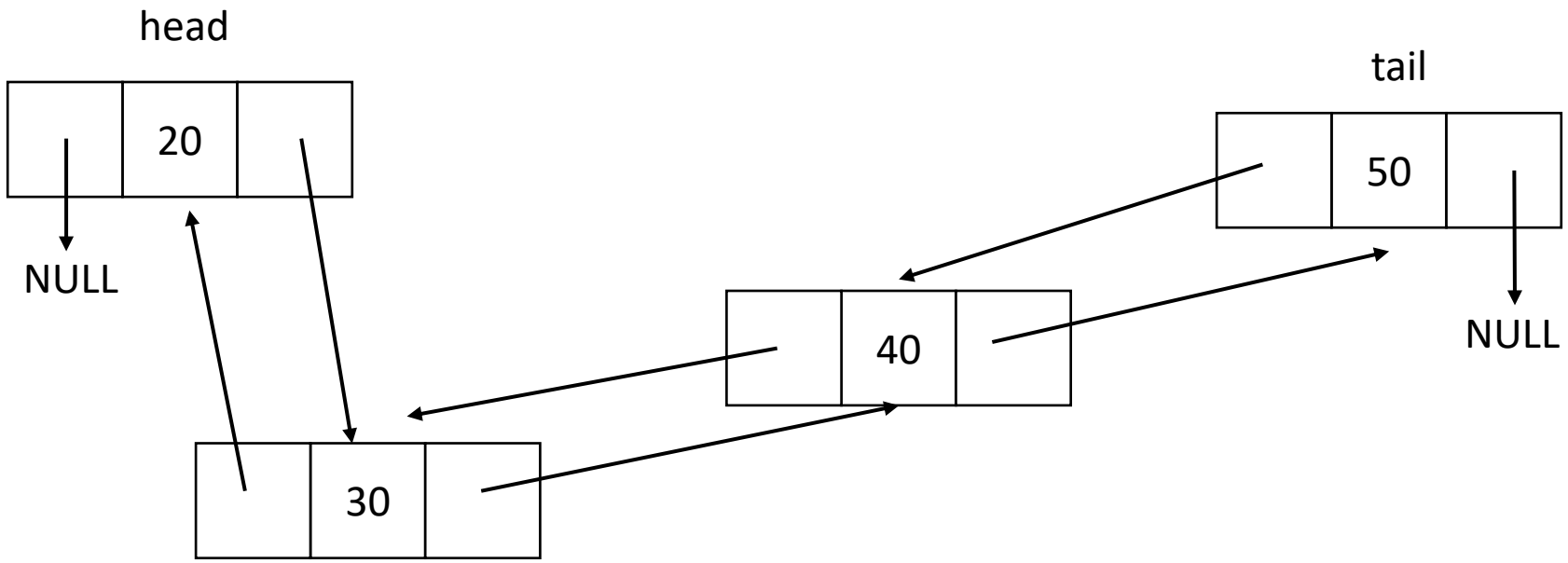
Uma lista duplamente ligada pode ser considerada vazia quando as variáveis que indicam o início ou o fim da lista apontam para NULL.

```
isEmpty() {
    return this.head === null;
}
```

Lista vazia

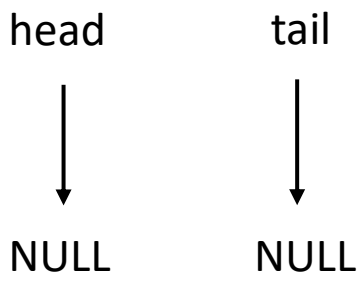


Lista com elementos



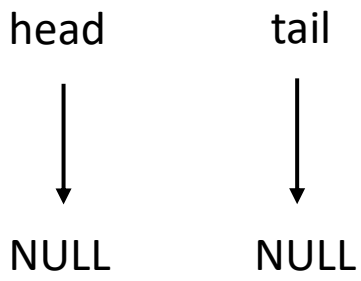
Inserir um elemento na lista vazia

add(20)

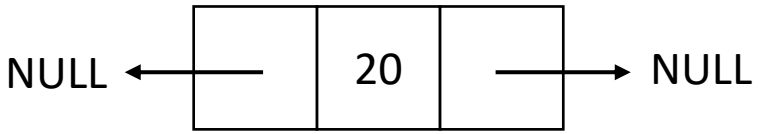


Inserir um elemento na lista vazia

add(20)

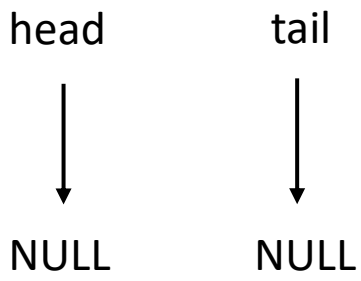


1) Instanciar o novo nó

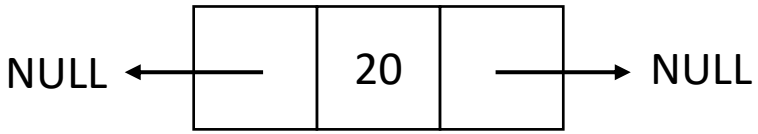


Inserir um elemento na lista vazia

add(20)



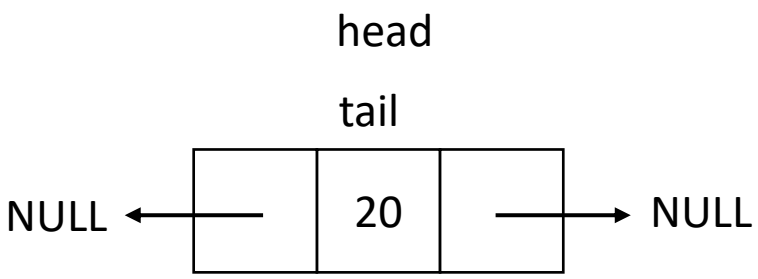
2) head e tail recebem o novo nó.



Inserir um elemento na lista vazia

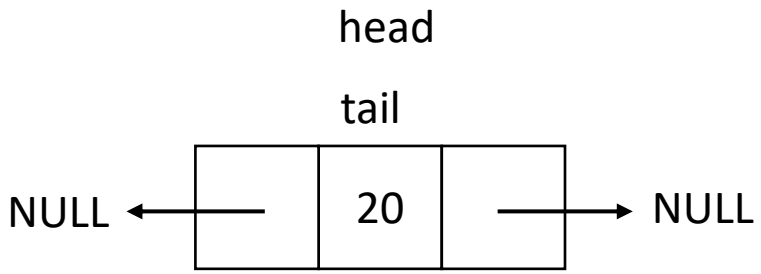
add(20)

2) head e tail recebem o novo nó.



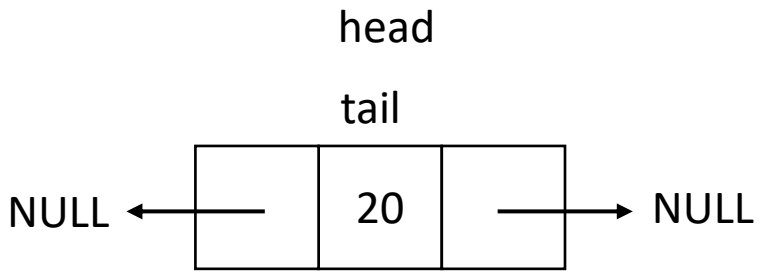
Inserir um elemento na lista não vazia

add(30)

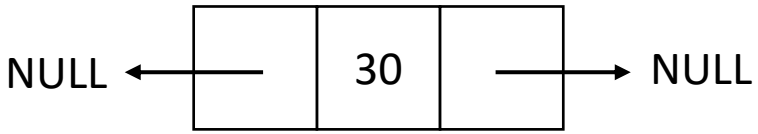


Inserir um elemento na lista não vazia

add(30)

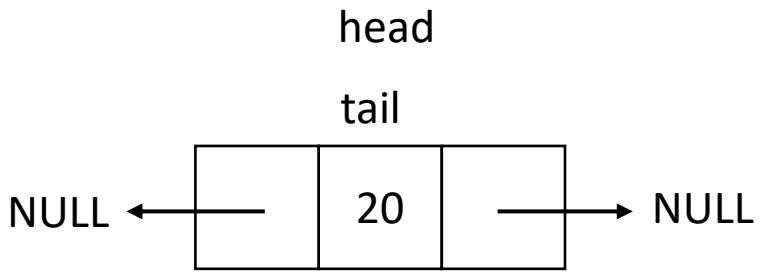


1) Instanciar o novo nó

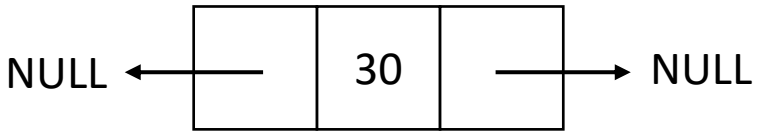


Inserir um elemento na lista não vazia

add(30)

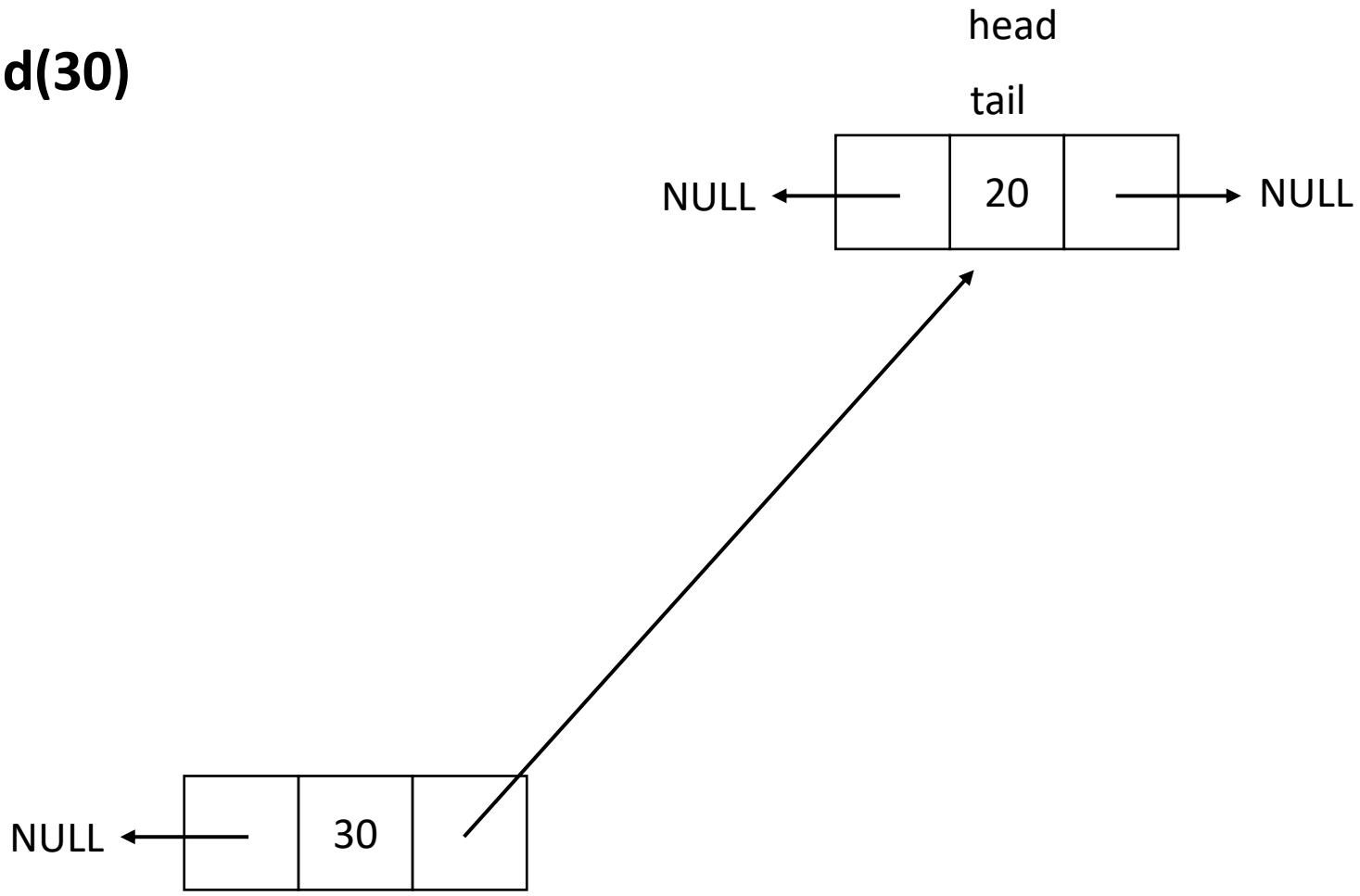


2) O ponteiro para o próximo do novo nó deve apontar para o nó que head aponta.
3) O ponteiro para o anterior do novo nó deve apontar para NULL.
4) O anterior do nó apontado pela cabeça será o novo nó.
5) A cabeça deve ser um apontador para o novo nó.



Inserir um elemento na lista não vazia

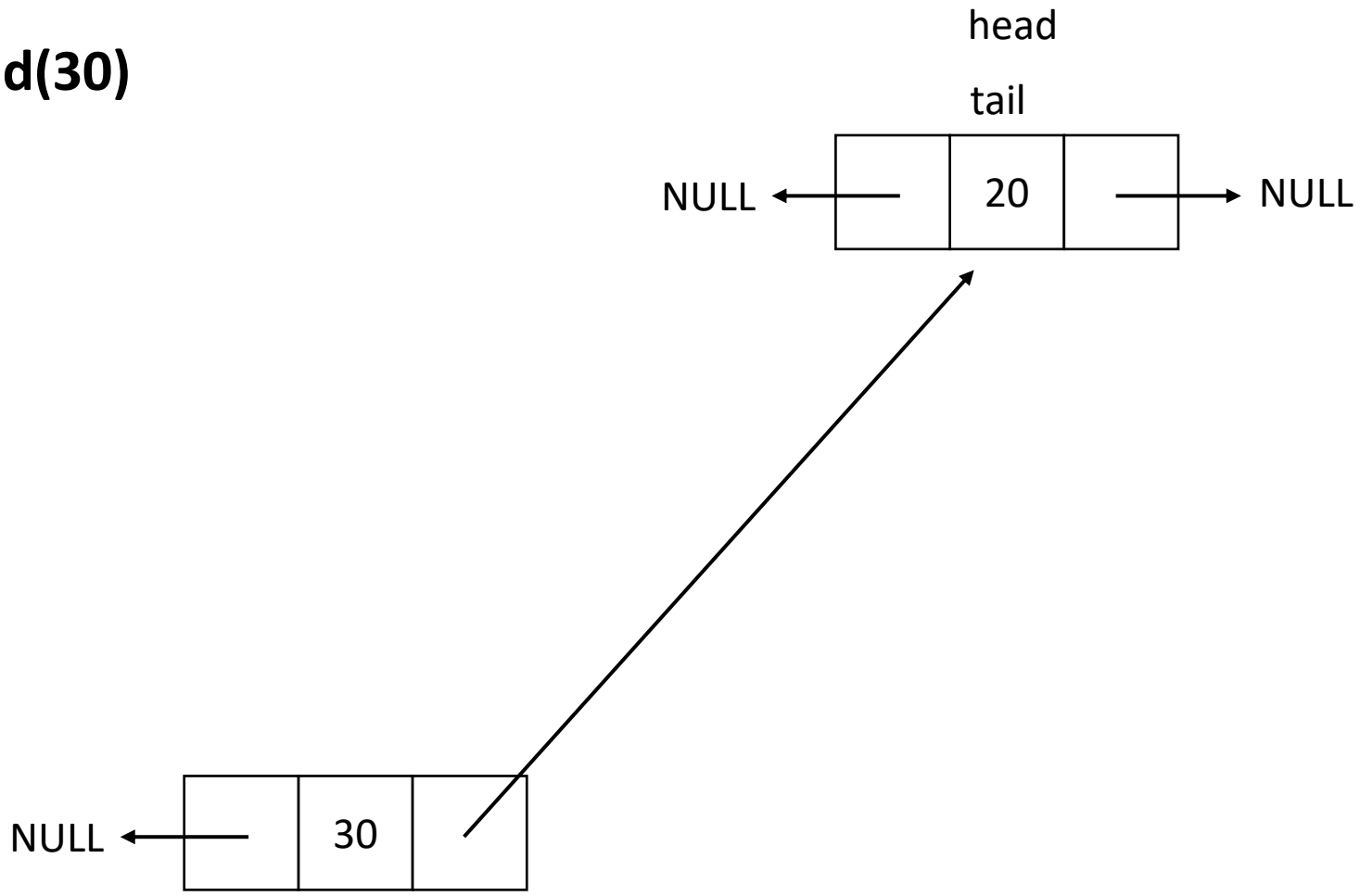
add(30)



2) O ponteiro para o próximo do novo nó deve apontar para o nó que head aponta.
3) O ponteiro para o anterior do novo nó deve apontar para NULL.
4) O anterior do nó apontado pela cabeça será o novo nó.
5) A cabeça deve ser um apontador para o novo nó.

Inserir um elemento na lista não vazia

add(30)



2) O ponteiro para o próximo do novo nó deve apontar para o nó que head aponta.

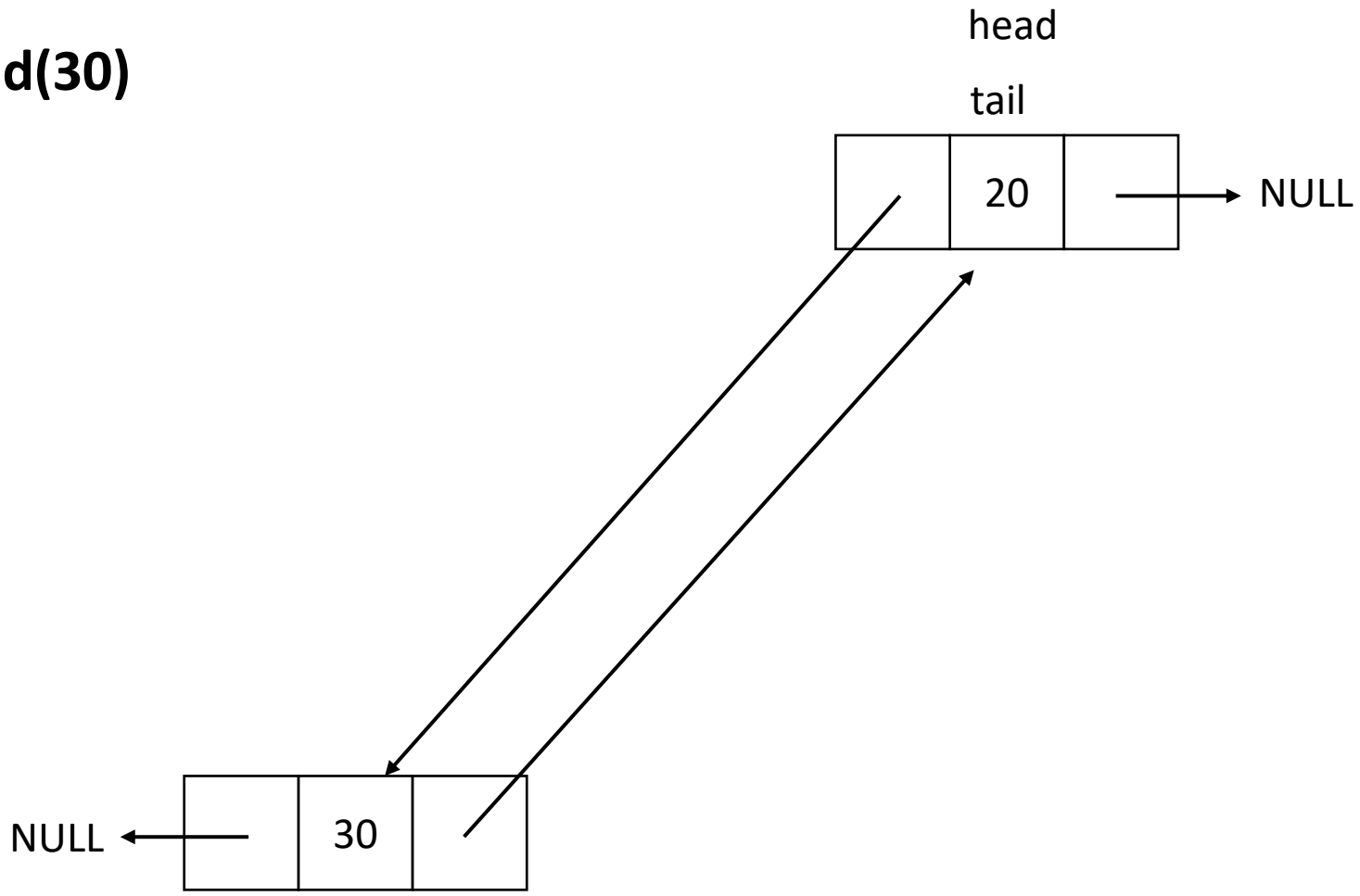
3) O ponteiro para o anterior do novo nó deve apontar para NULL.

4) O anterior do nó apontado pela cabeça será o novo nó.

5) A cabeça deve ser um apontador para o novo nó.

Inserir um elemento na lista não vazia

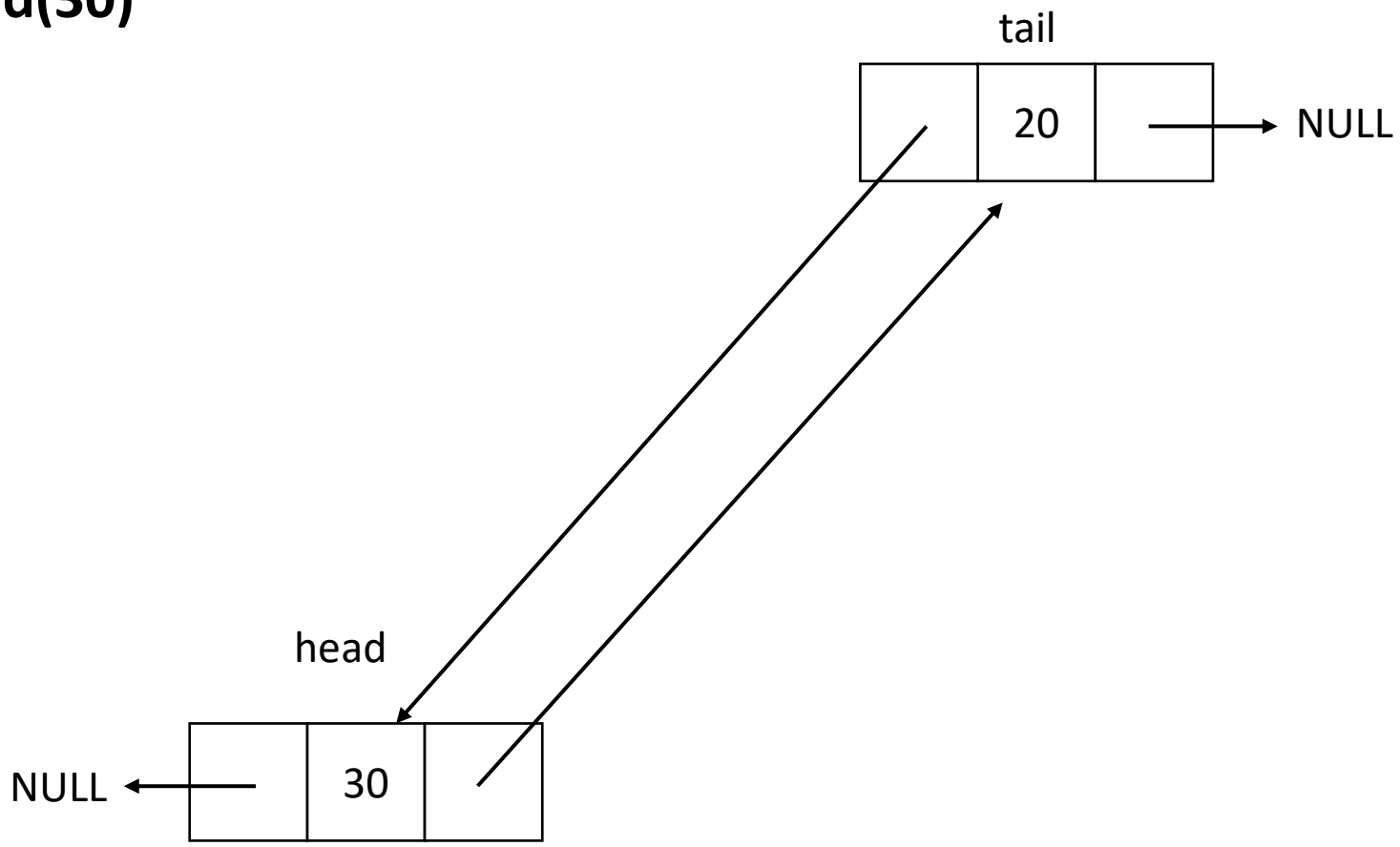
add(30)



2) O ponteiro para o próximo do novo nó deve apontar para o nó que head aponta.
3) O ponteiro para o anterior do novo nó deve apontar para NULL.
4) O anterior do nó apontado pela cabeça será o novo nó.
5) A cabeça deve ser um apontador para o novo nó.

Inserir um elemento na lista não vazia

add(30)



2) O ponteiro para o próximo do novo nó deve apontar para o nó que head aponta.
3) O ponteiro para o anterior do novo nó deve apontar para NULL.
4) O anterior do nó apontado pela cabeça será o novo nó.
5) A cabeça apontar para o novo nó.

Bibliografia Básica

- CORMEN, Thomas H et al. **Algoritmos: teoria e prática.** Rio de Janeiro: Elsevier, 2012. 926 p. ISBN: 9788535236996.
- ASCENCIO, Ana Fernanda Gomes. **Estruturas de dados: algoritmos, análise da complexidade e implementações em Java e C/C++.** São Paulo: Pearson, c2010. 432 p. ISBN: 9788576052216, 97885760581 6.
- PIVA JÚNIOR, Dilermando (et al). **Estrutura de dados e técnicas de programação**. 1. ed. Rio de Janeiro, RJ: Campus, 2014. 399 p. ISBN: 9788535274370.

Bibliografia Complementar

- FERRARI, Roberto et al. **Estruturas de dados com jogos**. 1. ed. Rio de Janeiro: Elsevier, 2014. 259p. ISBN: 9788535278040.
- GRONER, Loiane. **Estruturas de dados e algoritmos em Javascript**: aperfeiçoe suas habilidades conhecendo estruturas de dados e algoritmos clássicos em JavaScript. São Paulo: Novatec, 2017. 302 p. ISBN: 9788575225530.
- SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. **Estruturas de dados e seus algoritmos**. 3. ed. Rio de Janeiro: LTC, 2010. xv, 302 p. ISBN: 9788521617501.
- GOODRICH, Michael T; TAMASSIA, Roberto. **Estruturas de dados e algoritmos em Java**. 5. ed. Porto Alegre: Bookman, 2013. xxii, 713 p. ISBN: 9788582600184.
- GUIMARÃES, Ângelo M. **Algoritmos e estruturas de dados**. LTC, 1994.