Algoritmos e Estruturas de Dados I

LISTAS LIGADAS

Prof. Tiago Eugenio de Melo
tmelo@uea.edu.br

www.tiagodemelo.info

Observações

- O conteúdo dessa aula é parcialmente proveniente do Capítulo 7 do livro "*Data Structures and Algorithms in Python*" e do Capítulo 3 do livro "*Data Structure and Algorithmic Thinking with Python*".
- As palavras com a fonte `Courier` indicam uma palavra-reservada da linguagem de programação.

Introdução

Introdução

- Uma lista ligada é usada para armazenar uma coleção de objetos (dados).

Introdução

- Uma lista ligada é usada para armazenar uma coleção de objetos (dados).
- Propriedades:

Introdução

- Uma lista ligada é usada para armazenar uma coleção de objetos (dados).
- Propriedades:
  - Os elementos sucessivos são ligados por ponteiros.

Introdução

- Uma lista ligada é usada para armazenar uma coleção de objetos (dados).
- Propriedades:
  - Os elementos sucessivos são ligados por ponteiros.
  - O último elemento aponta para NULL.

Introdução

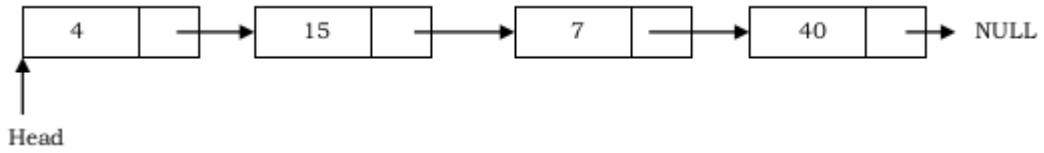
- Uma lista ligada é usada para armazenar uma coleção de objetos (dados).
- Propriedades:
  - Os elementos sucessivos são ligados por ponteiros.
  - O último elemento aponta para NULL.
  - Essa estrutura pode crescer ou diminuir de tamanho durante a sua existência.

Introdução

- Uma lista ligada é usada para armazenar uma coleção de objetos (dados).
- Propriedades:
  - Os elementos sucessivos são ligados por ponteiros.
  - O último elemento aponta para NULL.
  - Essa estrutura pode crescer ou diminuir de tamanho durante a sua existência.
  - Evita o desperdício de memória.

Introdução

- Uma lista ligada é usada para armazenar uma coleção de objetos (dados).
- Propriedades:
  - Os elementos sucessivos são ligados por ponteiros.
  - O último elemento aponta para NULL.
  - Essa estrutura pode crescer ou diminuir de tamanho durante a sua existência.
  - Evita o desperdício de memória.



Introdução

- NULL em Python
  - O objeto null é representado por `None` em Python.

Arrays versus listas ligadas

- Arrays

Arrays versus listas ligadas

- Arrays
  - Simples e fáceis de usar.

Arrays versus listas ligadas

- Arrays
  - Simples e fáceis de usar.
  - Acesso mais rápido aos elementos (objetos) (acesso constante).

Arrays versus listas ligadas

- Arrays
  - Simples e fáceis de usar.
  - Acesso mais rápido aos elementos (objetos) (acesso constante).
  - O tamanho do array é estático.

Arrays versus listas ligadas

- Arrays
  - Simples e fáceis de usar.
  - Acesso mais rápido aos elementos (objetos) (acesso constante).
  - O tamanho do array é estático.
  - Alocação de um bloco inteiro.

Arrays versus listas ligadas

- Arrays
  - Simples e fáceis de usar.
  - Acesso mais rápido aos elementos (objetos) (acesso constante).
  - O tamanho do array é estático.
  - Alocação de um bloco inteiro.
  - Operação de inserção pode ser complexa.

Arrays versus listas ligadas

- Arrays
  - Simples e fáceis de usar.
  - Acesso mais rápido aos elementos (objetos) (acesso constante).
  - O tamanho do array é estático.
  - Alocação de um bloco inteiro.
  - Operação de inserção pode ser complexa.
    - Necessidade de deslocar o restante do array.

Arrays versus listas ligadas

Arrays versus listas ligadas

- Listas ligadas

Arrays versus listas ligadas

- Listas ligadas
  - A expansão da lista pode ser realizada em tempo constante.

Arrays versus listas ligadas

- Listas ligadas
  - A expansão da lista pode ser realizada em tempo constante.
  - Tempo de acesso ao um elemento da lista [O(n)].

Arrays versus listas ligadas

- Listas ligadas
  - A expansão da lista pode ser realizada em tempo constante.
  - Tempo de acesso ao um elemento da lista [O(n)].
  - Memória extra para armazenamento dos ponteiros.

Listas ligadas versus arrays

Parameter	Linked list	Array	Dynamic array
Indexing	O(n)	O(1)	O(1)
Insertion/deletion at beginning	O(1)	O(n), if array is not full (for shifting the elements)	O(n)
Insertion at ending	O(n)	O(1), if array is not full	O(1), if array is not full O(n), if array is full
Deletion at ending	O(n)	O(1)	O(n)
Insertion in middle	O(n)	O(n), if array is not full (for shifting the elements)	O(n)
Deletion in middle	O(n)	O(n), if array is not full (for shifting the elements)	O(n)
Wasted space	O(n)	0	O(n)

Lista Ligada Simples

Lista Ligada Simples

- É uma estrutura de dados que consiste de uma sequência de nós, começando do primeiro nó (head).

Lista Ligada Simples

- É uma estrutura de dados que consiste de uma sequência de nós, começando do primeiro nó (head).
- Cada nó armazena:

Lista Ligada Simples

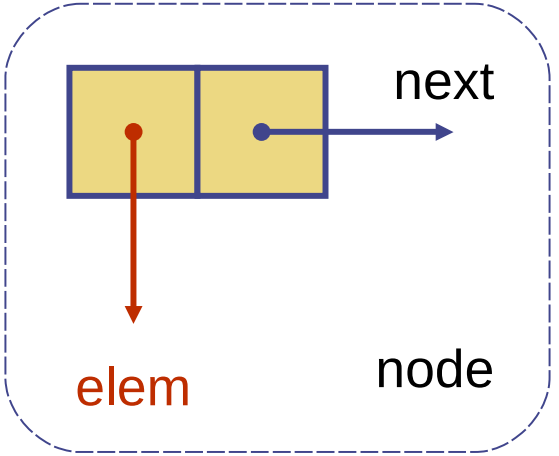
- É uma estrutura de dados que consiste de uma sequência de nós, começando do primeiro nó (head).
- Cada nó armazena:
  - O elemento (objeto).

Lista Ligada Simples

- É uma estrutura de dados que consiste de uma sequência de nós, começando do primeiro nó (head).
- Cada nó armazena:
  - O elemento (objeto).
  - Ligação (link) para o próximo nó.

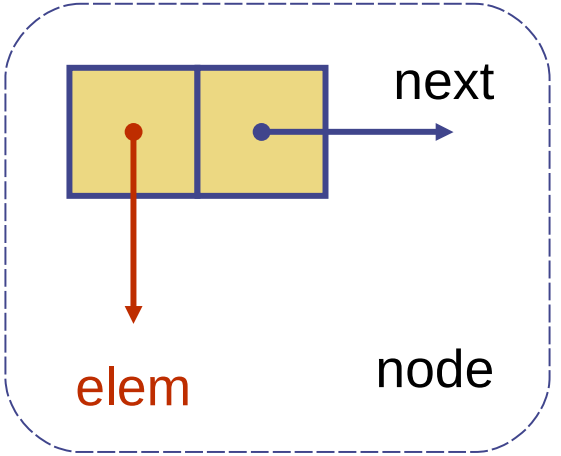
Lista Ligada Simples

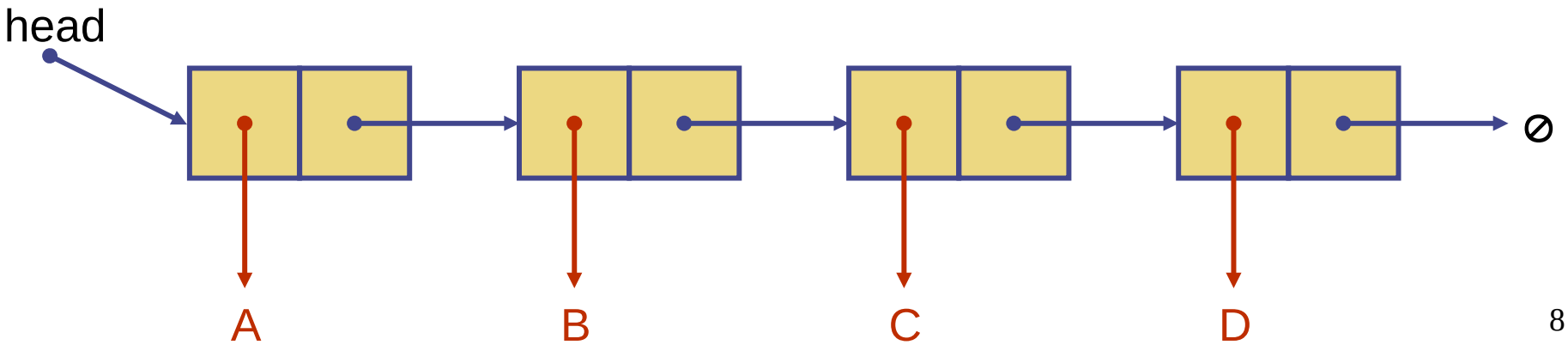
- É uma estrutura de dados que consiste de uma sequência de nós, começando do primeiro nó (head).
- Cada nó armazena:
  - O elemento (objeto).
  - Ligação (link) para o próximo nó.



Lista Ligada Simples

- É uma estrutura de dados que consiste de uma sequência de nós, começando do primeiro nó (head).
- Cada nó armazena:
  - O elemento (objeto).
  - Ligação (link) para o próximo nó.





Inserção de um novo elemento no início da lista ligada

Inserção de um novo elemento no início da lista ligada

- Passos

Inserção de um novo elemento no início da lista ligada

- Passos
  - Alocar um novo nó (elemento).

Inserção de um novo elemento no início da lista ligada

- Passos
  - Alocar um novo nó (elemento).
  - Inserir o novo elemento.

Inserção de um novo elemento no início da lista ligada

- Passos
  - Alocar um novo nó (elemento).
  - Inserir o novo elemento.
  - Novo nó deve apontar para o antigo nó-*head*.

Inserção de um novo elemento no início da lista ligada

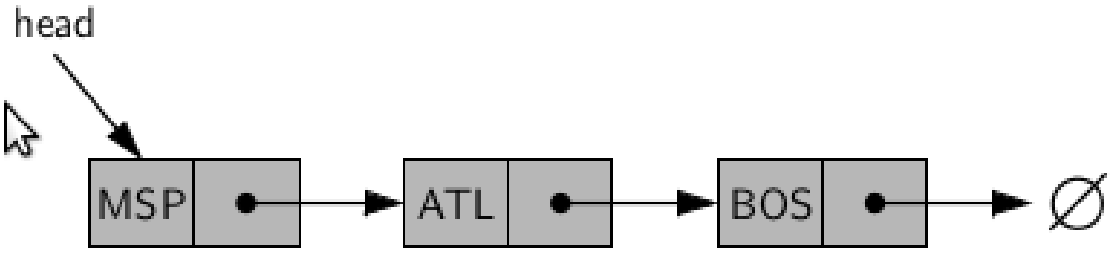
- Passos
  - Alocar um novo nó (elemento).
  - Inserir o novo elemento.
  - Novo nó deve apontar para o antigo nó-*head*.
  - *Head* deve apontar para o novo nó.

Inserção de um novo elemento no início da lista ligada

- Passos

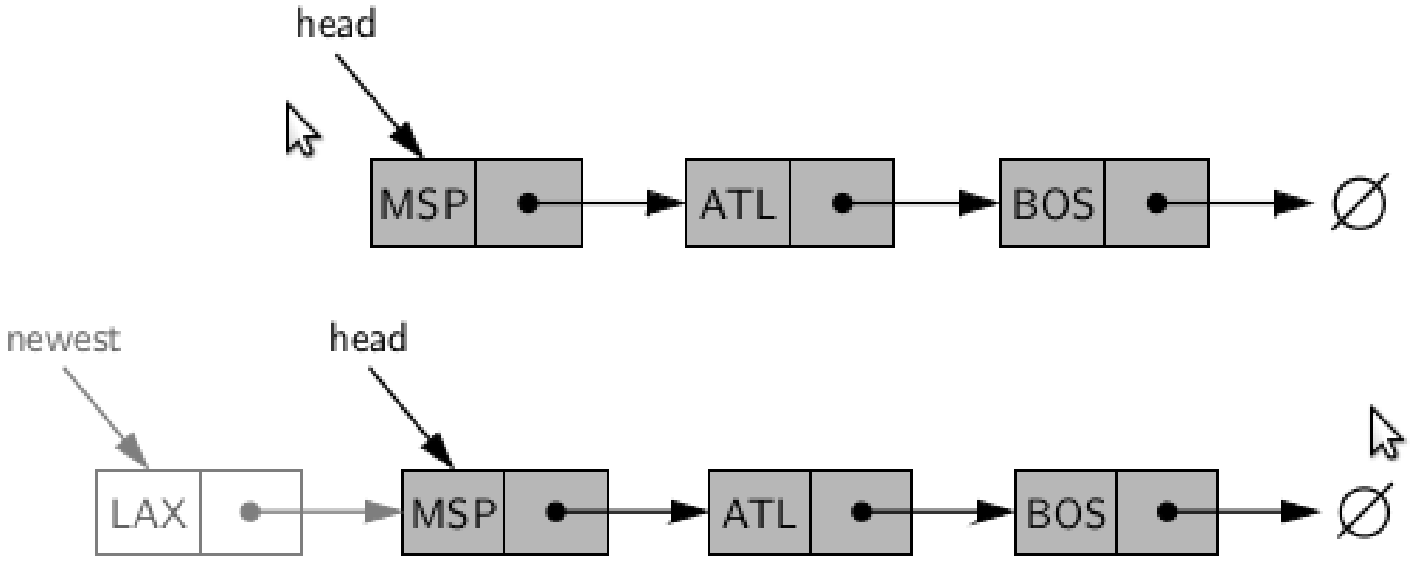
Inserção de um novo elemento no início da lista ligada

- Passos



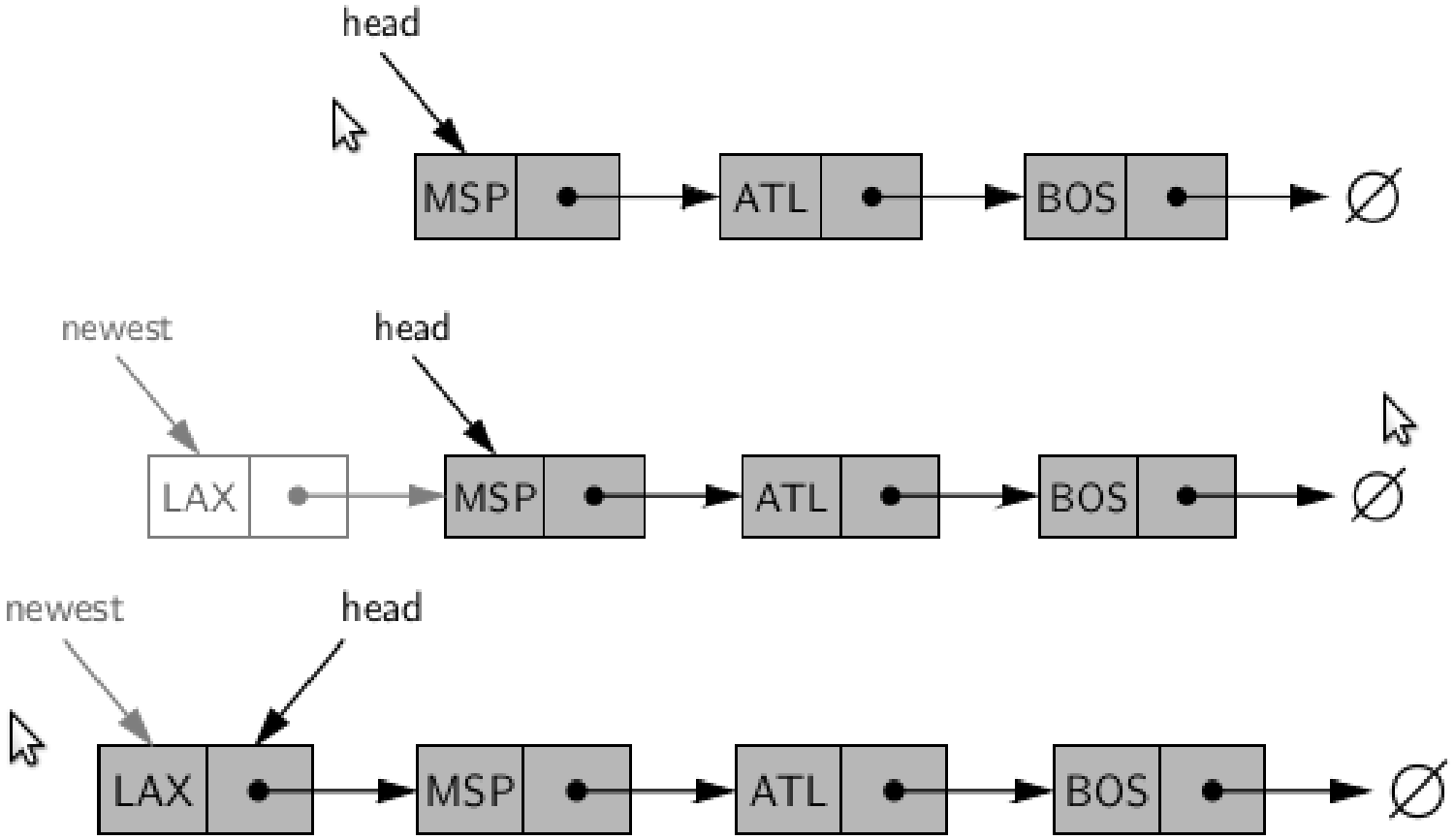
Inserção de um novo elemento no início da lista ligada

- Passos



Inserção de um novo elemento no início da lista ligada

- Passos



Inserção de um novo elemento no início da lista ligada

Algorithm add_first(L,e):

```
newest = Node(e)   {create new node instance storing reference to element e}
newest.next = L.head   {set new node's next to reference the old head node}
L.head = newest   {set variable head to reference the new node}
L.size = L.size + 1   {increment the node count}
```

Inserção de um novo elemento no fim da lista ligada

Inserção de um novo elemento no fim da lista ligada

- Passos:

Inserção de um novo elemento no fim da lista ligada

- Passos:
  - Alocar um novo nó (elemento).

Inserção de um novo elemento no fim da lista ligada

- Passos:
  - Alocar um novo nó (elemento).
  - Inserir o novo elemento.

Inserção de um novo elemento no fim da lista ligada

- Passos:
  - Alocar um novo nó (elemento).
  - Inserir o novo elemento.
  - O novo nó deve apontar para NULL.

Inserção de um novo elemento no fim da lista ligada

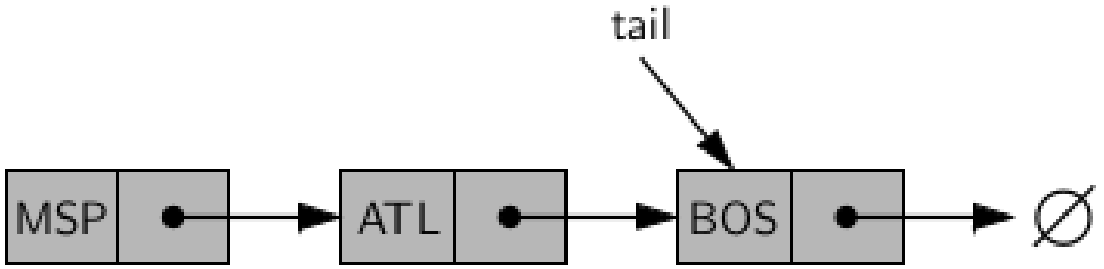
- Passos:
  - Alocar um novo nó (elemento).
  - Inserir o novo elemento.
  - O novo nó deve apontar para NULL.
  - O antigo último nó (tail) deve apontar para o novo elemento.

Inserção de um novo elemento no fim da lista ligada

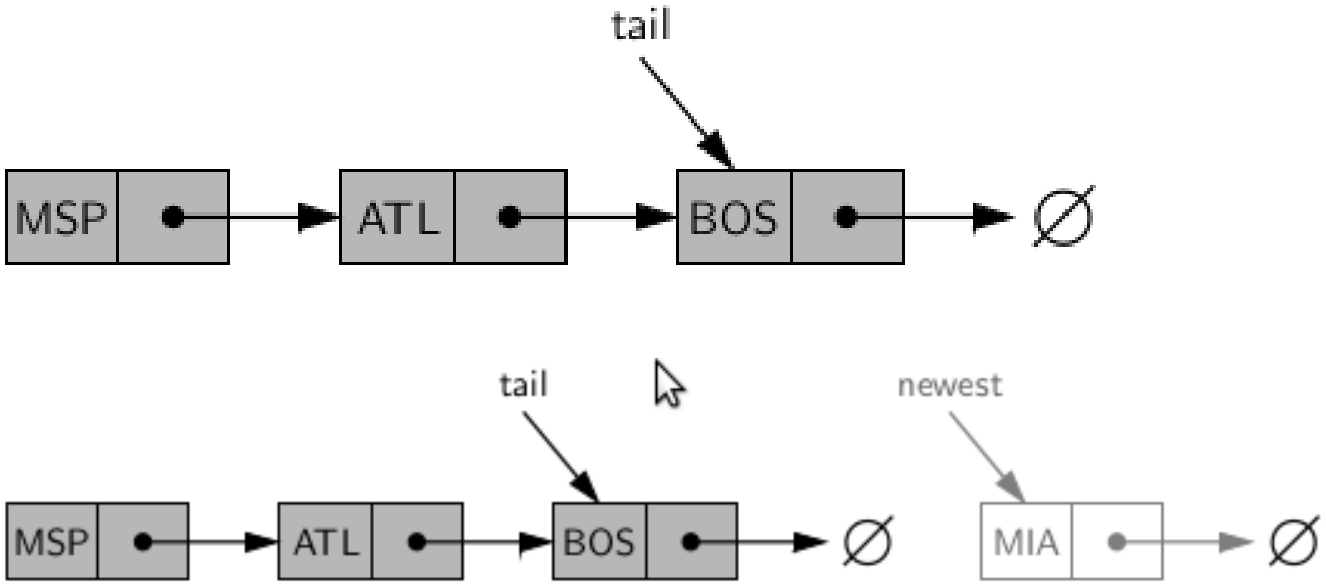
- Passos:
  - Alocar um novo nó (elemento).
  - Inserir o novo elemento.
  - O novo nó deve apontar para NULL.
  - O antigo último nó (tail) deve apontar para o novo elemento.
  - *Tail* deve apontar para o novo nó.

Inserção de um novo elemento no fim da lista ligada

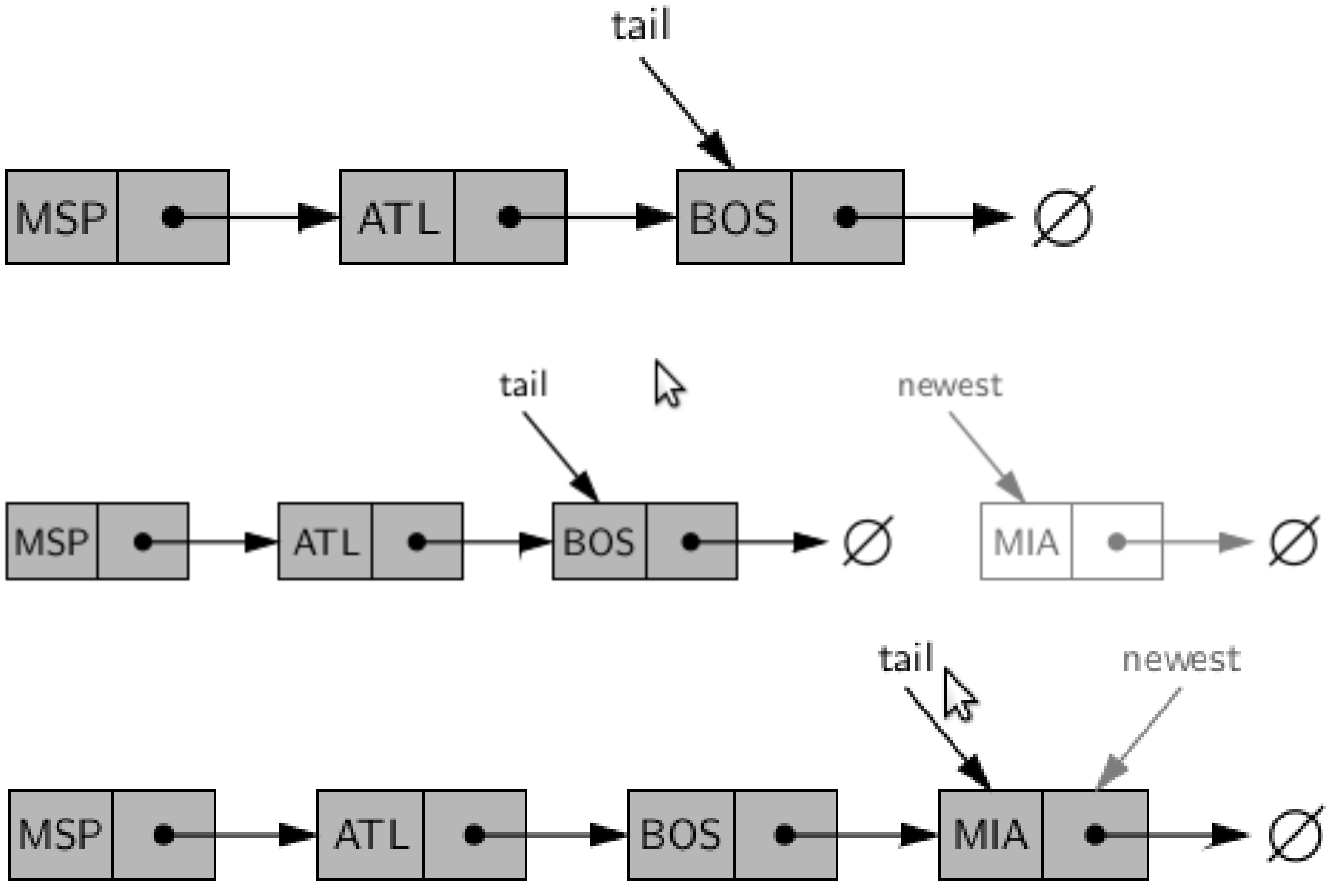
Inserção de um novo elemento no fim da lista ligada



Inserção de um novo elemento no fim da lista ligada



Inserção de um novo elemento no fim da lista ligada



Inserção de um novo elemento no fim da lista ligada

Algorithm add_last(L, e):

```
newest = Node(e)   {create new node instance storing reference to element e}
newest.next = None   {set new node's next to reference the None object}
L.tail.next = newest   {make old tail node point to new node}
L.tail = newest   {set variable tail to reference the new node}
L.size = L.size + 1   {increment the node count}
```

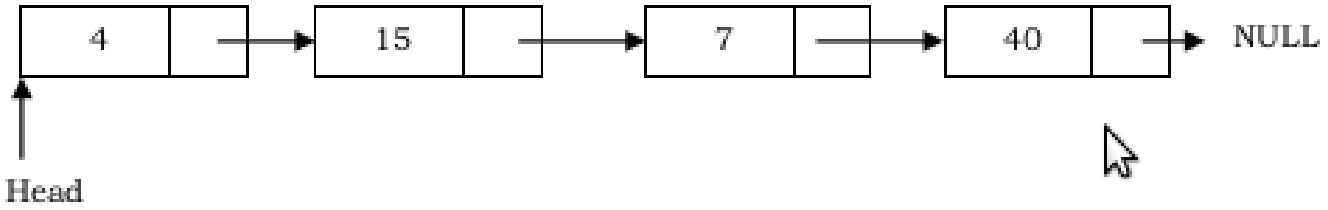
Inserção de um novo elemento no meio da lista ligada

Inserção de um novo elemento no meio da lista ligada

- Como inserir um novo elemento na 3ª posição?

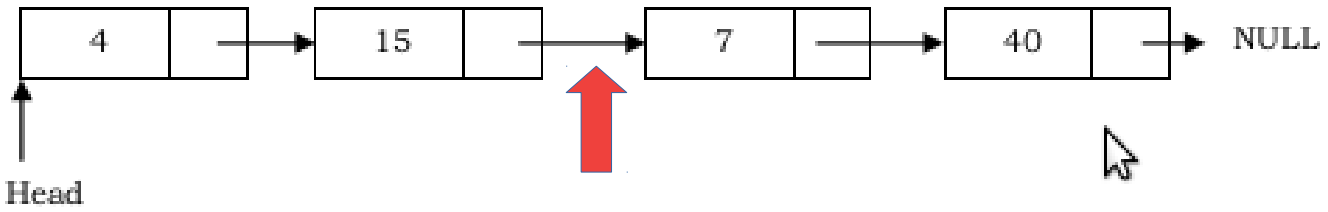
Inserção de um novo elemento no meio da lista ligada

- Como inserir um novo elemento na 3ª posição?



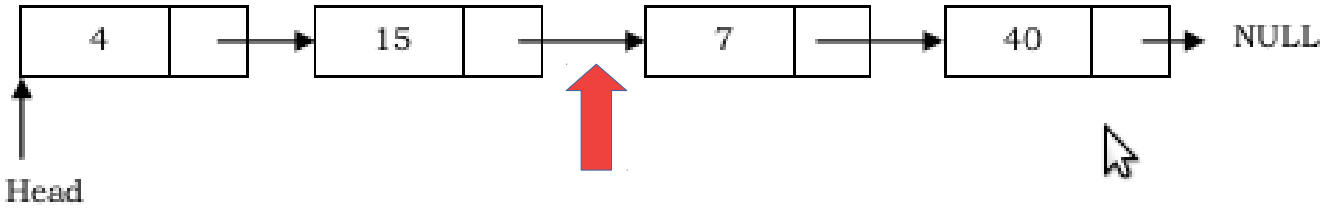
Inserção de um novo elemento no meio da lista ligada

- Como inserir um novo elemento na 3ª posição?



Inserção de um novo elemento no meio da lista ligada

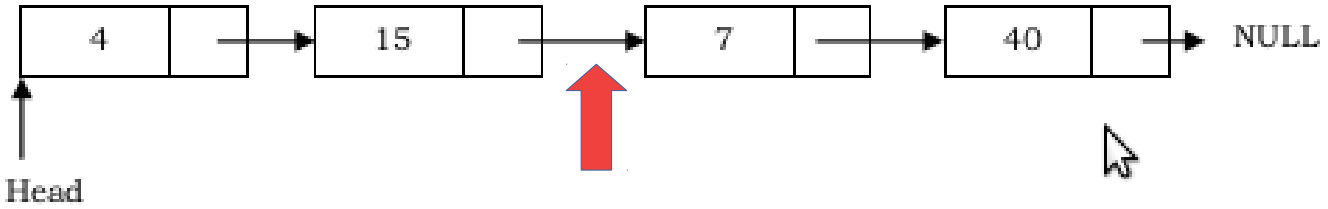
- Como inserir um novo elemento na 3ª posição?



  - Percorrer a lista até a posição desejada.

Inserção de um novo elemento no meio da lista ligada

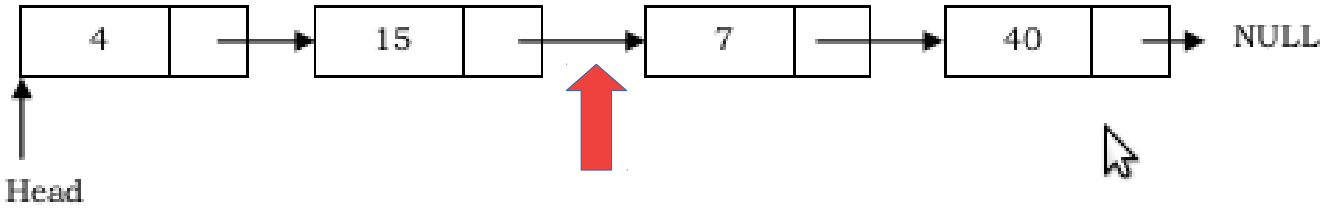
- Como inserir um novo elemento na 3ª posição?



  - Percorrer a lista até a posição desejada.
  - Criar um novo nó.

Inserção de um novo elemento no meio da lista ligada

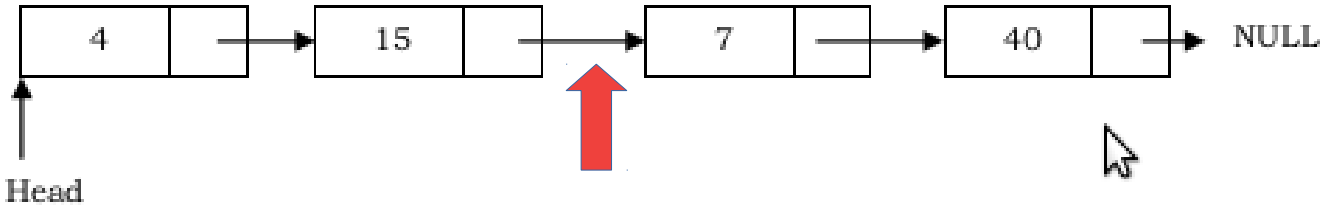
- Como inserir um novo elemento na 3ª posição?



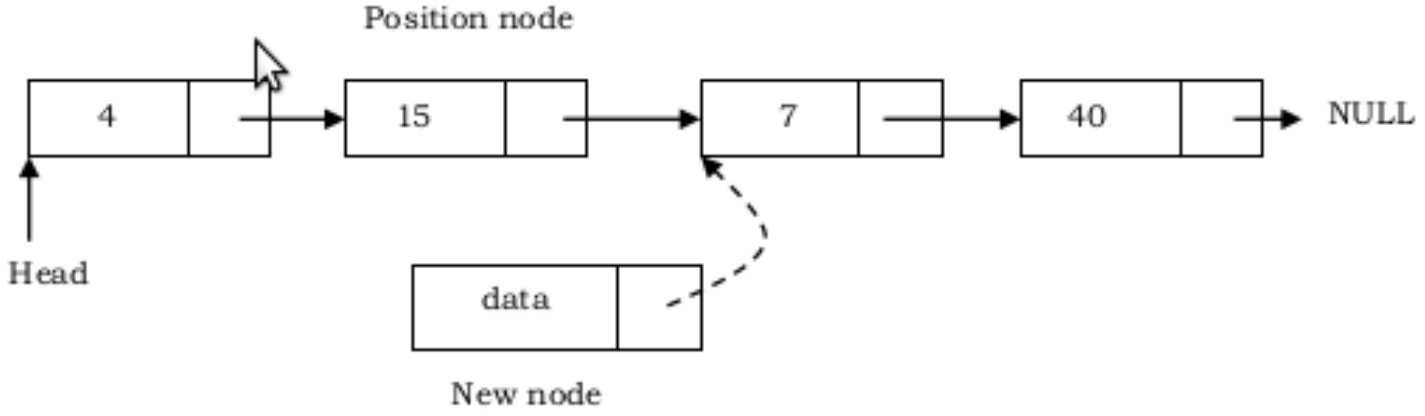
  - Percorrer a lista até a posição desejada.
  - Criar um novo nó.
  - O novo nó aponta para o próximo.

Inserção de um novo elemento no meio da lista ligada

- Como inserir um novo elemento na 3ª posição?

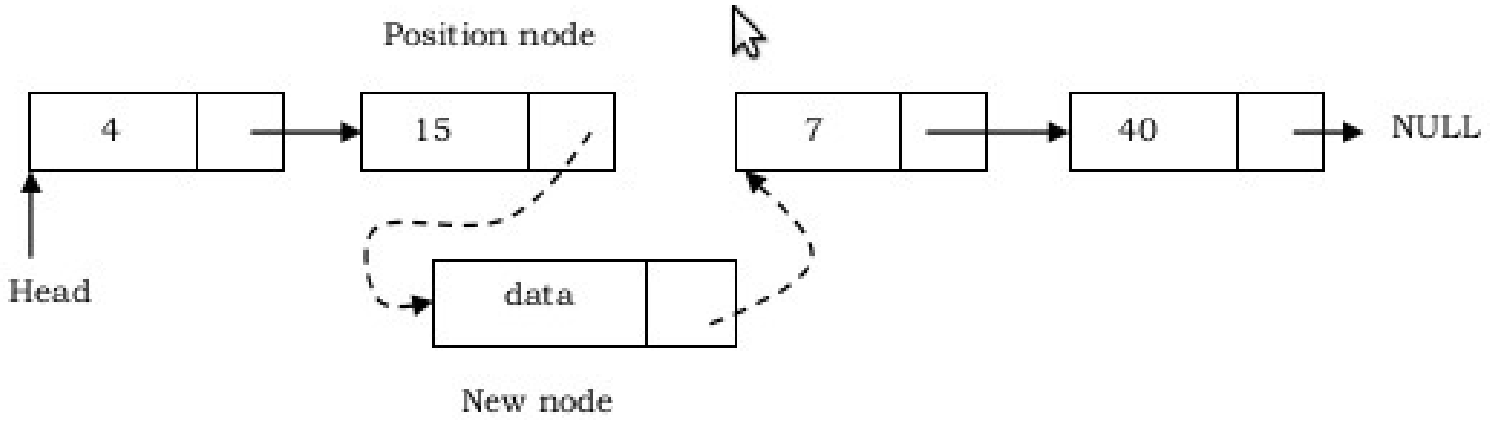


  - Percorrer a lista até a posição desejada.
  - Criar um novo nó.
  - O novo nó aponta para o próximo.



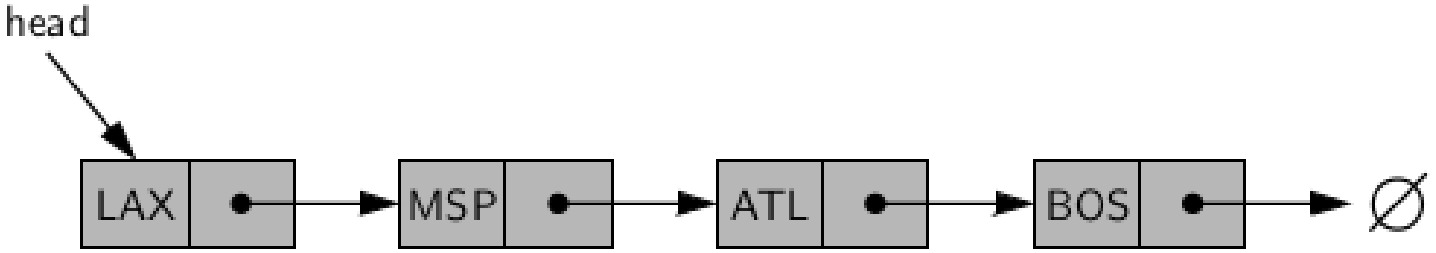
Inserção de um novo elemento no meio da lista ligada

- O position node aponta para o novo nó.

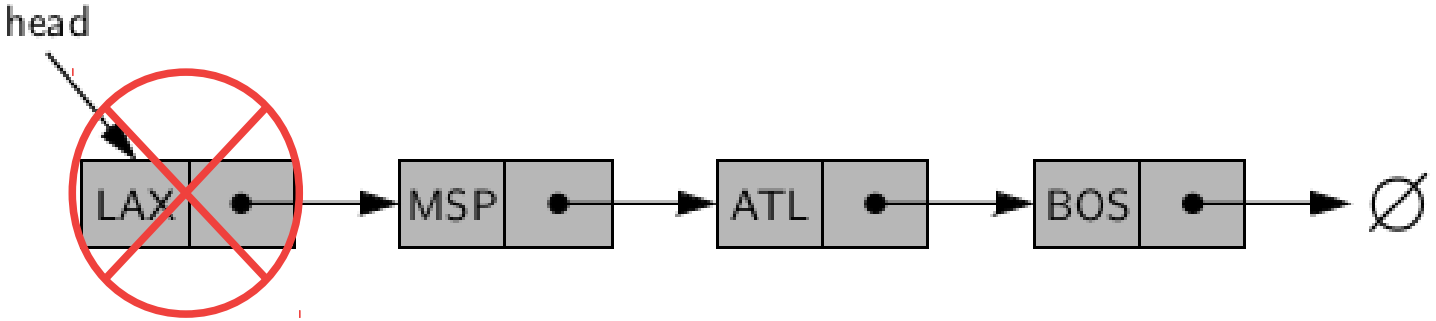


Remoção do primeiro elemento da lista ligada

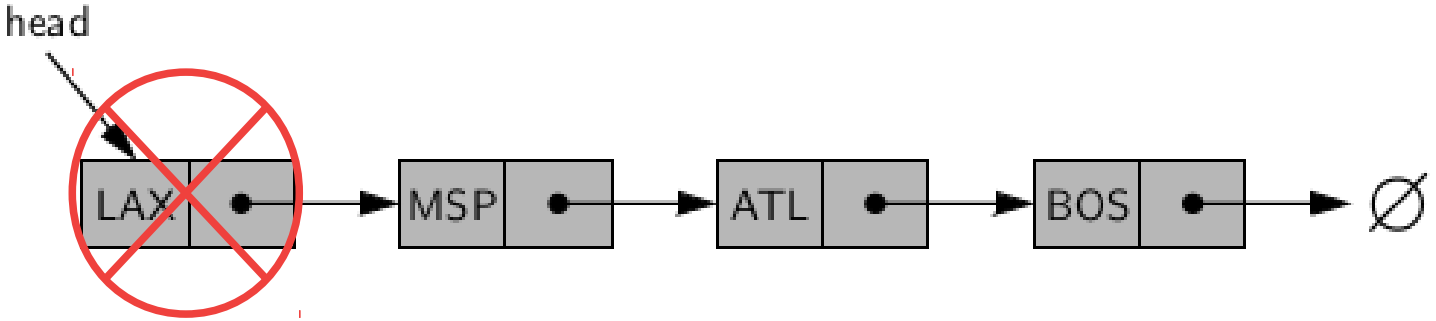
Remoção do primeiro elemento da lista ligada



Remoção do primeiro elemento da lista ligada

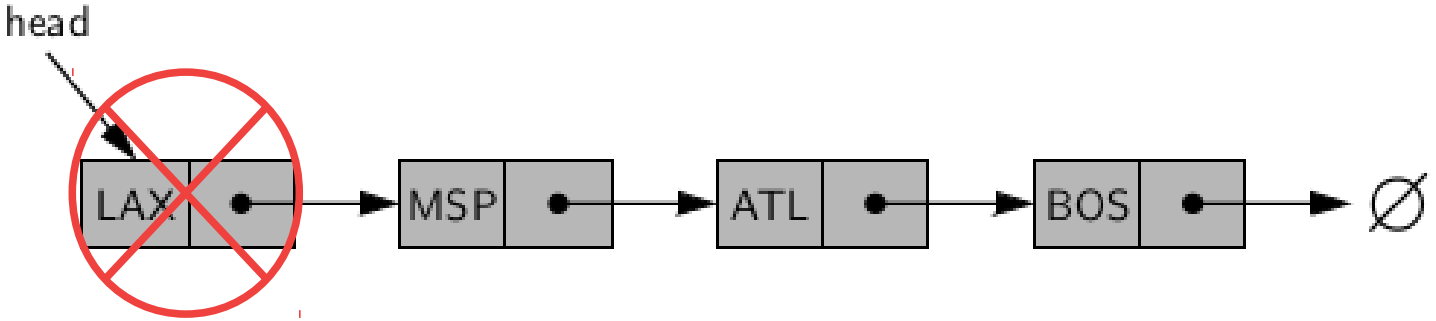


Remoção do primeiro elemento da lista ligada



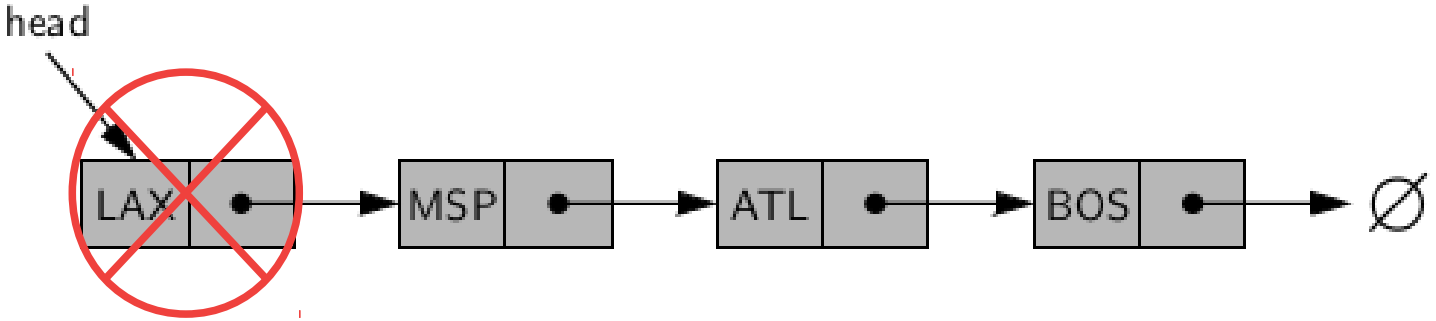
- Passos:

Remoção do primeiro elemento da lista ligada

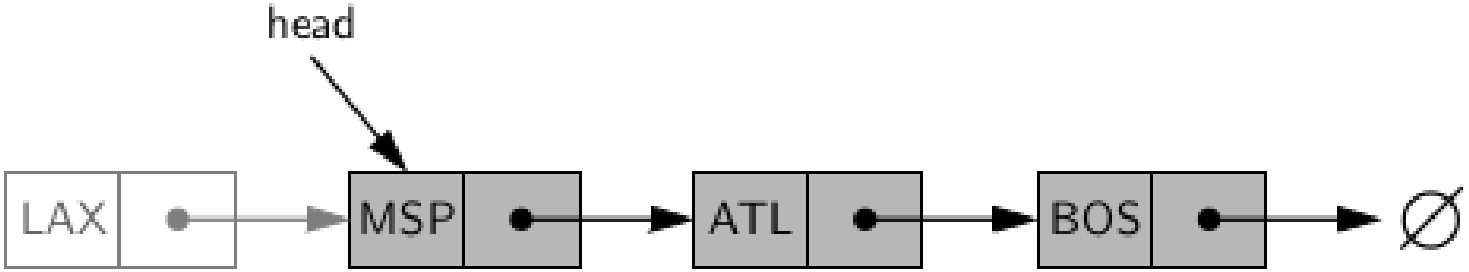


- Passos:
  - Atualize o *head* para apontar para o próximo nó da lista.

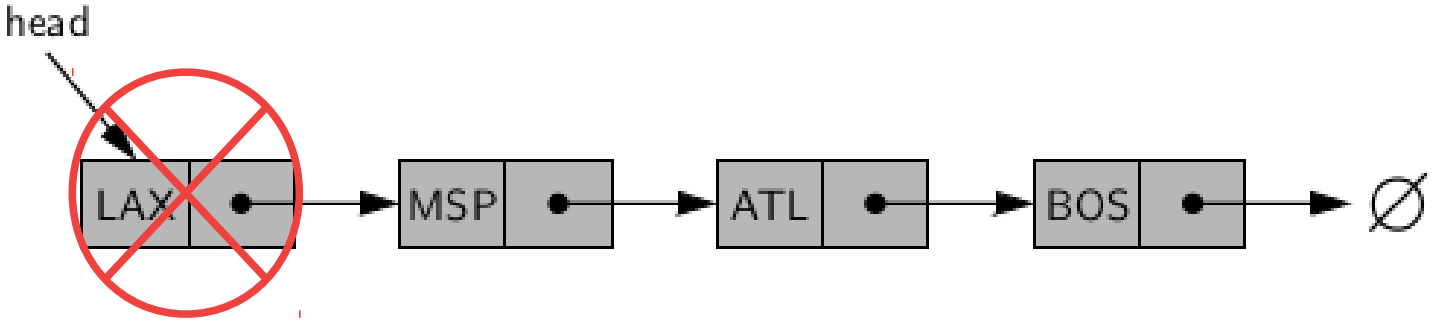
Remoção do primeiro elemento da lista ligada



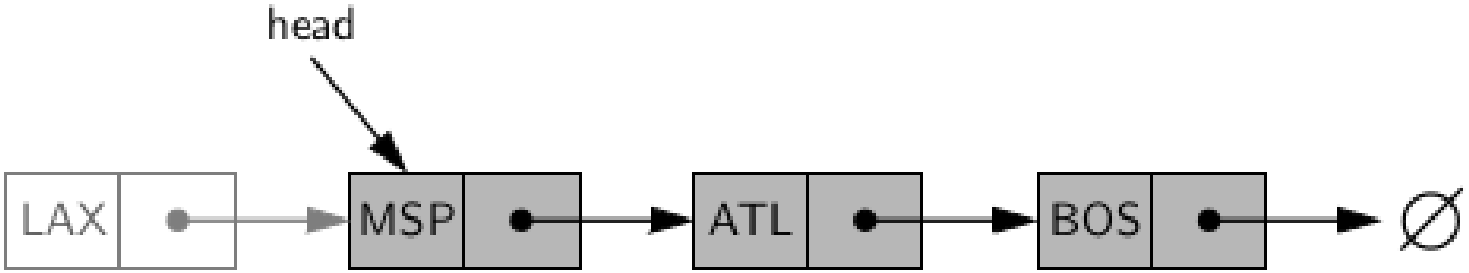
- Passos:
  - Atualize o *head* para apontar para o próximo nó da lista.



Remoção do primeiro elemento da lista ligada

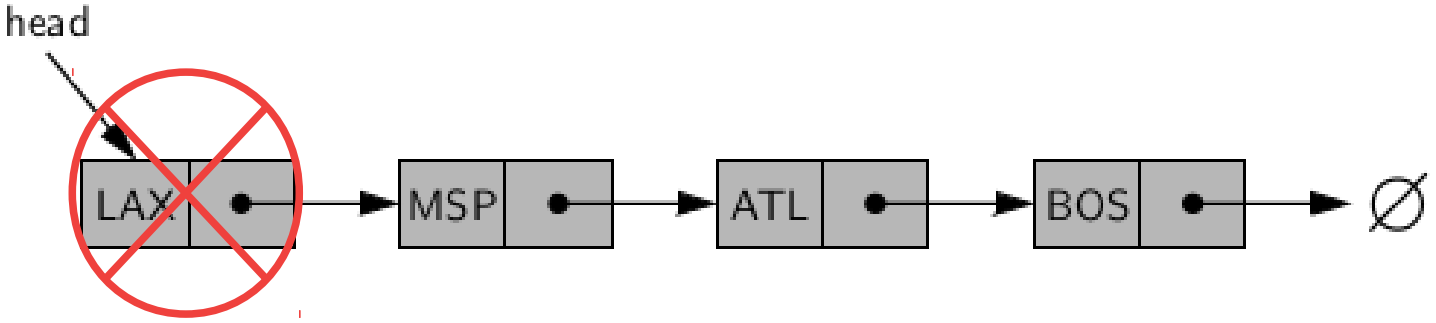


- Passos:
  - Atualize o *head* para apontar para o próximo nó da lista.

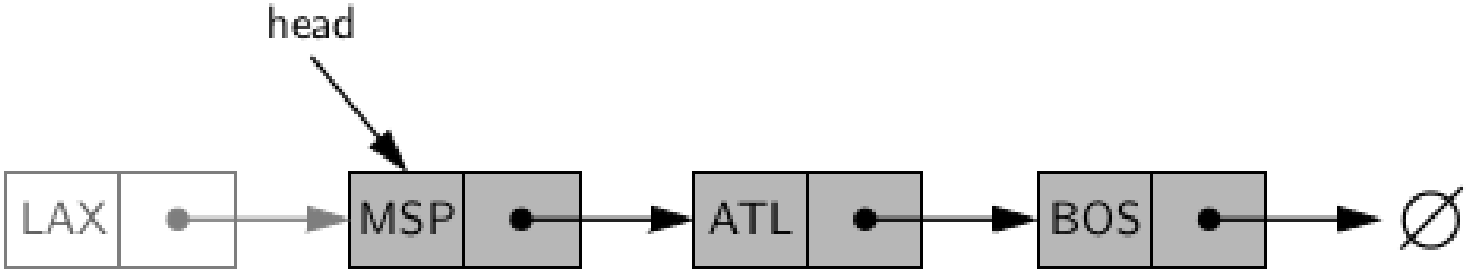


  - Permita que o *garbage collector* passe.

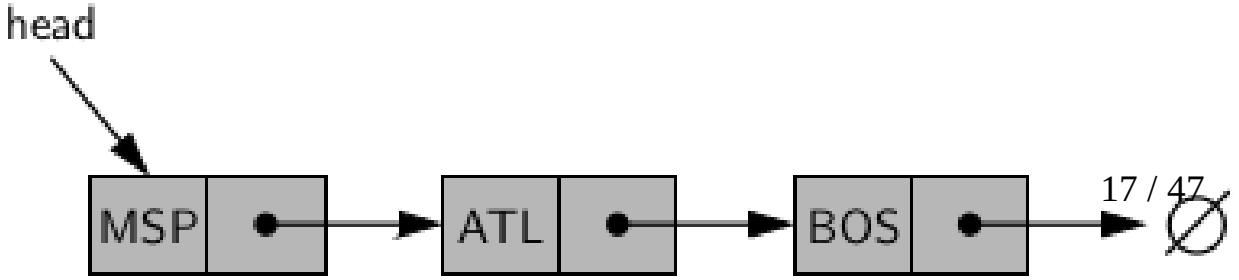
Remoção do primeiro elemento da lista ligada



- Passos:
  - Atualize o *head* para apontar para o próximo nó da lista.



  - Permita que o *garbage collector* passe.



Remoção do primeiro elemento da lista ligada

```
Algorithm remove_first(L):
    if L.head is None then
        Indicate an error: the list is empty.
    L.head = L.head.next          {make head point to next node (or None)}
    L.size = L.size − 1                                  {decrement the node count}
```

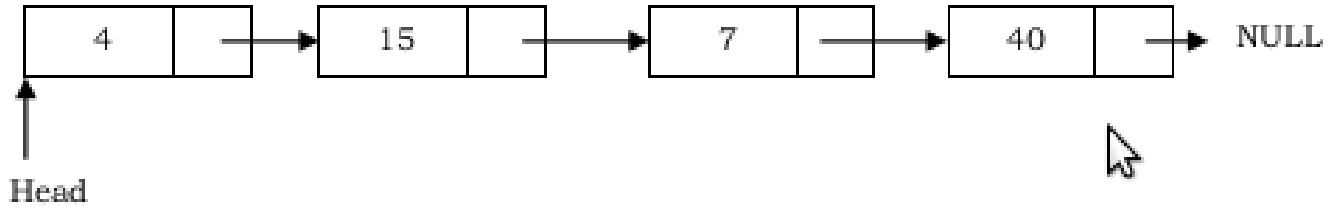
Remoção do último elemento da lista ligada

Remoção do último elemento da lista ligada

- Passos para remoção do último elemento:

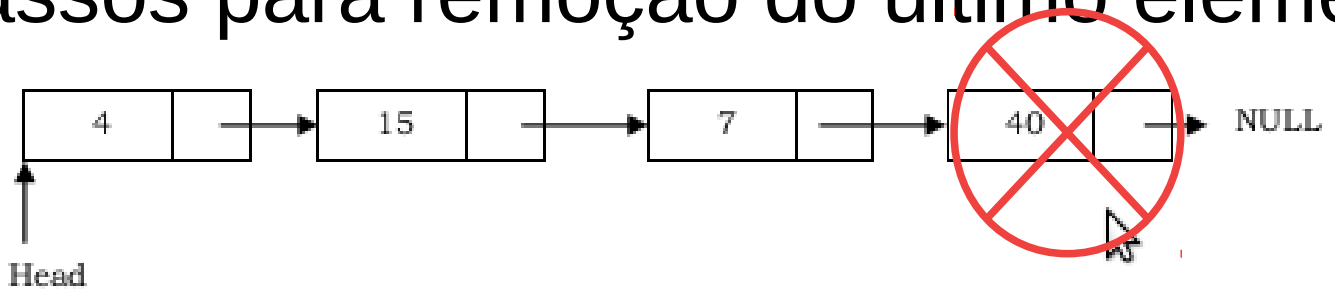
Remoção do último elemento da lista ligada

- Passos para remoção do último elemento:



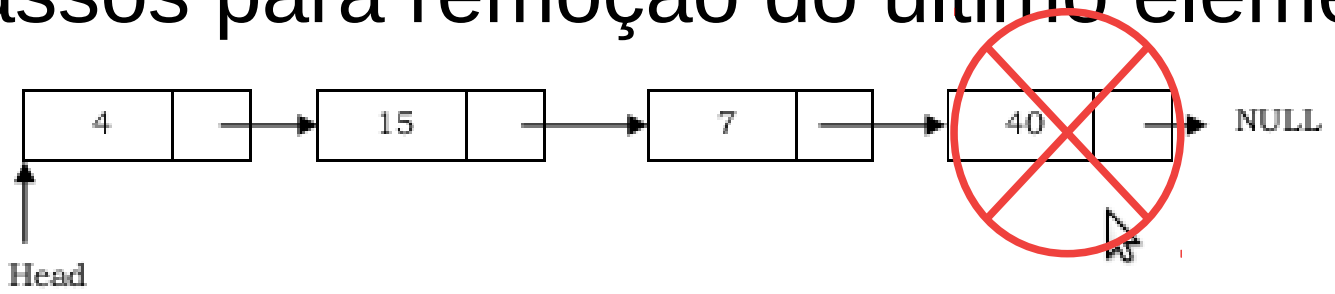
Remoção do último elemento da lista ligada

- Passos para remoção do último elemento:



Remoção do último elemento da lista ligada

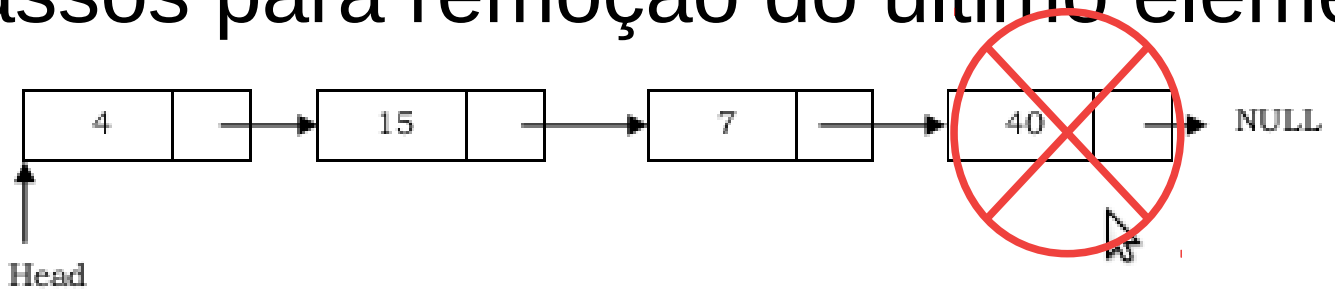
- Passos para remoção do último elemento:



  - Percorrer a lista e manter o endereço do nó anterior.

Remoção do último elemento da lista ligada

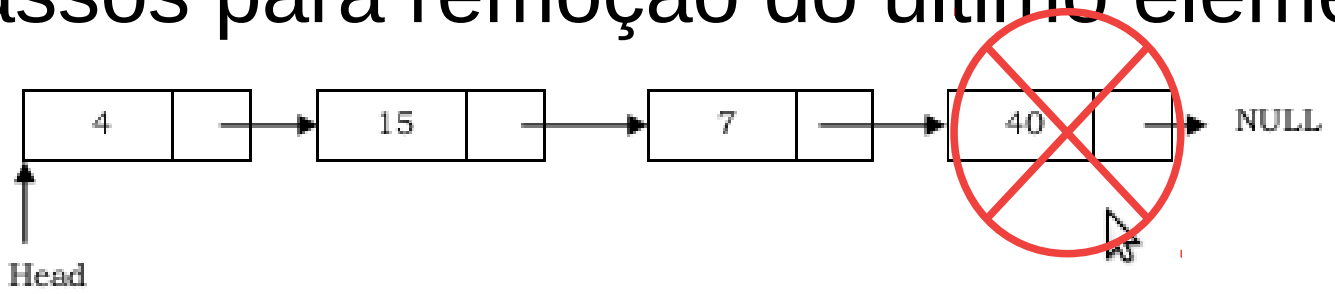
- Passos para remoção do último elemento:



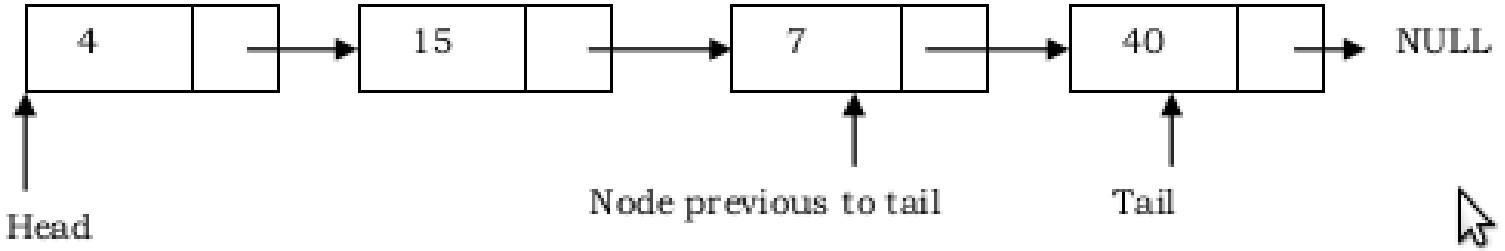
  - Percorrer a lista e manter o endereço do nó anterior.
  - Cada vez que alcançamos a lista, nós temos dois ponteiros: um apontando para o final da lista e outro apontando para o nó antes do final da lista.

Remoção do último elemento da lista ligada

- Passos para remoção do último elemento:



  - Percorrer a lista e manter o endereço do nó anterior.
  - Cada vez que alcançamos a lista, nós temos dois ponteiros: um apontando para o final da lista e outro apontando para o nó antes do final da lista.



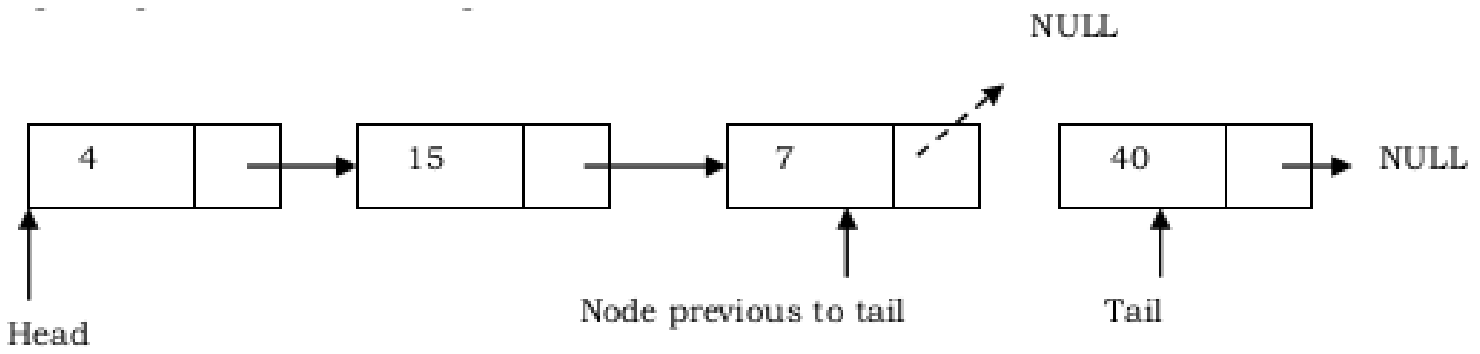
Remoção do último elemento da lista ligada

Remoção do último elemento da lista ligada

- O nó anterior ao último deverá apontar para NULL.

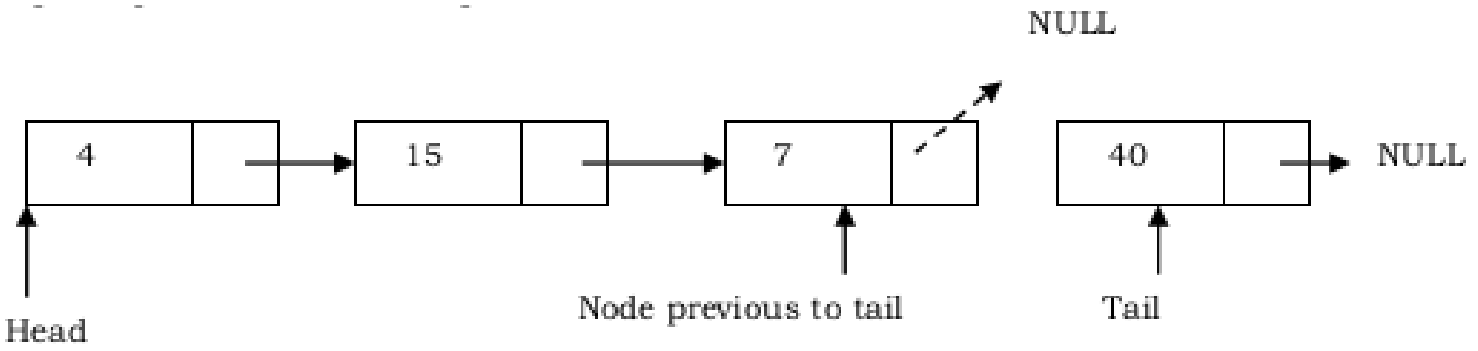
Remoção do último elemento da lista ligada

- O nó anterior ao último deverá apontar para NULL.



Remoção do último elemento da lista ligada

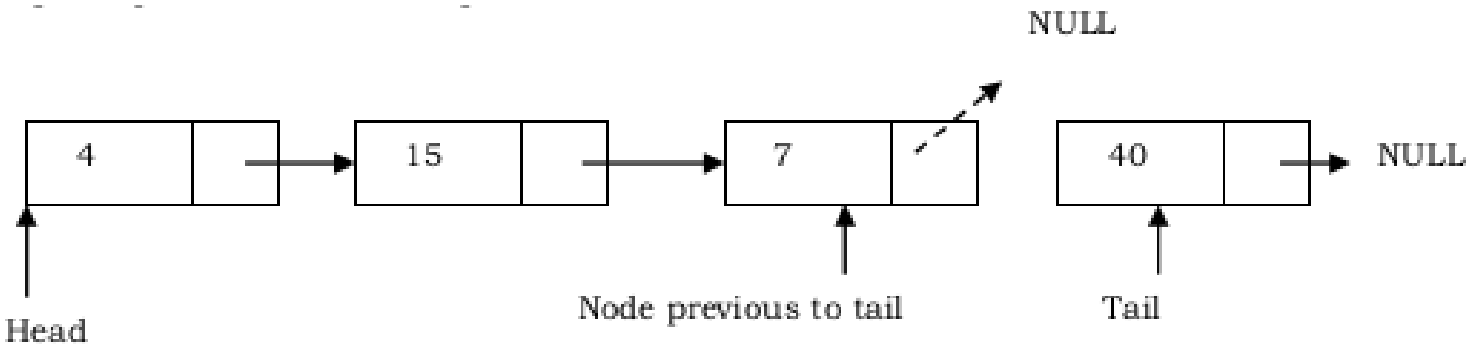
- O nó anterior ao último deverá apontar para NULL.



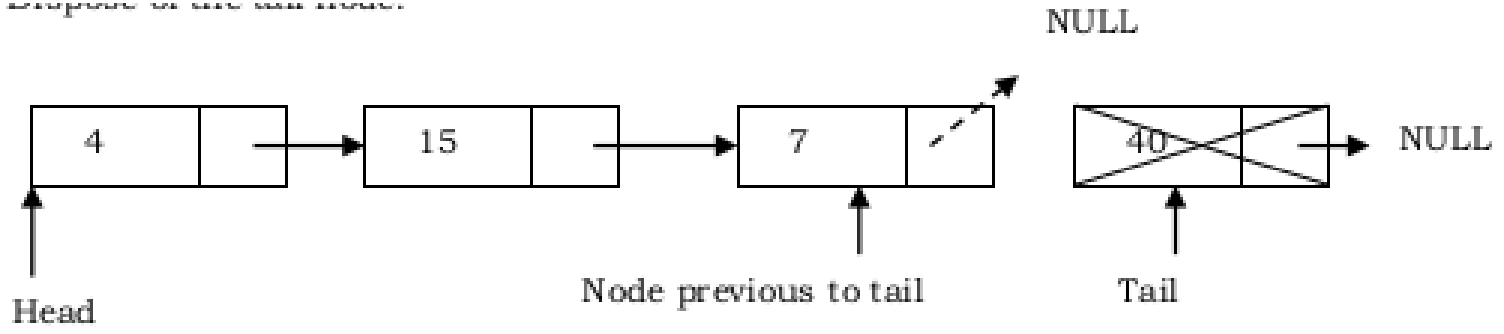
- O último nó será removido.

Remoção do último elemento da lista ligada

- O nó anterior ao último deverá apontar para NULL.



- O último nó será removido.



Remoção de elemento intermediário da lista ligada

Remoção de elemento intermediário da lista ligada

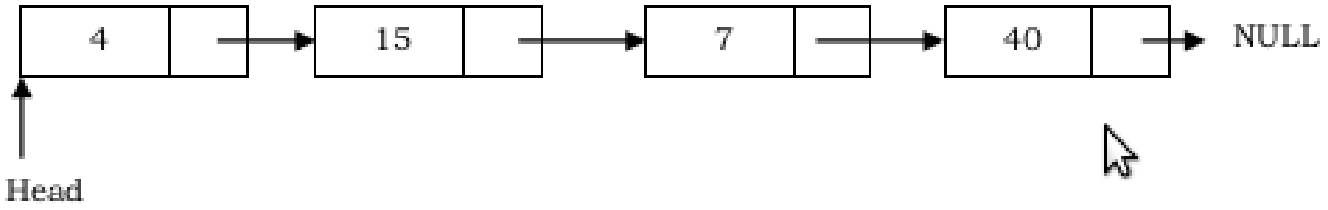
- Neste caso, um nó removido sempre estará entre dois outros nós.

Remoção de elemento intermediário da lista ligada

- Neste caso, um nó removido sempre estará entre dois outros nós.
- Os nós *head* e *tail* não serão atualizados.

Remoção de elemento intermediário da lista ligada

- Neste caso, um nó removido sempre estará entre dois outros nós.
- Os nós *head* e *tail* não serão atualizados.



Remoção de elemento intermediário da lista ligada

Remoção de elemento intermediário da lista ligada

- Passos:

Remoção de elemento intermediário da lista ligada

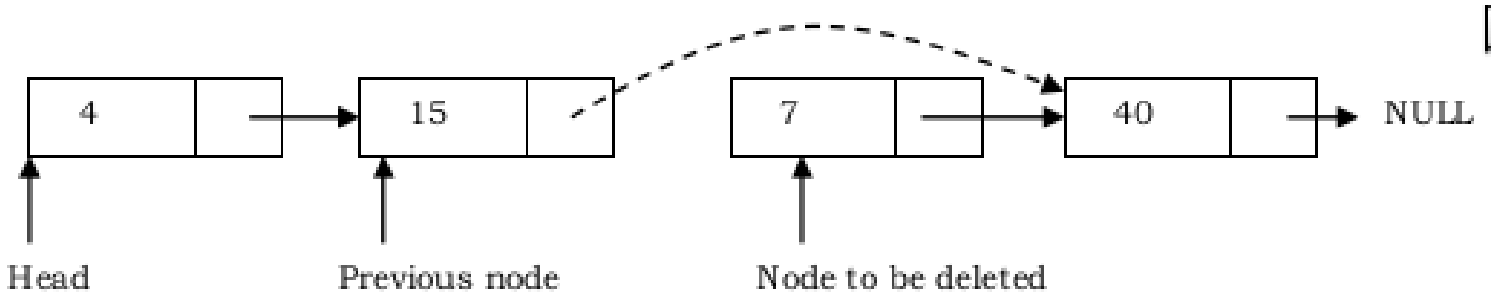
- Passos:
  - Deve-se manter o nó enquanto se percorre a lista.

Remoção de elemento intermediário da lista ligada

- Passos:
  - Deve-se manter o nó enquanto se percorre a lista.
  - Uma vez que o nó que será removido foi encontrado, o nó anterior (*previous node*) deverá apontar para o próximo do nó que será removido.

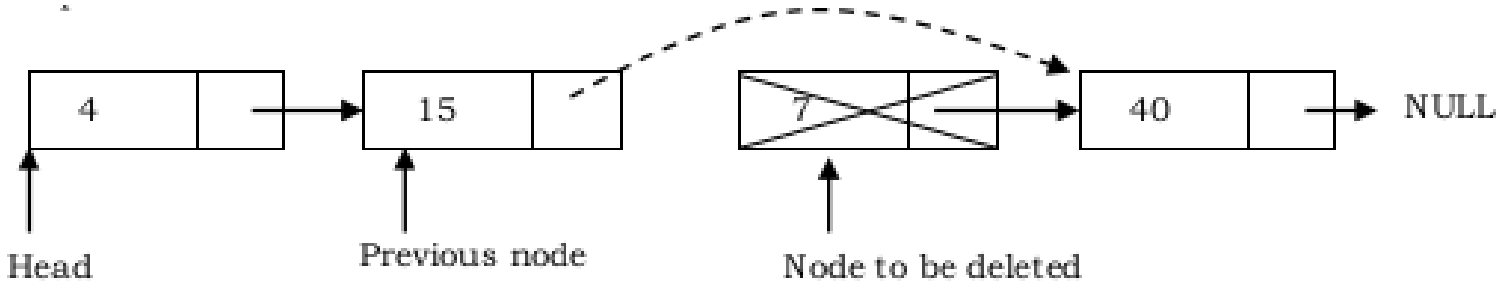
Remoção de elemento intermediário da lista ligada

- Passos:
  - Deve-se manter o nó enquanto se percorre a lista.
  - Uma vez que o nó que será removido foi encontrado, o nó anterior (*previous node*) deverá apontar para o próximo do nó que será removido.



Remoção de elemento intermediário da lista ligada

- Passos:
  - O nó atual será apagado.



Apagar uma lista

Apagar uma lista

- Como apagar uma lista?

Apagar uma lista

- Como apagar uma lista?
  - Python usa garbage-collection.

Apagar uma lista

- Como apagar uma lista?
  - Python usa garbage-collection.
  - Portanto:

```
def clear( self ) :
    self.head = None
```

Apagar uma lista

- Como apagar uma lista?
  - Python usa garbage-collection.
  - Portanto:

```
def clear( self ) :
    self.head = None
```

  - Tempo O(1) e espaço O(1).

Python Garbage Collection

Python Garbage Collection

- Em Python, o método de alocar e desalocar memória é **automático**.

Python Garbage Collection

- Em Python, o método de alocar e desalocar memória é **automático**.
- O programador não precisa alocar e nem desalocar memória usando alocação dinâmica de memória como nas linguagens C e C++.

Python Garbage Collection

- Em Python, o método de alocar e desalocar memória é **automático**.
- O programador não precisa alocar e nem desalocar memória usando alocação dinâmica de memória como nas linguagens C e C++.
- Python adota duas estratégias:

Python Garbage Collection

- Em Python, o método de alocar e desalocar memória é **automático**.
- O programador não precisa alocar e nem desalocar memória usando alocação dinâmica de memória como nas linguagens C e C++.
- Python adota duas estratégias:
  - Contagem de referências.

Python Garbage Collection

- Em Python, o método de alocar e desalocar memória é **automático**.
- O programador não precisa alocar e nem desalocar memória usando alocação dinâmica de memória como nas linguagens C e C++.
- Python adota duas estratégias:
  - Contagem de referências.
  - Coletor de lixo (*garbage collection*).

Python Garbage Collection

Python Garbage Collection

- Contagem de referência

Python Garbage Collection

- Contagem de referência
  - Era o único método antes do Python 2.0.

Python Garbage Collection

- Contagem de referência
  - Era o único método antes do Python 2.0.
  - Conta o número de referências a um objeto no sistema.

Python Garbage Collection

- Contagem de referência
  - Era o único método antes do Python 2.0.
  - Conta o número de referências a um objeto no sistema.
  - Quando as referências a um objeto são removidas, a quantidade de referências por um objeto é decrementada.

Python Garbage Collection

- Contagem de referência
  - Era o único método antes do Python 2.0.
  - Conta o número de referências a um objeto no sistema.
  - Quando as referências a um objeto são removidas, a quantidade de referências por um objeto é decrementada.
  - Quando o total de referências se torna igual a zero, o objeto é desalocado.

Python Garbage Collection

Python Garbage Collection

- Contagem de referência
  - Exemplo:

Python Garbage Collection

- Contagem de referência
  - Exemplo:

```
# Literal 9 is an object
b = 9

# Reference count of object 9
# becomes 0.
b = 4

```

Python Garbage Collection

- Contagem de referência
  - Exemplo:

```
1 # Literal 9 is an object
2 b = 9
3
4 # Reference count of object 9
5 # becomes 0.
6 b = 4
7
```

  - O valor literal 9 é um objeto (linha 2).

Python Garbage Collection

- Contagem de referência
  - Exemplo:

```
1 # Literal 9 is an object
2 b = 9
3
4 # Reference count of object 9
5 # becomes 0.
6 b = 4
7
```

  - O valor literal 9 é um objeto (linha 2).
  - A quantidade de referências ao objeto 9 é incrementada de 1 na linha 2.

Python Garbage Collection

- Contagem de referência
  - Exemplo:

```
1 # Literal 9 is an object
2 b = 9
3
4 # Reference count of object 9
5 # becomes 0.
6 b = 4
7
```

  - O valor literal 9 é um objeto (linha 2).
  - A quantidade de referências ao objeto 9 é incrementada de 1 na linha 2.
  - Na linha 6, a quantidade de referências ao objeto 9 se torna zero e, portanto, o objeto 9 é desalocado.

Python Garbage Collection

- Contagem de referência
  - Exemplo:

```
1 # Literal 9 is an object
2 b = 9
3
4 # Reference count of object 9
5 # becomes 0.
6 b = 4
7
```

  - O valor literal 9 é um objeto (linha 2).
  - A quantidade de referências ao objeto 9 é incrementada de 1 na linha 2.
  - Na linha 6, a quantidade de referências ao objeto 9 se torna zero e, portanto, o objeto 9 é desalocado.
  - O *garbage collector* desaloca o objeto.

Python Garbage Collection

Python Garbage Collection

- Automático Garbage Collection

Python Garbage Collection

- Automático Garbage Collection
  - Garbage collection é uma atividade agendada (realizada em ciclos).

Python Garbage Collection

- Automático Garbage Collection
  - Garbage collection é uma atividade agendada (realizada em ciclos).
  - O agendamento é baseado em limiar (*threshold*) de alocações e desalocações de objetos.

Python Garbage Collection

- Automático Garbage Collection
  - Garbage collection é uma atividade agendada (realizada em ciclos).
  - O agendamento é baseado em limiar (*threshold*) de alocações e desalocações de objetos.
  - Quando o número de alocações e desalocações é maior que esse limiar, então o *garbage collector* é executado.

Python Garbage Collection

Python Garbage Collection

- Automático Garbage Collection

Python Garbage Collection

- Automático Garbage Collection
  - É possível verificar esse limiar para novos objetos através do pacote **gc**.

Python Garbage Collection

- Automático Garbage Collection
  - É possível verificar esse limiar para novos objetos através do pacote **gc**.

```
# loading gc
import gc

# get the current collection
# thresholds as a tuple
print("Garbage collection thresholds:",
                      gc.get_threshold())
```

Python Garbage Collection

- Automático Garbage Collection
  - É possível verificar esse limiar para novos objetos através do pacote **gc**.

```
# loading gc
import gc

# get the current collection
# thresholds as a tuple
print("Garbage collection thresholds:",
                      gc.get_threshold())
```

```
('Garbage collection thresholds:', (700, 10, 10))
```

Python Garbage Collection

Python Garbage Collection

- Manual Garbage Collection

Python Garbage Collection

- Manual Garbage Collection
  - É possível também solicitar o GC quando houver muita manipulação de memória.

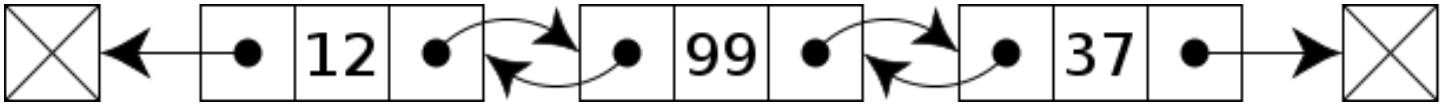
Python Garbage Collection

- Manual Garbage Collection
  - É possível também solicitar o GC quando houver muita manipulação de memória.

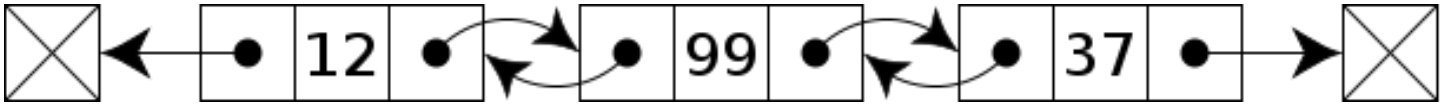
```
# Importing gc module
import gc

# Returns the number of
# objects it has collected
# and deallocated
collected = gc.collect()

# Prints Garbage collector
# as 0 object
print("Garbage collector: collected",
          "%d objects." % collected)
```

Lista Duplamente Ligada

Lista Duplamente Ligada

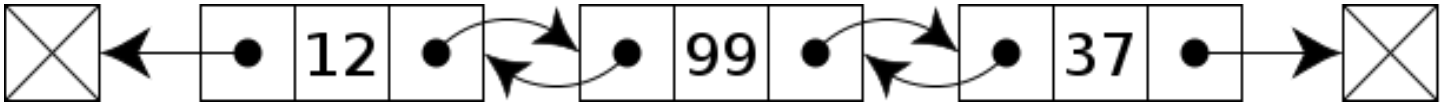


Lista Duplamente Ligada



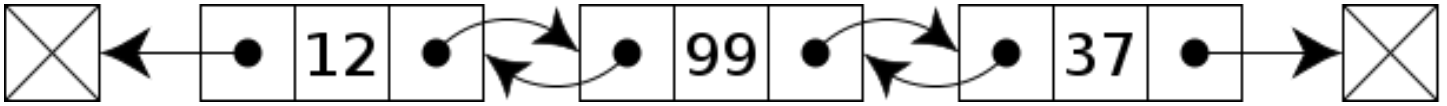
- A principal vantagem da lista duplamente ligada é a possibilidade de navegar em ambas as direções da lista.

Lista Duplamente Ligada



- A principal vantagem da lista duplamente ligada é a possibilidade de navegar em ambas as direções da lista.
- Para se remover um nó em uma lista ligada simples, é necessário ter o ponteiro (endereço) do nó anterior.

Lista Duplamente Ligada



- A principal vantagem da lista duplamente ligada é a possibilidade de navegar em ambas as direções da lista.
- Para se remover um nó em uma lista ligada simples, é necessário ter o ponteiro (endereço) do nó anterior.
- Em uma lista duplamente ligada não é necessário ter o endereço do nó anterior.

Lista Duplamente Ligada

Lista Duplamente Ligada

- Desvantagens deste tipo de lista:

Lista Duplamente Ligada

- Desvantagens deste tipo de lista:
  - Cada nó necessita de um ponteiro extra (mais espaço).

Lista Duplamente Ligada

- Desvantagens deste tipo de lista:
  - Cada nó necessita de um ponteiro extra (mais espaço).
  - As operações de inserção e remoção de um nó podem ser um pouco mais demoradas.

Inserção em Lista Duplamente Ligada

Inserção em Lista Duplamente Ligada

- Existem três casos:

Inserção em Lista Duplamente Ligada

- Existem três casos:
  - Novo nó antes do *head*.

Inserção em Lista Duplamente Ligada

- Existem três casos:
  - Novo nó antes do *head*.
  - Novo nó depois do *tail*.

Inserção em Lista Duplamente Ligada

- Existem três casos:
  - Novo nó antes do *head*.
  - Novo nó depois do *tail*.
  - Novo nó no meio da lista.

Inserção em Lista Duplamente Ligada

Inserção em Lista Duplamente Ligada

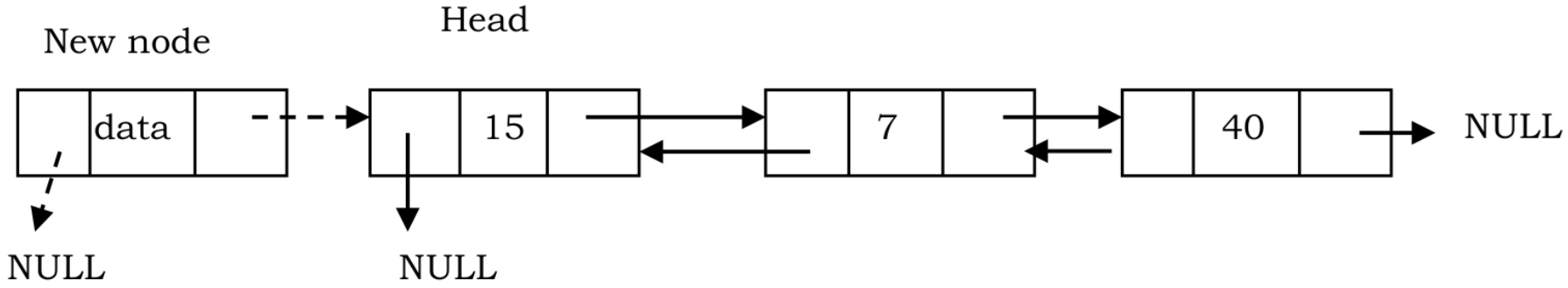
- **Início** da lista (2 passos):

Inserção em Lista Duplamente Ligada

- **Início** da lista (2 passos):
  - O novo nó deve apontar (*right*) para o nó head e também deve apontar (*left*) para NULL.

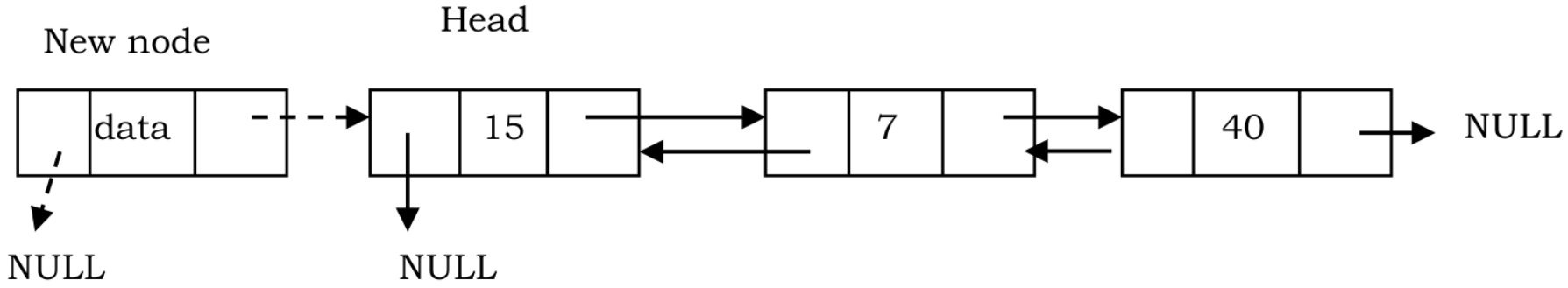
Inserção em Lista Duplamente Ligada

- **<u>Início</u>** da lista (2 passos):
  - O novo nó deve apontar (*right*) para o nó head e também deve apontar (*left*) para NULL.



Inserção em Lista Duplamente Ligada

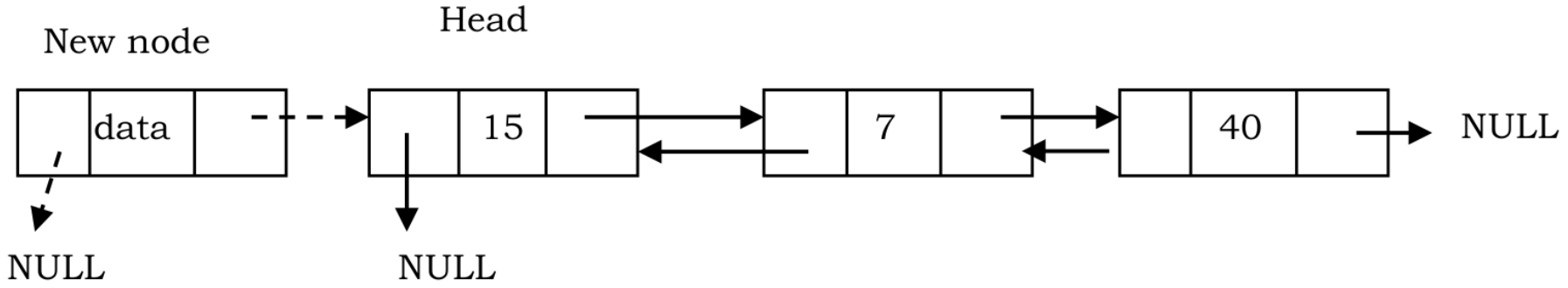
- **<u>Início</u>** da lista (2 passos):
  - O novo nó deve apontar (*right*) para o nó head e também deve apontar (*left*) para NULL.



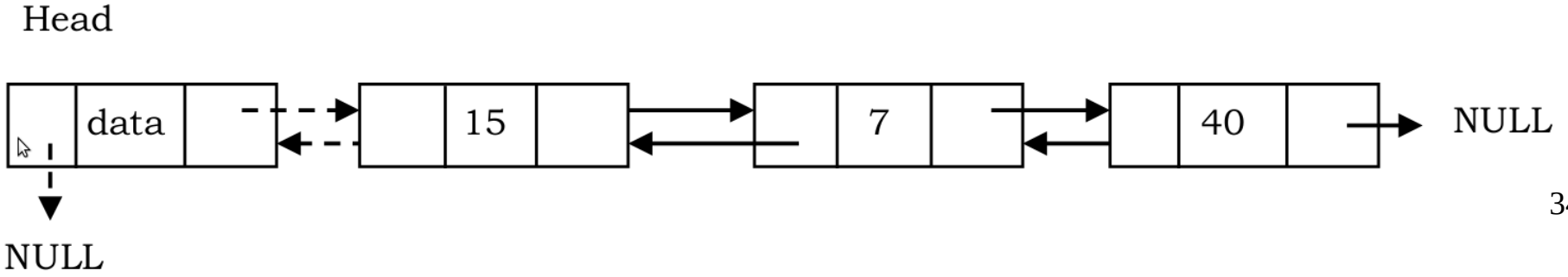
  - O ponteiro *left* do nó *head* deve apontar para o novo nó e este novo nó deve ser o *head* agora.

Inserção em Lista Duplamente Ligada

- **<u>Início</u>** da lista (2 passos):
  - O novo nó deve apontar (*right*) para o nó head e também deve apontar (*left*) para NULL.



  - O ponteiro *left* do nó *head* deve apontar para o novo nó e este novo nó deve ser o *head* agora.



Inserção em Lista Duplamente Ligada

Inserção em Lista Duplamente Ligada

- **Fim** da lista:

Inserção em Lista Duplamente Ligada

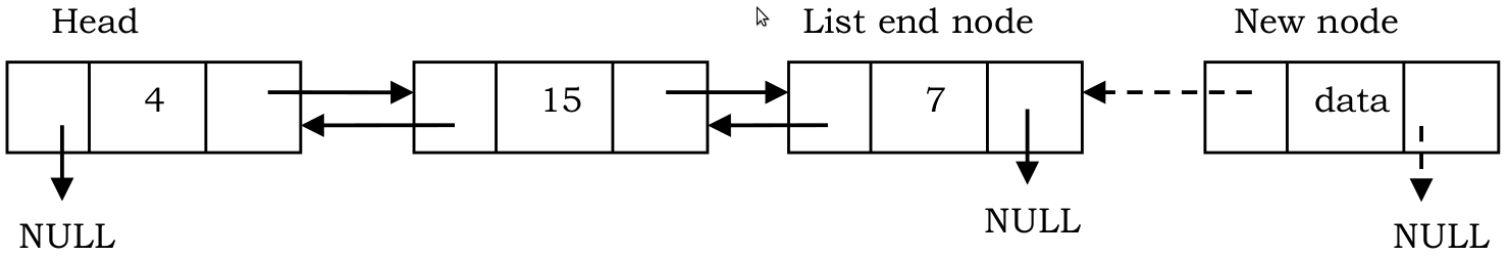
- **Fim** da lista:
  - Percorrer toda a lista e inserir ao final.

Inserção em Lista Duplamente Ligada

- **Fim** da lista:
  - Percorrer toda a lista e inserir ao final.
  - O ponteiro da direita do novo nó deve apontar para NULL e o ponteiro da esquerda deve apontar para o final da lista.

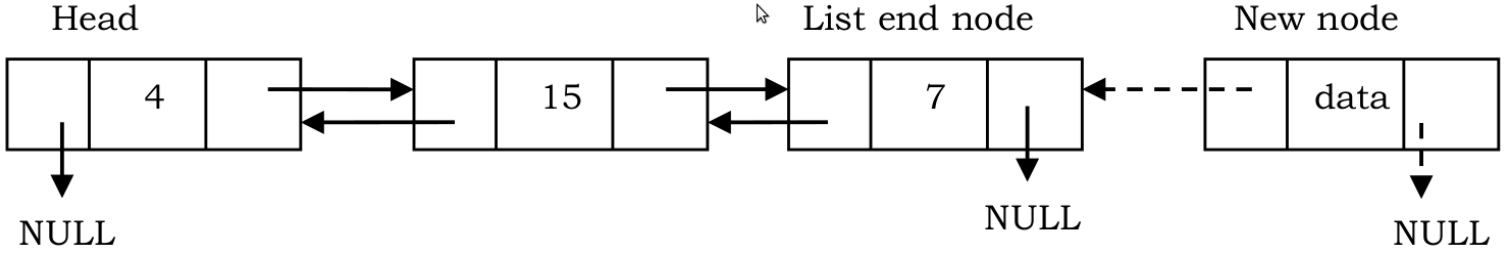
Inserção em Lista Duplamente Ligada

- **Fim** da lista:
  - Percorrer toda a lista e inserir ao final.
  - O ponteiro da direita do novo nó deve apontar para NULL e o ponteiro da esquerda deve apontar para o final da lista.



Inserção em Lista Duplamente Ligada

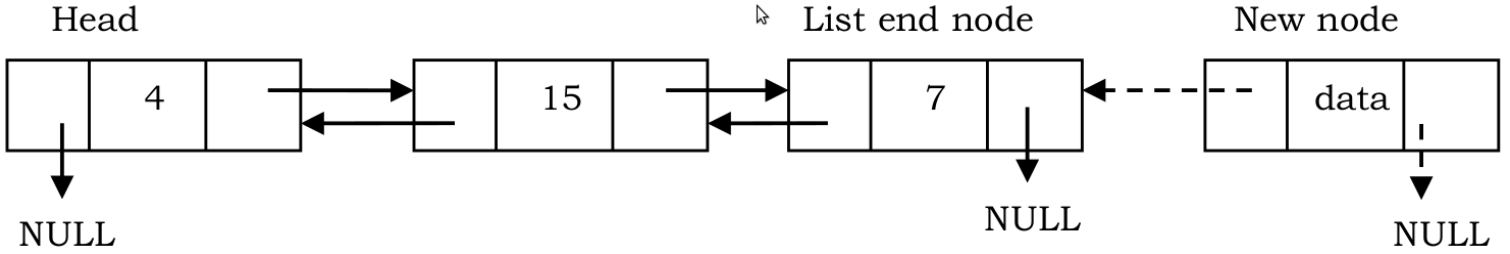
- **Fim** da lista:
  - Percorrer toda a lista e inserir ao final.
  - O ponteiro da direita do novo nó deve apontar para NULL e o ponteiro da esquerda deve apontar para o final da lista.



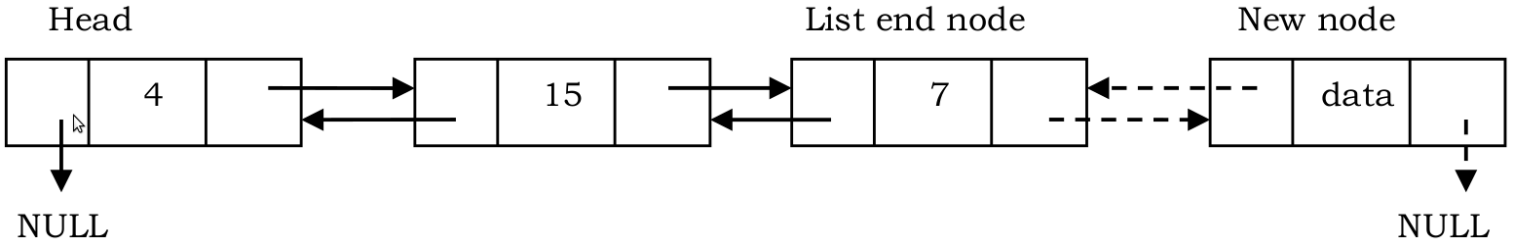
  - Atualize o ponteiro da direita do último nó para o novo nó.

Inserção em Lista Duplamente Ligada

- **Fim** da lista:
  - Percorrer toda a lista e inserir ao final.
  - O ponteiro da direita do novo nó deve apontar para NULL e o ponteiro da esquerda deve apontar para o final da lista.



  - Atualize o ponteiro da direita do último nó para o novo nó.



Inserção em Lista Duplamente Ligada (1/2)

Inserção em Lista Duplamente Ligada (1/2)

- No **meio** da lista:

Inserção em Lista Duplamente Ligada (1/2)

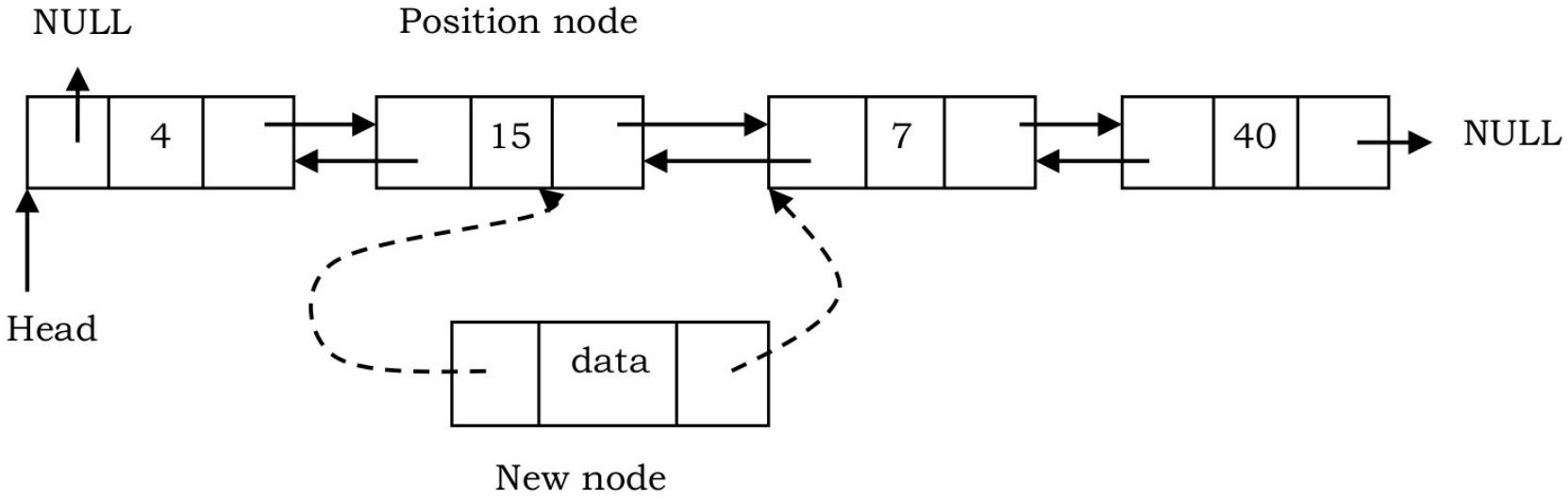
- No **meio** da lista:
  - Percorra a lista até a posição indicada.

Inserção em Lista Duplamente Ligada (1/2)

- No **meio** da lista:
  - Percorra a lista até a posição indicada.
  - O ponteiro da direita do novo nó irá apontar para o próximo elemento e o ponteiro da esquerda desse nó irá apontar para o nó atual (*position node*).

Inserção em Lista Duplamente Ligada (1/2)

- No **meio** da lista:
  - Percorra a lista até a posição indicada.
  - O ponteiro da direita do novo nó irá apontar para o próximo elemento e o ponteiro da esquerda desse nó irá apontar para o nó atual (*position node*).



Inserção em Lista Duplamente Ligada (2/2)

Inserção em Lista Duplamente Ligada (2/2)

- No **meio** da lista:

Inserção em Lista Duplamente Ligada (2/2)

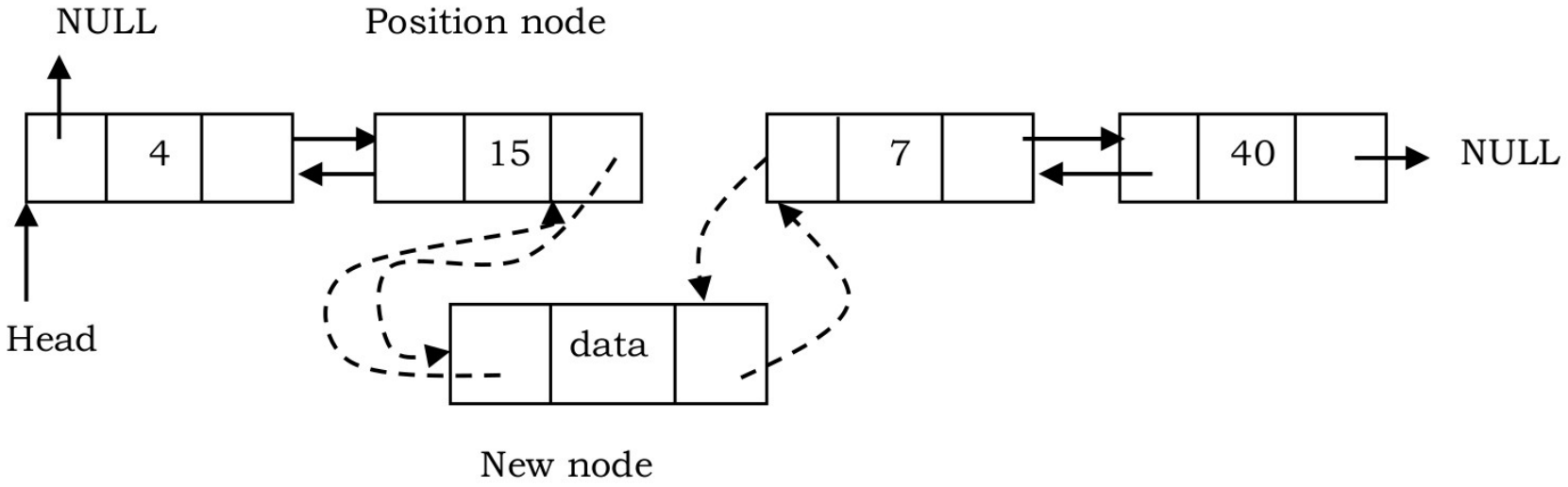
- No **meio** da lista:
  - O ponteiro da direita do nó atual (*position node*) aponta para o novo nó.

Inserção em Lista Duplamente Ligada (2/2)

- No **meio** da lista:
  - O ponteiro da direita do nó atual (*position node*) aponta para o novo nó.
  - O ponteiro da esquerda do próximo nó também aponta para o novo nó.

Inserção em Lista Duplamente Ligada (2/2)

- No **meio** da lista:
  - O ponteiro da direita do nó atual (*position node*) aponta para o novo nó.
  - O ponteiro da esquerda do próximo nó também aponta para o novo nó.



Remoção em Lista Duplamente Ligada

Remoção em Lista Duplamente Ligada

- Existem três casos:

Remoção em Lista Duplamente Ligada

- Existem três casos:
  - Remover o primeiro nó.

Remoção em Lista Duplamente Ligada

- Existem três casos:
  - Remover o primeiro nó.
  - Remover o último nó.

Remoção em Lista Duplamente Ligada

- Existem três casos:
  - Remover o primeiro nó.
  - Remover o último nó.
  - Remover um nó intermediário.

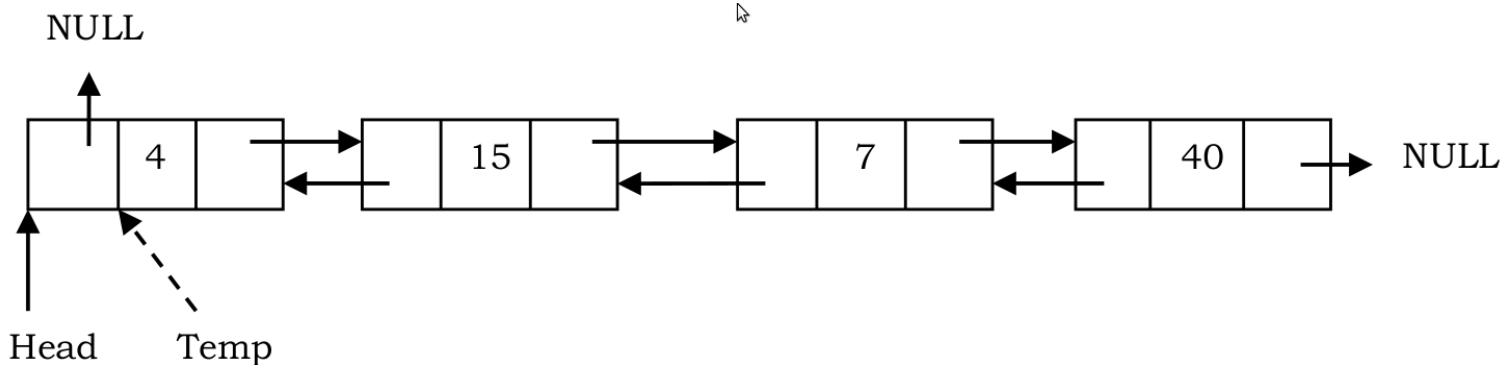
Remoção em Lista Duplamente Ligada (1/2)

Remoção em Lista Duplamente Ligada (1/2)

- **Primeiro nó**
  - Crie um ponteiro temporário (`temp`) que apontará para o mesmo endereço do ponteiro `head`.

Remoção em Lista Duplamente Ligada (1/2)

- **Primeiro nó**
  - Crie um ponteiro temporário (`temp`) que apontará para o mesmo endereço do ponteiro `head`.



Remoção em Lista Duplamente Ligada (2/2)

Remoção em Lista Duplamente Ligada (2/2)

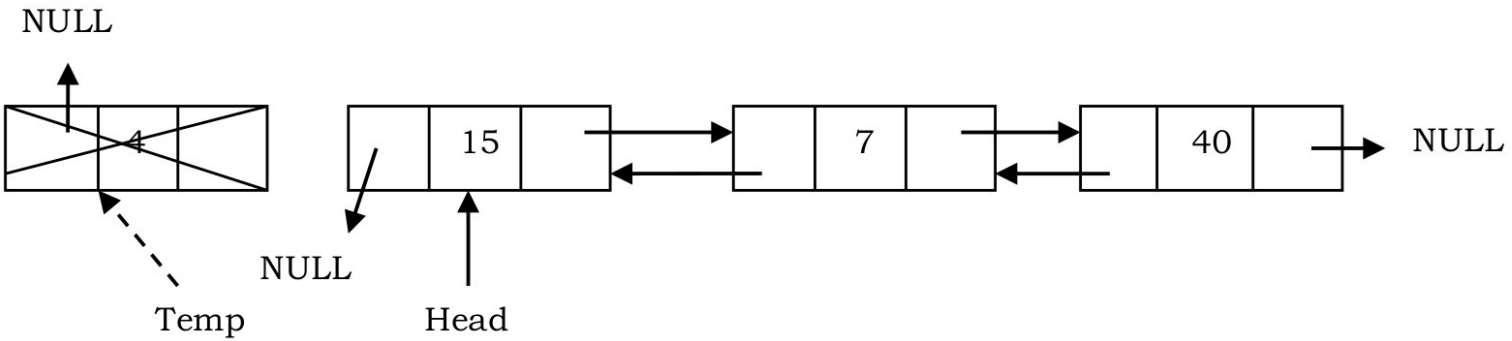
- **Primeiro nó**

Remoção em Lista Duplamente Ligada (2/2)

- **Primeiro nó**
  - Mova o ponteiro `head` para o próximo elemento e o ponteiro a esquerda do antigo primeiro nó deverá apontar para NULL.

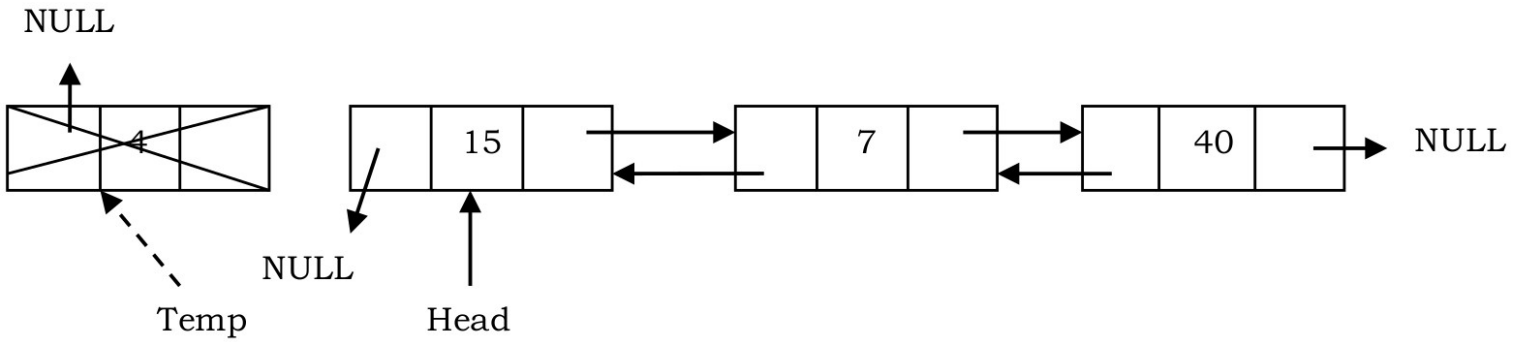
Remoção em Lista Duplamente Ligada (2/2)

- **Primeiro nó**
  - Mova o ponteiro `head` para o próximo elemento e o ponteiro a esquerda do antigo primeiro nó deverá apontar para NULL.



Remoção em Lista Duplamente Ligada (2/2)

- **Primeiro nó**
  - Mova o ponteiro `head` para o próximo elemento e o ponteiro a esquerda do antigo primeiro nó deverá apontar para NULL.



  - Então o ponteiro Temp deverá ser liberado.

Remoção em Lista Duplamente Ligada (1/2)

Remoção em Lista Duplamente Ligada (1/2)

- **Último nó**

Remoção em Lista Duplamente Ligada (1/2)

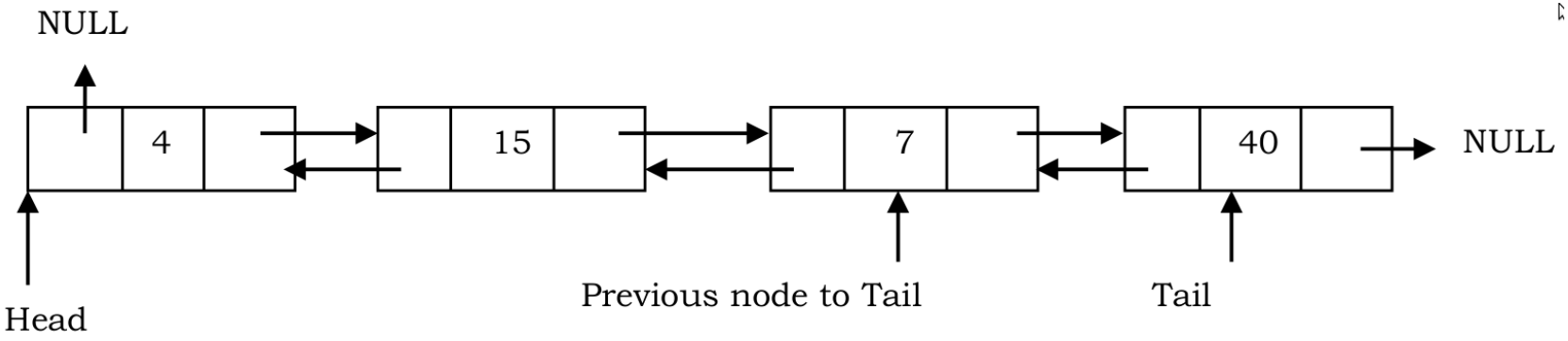
- **Último nó**
  - Percorrer a lista e manter o endereço do nó anterior.

Remoção em Lista Duplamente Ligada (1/2)

- **Último nó**
  - Percorrer a lista e manter o endereço do nó anterior.
  - Ao chegar no final da lista, teremos um ponteiro apontando para o último e outro apontando para o penúltimo nó.

Remoção em Lista Duplamente Ligada (1/2)

- **Último nó**
  - Percorrer a lista e manter o endereço do nó anterior.
  - Ao chegar no final da lista, teremos um ponteiro apontando para o último e outro apontando para o penúltimo nó.



Remoção em Lista Duplamente Ligada (2/2)

Remoção em Lista Duplamente Ligada (2/2)

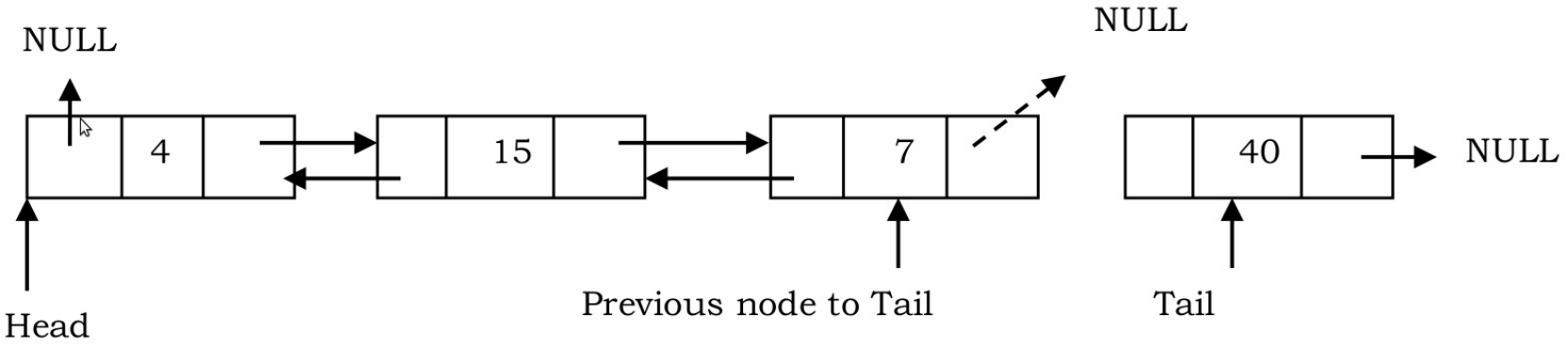
- **Último nó**

Remoção em Lista Duplamente Ligada (2/2)

- **Último nó**
  - Atualize o penúltimo nó para NULL.

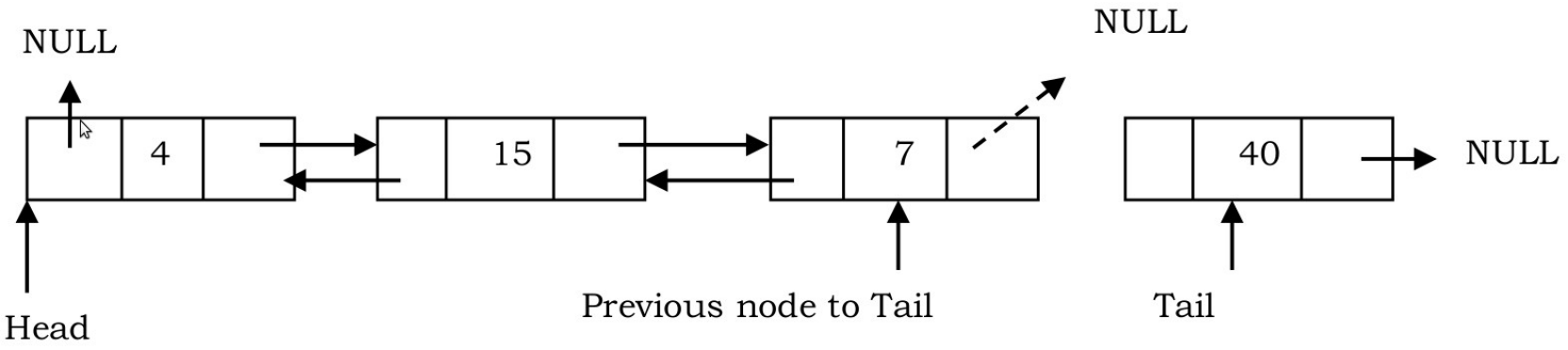
Remoção em Lista Duplamente Ligada (2/2)

- **Último nó**
  - Atualize o penúltimo nó para NULL.



Remoção em Lista Duplamente Ligada (2/2)

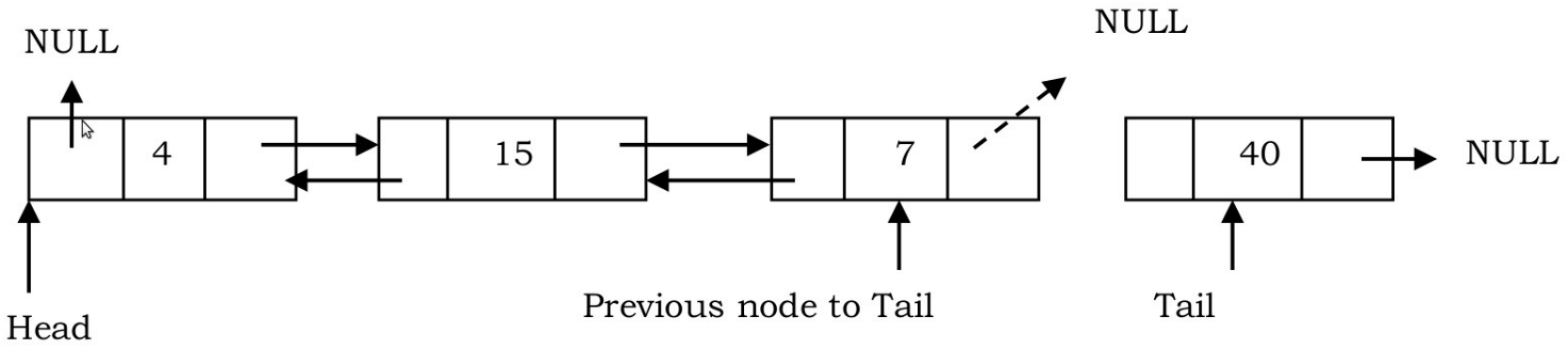
- **Último nó**
  - Atualize o penúltimo nó para NULL.



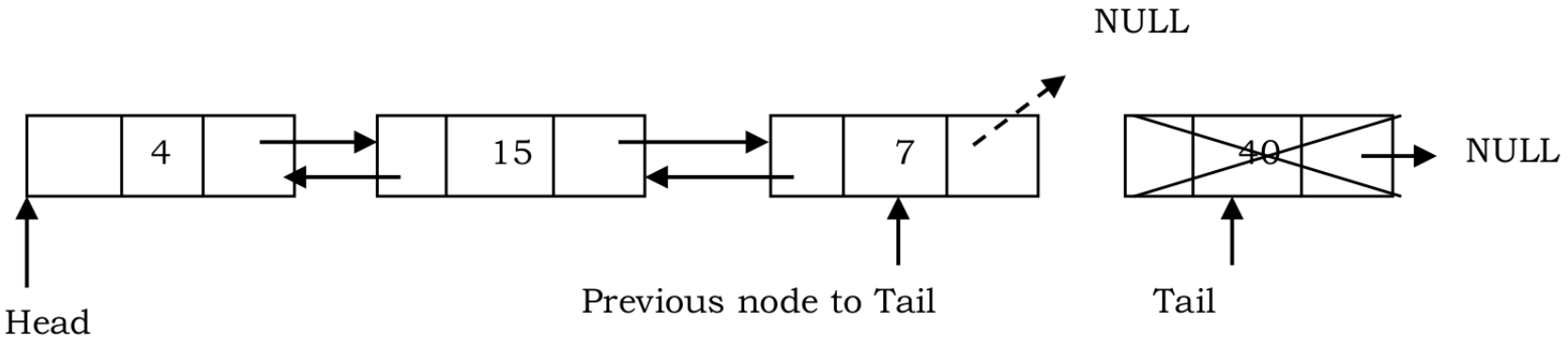
  - Libere o último (antigo) nó.

Remoção em Lista Duplamente Ligada (2/2)

- **Último nó**
  - Atualize o penúltimo nó para NULL.



  - Libere o último (antigo) nó.



Remoção em Lista Duplamente Ligada (1/2)

Remoção em Lista Duplamente Ligada (1/2)

- **Nó intermediário**

Remoção em Lista Duplamente Ligada (1/2)

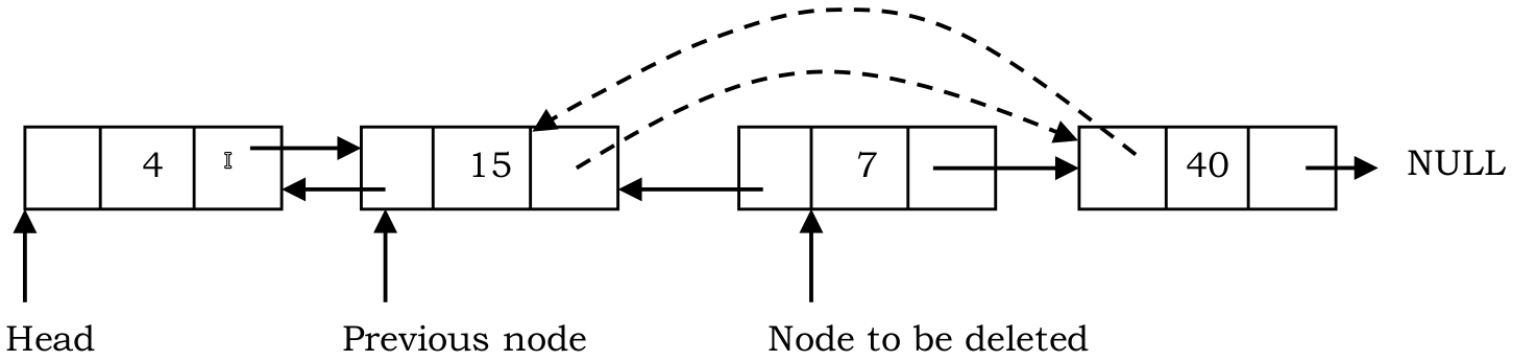
- **Nó intermediário**
  - Os nós *head* e *tail* não são alterados.

Remoção em Lista Duplamente Ligada (1/2)

- **Nó intermediário**
  - Os nós *head* e *tail* não são alterados.
  - Deve-se manter também o ponteiro do nó anterior enquanto percorre a lista.

Remoção em Lista Duplamente Ligada (1/2)

- **Nó intermediário**
  - Os nós *head* e *tail* não são alterados.
  - Deve-se manter também o ponteiro do nó anterior enquanto percorre a lista.



Remoção em Lista Duplamente Ligada (2/2)

Remoção em Lista Duplamente Ligada (2/2)

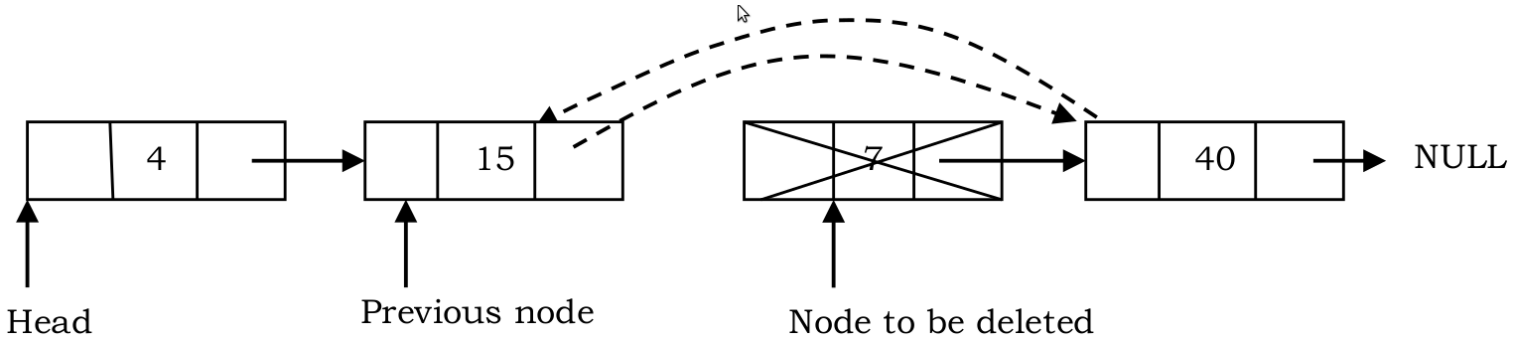
- **Nó intermediário**

Remoção em Lista Duplamente Ligada (2/2)

- **Nó intermediário**
  - Libere o nó que será removido.

Remoção em Lista Duplamente Ligada (2/2)

- **Nó intermediário**
  - Libere o nó que será removido.



Exercícios

- Escreva uma função em Python para remover elementos repetidos em uma lista ordenada. Assuma que seja fornecido uma lista encadeada armazenando números inteiros em ordem crescente. O objetivo da função é remover todos os elementos duplicados da lista. Por exemplo, dada a lista

  $L = 0 \rightarrow 2 \rightarrow 6 \rightarrow 8 \rightarrow 11 \rightarrow 11 \rightarrow \text{None}$, o seu programa deverá retornar a lista

  $L = 0 \rightarrow 2 \rightarrow 6 \rightarrow 8 \rightarrow 11 \rightarrow \text{None}$

Exercícios

- Escreva uma função em Python que receba uma lista de inteiros de entrada e crie uma lista encadeada com esses valores.
- Escreva um programa em Python que faça a concatenação de duas listas encadeadas.
- Rescreva o código do exercício anterior usando listas duplamente encadeadas.
- Escreva uma função em Python que receba uma lista ligada e retorne o tamanho da lista.

Exercícios

- É possível representar elementos de tipos distintos em uma lista ligada? Justifique a sua resposta.
- Escreva uma função em Python que receba como parâmetros uma lista ligada L e um valor inteiro X. A função deverá remover o elemento X da lista L, caso ele seja um dos elementos.