Algoritmos e Estruturas de Dados I

Profª Priscilla Abreu

priscilla.abreu@ime.uerj.br

2022.1

Algoritmos e Estruturas de Dados I

Roteiro da aula

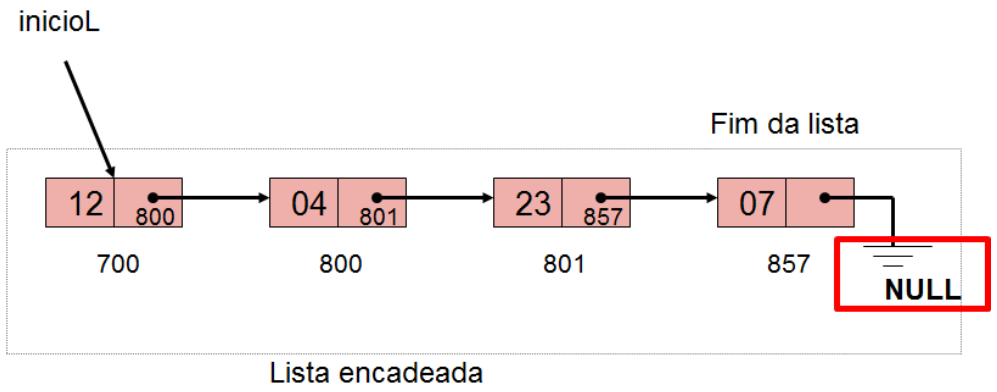
- Listas Duplamente Encadeadas

Revisando...

Algoritmos e Estruturas de Dados I

Alocação Dinâmica – Listas Encadeadas

- Nó da lista é representado por pelo menos dois campos:
  - a informação armazenada;
  - o ponteiro para o próximo elemento da lista.
- a lista é representada por um ponteiro para o primeiro nó;
- o campo próximo do último elemento é NULL.




```
typedef struct no{
      int info;
      struct no *prox;
}no;

no *inicioL;
```

Algoritmos e Estruturas de Dados I

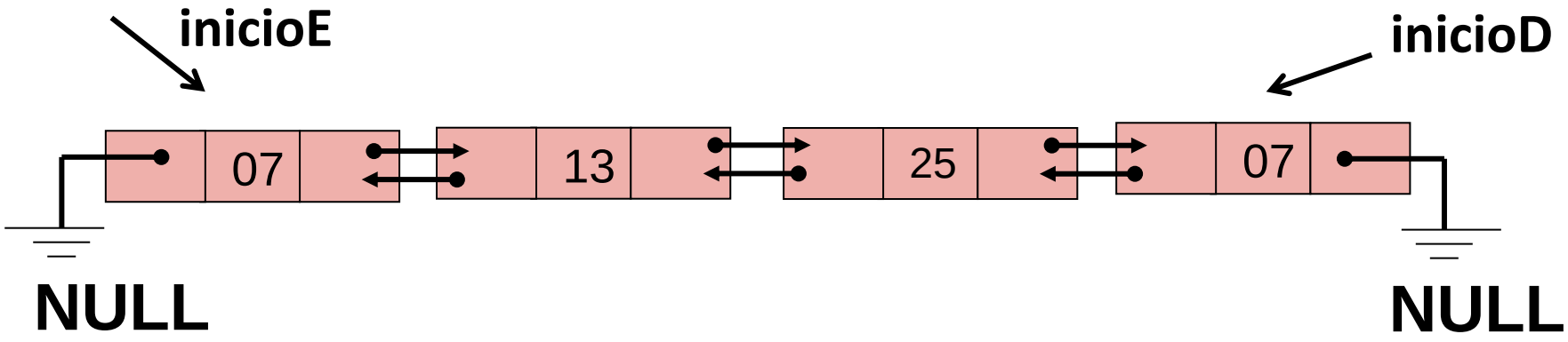
Listas Simplesmente Encadeadas

Limitações:

- Único sentido de percurso
- Necessidade de conhecer antecessor para inserir ou remover em uma k-ésima posição

Algoritmos e Estruturas de Dados I

Listas Duplamente Encadeadas

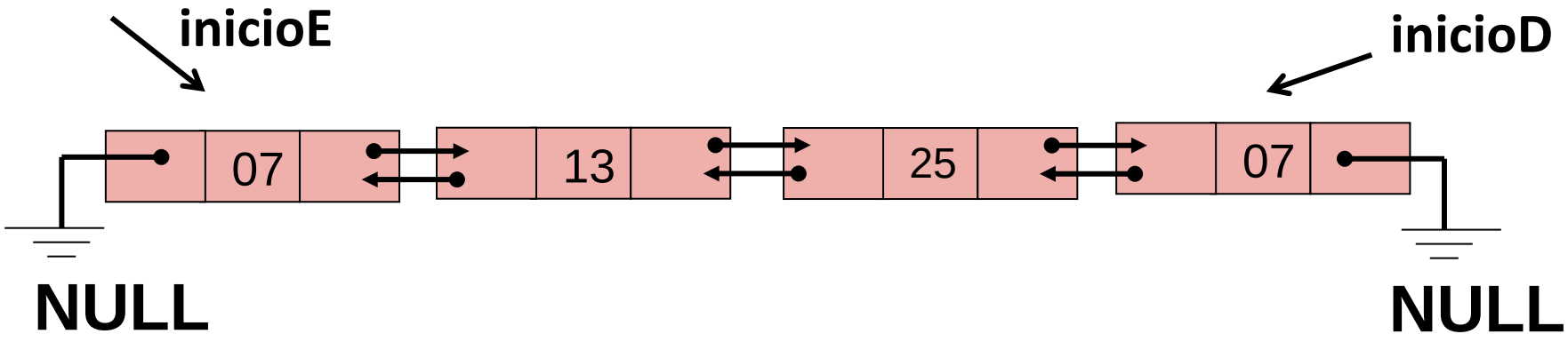


Estrutura básica da lista

Cada nó possui dois ponteiros: um para o elemento anterior e outro para o próximo elemento (ant e prox).

Algoritmos e Estruturas de Dados I

Listas Duplamente Encadeadas

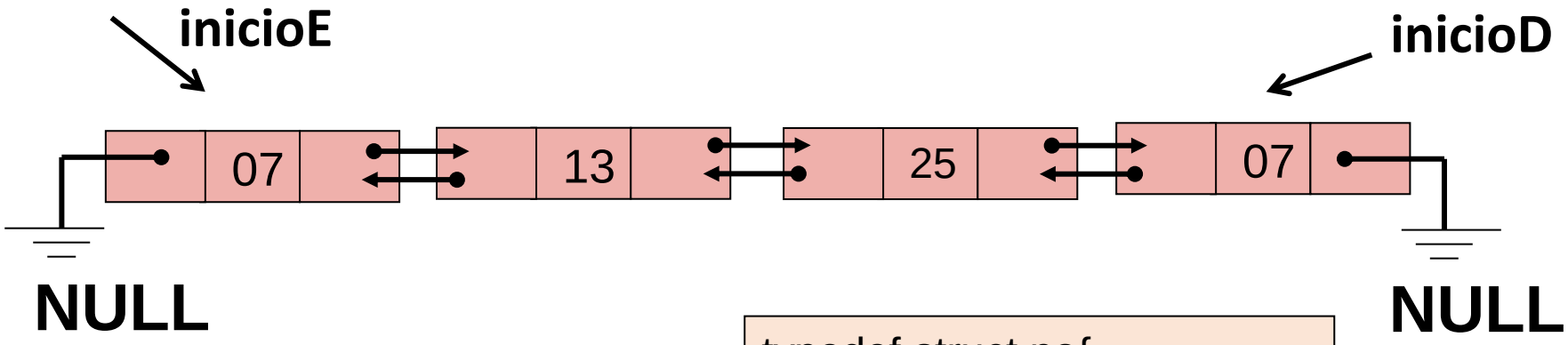


Estrutura básica da lista

Além disso, a lista possui uma referência para o início e outra para o final.

Algoritmos e Estruturas de Dados I

Listas Duplamente Encadeadas



Estrutura básica da lista

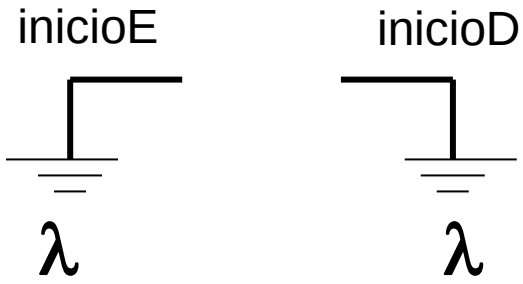
```
typedef struct no{
      int info;
      struct no *prox, *ant;
}no;

no *inicioE;
no *inicioD;
```

Algoritmos e Estruturas de Dados I


```
#include <stdio.h>
#include <stdlib.h>
typedef struct no{
   int info;
          struct no *ant, *prox;
}no;
no *inicioE;
no *inicioD;
int main(){
}
```

Algoritmos e Estruturas de Dados I

Inicializar a lista




```
void inicializa_lista ()
{
        inicioD = NULL;
        inicioE  =  NULL;

}
```

Algoritmos e Estruturas de Dados I

Lista Vazia

```
int lista_vazia () {
   if (inicioD == NULL)
         return 1;
   return 0;
}
```

Algoritmos e Estruturas de Dados I

Inserção na lista

Primeiro passo:

Alocar espaço para um elemento do tipo no;

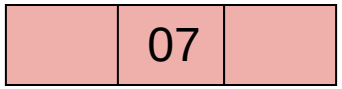
Algoritmos e Estruturas de Dados I

Criar a estrutura de um nó para posterior inserção:

1. SOLICITAR ALOCAÇÃO

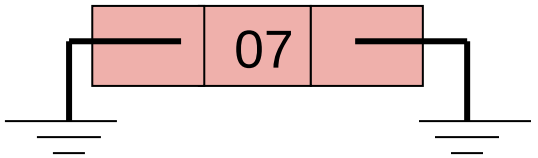
aux

2. ATRIBUIR VALOR AO CAMPO INFO



aux

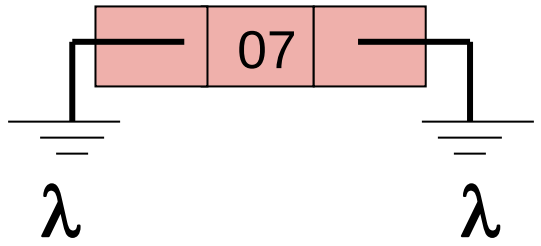
3. ATRIBUIR VALOR NULL AOS CAMPOS PROX E ANT.



Algoritmos e Estruturas de Dados I

Alocação do nó

```
no* cria_no (int valor){
    no *aux;
    aux = (no*) malloc (sizeof (no));
    if (aux == NULL) {
            printf("ERRO!!!");
            exit(0);
    }
    else{
            aux -> info= valor;
            aux -> ant = NULL;
            aux -> prox = NULL;
            return aux;
    }
}
```



Algoritmos e Estruturas de Dados I

INSERÇÃO NA LISTA:

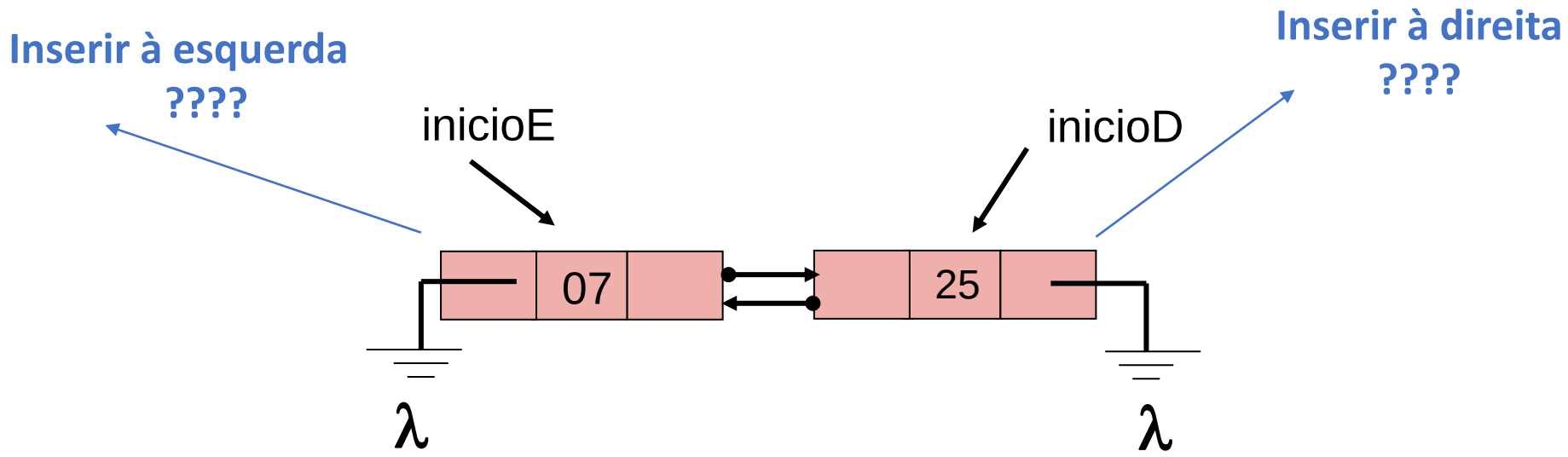
Para inserir um elemento na lista precisamos pensar em alguns pontos:

- É o primeiro elemento?
- Por qual lado ocorrerá a inserção?

Algoritmos e Estruturas de Dados I

Inserir elemento na lista:

- Por qual lado ocorrerá a inserção?

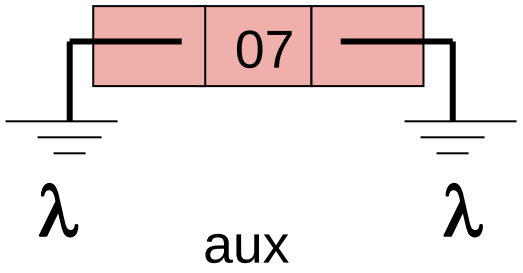


Algoritmos e Estruturas de Dados I

Inserir elemento na lista:

Solicitar a alocação do espaço para um novo nó...

criar_no(valor)

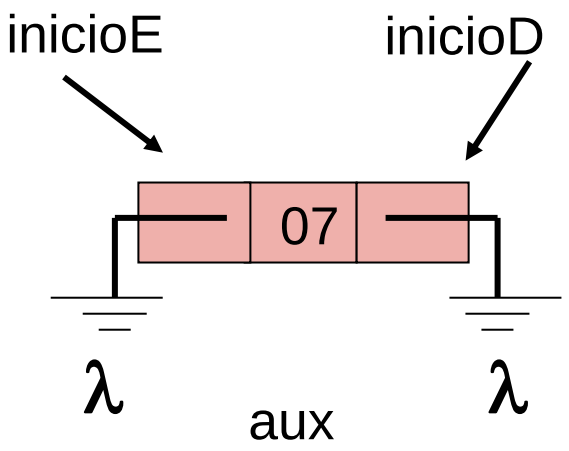


Algoritmos e Estruturas de Dados I

Inserir elemento na lista:

- 1º elemento?

Apontar os ponteiros inicioD e inicioE para o elemento criado:



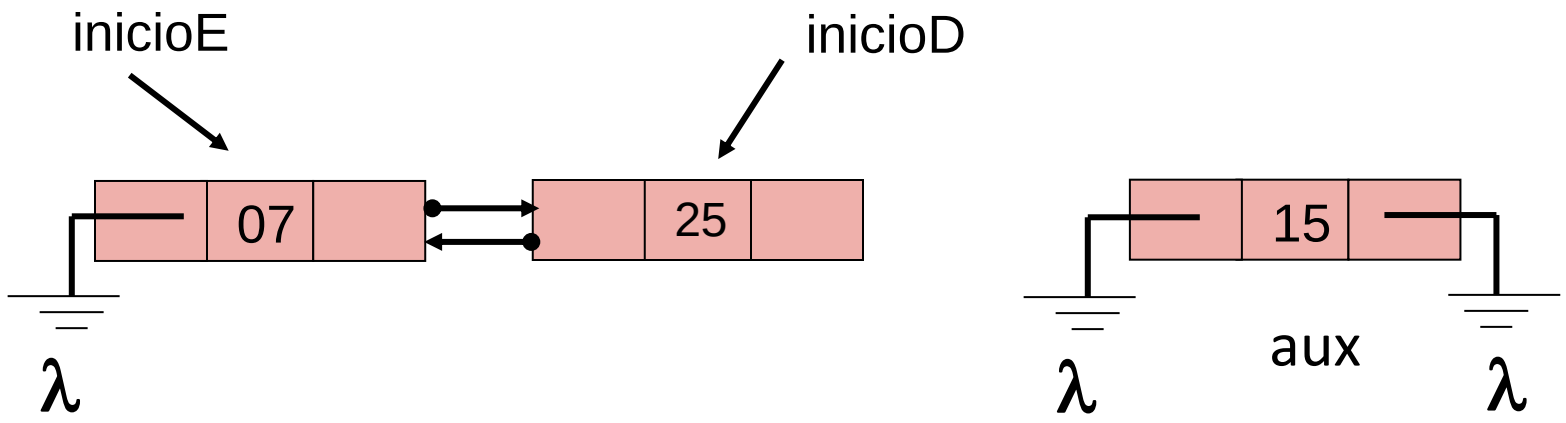
inicioD = aux
inicioE = aux

Algoritmos e Estruturas de Dados I

Inserir elemento à direita:

Suponha que não seja o primeiro elemento.

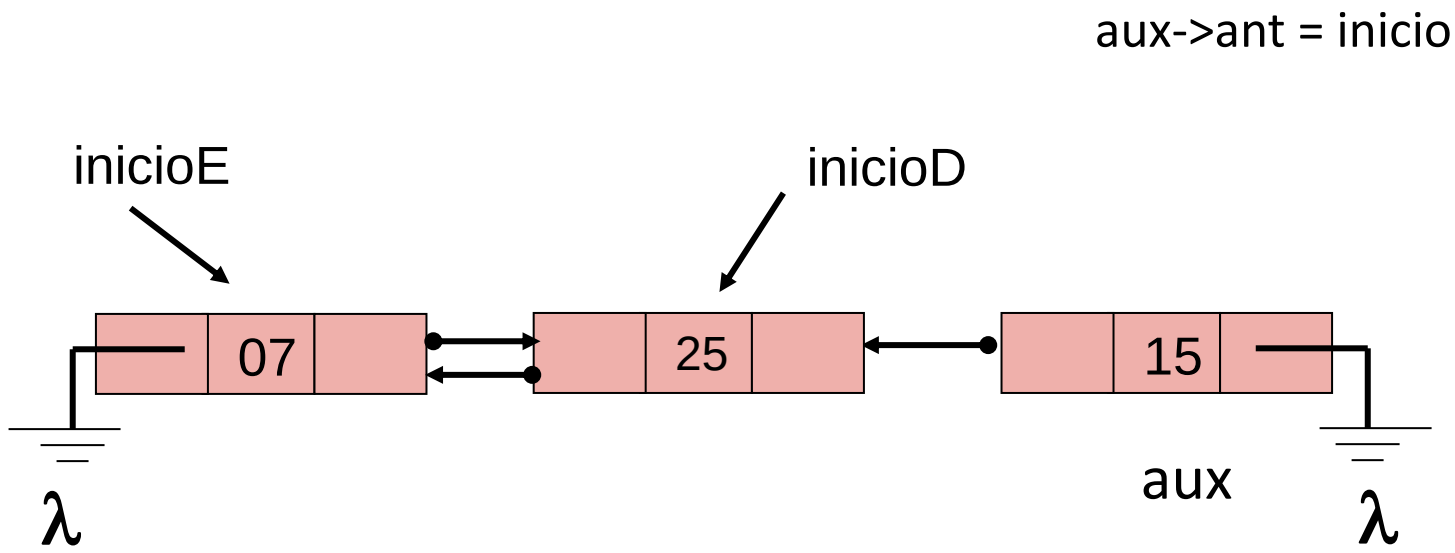
Ligar os ponteiros entre o elemento da direita e o novo nó...



Algoritmos e Estruturas de Dados I

Inserir elemento à direita:

Ligar os ponteiros entre o elemento da direita e o novo nó...



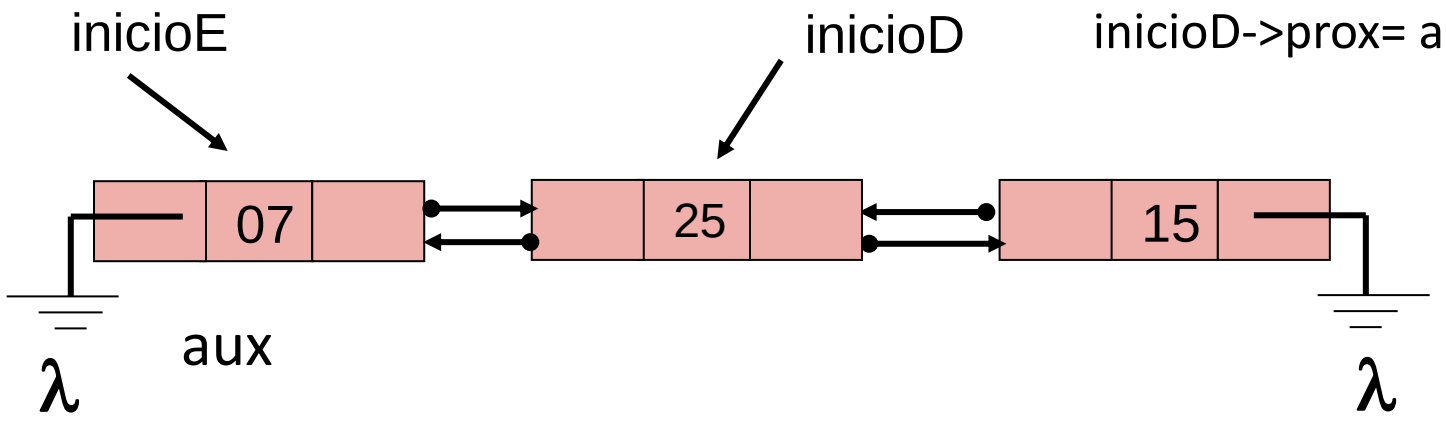
Algoritmos e Estruturas de Dados I

Inserir elemento à direita:

Ligar os ponteiros entre o elemento da direita e o novo nó...

aux->ant = inicioD

inicioD->prox= aux

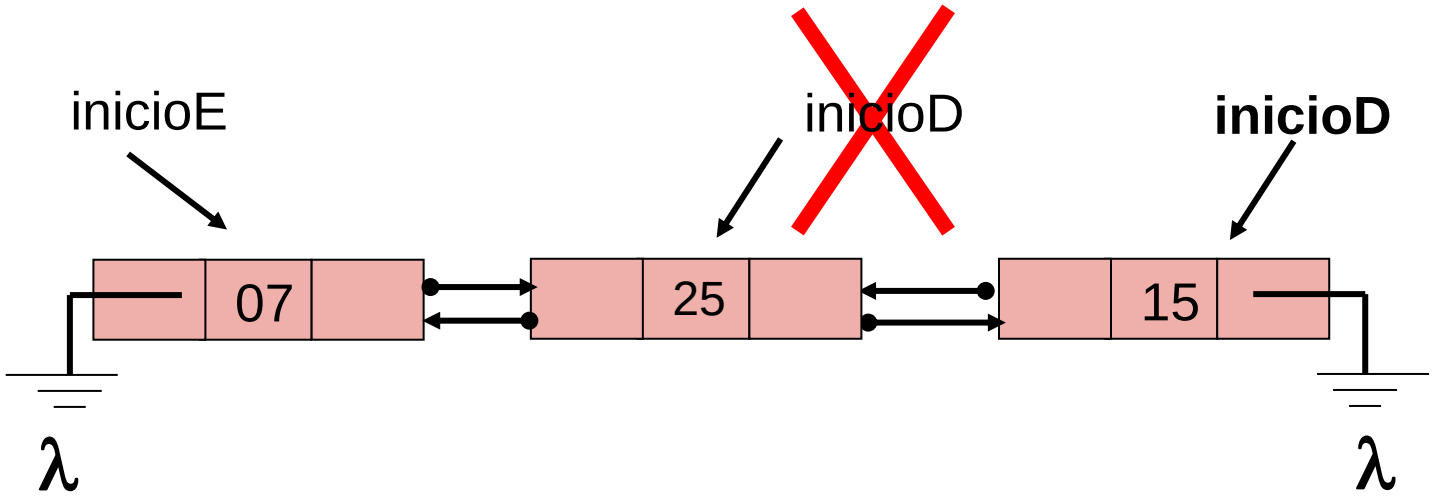


Algoritmos e Estruturas de Dados I

Inserir elemento à direita:

Atualizar o ponteiro inicioD

inicioD = aux



Algoritmos e Estruturas de Dados I

Inserir elemento à direita:

```
void inserir_Dir(int valor){
   no* aux;
            aux = cria_no(valor);
   if (lista_vazia()){
            inicioD = aux;
            inicioE = aux;
   }
   else{
            inicioD->prox = aux;
            aux->ant = inicioD;
            inicioD = aux;
   }
}
```

Algoritmos e Estruturas de Dados I

IMPRESSÃO:

Pela esquerda ou direita???

Algoritmos e Estruturas de Dados I

IMPRESSÃO: Pela esquerda ou direita???

```
void percorrer_Esq () {
   no * aux;
   aux = inicioE;
   while (aux!= NULL) {
          printf("%d",aux->info);
          aux = aux->prox;
          }
}
```

Algoritmos e Estruturas de Dados I

IMPRESSÃO: Pela esquerda ou direita???

```
void percorrer_Dir () {
   no * aux;
   aux = inicioD;
   while (aux!= NULL) {
          printf("%d",aux->info);
          aux = aux->ant;
          }
}
```

Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

Para remover um elemento na lista precisamos pensar em alguns pontos:

- Lista está vazia?
- O elemento a ser removido encontra-se na lista?

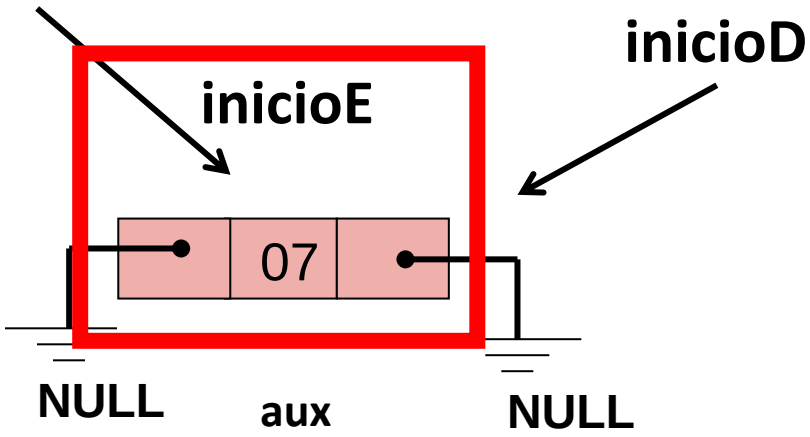
```
if (!lista_vazia()) {
    aux = busca_no(valor);
    ...
}
```

Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

- A lista tem um único elemento?

```
aux = busca_no(valor);
...
inicioE = NULL;
inicioD = NULL;
free(aux);
...
```

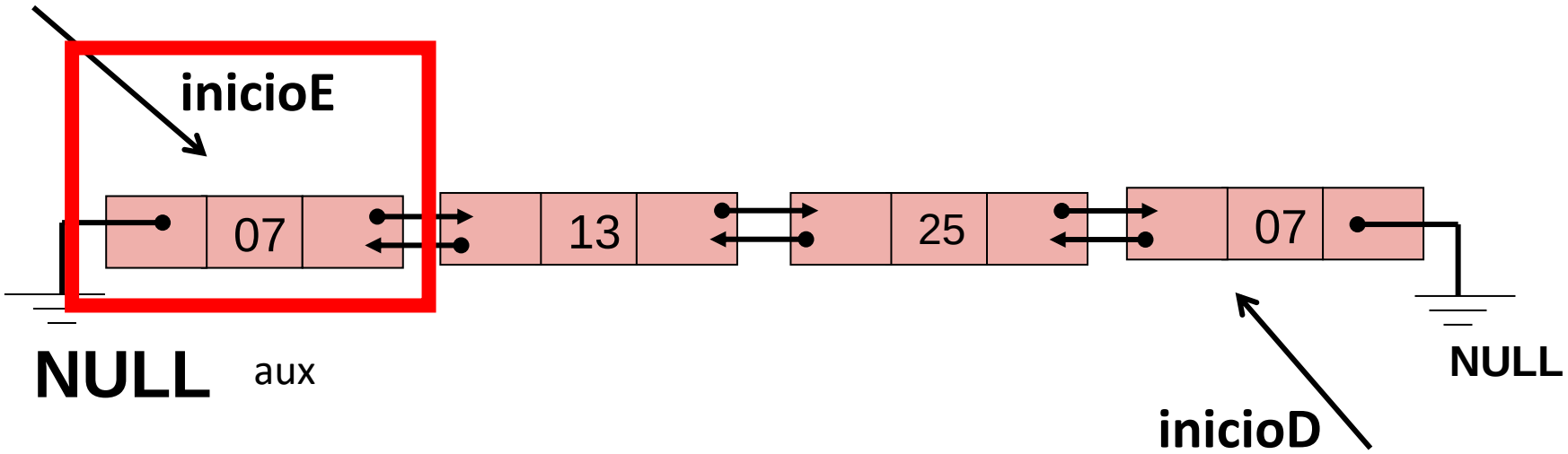


Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

```
aux = busca_no(valor);
...
```

- O elemento a ser removido é o primeiro da esquerda?

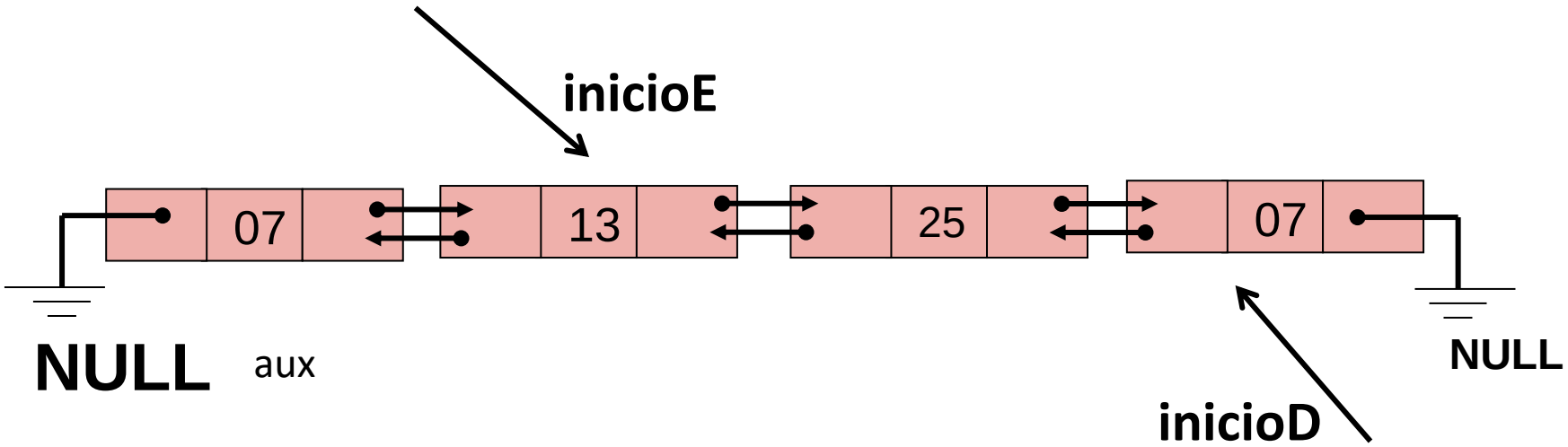


Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

```
aux = busca_no(valor);
...
inicioE = aux->prox;
```

- O elemento a ser removido é o primeiro da esquerda?

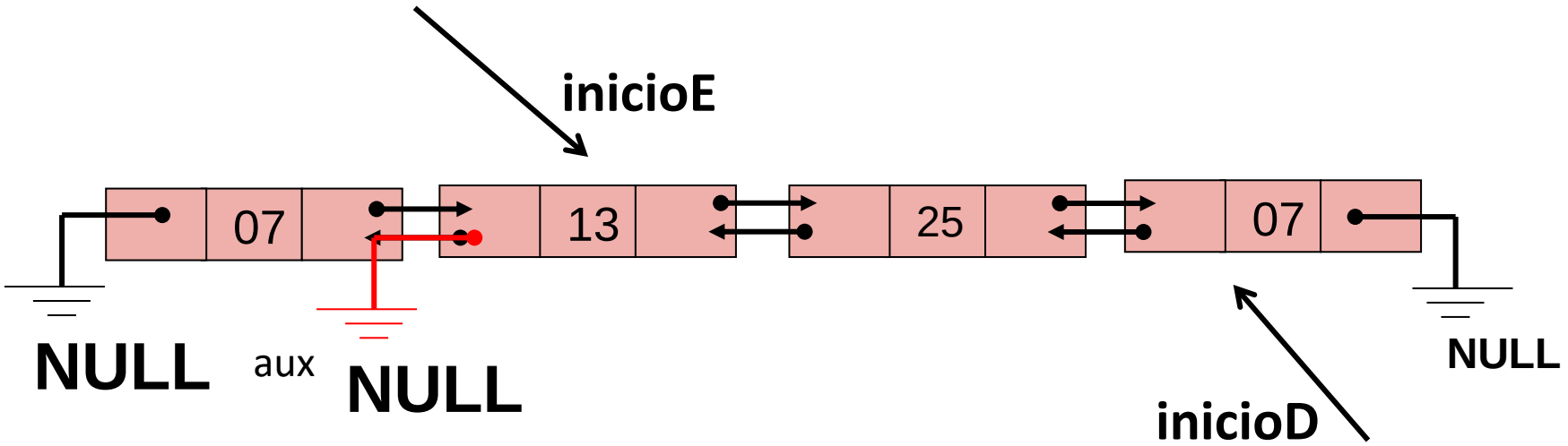


Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

- O elemento a ser removido é o primeiro da esquerda?

```
aux = busca_no(valor);
...
inicioE = aux->prox
inicioE->ant = NULL;
```

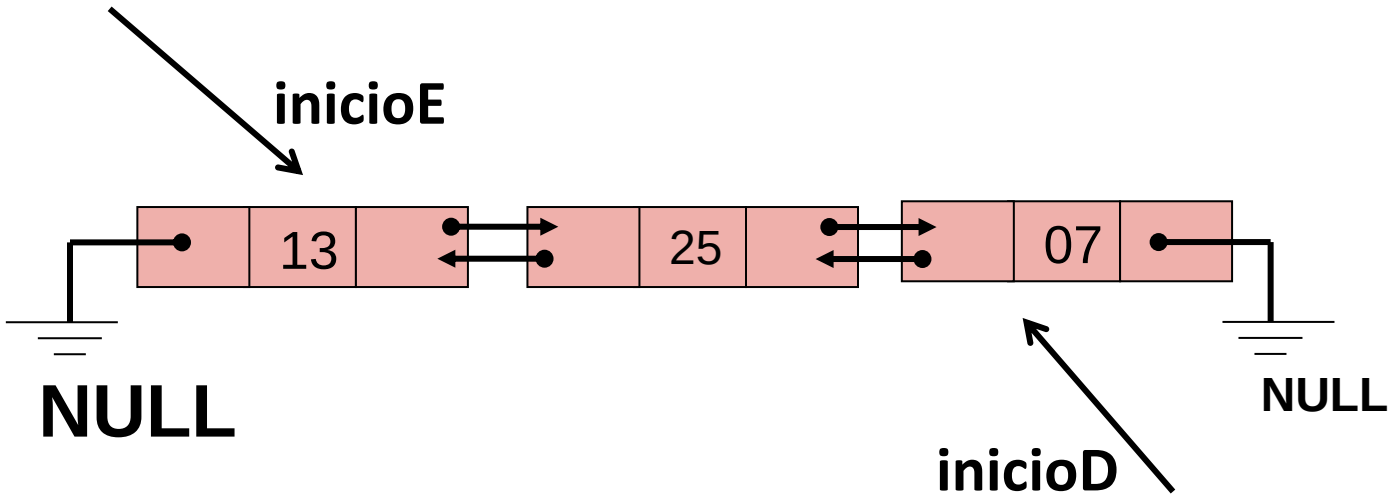


Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

- O elemento a ser removido é o primeiro da esquerda?

```
aux = busca_no(valor);
...
inicioE = aux->prox
inicioE->ant = NULL;
free(aux);
```

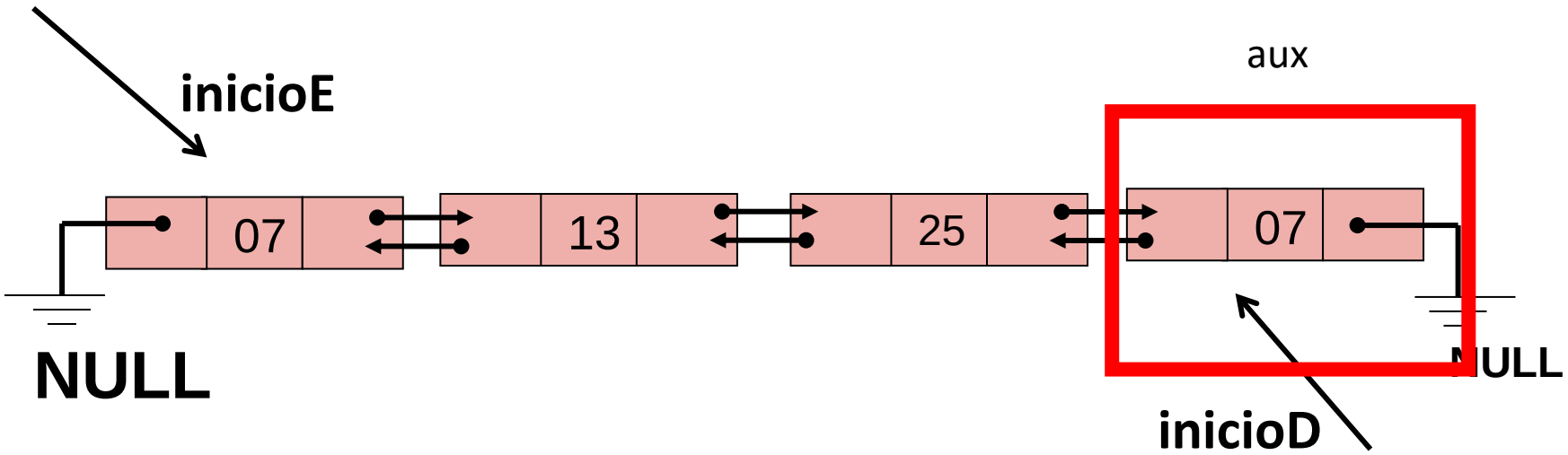


Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

```
aux = busca_no(valor);
...
```

- O elemento a ser removido é o primeiro da direita?

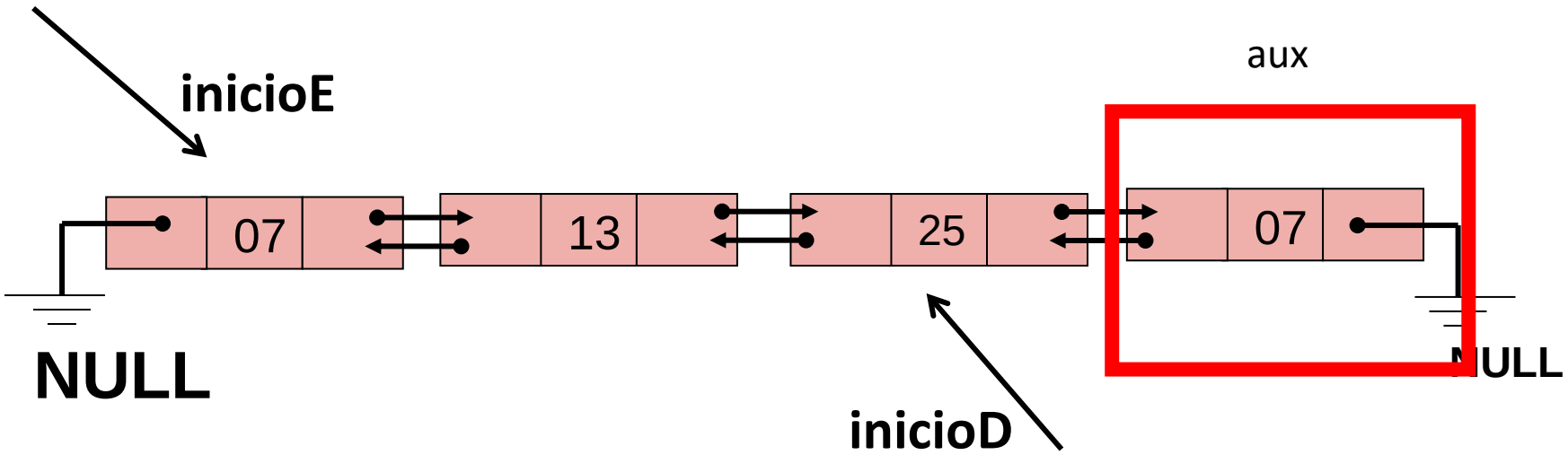


Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

- O elemento a ser removido é o primeiro da direita?

```
aux = busca_no(valor);
...
inicioD = aux->ant;
```

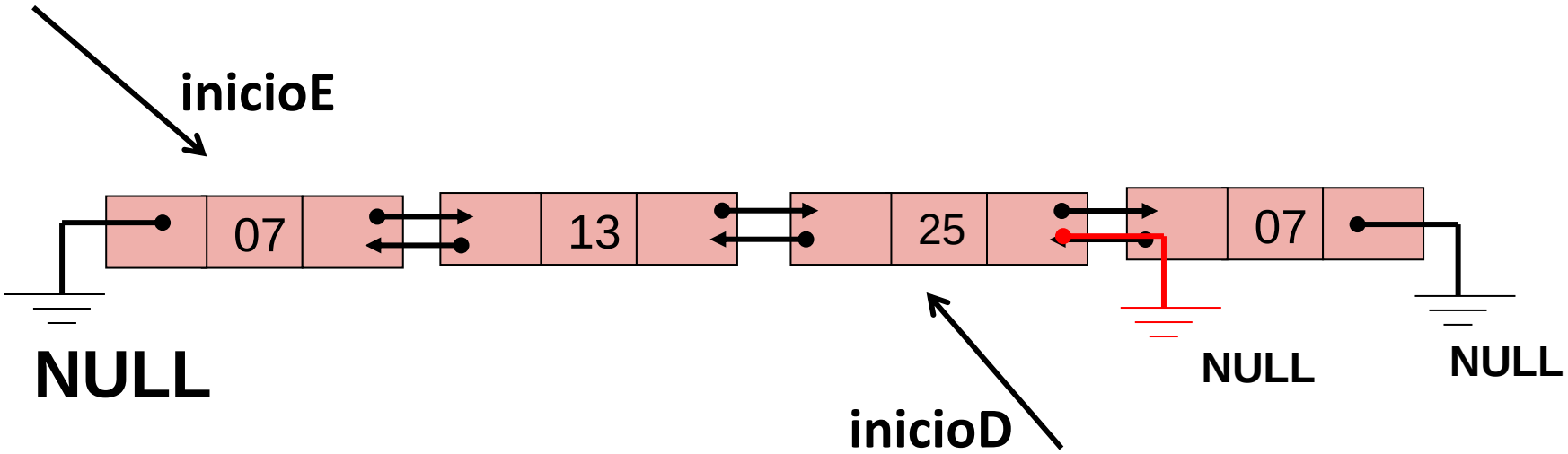


Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

- O elemento a ser removido é o primeiro da direita?

```
aux = busca_no(valor);
...
inicioD = aux->ant;
inicioD->prox = NULL;
```

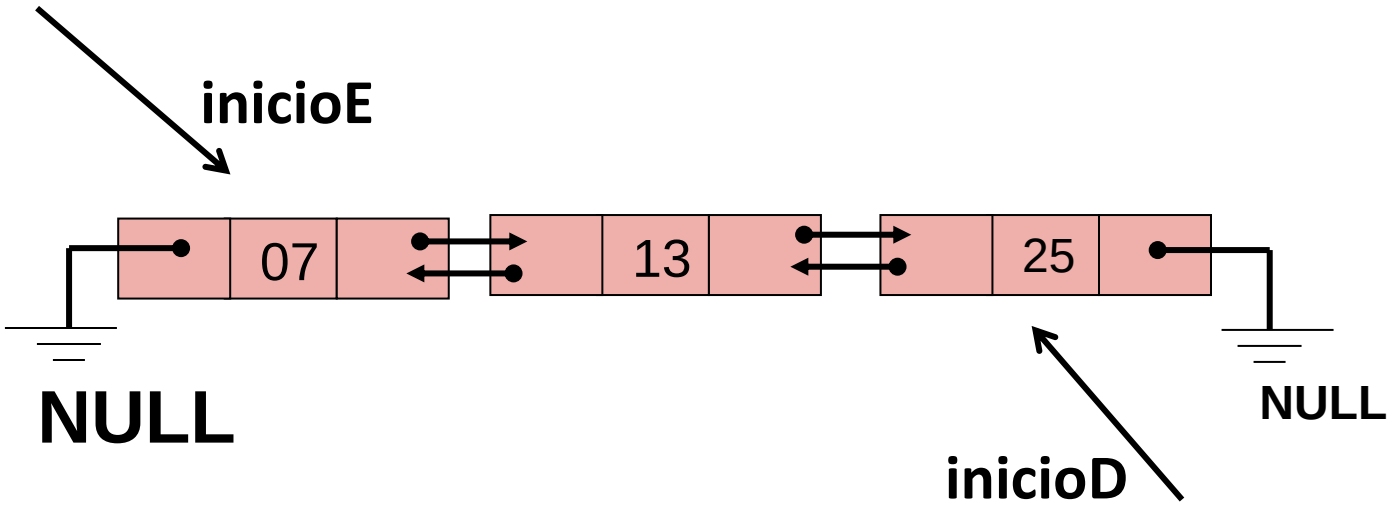


Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

- O elemento a ser removido é o primeiro da direita?

```
aux = busca_no(valor);
...
inicioD = aux->ant;
inicioD->prox = NULL;
free(aux);
```

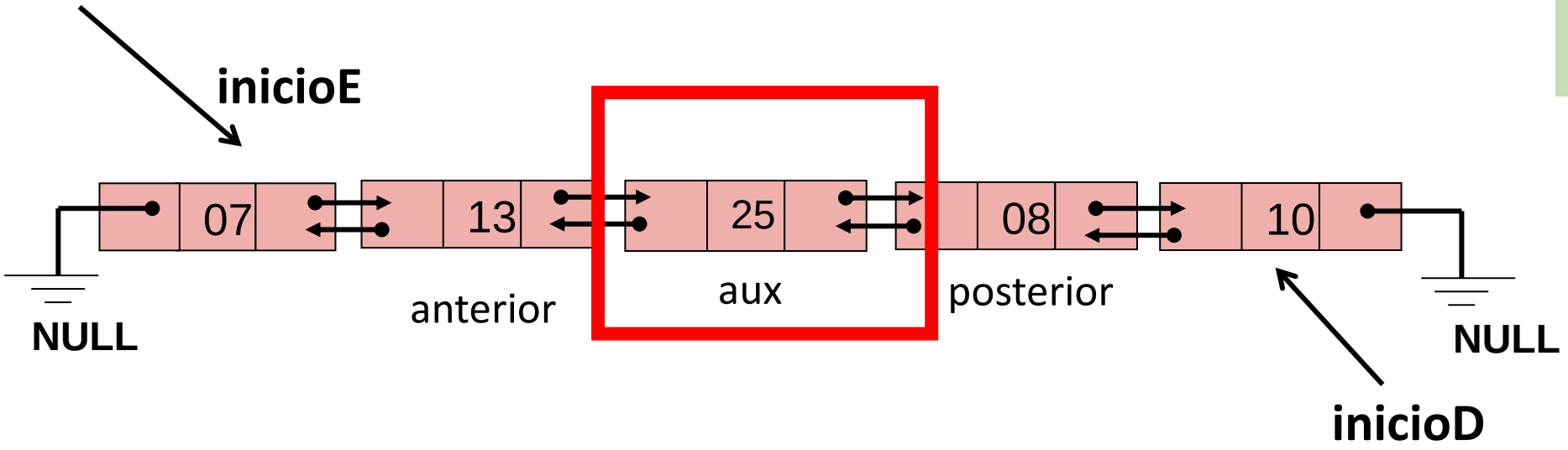


Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

- O elemento a ser removido está em qualquer outra posição?

```
aux = busca_no(valor);
...
anterior = aux->ant;
posterior = aux->prox;
```

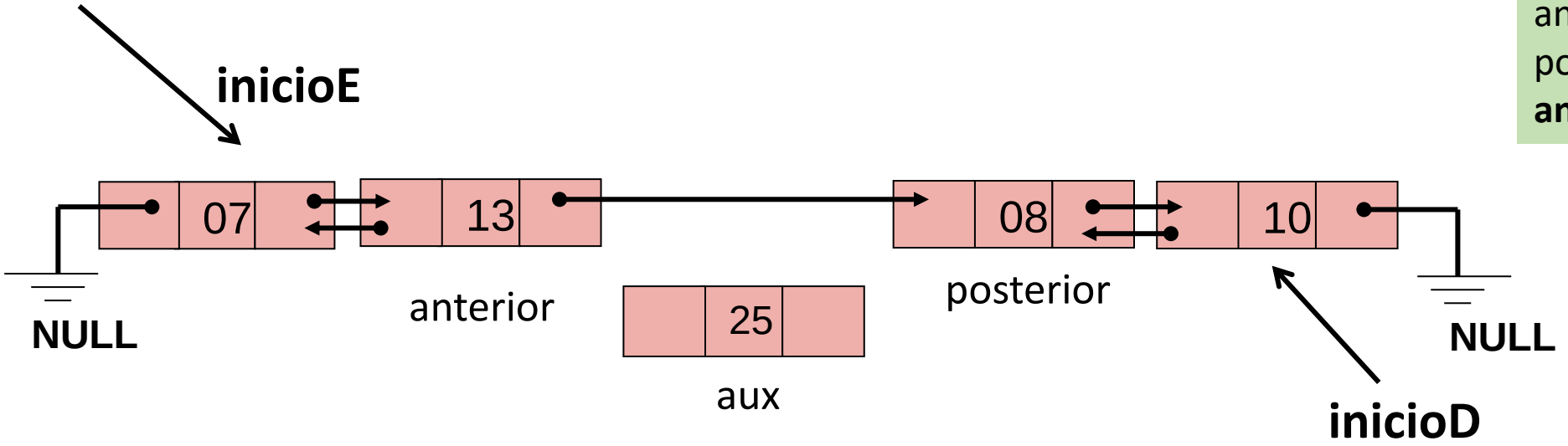


Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

- O elemento a ser removido está em qualquer outra posição?

```
aux = busca_no(valor);
...
anterior = aux->ant;
posterior = aux->prox;
anterior->prox = posterior;
```

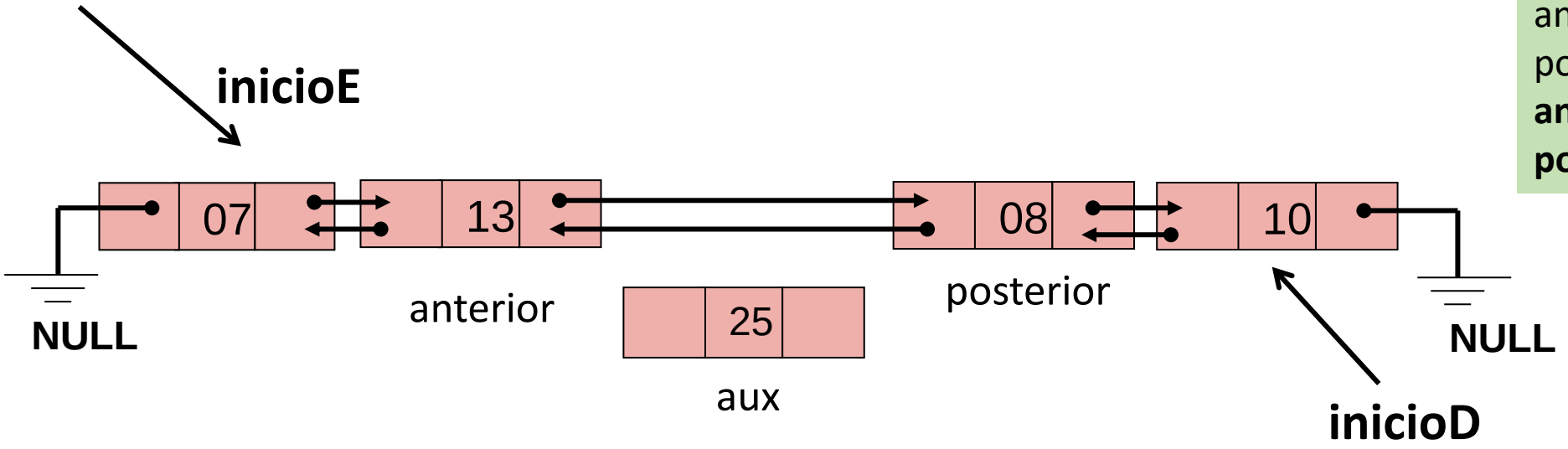


Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

- O elemento a ser removido está em qualquer outra posição?

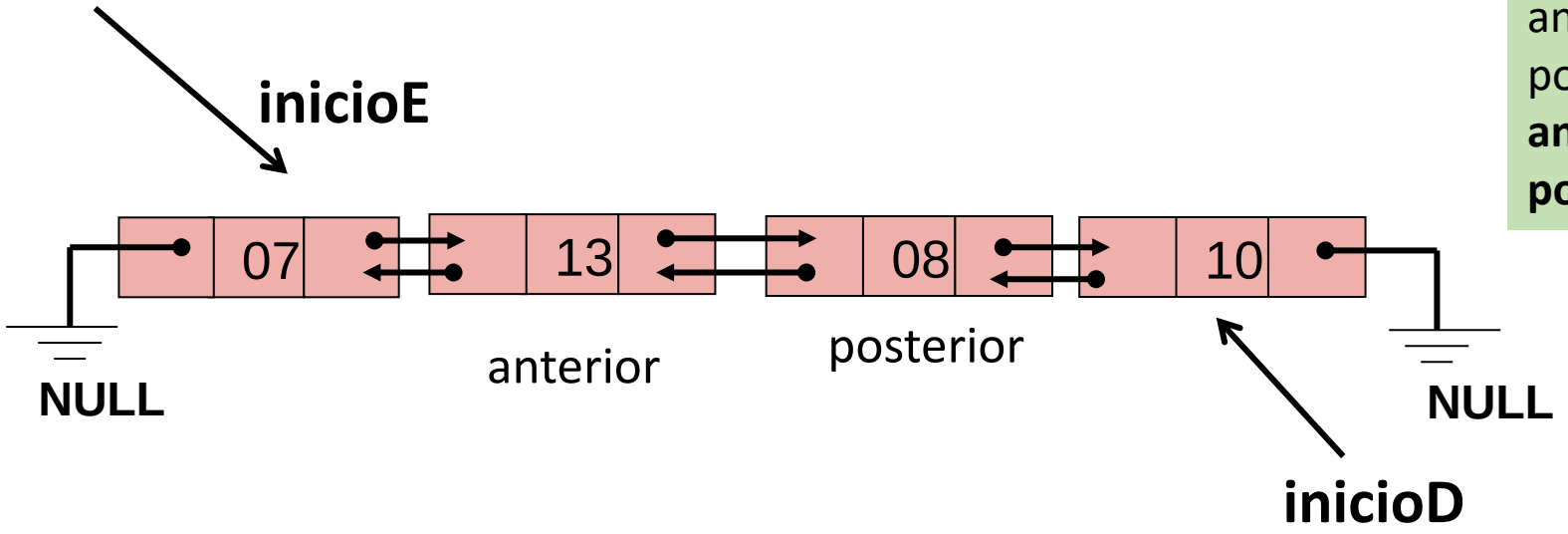
```
aux = busca_no(valor);
...
anterior = aux->ant;
posterior = aux->prox;
anterior->prox = posterior;
posterior->ant = anterior;
```



Algoritmos e Estruturas de Dados I

REMOÇÃO NA LISTA:

- O elemento a ser removido está em qualquer outra posição?




```
aux = busca_no(valor);
...
anterior = aux->ant;
posterior = aux->prox;
anterior->prox = posterior;
posterior->ant = anterior;
```

```
free(aux);
```

Algoritmos e Estruturas de Dados I

```
void remover (int valor) {
    no *aux;
    if (lista_vazia())
        printf("Lista vazia!");
    else{
        aux = buscaLDE(valor);
        if (aux == NULL)
            printf("Elemento não está na lista!");
        else{
```

Função que busca pelo elemento a ser removido.

Não encontrou o elemento.

Algoritmos e Estruturas de Dados I

```
if (inicioE == inicioD)  {
        inicioD = NULL;
        inicioE=NULL;
}
else if (aux->ant == NULL){
        inicioE = aux->prox;
        inicioE->ant=NULL;
}
```

Único elemento

Primeiro elemento da esquerda

Algoritmos e Estruturas de Dados I

Primeiro elemento da direita

Qualquer outra posição

```
            else
                if (aux->prox == NULL) {
                        inicioD = aux->ant;
                        inicioD->prox = NULL;
                }
                else {
                        no *anterior = aux->ant;
                        no *posterior = aux->prox;
                        anterior->prox = posterior;
                        posterior->ant = anterior;
            }
        free(aux);
        }
}
```

Algoritmos e Estruturas de Dados I

CONSIDERAÇÕES

- Lista que possibilita manipulação nos dois sentidos, através dos ponteiros ant e prox;
- Útil quando é preciso percorrer a lista na ordem inversa;
- Remoção de um elemento não precisa guardar anterior;
- Remoção de um elemento cujo ponteiro é informado não precisar percorrer a lista toda;
- Um conjunto maior de ligações precisam ser atualizadas.

DÚVIDAS???

Algoritmos e Estruturas de Dados I

EXERCÍCIO

1) Faça um procedimento para inserir um elemento na lista pela esquerda.

2) Faça uma função (buscaLDE) que realize a busca de um determinado valor de uma Lista Duplamente Encadeada e retorne ao programa que a chamou o nó que possui esse valor.

FIM