

Melhores momentos

AULA 7

Lista encadeadas - Motivação

Manter uma **lista** em um vetor sujeita a **remoções** e **inserções** pode dar muito trabalho com **movimentações**.

Listas encadeadas. Maneira alternativa que pode dar **menos trabalho** com **movimentações**, se estivermos disposto a gastar um pouco **mais de espaço**.

Estrutura de uma lista encadeada em C

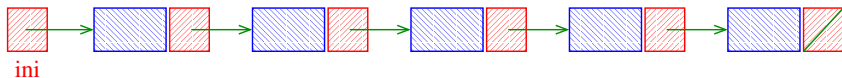
```
typedef struct celula Celula;  
struct celula {  
    int conteudo;  
    Celula *prox;  
};  
  
Celula *ini;  
/* inicialmente a lista esta vazia */  
ini = NULL;
```



ini

Imprime conteúdo de uma lista

Esta função `imprime` o `conteudo` de cada célula de uma lista encadeada `ini`.



```
void imprima (Celula *ini)
{
    Celula *p;
    for (p = ini; p != NULL; p = p->prox)
        printf("%d\n", p->conteudo);
    printf("\n");
}
```

Busca e Inserção em uma lista

Recebe lista `ini` e `insere` célula de conteúdo `x` antes da primeira célula de conteúdo `y`. Se nenhuma célula contém `y`, insere a célula com `y` no final da lista.

`Celula *`

```
buscaInsere(int x, int y, Celula *ini) {  
    Celula *p, *q, *nova;  
    nova = mallocSafe(sizeof(Celula));  
    nova->conteudo = x;  
    if (ini == NULL || ini->conteudo == y){  
        nova->prox = ini;  
        ini = nova;  
    }  
}
```

Busca e Inserção em uma lista

```
else {  
    p = ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=y){  
        p = q;  
        q = p->prox;  
    }  
    p->prox = nova;  
    nova->prox = q;  
}  
return ini;  
}
```

Chamadas de buscaInsere

```
Celula *ini, *ini2;  
ini = ini2 = NULL;
```

[...manipulação das listas ...]

```
ini = buscaInsere(22, 33, ini);  
ini2 = buscaInsere(x+1, y, ini2);  
ini2 = buscaInsere(x, 2*y, ini2);  
ini = buscaInsere(valor, meio, ini);
```


AULA 8

Mais listas encadeadas ainda



Fonte: <http://www.quickmeme.com/>

Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.

```
Celula *buscaRemove(int x, Celula *ini) {  
    Celula *p, *q;  
    if (ini == NULL) return ini;  
    if (ini->conteudo == x) {  
        q = ini;  
        ini = q->prox;  
        free(q);  
    }  
}
```

Busca e Remoção em uma lista

```
else {  
    p = ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=x){  
        p = q;  
        q = p->prox;  
    }  
    if (q != NULL) {  
        p->prox = q->prox;  
        free(q);  
    }  
}  
return ini;  
}
```

Exemplos de chamadas de buscaRemove

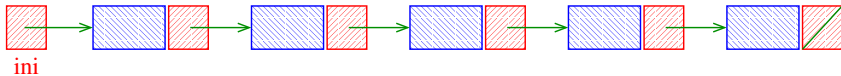
```
Celula *ini, *ini2;  
ini = ini2 = NULL;
```

[...manipulação das listas ...]

```
ini = buscaRemove(22, ini);  
ini2 = buscaRemove(x+1, ini2);  
ini2 = buscaRemove(x+y, ini2);  
ini = buscaRemove(valor, ini);
```

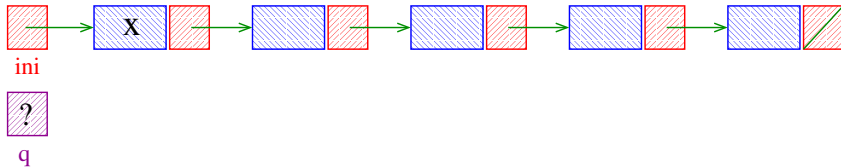
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



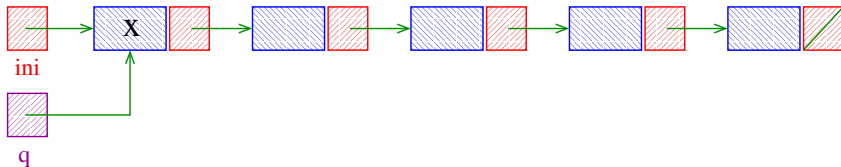
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



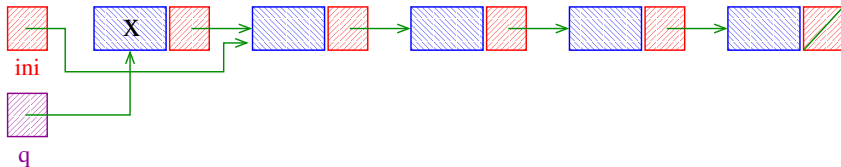
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



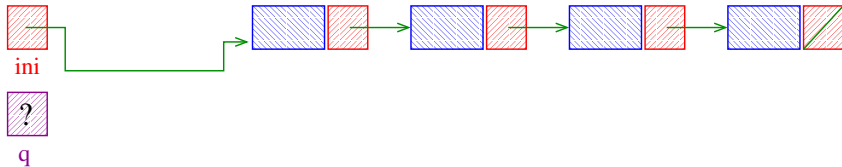
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



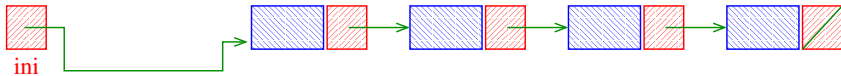
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



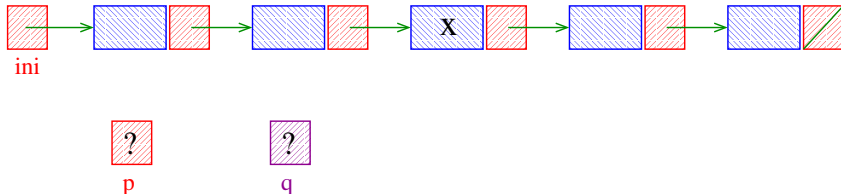
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



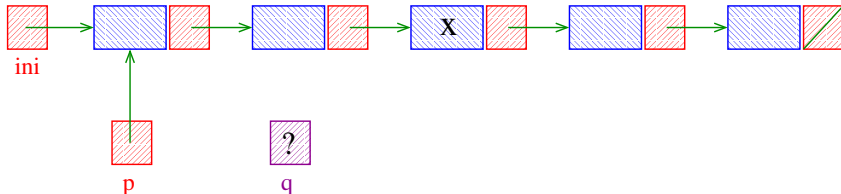
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



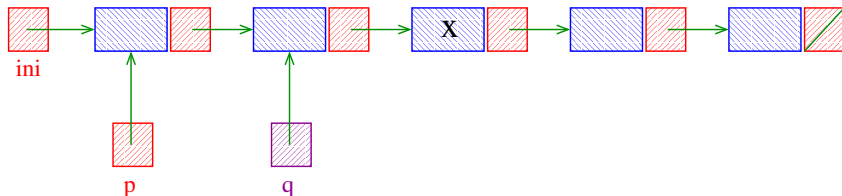
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



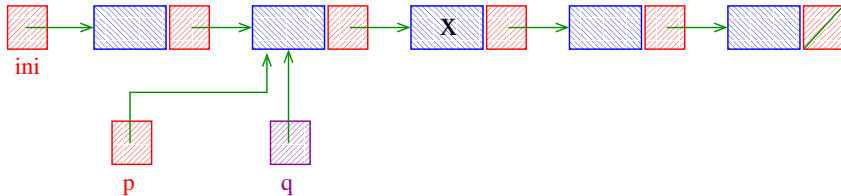
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



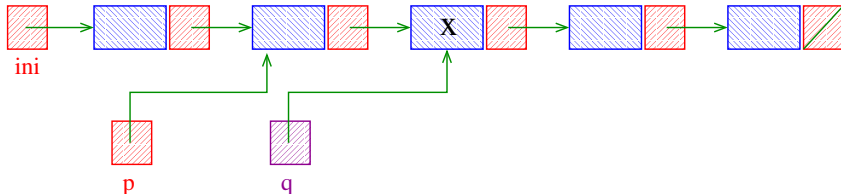
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



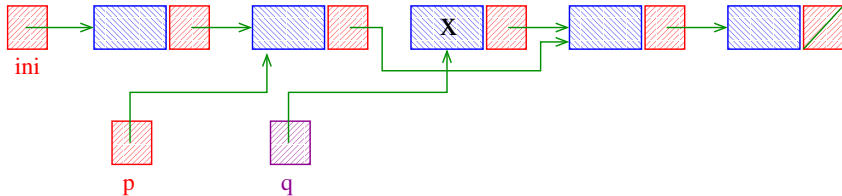
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



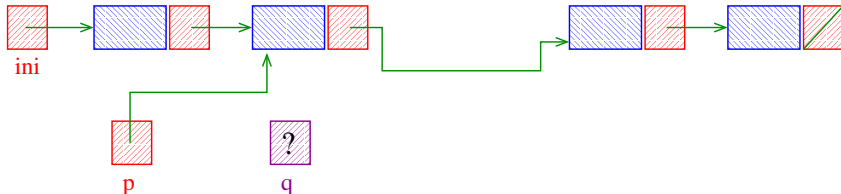
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



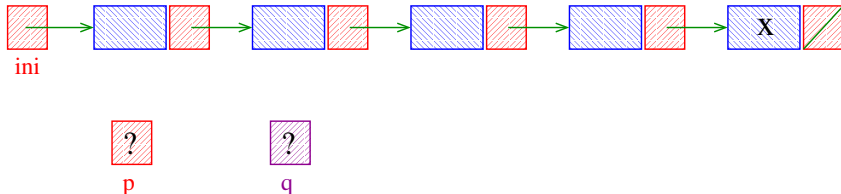
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



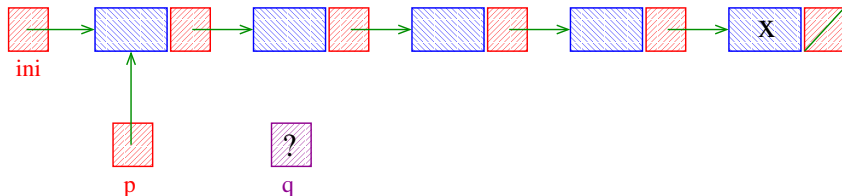
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



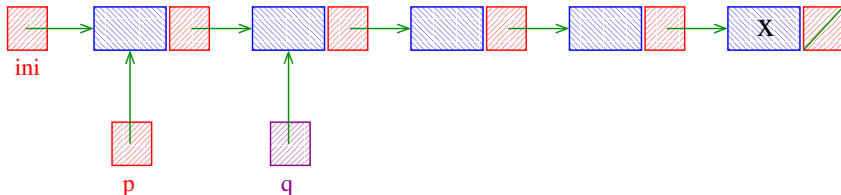
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



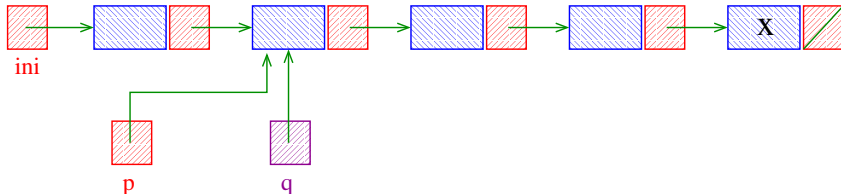
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



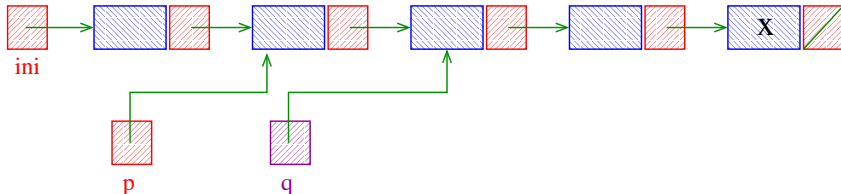
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



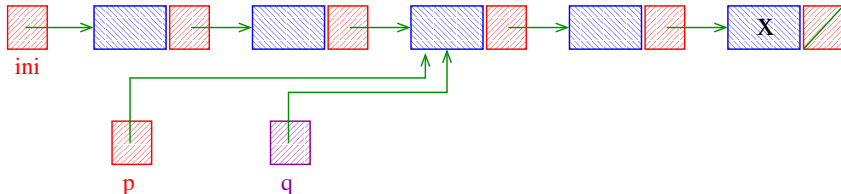
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



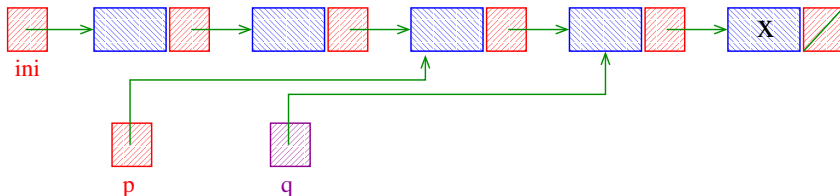
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



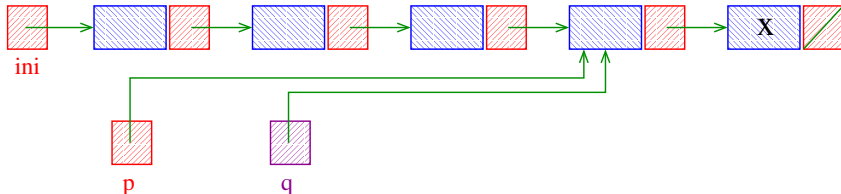
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



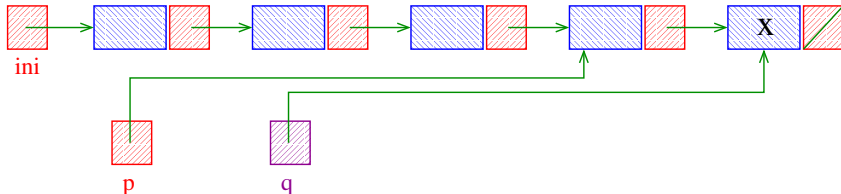
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



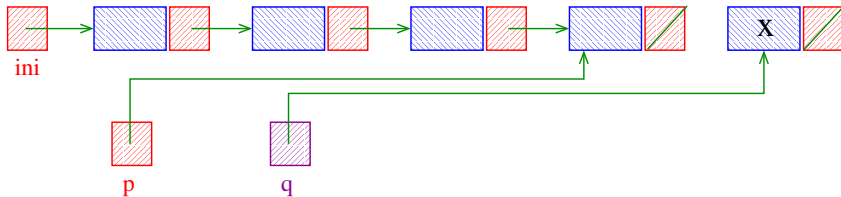
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



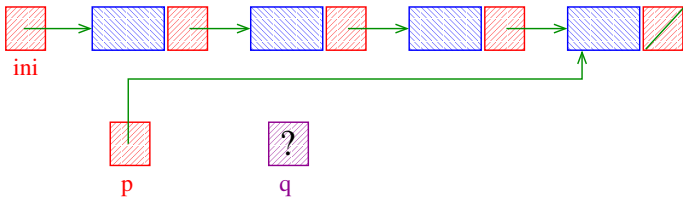
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



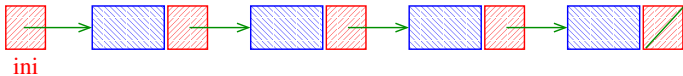
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.

```
void buscaRemove(int x, Celula **ini) {  
    Celula *p, *q;  
    if (*ini == NULL) return;  
    if ((*ini)->conteudo == x) {  
        /* -> tem mais precedencia que * */  
        q = *ini;  
        *ini = q->prox;  
        free(q);  
    }  
}
```


Busca e Remoção em uma lista

```
else {  
    p = *ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=x){  
        p = q;  
        q = p->prox;  
    }  
    if (q != NULL) {  
        p->prox = q->prox;  
        free(q);  
    }  
}  
}
```

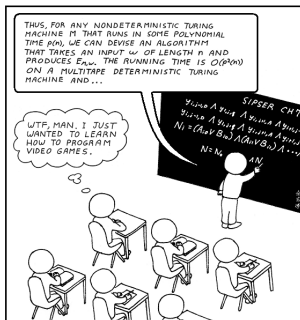
Exemplos de chamadas de buscaRemove

```
Celula *ini, *ini2;  
ini = ini2 = NULL;
```

[...manipulação das listas ...]

```
buscaRemove(22,&ini);  
buscaRemove(x+1,&ini2);  
buscaRemove(x+y,&ini2);  
buscaRemove(valor,&ini);
```

Listas com cabeça



Fonte: <http://www.hobbygamedev.com/>

PF 4, S 3.3

<http://www.ime.usp.br/~pf/algoritmos/aulas/lista.html>

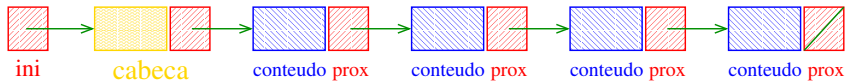
Listas encadeadas com cabeça

O conteúdo da **primeira célula** é irrelevante: ela serve apenas para **marcar o início da lista**. A primeira célula é a **cabeça** (*= head cell = dummy cell*) da lista.

A primeira célula está sempre no **mesmo lugar na memória**, mesmo que a lista fique vazia.

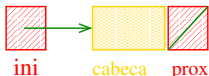
ini → **prox** == **NULL** se e somente se a **lista está vazia**.

Ilustração de uma **lista encadeada** “**com cabeça**”:



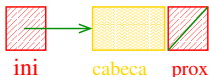
Estrutura de uma lista com cabeça

```
struct celula {  
    int conteudo;  
    struct celula *prox;  
};  
typedef struct celula Celula;  
  
Celula *ini, cabeca;  
/* inicialmente a lista esta vazia */  
cabeca.prox = NULL;  
ini = &cabeca;
```



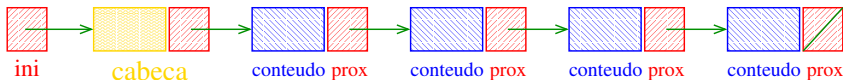
Estrutura de uma lista com cabeça

```
struct celula {  
    int conteudo;  
    struct celula *prox;  
};  
typedef struct celula Celula;  
  
Celula *ini;  
/* inicialmente a lista esta vazia */  
ini = malloc(sizeof(Celula));  
ini->prox = NULL;
```



Imprime conteúdo de uma lista com cabeça

Esta função `imprime` o `conteudo` de cada célula de uma lista encadeada `com cabeça ini`.



```
void imprima (Celula *ini)
{
    Celula *p;
    for (p=ini->prox; p != NULL; p=p->prox)
        printf("%d\n", p->conteudo);
}
```

Busca em uma lista com cabeça

Esta função **recebe** um inteiro **x** e uma lista **ini**. A função **devolve** o endereço de uma célula que contém **x**. Se tal célula não existe, a função **devolve** **NULL**.

```
Celula *busca (int x, Celula *ini)
{
    Celula *p;
    p = ini->prox;
    while (p != NULL && p->conteudo != x)
        p = p->prox;
    return p;
}
```

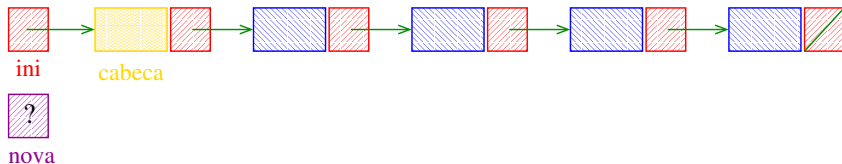

Inserção no início de uma lista com cabeça

Cria uma célula para guardar um elemento **x** e **insere** esta célula no início da lista com cabeça **ini**.

```
void insere (int x, Celula *ini) /*void!*/  
{  
    Celula *nova;  
    nova = mallocSafe(sizeof(Celula));  
    nova->conteudo = x;  
    nova->prox = ini->prox;  
    ini->prox = nova;  
}
```

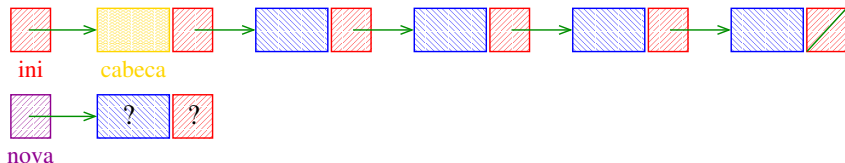
Inserção no início de uma lista com cabeça

Cria uma célula para guardar um elemento **x** e
insere esta célula no início da lista com cabeça **ini**.



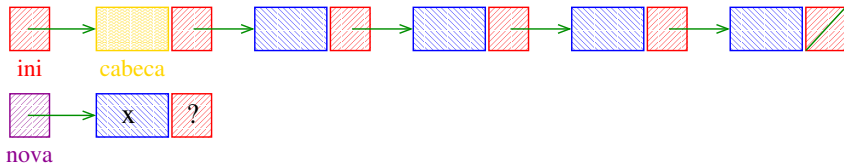
Inserção no início de uma lista com cabeça

Cria uma célula para guardar um elemento **x** e
insere esta célula no início da lista com cabeça **ini**.



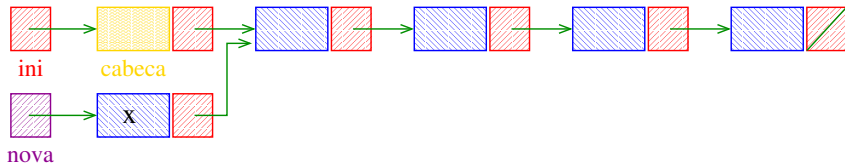
Inserção no início de uma lista com cabeça

Cria uma célula para guardar um elemento **x** e
insere esta célula no início da lista com cabeça **ini**.



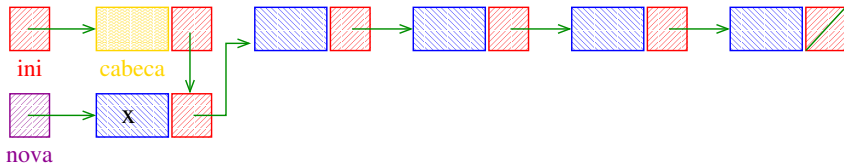
Inserção no início de uma lista com cabeça

Cria uma célula para guardar um elemento **x** e
insere esta célula no início da lista com cabeça **ini**.



Inserção no início de uma lista com cabeça

Cria uma célula para guardar um elemento **x** e **insere** esta célula no início da lista com cabeça **ini**.



Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça `ini` que contém o elemento `x`.

```
Celula *buscaRemove (int x, Celula *ini){  
    Celula *p, *q;  
    if (ini == NULL) return ini;  
    if (ini->conteudo == x) {  
        q = ini;  
        ini = q->prox;  
        free(q);  
    }  
}
```

Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça *ini* que contém o elemento *x*.

```
Celula *buscaRemove (int x, Celula *ini){  
    Celula *p, *q;  
    if (ini == NULL) return ini;  
    if (ini->conteudo == x) {  
        q = ini;  
        ini = q->prox;  
        free(q);  
    }
```


Busca e Remoção em uma lista com cabeça

```
else {  
    p = ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=x){  
        p = q;  
        q = p->prox;  
    }  
    if (q != NULL) {  
        p->prox = q->prox;  
        free(q);  
    }  
}  
return ini;  
}
```

Busca e Remoção em uma lista com cabeça

```
else {  
    p = ini;  
    q = p->prox;  
    while (q != NULL && q->conteudo != x) {  
        p = q;  
        q = p->prox;  
    }  
    if (q != NULL) {  
        p->prox = q->prox;  
        free(q);  
    }  
}  
return ini;  
}
```

Busca e Remoção em uma lista com cabeça

```
void buscaRemove (int x, Celula *ini) {  
    Celula *p, *q;  
    p = ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=x){  
        p = q;  
        q = p->prox;  
    }  
    if (q != NULL) {  
        p->prox = q->prox;  
        free(q);  
    }  
}
```

Exemplos de chamadas de buscaRemove

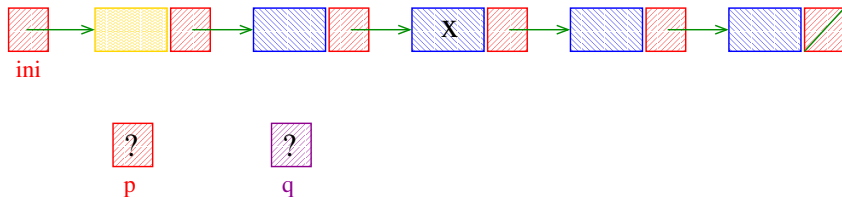
```
Celula *ini, *ini2;  
Celula cabeca;  
ini = &cabeca  
cabeca.prox = NULL;  
ini2 = mallocSafe(sizeof(Celula));  
ini2->prox = NULL;
```

[...manipulação das listas ...]

```
buscaRemove(22,&cabeca);  
buscaRemove(33,ini);  
buscaRemove(x+1,ini2);  
buscaRemove(x+y,ini2);  
buscaRemove(valor,ini);
```

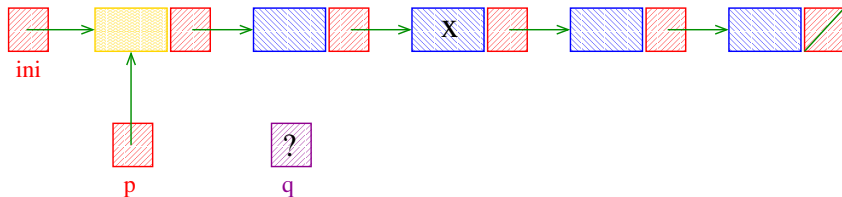
Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça *ini* que contém o elemento *x*.



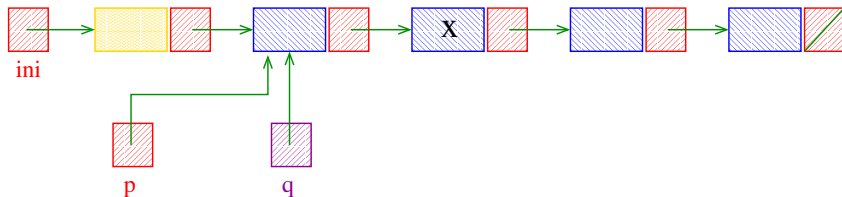
Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça *ini* que contém o elemento *x*.



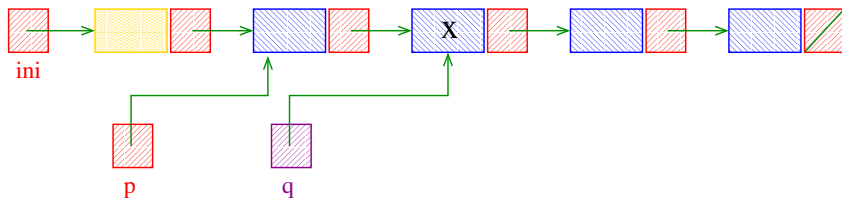
Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça *ini* que contém o elemento *x*.



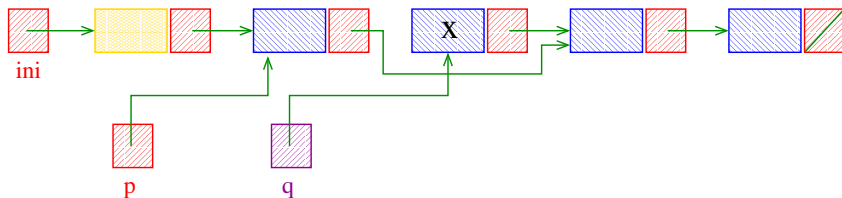
Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça *ini* que contém o elemento *x*.



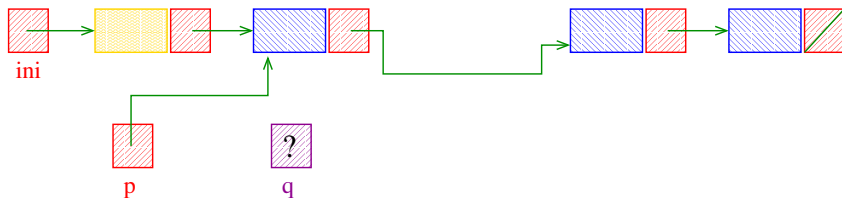
Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça *ini* que contém o elemento *x*.



Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça *ini* que contém o elemento *x*.



Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça `ini` que contém o elemento `x`.



Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça `ini` e insere uma célula de conteúdo `x` antes da primeira célula de conteúdo `y`. Se nenhuma célula contém `y`, insere a célula com `y` no final da lista.

`Celula *`

```
buscaInsere(int x, int y, Celula *ini) {  
    Celula *p, *q, *nova;  
    nova = mallocSafe(sizeof(Celula));  
    nova->conteudo = x;  
    if (ini == NULL || ini->conteudo == y){  
        nova->prox = ini;  
        ini = nova;  
    }  
}
```

Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça `ini` e insere uma célula de conteúdo `x` antes da primeira célula de conteúdo `y`. Se nenhuma célula contém `y`, insere a célula com `y` no final da lista.

`Celula *`

```
buscaInsere(int x, int y, Celula *ini) {  
    Celula *p, *q, *nova;  
    nova = mallocSafe(sizeof(Celula));  
    nova->conteudo = x;  
    if (ini == NULL || ini->conteudo == y){  
        nova->prox = ini;  
        ini = nova;  
    }  
}
```

Busca e Inserção em uma lista com cabeça

```
else {  
    p = ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=y){  
        p = q;  
        q = p->prox;  
    }  
    p->prox = nova;  
    nova->prox = q;  
}  
return ini;  
}
```

Busca e Inserção em uma lista com cabeça

```
else {  
    p = ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=y){  
        p = q;  
        q = p->prox;  
    }  
    p->prox = nova;  
    nova->prox = q;  
}  
return ini;  
}
```


Busca e Inserção em uma lista com cabeça

void

```
buscaInsere(int x, int y, Celula *ini){  
    Celula *p, *q, *nova;  
    nova = mallocSafe(sizeof(Celula));  
    nova->conteudo = x;  
    p = ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=y){  
        p = q;  
        q = p->prox;  
    }  
    p->prox = nova;  
    nova->prox = q;  
}
```

Exemplos de chamadas de buscaInsere

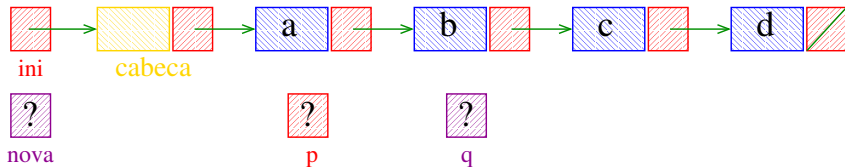
```
Celula *ini, *ini2;  
Celula cabeca;  
ini = &cabeca  
cabeca.prox = NULL;  
ini2 = mallocSafe(sizeof(Celula));  
ini2->prox = NULL;
```

[...manipulação das listas ...]

```
buscaInsere(22,24,&cabeca);  
buscaInsere(33,-10,ini);  
buscaInsere(x+1,y-5,ini2);  
buscaInsere(x+y,ini2);  
buscaInsere(valor,ini);
```

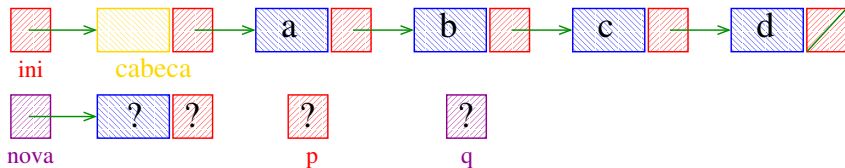
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e **insere** uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



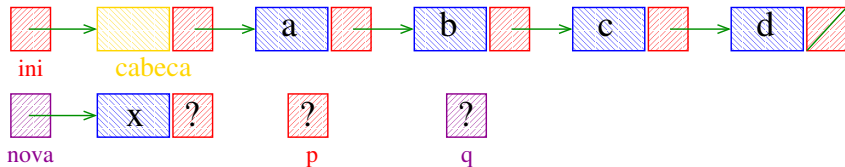
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e insere uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



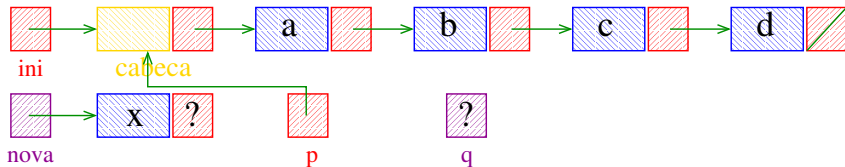
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e **insere** uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



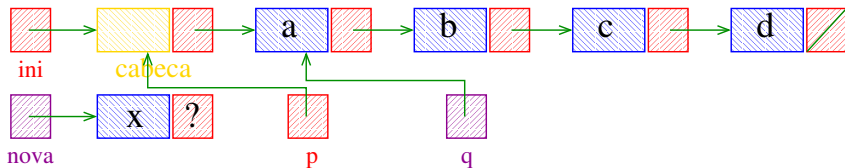
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e insere uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



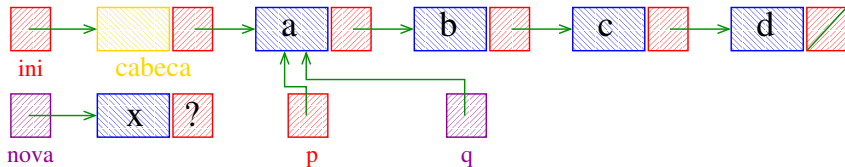
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e insere uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



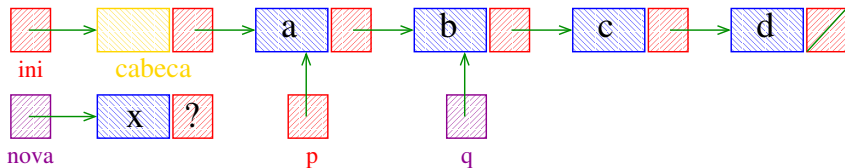
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e **insere** uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



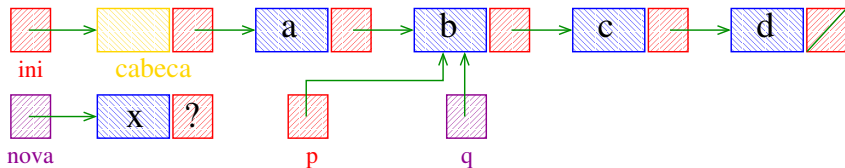
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e insere uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



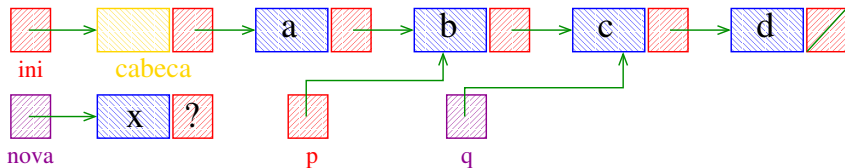
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e insere uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



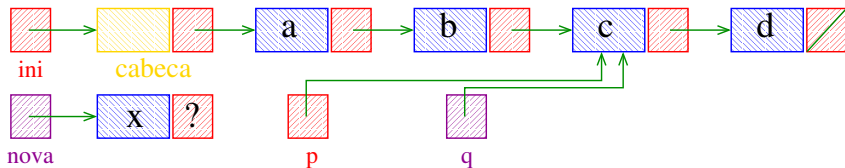
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e **insere** uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



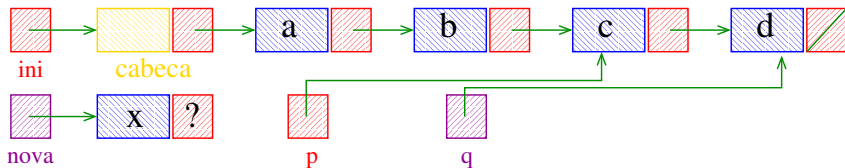
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e insere uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



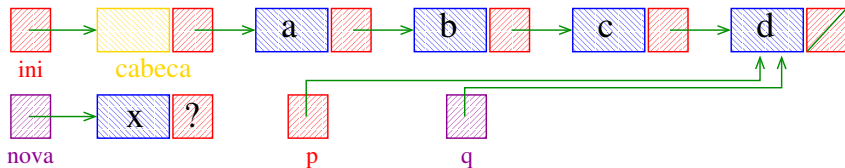
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e **insere** uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



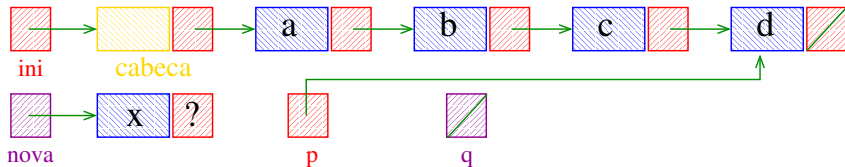
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e **insere** uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



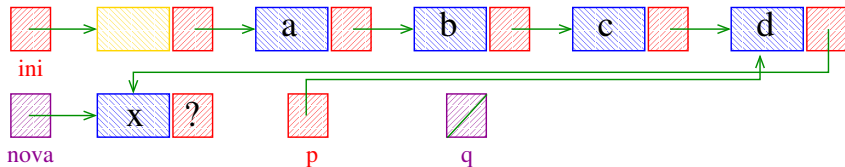
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e **insere** uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



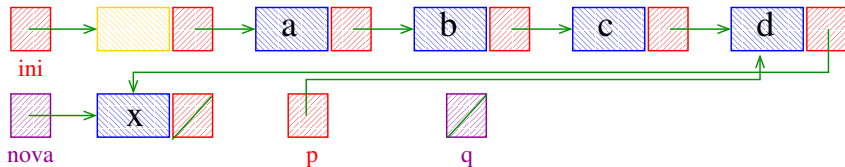
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e insere uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



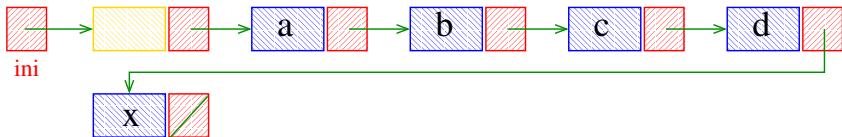
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça **ini** e insere uma célula de conteúdo **x** antes da primeira célula de conteúdo **y**. Se nenhuma célula contém **y**, insere a célula com **y** no final da lista.



Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



Inversão de uma lista



Fonte: <http://geeksandglitter.wordpress.com/>

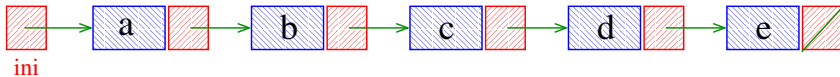
PF 4, S 3.3

<http://www.ime.usp.br/~pf/algoritmos/aulas/lista.html>

Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando **apenas os ponteiros** (!).

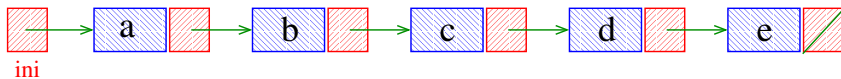
Lista **antes** da inversão:



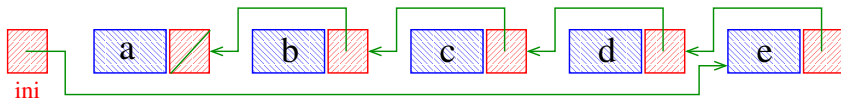
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando **apenas os ponteiros** (!).

Lista **antes** da inversão:



Lista **depois** da inversão:



Inversão de uma lista

Recebe uma lista `ini` e `inverte` a ordem de suas células alterando apenas os ponteiros.

```
Celula *inverta(Celula *ini) {  
    Celula *p, *q, *r;  
    p = NULL; q = ini;  
    while (q != NULL) {  
        r = q->prox;  
        q->prox = p;  
        p = q;  
        q = r;  
    }  
    return p;  
}
```

Exemplos de chamadas

```
Celula *ini, *ini2;  
ini = ini2 = NULL;
```

[...manipulação da lista ...]

```
ini = inverta(ini);  
ini2 = inverta(ini2);
```

Exemplos de chamadas

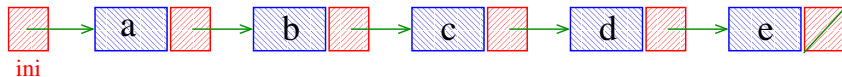
```
Celula *ini, *ini2;  
Celula cabeca;  
ini = &cabeca  
cabeca.prox = NULL;  
ini2 = mallocSafe(sizeof(Celula));  
ini2->prox = NULL;
```

[...manipulação das listas ...]

```
ini->prox= inverta(ini->prox);  
cabeca.prox = inverta(cabeca.prox);  
ini->prox = inverta(cabeca.prox);  
cabeca.prox = inverta(ini->prox);  
ini2->prox= inverta(ini2->prox);
```

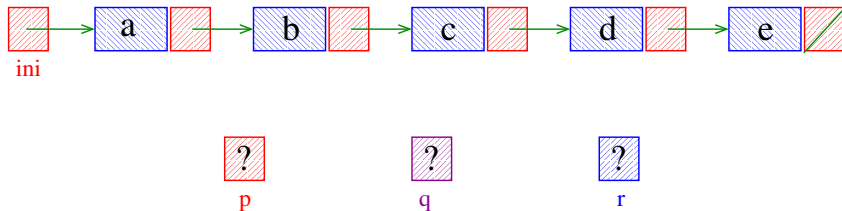

Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



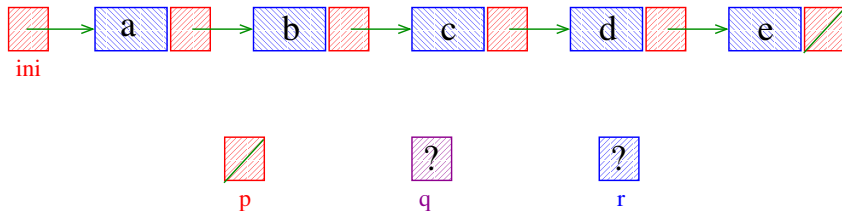
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



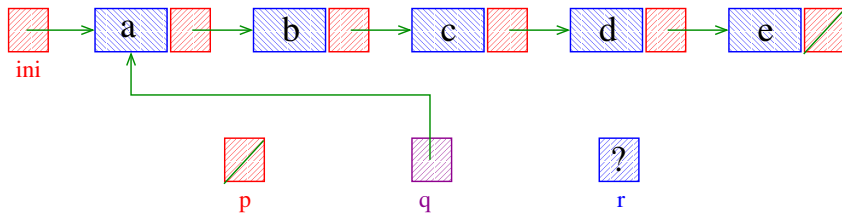
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



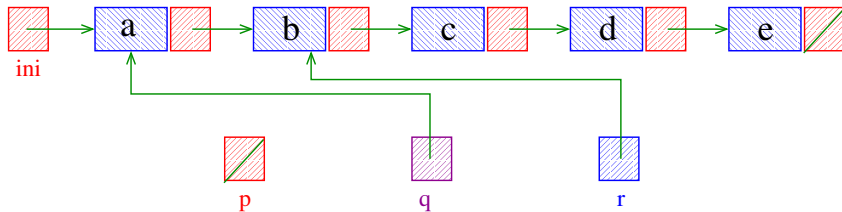
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



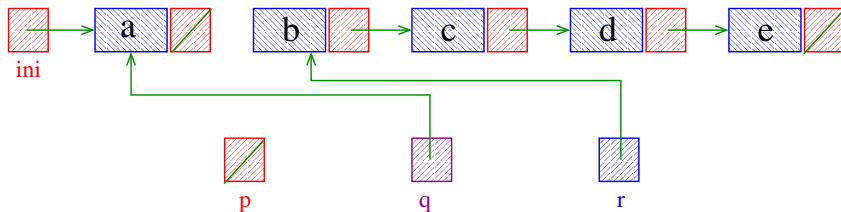
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



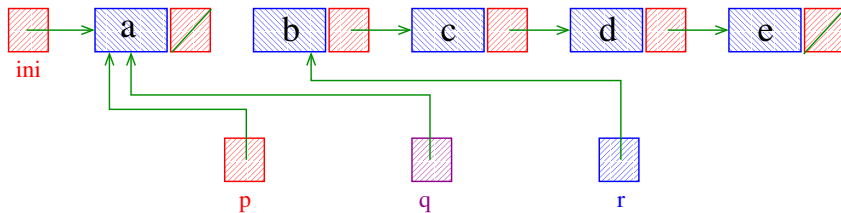
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



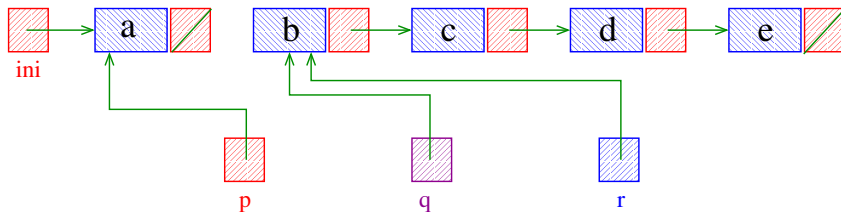
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



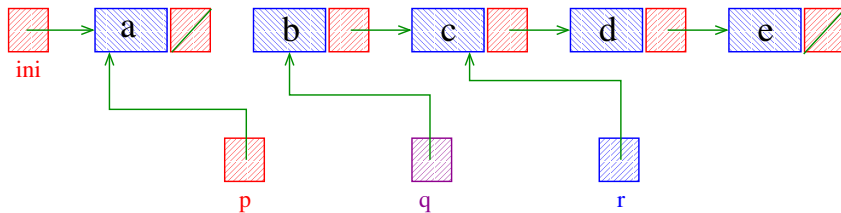
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



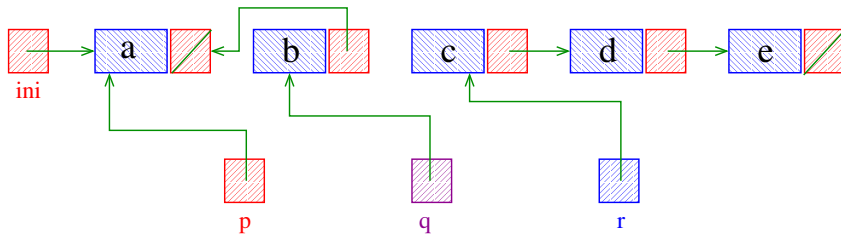
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



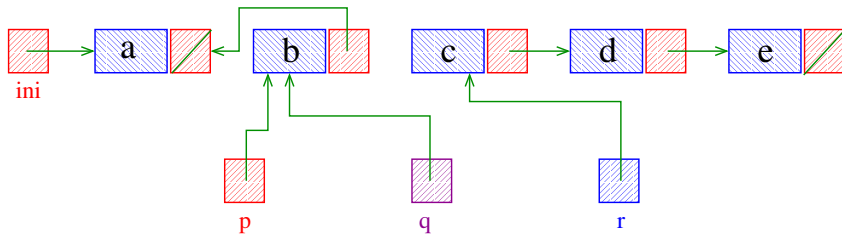
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



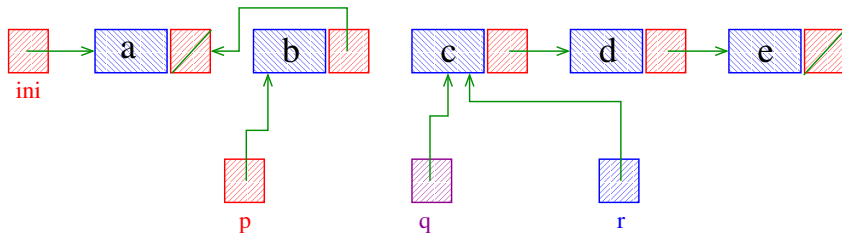
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



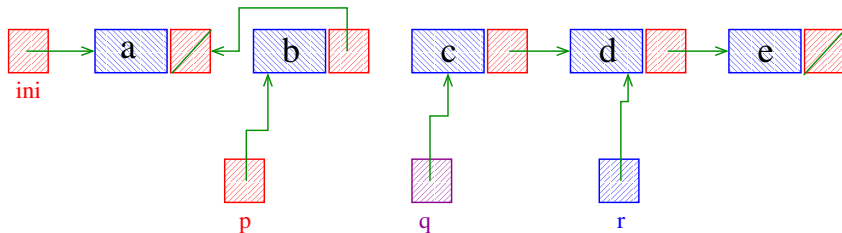
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



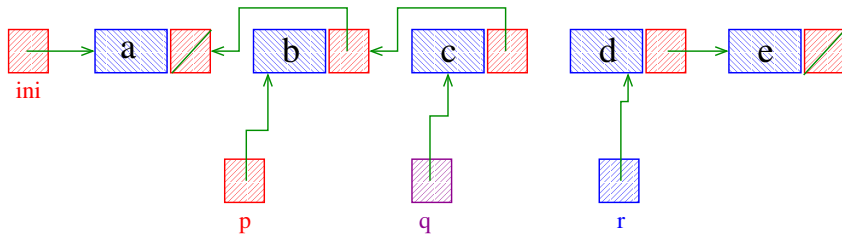
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



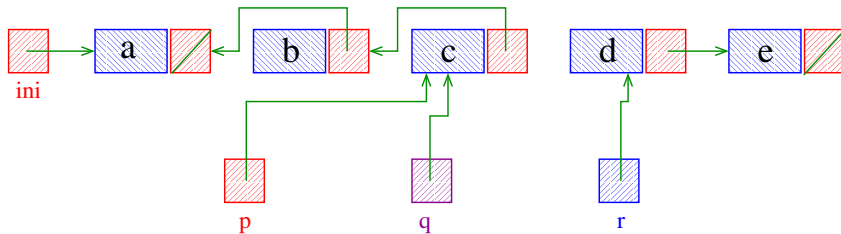
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



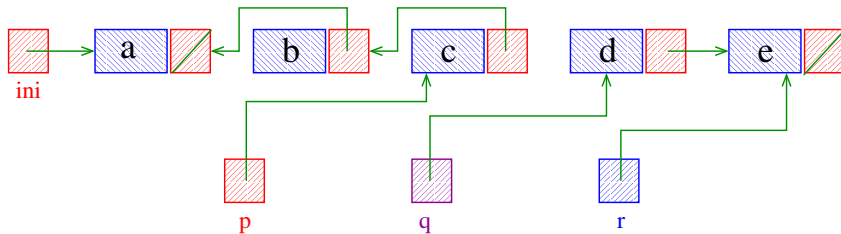
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



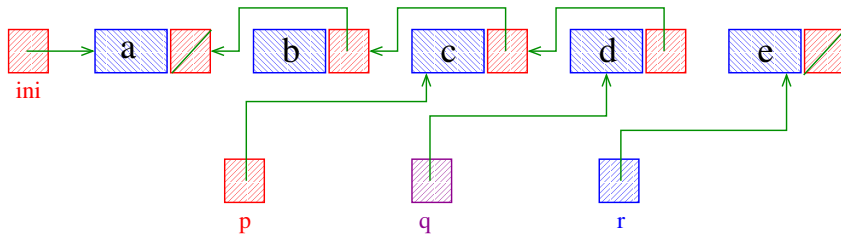
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



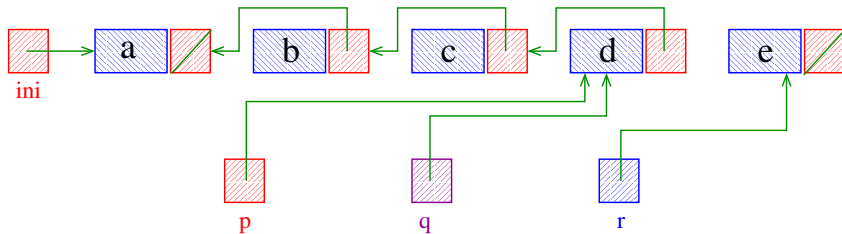
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



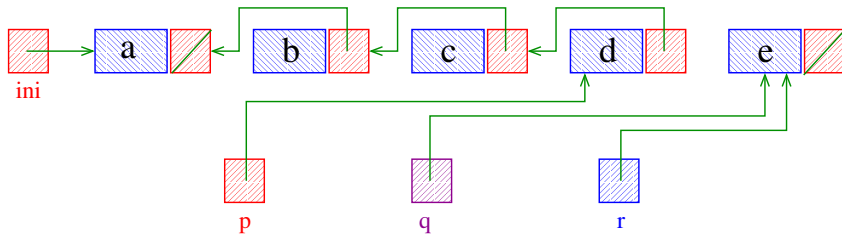
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



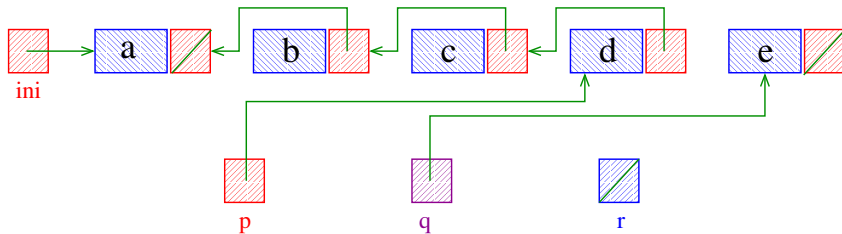
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



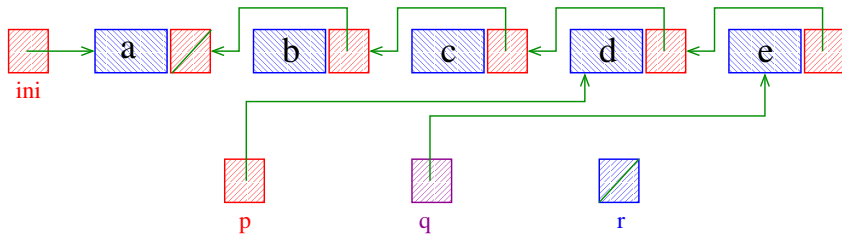
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



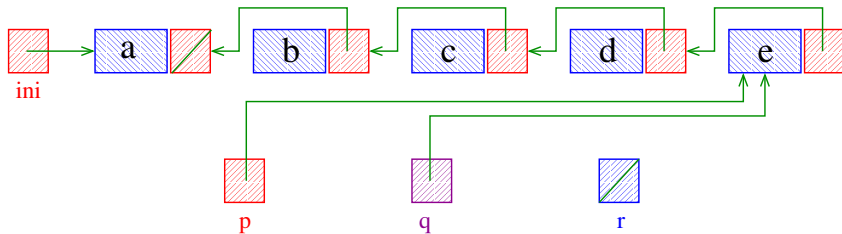
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



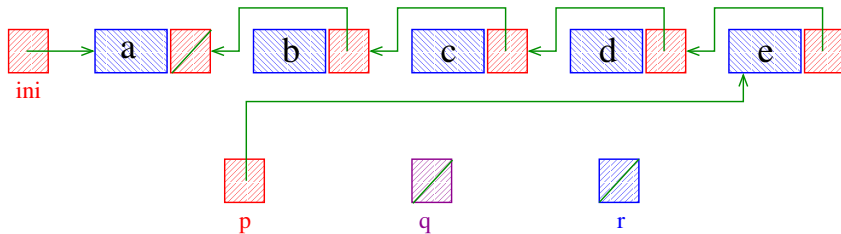
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



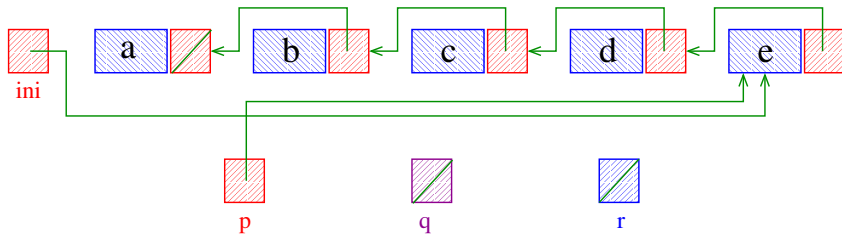
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



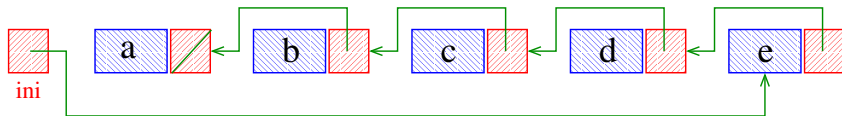
Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



Inversão de uma lista

Recebe uma lista **ini** e **inverte** a ordem de suas células alterando apenas os ponteiros.



Consumo de tempo e espaço

O consumo de tempo da função `inverte(ini)` é proporcional a `n`, onde `n` é o número de células na lista `ini`.

O espaço extra utilizado pela função `inverte(ini)` é constante, ou seja, independe do número de células na lista `ini`.

Inversão de uma lista

Recebe uma lista `ini` e `inverte` a ordem de suas células alterando apenas os ponteiros.

```
void inverta(Celula **ini) {  
    Celula *p, *q, *r;  
    p = NULL; q = *ini;  
    while (q != NULL) {  
        r = q->prox;  
        q->prox = p;  
        p = q;  
        q = r;  
    }  
    *ini = p;  
}
```

Exemplos de chamadas

```
Celula *ini, *ini2;  
ini = ini2 = NULL;
```

[...manipulação da lista ...]

```
inverte(&ini);  
inverte(&ini2);
```

Exemplos de chamadas

```
Celula *ini, *ini2;  
Celula cabeca;  
ini = &cabeca  
cabeca.prox = NULL;  
ini2 = mallocSafe(sizeof(Celula));  
ini2->prox = NULL;
```

[...manipulação das listas ...]

```
inverte(&ini->prox);  
inverte(&cabeca.prox);  
inverte(&ini->prox);  
inverte(&ini2->prox);
```