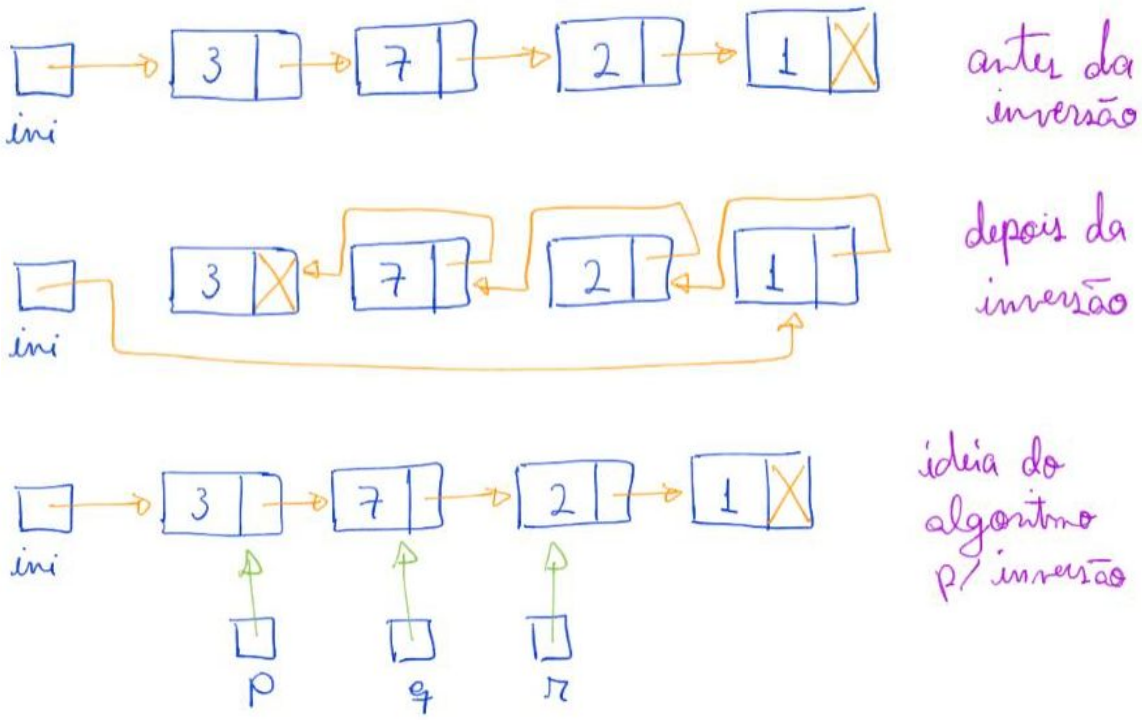
AED1 - Aula 13
Inversão de uma lista, listas encadeadas em vetores

Inversão de uma lista



Ao longo do algoritmo:

- q aponta para a célula corrente,
- p aponta para a anterior,
- r aponta para a seguinte.

Em cada iteração:

- p->prox passa a aponta para q

Termina o laço quando:

- q aponta para NULL,
- caso em que p aponta para a última célula,
    - que agora é a primeira.

Sem nó cabeça:

```
Celula *inverta1(Celula *lst)
{
   Celula *p, *q, *r;
   p = NULL;
   q = lst;
   while (q != NULL)
   {
       r = q->prox;
       q->prox = p;
```

```
        p = q;
        q = r;
    }
    return p;
}
```

- exemplos de uso:

```
    ini = inverta1(ini);
    ini->prox = inverta1(ini->prox);
```

```
void inverta2(Celula **lst)
{
    Celula *p, *q, *r;
    p = NULL;
    q = *lst;
    while (q != NULL)
    {
        r = q->prox;
        q->prox = p;
        p = q;
        q = r;
    }
    *lst = p;
}
```

- exemplos de uso:

```
    inverta2(&ini);
    inverta2(&ini->prox);
```

Com nó cabeça:

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

- exemplos de uso:

```
    inverta(ini);
    inverta(ini->prox);
```

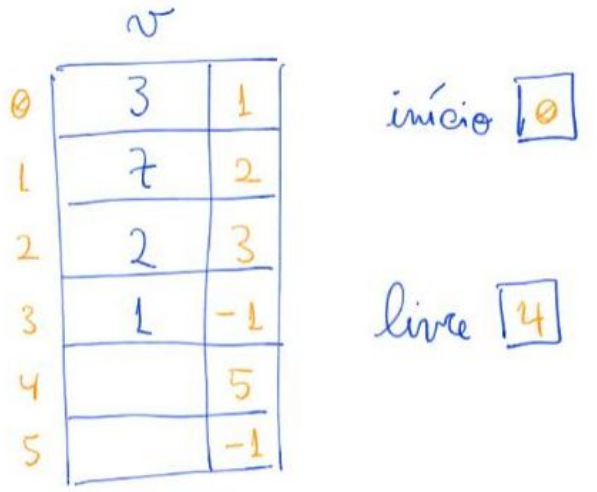
Qual a eficiência de tempo destes algoritmos?

- O(n), sendo n o número de elementos da lista.

Qual a eficiência de espaço destes algoritmos?

- O(1), pois só usa auxiliares cujo tamanho não é proporcional à lista.

Listas encadeadas em vetores




```
#define MAX 1000
#define NULO -1

typedef struct celula Celula;
struct celula
{
   int conteudo;
   int prox;
};

   Celula v[MAX];
   int inicio, livre;

   // Inicializa a lista livres com todas as posições
   for (i = 0; i < MAX - 1; i++)
       v[i].prox = i + 1;
   v[MAX - 1].prox = NULO;
   livre = 0;
   // e declara a lista de fato vazia
   inicio = NULO;

void imprime(Celula v[], int inicio)
{
   int p;
   for (p = inicio; p != NULO; p = v[p].prox)
       printf("%d ", v[p].conteudo);
```

```
    printf("\n");
}
```

- exemplo de uso:

```
    imprime(v, inicio);

int busca(Celula v[], int inicio, int x)
{
    int p;
    for (p = inicio; p != NULO && v[p].conteudo != x; p = v[p].prox)
        ;
    return p;
}
```

- exemplo de uso:

```
    int p = busca(v, inicio, 10);
    if (p != NULO)
        printf("%d\n", v[p].conteudo);
    else
        printf("nao encontrou\n");

int selecao(Celula v[], int inicio, int k)
{
    int p, q;
    for (p = inicio, q = 0; p != NULO && q < k; p = v[p].prox, q++)
        ;
    return p;
}
```

- exemplo de uso:

```
    printf("Selecionando elementos\n");
    int q = selecao(v, inicio, 10);
    if (q != NULO)
        printf("%d\n", v[q].conteudo);
    else
        printf("nao existe\n");

// insere o elemento x no inicio da lista
void insereInicio(Celula v[], int *inicio, int x, int *livre)
{
    int nova;
    nova = *livre;
    *livre = v[*livre].prox;
    v[nova].conteudo = x;
    v[nova].prox = *inicio;
    *inicio = nova;
}
```

- exemplo de uso:

```
    printf("Insere n elementos na lista\n");
    for (i = 0; i < n; i++)
```

```
        insereInicio(v, &inicio, i, &livre);
    imprime(v, inicio);

// insere o elemento x entre v[p] e v[p].prox
void insereDepois(Celula v[], int p, int x, int *livre)
{
    int nova;
    nova = *livre;
    *livre = v[*livre].prox;
    v[nova].conteudo = x;
    v[nova].prox = v[p].prox;
    v[p].prox = nova;
}
```

- exemplos de uso:

```
    insereDepois(v, inicio, -77, &livre);
    insereDepois(v, v[inicio].prox, -44, &livre);

// remove a celula do inicio
void removeInicio(Celula v[], int *inicio, int *livre)
{
    int q = *inicio;
    *inicio = v[q].prox;
    v[q].prox = *livre;
    *livre = q;
}
```

- exemplo de uso:

```
    removeInicio(v, &inicio, &livre);

// remove a celula de índice v[p].prox
void removeProximo(Celula v[], int p, int *livre)
{
    int q = v[p].prox;
    v[p].prox = v[q].prox;
    v[q].prox = *livre;
    *livre = q;
}
```

- exemplos de uso:

```
    removeProximo(v, inicio, &livre);
    removeProximo(v, v[inicio].prox, &livre);
```

Qual a eficiência de tempo destes algoritmos?

- Imprime, busca e seleção são O(n), sendo n o número de elementos da lista.
- As inserções e remoções são O(1).

Qual a eficiência de espaço destes algoritmos?

- O(1), pois só usa auxiliares cujo tamanho não é proporcional à lista.