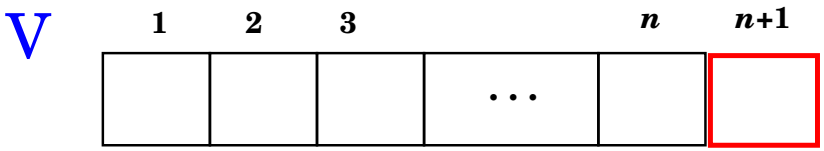
Aula 6: Manipulação de Listas Lineares

- Algoritmo de inserção
- Algoritmo de remoção
- Complexidade dos métodos

Algoritmo de Inserção

Objetivo: inserir o elemento *x* no vetor V com *n* elementos.



Inserir Voltar

- A inserção será feita no final do vetor V.

Prevenção de "overflow" (Sobrecarga)

- Seja M o número máximo de elementos que o vetor V pode armazenar.
- Somente poderemos efetuar a inserção de x quando $n < M$, isto é, quando ainda existe espaço disponível no vetor V.
- Quando $n = M$, o vetor V está com sua capacidade de armazenamento esgotada. Nessa situação, a inserção de um novo elemento em V provocaria *overflow* - o que deve ser evitado.

Prevenção de Elementos Repetidos

- Para evitar a existência de elementos repetidos, deve-se efetuar previamente uma busca do elemento x que desejamos inserir em V.

- Se a busca retornar sinalizando que x já pertence a V, não efetuaremos a inserção.

Prevenção de Elementos Repetidos

Algoritmo: Inserção de um elemento x no vetor V

```
se n < M                          %  V  possui espaço livre
então
    se BUSCA1(x) = 0              %  x  não é repetido
    então
        V[n+1] := x;
        n      := n + 1
    senão
            "elemento já existe na tabela"
senão
    "overflow"
```

Complexidade: $O(n)$ (Por que?)

Exercício

Suponha que o vetor V esteja ordenado, e deseja-se inserir o elemento x em V respeitando-se a ordem dos elementos (isto significa que x pode ser inserido no meio do vetor V).

Como você modificaria o algoritmo anterior a fim de atender esta exigência?

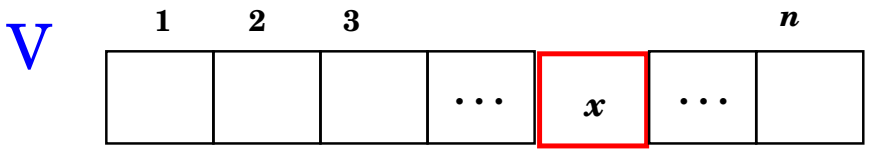
Tempo: 8 minutos

Solução

```
se n = M
então  "Overflow"
senão
    V[n+1] : = x;                    % sentinela
    i : = 1;
    enquanto  V[i]  <  x   faça
        i  : =  i + 1;
    se i < n + 1 e V[i] ≠  x   % inserir  x  no meio
    então              %executar movimentação de elementos
        para  j = n ... i  passo - 1
            V[j+1] : =  V[j] ;
        V[i] : =  x;
        n : =  n + 1;
    senão
        se  i = n + 1          %  x  é o maior de todos
            então   n : = n + 1
            senão  "elemento já existe na tabela"
```

Algoritmo de Remoção

Objetivo: retirar o elemento x do vetor V com n elementos.



Remover Voltar

Observe que a remoção de x pode deixar uma posição vaga dentro de V. Nesse caso, é preciso reagrupar os elementos restantes dentro do vetor.

Algoritmo de Remoção

Algoritmo: Remoção de um elemento x do vetor V

```
se n = 0
então  "Underflow"              %  V está vazio - nada a fazer
senão
    j  : =  BUSCA1(x);          % verificar se x ∈ V
    se  j  = 0
    então       "elemento não está na tabela" % nada a fazer
   senão            %  reagrupamento dos elementos restantes
        para   i  =  j ... n - 1  faça
            V[i] : = V[i+1];
        n : = n - 1;
```

Complexidade: $O(n)$

Exercício Final

Responda: No caso de o vetor V estar ordenado, a remoção de um elemento apresentaria modificações significativas em relação ao algoritmo de remoção exposto?