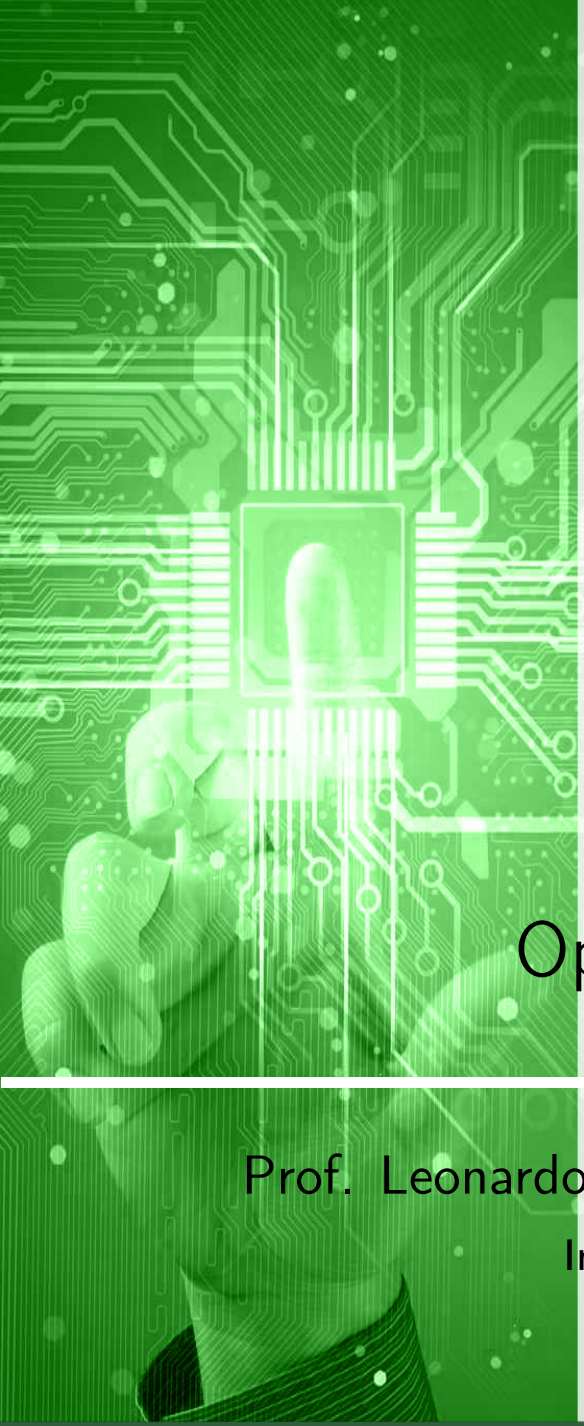


INSTITUTO FEDERAL
Sudeste de Minas Gerais
Campus Manhuaçu

Estruturas de Dados I
Operações básicas com vetores

Prof. Leonardo C. R. Soares - *leonardo.soares@ifsudestemg.edu.br*

Instituto Federal do Sudeste de Minas Gerais

1 de novembro de 2024

O que precisamos saber antes de começarmos

Base necessária

- Definir métodos em Java;
- Definição e utilização de vetores em Java;
- Tratamento de erros em Java.

Introdução

- Algumas operações sobre estruturas de dados homogêneas são rotineiras. Nesta aula abordaremos de forma superficial algumas dessas operações.

Inserindo dados em um vetor

- Os dados podem ser inseridos em três locais do vetor:

Inserindo dados em um vetor

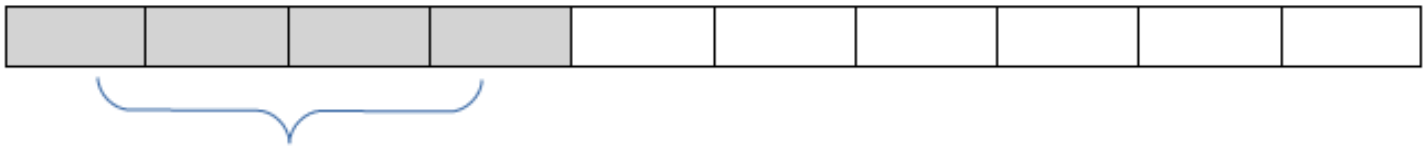
- Os dados podem ser inseridos em três locais do vetor:
- No ínicio;
- No final;
- Entre dois elementos quaisquer;

Inserindo dados em um vetor

- Os dados podem ser inseridos em três locais do vetor:
- No ínicio;
- No final;
- Entre dois elementos quaisquer;
- Cada uma dessas operações demanda um reajuste na estrutura de dados.

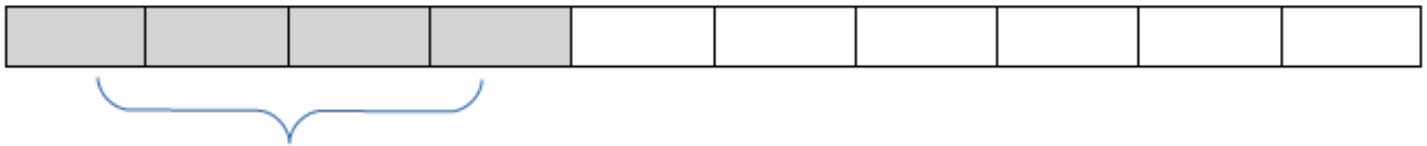
Inserção no início

- Há espaço disponível?



Inserção no início

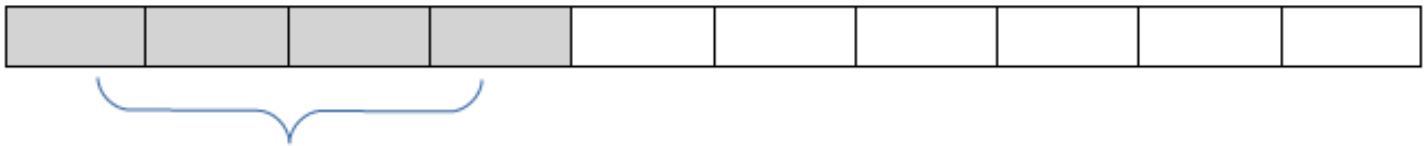
- Há espaço disponível?



- Move-se os elementos já cadastrados para a posição $i + 1$ (onde i é a posição atual do elemento);

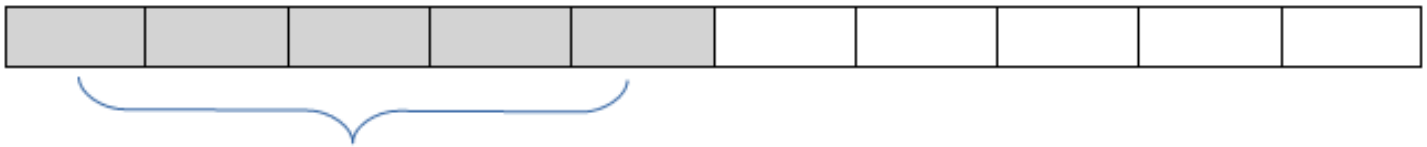
Inserção no início

- Há espaço disponível?



- Move-se os elementos já cadastrados para a posição $i + 1$ (onde i é a posição atual do elemento);

- Insere-se o novo elemento na posição 0.



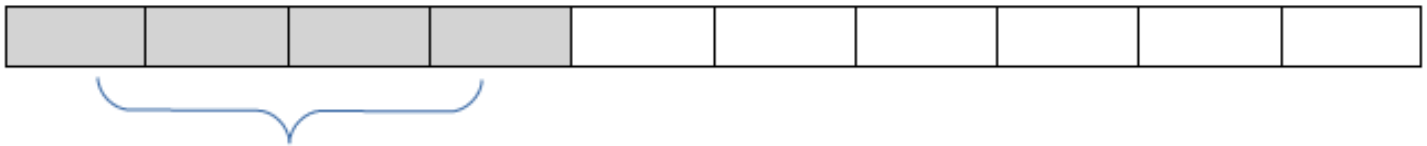
Inserção no início

Vamos considerar que **uPosição** é uma variável inteira que aponta para a última posição ocupada no vetor. Inicialmente, o valor da variável é -1, indicando que nenhuma posição foi utilizada. Quando o valor da variável for igual ao tamanho do vetor -1, o vetor estará cheio. Aqui e nos demais exemplos.

```
1 public static void insereNoInicio(int[] vet, int eleme
2         if (uPosicao==vet.length-1)
3          throw new Exception ("Nao ha posicoes livres"
4         for (int i=uPosicao+1;i>0;--i)
5          vet[i]=vet[i-1];
6         vet[0] = elemento;
7         uPosicao++;
8 }
```

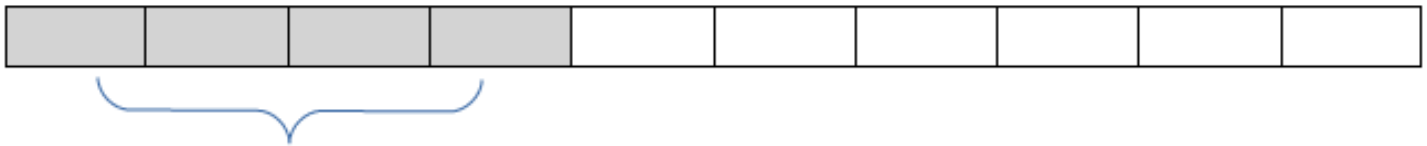
Inserção no final

- Há espaço disponível?



Inserção no final

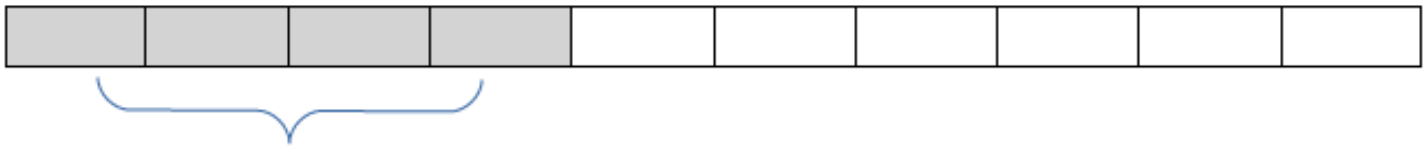
- Há espaço disponível?



- Insere-se o elemento na primeira posição livre;

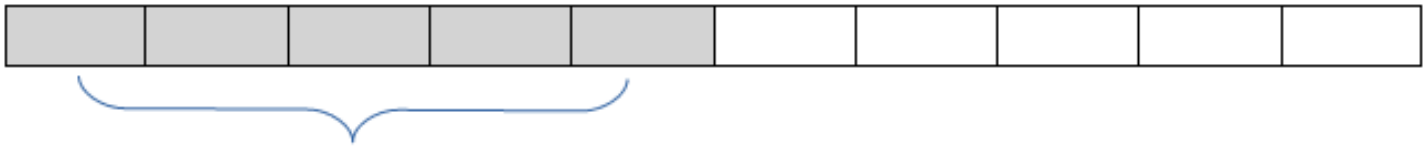
Inserção no final

- Há espaço disponível?



- Insere-se o elemento na primeira posição livre;

- Atualiza-se o número de elementos ocupados.

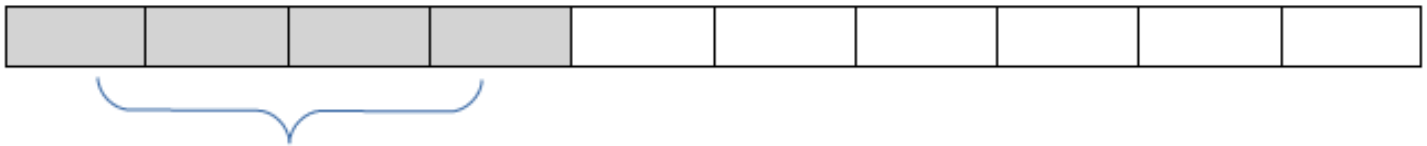


Inserção no final

```
1 public static void insereNoFinal(int[] vet, int
     elemento) throws Exception{
2  if (uPosicao==vet.length-1)
3     throw new Exception ("Nao ha posicoes livres"
        );
4  vet[++uPosicao] = elemento;
5 }
```

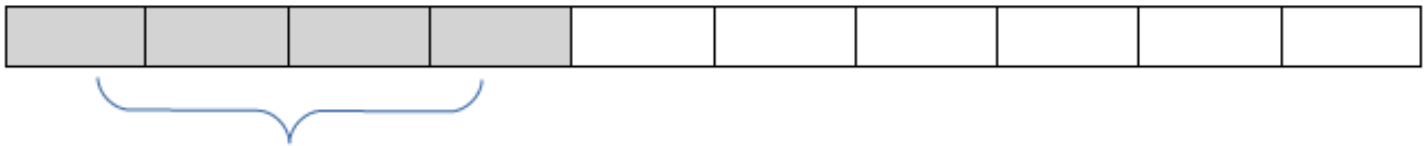
Inserção entre elementos

- Há espaço disponível?

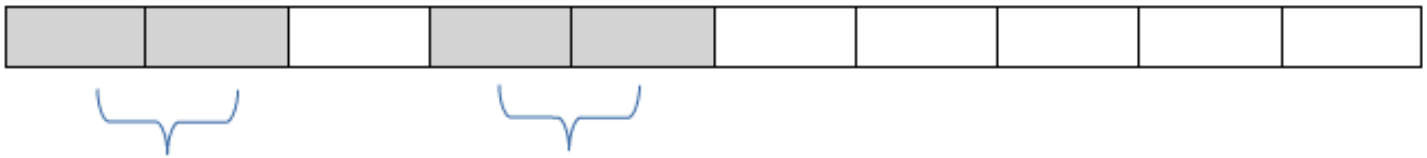


Inserção entre elementos

- Há espaço disponível?

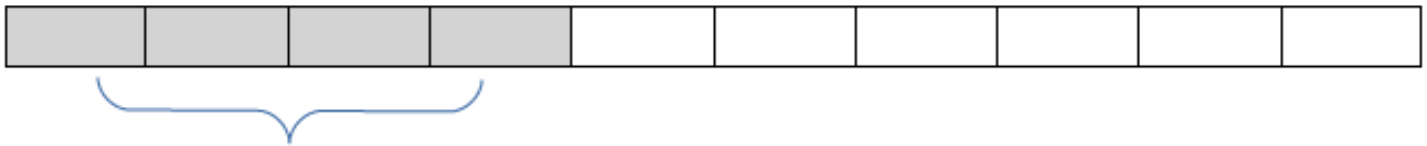


- Abre-se o espaço na posição desejada;

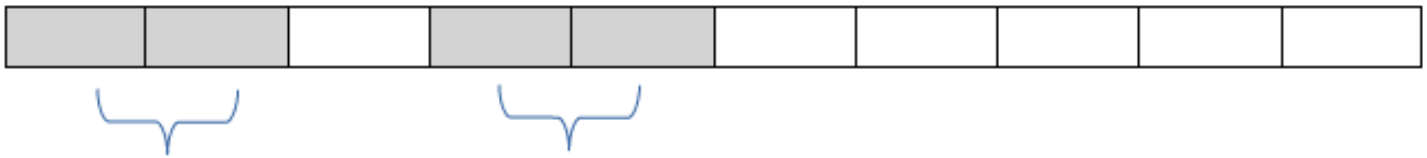


Inserção entre elementos

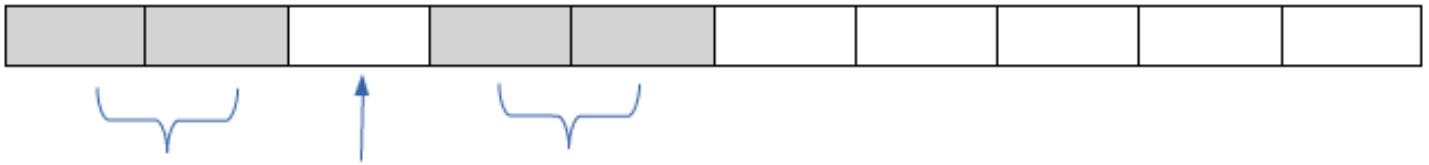
- Há espaço disponível?



- Abre-se o espaço na posição desejada;

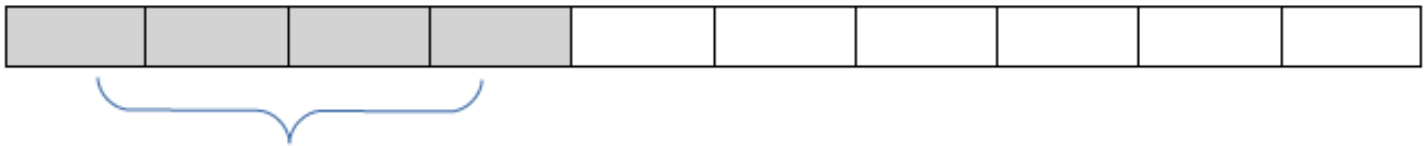


- Insere-se o novo elemento na posição liberada;

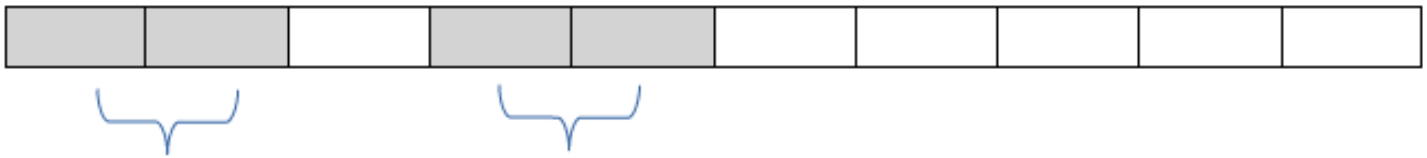


Inserção entre elementos

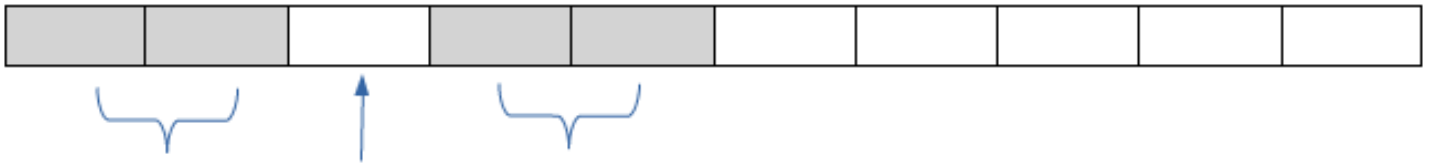
- Há espaço disponível?



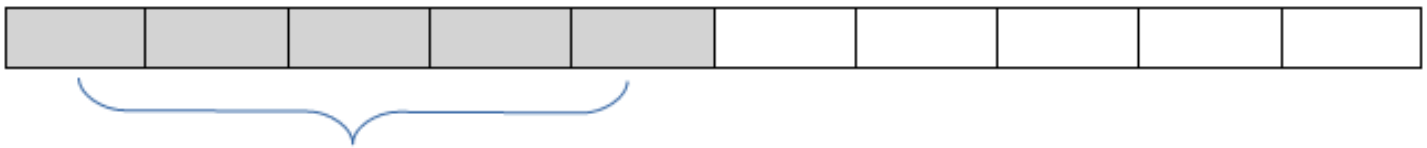
- Abre-se o espaço na posição desejada;



- Insere-se o novo elemento na posição liberada;



- Atualiza-se o número de elementos ocupados.



Inserção entre elementos

```
1  public static void insereEntreElementos(int[]
       vet, int elemento, int p) throws Exception{
2   if (uPosicao==vet.length-1)
3    throw new Exception ("Sem espaco.");
4   if (p>uPosicao)
5    insereNoFinal(vet,elemento);
6   else {
7    for (int i=uPosicao+1;i>p;--i)
8     vet[i]=vet[i-1];
9    vet[p] = elemento;
10   uPosicao++;
11  }
12 }
```

Analisando um pouco...

- Qual o custo computacional da nossa inserção no inicio?

Analisando um pouco...

- Qual o custo computacional da nossa inserção no inicio? $\Theta(n)$;

Analisando um pouco...

- Qual o custo computacional da nossa inserção no inicio? $\Theta(n)$;
- Qual o custo computacional da nossa inserção no final?

Analisando um pouco...

- Qual o custo computacional da nossa inserção no inicio? $\Theta(n)$;
- Qual o custo computacional da nossa inserção no final? $\Theta(1)$;

Analisando um pouco...

- Qual o custo computacional da nossa inserção no inicio? $\Theta(n)$;
- Qual o custo computacional da nossa inserção no final? $\Theta(1)$;
- Qual o custo computacional da nossa inserção entre elementos?

Analisando um pouco...

- Qual o custo computacional da nossa inserção no inicio? $\Theta(n)$;
- Qual o custo computacional da nossa inserção no final? $\Theta(1)$;
- Qual o custo computacional da nossa inserção entre elementos? $\Theta(n)$;

Analisando um pouco...

- Qual o custo computacional da nossa inserção no inicio? $\Theta(n)$;
- Qual o custo computacional da nossa inserção no final? $\Theta(1)$;
- Qual o custo computacional da nossa inserção entre elementos? $\Theta(n)$;

Você consegue pensar em formas de melhorar nossos métodos?

Exclusão de dados

De forma análoga à inclusão de dados:

- Os dados podem ser excluídos em três locais do vetor:

Exclusão de dados

De forma análoga à inclusão de dados:

- Os dados podem ser excluídos em três locais do vetor:
- No ínicio;

Exclusão de dados

De forma análoga à inclusão de dados:

- Os dados podem ser excluídos em três locais do vetor:
- No ínicio;
- No final;

Exclusão de dados

De forma análoga à inclusão de dados:

- Os dados podem ser excluídos em três locais do vetor:
- No ínicio;
- No final;
- Entre dois elementos quaisquer;

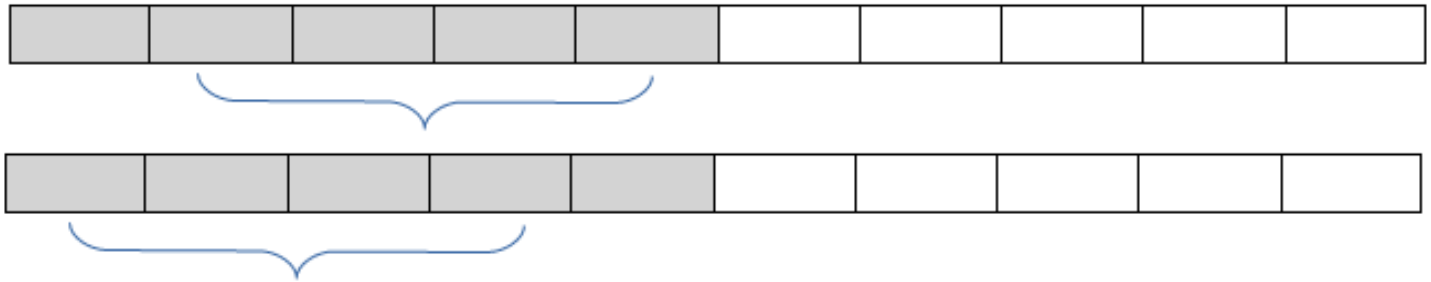
Exclusão de dados

De forma análoga à inclusão de dados:

- Os dados podem ser excluídos em três locais do vetor:
- No ínicio;
- No final;
- Entre dois elementos quaisquer;
- Cada uma dessas operações demanda um reajuste na estrutura de dados.

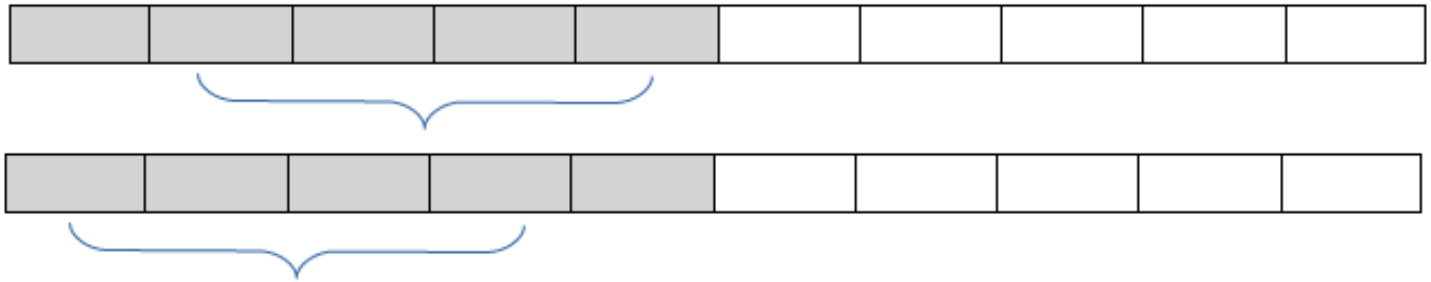
Exclusão no início

- A partir da posição $i = 1$, move-se os elementos para a posição $i - 1$ (onde i é a posição atual do elemento);

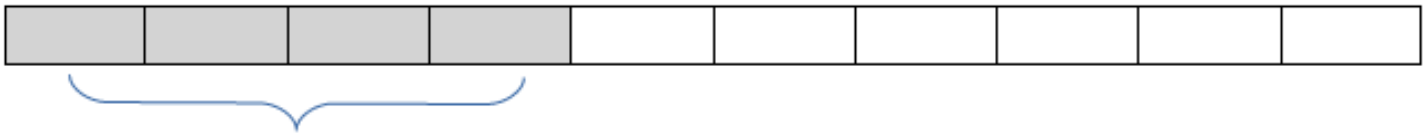


Exclusão no início

- A partir da posição $i = 1$, move-se os elementos para a posição $i - 1$ (onde i é a posição atual do elemento);



- Decrementa-se o total de posições utilizadas.

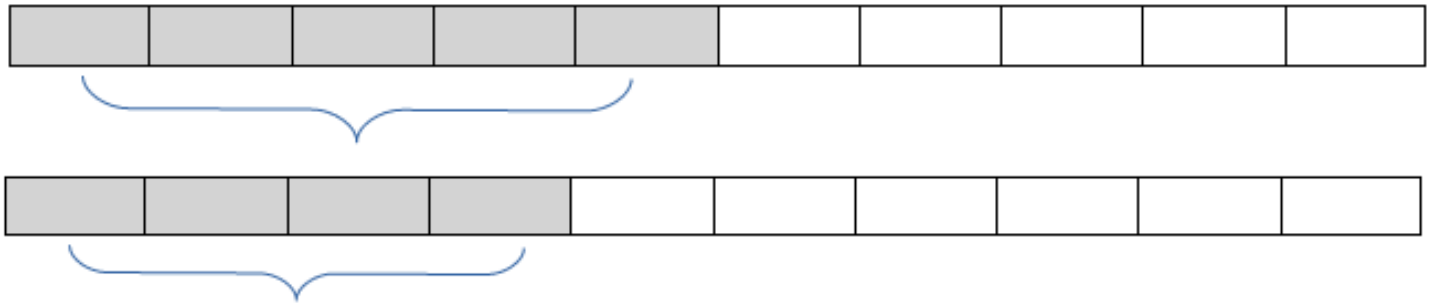


Exclusão no início

```
1  public static void excluirNoInicio(int[] vet)
       throws Exception{
2   if (uPosicao == -1)
3    throw new Exception ("O vetor esta vazio");
4   for (int i=1;i<=uPosicao;++i)
5    vet[i-1]=vet[i];
6   --uPosicao;
7  }
```

Exclusão no final

- Decrementa-se o total de posições utilizadas.

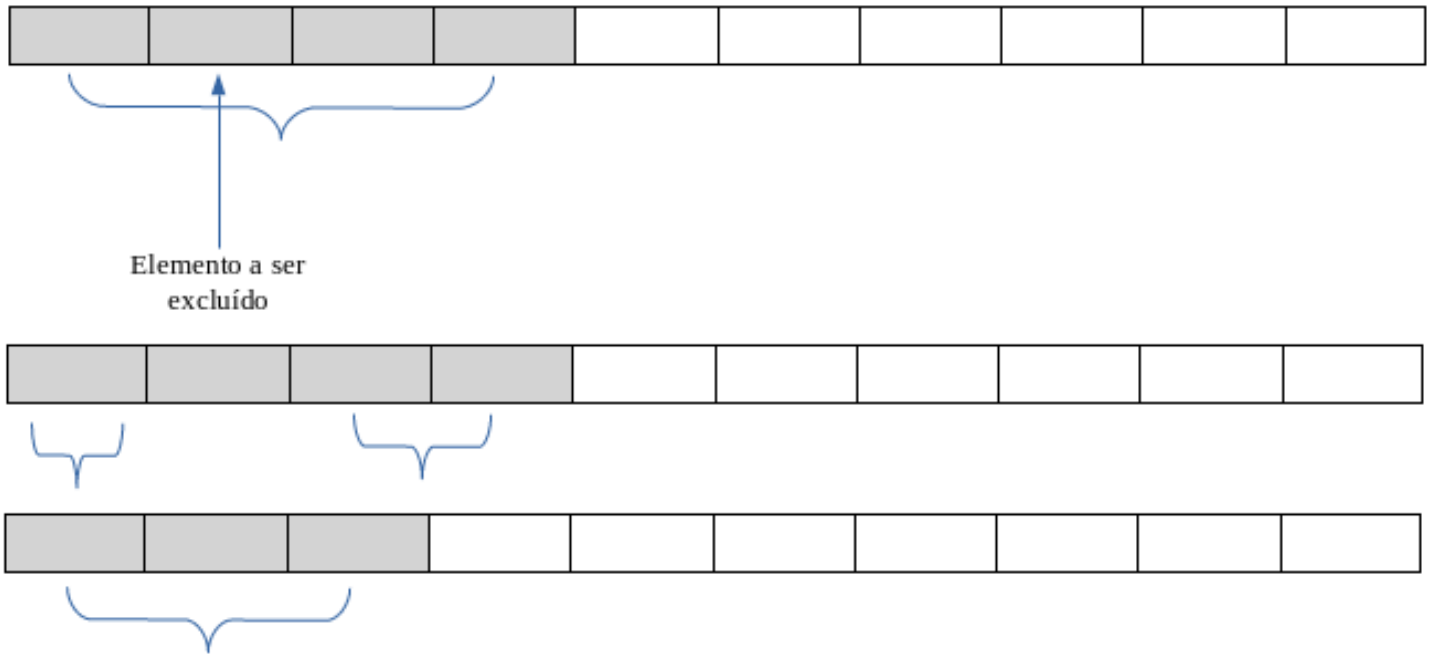


Exclusão no final

```
1  public static void excluirNoFinal(int[] vet)
       throws Exception{
2   if (uPosicao == -1)
3    throw new Exception ("O vetor esta vazio");
4   --uPosicao;
5  }
```

Exclusão entre elementos

- A partir da posição i que se deseja excluir, move-se todos os elementos adjacentes para a posição $i-1$ e atualiza-se o número de elementos ocupados



Exclusão entre elementos

```
1  public static void excluirElemento(int[] vet,
       int p) throws Exception{
2   if (uPosicao == -1)
3    throw new Exception ("O vetor esta vazio.");
4   if (p>uPosicao)
5    throw new Exception ("Esta posicao nao esta em
         uso.");
6   for (int i=p;i<uPosicao;++i)
7    vet[i]=vet[i+1];
8   --uPosicao;
9  }
```

Analisando um pouco...

- Qual o custo computacional da nossa exclusão no inicio?

Analisando um pouco...

- Qual o custo computacional da nossa exclusão no inicio? $\Theta(n)$;

Analisando um pouco...

- Qual o custo computacional da nossa exclusão no inicio? $\Theta(n)$;
- Qual o custo computacional da nossa exclusão no final?

Analisando um pouco...

- Qual o custo computacional da nossa exclusão no inicio? $\Theta(n)$;
- Qual o custo computacional da nossa exclusão no final? $\Theta(1)$;

Analisando um pouco...

- Qual o custo computacional da nossa exclusão no inicio? $\Theta(n)$;
- Qual o custo computacional da nossa exclusão no final? $\Theta(1)$;
- Qual o custo computacional da nossa inserção entre elementos?

Analisando um pouco...

- Qual o custo computacional da nossa exclusão no inicio? $\Theta(n)$;
- Qual o custo computacional da nossa exclusão no final? $\Theta(1)$;
- Qual o custo computacional da nossa inserção entre elementos? $\Theta(n)$;

Analisando um pouco...

- Qual o custo computacional da nossa exclusão no inicio? $\Theta(n)$;
- Qual o custo computacional da nossa exclusão no final? $\Theta(1)$;
- Qual o custo computacional da nossa inserção entre elementos? $\Theta(n)$;

Você consegue pensar em formas de melhorar nossos métodos?

Analisando um pouco...

- Qual o custo computacional da nossa exclusão no inicio? $\Theta(n)$;
- Qual o custo computacional da nossa exclusão no final? $\Theta(1)$;
- Qual o custo computacional da nossa inserção entre elementos? $\Theta(n)$;

Você consegue pensar em formas de melhorar nossos métodos? Tente não pensar de forma linear, tente pensar de forma circular...

Exercícios

1. Faça um programa que preencha um vetor de 10 posições com números aleatórios. Exiba o vetor preenchido. Remova do vetor todos os números pares e exiba seu conteúdo novamente.

2. Faça um programa que preencha um vetor de 100 posições com números aleatórios de acordo com a seguinte regra: Para cada número a ser inserido, sorteie um número aleatório (opção) entre um e três. Se opção for um, o número deverá ser inserido no início do vetor. Se opção for dois, o número deverá ser inserido na primeira posição livre (última posição). Se opção for três, o número deverá ser inserido imediatamente antes do valor inserido na iteração anterior. Ao final da alocação, exiba o vetor preenchido.

Exercícios

3. Desenvolva uma aplicação (modo texto ou modo gráfico) que permita ao usuário cadastrar até 10 nomes em uma lista de convidados (um vetor de String). As inclusões devem ser feitas sempre no final do vetor. Crie um pequeno menu para listar e controlar as ações do usuário. Seu programa deverá permitir, além da inclusão, exibir a lista de convidados, excluir um convidado específico e visualizar quantas vagas restam.

4. Desenvolva uma aplicação que permita ao usuário cadastrar até 10 números inteiros. Desenvolva opções para que sejam feitas inclusões e exclusões em todas as posições possíveis. Para inserções e exclusões *no meio*, apresente ao usuário o intervalo de escolha possível.

Exercícios

5. Escreva um método que imprima, em ordem crescente, todos os elementos de um vetor de inteiros. O método não pode ordenar o vetor e, obviamente, o vetor não está ordenado.

6. Escreva um método que receba como parâmetro dois vetores de inteiros (de tamanho **n**) e retorne um vetor de inteiro (de mesmo tamanho) preenchido de acordo com a seguinte regra: Para cada posição do terceiro vetor, sorteie um número aleatório. Caso este número seja par, receba um elemento do primeiro vetor, caso ímpar, do segundo vetor. Os valores dos dois primeiros vetores devem ser acessados na ordem em que foram enviados.


thank you
danke
謝謝
ngiyabonga
teşekkür ederim
gracias
merci
obrigado
go raibh maith agat
dziękuję
bedankt
mauruuru
hvala
sukriya
kop khun krap
terima kasih
grazie
arigatô
dakujem
takk
tapadh leat
mochchakkeram
спасибо
рахмат
Баярлалаа
faafetai lava
kiitos
dankie
dhanyavad
nanni
nandri
bayarlalaa
gracie
enkosi
dêkuji
sobodi
mesi
didi madloba
sagolun
najis tuke
kam sah hamnida
rahmat
তোমাকে ধন্যবাদ
감사합니다
xiexie
ευχαριστώ
ありがとう
tanemirt
rahmet
diolch
dhanyavadagalu
shukriya
merce
мерси
trugarez
chokrane
murakoze
tenki
obrigada
asante
manana
хвала
mamnun
дякую
djiere dieuf
tau
paldies
grazzi
mahalo
matondo
misaotra
dank je
barka
welalin
tack
mersi
kia ora
spas
vinaka
спасибі
blagodaram
akun
dankon
aciu
koszonom
chnorakaloutioun
gratias ago
gracies
sulpay
taiku
takk