Listas Lineares

Estruturas de Dados e Algoritmos

Prof. Vinicius Ramos

Adaptado: *prof.* ***Cristian Cechinel***

Referências utilizadas

1. http://www.inf.ufes.br/~pdcosta/ensino/2019-2-estruturas-de-dados/slides/Aula9%28listas%29.pdf

2. Sedgewick, Robert, and Kevin Wayne. Algorithms. Addison-Wesley Professional, 2011.

Listas Lineares

1. Uma lista é uma sequência finita de elementos relacionados
2. Diferentes tipos de dados podem ser representados por meio de listas: lista de alunos de uma turma, itens de estoque, lista de livros.

Listas Lineares

1. Operações possíveis: Inserir, retirar e localizar itens
2. *Possibilidade de crescer ou diminuir de tamanho durante a execução de um programa
3. Concatenação de duas listas para formar uma nova ou *particionamento de uma lista em duas ou mais listas
4. *Devem permitir a manipulação de quantidades indefinidas de dados

Listas Lineares

1. Operações possíveis: Inserir, retirar e localizar itens
2. Concatenação de duas listas para formar uma nova ou *particionamento de uma lista em duas ou mais listas
3. *Possibilidade de crescer ou diminuir de tamanho durante a execução de um programa
4. *Devem permitir a manipulação de quantidades indefinidas de dados

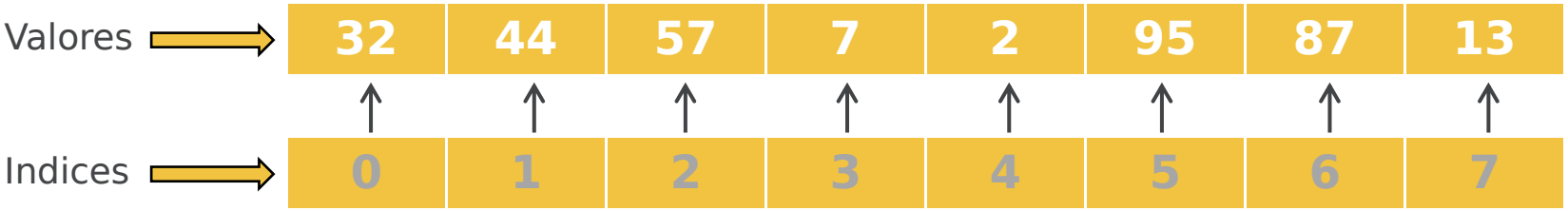
* Comumente, os Arrays/Vetores não possuem esses recursos nativos nas linguagens de programação

Listas Lineares

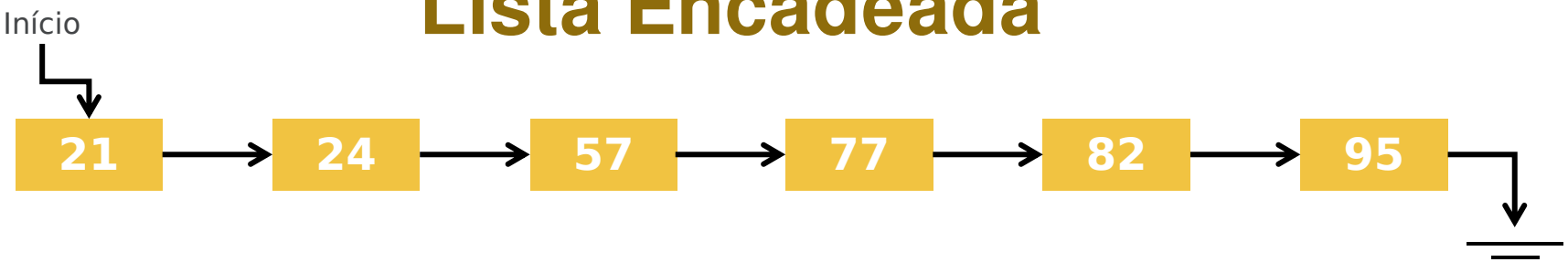
Podem ser representadas (construídas), basicamente, de duas formas distintas:

1. Utilizando alocação sequencial (com vetores)
2. Utilizando alocação não sequencial e dinâmica (com estruturas encadeadas)

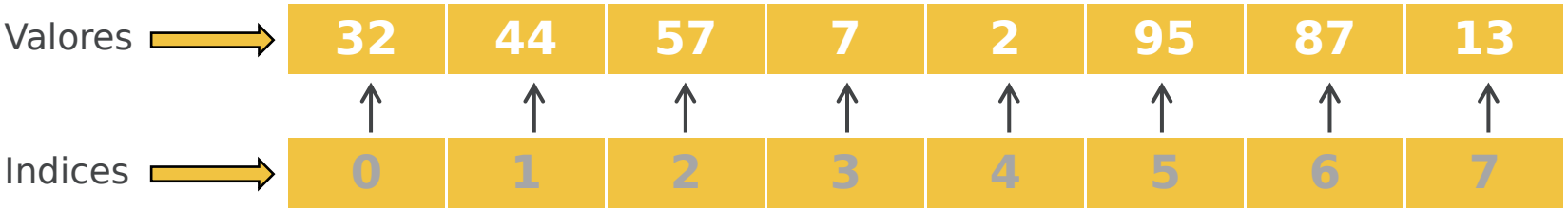
Vetores



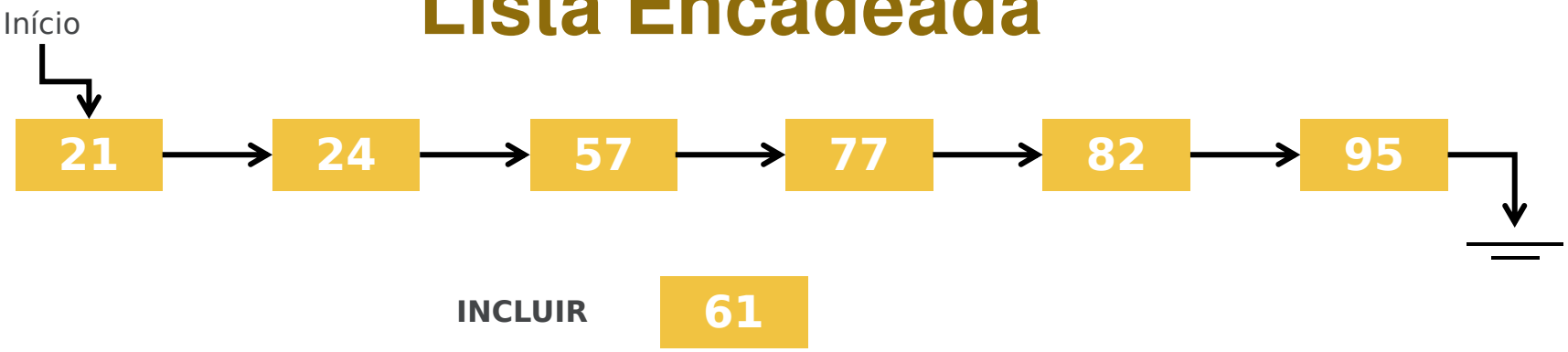
Lista Encadeada



Vetores

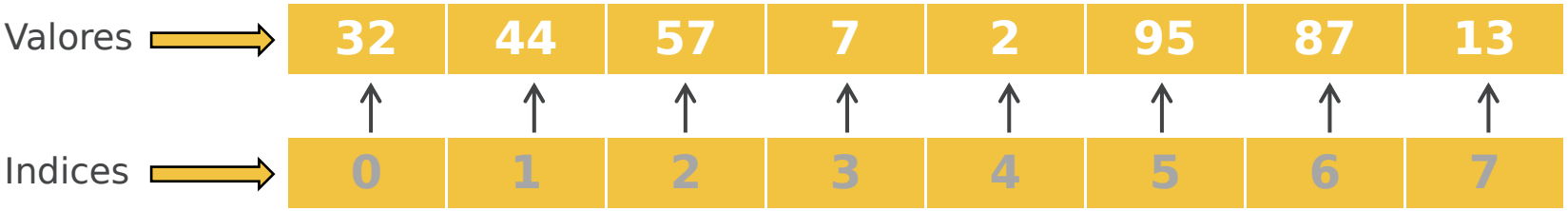


Lista Encadeada

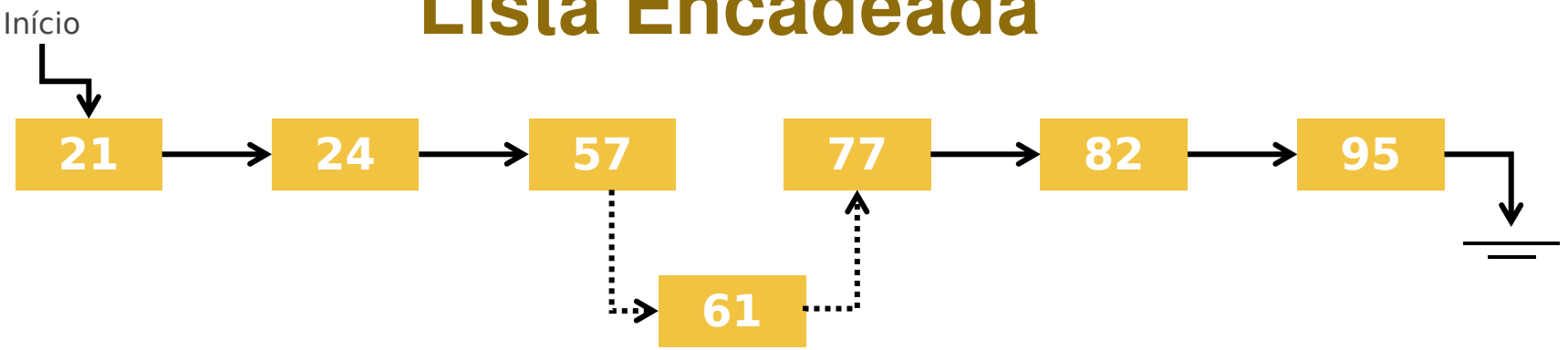


INCLUIR 61

Vetores



Lista Encadeada



Listas Lineares

Estruturas de Dados e Algoritmos

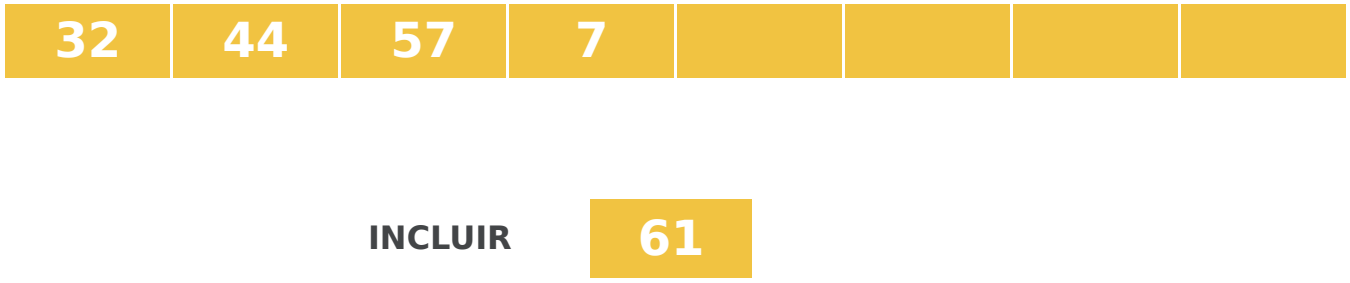
Prof. Vinicius Ramos

Adaptado: *prof.* ***Cristian Cechinel***

parte 2/3

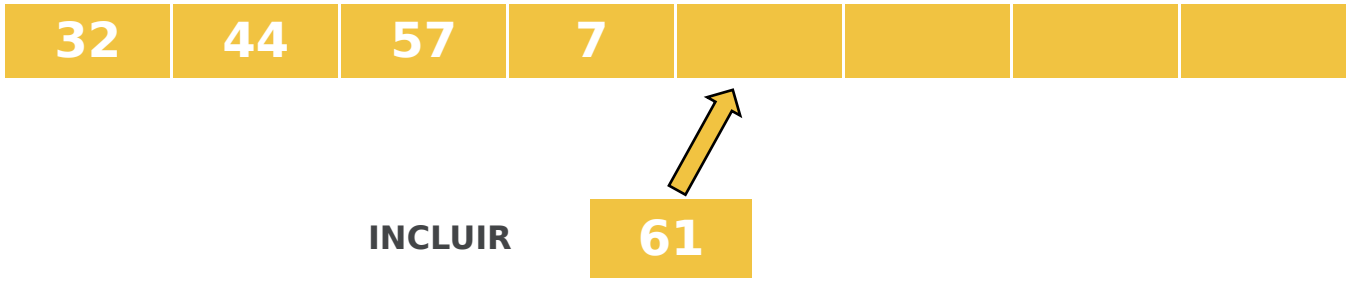
Listas Sequenciais (com vetores)

1. O armazenamento dos itens é realizado em posições contíguas da memória
2. A inserção de um novo item é realizada ao final sem precisar deslocar elementos



Listas Sequenciais (com vetores)

1. O armazenamento dos itens é realizado em posições contíguas da memória
2. A inserção de um novo item é realizada ao final sem precisar deslocar elementos

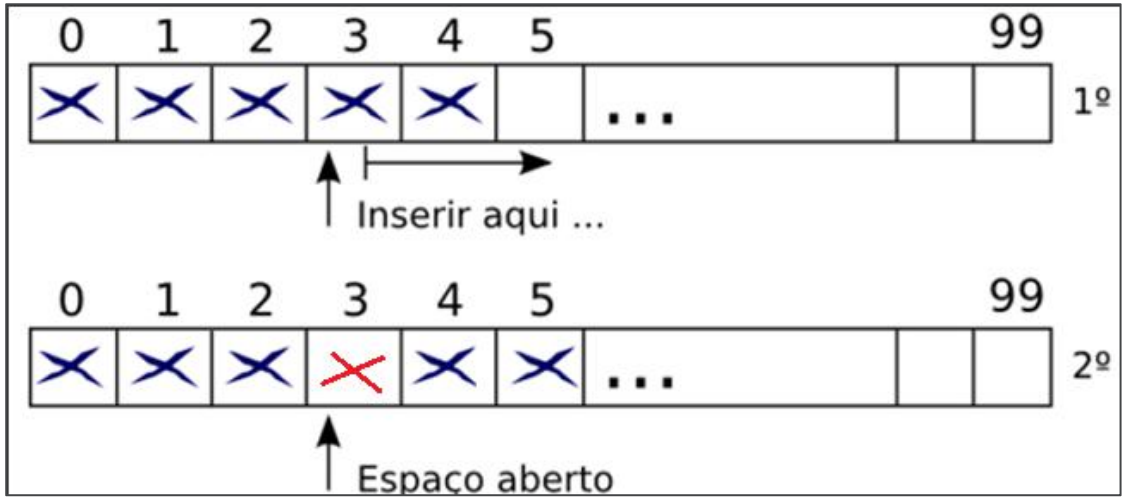


Listas Sequenciais (com vetores)

1. O armazenamento dos itens é realizado em posições contíguas da memória
2. A inserção de um novo item é realizada ao final sem precisar deslocar elementos

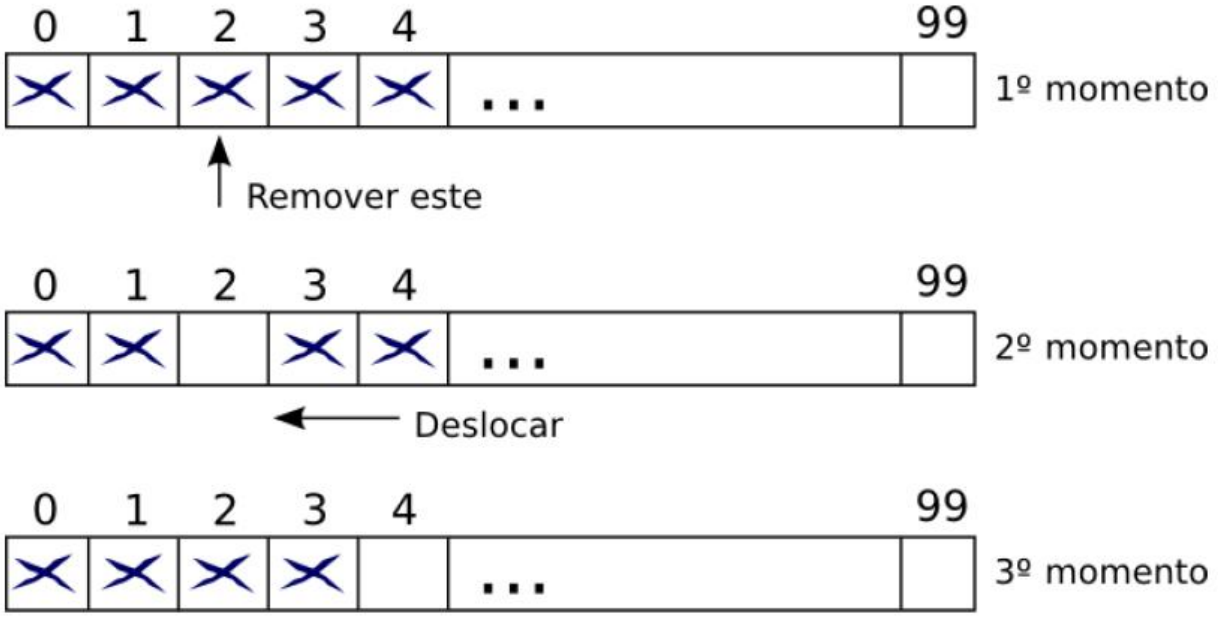
Listas Sequenciais (com vetores)

1. A inserção de um novo item no meio (ou no início) necessita de deslocamento de todos os itens localizados após o ponto de inserção



Listas Sequenciais (com vetores)

1. A retirada de um item do meio (ou do início) envolve o deslocalmento dos itens localizados após o ponto removido para evitar espaços vazios na lista



Listas Sequenciais (com vetores)

Algumas das operações que podem ser implementadas em uma lista:

1. Adicionar no inicio (exige deslocamento)
2. Adicionar no final
3. Adicionar no meio (exemplo: mantendo uma lista ordenada) (exige deslocamento)
4. Remover do início
5. Remover do final
6. Remover do meio

Listas Sequenciais (com vetores)

Algumas das operações que podem ser implementadas em uma lista:

1. Verificar se um elemento está contido na lista
2. Retornar a quantidade de elementos da lista
3. Imprimir a lista

Listas Sequenciais (com vetores)

Mais operações possíveis

1. Retornar uma cópia da lista
2. Dividir uma lista em duas partes
3. Concatenar duas listas

Listas Sequenciais (com vetores)

Atributos

1. Um vetor de dados do tipo que deve ser armazenado
2. A quantidade de dados armazenados (servirá como referência para localizar o último elemento da lista).

Listas Sequenciais (com vetores)

Classe ListaVetores que armazena uma lista de inteiros

```
13      public class ListaVetores {
            private int[] dados;
15          private int quantidade;
16
17          public ListaVetores(int tamanho){
18              dados = new int[tamanho];
19          }
```

Listas Sequenciais (com vetores)

Adicionar elemento ao final

1. Verificar se a lista está cheia
2. Caso não esteja cheia, inserir elemento ao final e aumentar a quantidade de elementos

```
21      public void add (int n) throws IndexOutOfBoundsException{
22          if (this.quantidade == dados.length)
23              throw new IndexOutOfBoundsException("Estouro da lista");
24
25          this.dados[this.quantidade] = n;
26          this.quantidade++;
27      }
```

Listas Sequenciais (com vetores)

Remover um elemento de uma determinada posição da lista

1. Verificar se existe algum elemento na lista
2. Retirar o elemento e diminuir a quantidade de elementos existentes
3. Deslocar para a esquerda os dados que estão localizados após o elemento removido

Listas Sequenciais (com vetores)

Remover um elemento de uma determinada posição da lista

```
28      public int remove(int i) throws IndexOutOfBoundsException {
29          if (i < 0 || i >= quantidade) throw new IndexOutOfBoundsException("Indice invalido");
30          int item = dados[i];
31          for (int j = i+1; j < quantidade; j++){
32              dados[j-1] = dados[j];
33          }
            dados[quantidade-1] = 0;
35          quantidade--;
36          return item;
37      }
```

Listas Sequenciais (com vetores)

1. O ideal ao retirar cada elemento é deixar o campo daquele elemento retirado com o valor **null**
2. Entretanto, como estamos trabalhando com um vetor do tipo primitivo **int,** isso não é possível pois o valor de inicialização desse tipo é zero.
3. Caso queiramos deixar os elementos do vetor inicializados com null, podemos utilizar uma classe Wrapper (Invólucro)
4. Uma classe Wrapper fornece uma maneira de usar os tipos primitivos como objetos. Para cada tipo primitivo a gente possui uma classe Wrapper.

Listas Sequenciais (com vetores)

```
13  public class ListaVetores {
    private int[] dados;
15  private int quantidade;
16
17  public ListaVetores(int tamanho){
18      dados = new int[tamanho];
19  }
```

```
13  public class ListaVetores {
    private Integer[] dados;
15  private int quantidade;
16
17  public ListaVetores(int tamanho){
18      dados = new Integer[tamanho];
19  }
```

Listas Sequenciais (com vetores)

```
28      public int remove(int i) throws IndexOutOfBoundsException {
29          if (i < 0 || i >= quantidade) throw new IndexOutOfBoundsException("Indice invalido");
30          int item = dados[i];
31          for (int j = i+1; j < quantidade; j++){
32              dados[j-1] = dados[j];
33          }
34          dados[quantidade-1] = null;
35          quantidade--;
36          return item;
37      }
```

Exercícios

Desenvolva os seguintes métodos:

1. boolean vazia(): verificar se a lista está vazia, retornando True para vazia e False para não-vazia.
2. void imprimeLista(): imprime os elementos da lista
3. int removeFinal(): remove o último elemento da lista.

Exercícios

Resolução

1. int removeFinal(): remove o último elemento da lista.

```
39      public int removeFinal() throws IndexOutOfBoundsException{
40          if (this.quantidade == 0) throw new IndexOutOfBoundsException("não há elementos");
41          int item = this.dados[quantidade -1];
42          this.quantidade--;
43          return item;
44      }
```

Listas Lineares

Estruturas de Dados e Algoritmos

Prof. Vinicius Ramos

Adaptado: *prof.* ***Cristian Cechinel***

parte 3/3

Listas Sequenciais (com vetores)

1. **A implementação realizada até o momento restringe o uso da lista a um determinado tamanho previamente fixado, deixando a lista estática.**
2. **É possível trabalhar com listas sequenciais de maneira que as mesmas sejam ampliadas antes que elas encham.**

Listas Sequenciais (com vetores)

1. **Para isso é necessário redefinir o tamanho do vetor da seguinte maneira:**

1. Cria-se um novo vetor temporário com o dobro do tamanho do vetor atual
2. Copia-se todos os elementos do vetor atual para o vetor temporário
3. O vetor temporário passa a ser o vetor atual

Listas Sequenciais (com vetores)

Método para realizar o redimensionamento de uma lista

```
20      private void resize(int max){
21          Integer[] temp = new Integer[max];
            for (int i = 0; i < this.quantidade; i++)
23              temp[i] = this.dados[i];
24          this.dados = temp;
25      }
```

Listas Sequenciais (com vetores)

No momento de inserir elementos na lista, ao invés de avisar que a lista está cheia, é possível então redimensionar o tamanho da mesma utilizando o método rezise(int max)

```
27      public void add (int n){// throws IndexOutOfBoundsException{
28          if (this.quantidade == dados.length)
29              resize(dados.length * 2);
30           //       throw new IndexOutOfBoundsException("Estouro da lista");
31
32          this.dados[this.quantidade] = n;
33          this.quantidade++;
34      }
```

Listas Sequenciais (com vetores)

No momento de inserir elementos na lista, ao invés de avisar que a lista está cheia, é possível então redimensionar o tamanho da mesma utilizando o método rezise(int max)

```
27      public void add (int n){// throws IndexOutOfBoundsException{
28          if (this.quantidade == dados.length)
29              resize(dados.length * 2);
30      //          throw new IndexOutOfBoundsException("Estouro da lista");
31
32          this.dados[this.quantidade] = n;
33          this.quantidade++;
34      }
```

No exemplo acima estamos dobrando o espaço da lista

(dados.length * 2)

Listas Sequenciais (com vetores)

Do mesmo modo que é possível aumentar o tamanho da lista (aumentar o vetor) quando a lista está enchendo, é possível diminuir o seu tamanho quando existem poucos elementos e muito espaço sem ser utilizado.

```
47      public int removeFinal() throws IndexOutOfBoundsException{
48          if (this.quantidade == 0) throw new IndexOutOfBoundsException("não há elementos");
49          int item = this.dados[quantidade -1];
50          this.quantidade--;
51          if (this.quantidade > 0 && this.quantidade == this.dados.length/4) {
52              resize(dados.length/2);
53          }
54          return item;
55      }
```

Listas Sequenciais (com vetores)

Do mesmo modo que é possível aumentar o tamanho da lista (aumentar o vetor) quando a lista está enchendo, é possível diminuir o seu tamanho quando existem poucos elementos e muito espaço sem ser utilizado.

```
47      public int removeFinal() throws IndexOutOfBoundsException{
48          if (this.quantidade == 0) throw new IndexOutOfBoundsException("não há elementos");
49          int item = this.dados[quantidade -1];
50          this.quantidade --;
51          if (this.quantidade > 0 && this.quantidade == this.dados.length/4) {
52              resize(dados.length/2);
53          }
54          return item;
55      }
```

No exemplo acima estamos dividindo

pela metade o espaço da lista

(dados.length / 2)

Referências utilizadas

1. http://www.inf.ufes.br/~pdcosta/ensino/2019-2-estruturas-de-dados/slides/Aula9%28listas%29.pdf

2. Sedgewick, Robert, and Kevin Wayne. Algorithms. Addison-Wesley Professional, 2011.