Lista Ordenada

Estrutura de Dados

Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação

Conteúdo

Lista Ordenada

- **Definição**
- Foi definida uma **lista** como uma estrutura linear, composta de um conjunto de n elementos $x_1, x_2, \ldots, x_n$ chamados nós, organizados de forma a manter uma relação entre eles.
  - Além das relações já apresentadas anteriormente, a posição dos itens em uma **lista ordenada** não é arbitraria.
  - Os itens devem respeitar uma *ordem total*:
    - Seja um conjunto $\mathbb{Z}$ de elementos
    - Para todo par de elementos (i,j) $\in \mathbb{Z}x\mathbb{Z}$, tal que $i \neq j$, vale exatamente uma das seguintes relações $i < j$ ou $j < i$

Lista Ordenada

- Operações comuns:
  - consultar o nó x_k;
  - inserir um novo nó x_k em uma lista ordenada;
  - remover o nó x_k;

- Obs: Lembrando que uma lista é uma estrutura de dados **dinâmica**, isto é, o seu comprimento (= número de nós) **pode ser alterado em tempo de execução**.

Representação de Lista Ordenada

- Uma lista ordenada pode ser representada por:
  - **contiguidade** dos nós;
  - **encadeamento** dos nós.

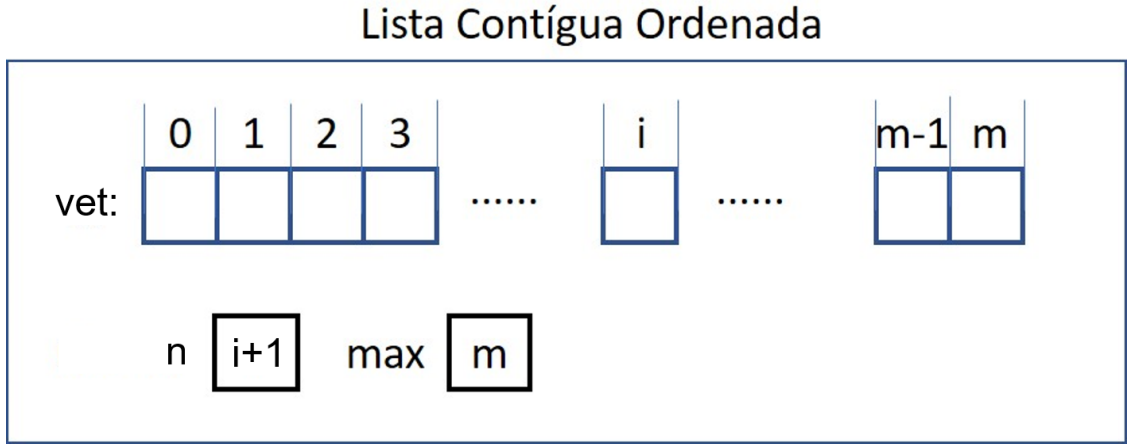
Lista Ordenada Contígua

Lista Ordenada Contígua

- Usando a alocação contígua (ou sequencial), os nós da lista são alocados fisicamente em posições consecutivas da memória.
- Assim, é comum usar um vetor como meio de alocação de posições consecutivas.
- Esquematicamente, uma lista contígua com n nós pode ser armazenada em um vetor `x` com n elementos.
- Mas como lista é uma **estrutura de dados dinâmica**, pode ser necessário a utilização de mais de n nós. Desta forma, o mais usual é trabalhar com um vetor de max elementos, sendo $n \leq max$.

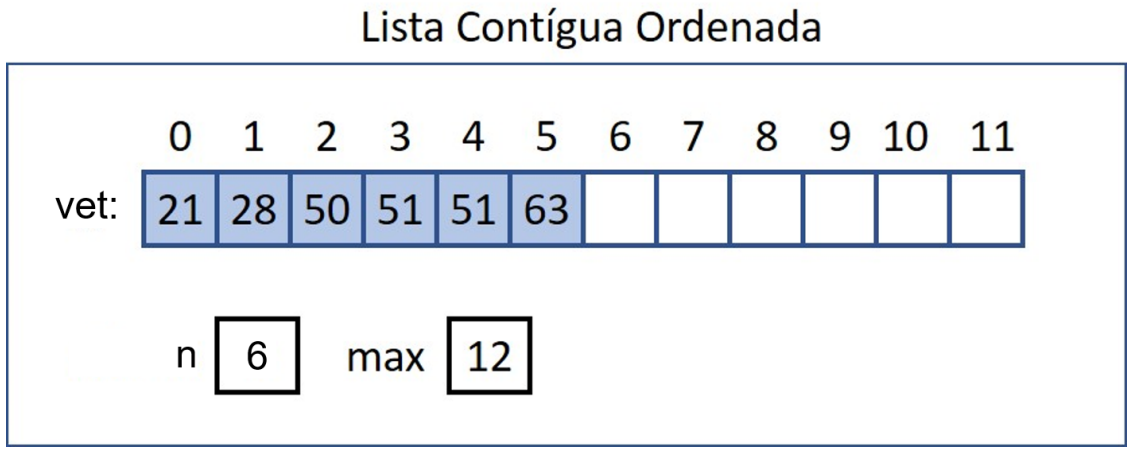
Lista Ordenada Contígua

- Esta lista é então composta por:
  - o número máximo de elementos (*max*);
  - um vetor `vet` com capacidade para *max* elementos;
  - quantidade de elementos na lista (*n*).
- Em uma lista com *n* elementos, tem-se `ultimo = n - 1`, onde `ultimo` é o índice do último elemento.



Lista Ordenada Contígua

- Exemplo de uma lista ordenada de valores inteiros usando a representação por contiguidade dos nós.
- Capacidade máxima: 12.
- Itens armazenados: 6.
- Vamos trabalhar aqui com listas de valores inteiros, mas pode-se criar listas de qualquer tipo de dados (lista de alunos, lista de pontos, etc), desde que se defina um campo chave para a ordenação.



Lista Ordenada Contígua

- TAD `ListaOrdCont` para listas ordenadas de valores inteiros.
  - Construtor
  - Destrutor
  - Consulta (`get`) o valor do elemento na posição `k`
  - `Insere` um valor `val` na ordem
  - `Remove` o nó com um valor `val`

Lista Ordenada Contígua

```
class ListaOrdCont
{
 public:
   ListaOrdCont(int tam);
   ~ListaOrdCont();

   int get(int k);
   void insere(int val);  // insere na ordem
   void remove(int val);  // remove no
   void imprime();
   bool busca(int val);
 private:
   int max;     // capacidade maxima de elementos
   int n;       // quantidade de nos na lista
   int *vet;    // vetor que armazena a lista
   int buscaBinaria(int val);
};
```

Lista Ordenada Contígua

- Construtor
  - Inicializa os dados.
  - Aloca memória de forma dinâmica para o vetor que representa a lista.
- Destrutor: desaloca a memória alocada de forma dinâmica.

```
ListaOrdCont::ListaOrdCont(int tam)
{
  max = tam;
  n = 0;
  vet = new int[max];
}
```

```
ListaOrdCont::~ListaOrdCont()
{
  delete [] vet;
}
```

Lista Ordenada Contígua

- Operação que retorna o conteúdo do nó na posição `k`

```
int ListaOrdCont::get(int k)
{
  if (k >= 0 && k < n)
     return vet[k];
  else
  {
     cout << "Indice invalido!" << endl;
     exit(1);
  }
}
```

Lista Ordenada Contígua

- Como a lista está ordenada podemos usar um algoritmo de **busca binária**.

Lista Ordenda Contígua

- Busca Binária:

```
int ListaOrdCont::buscaBinaria(int val)
{
  int esq = 0;
  int dir = n-1;
  while(esq <= dir)
  {
    int meio = (esq + dir) / 2;
    if(val > vet[meio])
      // se existir, val esta na segunda metade
      esq = meio + 1;
    else if (val < vet[meio])
      // se existir, val esta na primeira metade
      dir = meio - 1;
    else
      return meio; // val na posicao meio
  }
  return -1; // val nao encontrado
}
```

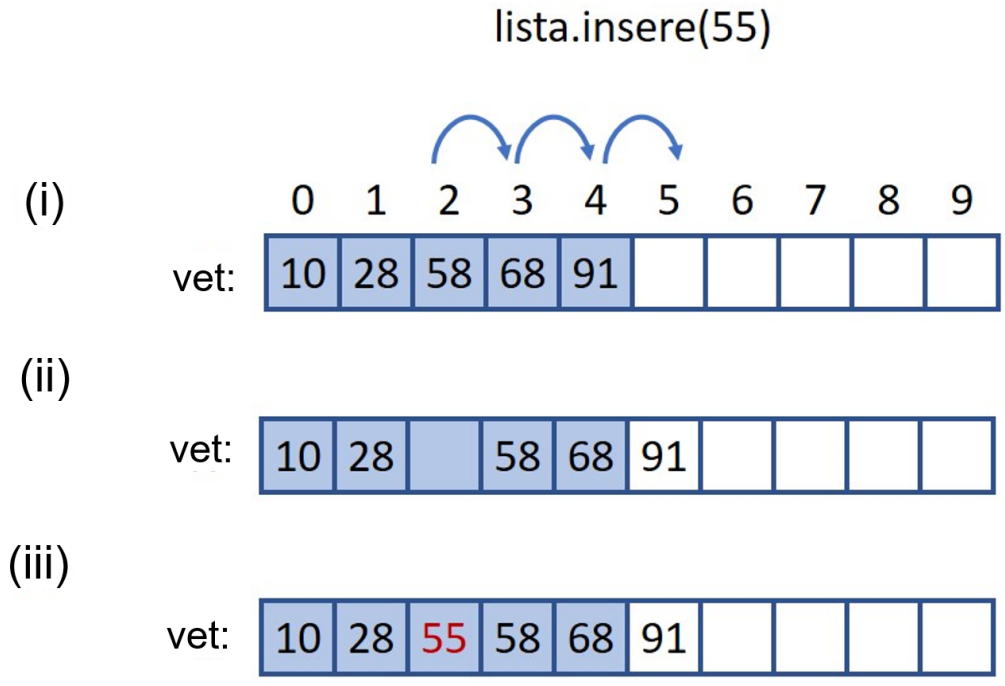
Lista Ordenda Contígua

- Busca por um valor e retorna
  - `true` se encontrá-lo
  - `false` se não encontrá-lo

```
bool ListaOrdCont::busca(int val)
{
  int k = buscaBinaria(val);
  // retorna true, se k eh um indice valido
  // returna false, caso contrario
  return k >= 0 && k < n;
}
```

Lista Ordenada Contígua

- Para inserir um novo nó na posição correta k é preciso "abrir espaço" e deslocar os itens à frente do nó vet_k para direita.
- A lista deve estar ordenada antes e deve permanecer ordenada após a inserção.

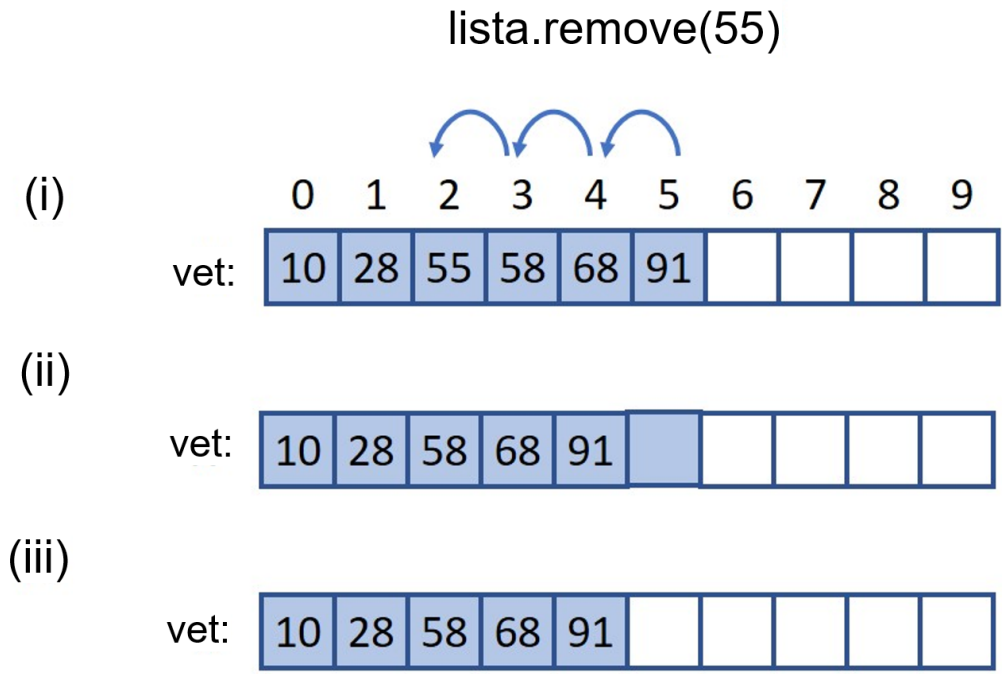


Lista Ordenada Contígua

```
void ListaOrdCont::insere(int val)
{
  int i;
  if(n == max)
  {
    cout << "Vetor Cheio!" << endl;
    exit(3);
  }
  for(i = n-1; i >= 0 && vet[i] >= val; i--)
  {
    vet[i+1] = vet[i];
  }
  vet[i+1] = val;
  n = n + 1;
}
```

Lista Ordenada Contígua

- Para remover um novo nó na posição k é preciso "fechar o seu espaço" e deslocar os itens à frente do nó vet_k para esquerda.



Lista Ordenada Contígua

```
void ListaOrdCont::remove(int val)
{
  // busca indice
  int k = buscaBinaria(val);

  if (k >= 0 && k < n)
  {
    // copia da dir. para esq.
    for(int i = k; i < n-1; i++)
      vet[i] = vet[i+1];
    n = n - 1;
  }
}
```

Lista Ordenada Contígua

Complexidade

- Considere as operações:
  - (i) inserir na lista ordenada; ou
  - (ii) encontrar itens na lista; ou
  - (iii) remover item da lista.
- Qual a complexidade das operações (i), (ii) e (iii)?

Listas Contíguas

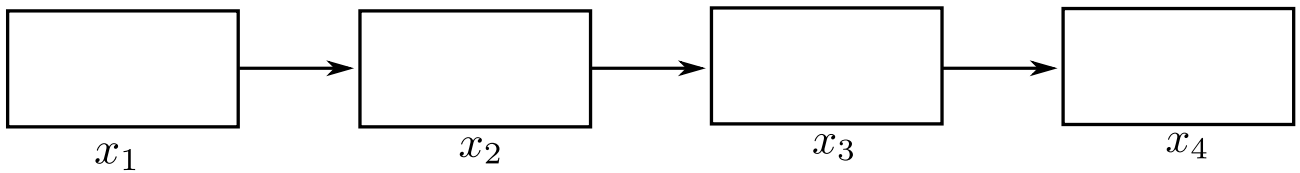
- **Exercício 1**: desenvolva um programa que cria uma lista ordenada e a inicializa com valores inteiros aleatórios (use o TAD `ListaOrdCont`) com capacidade máxima 100.
  - Escreva uma função que passe os dados de uma lista tradicional para uma lista ordenada. Qual a ordem de complexidade computacional?
  - Escreva uma operação de lista ordenada que intercale o conteúdo de duas listas ordenadas gerando uma terceira. Qual a ordem de complexidade computacional?

```
ListaOrdCont* ListaOrdCont::intercala(
        ListaOrdCont *l1, ListaOrdCont *l2);
```

Lista Ordenada Encadeada

Listas Encadeadas

- Lembrando que nessa estrutura de dados, um nó deve conter além de seu valor, um ponteiro para o nó seguinte, representando uma sequência dos nós da lista.
- Esquematicamente:



- Além disso, em uma lista simplesmente encadeada é necessário dispor de um ponteiro para o primeiro nó da lista.
- E o ponteiro do último nó de uma lista simplesmente encadeada sempre termina com o valor `NULL`.

Lista Simplesmente Encadeada

- Uma lista simplesmente encadeada consiste de dois objetos distintos:
  - o nó;
  - e a própria lista.
- Assim, é necessário definir dois TADs: `No` e `ListaOrdEncad`.
- A seguir são apresentados os TADs de uma **lista ordenada** simplesmente encadeada de valores inteiros.

Nó da Lista Simplesmente Encadeada

```
class No
{
 public:
   No();
   ~No();
   int getInfo();
   No* getProx();
   void setInfo(int val);
   void setProx(No *p);

 private:
   int info;   // informacao
   No *prox;   // ponteiro para o proximo
};
```

Nó da Lista Simplesmente Encadeada

```
No::No()  { }

No::~No() { }

int No::getInfo() {
  return info;
}

No* No::getProx() {
  return prox;
}

void No::setInfo(int val) {
  info = val;
}

void No::setProx(No *p) {
  prox = p;
}
```

Lista Ordenada Simplesmente Encadeada

```
class ListaOrdEncad
{
public:
   ListaOrdEncad();
   ~ListaOrdEncad();

   void insere(int val);
   bool busca(int val);
   void imprime();

 private:
   No *primeiro; // ponteiro para o primeiro
};
```

Lista Ordenada Simplesmente Encadeada

- Construtor

```
ListaOrdEncad::ListaOrdEncad()
{
  primeiro = NULL;
}
```

- Destrutor

```
ListaOrdEncad::~ListaOrdEncad()
{
  No *p = primeiro;
  while(p != NULL)
  {
    No *t = p->getProx();
    delete p;
    p = t;
  }
}
```

Lista Ordenada Simplesmente Encadeada

Insere na lista ordenada

```
void ListaOrdEncad::insere(int val)
{
  No *p = new No();
  No *ant = NULL, *atual = primeiro;
  p->setInfo(val);
  while(atual != NULL && val >= atual->getInfo()) {
    ant = atual;
    atual = atual->getProx();
  }
  if(ant == NULL) {
    p->setProx(primeiro);
    primeiro = p;
  } else {
    ant->setProx(p);
    p->setProx(atual);
  }
}
```

Lista Ordenada Simplesmente Encadeada

Programa

```
int main()
{
  ListaOrdEncad *lista = new ListaOrdEncad();
  int n;
  cin >> n;
  for(int i = 0; i < n; i++) {
    int valor;
    cin >> valor;
    lista->insere(valor);
  }
  cout << "Lista Ordenada: " << endl;
  lista->imprime();
  delete lista;
  return 0;
};
```

Lista Ordenada Simplesmente Encadeada

- **Exercício 2:** Desenvolva um programa que cria uma lista ordenada e a inicializa com valores inteiros aleatórios (use o TAD `ListaOrdEncad`).
- **Exercício 3:** Implemente uma função de busca para identificar se um elemento pertence à lista. Use o protótipo:

  ```
  bool ListaOrdEncad::busca(int val)
  ```

  A função deve retornar verdadeiro caso encontre o valor e falso caso contrário.

Comparação de Desempenho:

Contígua x Encadeada

Método	Contígua	Encadeada
inserir	$O(n)$	$O(n)$
encontrar	$O(\log n)$	$O(n)$
remover	$O(n)$	$O(n)$

Aplicação

Exemplo

- Um polinômio de ordem n:

$$\sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n,$$

- onde $a_n \neq 0$, pode ser representado por uma sequência de pares ordenados como:

$$(a_0, 0), (a_1, 1), (a_2, 2), \ldots, (a_n, n).$$

- e cada sequência pode ser representada por uma lista.

Aplicação

Exemplo

- Existem operações em polinômios cujo tempo de processamento depende da ordem dos termos.
- Por exemplo, vamos considerar adição de dois polinômios:

$$(a_0 + a_1x + a_2x^2) + (b_3x^3 + b_2x^2 + b_1x) = \\ (a_0) + (a_1 + b_1)x + (a_2 + b_2)x^2 + (b_3)x^3.$$

- para realizar a adição, todos os termos que envolvem x elevados a uma mesma potência devem ser agrupados.
- Podemos representar cada um dos polinômios como uma **lista ordenada**.

Exercícios

1. Implemente um programa para realizar a soma de dois polinômios representando-os através de **listas ordenadas encadeadas**
   a) Crie um novo TAD para representar um polinômio.
   Obs: Pense em como vai representar os termos.
   b) Implemente uma função que receba como parâmetros dois polinômios e retorna o polinômio resultante da soma dos termos em ordem.

Exercícios

2. Implemente uma operação que compute o valor de um polinômio para um dado valor de x. **Dica:** percorra todos os termos do polinômio e acumule o resultado.
3. Escreva um algoritmo para computar a k-ésima potência de um polinômio, onde k é um inteiro positivo. Qual o tempo de processamento do seu algoritmo?