

# Estruturas de Dados

## aula 01: Inspeccionando e manipulando dados

### 1 Introdução

- o que é um computador, e como ele funciona

### 2 Inspeção e manipulação de listas

Nós vamos começar o nosso estudo das estruturas de dados pelas listas.

Na aula de hoje, nós vamos considerar apenas listas de números armazenadas em um vetor  $V[1..N]$

E nós também vamos considerar apenas listas desordenadas — (*onde os números podem aparecer em uma ordem qualquer*)

Por exemplo,

|   |   |   |     |    |    |    |   |    |   |    |   |
|---|---|---|-----|----|----|----|---|----|---|----|---|
|   | 1 | 2 | ... |    |    |    |   |    |   |    | N |
| V | 8 | 2 | 5   | 11 | 29 | 13 | 6 | 16 | 7 | 10 |   |

A nossa primeira tarefa é a seguinte

→ *Verificar se o número  $k$  está na lista ou não*

*Como é que a gente faz isso?*

Bom, todo mundo sabe.

Quer dizer, a gente coloca o dedo no primeiro elemento.

Daí, a gente percorre a lista da esquerda para a direita.

E para cada elemento, a gente pergunta: *Você é o  $k$ ?*

É basicamente isso que o programa abaixo faz

```
0.      i = 1                                // colocando o dedo no 1o elemento
1.      Enquanto ( i ≤ N )                  // enquanto não chegar no fim
2.          Se ( V[i] == k )                // Você é o k?
3.              Imprime ( 'Achei' )
4.              Pára                          // pára o programa
5.          i = i + 1                        // desloca o dedo
6.      Imprime ( 'Não está lá ...' )
```

A seguir, nós vamos ver mais exemplos.

## Exemplos

### a. Contagem de ocorrências

Agora imagine que a lista pode conter elementos repetidos.

Por exemplo,

|   |   |   |     |   |   |   |   |   |   |   |   |   |
|---|---|---|-----|---|---|---|---|---|---|---|---|---|
|   | 1 | 2 | ... |   |   |   |   |   |   |   |   | N |
| V | 2 | 9 | 5   | 7 | 9 | 3 | 3 | 5 | 9 | 8 | 4 | 2 |

A tarefa consiste em

→ *Contar quantas vezes o número  $k$  aparece na lista*

Bom, dessa vez, um dedo só não basta.

Quer dizer, a gente também vai precisar de um contador.

A coisa fica assim

```
0.      i = 1                                // o dedo
1.      cont = 0                            // começando a contar do zero
2.      Enquanto ( i ≤ N )                  // percorre a lista
3.          Se ( V[i] == k )                // Você é o k?
4.              cont = cont + 1              // incrementa o contador
5.          i = i + 1                        // desloca o dedo
6.      Imprime (cont, 'ocorrências')
```

◇

### b. O maior de todos

Imagine mais uma vez que nós temos uma lista desordenada de números.

Por exemplo,

|   |    |   |     |    |   |   |    |   |   |    |   |    |
|---|----|---|-----|----|---|---|----|---|---|----|---|----|
|   | 1  | 2 | ... |    |   |   |    |   |   |    |   | N  |
| V | 13 | 6 | 2   | 11 | 4 | 1 | 15 | 5 | 7 | 10 | 9 | 12 |

A tarefa consiste em

→ *Encontrar o maior elemento da lista*

Bom, agora a gente vai usar o dedo, e outra variável  $M$  para lembrar o maior — (*i.e.*, o maior elemento que a gente viu até o momento)

Daí, a gente percorre a lista.

E, para cada elemento, a gente pergunta se ele é maior do que  $M$ .

Se for, a gente atualiza o valor de  $M$

A coisa fica assim

A coisa fica assim

```
0.      M = V[1]                                // o maior até aqui
1.      i = 2                                    // começando na segunda posição
2.      Enquanto ( i ≤ N )                      // percorre a lista
3.          Se ( V[i] > M )                    // Você é maior que M?
4.              M = V[i]                      // atualiza o valor de M
5.              i = i + 1                      // desloca o dedo
6.      Imprime ( 'o maior é:', M )
```

◇

c. O maior no fim

Imagine que nós temos uma lista desordenada de números.

Por exemplo,

|   |   |   |     |    |   |   |    |   |   |    |   |   |
|---|---|---|-----|----|---|---|----|---|---|----|---|---|
|   | 1 | 2 | ... |    |   |   |    |   |   |    |   | N |
| V | 7 | 2 | 16  | 13 | 4 | 5 | 18 | 9 | 6 | 10 | 1 | 8 |

A tarefa consiste em

→ Colocar o maior elemento na última posição

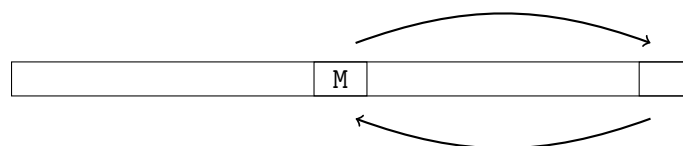
Bom, isso parece fácil.

Quer dizer, basta modificar a última linha do programa anterior

```
0.      M = V[1]
1.      i = 2
2.      Enquanto ( i ≤ N )
3.          Se ( V[i] > M )
4.              M = V[i]
5.              i = i + 1
6.      V[N] = M                                // coloca o maior na última posição
```

Mas, quando a gente faz isso, o número que estava na última posição é apagado — (*a gente perde informação ...*)

Então, a ideia é trocar os dois de lugar



Mas, para trocá-los de lugar, é preciso saber onde o maior está.

E por enquanto, só o que nós sabemos é o seu valor.

Então, agora, a gente precisa da variável `posM`, que lembra a posição do maior.

A coisa fica assim

```
0.      M = V[1];      posM = 1           // o maior até aqui
1.      i = 2           // começando na segunda posição
2.      Enquanto ( i ≤ N )           // percorre a lista
3.          Se ( V[i] > M )           // Você é maior que M?
4.              M = V[i];      posM = i   // atualiza M e posM
5.              i = i + 1           // desloca o dedo
6.
7.      aux = V[N]       //
8.      V[N] = M         // troca os dois de lugar
9.      V[posM] = aux    //
```

◇

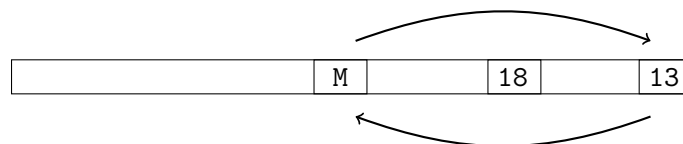
#### d. O maior no fim v.2.0

Agora nós vamos complicar um pouquinho as coisas.

Quer dizer, nós vamos adicionar uma restrição à tarefa anterior

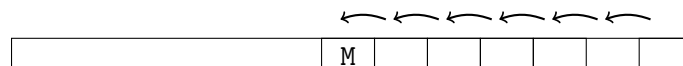
→ *Colocar o maior elemento na última posição,  
sem alterar a ordem dos demais elementos*

Bom, o ponto é que agora a gente não pode mais fazer a troca



Porque isso iria alterar a ordem do 18 e do 13, por exemplo.

Então, a solução é deslocar todos os elementos após o maior para a esquerda



E daí, a gente coloca o maior na última posição.

A coisa fica assim

```

0.      // 1a Etapa:  encontrar o maior
1.      M = V[1];    posM = 1
2.      i = 2                                // começando na segunda posição
3.      Enquanto ( i ≤ N )                    // percorre a lista
4.          Se ( V[i] > M )                    // Você é maior que M?
5.              M = V[i];    posM = i        // atualiza M e posM
6.              i = i + 1                    // desloca o dedo
7.
8.      // 2a Etapa:  deslocar elementos
9.      i = posM + 1                          // dedo no elem.  após o maior
10.     Enquanto ( i ≤ N )                    // vá até o fim
11.         V[i-1] = V[i]                    // move o elemento p/ a esquerda
12.         i = i + 1                        // desloca o dedo
13.
14.     // 3a Etapa:  coloca o maior no fim
15.     V[N] = M

```

◇

e. Todas as cópias do maior no fim

Imagine outra vez que a lista pode ter elementos repetidos.

Por exemplo,

|   |   |   |     |   |   |   |   |   |   |   |   |   |   |
|---|---|---|-----|---|---|---|---|---|---|---|---|---|---|
|   | 1 | 2 | ... |   |   |   |   |   |   |   |   |   | N |
| V | 2 | 9 | 5   | 7 | 9 | 3 | 3 | 5 | 9 | 8 | 4 | 2 |   |

A tarefa consiste em

→ *Colocar todas as cópias do maior elemento no fim da lista*

*Mas, como é que a gente faz isso?*

Bom, a maneira mais fácil é utilizando uma lista auxiliar.

E daí, a gente trabalha em 4 etapas

1. primeiro, a gente encontra o maior elemento M
2. depois, a gente copia os elementos menores do que M para a lista auxiliar
3. daí, a gente preenche as posições restantes com M
4. e daí, a gente copia tudo de volta para o vetor V

A coisa fica assim

```

0.      // 1a Etapa:  encontrar o maior
1.      M = V[1]
2.      i = 2                                // começando na segunda posição
3.      Enquanto ( i ≤ N )                    // percorre a lista
4.          Se ( V[i] > M )                    // Você é maior que M?
5.              M = V[i]                      // atualiza M
6.              i = i + 1                      // desloca o dedo
7.
8.      // 2a Etapa:  copiar menores que M para a lista A
9.      i = 1;    j = 1                        // um dedo em cada lista
10.     Enquanto ( i ≤ N )                     // vá até o fim
11.         Se ( V[i] < M )                     // desloca elemento p/ a esquerda
12.             A[j] = V[i]
13.             j = j + 1
14.             i = i + 1
15.
16.     // 3a Etapa:  completa a lista A com cópias de M
17.     Enquanto ( j ≤ N )                     // vá até o fim da lista A
18.         A[j] = M                           // coloca cópia de M
19.         j = j + 1
20.
21.     // 4a Etapa:  completa a lista A com cópias de M
22.     i = 1                                  // apenas um dedo
23.     Enquanto ( j ≤ N )                     // percorre a lista inteira
24.         V[i] = A[i]                        // copia elemento
25.         i = i + 1

```

Legal.

Mas, essa não é a única maneira de fazer as coisas.

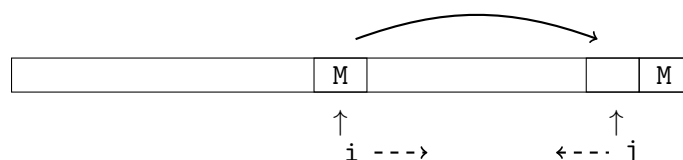
Quer dizer, a gente não precisa realmente utilizar uma lista auxiliar.

A ideia é que primeiro a gente descobre quem é o maior.

Daí, a gente percorre a lista outra vez.

E cada vez que a gente encontra uma cópia do maior, a gente joga ele para o fim.

Para saber o lugar onde a cópia é colocada, a gente utiliza um segundo dedo



A coisa fica assim

```

0.      // 1a Etapa:  encontrar o maior
1.      M = V[1]
2.      i = 2                                // começando na segunda posição
3.      Enquanto ( i ≤ N )                    // percorre a lista
4.          Se ( V[i] > M )                    // Você é maior que M?
5.              M = V[i]                      // atualiza M
6.              i = i + 1                      // desloca o dedo
7.
8.      // 2a Etapa:  coloca cópias de M no fim
9.      i = 1;    j = N                        // um dedo no início, outro no fim
10.     Enquanto ( i < j )                      // vá até o fim
11.         Se ( V[i] == M )                    // Você é uma cópia de M?
12.             aux = V[i]                      //
13.             V[i] = V[j]                    // troca
14.             V[j] = aux                      //
15.             j = j - 1                      // volta o dedo j uma posição
16.         Senão
17.             i = i + 1                      // avança o dedo i uma posição

```

◇

#### f. Separando pares e ímpares

Imagine que nós temos uma lista desordenada de números.

Por exemplo,

|   |    |   |     |   |    |   |   |   |    |   |    |   |
|---|----|---|-----|---|----|---|---|---|----|---|----|---|
|   | 1  | 2 | ... |   |    |   |   |   |    |   |    | N |
| V | 12 | 9 | 4   | 7 | 19 | 3 | 6 | 5 | 14 | 8 | 11 | 2 |

A tarefa consiste em

→ Colocar todos os números pares no início  
e colocar todos os números ímpares no fim da lista

No exemplo acima, isso nos daria

|   |    |   |     |    |   |   |   |   |    |   |   |    |
|---|----|---|-----|----|---|---|---|---|----|---|---|----|
|   | 1  | 2 | ... |    |   |   |   |   |    |   |   | N  |
| V | 12 | 4 | 6   | 14 | 8 | 2 | 9 | 7 | 19 | 3 | 5 | 11 |

Bom, é fácil resolver esse problema utilizando uma lista auxiliar.

Quer dizer,

1. primeiro, a gente copia os números pares para o início da lista auxiliar
2. depois, a gente copia os números ímpares para o fim da lista auxiliar
3. e daí, basta copiar tudo de volta para o vetor  $V$

Mas, não é realmente preciso utilizar uma lista auxiliar.

Quer dizer, esse problema é igual ao anterior.

Para ver isso, basta imaginar que os números ímpares são cópias do maior elemento  $M$ .

Então, basta percorrer a lista, jogando os números ímpares para o final.

A coisa fica assim

```
0.      i = 1;      j = N                                // um dedo no inicio, outro no fim
1.      Enquanto ( i < j )                                // vá até o fim
2.          Se ( V[i] é impar )                            // Você é um número impar?
3.              aux = V[i]                                  //
4.              V[i] = V[j]                                // troca
5.              V[j] = aux                                  //
6.              j = j - 1                                    // volta o dedo j uma posição
7.      Senão
8.          i = i + 1                                       // avança o dedo i uma posição
```

◇