



# **BUSCA LINEAR E BUSCA BINÁRIA**

ECM404

# DEFINIÇÃO DO PROBLEMA

- ❑ Dada uma chave de busca e uma coleção de elementos, onde cada elemento possui um identificador único, desejamos encontrar o elemento da coleção que possui o identificador igual ao da chave de busca ou verificar que não existe nenhum elemento na coleção com a chave fornecida.

# DEFINIÇÃO DO PROBLEMA

- ❑ Exemplo: Desejamos buscar o valor 1 no array abaixo.

|     |   |   |   |   |   |   |
|-----|---|---|---|---|---|---|
| Vet | 3 | 4 | 1 | 0 | 2 | 4 |
|     | 0 | 1 | 2 | 3 | 4 | 5 |

- ❑ Neste caso, o algoritmo retornará 2, pois é a posição na qual ele encontrou o valor que estava buscando.

# DEFINIÇÃO DO PROBLEMA

- ❑ Exemplo: Desejamos buscar o valor 5 no array abaixo.

Vet

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 3 | 4 | 1 | 0 | 2 | 4 |
| 0 | 1 | 2 | 3 | 4 | 5 |

- ❑ Neste caso, o algoritmo retornará -1, pois não encontrou o valor que estava buscando.

# ALGORITMOS DE BUSCA

- ❑ Existem diversos algoritmos de busca. Entretanto, em nossa disciplina estudaremos dois:
  - ❑ Busca linear ou busca sequencial;
  - ❑ Busca binária.
- ❑ Para mais informações: [geeksforgeeks.org](https://www.geeksforgeeks.org)

# BUSCA LINEAR

- ❑ A busca linear é o algoritmo mais simples de busca:
  - ❑ Percorra o array comparando a chave com os valores dos elementos em cada uma das posições;
  - ❑ Se a chave for igual a algum dos elementos, retorne a posição correspondente no array;
  - ❑ Se o array todo foi percorrido e a chave não foi encontrada, retorne o valor -1.

# ATIVIDADE - BUSCA LINEAR

- ❑ Implemente a busca linear como uma função que receba como parâmetros o array e a chave desejada.

# BUSCA LINEAR

- ❑ **Vantagem:** fácil implementação.
- ❑ **Desvantagem:** Suponha que você queira buscar informações de uma estrela em particular, cadastrada no catálogo Tycho-2. Este catálogo possui informações sobre as 2.539.913 estrelas mais brilhantes na nossa galáxia. Assim, no pior caso, o algoritmo irá comparar todas as 2.539.913 estrelas para encontrar a que você deseja obter informações.



# BUSCA BINÁRIA

- ❑ Caso o catálogo esteja organizado segundo um critério, utilizando o algoritmo **busca binária** não seriam necessárias mais do que 22 comparações para encontrar a estrela desejada, mesmo no pior caso.

# BUSCA BINÁRIA

- ❑ A busca binária é um algoritmo mais eficiente, entretanto, requer que a lista esteja ordenada pelos valores da chave de busca.
- ❑ A ideia do algoritmo é a seguinte:
  - ❑ Verifique se a chave de busca é igual ao valor da posição do meio do array;
  - ❑ Caso seja igual, devolva esta posição;
  - ❑ Caso o valor central seja maior que a chave, então repita o processo, mas considere um array reduzido, com os elementos do começo do array até a posição anterior a do meio;
  - ❑ Caso o valor central seja menor que a chave, então repita o processo, mas considere um array reduzido, com os elementos da posição seguinte a do meio até o fim do array;

# BUSCA BINÁRIA – EXEMPLO

- ❑ Considere que desejamos buscar a chave 15.

chave = 15

|   |   |    |    |    |    |    |    |    |     |
|---|---|----|----|----|----|----|----|----|-----|
| 1 | 5 | 15 | 20 | 24 | 45 | 67 | 76 | 78 | 100 |
| 0 | 1 | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9   |

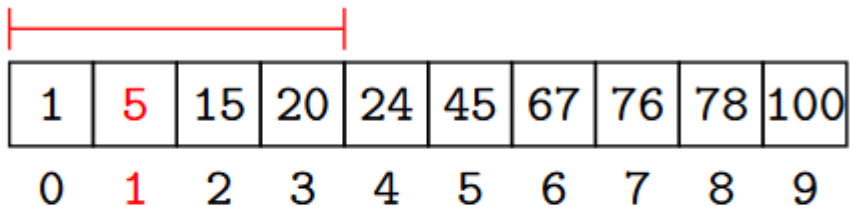
pos\_ini = 0  
pos\_fim = 9  
pos\_meio = 4

- ❑ Como  $[\text{pos\_meio}] > \text{chave}$ , devemos continuar a busca na primeira metade da região.

# BUSCA BINÁRIA – EXEMPLO

- ❑ Considere que desejamos buscar a chave 15.

chave = 15



|   |   |    |    |    |    |    |    |    |     |
|---|---|----|----|----|----|----|----|----|-----|
| 1 | 5 | 15 | 20 | 24 | 45 | 67 | 76 | 78 | 100 |
| 0 | 1 | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9   |


pos\_ini = 0  
pos\_fim = 3  
pos\_meio = 1

- ❑ Como  $[\text{pos\_meio}] < \text{chave}$ , devemos continuar a busca na segunda metade da região.

# BUSCA BINÁRIA – EXEMPLO

- ❑ Considere que desejamos buscar a chave 15.

chave = 15



|   |   |    |    |    |    |    |    |    |     |
|---|---|----|----|----|----|----|----|----|-----|
| 1 | 5 | 15 | 20 | 24 | 45 | 67 | 76 | 78 | 100 |
| 0 | 1 | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9   |

pos\_ini = 2  
pos\_fim = 3  
pos\_meio = 2

- ❑ Finalmente, encontramos a chave (  $[pos\_meio] = chave$  ). Assim, devolvemos sua posição no array (pos\_meio).

# BUSCA BINÁRIA – EXEMPLO

- ❑ Considere que desejamos buscar a chave 50.

chave = 50

|   |   |    |    |    |    |    |    |    |     |
|---|---|----|----|----|----|----|----|----|-----|
| 1 | 5 | 15 | 20 | 24 | 45 | 67 | 76 | 78 | 100 |
| 0 | 1 | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9   |


pos\_ini = 0  
pos\_fim = 9  
pos\_meio = 4

- ❑ Como  $[\text{pos\_meio}] < \text{chave}$ , devemos continuar a busca na segunda metade da região.

# BUSCA BINÁRIA – EXEMPLO

- ❑ Considere que desejamos buscar a chave 50.

chave = 50



|   |   |    |    |    |    |    |    |    |     |
|---|---|----|----|----|----|----|----|----|-----|
| 1 | 5 | 15 | 20 | 24 | 45 | 67 | 76 | 78 | 100 |
| 0 | 1 | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9   |

pos\_ini = 5

pos\_fim = 9

pos\_meio = 7

- ❑ Como  $[\text{pos\_meio}] > \text{chave}$ , devemos continuar a busca na primeira metade da região.

# BUSCA BINÁRIA – EXEMPLO

- ❑ Considere que desejamos buscar a chave 50.

chave = 50

|   |   |    |    |    |    |    |    |    |     |
|---|---|----|----|----|----|----|----|----|-----|
| 1 | 5 | 15 | 20 | 24 | 45 | 67 | 76 | 78 | 100 |
| 0 | 1 | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9   |

pos\_ini = 5  
pos\_fim = 6  
pos\_meio = 5


- ❑ Como  $[\text{pos\_meio}] < \text{chave}$ , devemos continuar a busca na segunda metade da região.



# BUSCA BINÁRIA – EXEMPLO

- ❑ Considere que desejamos buscar a chave 50.

chave = 50



|   |   |    |    |    |    |    |    |    |     |
|---|---|----|----|----|----|----|----|----|-----|
| 1 | 5 | 15 | 20 | 24 | 45 | 67 | 76 | 78 | 100 |
| 0 | 1 | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9   |

pos\_ini = 6

pos\_fim = 6

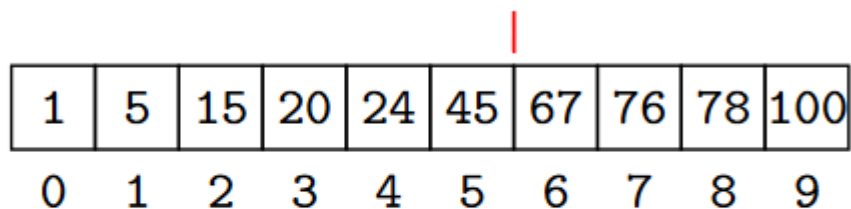
pos\_meio = 6

- ❑ Como  $[\text{pos\_meio}] > \text{chave}$ , devemos continuar a busca na primeira metade da região.

# BUSCA BINÁRIA – EXEMPLO

- ❑ Considere que desejamos buscar a chave 50.

chave = 50



|   |   |    |    |    |    |    |    |    |     |
|---|---|----|----|----|----|----|----|----|-----|
| 1 | 5 | 15 | 20 | 24 | 45 | 67 | 76 | 78 | 100 |
| 0 | 1 | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9   |

pos\_ini = 6

pos\_fim = 5

pos\_meio = 5

- ❑ Como  $\text{pos\_ini} > \text{pos\_fim}$ , determinamos que a chave não está na lista e assim retornamos o valor -1.

# ATIVIDADE – BUSCA BINÁRIA

- ❑ Implemente o algoritmo de busca binária como uma função que recebe como parâmetro uma matriz e a chave que deseja-se buscar.
- ❑ Caso a matriz esteja ordenada, utilize a busca binária;
- ❑ Caso a matriz esteja desordenada, utilize a busca sequencial;