

# Unidade I:

## Introdução - Algoritmos de Pesquisa




**PUC Minas**

Instituto de Ciências Exatas e Informática  
Departamento de Ciência da Computação

# Agenda

- Pesquisa sequencial
- Pesquisa binária

- **Pesquisa sequencial** 
- Pesquisa binária

# Pesquisa Sequencial

```
boolean resp = false;

for (int i = 0; i < n; i++){
    if (array[i] == x){
        resp = true;
        i = n;
    }
}
```

# Pesquisa Sequencial

```
boolean resp = false;  
  
for (int i = 0; i < n; i++){  
    if (array[i] == x){  
        resp = true;  
        i = n;  
    }  
}
```

1º) Qual é a operação relevante?

R: Comparação entre elementos do array

2º) Quantas vezes ela será executada?

# Pesquisa Sequencial

```
boolean resp = false;

for (int i = 0; i < n; i++){
    if (array[i] == x){
        resp = true;
        i = n;
    }
}
```

1º) Qual é a operação relevante?

R: Comparação entre elementos do array

2º) Quantas vezes ela será executada?

R: Em qual dos casos?

# Pesquisa Sequencial

```
boolean resp = false;

for (int i = 0; i < n; i++){
    if (array[i] == x){
        resp = true;
        i = n;
    }
}
```

1º) Qual é a operação relevante?

R: Comparação entre elementos do array

2º) Quantas vezes ela será executada?

R: Melhor caso:  $f(n) = 1$

Pior caso:  $f(n) = n$

Caso médio:  $f(n) = (n + 1) / 2$

# Exercício Resolvido (1)

- Quando acontece o melhor e o pior caso do algoritmo de pesquisa sequencial?



## Exercício Resolvido (1)

- Quando acontece o melhor e o pior caso do algoritmo de pesquisa sequencial?



### Resposta:

O melhor caso acontece quando o elemento procurado está na primeira posição

O pior acontece quando o elemento desejado está na última posição ou quando ele não está no *array*

## Exercício Resolvido (2)

- Supondo que temos uma informação extra sobre o *array*: que ele está ordenado. Conseguimos fazer algo mais eficiente?

## Exercício Resolvido (2)

- Supondo que temos uma informação extra sobre o *array*: que ele está ordenado. Conseguimos fazer algo mais eficiente?



Resposta: Começamos a pesquisa pela posição da metade. Se o elemento desejado for maior que o da metade, descartamos a primeira metade dos elementos. Se ele for menor, descartamos a segunda metade

Continuando o processo, considerando a metade não descartada

Essa é a pesquisa binária

- Pesquisa sequencial

- **Pesquisa binária** 

# Pesquisa Binária

- Exemplo: procurar o 35

<b>2</b>	<b>3</b>	<b>5</b>	<b>7</b>	<b>9</b>	<b>11</b>	<b>15</b>	<b>17</b>	<b>20</b>	<b>21</b>	<b>30</b>	<b>43</b>	<b>49</b>	<b>70</b>	<b>71</b>	<b>82</b>
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

# Pesquisa Binária

- Exemplo: procurar o 35

início																fim	
↓ esq																dir ↓	
2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		

## Pesquisa Binária

- Exemplo: procurar o 35

início

esq

↓

(início + fim) / 2

meio

↓

fim

dir

↓

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

# Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

resp false

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

resp false

esq ↓

dir ↓

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

(0 <= 15): true

resp false

esq ↓

dir ↓

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

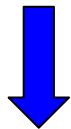
## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

$(0 + 15) / 2: 7$

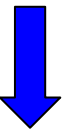
resp false



esq



meio



dir

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

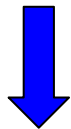
## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

(35 == 17): false

resp false



esq



meio



dir

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

resp false

(35 &gt; 17): true



esq



meio



dir

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

resp false

Com uma comparação, reduzimos o espaço de busca pela metade


## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

(8 <= 15): true

resp false

esq ↓

dir ↓

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

$(8 + 15) / 2: 11$

resp false

esq								meio				dir			
2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

(35 == 43): false

resp false

								esq					meio			dir
2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

resp false

(35 &gt; 43): false

esq ↓                      ↓ meio                      dir ↓

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

resp false

esq ↓      ↓ dir      ↓ meio

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

(8 <= 10): true

resp false

dir

esq

meio

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

$(8 + 10) / 2: 9$

resp false

esq ↓ meio ↓ dir ↓

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

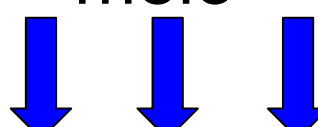
- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

(35 == 21): false

resp false

esq      meio      dir



2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

resp false

(35 &gt; 21): true

esq      meio      dir

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

resp false

meio  
↓  
esq  
↓  
dir

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;
```

```
while (esq <= dir) {
```

```
    meio = (esq + dir) / 2;
```

```
    if (x == array[meio]){
```

```
        resp = true;
```

```
        esq = n;
```

```
    } else if (x > array[meio]){
```

```
        esq = meio + 1;
```

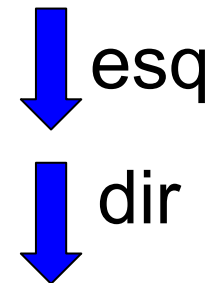
```
    } else {
```

```
        dir = meio - 1;
```

```
    } }
```

(10 <= 10): true

resp false



2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

- Exemplo: procurar o 35

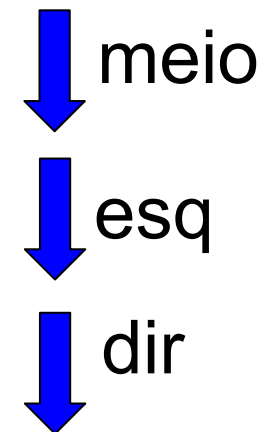
```

boolean resp = false;
int dir = n - 1, esq = 0, meio;
while (esq <= dir) {
    meio = (esq + dir) / 2;
    if (x == array[meio]){
        resp = true;
        esq = n;
    } else if (x > array[meio]){
        esq = meio + 1;
    } else {
        dir = meio - 1;
    }
}

```

$(10 + 10) / 2: 10$

resp false



2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

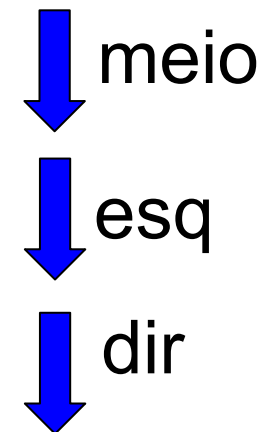
## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

(35 == 30): false

resp false



2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

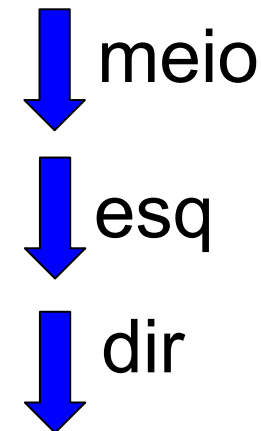
## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

(35 > 30): true

resp false



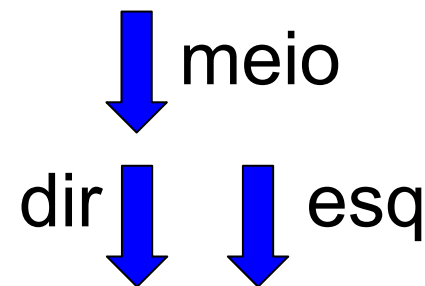
2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

- Exemplo: procurar o 35

```
boolean resp = false;  
int dir = n - 1, esq = 0, meio;  
while (esq <= dir) {  
    meio = (esq + dir) / 2;  
    if (x == array[meio]){  
        resp = true;  
        esq = n;  
    } else if (x > array[meio]){  
        esq = meio + 1;  
    } else {  
        dir = meio - 1;  
    }  
}
```

resp false



2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

## Pesquisa Binária

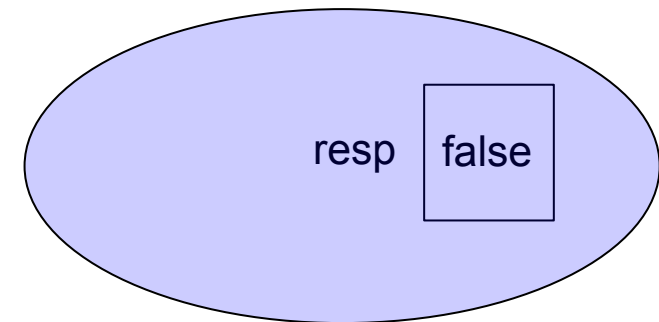
- Exemplo: procurar o 35

```

boolean resp = false;
int dir = n - 1, esq = 0, meio;
while (esq <= dir) {
    meio = (esq + dir) / 2;
    if (x == array[meio]){
        resp = true;
        esq = n;
    } else if (x > array[meio]){
        esq = meio + 1;
    } else {
        dir = meio - 1;
    }
}

```

(11 <= 10): false



dir ↓ ↓ esq

2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

# Pesquisa Binária

## • Exemplo

```
boolean resp = false;
int dir = n - 1, esq = 0, meio;
while (esq <= dir) {
    meio = (esq + dir) / 2;
    if (x == array[meio]){
        resp = true;
        esq = n;
    } else if (x > array[meio]){
        esq = meio + 1;
    } else {
        dir = meio - 1;
    }
}
```

1º) Qual é a operação relevante?

R: Comparação entre elementos do *array*.

2º) Quantas vezes ela será executada?



# Pesquisa Binária

## • Exemplo

```
boolean resp = false;
int dir = n - 1, esq = 0, meio;
while (esq <= dir) {
    meio = (esq + dir) / 2;
    if (x == array[meio]){
        resp = true;
        esq = n;
    } else if (x > array[meio]){
        esq = meio + 1;
    } else {
        dir = meio - 1;
    }
}
```

1º) Qual é a operação relevante?

R: Comparação entre elementos do *array*.

2º) Quantas vezes ela será executada?

R: Melhor caso:  $f(n) = 1$   
Pior caso:  $f(n) = \lg(n)$

## Exercício Resolvido (3)

- Quando acontece o melhor e o pior caso do algoritmo de pesquisa binária?

## Exercício Resolvido (3)

- Quando acontece o melhor e o pior caso do algoritmo de pesquisa binária?

### Resposta:



O melhor caso acontece quando o elemento procurado está na primeira posição de procura

O pior acontece quando o elemento desejado está na última posição de procura ou quando ele não está no *array*

## Exercício (1)

- A solução apresentada para pesquisa binária faz **duas comparações entre elementos do *array*** em cada execução da repetição, modifique o código para efetuarmos apenas **uma**