

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: $f(n) = 1 = \Theta(1)$

Pior caso: $f(n) = n = \Theta(n)$

Caso médio: $f(n) = (n + 1) / 2 = \Theta(n)$

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:

Melhor caso: Elemento procurado está na primeira posição

Pior caso: Ele está na última posição OR 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? Como?

Exercício Resolvido (2)

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



Resposta: Sim.

Pesquisamos a partir da metade do *array*. Se o elemento procurado for maior que o da metade, descartamos a primeira metade do *array*; senão, a segunda metade. Repetimos o processo com a metade não descartada

Essa é a pesquisa binária

- Pesquisa sequencial

- **Pesquisa binária** 

Pesquisa Binária

- Exemplo: procurar o 35

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

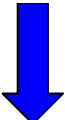
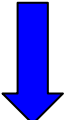

Pesquisa Binária

- Exemplo: procurar o 35

início																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		$(\text{início} + \text{fim}) / 2$										fim			
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;  
    }  
}
```

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

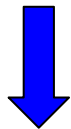
Pesquisa Binária

- 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;  
    }  
}
```

$(0 + 15) / 2: 7$

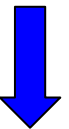
resp false



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

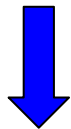
Pesquisa Binária

- 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

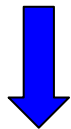
Pesquisa Binária

- 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 > 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 > 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 > 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;
```

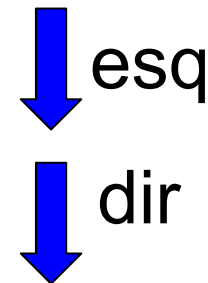
```
    } else {
```

```
        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

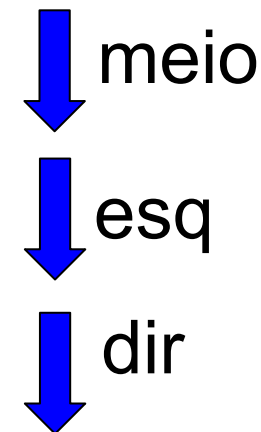
Pesquisa Binária

- 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

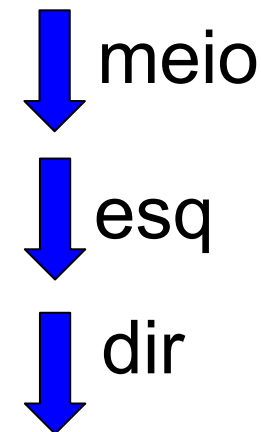
```

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

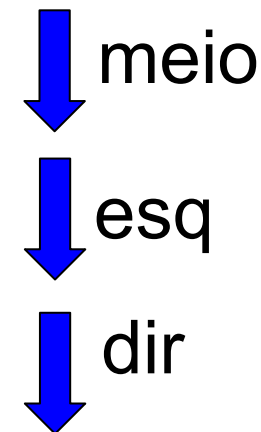
Pesquisa Binária

- 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



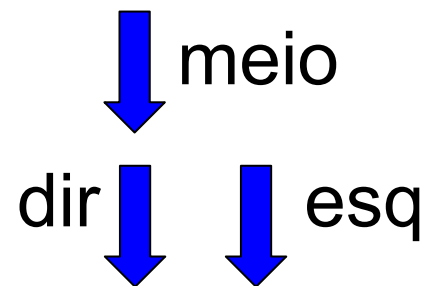
2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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



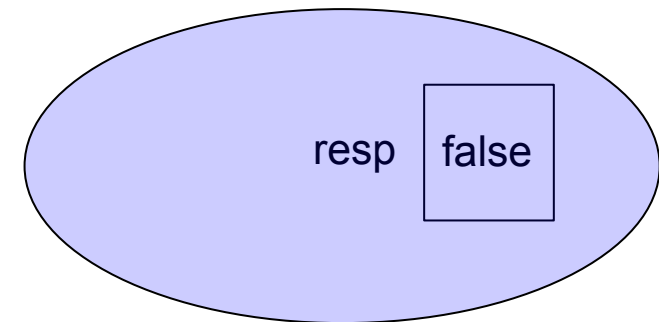
2	3	5	7	9	11	15	17	20	21	30	43	49	70	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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: $f(n) = 1 = \Theta(1)$

Pior caso: $f(n) = 2 \times \lg(n) = \Theta(\lg n)$

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:



Melhor caso: Elemento procurado está na metade do *array*

Pior caso: Está na última posição de procura ($\lg n$) ou 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**