

Unidade III:


Ordenação Interna - Quicksort



PUC Minas

Instituto de Ciências Exatas e Informática
Departamento de Ciência da Computação

- Funcionamento básico
- Algoritmo em C *like*
- Análise dos número de comparações e movimentações
- Escolha do Pivô
- Conclusão

- **Funcionamento básico** 
- Algoritmo em C *like*
- Análise dos número de comparações e movimentações
- Escolha do Pivô
- Conclusão

Introdução

- Proposto por Hoare em 1960 e publicado em 1962
- Algoritmo de ordenação mais rápido para a maioria das situações
- Provavelmente, ele é o mais utilizado
- Algoritmo do tipo dividir para conquistar

Funcionamento Básico


- Divide o *array* em duas partes que serão independentemente ordenadas e a combinação de seus resultados produz a solução final
 - A parte da esquerda terá elementos menores ou iguais a um pivô
 - A parte da direita terá elementos maiores ou iguais a um pivô

Funcionamento Básico

- Particionamento:
 - Escolha arbitrariamente um pivô
 - Percorra o *array* a partir da esquerda enquanto $array[i] < \text{pivô}$
 - Percorra o *array* a partir da direita enquanto $array[j] > \text{pivô}$
 - Se $i \leq j$ então troque $array[i]$ com $array[j]$
 - Continue o processo enquanto $i \leq j$

Funcionamento Básico

- No final do particionamento, o *array* estará particionado de tal forma que:
 - Os elementos *array[esq]*, *array[esq+1]*, . . . , *array[j]* são \leq que pivô
 - Os elementos *array[i]*, *array[i+1]*, . . . , *array[dir]* são \geq que pivô

- Funcionamento básico
- **Algoritmo em C *like*** 
- Análise dos número de comparações e movimentações
- Escolha do Pivô
- Conclusão

Algoritmo em C like

```

void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   swap(i, j);   i++;   j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
    
```

Na primeira chamada, teremos:
quicksort(0, n-1)

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Algoritmo em C like

```

void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   swap(i, j);   i++;   j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
    
```

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Algoritmo em C like

```

void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   swap(i, j);   i++;   j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
    
```

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

Algoritmo em C like

```

void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
    
```

i															
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

Algoritmo em C like

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {
            swap(i, j);
            i++;
            j--;
        }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

<div><div>i</div><div>j</div></div>															
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<div><div>esq</div><div>dir</div></div>															

10

pivô

10

$(0 + 15) / 2: 7$

pivô

10

Algoritmos e Estruturas de Dados II (14)

Algoritmo em C like

pivô	10
------	----

0 <= 15: true

```
void quicksort(int esq, int dir) {  
    int i = esq,    j = dir,    pivo = array[(esq+dir)/2];  
    while (i <= j) {  
        while (array[i] < pivo)  
            i++;  
        while (array[j] > pivo)  
            j--;  
        if (i <= j)  
            { swap(i, j);    i++;    j--; }  
    }  
    if (esq < j)  
        quicksort(esq, j);  
    if (i < dir)  
        quicksort(i, dir);  
}
```

<div><div>i</div><div>j</div></div>															
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<div><div>esq</div><div>dir</div></div>															

Algoritmo em C like

pivô

10

0 < 10: true

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {
            swap(i, j);    i++;    j--;
        }
        if (esq < j)
            quicksort(esq, j);
        if (i < dir)
            quicksort(i, dir);
    }
}
```

i

j

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

esq dir

Algoritmo em C like

pivô

10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i

j

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

Algoritmo em C like

pivô 10

1 < 10: true

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i

j

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

Algoritmo em C like

pivô

10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i

j

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

Algoritmo em C like

pivô

10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

			<i>i</i>												<i>j</i>	
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
<i>esq</i>																<i>dir</i>

Algoritmo em C like

pivô

10

3 < 10: true

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

			<i>i</i>												<i>j</i>	
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
<i>esq</i>																<i>dir</i>

Algoritmo em C like

pivô

10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

				i											j
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

Algoritmo em C like

pivô 10

15 < 10: false

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

				i													j
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
esq																	dir

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

82 > 10: true

				<i>i</i>												<i>j</i>	
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
<i>esq</i>																<i>dir</i>	

Algoritmo em C like

pivô

10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

				i											j	
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

71 > 10: true

				i											j	
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

				<i>i</i>												<i>j</i>	
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
<i>esq</i>																	<i>dir</i>

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

48 > 10: true

				i									j			
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

				i					j							
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq															dir	

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

20 > 10: true

				i									j		
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															
														dir	

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

				<i>i</i>								<i>j</i>			
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<i>esq</i>												<i>dir</i>			

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

5 > 10: false

				<i>i</i>								<i>j</i>			
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<i>esq</i>															
												<i>dir</i>			

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

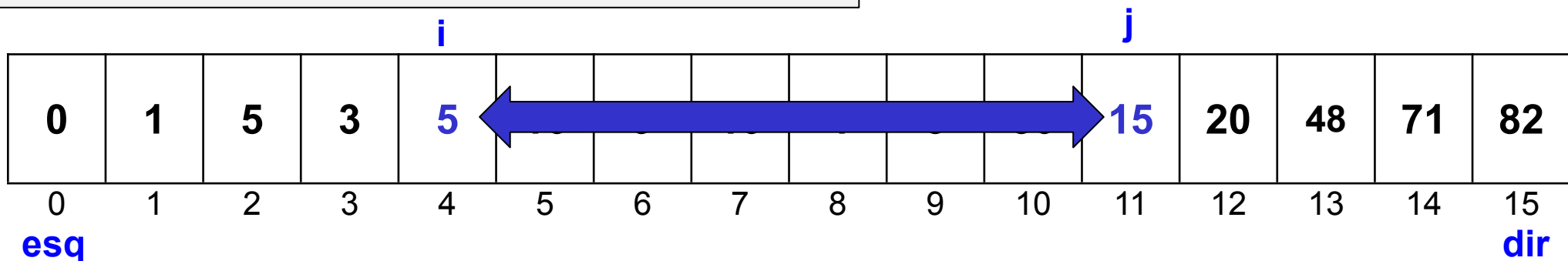
4 <= 11: true

				i												j	
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
esq																	dir

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```



Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {
            swap(i, j);
            i++;
            j--;
        }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

Algoritmo em C like

pivô

10

5 <= 10: true

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   swap(i, j);   i++;   j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

					i						j					
0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq															dir	

Algoritmo em C like

pivô 10

16 < 10: false

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

					i						j				
0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															
															dir

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

30 > 10: true

					i					j					
0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

Algoritmo em C like

pivô

10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

					i				j							
0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

3 > 10: false

					i				j							
0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

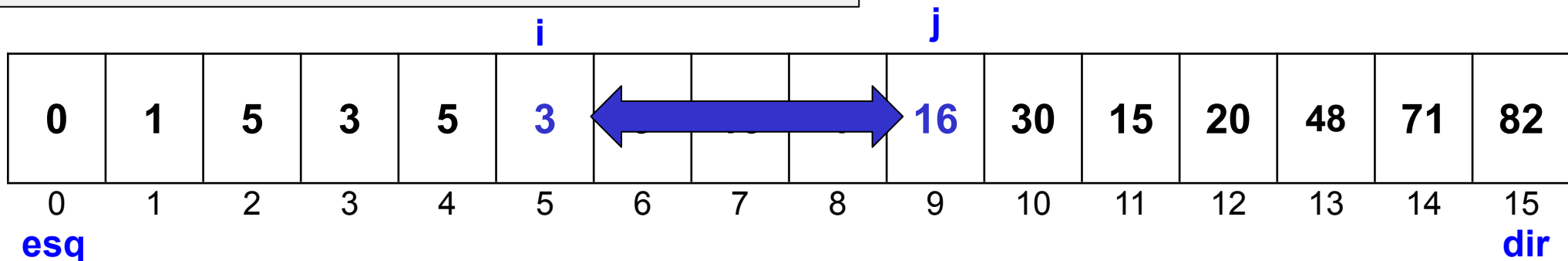
5 <= 9: true

					i				j							
0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```



Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {
            swap(i, j);
            i++; j--;
        }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

						i	j									
0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

Algoritmo em C like

pivô

10

6 <= 8: true

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   swap(i, j);   i++;   j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

						i	j									
0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

Algoritmo em C like

pivô

10

9 < 10: true

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

Algoritmo em C like

pivô

10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

							i	j								
0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

Algoritmo em C like

pivô 10

10 < 10: false

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

							i	j								
0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
        if (esq < j)
            quicksort(esq, j);
        if (i < dir)
            quicksort(i, dir);
    }
}
```

4 > 10: false

0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

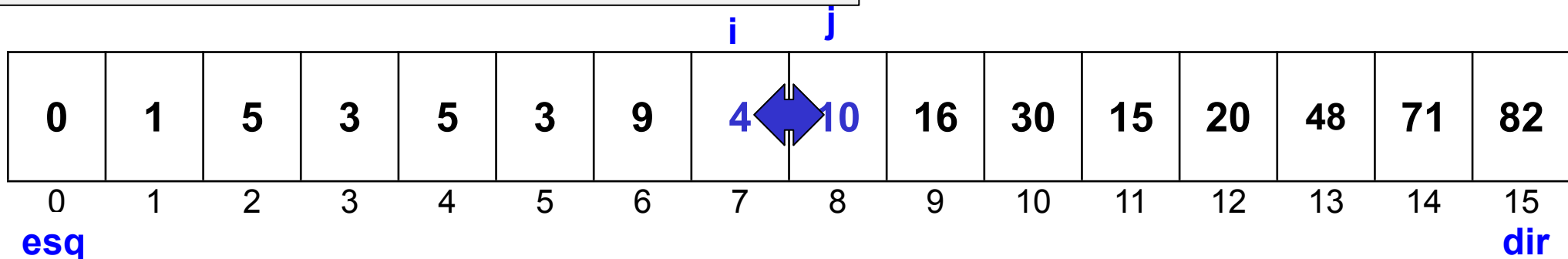
7 <= 8: true

							i	j								
0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```



Algoritmo em C like

pivô

10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {
            swap(i, j);
            i++; j--;
        }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

Algoritmo em C like

pivô

10

8 <= 7: false

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   swap(i, j);   i++;   j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

Algoritmo em C like

pivô 10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
}
```

0 < 7: true

```
if (esq < j)
    quicksort(esq, j);
if (i < dir)
    quicksort(i, dir);
}
```

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							j	i							dir

Algoritmo em C like

pivô

10

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							j	i							dir

Algoritmo em C like

pivô

?

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

Obs(1): esq e dir são 0 e 7, respectivamente.

Obs(2): o pivô será definido novamente.

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

```

void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
    
```

i															
0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq								dir							

Algoritmo em C like

```

void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   swap(i, j);   i++;   j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
    
```

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

```

void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}

```

 $(0 + 7) / 2 = 3$

pivô

3

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

0 <= 7: true

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   swap(i, j);   i++;   j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

0 < 3: true

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

1 < 3: true

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

5 < 3: false

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

4 > 3: true

		i					j								
0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq								dir							

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

9 > 3: true

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq			dir												

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

3 > 3: false

		i			j											
0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq							dir									

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

2 <= 5: true

		i			j											
0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq							dir									

Algoritmo em C like

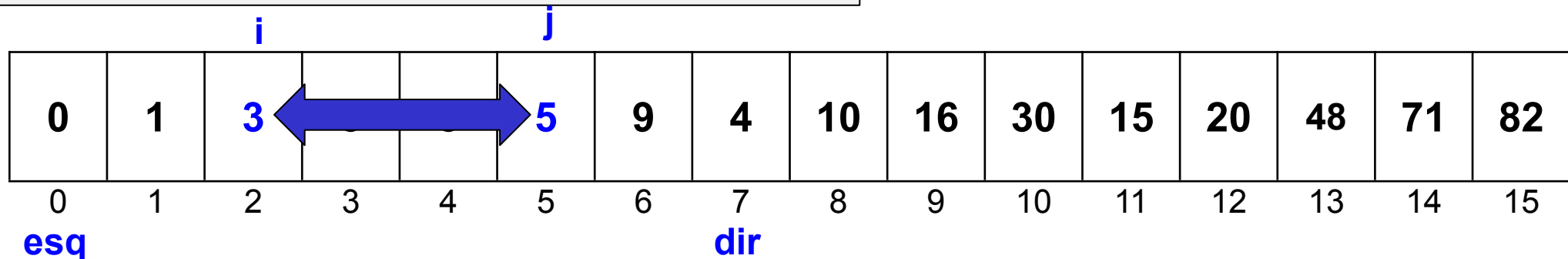
pivô

3

```

void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}

```



Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {
            swap(i, j);
            i++; j--;
        }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i j

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

3 <= 4: true

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   swap(i, j);   i++;   j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i j

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

3 < 3: false

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

			i	j											
0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
        if (esq < j)
            quicksort(esq, j);
        if (i < dir)
            quicksort(i, dir);
    }
}
```

5 > 3: true

			i	j											
0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i j

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            swap(i, j);
        if (esq < j)
            quicksort(esq, j);
        if (i < dir)
            quicksort(i, dir);
    }
}
```

3 > 3: false

i j

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

3 <= 3: true

i j

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

i j

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq				dir											

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {  
    int i = esq, j = dir, pivo = array[(esq+dir)/2];  
    while (i <= j) {  
        while (array[i] < pivo)  
            i++;  
        while (array[j] > pivo)  
            j--;  
        if (i <= j)  
        {  
            swap(i, j);  
            i++; j--;  
        }  
    }  
    if (esq < j)  
        quicksort(esq, j);  
    if (i < dir)  
        quicksort(i, dir);  
}
```

Obs: O elemento [3] está na posição correta

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq				dir											

Algoritmo em C like

pivô

3

4 <= 2: false

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   swap(i, j);   i++;   j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

j

i

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
}
```

0 < 2: true

if (esq < j)

quicksort(esq, j);

if (i < dir)

quicksort(i, dir);

}

j

i

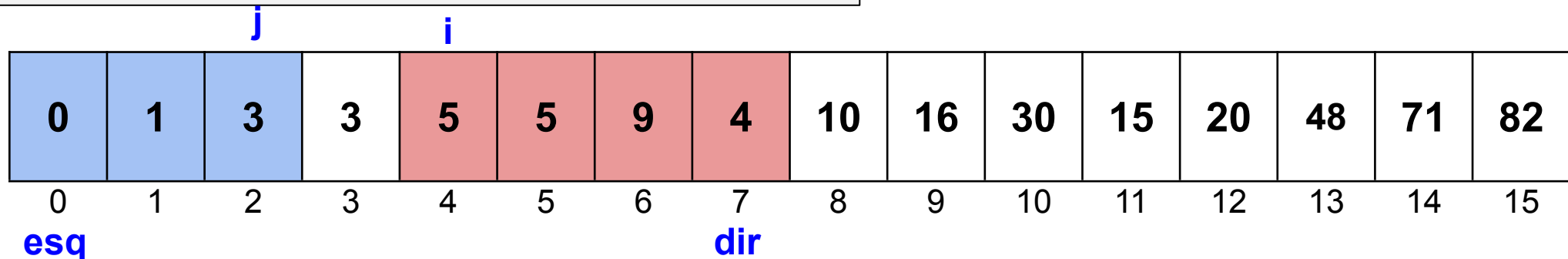
0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

Algoritmo em C like

pivô

3

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```



Algoritmo em C like

pivô 10

Voltando para a primeira chamada do Quicksort

8 < 15: true

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

0	1	3	3	4	5	5	9	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir


Algoritmo em C like

pivô 10

Voltando para a primeira chamada do Quicksort

```
void quicksort(int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { swap(i, j); i++; j--; }
    }
    if (esq < j)
        quicksort(esq, j);
    if (i < dir)
        quicksort(i, dir);
}
```

0	1	3	3	4	5	5	9	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

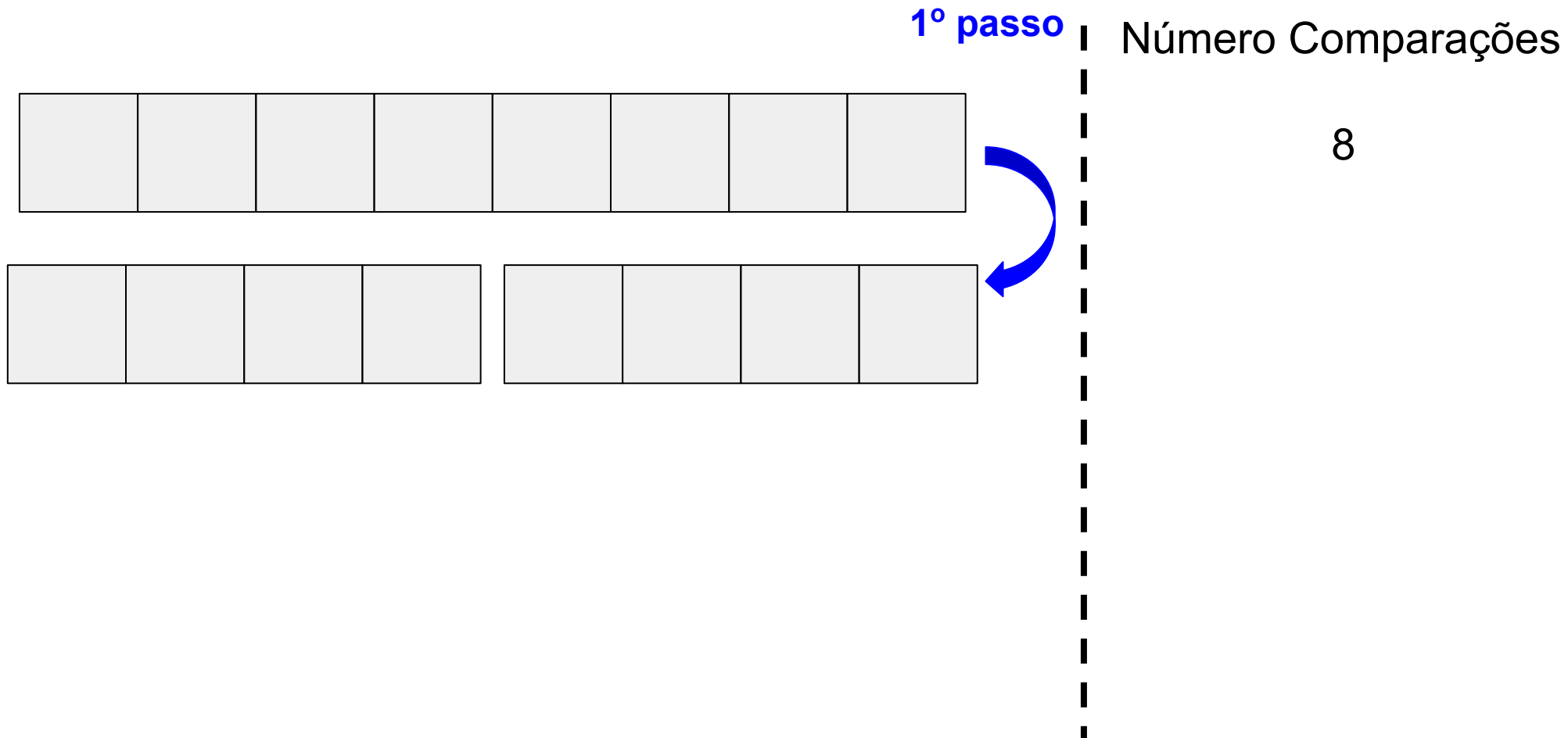
- Funcionamento básico
- Algoritmo em C *like*
- **Análise dos número de comparações e movimentações** 
- Escolha do Pivô
- Conclusão

Análise do Número de Comparações

- **Melhor caso:** Sistemáticamente, cada partição divide o *array* em duas partes iguais

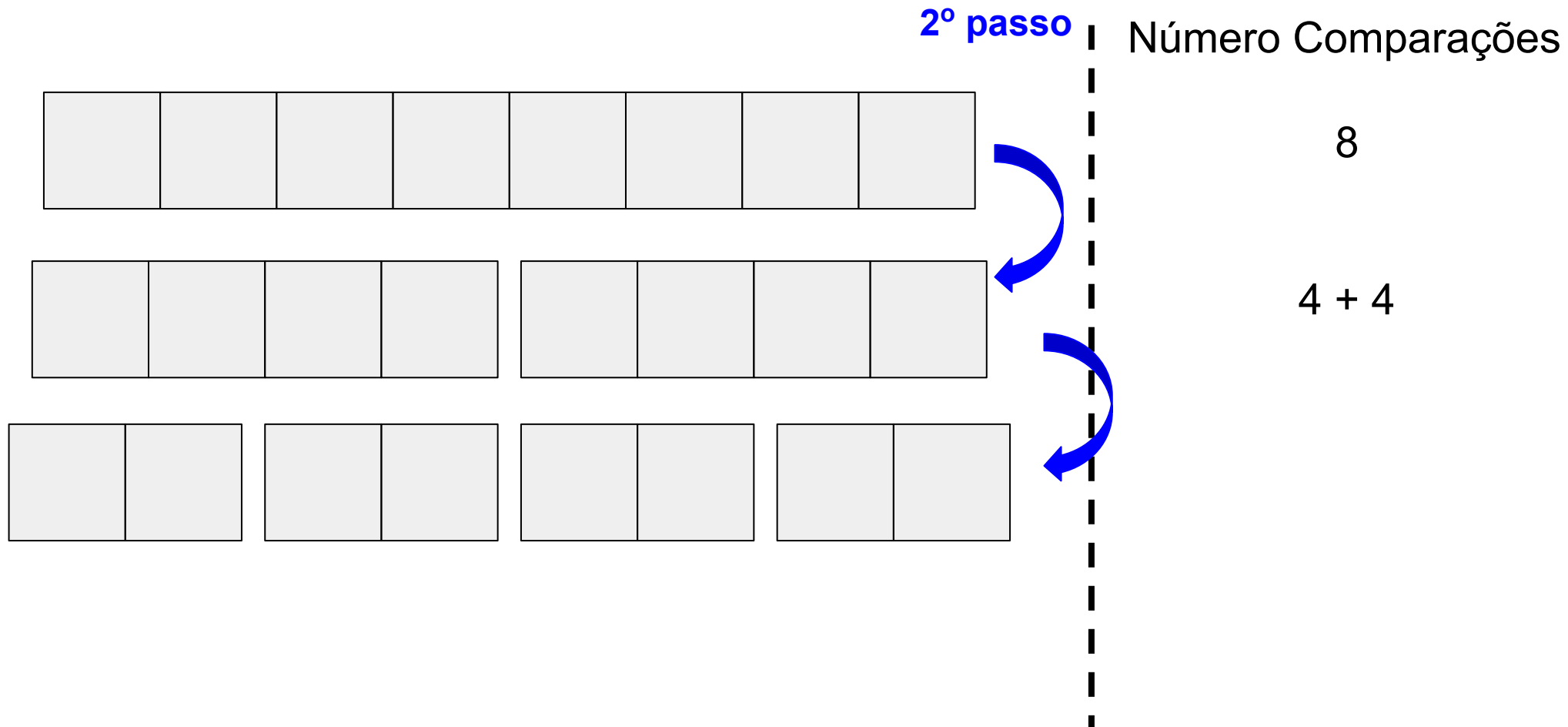
Análise do Número de Comparações

- **Melhor caso:** Sistemáticamente, cada partição divide o *array* em duas partes iguais



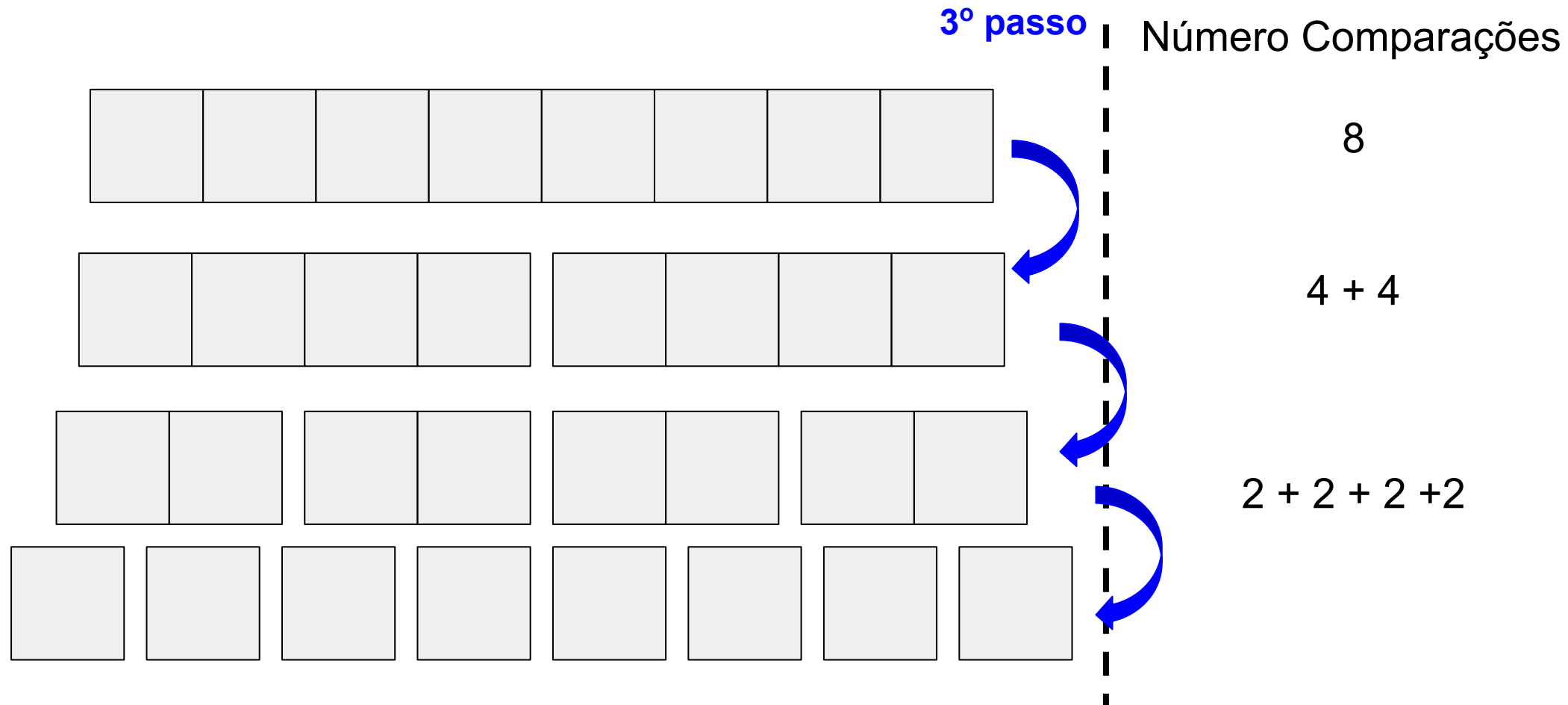
Análise do Número de Comparações

- **Melhor caso:** Sistemáticamente, cada partição divide o *array* em duas partes iguais



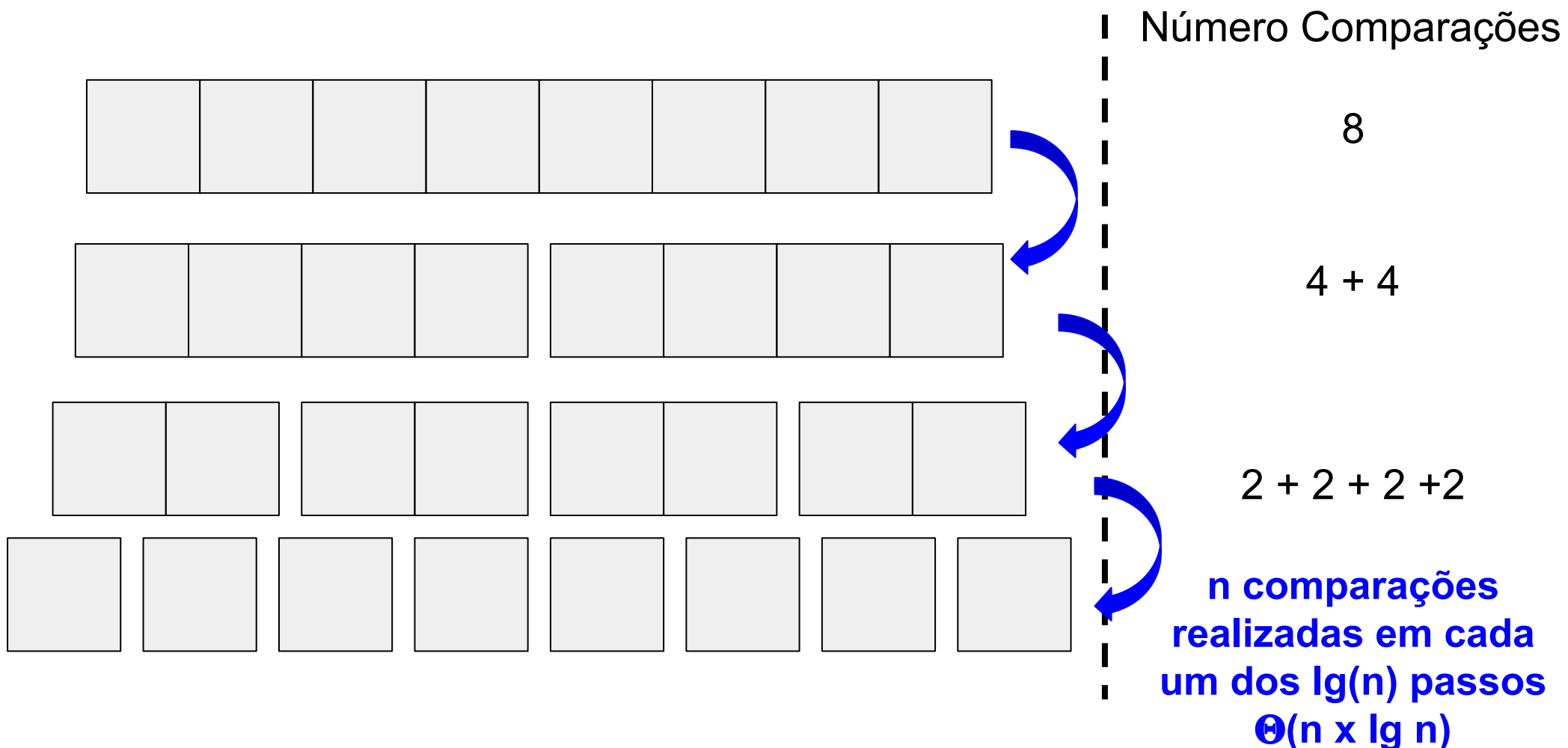
Análise do Número de Comparações

- **Melhor caso:** Sistemáticamente, cada partição divide o *array* em duas partes iguais



Análise do Número de Comparações

- **Melhor caso:** Sistemáticamente, cada partição divide o *array* em duas partes iguais



Análise do Número de Comparações

- **Melhor caso:** Sistemáticamente, cada partição divide o *array* em duas partes iguais

$$C(n) = 2 * C\left(\frac{n}{2}\right) + n = n * \lg(n) - n + 1$$

A análise de complexidade do Quicksort depende de equação de recorrência (vista nas disciplinas Teoria dos Grafos e Computabilidade e Projeto e Análise de Algoritmos)

Análise do Número de Comparações

- **Pior caso:** Sistemáticamente, o pivô é o menor ou o maior elemento do *array*, eliminando um elemento em cada chamada do algoritmo

Análise do Número de Comparações

- **Pior caso:** Sistemáticamente, o pivô é o menor ou o maior elemento do *array*, eliminando um elemento em cada chamada do algoritmo

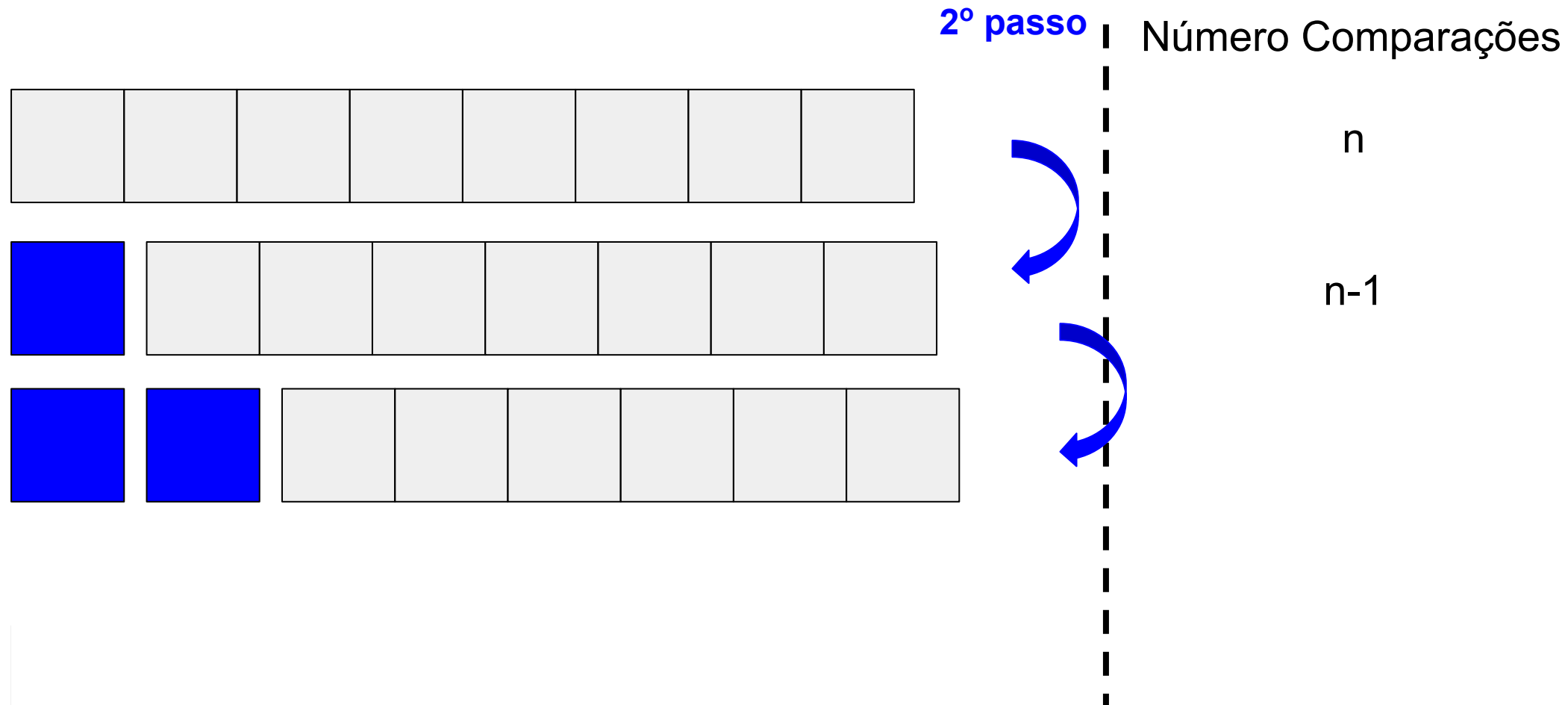
1º passo | Número Comparações

n



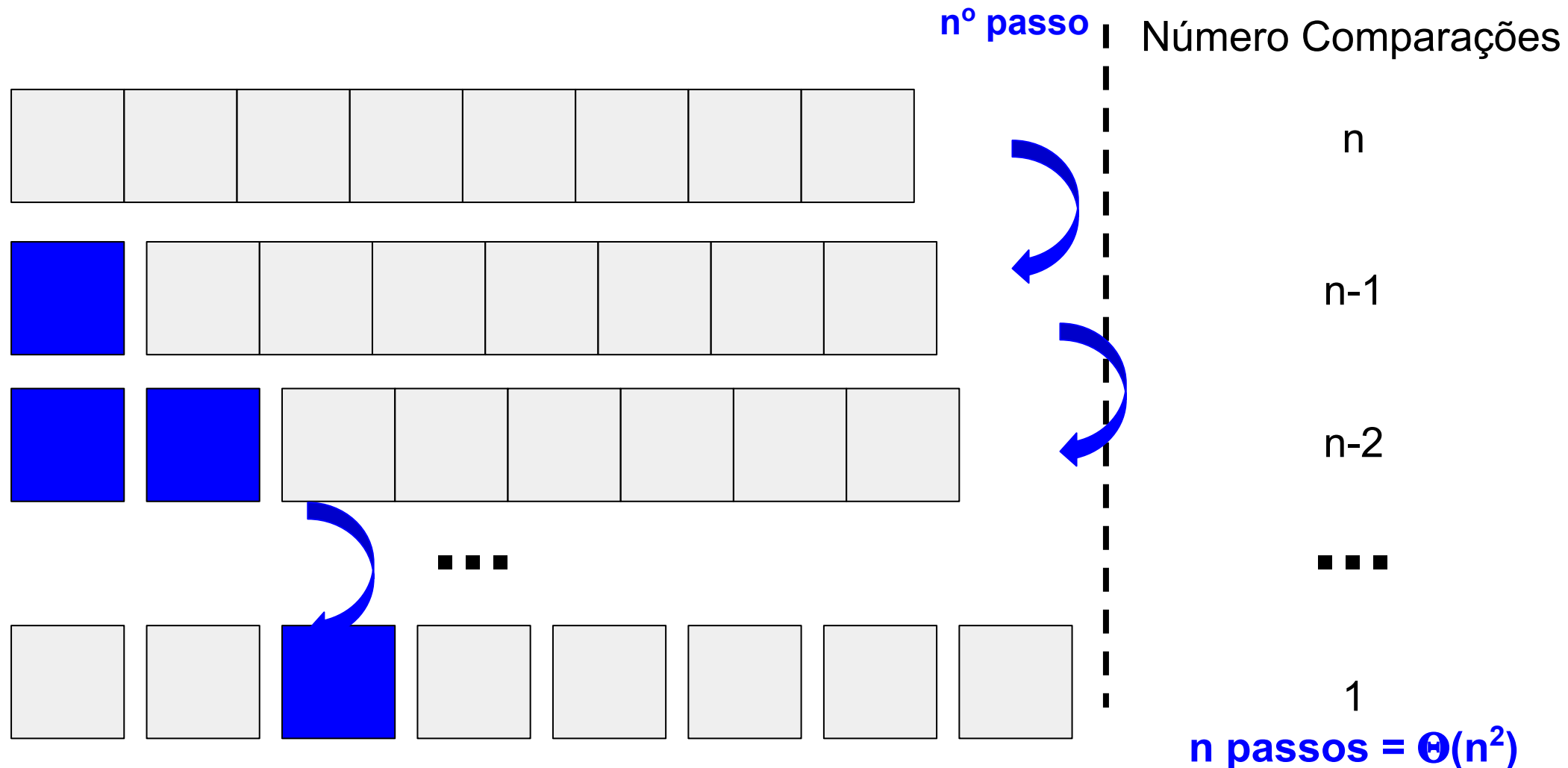
Análise do Número de Comparações

- Pior caso:** Sistemáticamente, o pivô é o menor ou o maior elemento do *array*, eliminando um elemento em cada chamada do algoritmo



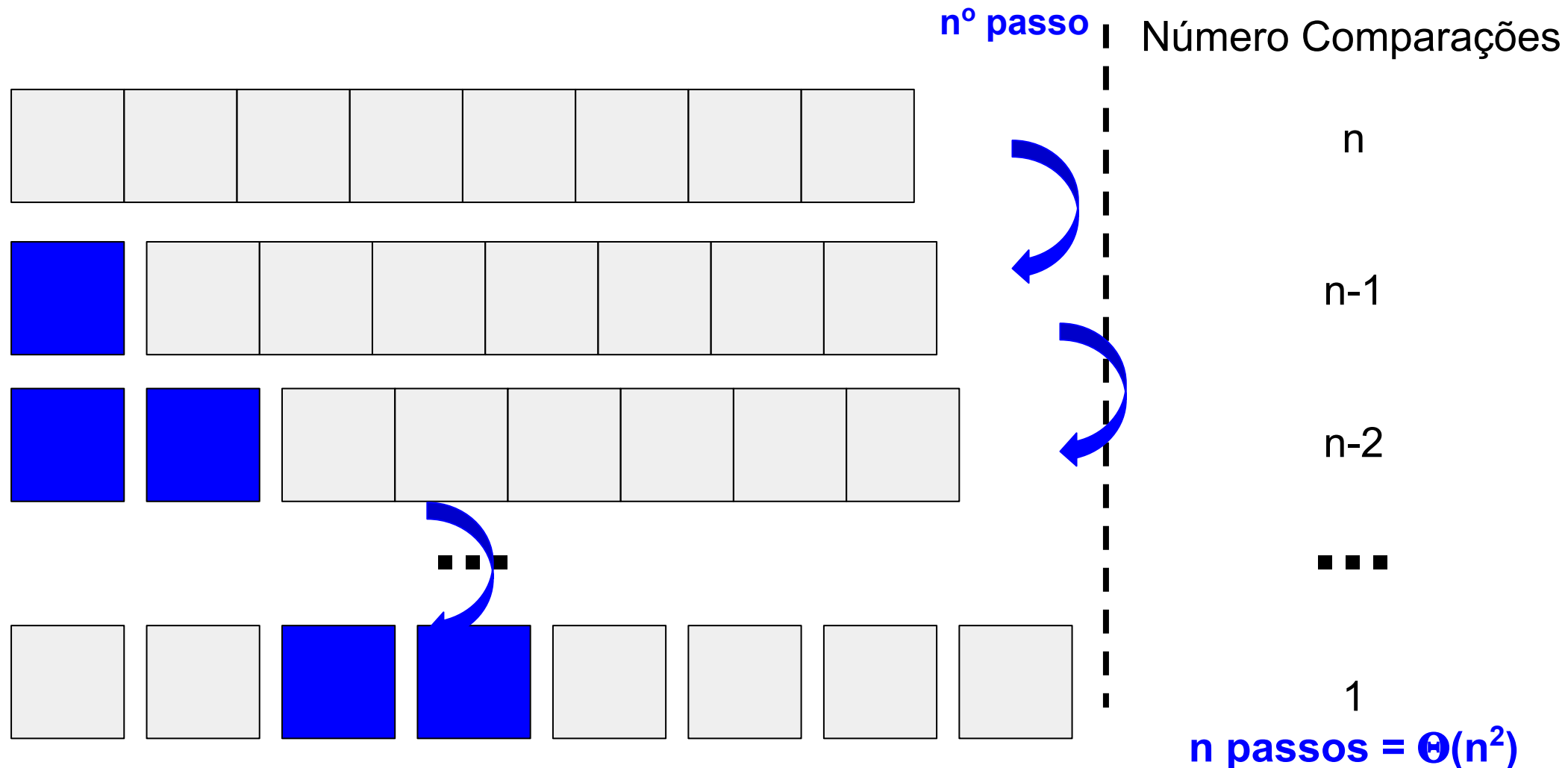
Análise do Número de Comparações

- Pior caso:** Sistemáticamente, o pivô é o menor ou o maior elemento do *array*, eliminando um elemento em cada chamada do algoritmo



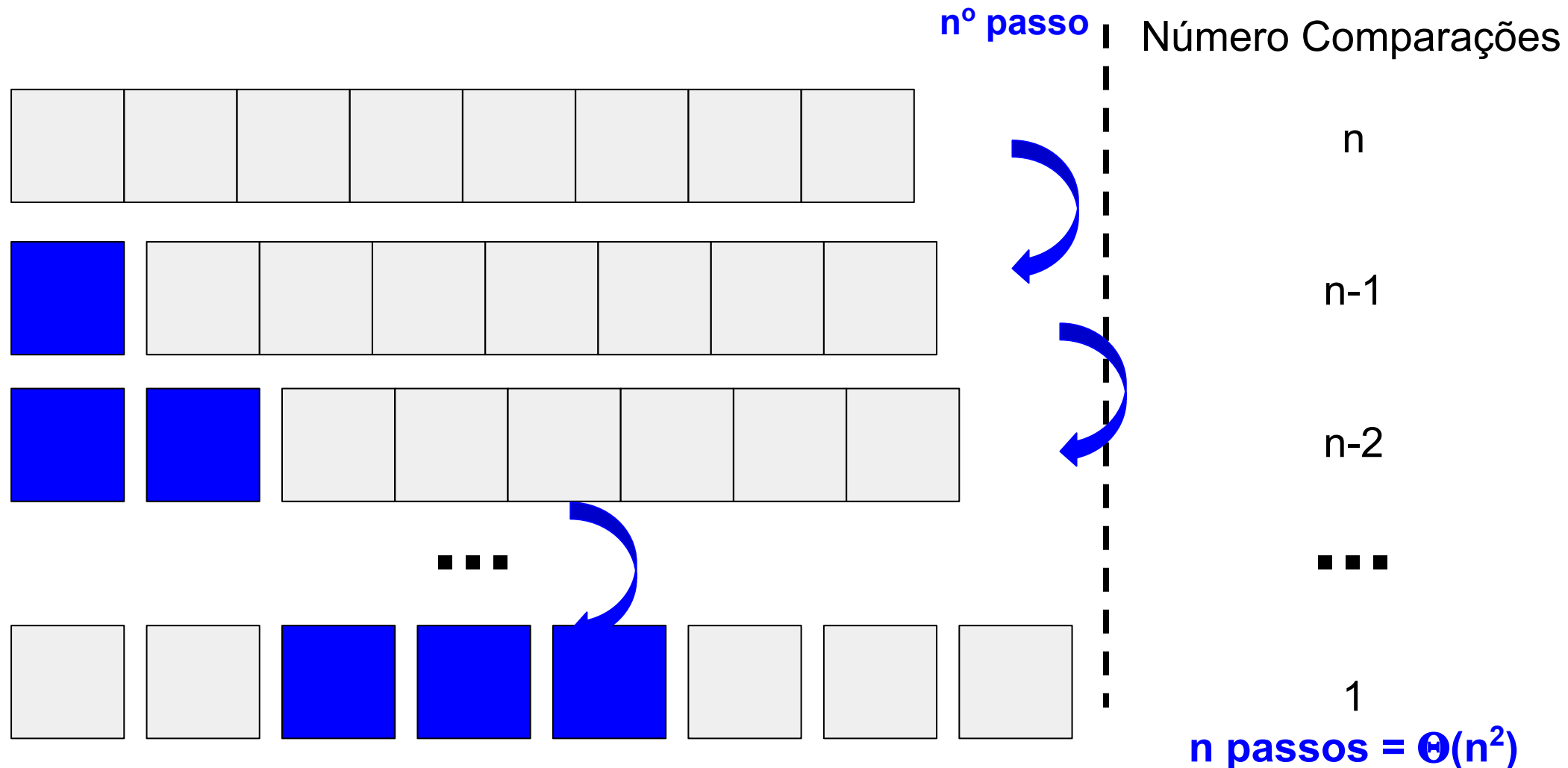
Análise do Número de Comparações

- Pior caso:** Sistemáticamente, o pivô é o menor ou o maior elemento do *array*, eliminando um elemento em cada chamada do algoritmo



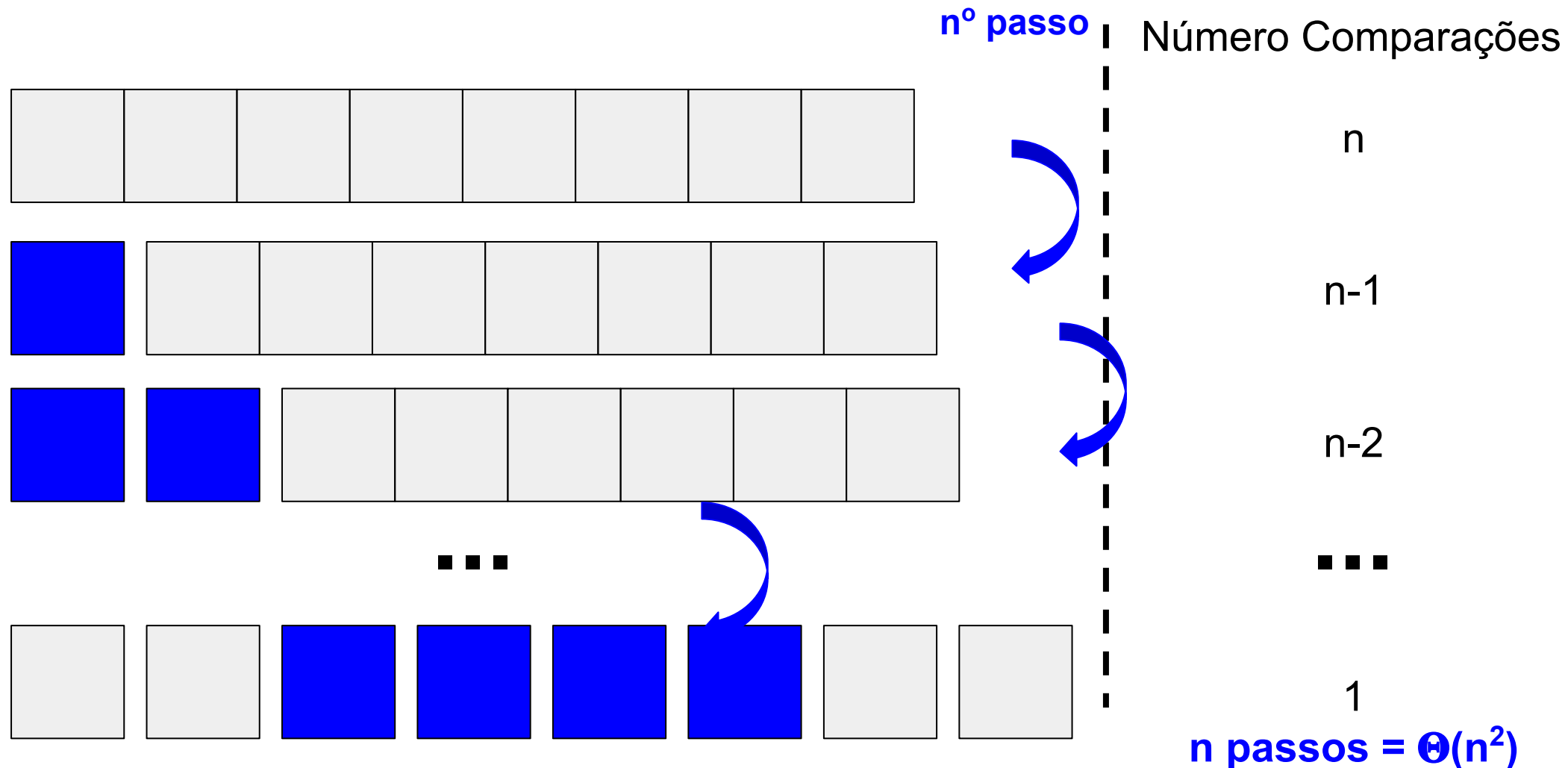
Análise do Número de Comparações

- Pior caso:** Sistemáticamente, o pivô é o menor ou o maior elemento do *array*, eliminando um elemento em cada chamada do algoritmo



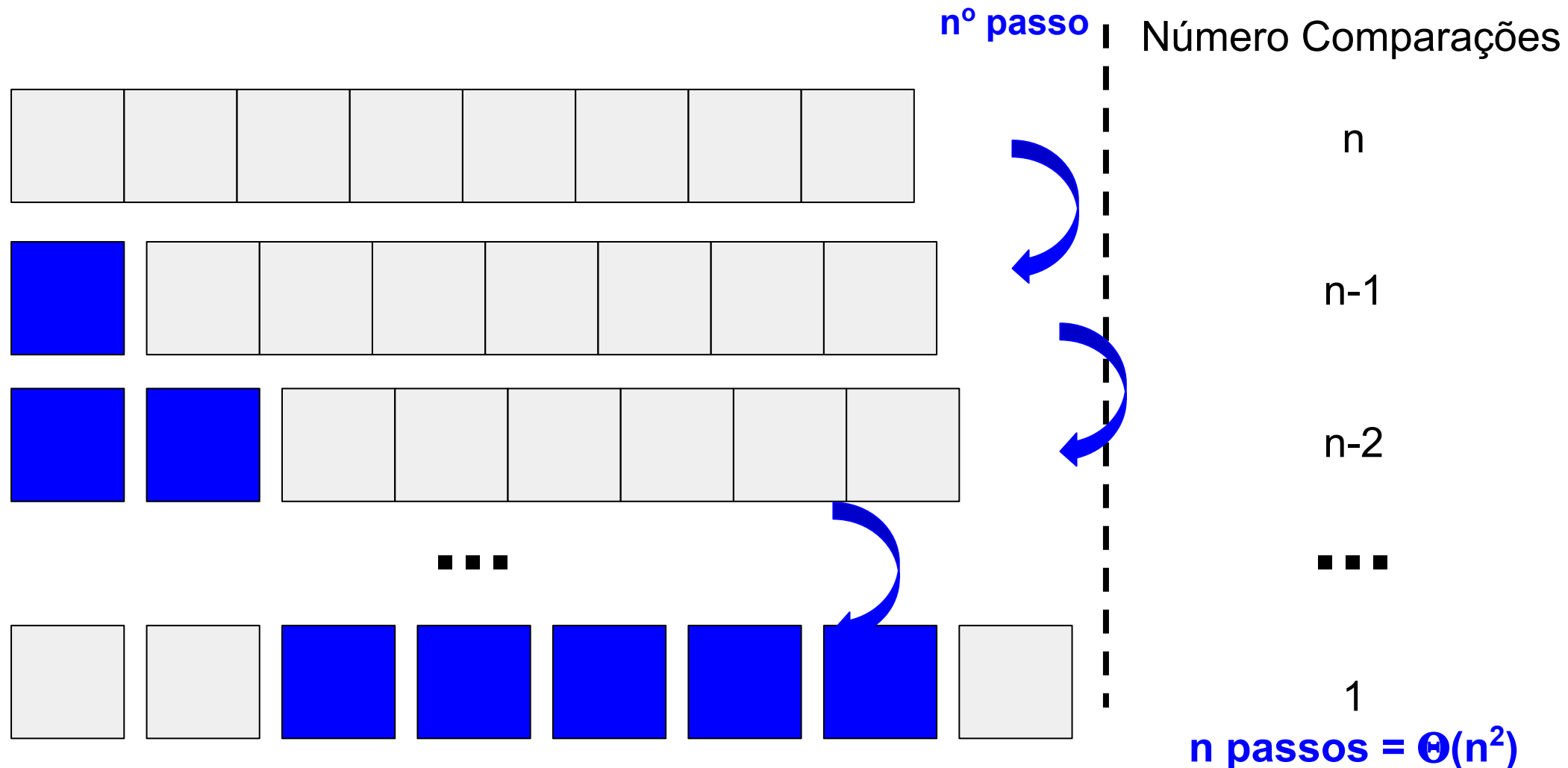
Análise do Número de Comparações

- Pior caso:** Sistemáticamente, o pivô é o menor ou o maior elemento do *array*, eliminando um elemento em cada chamada do algoritmo



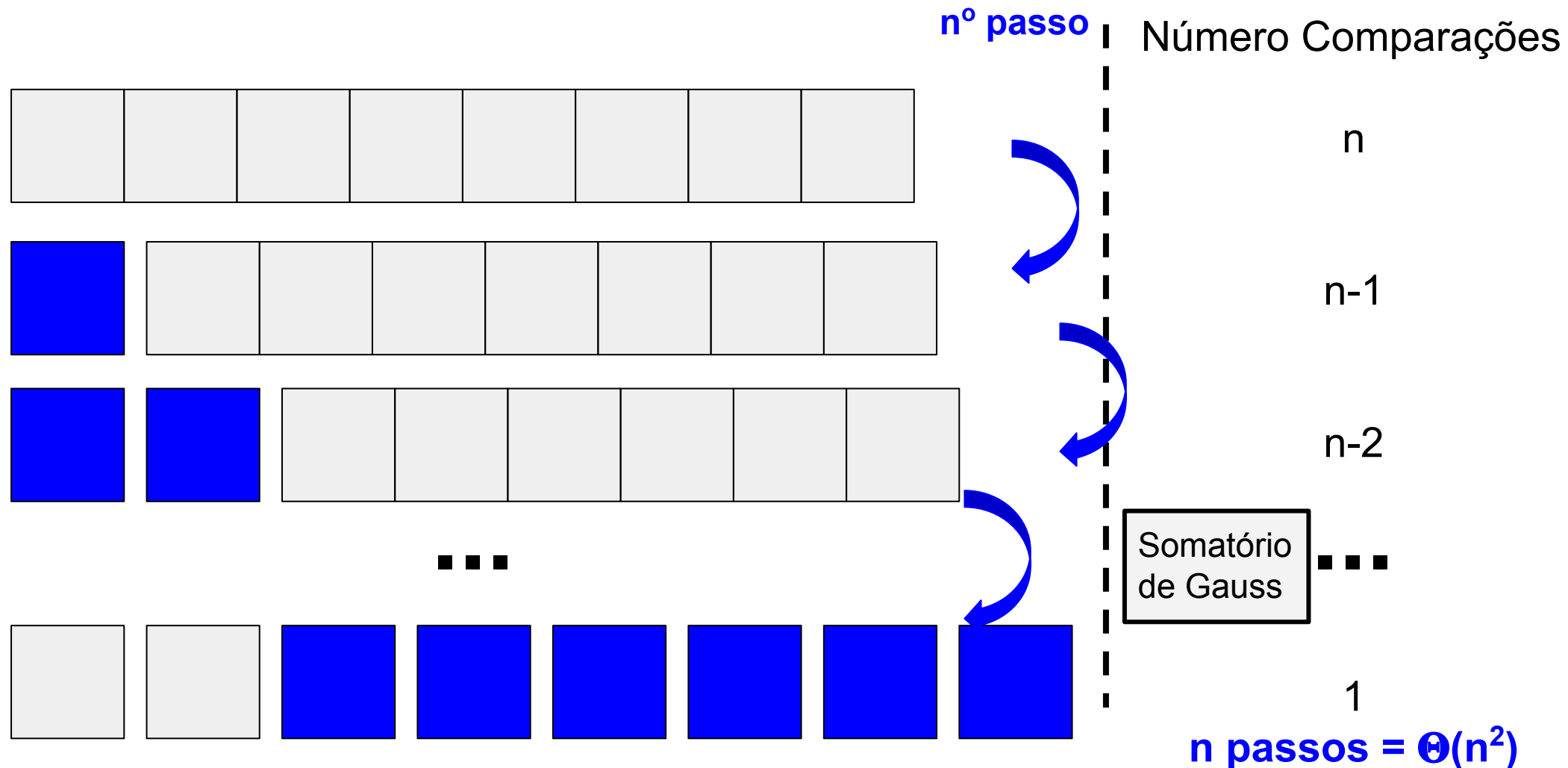
Análise do Número de Comparações

- Pior caso:** Sistemáticamente, o pivô é o menor ou o maior elemento do *array*, eliminando um elemento em cada chamada do algoritmo



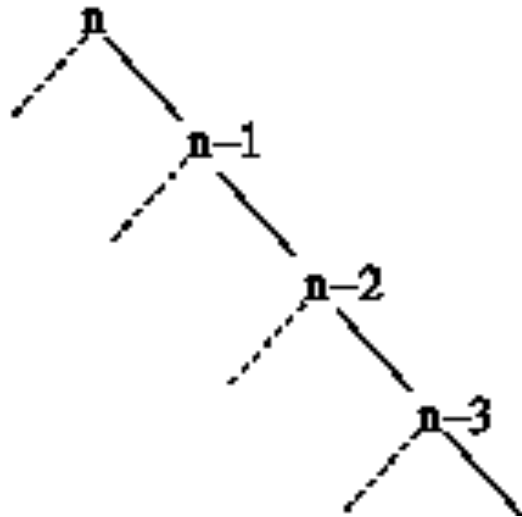
Análise do Número de Comparações

- Pior caso:** Sistemáticamente, o pivô é o menor ou o maior elemento do *array*, eliminando um elemento em cada chamada do algoritmo



Análise do Número de Comparações

- **Pior caso:** Sistemáticamente, o pivô é o menor ou o maior elemento do *array*, eliminando um elemento em cada chamada do algoritmo



$$C(n) = \Theta(n^2)$$

- Existem diversas técnicas para **evitar** o pior caso como, por exemplo, fazer com que o pivô seja a mediana de três elementos do *array*

Análise do Número de Comparações

- **Caso Médio:** Sedgewick e Flajolet (1996, p. 17):

$$C(n) \approx 1,386 * n * \lg(n) - 0,846 * n$$

Análise do Número de Comparações

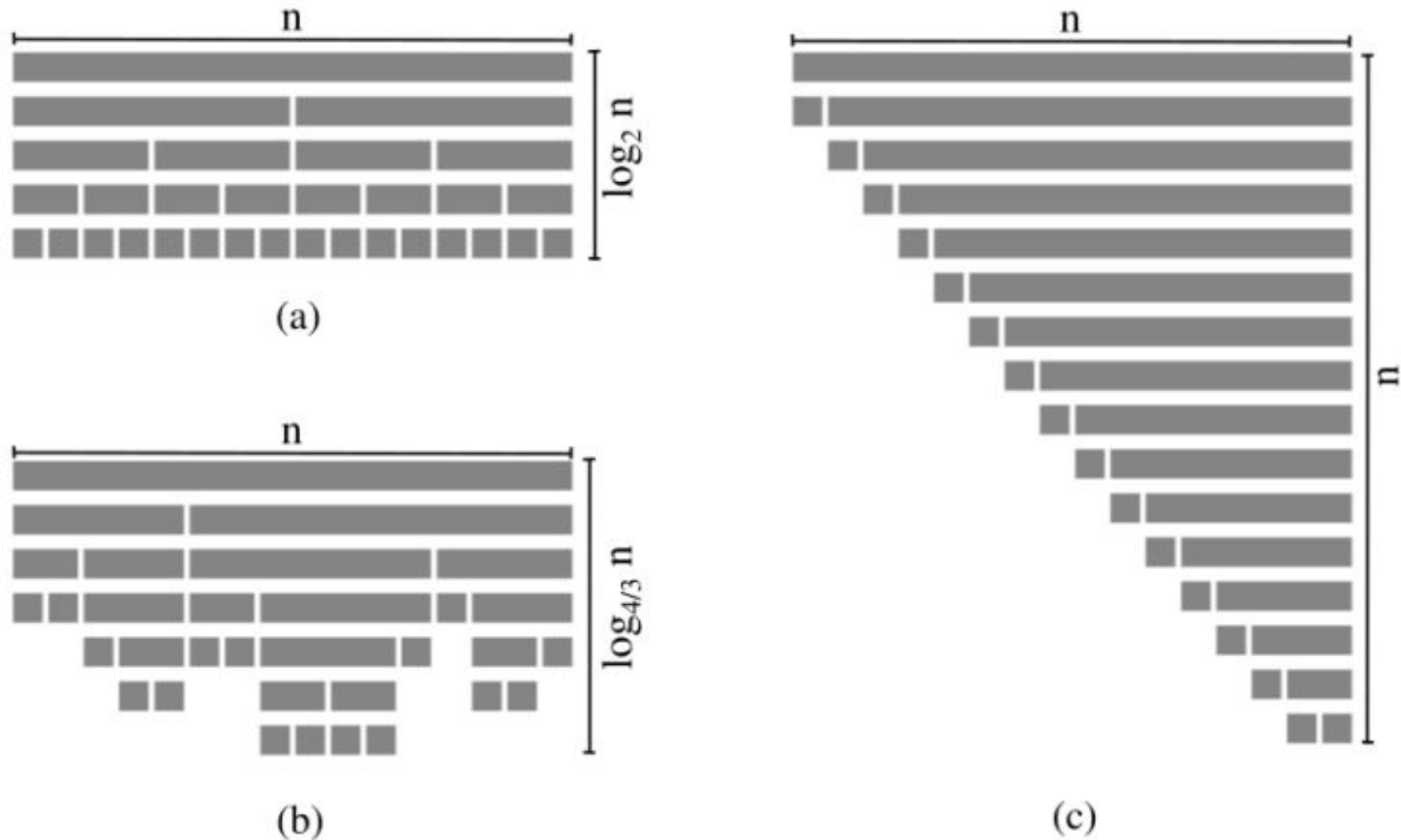



Figura: (a) Melhor caso. (b) Caso médio. (c) Pior caso.

Análise do Número de Movimentações

- No **pior caso**, há $\lceil n/2 \rceil$ trocas em cada execução da função de partição
- Nesse caso, o pivô está no meio do *array* e os elementos superiores estão sistematicamente no início da lista e; os inferiores, no fim
- Lembrando que em cada troca temos 3 movimentações

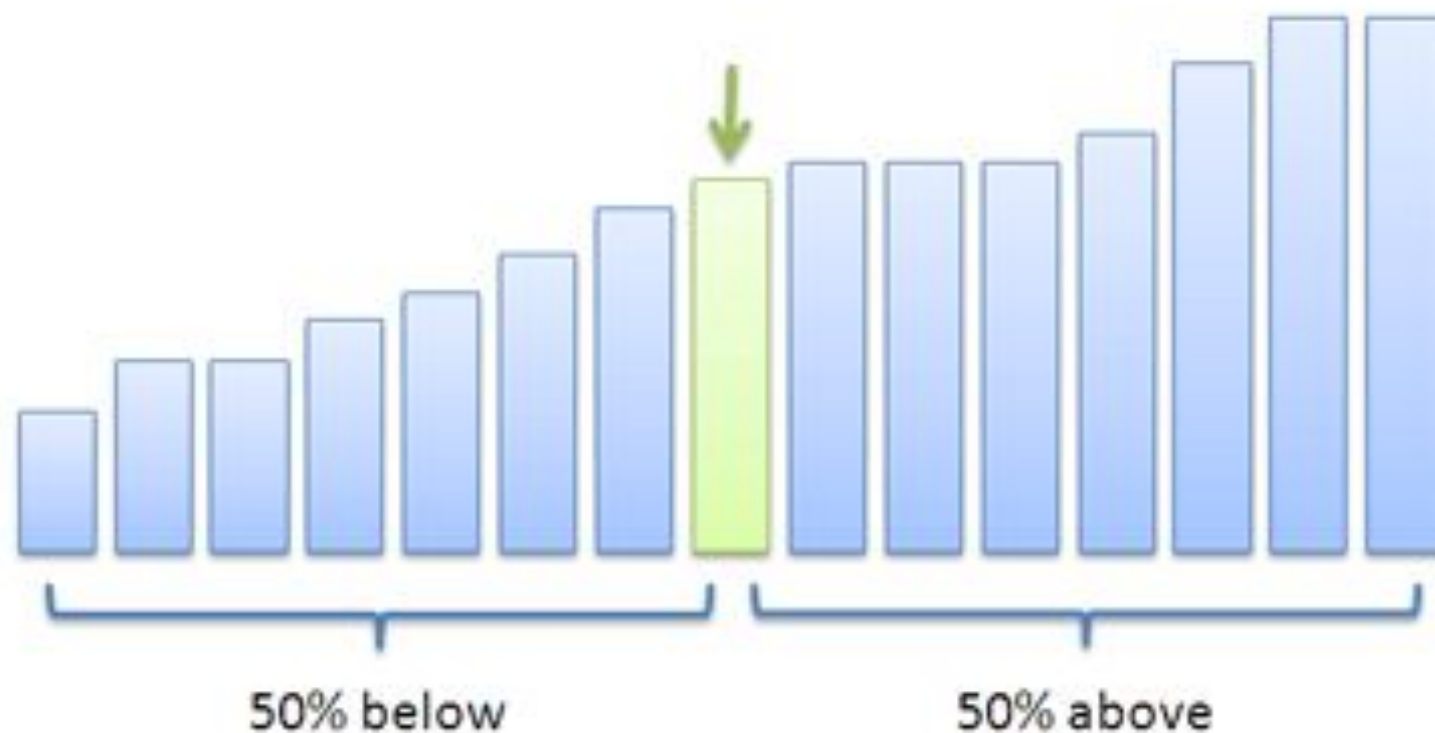
Análise do Número de Movimentações

- No **melhor caso**, temos apenas a troca do ($i == j$), logo, teremos uma troca (três movimentações) por chamada recursiva

- Funcionamento básico
- Algoritmo em C *like*
- Análise dos número de comparações e movimentações
- **Escolha do Pivô** 
- Conclusão

Mediana, o Pivô Perfeito

Median



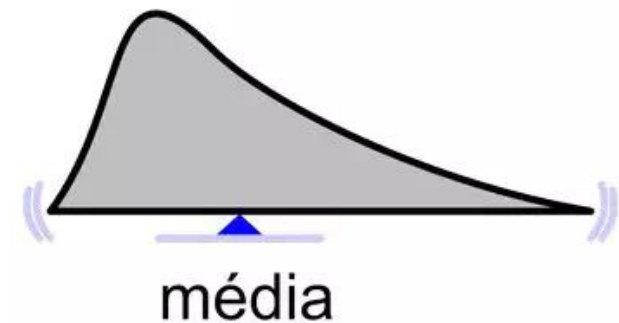
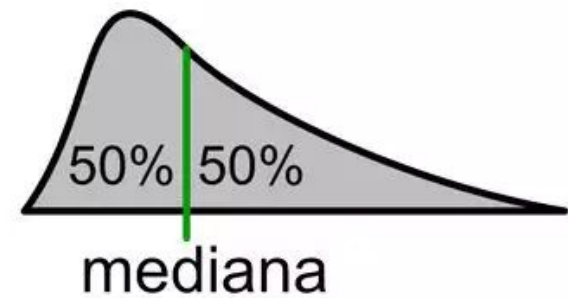
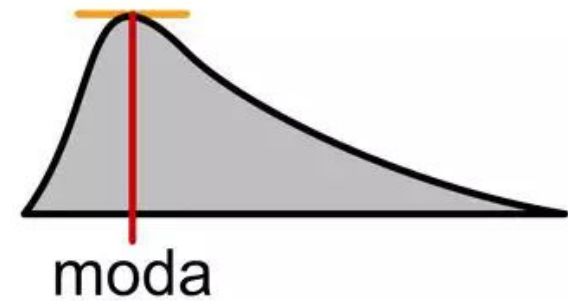
Mediana depende da Ordenação



Média como Pivô

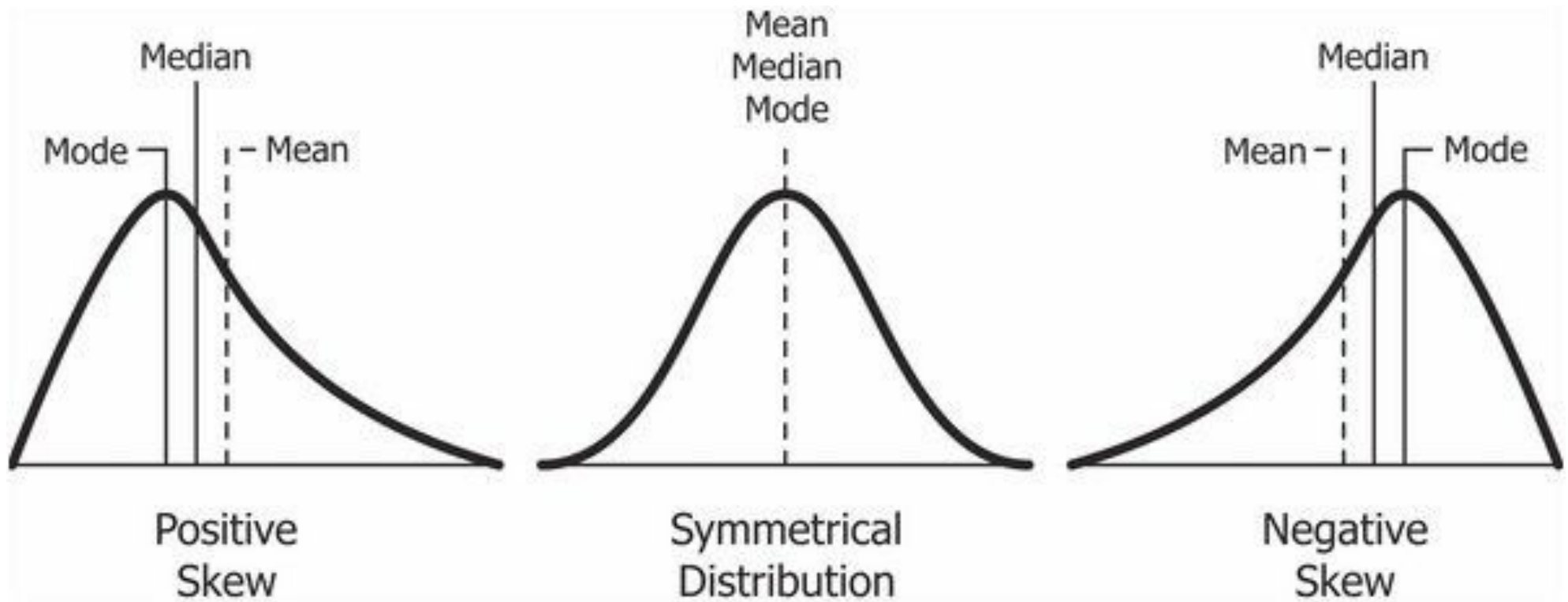
Moda: custo $\Theta(n)$

Média: custo $\Theta(n)$



Informação Privilegiada

- Quando conhecemos a distribuição dos dados, podemos utilizar a estratégia de escolha do pivô mais adequada àquela distribuição



Escolha Aleatória

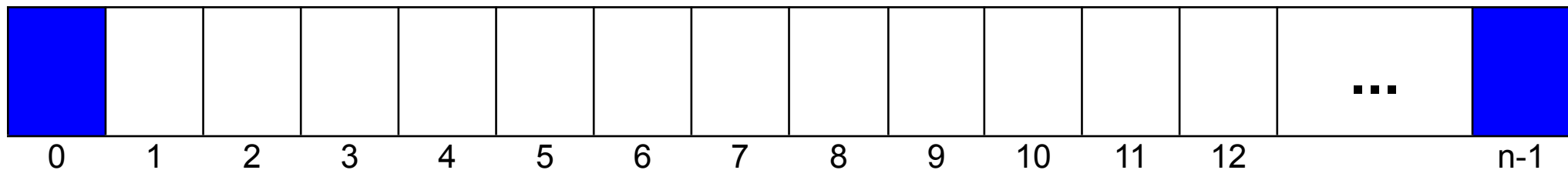
- Escolher aleatoriamente um item da lista como pivô
- Na média, teremos uma partição da lista na proporção: $\frac{1}{4}$ e $\frac{3}{4}$
- Se a partição da lista ocorrer pelo menos metade das vezes nessa proporção, o tempo de execução esperado é $\Theta(n \times \log n)$

Mediana de Três

- Escolhemos três elementos aleatoriamente
- A mediana dos três será o pivô
- Esta estratégia aumenta ainda mais as chances de se obter o caso médio $\Theta(n \times \log n)$
- Como existe um custo para se obter três elementos aleatórios e obter a mediana, essa estratégia é utilizada apenas em listas “grandes”. Em listas menores, a escolha aleatória simples é mais adequada

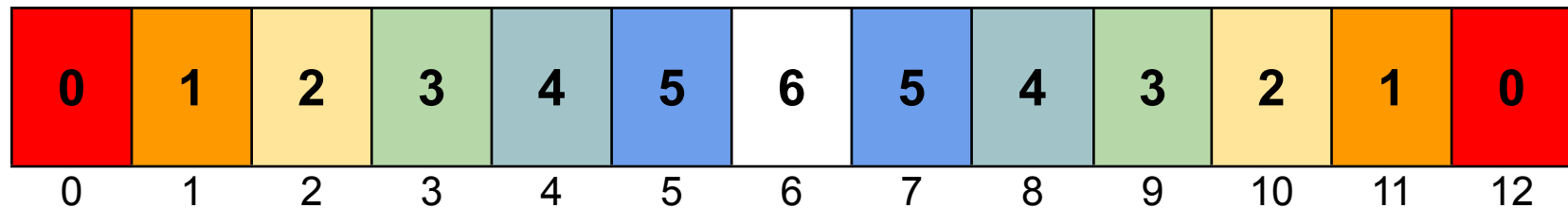
Primeiro ou Último Elemento


- É uma técnica simples, contudo, se os elementos estiverem em ordem crescente ou decrescente teremos o **pior caso** do Quicksort



Elemento do Meio

- É uma técnica simples em que o **pior caso** do Quicksort acontece se, sistematicamente, os elementos formarem uma espécie de triângulo



- Funcionamento básico
- Algoritmo em C *like*
- Análise dos número de comparações e movimentações
- Escolha do Pivô
- **Conclusão** 

Conclusão

- A razão de sua velocidade é a simplicidade do seu anel interno
- Vantagens:
 - Extremamente eficiente
 - Necessita de apenas uma pequena pilha como memória auxiliar
 - Faz em média $\Theta(n \times \lg(n))$ comparações

- Desvantagens:
 - Seu pior caso para comparações é quadrático
 - Sua implementação é delicada e difícil
 - Método não estável

Exercício (1)

- Mostre todas as comparações e movimentações do algoritmo anterior para o *array* abaixo:

12	4	8	2	14	17	6	18	10	16	15	5	13	9	1	11	7	3
----	---	---	---	----	----	---	----	----	----	----	---	----	---	---	----	---	---