Aula anterior

- Métodos de Ordenação Simples
  - Ordenação Método Bolha (Bublle-Sort)
  - Ordenação Método Seleção (Selection-Sort)
  - Ordenação Método de Inserção (insertion-Sort
  - Ordenação por contagem (Couting-Sort)
  - Ordenação por radicais (Radix-Sort)

Métodos de Ordenação Eficentes

Prof. Diego Silva Caldeira Rocha

Objetivos

Métodos de Ordenação Eficientes:

- Método Shell-Sort
- Método Quick-Sort
- Método Merge-Sort
- Método Heap-Sort

Conclusão dos métodos de Ordenação

Introdução

- Proposto por Shell em 1959 como uma extensão da Inserção

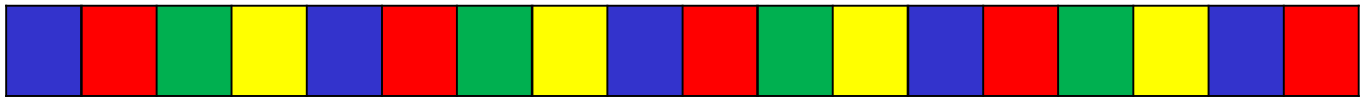
- O problema da Inserção é efetuamos *(n – 1)* comparações e movimentações quando o menor elemento está à direita

- O método de Shell contorna esse problema permitindo trocas de registros distantes um do outro (passo h)

Funcionamento Básico

- Os elementos separados de h posições são ordenados via Inserção

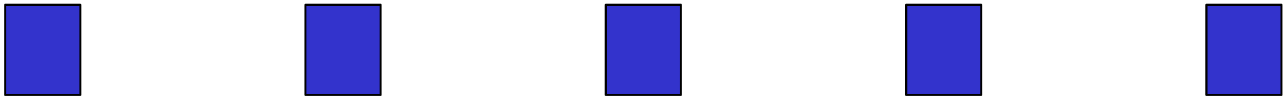
- Por exemplo, se h = 4, ordenamos com a Inserção cada um dos pseudo *array*



Funcionamento Básico

- Os elementos separados de h posições são ordenados via Inserção

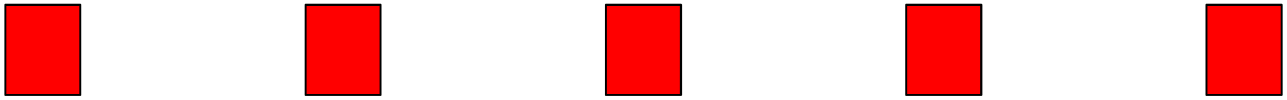
- Por exemplo, se h = 4, ordenamos com a Inserção cada um dos pseudo *array*



Funcionamento Básico

- Os elementos separados de h posições são ordenados via Inserção

- Por exemplo, se h = 4, ordenamos com a Inserção cada um dos pseudo *array*



Funcionamento Básico

- Os elementos separados de h posições são ordenados via Inserção

- Por exemplo, se h = 4, ordenamos com a Inserção cada um dos pseudo *array*

Funcionamento Básico

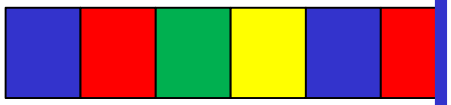
- Os elementos separados de h posições são ordenados via Inserção

- Por exemplo, se h = 4, ordenamos com a Inserção cada um dos pseudo *array*

Funcionamento Básico

- Os elementos separados de h posições são ordenados via Inserção

- Por exemplo, se h = 4, ordenamos com a Inserção cada um dos pseudo *array*



Pode parecer loucura, mas repetiremos a Inserção para cada pseudo *array*

Funcionamento Básico

- Neste momento, nossa sequência é dita h-ordenada (para nosso exemplo 4-ordenada)

- Em seguida, reduzimos o valor de h e repetimos o processo. Isso até que o valor de h seja um e efetuamos a Inserção pela última vez

Funcionamento Básico

- Neste momento, nossa sequência é dita h-ordenada (para nosso exemplo 4-ordenada)

- Em seguida, reduzimos o valor de h e repetimos o processo. Isso até que o valor de h seja um e efetuamos a Inserção pela última vez

Traduzindo: Nós já fizemos quatro vezes a Inserção e vamos fazer mais...

Funcionamento Básico

- Neste momento, nossa sequência é dita h-ordenada (para nosso exemplo 4-ordenada)

- Em seguida, reduzimos o valor de h e repetimos o processo. Isso até que o valor de h seja um e efetuamos a Inserção pela última vez

... você acredita que isso ainda é bom!!! Fala sério!

Algoritmo

```
public static void shellSort ( int [ ] v )
{
  int i , j , h = 1, value ;
    do {
      h = 3 * h + 1;
    } while ( h < v.length );
    do {
      h = h / 3;
      for ( i = h; i < v.length; i++) {
        value = v [ i ];
        j = i - h;
        while (j >= 0 && value < v [ j ]) {
          v [ j + h ] = v [ j ];
          j = j - h;
        }
        v [ j + h ] = value;
      }
  } while ( h > 1 )

}
```

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

90	45	32	11	15	60	45	70	20	13	40	39	25	10

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

90	45	32	11	15	60	45	70	20	13	40	39	25	5

Como h = 4, define-se quatro pseudo *arrays* com as cores azul, vermelha, verde e amarela

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

90	45	32	11	15	60	45	70	20	13	40	39	25	10

Fazendo o algoritmo de inserção nos azuis

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	45	32	11	20	60	45	70	25	13	40	39	90	10

Fazendo o algoritmo de inserção nos azuis

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	45	32	11	20	60	45	70	25	13	40	39	90	10

Fazendo o algoritmo de inserção nos vermelhos

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15 **45** 32 11 20 **60** 45 70 25 **13** 40 39 90 **10**

Fazendo o algoritmo de inserção nos vermelhos

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15 **10** 32 11 20 **13** 45 70 25 **45** 40 39 90 **60**

Fazendo o algoritmo de inserção nos vermelhos

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	10	32	11	20	13	45	70	25	45	40	39	90	60

Fazendo o algoritmo de inserção nos verdes

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	10	32	11	20	13	45	70	25	45	40	39	90	60

Fazendo o algoritmo de inserção nos verdes

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	10	32	11	20	13	40	70	25	45	45	39	90	60

Fazendo o algoritmo de inserção nos verdes

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15 10 32 **11** 20 13 40 **70** 25 45 45 **39** 90 60

Fazendo o algoritmo de inserção nos amarelos

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	10	32	11	20	13	40	39	25	45	45	70	90	60

Fazendo o algoritmo de inserção nos amarelos

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	10	32	11	20	13	40	39	25	45	45	70	90	60

Com h = 4, o *array* não está ordenado, mas os números estão mais próximos de suas posições

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	10	32	11	20	13	40	39	25	45	45	70	90	60

Fazendo h = 2, define-se dois pseudo *arrays* com as cores azul e vermelha

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	10	32	11	20	13	40	39	25	45	45	70	90	60

Fazendo o algoritmo de inserção nos azuis

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	10	32	11	20	13	40	39	25	45	45	70	90	60

Fazendo o algoritmo de inserção nos azuis

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	10	20	11	25	13	32	39	40	45	45	70	90	60

Fazendo o algoritmo de inserção nos azuis

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	10	32	11	20	13	40	39	25	45	45	70	90	60

Com h = 4, o *array* não está ordenado, mas os números estão mais próximos de suas posições

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	10	20	11	25	13	32	39	40	45	45	70	90	60

Fazendo o algoritmo de inserção nos vermelhos

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15 **10** 20 **11** 25 **13** 32 **39** 40 **45** 45 **60** 90 **70**

Fazendo o algoritmo de inserção nos vermelhos

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	10	20	11	25	13	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

15	10	20	11	25	13	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	15	20	11	25	13	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	15	20	11	25	13	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	15	11	20	25	13	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	11	15	20	25	13	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	11	15	20	25	13	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	11	15	20	13	25	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	11	15	13	20	25	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

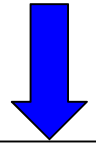
- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	11	13	15	20	25	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

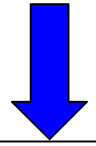


10	11	13	15	20	25	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1



10	11	13	15	20	25	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	11	13	15	20	25	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	11	13	15	20	25	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	11	13	15	20	25	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	11	13	15	20	25	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	11	13	15	20	25	32	39	40	45	45	60	90	70

Fazendo h = 1, ou seja, a inserção original

Exemplo

- Ordene o *array* abaixo com o Shellsort, fazendo h = 4, 2 e 1

10	11	13	15	20	25	32	39	40	45	45	60	70	90

Fazendo h = 1, ou seja, a inserção original

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] < pivô
  - Percorra o *array* a partir da direita enquanto *array*[j] > pivô
  - Se i ≤ j então troque *array*[i] com *array*[j]
  - Continue o processo enquanto i ≤ 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 ≤ que pivô
  - Os elementos *array*[i], *array*[i+1], . . . , *array*[dir] são ≥ que pivô

Algoritmo

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

```
public void swap(int [] array ,int i, int j)
{
    int temp = array[i];
    array[i] = array[j];
    array[j] = temp;
}
```

Exemplo

```
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

Exemplo

```
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

Exemplo

```
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

Exemplo

```
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

Exemplo

```
void quicksort(int esq, int dir) {
        int i = esq,                    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

Exemplo

```
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 + 15) / 2: 7

pivô | 10

	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**

Exemplo

```
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);
}
```

pivô | 10

0 <= 15: 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

Exemplo

```
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);
}
```

pivô | 10

0 < 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

Exemplo

```
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);
}
```

pivô | 10

	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

Exemplo

```
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);
}
```

pivô | 10

1 < 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**

Exemplo

```
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);
}
```

pivô | 10

		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

Exemplo

```
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);
}
```

pivô | 10

5 < 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**

Exemplo

```
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);
}
```

pivô | 10

			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

Exemplo

```
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);
}
```

pivô | 10

3 < 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**

Exemplo

```
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);
}
```

pivô | 10

				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

Exemplo

```
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);
}
```

pivô | 10

15 < 10: false

				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**

Exemplo

```
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);
}
```

pivô | 10

82 > 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

Exemplo

```
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);
}
```

pivô | 10

				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

Exemplo

```
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);
}
```

pivô | 10

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**

Exemplo

```
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);
}
```

pivô | 10

				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

Exemplo

```
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);
}
```

pivô | 10

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**

Exemplo

```
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);
}
```

pivô | 10

				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

Exemplo

```
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);
}
```

pivô | 10

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**

Exemplo

```
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);
}
```

pivô | 10

				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

Exemplo

```
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);
}
```

pivô | 10

5 > 10: false

				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**

Exemplo

```
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);
}
```

pivô | 10

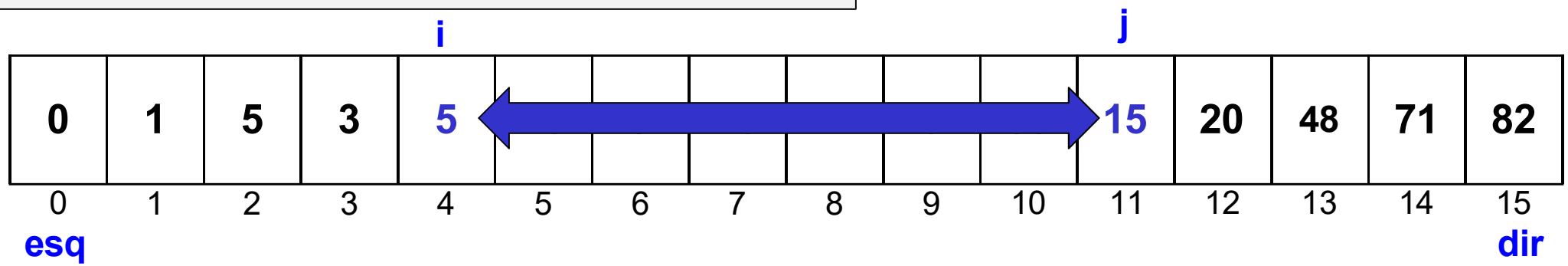
4 <= 11: true

				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**

Exemplo

```
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);
}
```

pivô | 10



Exemplo

```
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);
}
```

pivô | 10

					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

Exemplo

```
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);
}
```

pivô 10

5 <= 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**

Exemplo

```
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);
}
```

pivô | 10

16 < 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**

Exemplo

```
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);
}
```

pivô | 10

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**

Exemplo

```
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);
}
```

pivô | 10

					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

Exemplo

```
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);
}
```

pivô | 10

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**

Exemplo

```
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);
}
```

pivô | 10

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**

Exemplo

```
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);
}
```

pivô | 10

					i				j						
0	1	5	3	5	3	↔			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

Exemplo

```
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);
}
```

pivô | 10

						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

Exemplo

```
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);
}
```

pivô | 10

6 <= 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**

Exemplo

```
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);
}
```

pivô 10

9 < 10: 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**

Exemplo

```
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);
}
```

pivô | 10

							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

Exemplo

```
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);
}
```

pivô | 10

10 < 10: false

							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**

Exemplo

```
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);
}
```

pivô | 10

4 > 10: false

							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**

Exemplo

```
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);
}
```

pivô | 10

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**

Exemplo

```
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);
}
```

pivô | 10

							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

Exemplo

```
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);
}
```

pivô | 10

							j	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

Exemplo

```
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);
}
```

pivô | 10

8 <= 7: false

							j	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**

Exemplo

```
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);
}
```

pivô | 10

0 < 7: true

							j	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**

Exemplo

```
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);
}
```

pivô | 10

							j	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

Exemplo

```
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);
}
```

pivô ?

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								

Exemplo

```
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

Exemplo

```
void quicksort(int esq, int dir) {
        int i = esq,                    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								

Exemplo

```
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

	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**								

Exemplo

```
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);
}
```

pivô | 3

0 <= 7: 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**								

Exemplo

```
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);
}
```

pivô | 3

0 < 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**								

Exemplo

```
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);
}
```

pivô | 3

	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								

Exemplo

```
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);
}
```

pivô | 3

1 < 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**								

Exemplo

```
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);
}
```

pivô | 3

		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								

Exemplo

```
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);
}
```

pivô | 3

5 < 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**								

Exemplo

```
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);
}
```

pivô | 3

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**								

Exemplo

```
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);
}
```

pivô | 3

		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								

Exemplo

```
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);
}
```

pivô | 3

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**								

Exemplo

```
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);
}
```

pivô | 3

		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								

Exemplo

```
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);
}
```

pivô | 3

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**								

Exemplo

```
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);
}
```

pivô | 3

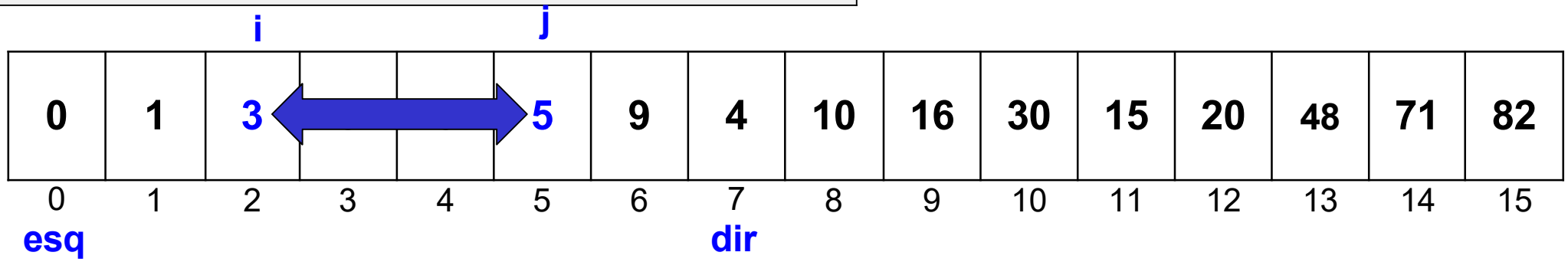
2 <= 5: true

		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**								

Exemplo

```
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);
}
```

pivô 3



Exemplo

```
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);
}
```

pivô | 3

			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								

Exemplo

```
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);
}
```

pivô | 3

3 <= 4: 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								

Exemplo

```
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);
}
```

pivô | 3

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								

Exemplo

```
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);
}
```

pivô | 3

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								

Exemplo

```
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);
}
```

pivô | 3

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								

Exemplo

```
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);
}
```

pivô | 3

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								

Exemplo

```
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);
}
```

pivô | 3

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**								

Exemplo

```
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);
}
```

pivô 3

			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								

Exemplo

```
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);
}
```

pivô | 3

Obs: O elemento [3] está na posição correta

		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								

Exemplo

```
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);
}
```

pivô | 3

4 <= 2: false

		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								

Exemplo

```
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);
}
```

pivô | 3

0 < 2: true

		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								

Exemplo

```
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);
}
```

pivô | 3

		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								

Exemplo

```
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);
}
```

pivô | 10

Voltando para a primeira chamada do Quicksort

8 < 15: true

							j	i							
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**

Exemplo

```
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);
}
```

pivô | 10

Voltando para a primeira chamada do Quicksort

							j	i							
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

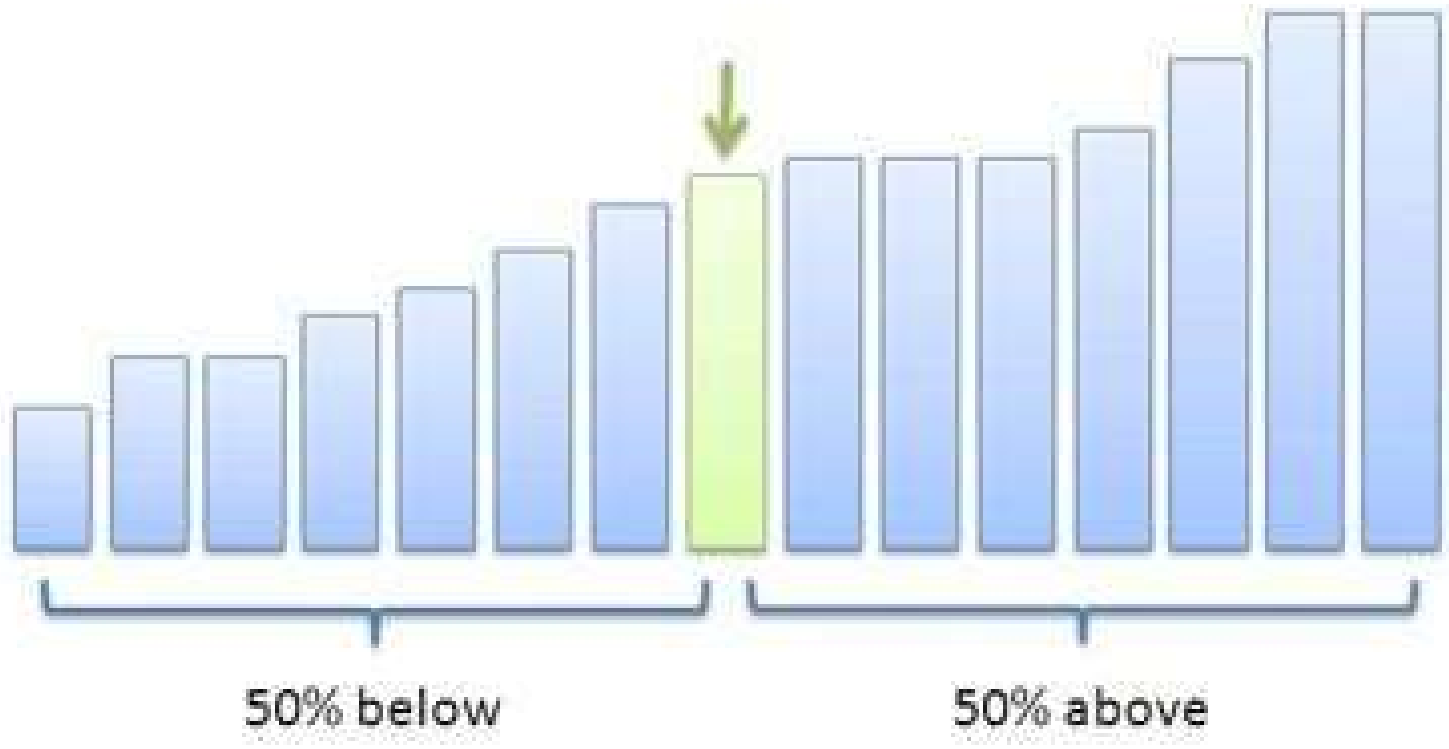
Análise do Número de Comparações

- Melhor caso: Sistematicamente, cada partição divide o array em duas partes iguais.

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

- 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)

Mediana, o Pivô Perfeito



Análise do Número de Comparações

- Pior Caso: Sistematicamente, o pivô é o menor ou o maior elemento do array, eliminando um elemento em cada chamada do algoritmo.

  - $O(n^2)$

- 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

- Pior Caso: também acontece na escolha do pivô como elemento do meio e os elementos do vetor formarem uma espécie de triângulo.

- $O(n^2)$

0	1	2	3	4	5	6	5	4	3	2	1	0
0	1	2	3	4	5	6	7	8	9	10	11	12

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

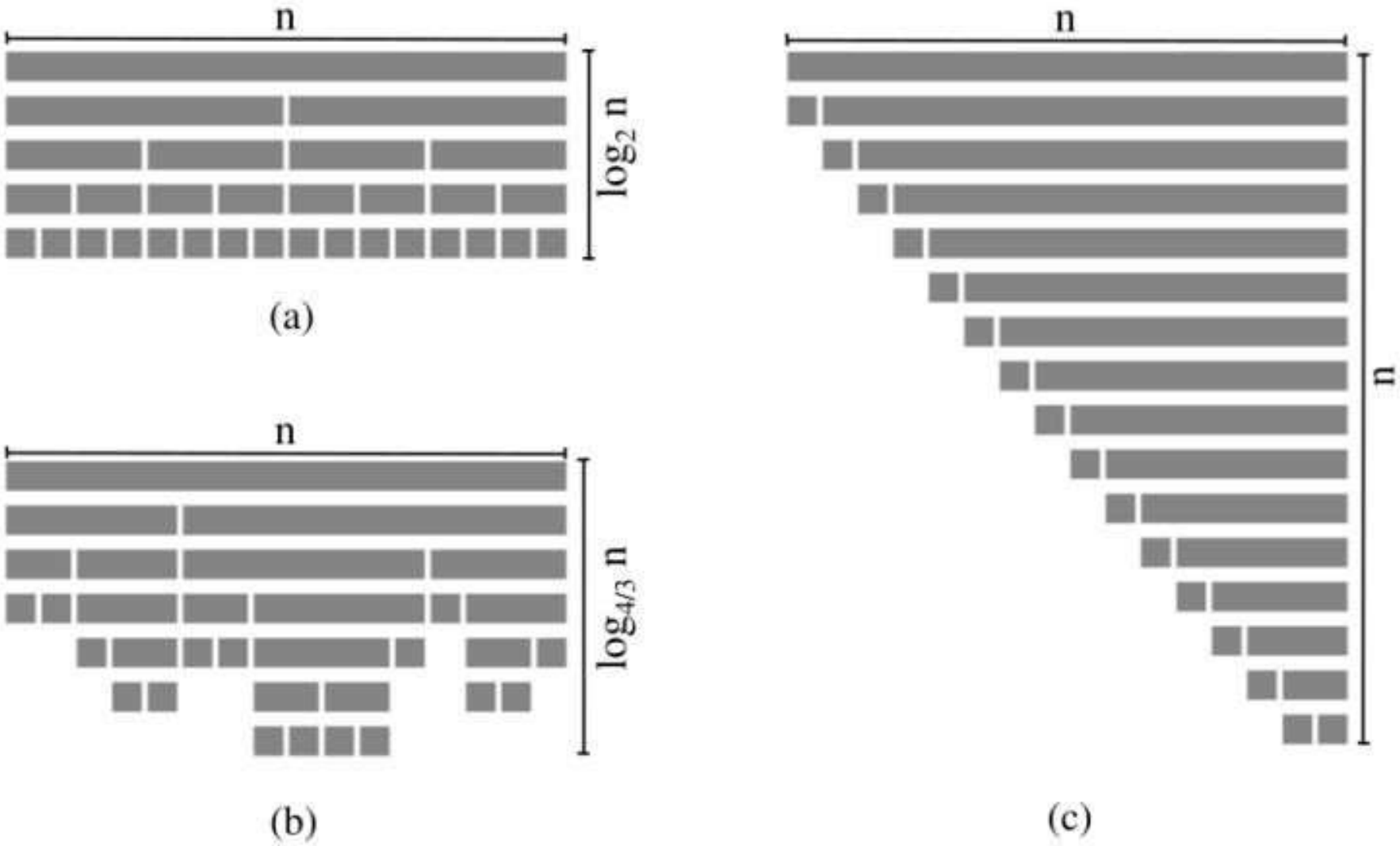


Figura: (a) Melhor caso. (b) Caso médio. (c) Pior caso.

Introdução

- Ordenação por intercalação

- Algoritmo de ordenação do tipo dividir para conquistar

- Normalmente, implementado de forma recursiva e demandando um espaço adicional de memória (não é um algoritmo *in-place*)

Um **algoritmo** de ordenação é ***in-place*** se a memória adicional requerida é independente do tamanho do *array*

Funcionamento Básico

- Dividir sistematicamente o *array* em *subarrays* até que os mesmos tenham tamanho um

- Conquistar através da intercalação (ordenada) sistemática de dois em dois *subarrays*

Algoritmo

```
// Método principal que inicia o algoritmo de ordenação
  public static void mergeSort(int[] array) {
    if (array == null) {
      return;
    }
    if (array.length > 1) {
      int[] left = leftHalf(array);
      int[] right = rightHalf(array);

      mergeSort(left);
      mergeSort(right);

      merge(array, left, right);
    }
  }

  // Método para dividir a parte esquerda do array

  public static int[] leftHalf(int[] array) {
    int size1 = array.length / 2;
    int[] left = new int[size1];
    for (int i = 0; i < size1; i++) {
      left[i] = array[i];
    }
    return left;
  }
```

```
// Método para dividir a parte direita do array
  public static int[] rightHalf(int[] array) {
    int size1 = array.length / 2;
    int size2 = array.length - size1;
    int[] right = new int[size2];
    for (int i = 0; i < size2; i++) {
      right[i] = array[i + size1];
    }
    return right;
  }

// Método para fundir as partes esquerda e direita do array ordenado
  public static void merge(int[] result, int[] left, int[] right) {
    int i1 = 0; // Índice para a parte esquerda
    int i2 = 0; // Índice para a parte direita

    for (int i = 0; i < result.length; i++) {
      if (i2 >= right.length || (i1 < left.length && left[i1] <= right[i2])) {
        result[i] = left[i1];
        i1++;
      } else {
        result[i] = right[i2];
        i2++;
      }
    }
  }
```

Exemplo

95	**15**	**42**	**4**	**23**	**16**	**8**	**77**

Dividir

Conquistar (intercalação ou *merge*)

Subarray aguardando divisão

Elemento a ser intercalado no *subarray*

Exemplo

95	15	42	4	23	16	8	77

Exemplo

95	15	42	4	23	16	8	77

95	15	42	4

23	16	8	77

Exemplo

95	15	42	4	23	16	8	77

95	15	42	4

23	16	8	77

Exemplo

95	15	42	4	23	16	8	77

95	15	42	4

23	16	8	77

95	15

42	4

Exemplo

95	15	42	4	23	16	8	77

95	15	42	4

23	16	8	77

95	15

42	4

Exemplo

95	15	42	4	23	16	8	77

95	15	42	4

23	16	8	77

95	15

42	4

95	15

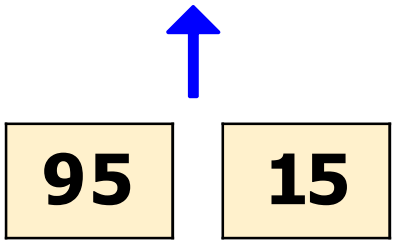
Exemplo

95	15	42	4	23	16	8	77

95	15	42	4

23	16	8	77

42	4



Exemplo

95	15	42	4	23	16	8	77

95	15	42	4

23	16	8	77

15	

42	4

95	~~15~~

Exemplo

95	15	42	4	23	16	8	77

95	15	42	4

23	16	8	77

15	95

42	4

~~95~~	~~15~~

Exemplo

95	15	42	4	23	16	8	77

95	15	42	4

23	16	8	77

15	95

42	4

Exemplo

95	15	42	4	23	16	8	77

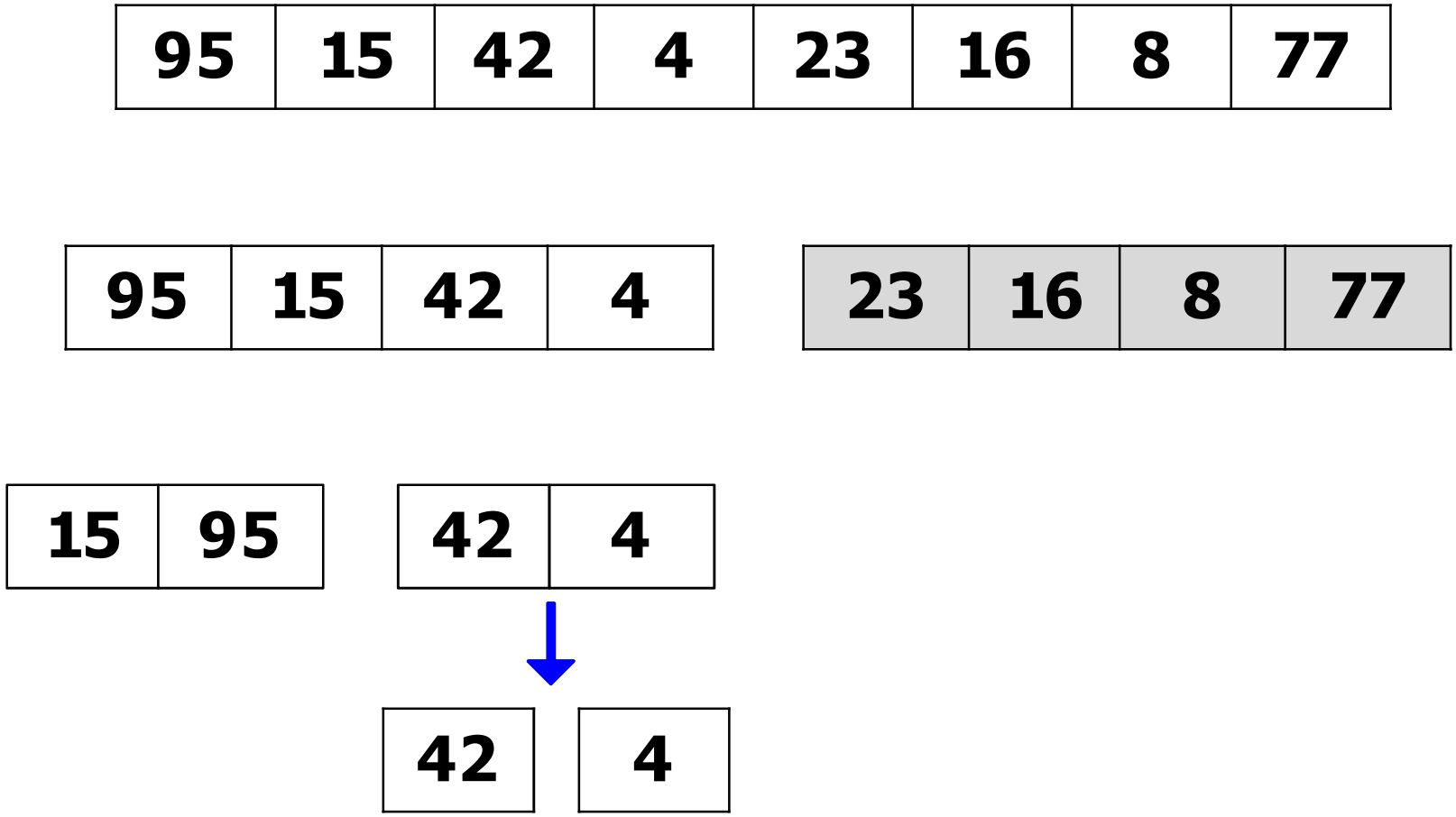
95	15	42	4

23	16	8	77

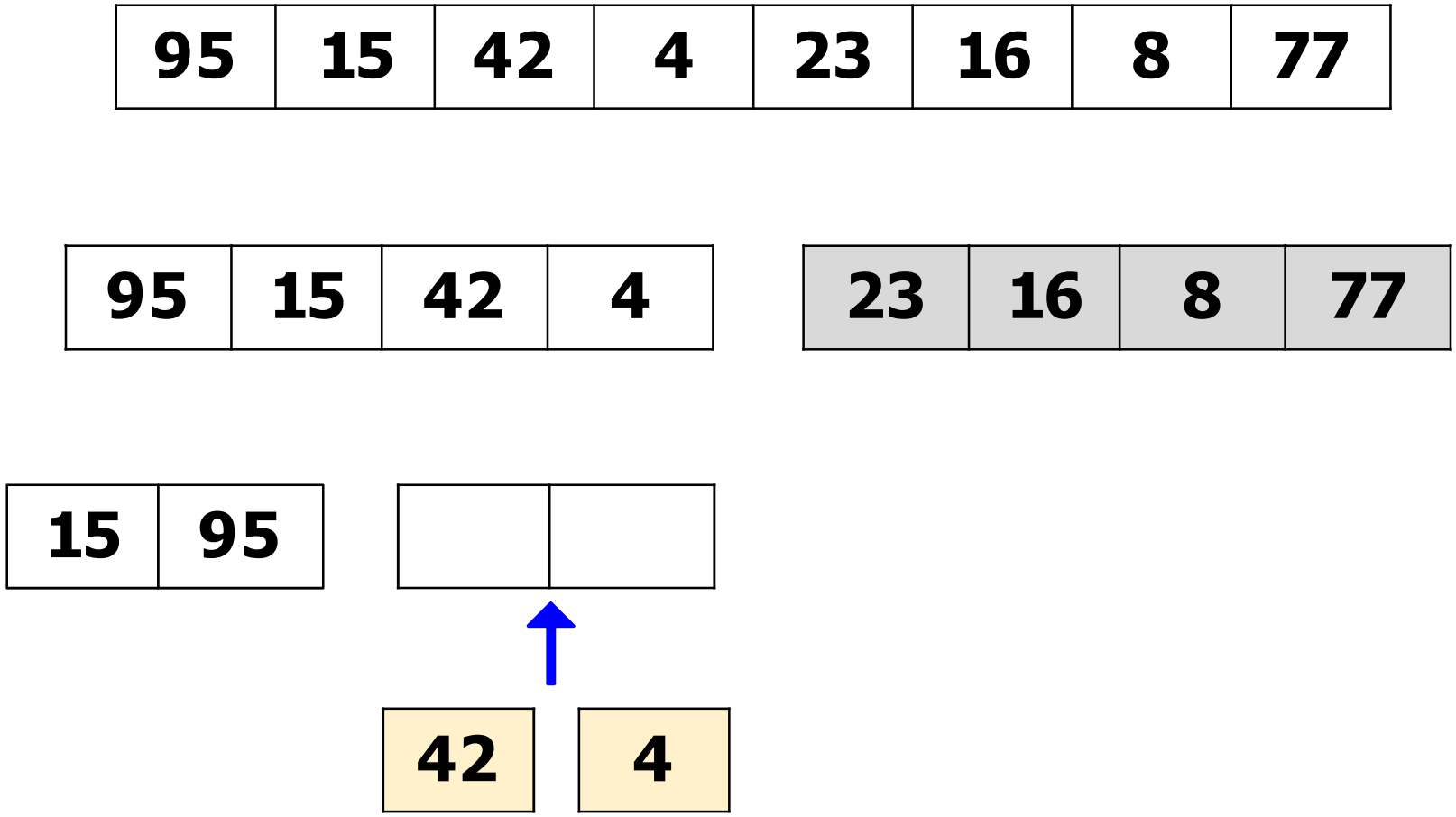
15	95

42	4

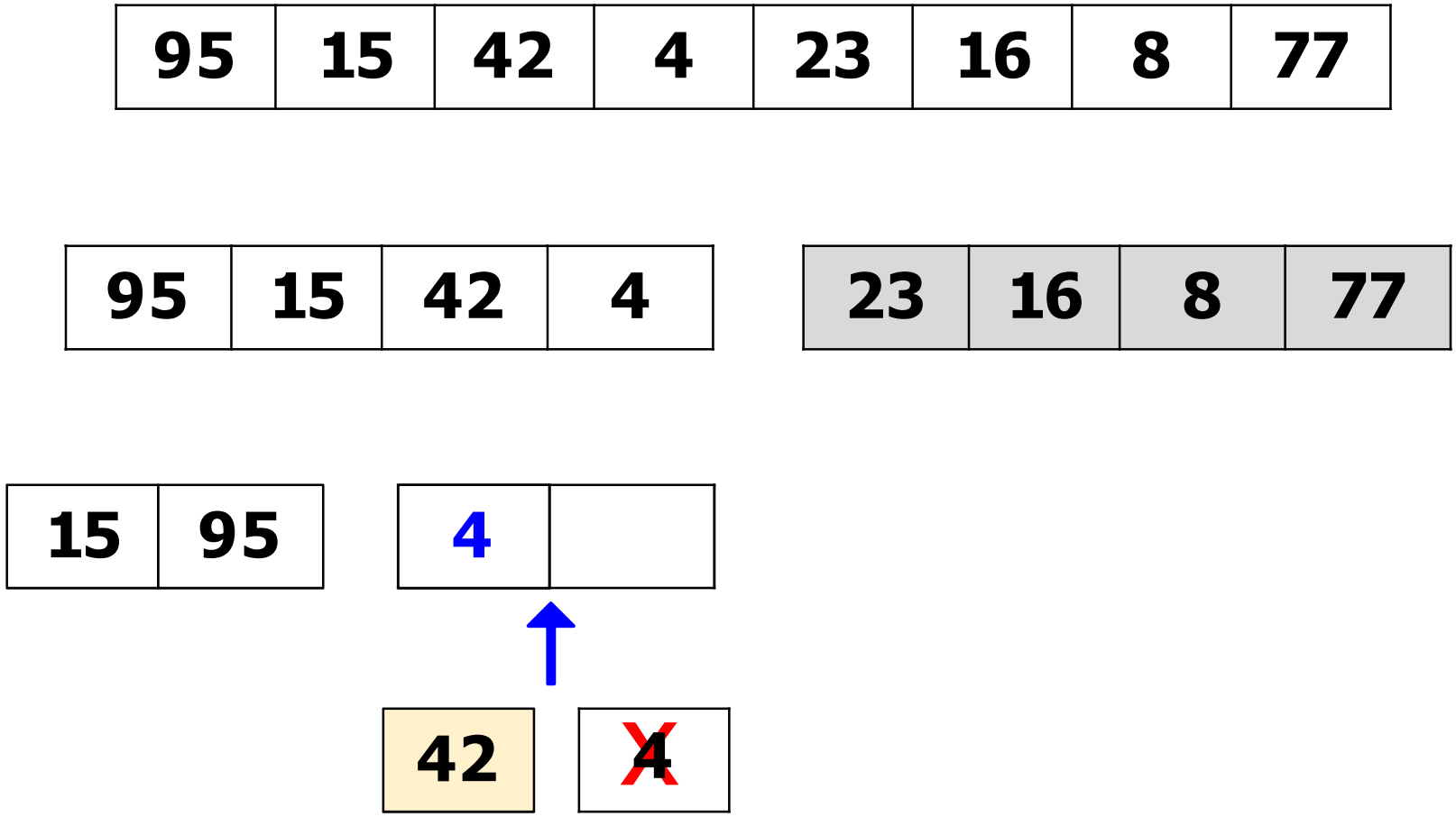
Exemplo



Exemplo



Exemplo



Exemplo

95	15	42	4	23	16	8	77

95	15	42	4

23	16	8	77

15	95

4	42

~~42~~	~~4~~

Exemplo

95	15	42	4	23	16	8	77

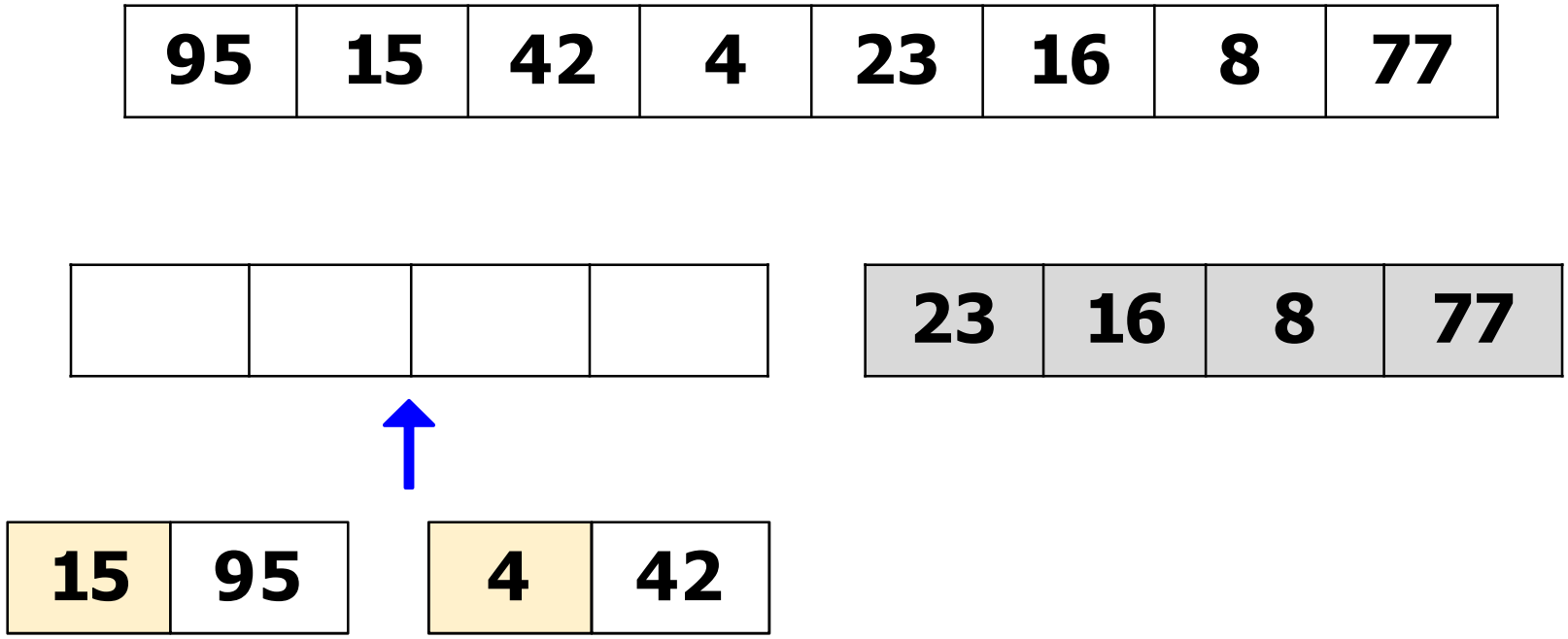
95	15	42	4

23	16	8	77

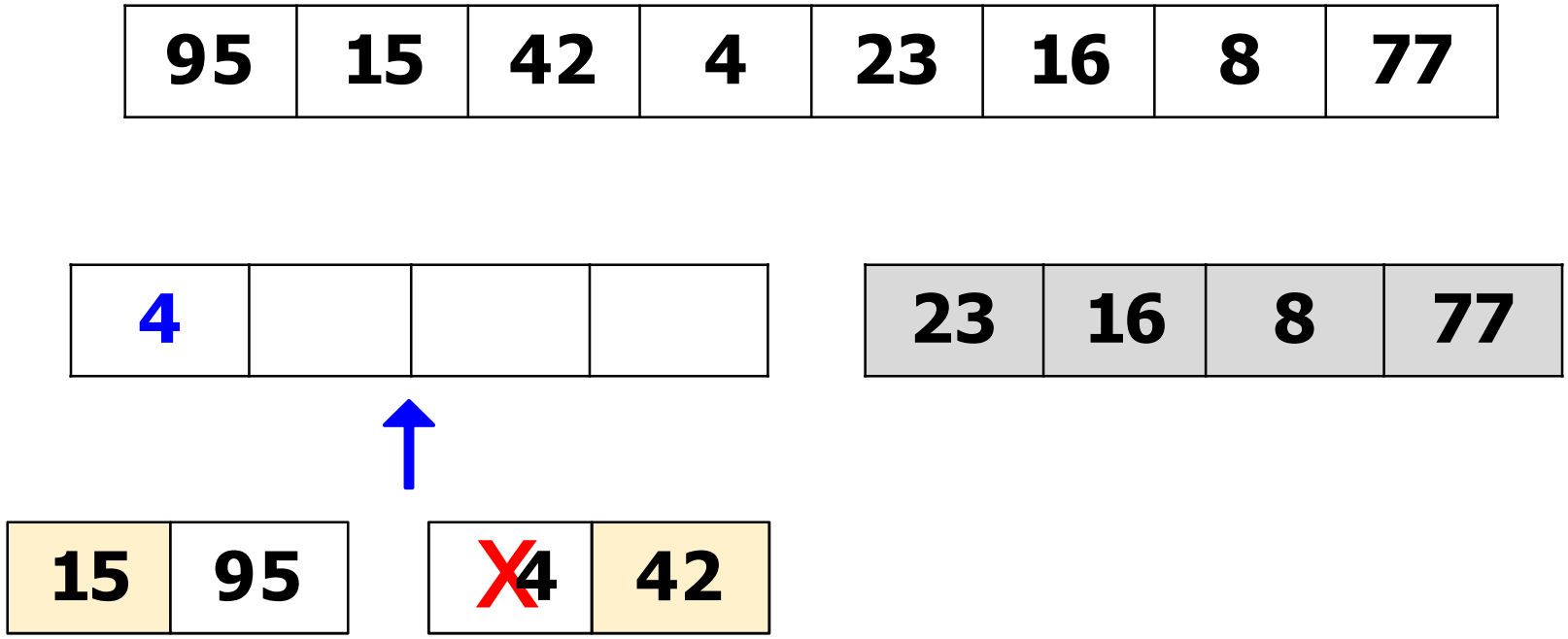
15	95

4	42

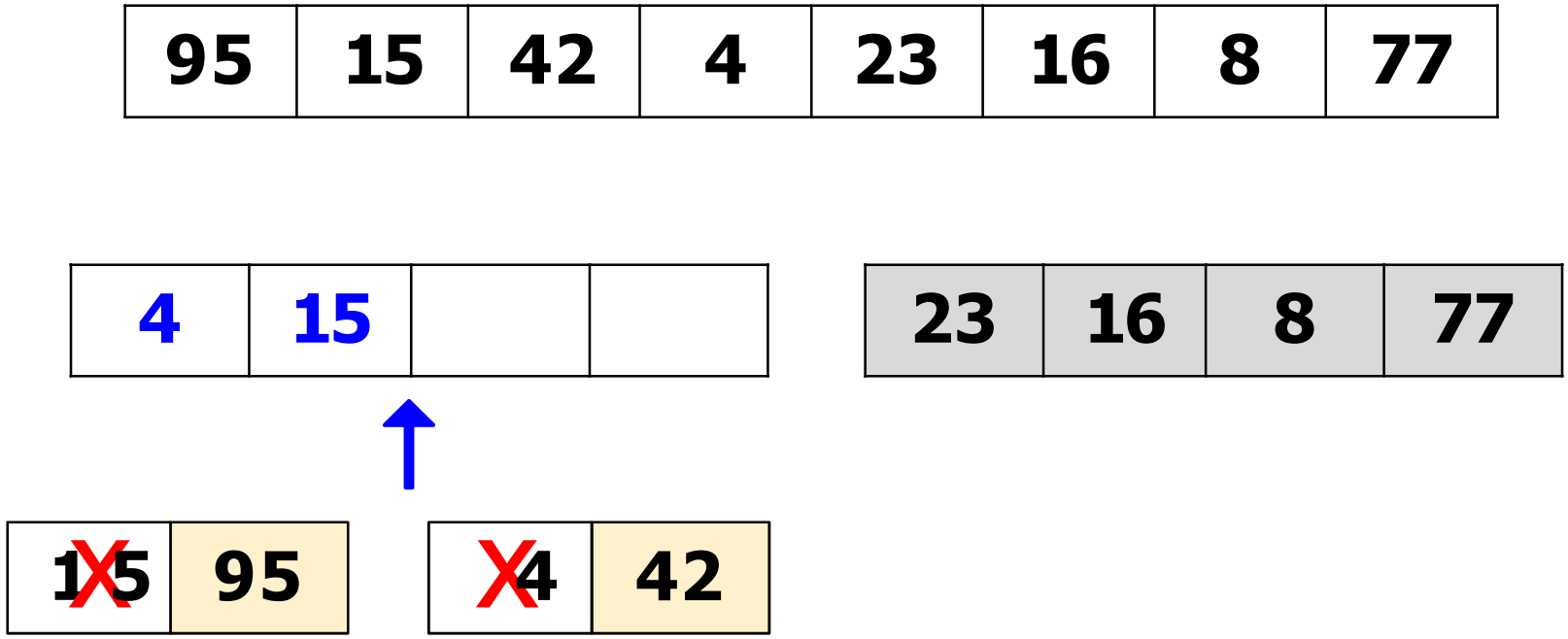
Exemplo



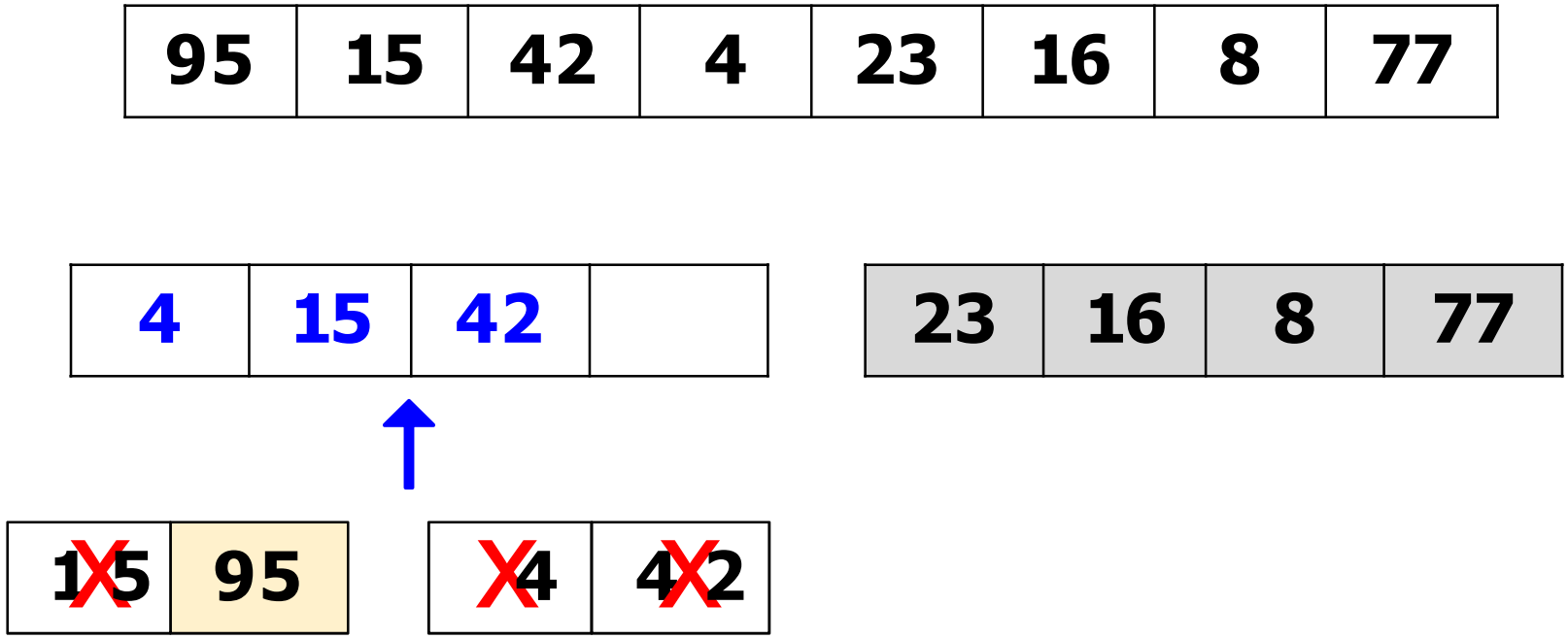
Exemplo



Exemplo



Exemplo



Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

23	16	8	77

~~1~~5	~~9~~5

~~X~~4	4~~X~~2

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

23	16	8	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

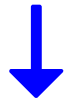
23	16	8	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

23	16	8	77



23	16

8	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

23	16	8	77

23	16

8	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

23	16	8	77

23	16

8	77

23	16

Exemplo

95	15	42	4	23	16	8	77

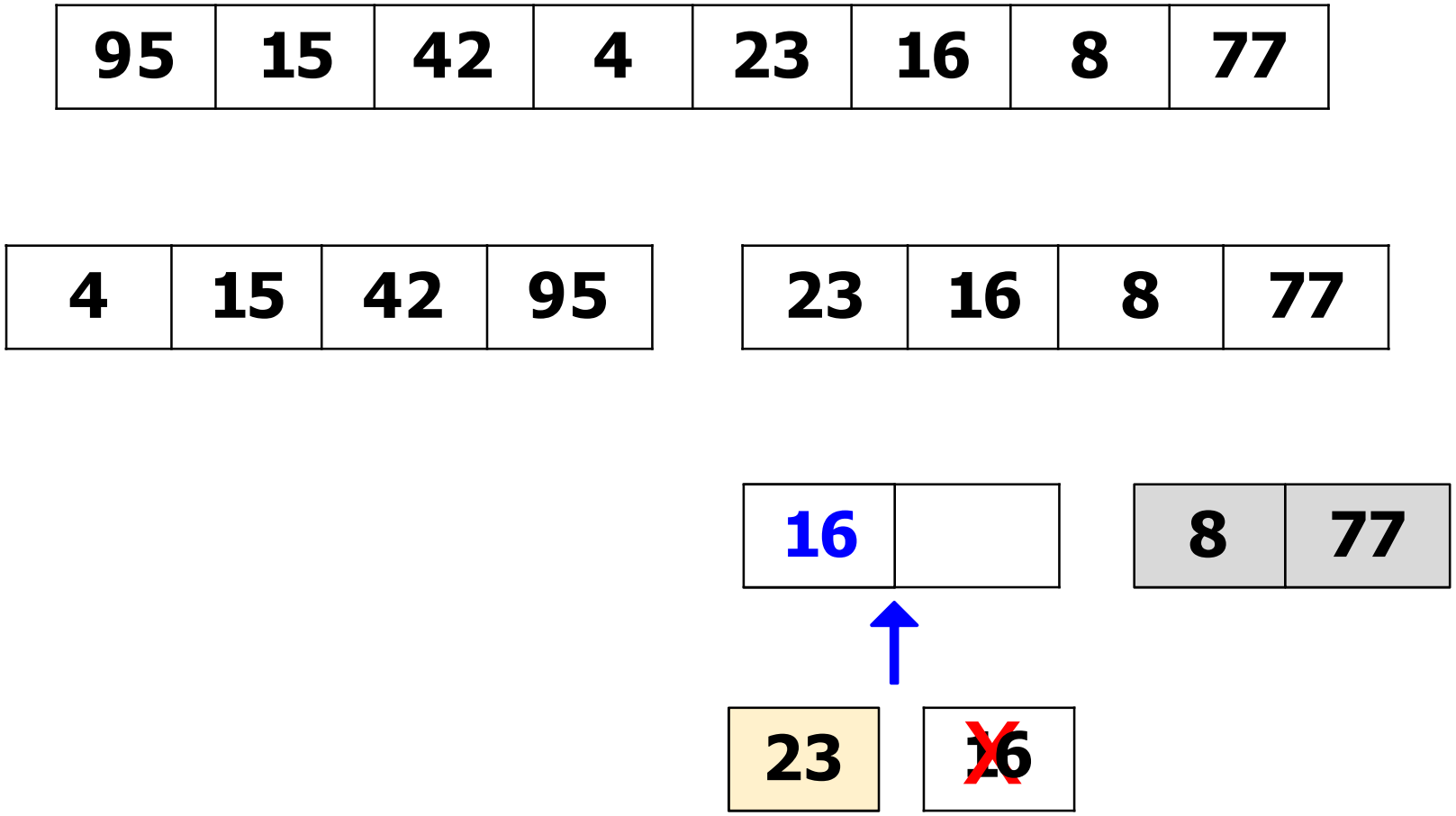
4	15	42	95

23	16	8	77

8	77

23	16

Exemplo



Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

23	16	8	77

16	23

8	77

~~23~~	~~16~~

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

23	16	8	77

16	23

8	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

23	16	8	77

16	23

8	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

23	16	8	77

16	23

8	77

↓

8	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

23	16	8	77

16	23

8	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

23	16	8	77

16	23

8	

~~8~~	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

23	16	8	77

16	23

8	77

~~8~~	~~77~~

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

23	16	8	77

16	23

8	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

16	23

8	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

8			

16	23

~~8~~	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

8	16		

~~16~~	23

~~8~~	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

8	16	23	

~~1~~6	~~2~~3

~~X~~8	77

Exemplo

95	15	42	4	23	16	8	77

4	15	42	95

8	16	23	77

~~1~~6	~~2~~3

~~X~~8	~~7~~7

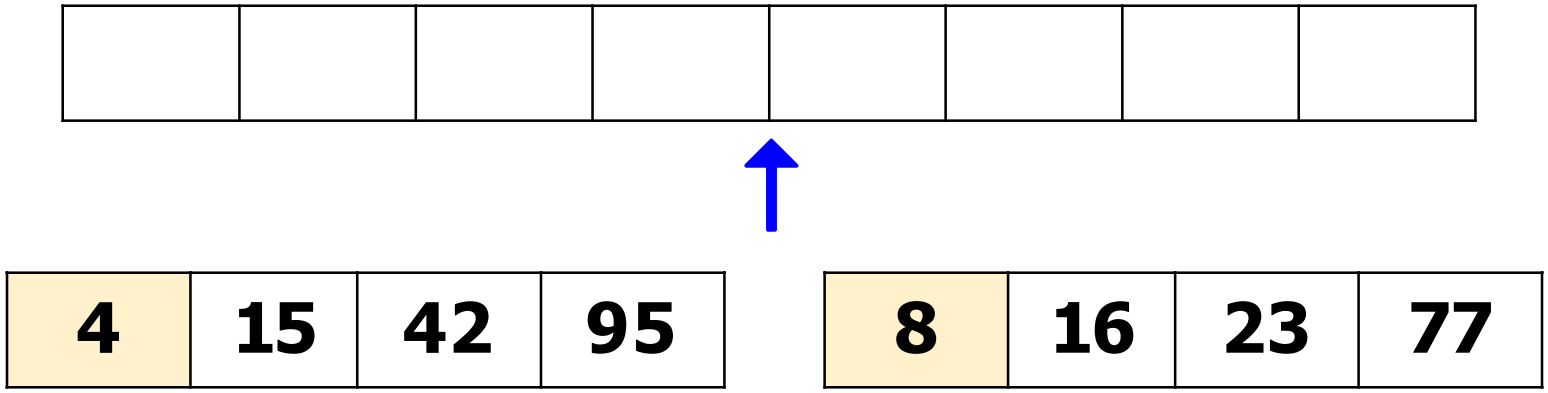
Exemplo

95	15	42	4	23	16	8	77

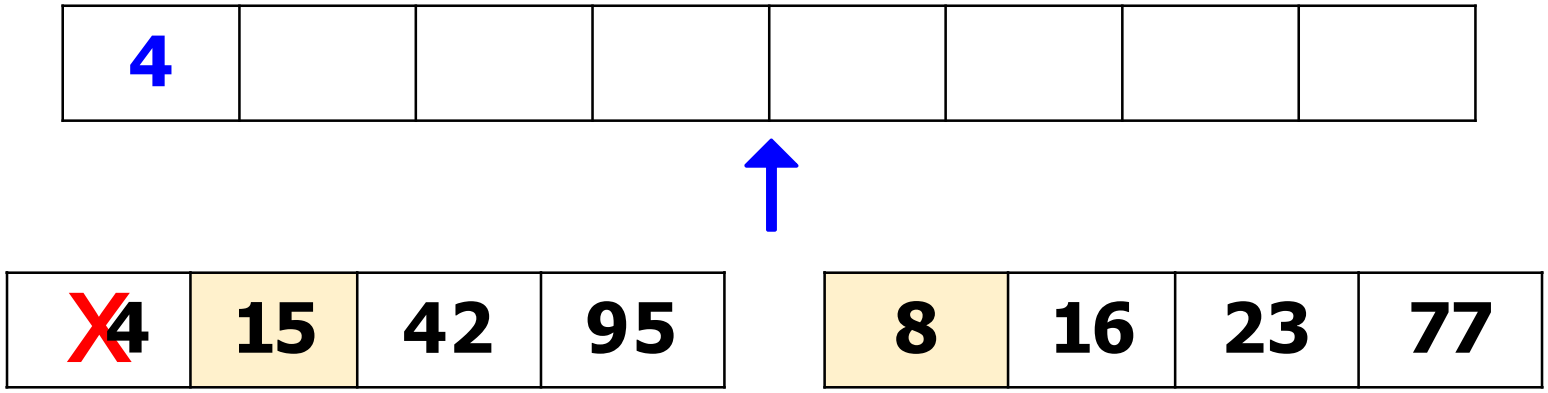
4	15	42	95

8	16	23	77

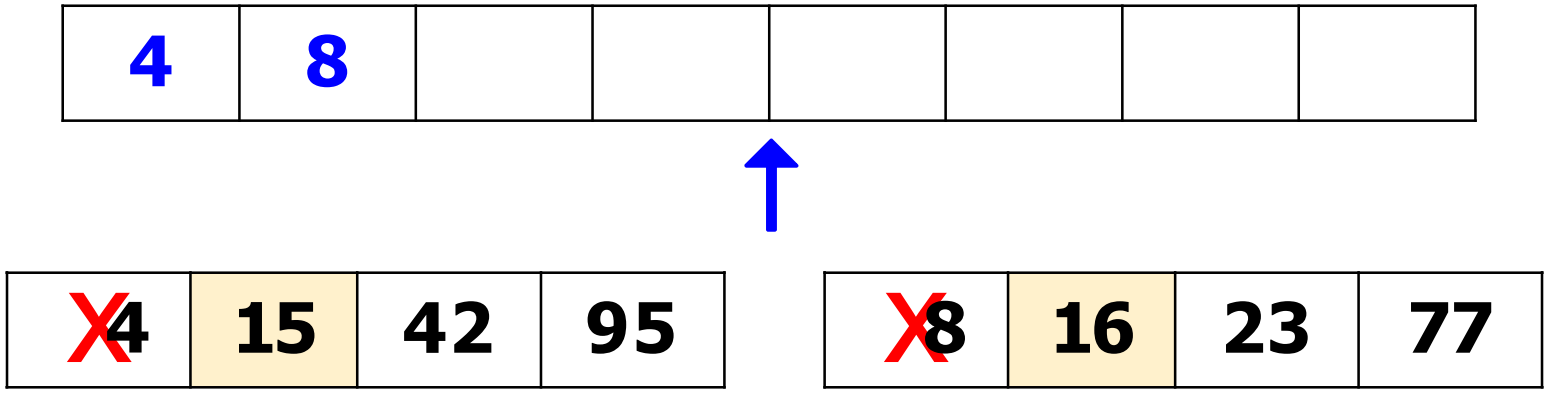
Exemplo



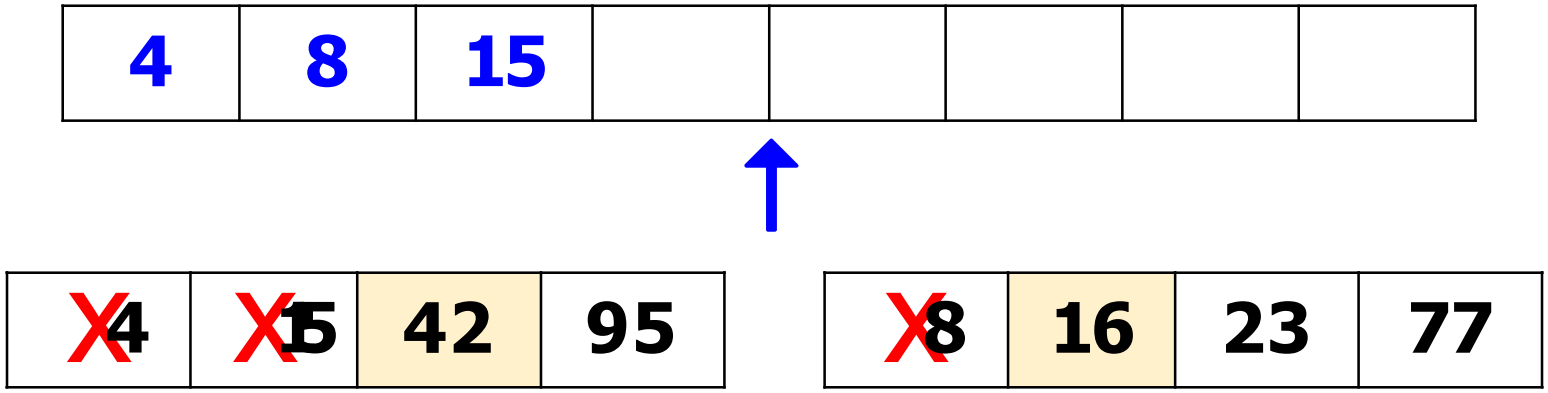
Exemplo



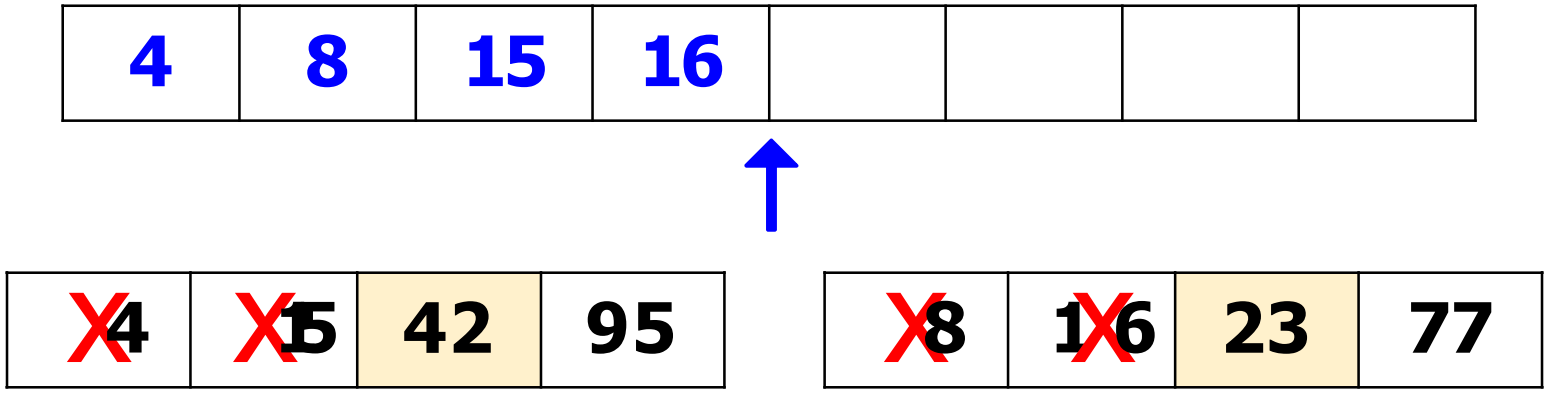
Exemplo



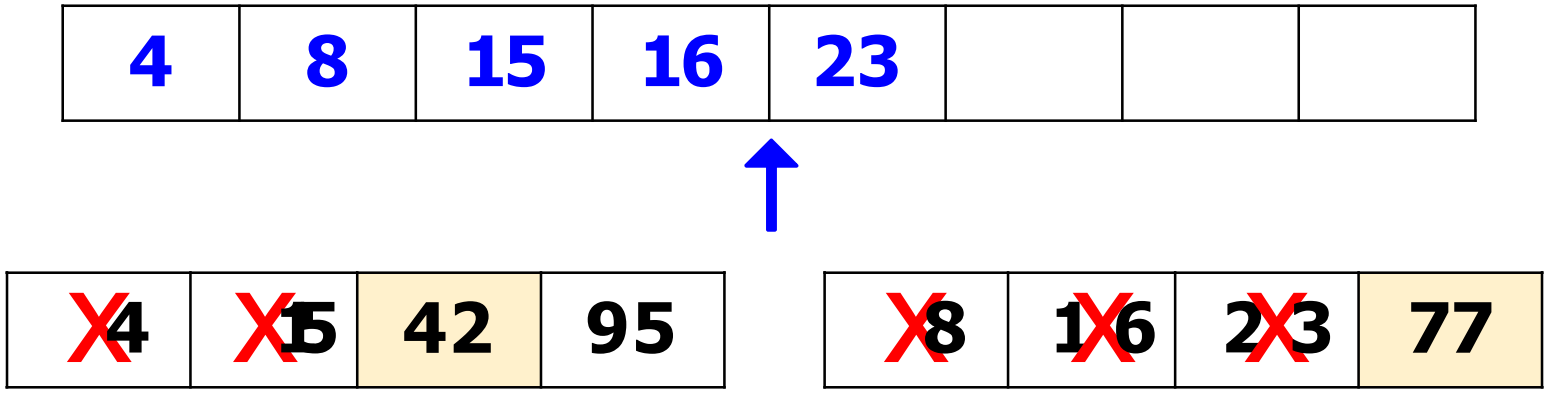
Exemplo



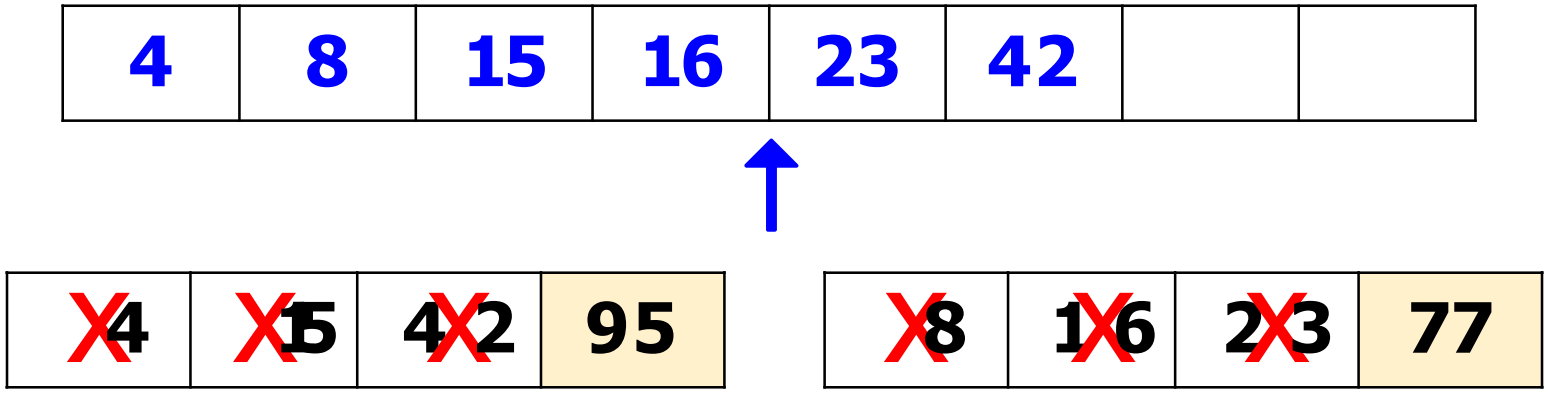
Exemplo



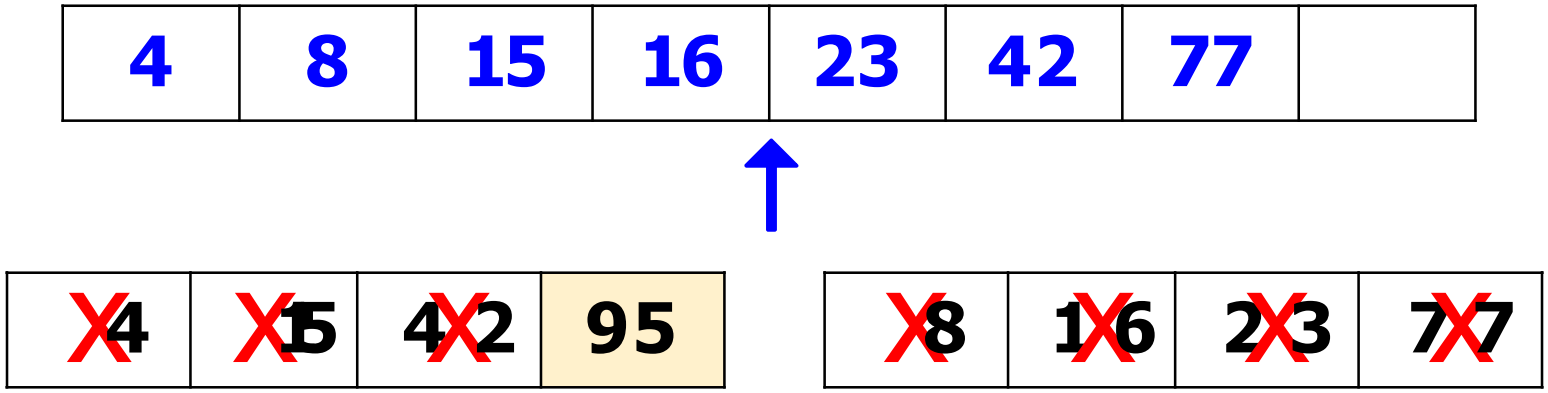
Exemplo



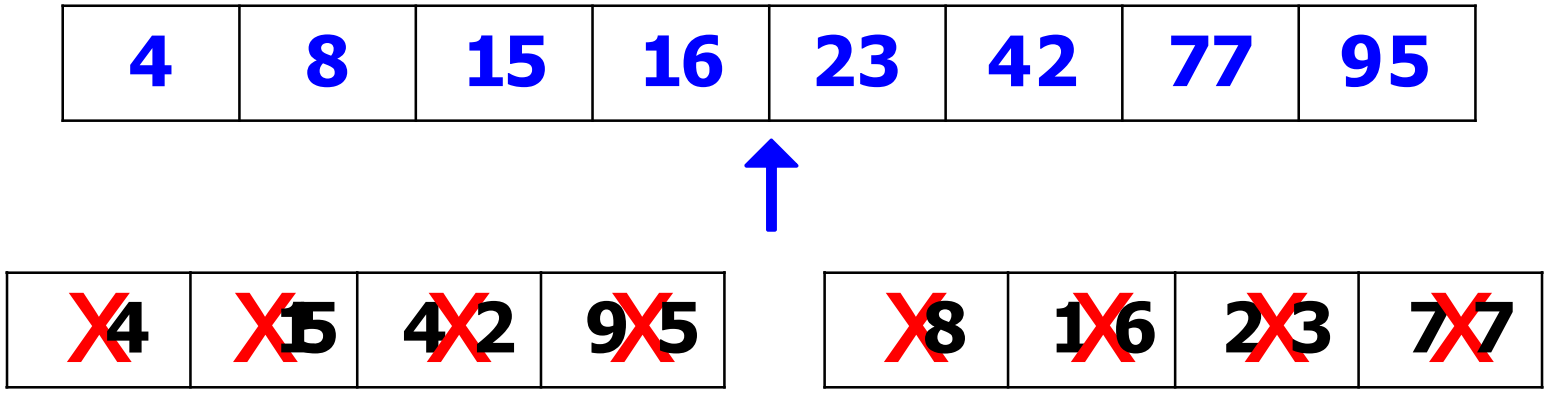
Exemplo



Exemplo



Exemplo



Exemplo

4	8	15	16	23	42	77	95

Conclusão

- Método estável

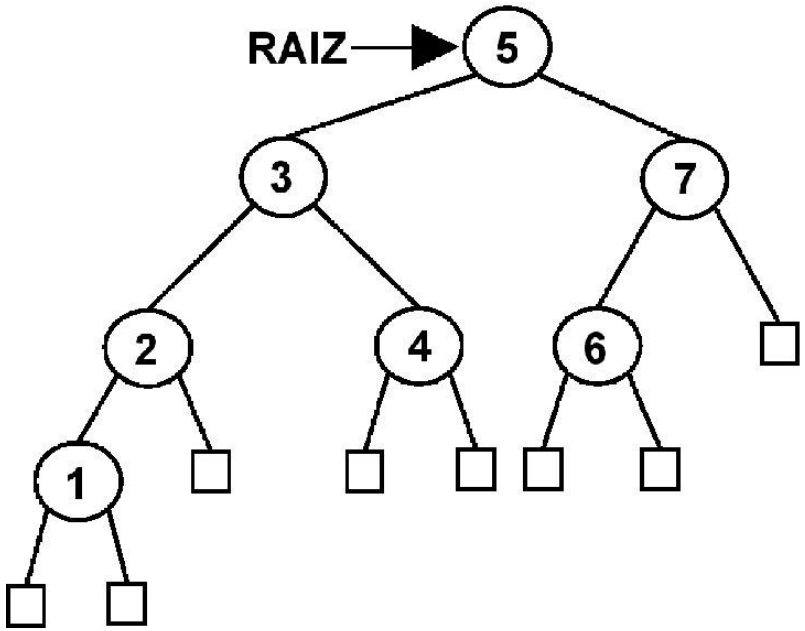
- Normalmente, implementado de forma recursiva e demandando memória adicional

- Faz Θ(n*lg(n)) comparações nos três casos (melhor, médio e pior)

Árvore

- Estrutura de dados cujas operações de inserção, remoção e substituição possuem a mesma eficiência

- Estrutura de dados que contém um conjunto finito de vértices (nós) e outro de arcos (arestas) que conectam os vértices

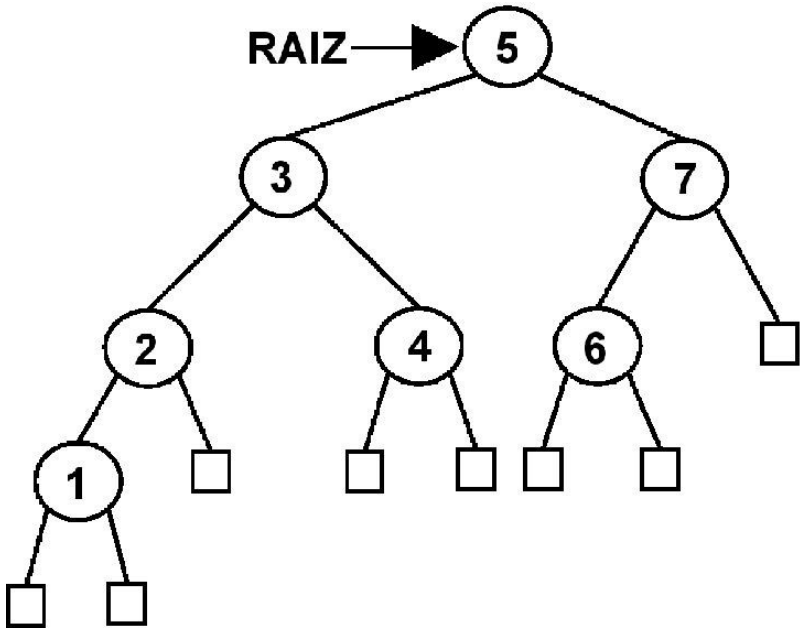


Árvore

- Estrutura de dados cujas operações de inserção, remoção e substituição possuem a mesma eficiência

- Estrutura de dados que contém um conjunto finito de vértices (nós) e outro de arcos (arestas) que conectam os vértices

O nó 5 é denominado nó raiz e ele está no nível 0

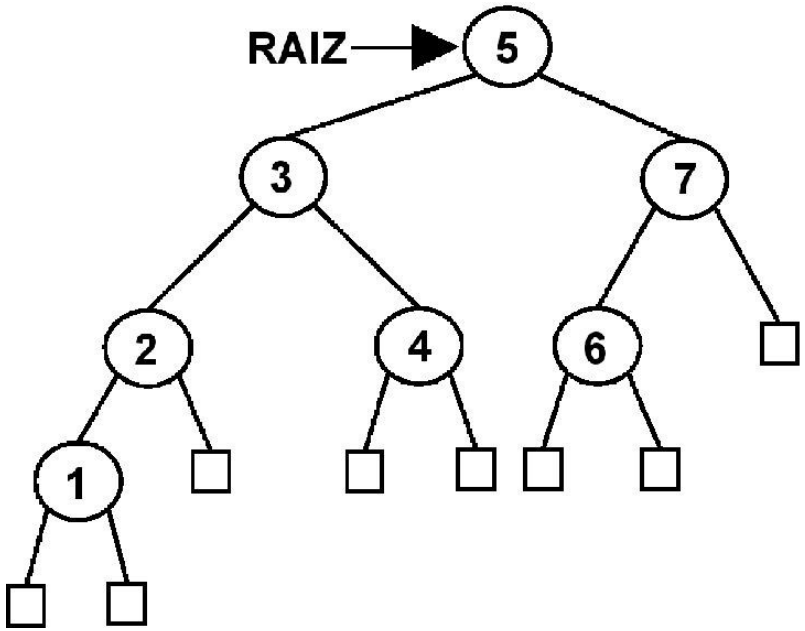


Árvore

- Estrutura de dados cujas operações de inserção, remoção e substituição possuem a mesma eficiência

- Estrutura de dados que contém um conjunto finito de vértices (nós) e outro de arcos (arestas) que conectam os vértices

Os nós 3 e 7 são filhos do 5 e esse é pai dos dois primeiros

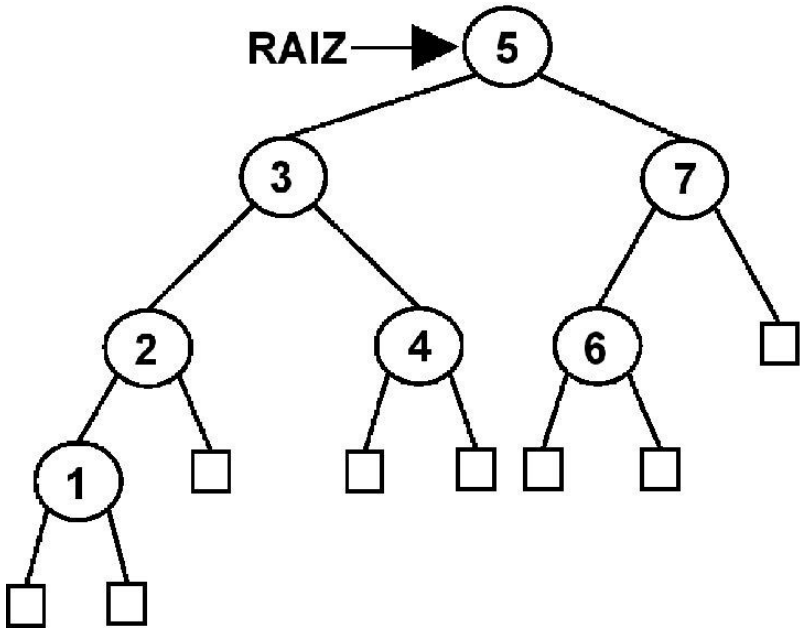


Árvore

- Estrutura de dados cujas operações de inserção, remoção e substituição possuem a mesma eficiência

- Estrutura de dados que contém um conjunto finito de vértices (nós) e outro de arcos (arestas) que conectam os vértices

Um nó com filho(s) é chamado de nó interno e outro sem, de folha

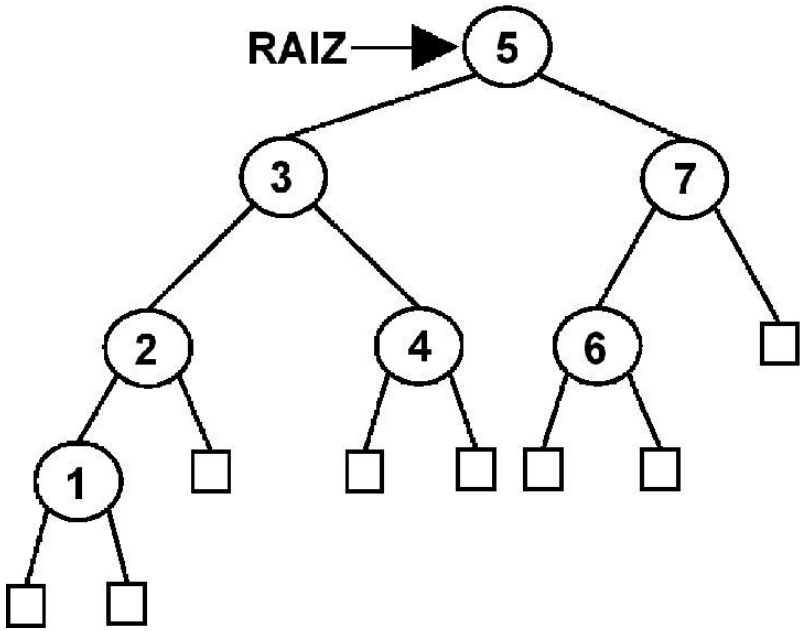


Árvore

- Estrutura de dados cujas operações de inserção, remoção e substituição possuem a mesma eficiência

- Estrutura de dados que contém um conjunto finito de vértices (nós) e outro de arcos (arestas) que conectam os vértices

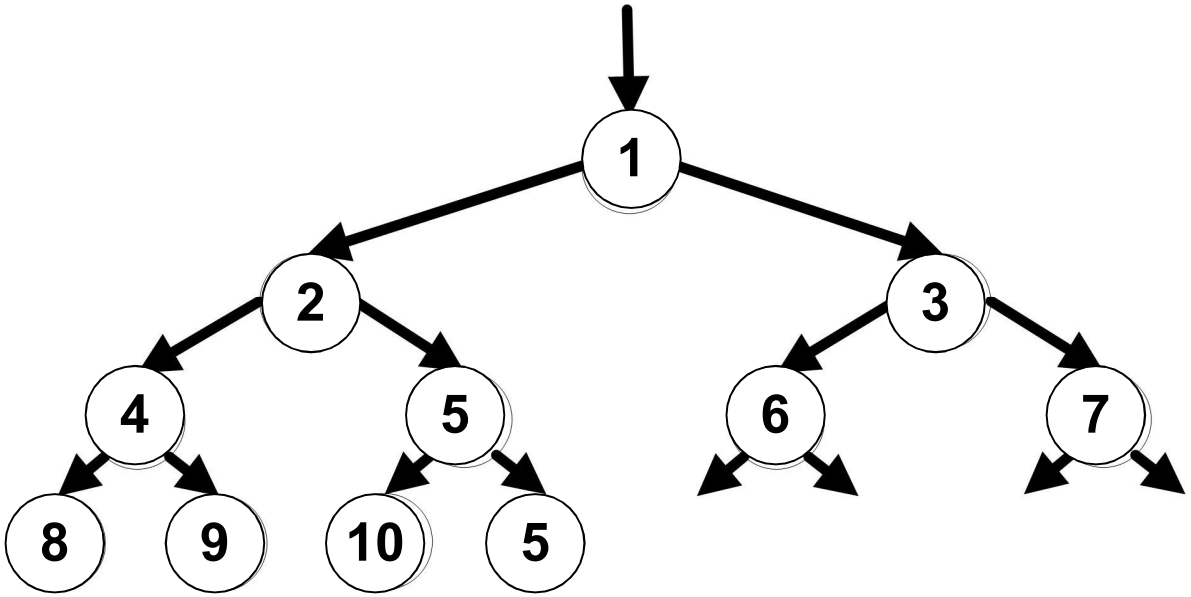
Nosso exemplo é uma árvore binária, pois cada nó tem no máximo dois filhos



Heap

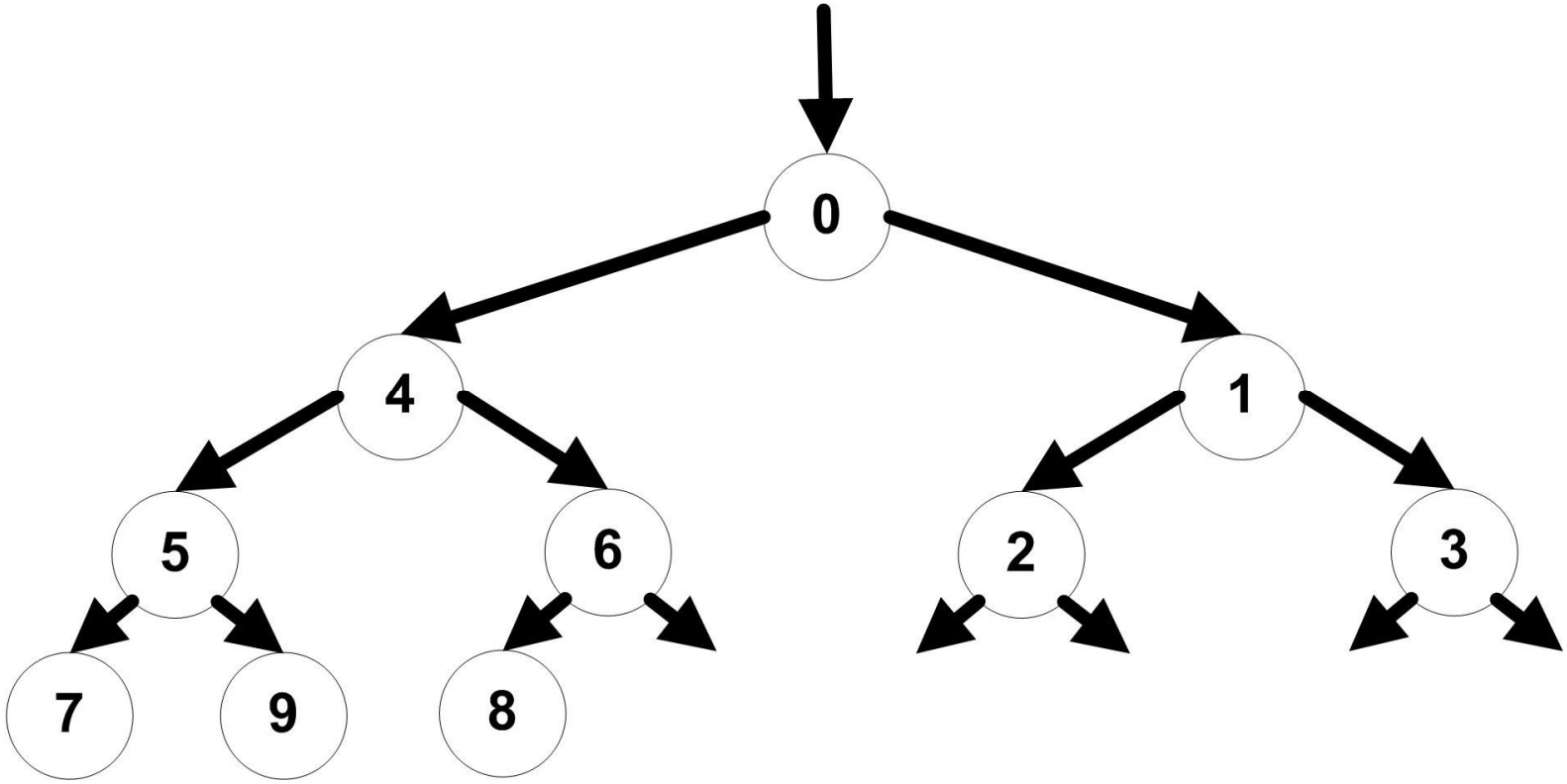
- Árvore binária em que cada nó é menor ou igual que seus filhos, fazendo com que a raiz tenha o menor valor

- Suas folhas ocupam um ou dois níveis sendo que o penúltimo é completo e as folhas do último nível se agrupam o mais à esquerda possível



Exemplo

- Heap com os elementos 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9



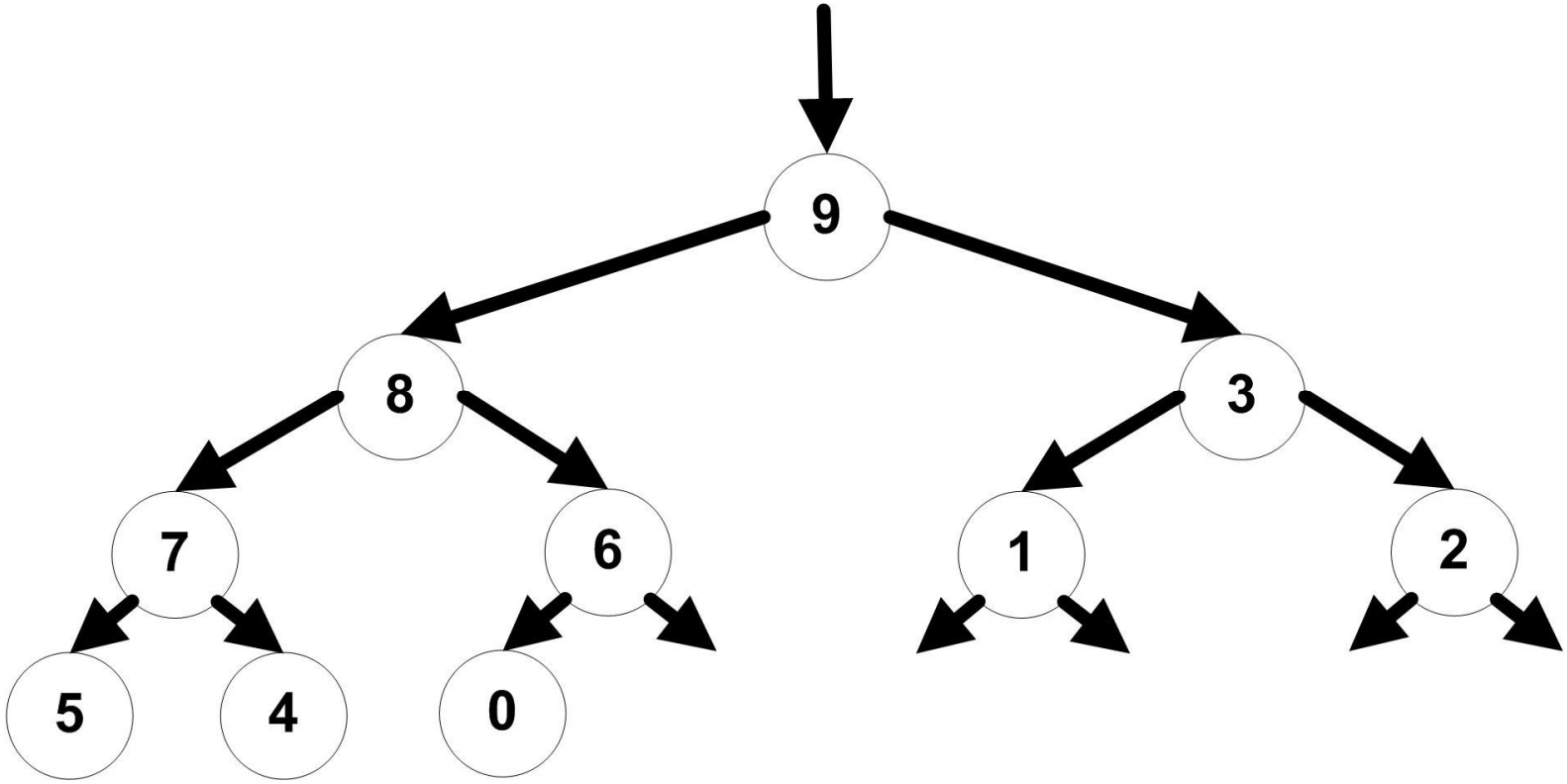
Heap Invertido

- Árvore binária em que cada nó é **maior** ou igual que seus filhos, fazendo com que a raiz tenha o **maior** valor

- Suas folhas ocupam um ou dois níveis sendo que o penúltimo é completo e as folhas do último nível se agrupam o mais à esquerda possível

Exemplo

- Heap invertido com os elementos 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9



Ordenação por seleção

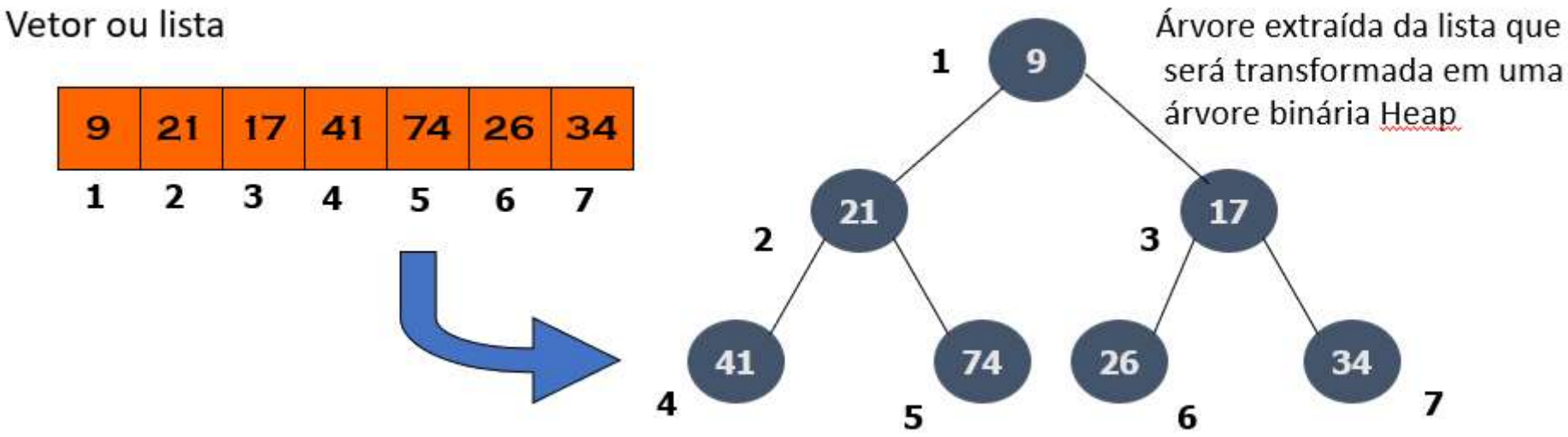
- Método Heapsort
  - Utiliza a seleção em árvore binária para a obtenção do elementos de um vetor na ordem desejada
- Possui duas fase distintas:
  - Construção da árvore binária com elementos do vetor;
  - Usa-se este heap construído para a seleção dos elementos na ordem desejada.

Ordenação por seleção

- Método Heapsort
  - Uma árvore binária Heap é uma árvore em que os nós ascendentes são sempre maiores que os descendentes, ou seja, o nó pai é sempre maior que os nós filhos.

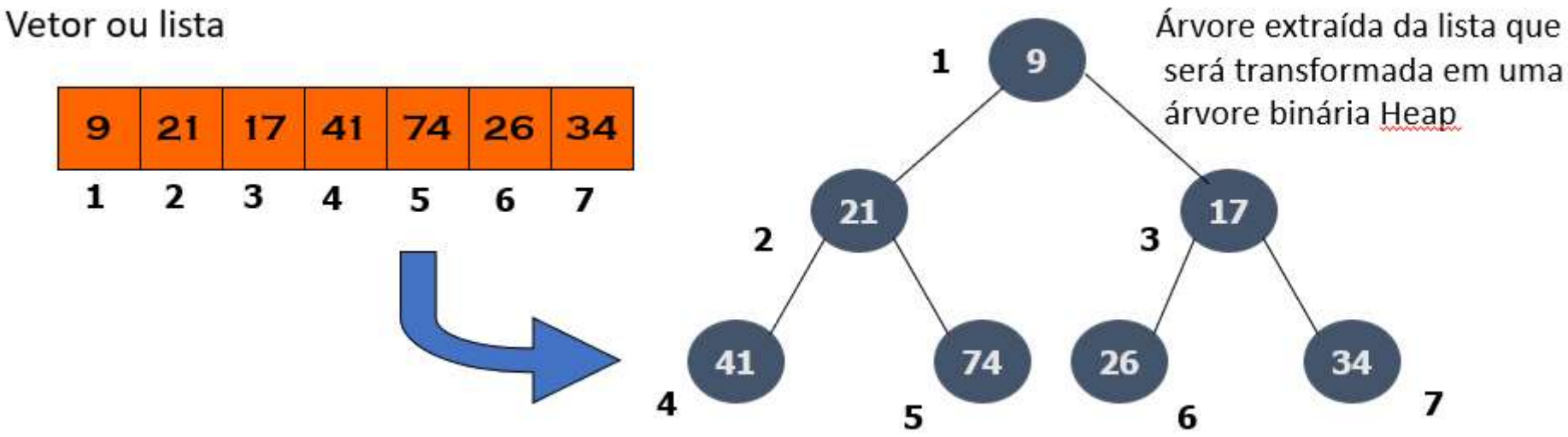
Ordenação por seleção

- Método heap-sort:
- Construindo a árvore heap a partir de um vetor:



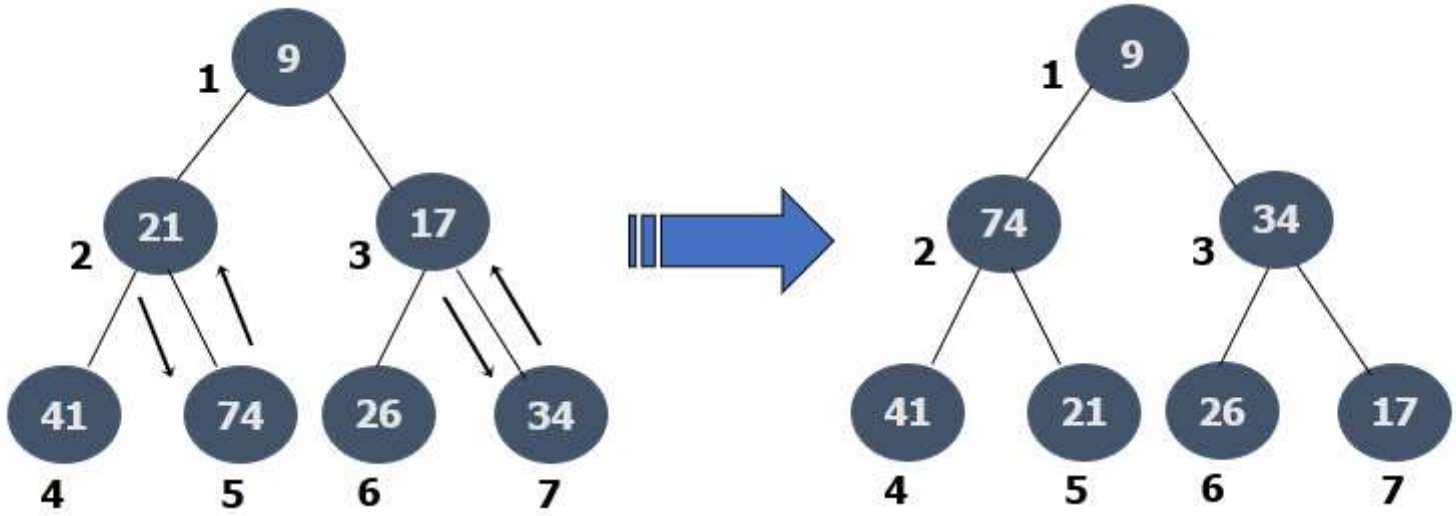
Ordenação por seleção

- Método heap-sort:
- Construindo a árvore heap a partir de um vetor:



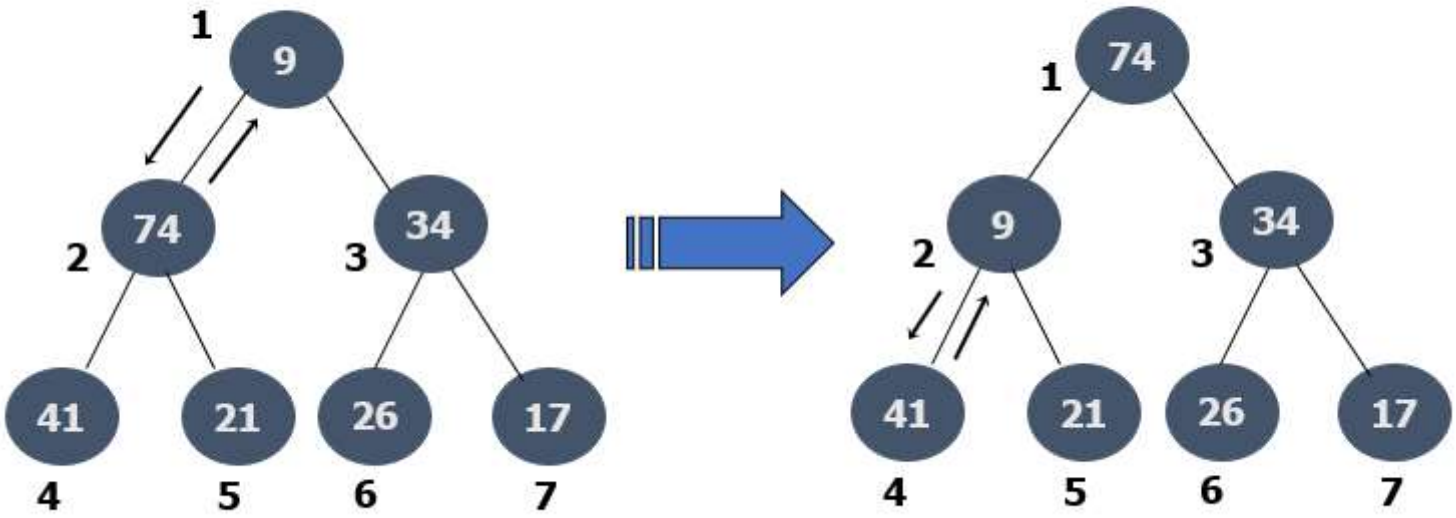
Ordenação por seleção

- O algoritmo testa se o Nó Pai é maior que os nós filhos, se for ele não faz nada.
- Caso contrário, ele testa qual filho é maior (se o esquerdo ou direito) e o troca pelo Nó Pai.



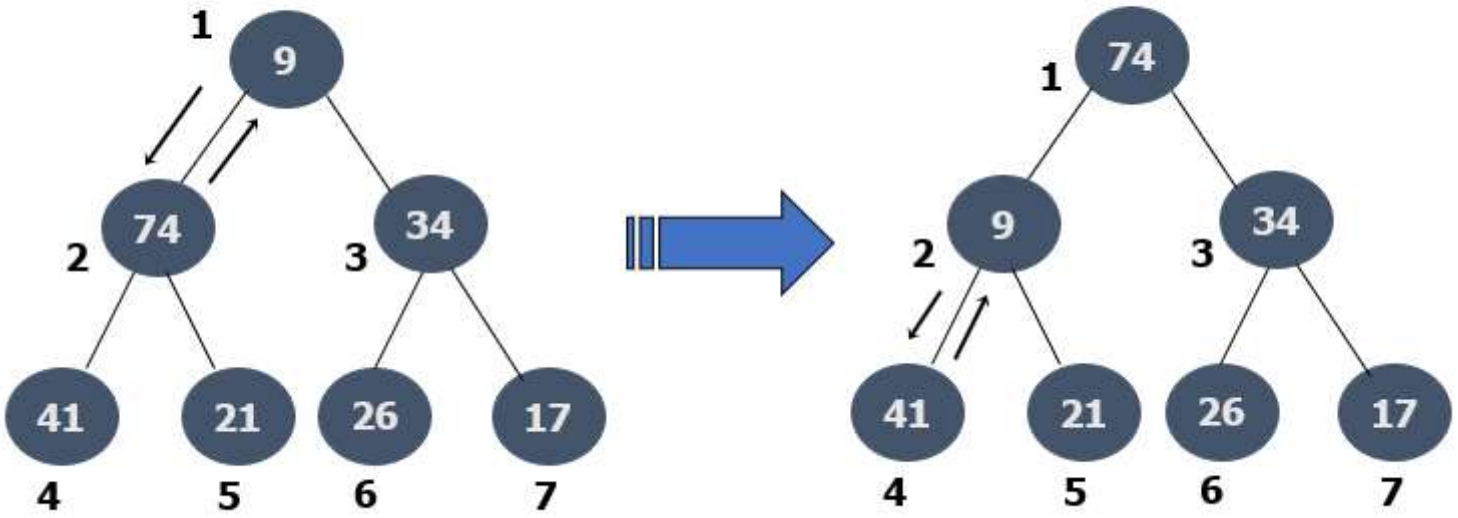
Ordenação por seleção

- Continuando a montar a árvore heap



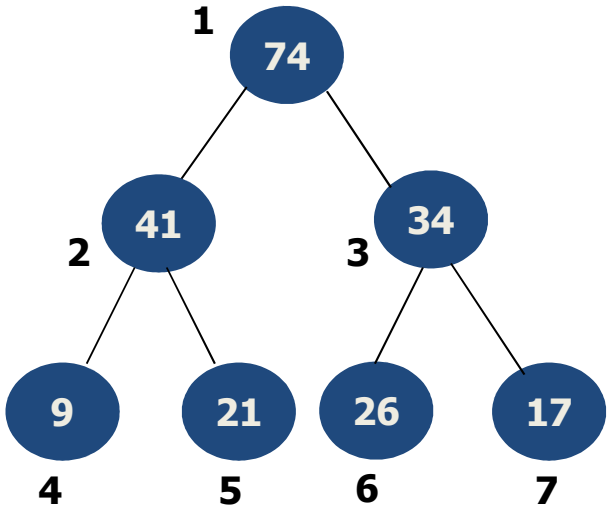
Ordenação por seleção

- Continuando a montar a árvore heap



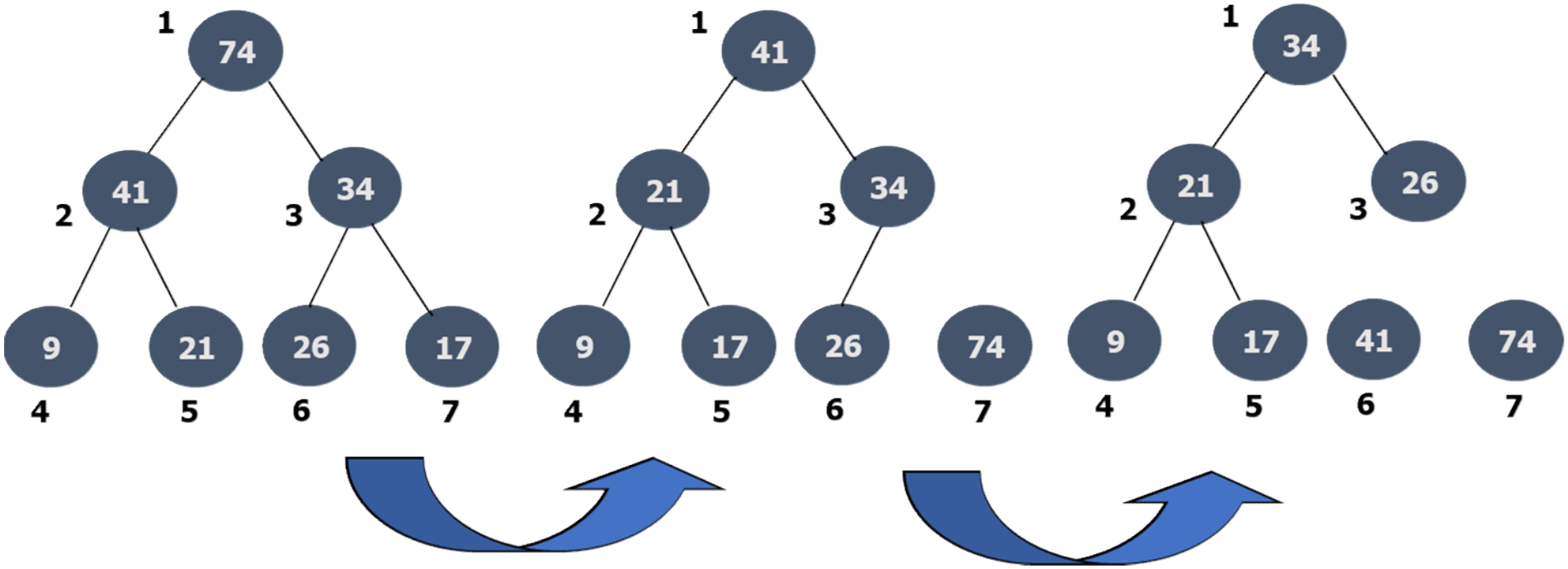
Ordenação por seleção

- Está é uma árvore Heap, todos os nós descendentes são menores que os nós ascendentes, ou seja os nós pais maiores que os nós filhos.
- Outra propriedade importante é que na Árvore Heap, a raiz principal terá sempre o maior elemento da árvore.



Ordenação por seleção

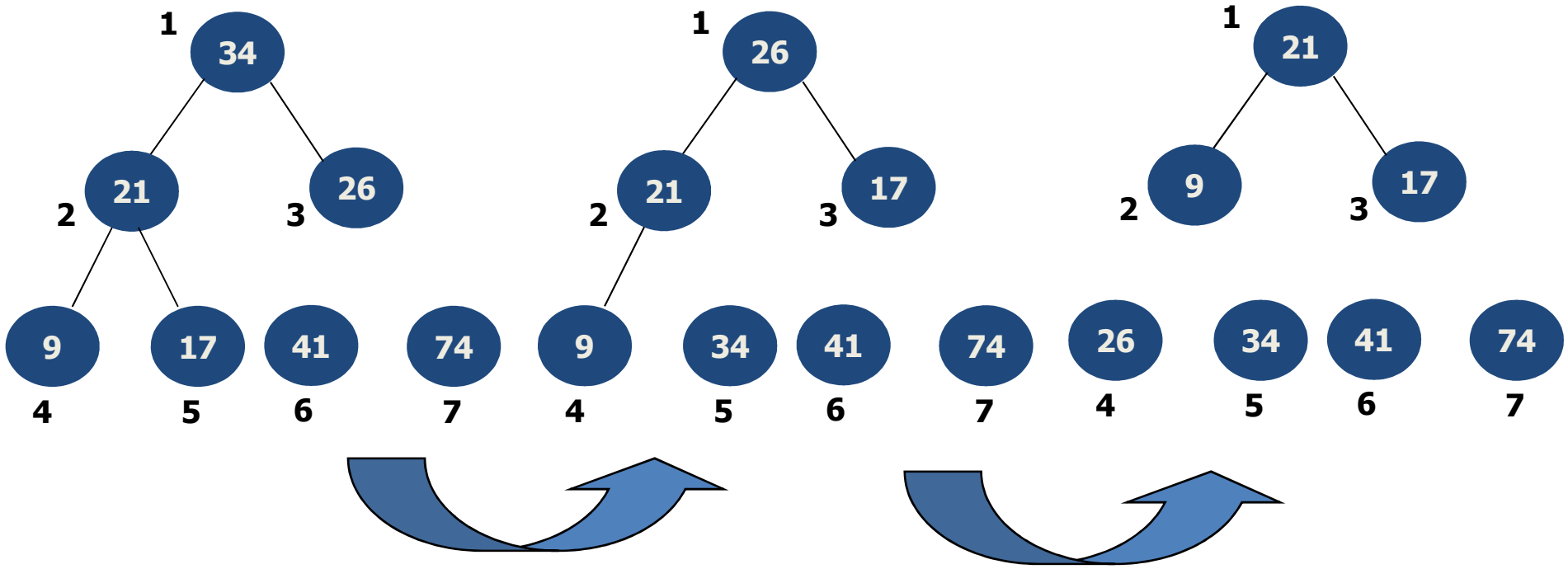
- Agora vamos ordem o vetor em ordem crescente, retirando a raiz da Árvore Heap;



- Cada vez que extrai a raiz principal, chama-se a função Max_Heap, que garante a maior chave sempre no topo, é um Heap Máximo.

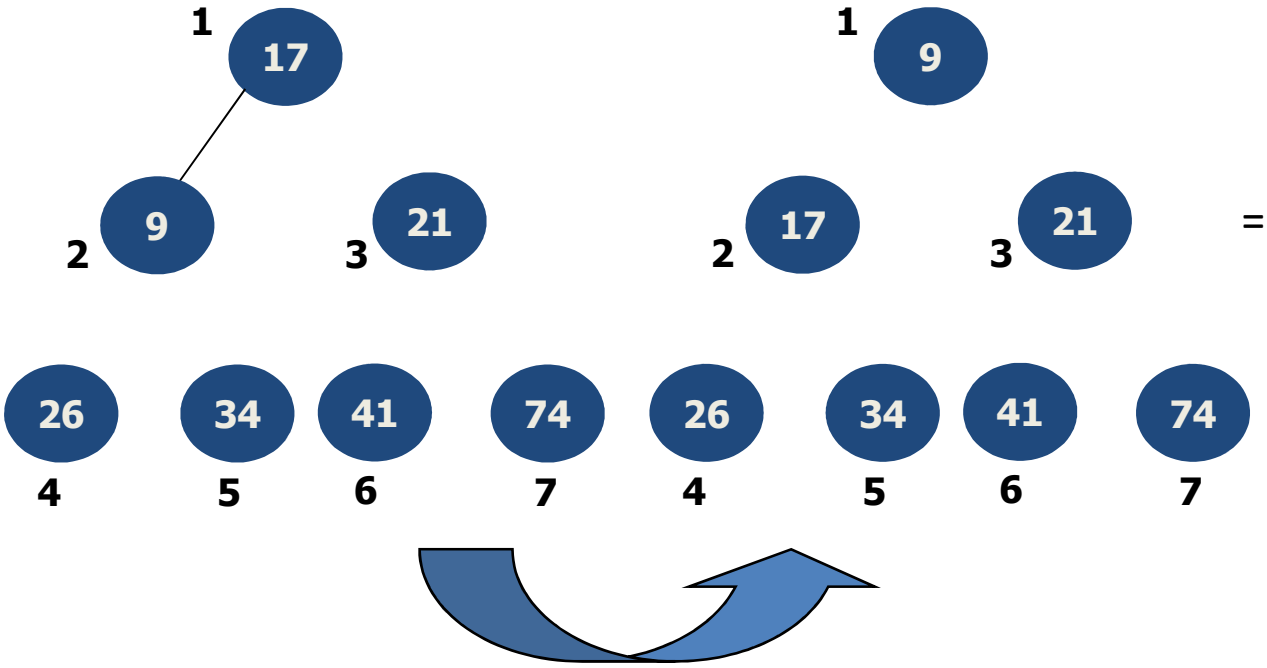
Ordenação por seleção

- Agora vamos ordem o vetor em ordem crescente, retirando a raiz da Árvore Heap;



Ordenação por seleção

- Agora vamos ordem o vetor em ordem crescente, retirando a raiz da Árvore Heap;



Lista em ordem crescente

=

1	2	3	4	5	6	7
9	17	21	26	34	41	74

Considerações

- É um algoritmo ótimo;
- Complexidade de Tempo: O(n . log n);
- Não requer espaço extra de memória, por isso é mais

  Rápido que o Merge-Sort;
- Na prática é um pouco mais lento do que o Quick-Sort.

Algoritmo

```
public static void heapSort(int[] arr) {
    int n = arr.length;

    // Construir um max-heap
    for (int i = n / 2 - 1; i >= 0; i--) {
        heapify(arr, n, i);
    }

    // Extrair elementos do heap um por um
    for (int i = n - 1; i > 0; i--) {
        // Move a raiz (o maior elemento) para o final
do array
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;

        // Chama o heapify na sub-árvore reduzida
        heapify(arr, i, 0);
    }
  }
```

```
public static void heapify(int[] arr, int n, int i) {
    int largest = i; // Inicializa o maior como a raiz
    int left = 2 * i + 1; // Índice do filho esquerdo
    int right = 2 * i + 2; // Índice do filho direito

    // Se o filho esquerdo for maior que a raiz
    if (left < n && arr[left] > arr[largest]) {
        largest = left;
    }

    // Se o filho direito for maior que o maior até agora
    if (right < n && arr[right] > arr[largest]) {
        largest = right;
    }

    // Se o maior não for a raiz
    if (largest != i) {
        int swap = arr[i];
        arr[i] = arr[largest];
        arr[largest] = swap;

        // Recursivamente chama o heapify na sub-árvore afetada
        heapify(arr, n, largest);
    }
  }
```

Qual é o melhor algoritmo para ordenação

- A resposta não é simples, pois os resultados obtido não levam em conta outras operações como controle dos laços e cálculos dos índices. Além disso, eles desconsideram arquitetura, SO, compilador e hardware.
- Uma forma de ajudar na resposta é a avaliação experimental dos métodos de ordenação

Avaliação para TAD na Ordem Aleatória

Algoritmo	500	5000	10000	30000
Inserção	11,3	87	161	-
Seleção	16,2	124	228	-
Shellsort	1,2	1,6	1,7	2
Quicksort	1	1	1	1
Heapsort	1,5	1,6	1,6	1,6

Avaliação para TAD na Ordem Crescente

Algoritmo	500	5000	10000	30000
Inserção	1	1	1	1
Seleção	128	1524	3066	-
Shellsort	3,9	6,8	7,3	8,1
Quicksort	4,1	6,3	6,8	7,1
Heapsort	12,2	20,8	22,4	24,6

Avaliação para TAD na Ordem Decrescente

Algoritmo	500	5000	10000	30000
Inserção	40,3	305	575	-
Seleção	29,3	221	417	-
Shellsort	1,5	1,5	1,6	1,6
Quicksort	1	1	1	1
Heapsort	2,5	2,7	2,7	2,9

Complexidade de tempo: Algoritmos de Ordenação

Método	*Melhor*	*Médio*	*Pior*	*Estável*	*Eficiente*	*in-place*	*Limitado*
Bubblesort	$O(n^2)$			sim	não	sim	não
Selectionsort	$O(n^2)$			*não	não	sim	não
Insertsort	$O(n)$	$O(n^2)$		sim	não	sim	não
Countingsort	$O(n)$			sim	não	não	sim
Radixsort	O (nk)	O (nk)	O (nk)	sim	não	não	sim
Shellsort	O(n^1.25) ou O(n (ln n)^2)			não	sim	sim	não
Mergesort	O(n log(n))			sim	sim	não	não
Quicksort	O(n log(n))		O (nk)	*não	sim	sim	não
Heapsort	O(n log(n))			não	sim	sim	não

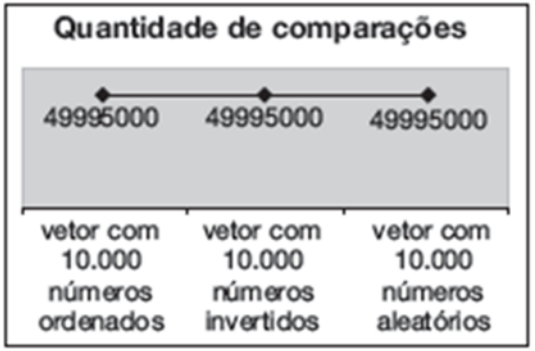
*Existe versões estáveis.
In-place: não requerem alocação de memória extra para ordenar os elementos.
Limitados: não funciona em todos os casos de ordenação.

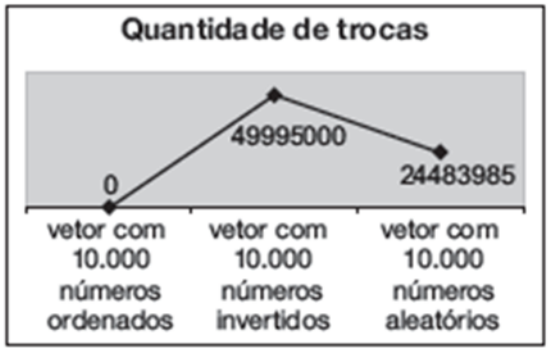
Comparação de Resultado segundo a literatura

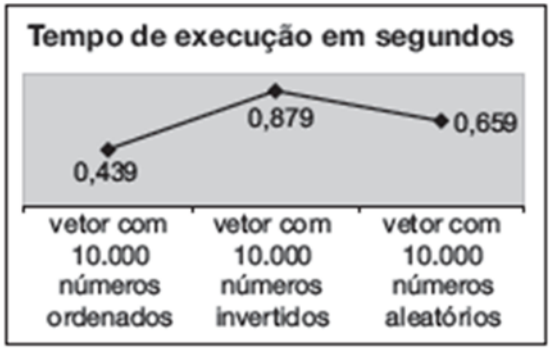
Comparações de desempenho dos algoritmos segundo livro:

Comparação de Resultado segundo a literatura

BubbleSort

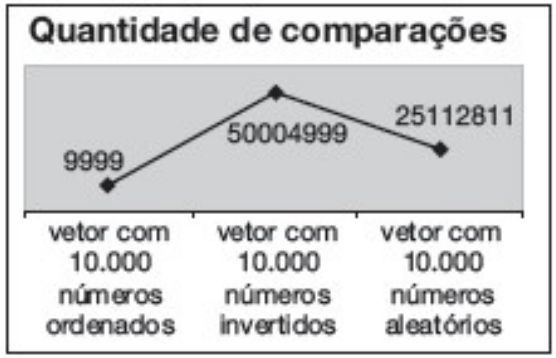


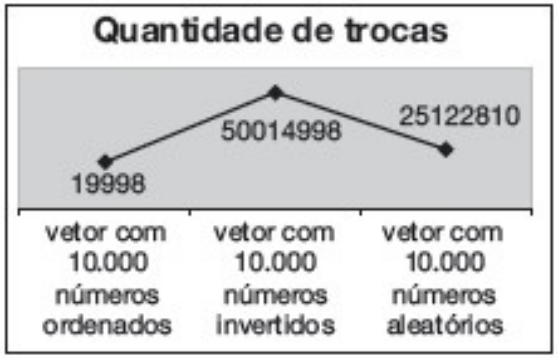


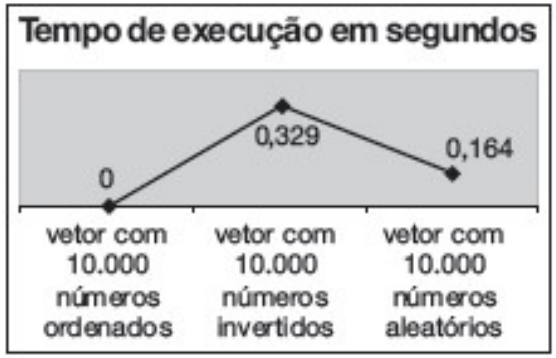


Comparação de Resultado segundo a literatura

InsertionSort

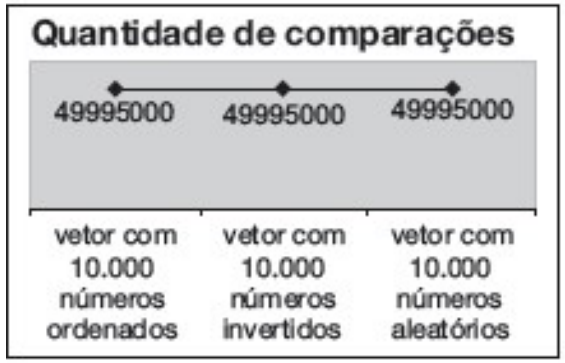


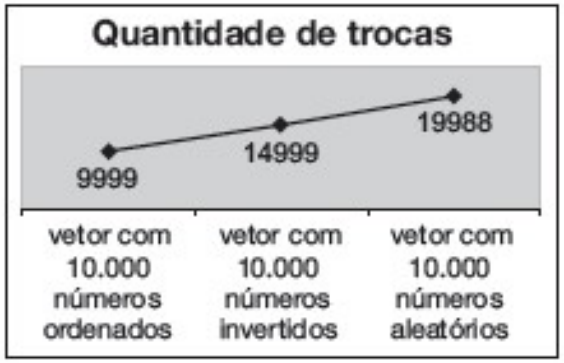


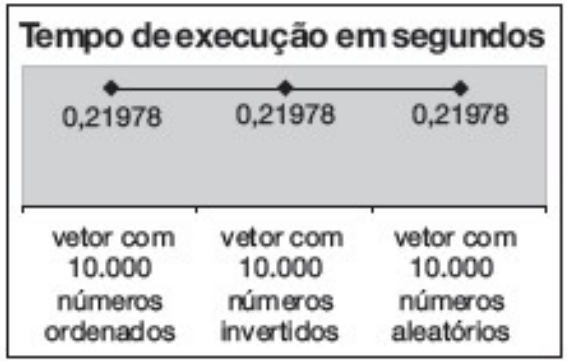


Comparação de Resultado segundo a literatura

SelectionSort

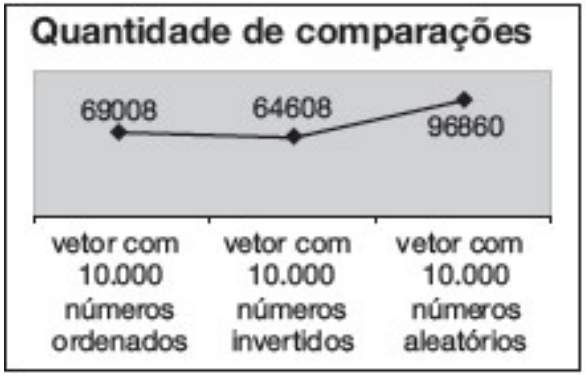


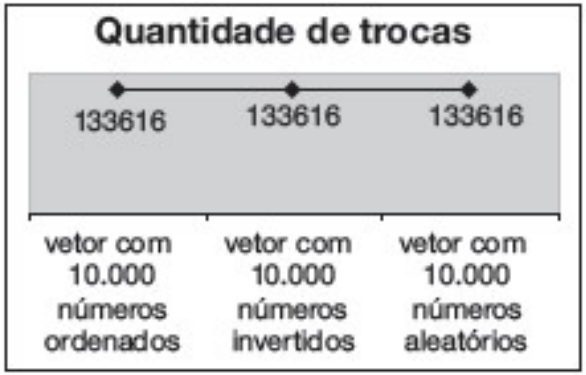


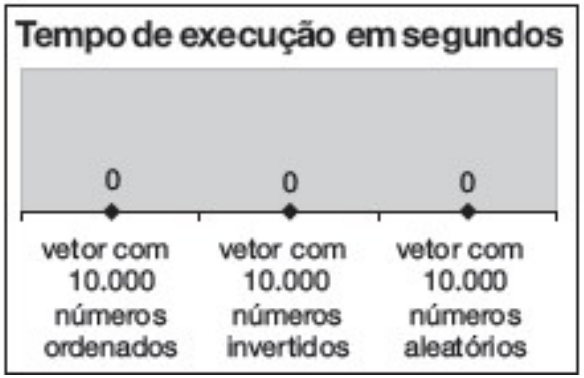


Comparação de Resultado segundo a literatura

MergeSort

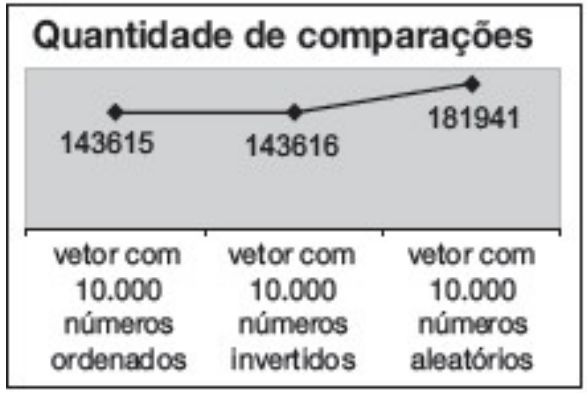


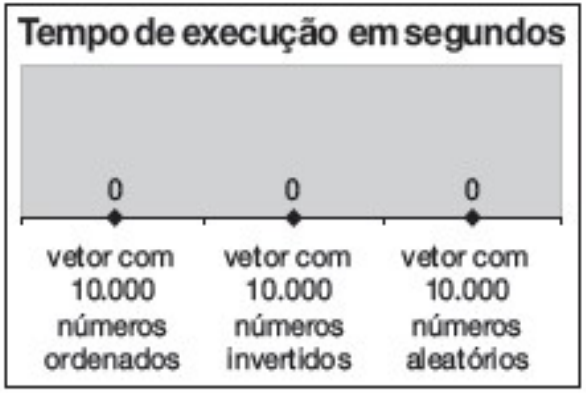




Comparação de Resultado segundo a literatura

QuickSort





Qual é o melhor algoritmo para ordenação

- A vantagem do algoritmo de seleção é seu número de movimentos de objetos que é $\Theta(n)$.
- Os algoritmos de Inserção é interessante para *arrays* ordenados (ou praticamente).
- Os métodos de Inserção é estável.
- O Quicksort é o mais eficiente para uma grande variedade de situações
- O pior caso do Quicksort é $\Theta(n^2)$
- O pior caso do Mergesort e do Heapsort é $\Theta(n*lg(n))$
- Uma desvantagem do Mergesort é o fato dele “duplicar” sistematicamente os vetores
- Uma vantagem do Quicksort é sua pilha de recursividade reduzida

Próxima aula

Tipo abstrato de dado flexível