Ordenação - Mergesort

FURB
UNIVERSIDADE DE BLUMENAU

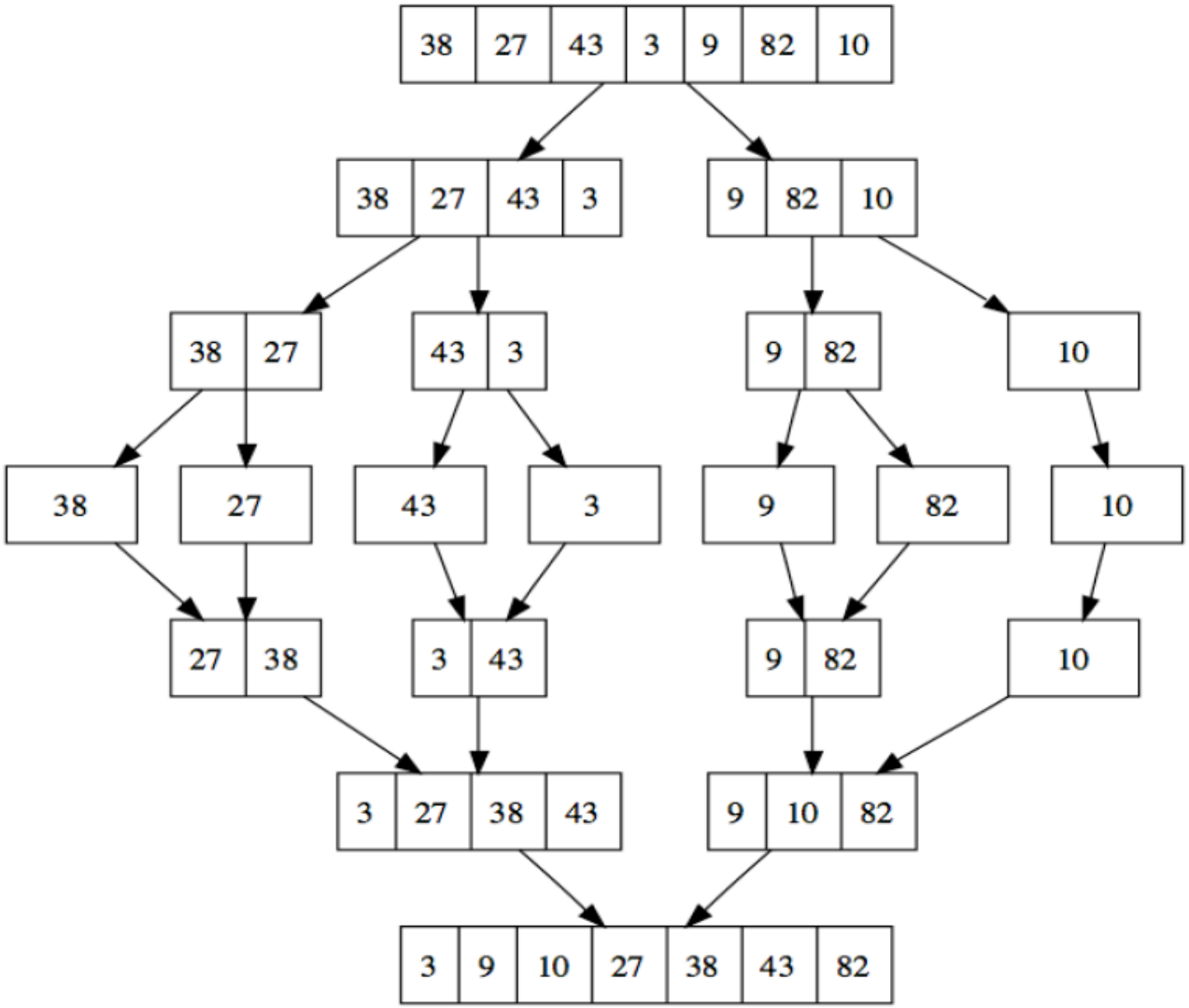
Mergesort

- Utiliza a abordagem dividir para conquistar
  - Dividir o problema em problemas menores
  - Conquistar a solução, resolvendo os problemas recursivamente
- Combinar as soluções dadas aos sub-problemas
- Possui ordem de complexidade O(n log(n))

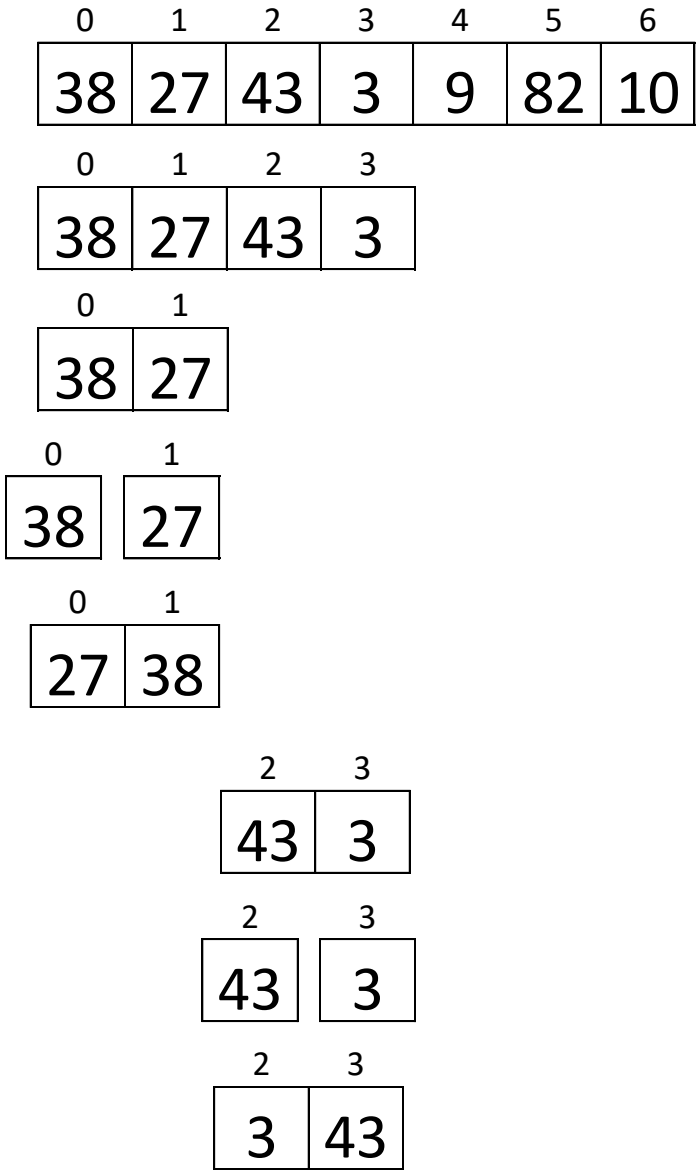
Visão geral

- O mergesort divide um vetor pela metade, então use a função merge para unir cada metade produzindo um terceiro vetor ordenado.
  - Divide-se o vetor a ser classificado ao meio: a cada função recursiva do mergesort, o vetor é dividido em duas partes até que resultem apenas dois vetores com apenas um único elemento
  - O algoritmo deve unificar dois vetores, unindo-os num vetor ordenado
- Também conhecido como “ordenação por intercalação”

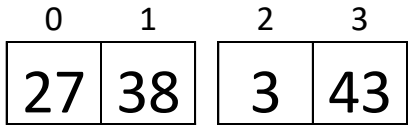
Exemplo

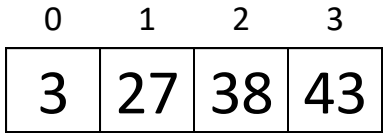

38 27 43 3 9 82 10
38 27 43 3
9 82 10
38 27
43 3
9 82
10
38
27
43
3
9
82
10
27 38
3 43
9 82
10
3 27 38 43
9 10 82
3 9 10 27 38 43 82

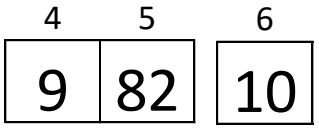
Exemplo

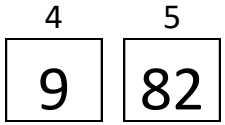

0 1 2 3 4 5 6
38 27 43 3 9 82 10
0 1 2 3
38 27 43 3
0 1
38 27
0 1
38 27
0 1
27 38
2 3
43 3
2 3
43 3
2 3
3 43

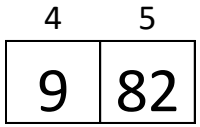
Exemplo

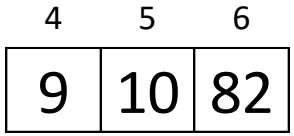




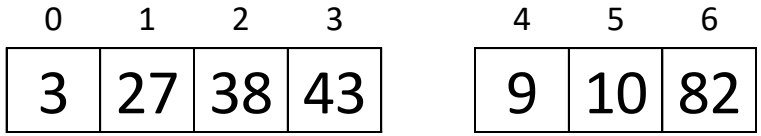


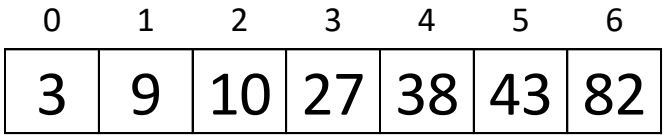






Exemplo





Algoritmo

Algoritmo: mergeSort()
n ← size(info)-1;
mergeSort(0, n);

Algoritmo: mergeSort(int inicio, int fim)

se (inicio < fim) **então**
  meio ← ⌊(inicio + fim)/2⌋;
  mergeSort(inicio, meio);
  mergeSort(meio+1, fim);
  merge(inicio, fim, meio);
fim-se

Algoritmo

Algoritmo: merge (int inicio, int fim, int meio)

```
tamEsquerda ← meio – inicio+ 1;
esquerda ← new int[tamEsquerda];
para i ← 0 até tamEsquerda-1 faça
  esquerda[i] ← info[inicio+i];
fim-para;

tamDireita ← fim- meio;
direita ← new int[tamDireita];
para i ← 0 até tamDireita-1 faça
  direita[i] ← info[meio+1+i];
fim-para;
```

Continua...

Algoritmo (continuação)

.. continuação

Algoritmo: merge (int inicio, int fim, int meio)

```
cEsq ← 0;
cDir ← 0;

para i ← inicio até fim faça
   se (cEsq < tamEsquerda) e (cDir < tamDireita) então
      se (esquerda[cEsq] < direita[cDir]) então
         info[i] ← esquerda[cEsq];
         cEsq ← cEsq+1;
      senão
         info[i] ← direita[cDir];
         cDir ← cDir+1;
      fim-se
   senão
      break;
   fim-se;
fim-para;
```

Continua...

Algoritmo

... continuação

Algoritmo: merge (int inicio, int fim, int meio)

```
enquanto (cEsq < tamEsquerda)
  info[i] ← esquerda[cEsq];
  cEsq ← cEsq + 1;
  i ← i + 1;
fim-enquanto;

enquanto (cDir < tamDireira)
  info[i] ← direita[cDir];
  cDir ← cDir + 1;
  i ← i + 1;
fim-enquanto;
```

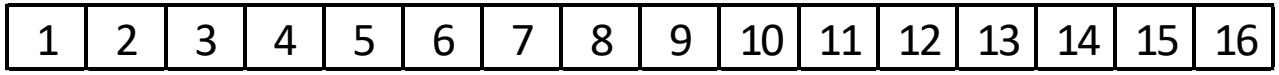
Usar o valor da variável "i" do laço anterior

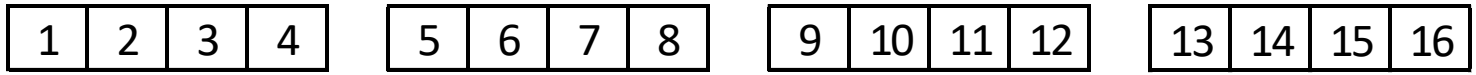
Observações

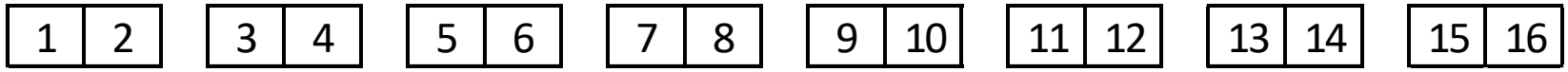
- Para instanciar um vetor genérico de objetos que realizam a interface Comparable, executar:

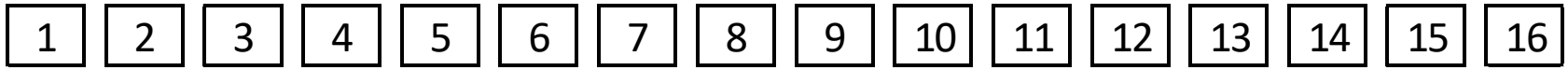
```
T[] vetor = (T[]) new Comparable[tamanho];
```

Análise do algoritmo - particionamento

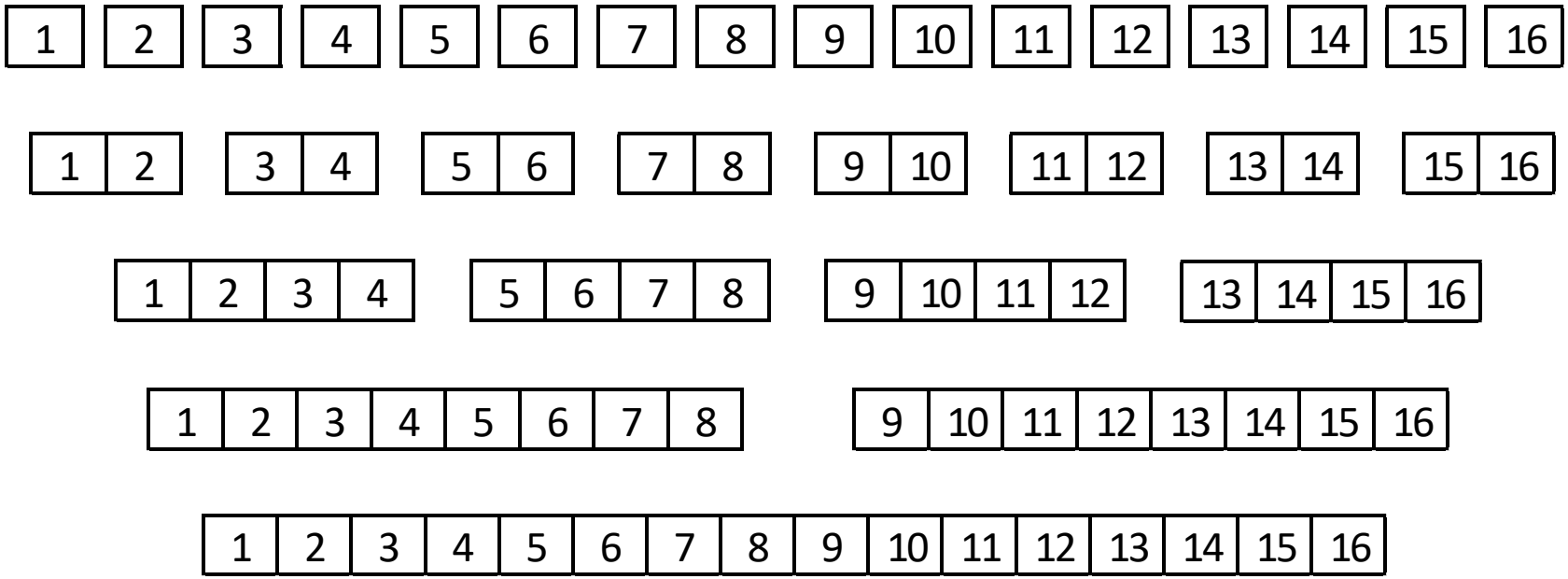








Análise do algoritmo - merge



O número de etapas é: log n

Exemplo: log 16 = 4

Copia para área temporária (n)
Ordenação do vetor resultante (n)

2n . log(n)

n . log(n)