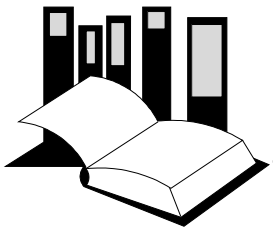
## Ordenação em Vetores



- Esta aula introduz métodos de ordenação em vetores que está entre as tarefas mais freqüentemente encontradas em programação de computadores
- Serão abordados métodos diretos de ordenação por inserção, seleção e permutação

**Prof. Dr. José Augusto Baranauskas**
DFM-FFCLRP-USP



## Ordenação

- Ordenação é o processo de rearranjo de um certo conjunto de objetos (elementos) de acordo com um critério (ordem) específico
- O objetivo da ordenação é facilitar a localização dos membros de um conjunto de objetos
- Assim sendo, é uma atividade fundamental e universalmente utilizada para a elaboração de algoritmos mais complexos.
- Exemplos de casos em que os objetos estão ordenados podem ser encontrados em listas telefônicas, imposto de renda, índices, dicionários, almoxarifados etc. e em quase todos os casos em que estejam colecionados objetos sujeitos à procura e alteração



## Notação

- Assumindo-se que os elementos são dados em um vetor **a** de **N** elementos, ou seja:

  a[1], a[2], ..., a[N]
- a ordenação consistirá em permutar tais elementos, levando ao vetor

  a[k1], a[k2], ..., a[kN]
- de forma tal que, dada uma função **f** de ordenação, seja satisfeita a seguinte relação:

  f(a[k1]) <= f(a[k2]) <= ... <= f(a[kN]).
- A função de ordenação não é avaliada segundo uma regra específica de cálculo mas armazenada como um componente explícito (campo) de cada elemento. Seu valor é denominado *chave* do elemento. Como conseqüência deste fato, as estruturas do tipo registro (struct) são particularmente mais adequadas para representar tais elementos. Por exemplo:

```
typedef struct item
{ int key;      // chave de ordenação
  ...           // demais campos da estrutura
};
item a[N+1];
```



## Notação

- No que se diz respeito aos algoritmos de ordenação aqui vistos, a chave é o único componente relevante, não sendo necessário definir nenhum outro campo além dele
- Portanto, nas discussões seguintes, serão descartadas quaisquer informações adicionais e os elementos serão considerados como sendo todos do tipo **int**, ou seja:

```
int a[N+1];
```

- A escolha do tipo **int** para a chave é, de certa forma, arbitrária, podendo evidentemente ser utilizado qualquer outro tipo na definição da relação de ordenação
- Assume-se os elementos do vetor **a** com índices 1, 2, ..., N em todos os algoritmos seguintes, não sendo utilizado o elemento de índice zero, exceto quando citado explicitamente no texto



## Estabilidade x Instabilidade

- Um método de ordenação é denominado *estável* se a ordem relativa dos elementos que exibam a mesma chave permanecer inalterada ao longo de todo o processo de ordenação; caso contrário, ele é denominado *instável*
- Em geral, a estabilidade da ordenação é desejável, especialmente quando os elementos já estiverem ordenados em relação a uma ou mais chaves secundárias



## Alguns Métodos de Ordenação

Diretos

- Inserção
  - Inserção Direta
  - Inserção Binária
- Seleção
  - Seleção Direta
- Permutação
  - Borbulhamento
  - Agitação

Avançados

- Shellsort (inserção)
- Quicksort (partição)
- Heapsort (árvore)

## Inserção Direta

- Neste método, os elementos são conceitualmente divididos em uma seqüência destino a[1], a[2], ...,a[i-1], e uma seqüência fonte a[i], a[i+1], ... a[N]
- Em cada passo, iniciando-se com i = 2 e incrementando-se i de uma em uma unidade, o i-ésimo elemento da seqüência vai sendo retirado e transferido para a seqüência destino, e inserido na posição apropriada (ordenada)

```
for(i = 2; i <= N; i++)
{ x = a[i];
  inserir x no local adequado em a[1], a[2], …, a[i]
}
```



## Inserção Direta

```
for(i = 2; i <= N; i++)
{ x = a[i];
  inserir x no local adequado em a[1], a[2], …, a[i]
}
```

| Vetor inicial | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
|---|---|---|---|---|---|---|---|---|
| i = 2 | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
| i = 3 | 12 | 45 | 56 | 43 | 95 | 19 | 8 | 67 |
| i = 4 | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |
| i = 5 | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |
| i = 6 | 12 | 19 | 43 | 45 | 56 | 95 | 8 | 67 |
| i = 7 | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |
| i = 8 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |



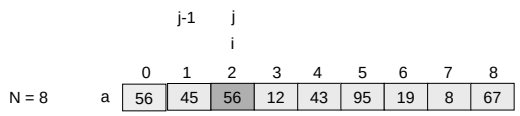
## Inserção Direta

- No processo de procurar o local apropriado para o elemento **x**, é conveniente utilizar, de modo alternado, operações de comparação e de movimentação, examinado cuidadosamente **x**, e comparando-o com o próximo elemento a[j] e então efetuando a inserção de x ou efetuando a movimentação do elemento a[j] para a direita, prosseguido-se, em seguida, para a esquerda
- Note que existem duas condições distintas que causam o término deste processo de análise:
  - Um elemento a[j] é encontrado com uma chave de valor menor do que o da chave do elemento **x**;
  - A extremidade esquerda do vetor **a** é atingida.
- Este caso de um *loop* com duas condições de término conduz uso da conhecida técnica da sentinela vista na aula sobre busca em vetores
- Observe que esta técnica é facilmente aplicada neste caso colocando-se uma sentinela com valor de **x** em a[0].



## Inserção Direta

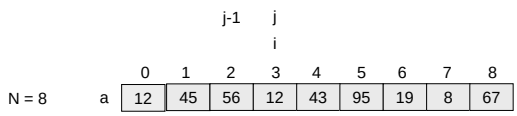
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```






## Inserção Direta

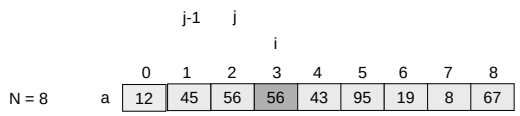
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```






## Inserção Direta

```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```



## Inserção Direta

```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```

j-1 j

i

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| N = 8 | a | 12 | 45 | 45 | 56 | 43 | 95 | 19 | 8 | 67 |

13

## Inserção Direta

```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```

j-1 j

i

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| N = 8 | a | 12 | 12 | 45 | 56 | 43 | 95 | 19 | 8 | 67 |

14

## Inserção Direta

```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```

j-1 j

i

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| N = 8 | a | 43 | 12 | 45 | 56 | 43 | 95 | 19 | 8 | 67 |

15

## Inserção Direta

```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```

j-1 j

i

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| N = 8 | a | 43 | 12 | 45 | 56 | 56 | 95 | 19 | 8 | 67 |

16

## Inserção Direta

```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```

j-1 j

i

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| N = 8 | a | 43 | 12 | 45 | 45 | 56 | 95 | 19 | 8 | 67 |

17

## Inserção Direta

```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```

j-1 j

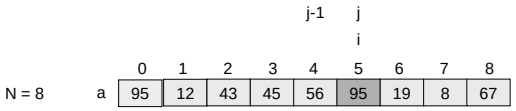
i

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| N = 8 | a | 43 | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |

18

## Inserção Direta

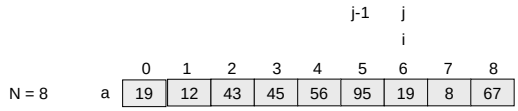
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




19

## Inserção Direta

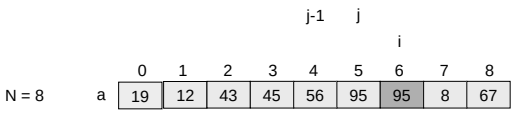
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




20

## Inserção Direta

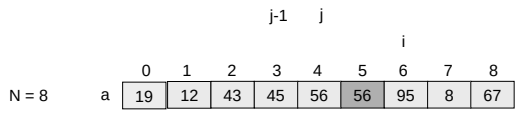
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




21

## Inserção Direta

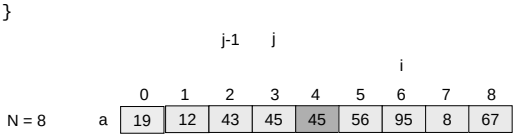
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




22

## Inserção Direta

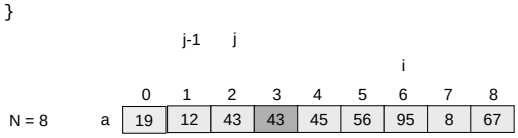
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




23

## Inserção Direta

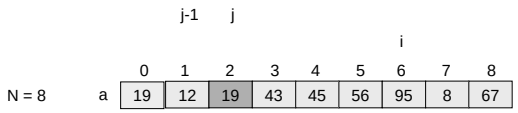
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




24

## Inserção Direta

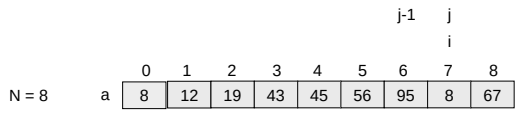
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




25

## Inserção Direta

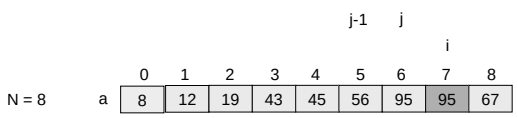
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




26

## Inserção Direta

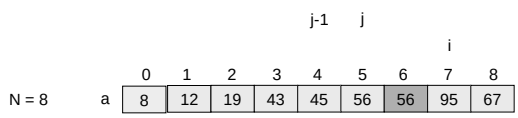
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




27

## Inserção Direta

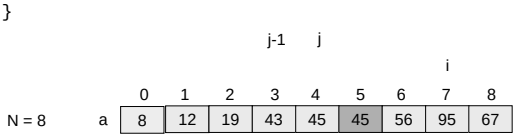
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




28

## Inserção Direta

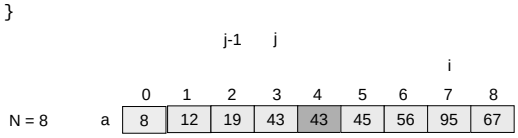
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




29

## Inserção Direta

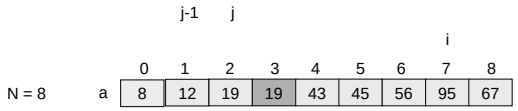
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




30

## Inserção Direta

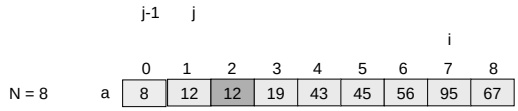
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




31

## Inserção Direta

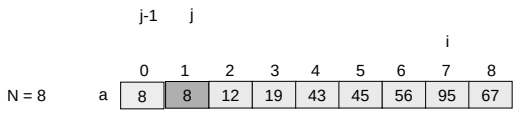
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




32

## Inserção Direta

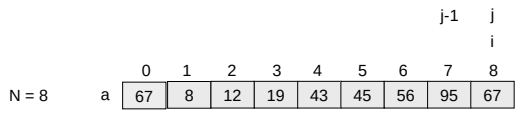
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




33

## Inserção Direta

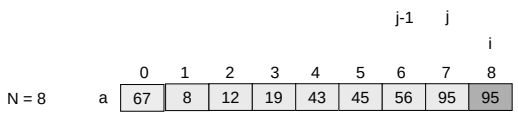
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




34

## Inserção Direta

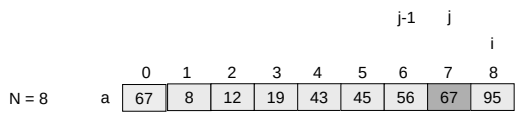
```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




35

## Inserção Direta

```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```




36

## Inserção Direta

```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| N = 8 | a | 67 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |

Vetor ordenado



## Inserção Direta: Análise

- O número **Ci** de comparações das chaves no **i**-ésimo passo é de, no máximo **i**; no mínimo 1 e admitindo-se que todas as **N** chaves sejam igualmente prováveis, em média $\frac{1}{i}\sum_{j=1}^{i}1=\frac{i+1}{2}$
- O número Mi de movimentos é (**Ci**-1+3), incluindo a sentinela
- O número mínimo ocorre se os elementos já estiverem, inicialmente, ordenados
- O pior caso ocorre se eles estiverem, inicialmente, em ordem reversa. Neste contexto, o algoritmo exibe um comportamento natural
- O método é estável



## Inserção Direta: Análise

```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```

$$C_{\text{mín}}=\sum_{i=2}^{N}1=N-1=O(N) \qquad M_{\text{mín}}=\sum_{i=2}^{N}1+2=3(N-1)=O(N)$$

$$C_{\text{méd}}=\sum_{i=2}^{N}\frac{i+1}{2}=\frac{N^2+3N-4}{4}=O(N^2) \qquad M_{\text{méd}}=\sum_{i=2}^{N}\frac{i+1}{2}+2=\frac{N^2+11N-12}{4}=O(N^2)$$

$$C_{\text{máx}}=\sum_{i=2}^{N}i=\frac{N^2+N-2}{2}=O(N^2) \qquad M_{\text{máx}}=\sum_{i=2}^{N}i+2=\frac{N^2+5N-6}{2}=O(N^2)$$



## Exercício

```
for(i = 2; i <= N; i++)
{ x = a[i];
  a[0] = x;
  j = i;
  while(x < a[j-1])
  { a[j] = a[j-1];
    j--;
  }
  a[j] = x;
}
```

- Utilizando o algoritmo de inserção direta, obtenha o número de comparações e movimentações em cada passo para os seguintes vetores
  - 45,56,12,43,95,19,8,67
  - 8,12,19,43,45,56,67,95
  - 95,67,56,45,43,19,12,8
  - 19,12,8,45,43,56,67,95



## Solução

| i | Ci | Mi | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 3 | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
| 3 | 3 | 5 | 12 | 45 | 56 | 43 | 95 | 19 | 8 | 67 |
| 4 | 3 | 5 | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |
| 5 | 1 | 3 | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |
| 6 | 5 | 7 | 12 | 19 | 43 | 45 | 56 | 95 | 8 | 67 |
| 7 | 7 | 9 | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |
| 8 | 2 | 4 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | 22 | 36 | | | | | | | | |

| i | Ci | Mi | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 3 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 4 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 5 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 6 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 7 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 8 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | 7 | 21 | | | | | | | | |

| i | Ci | Mi | 19 | 12 | 8 | 45 | 43 | 56 | 67 | 95 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 4 | 12 | 19 | 8 | 45 | 43 | 56 | 67 | 95 |
| 3 | 3 | 5 | 8 | 12 | 19 | 45 | 43 | 56 | 67 | 95 |
| 4 | 1 | 3 | 8 | 12 | 19 | 45 | 43 | 56 | 67 | 95 |
| 5 | 2 | 4 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 6 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 7 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 8 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | 11 | 25 | | | | | | | | |

| i | Ci | Mi | 95 | 67 | 56 | 45 | 43 | 19 | 12 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 4 | 67 | 95 | 56 | 45 | 43 | 19 | 12 | 8 |
| 3 | 3 | 5 | 56 | 67 | 95 | 45 | 43 | 19 | 12 | 8 |
| 4 | 4 | 6 | 45 | 56 | 67 | 95 | 43 | 19 | 12 | 8 |
| 5 | 5 | 7 | 43 | 45 | 56 | 67 | 95 | 19 | 12 | 8 |
| 6 | 6 | 8 | 19 | 43 | 45 | 56 | 67 | 95 | 12 | 8 |
| 7 | 7 | 9 | 12 | 19 | 43 | 45 | 56 | 67 | 95 | 8 |
| 8 | 8 | 10 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | 35 | 49 | | | | | | | | |



## Inserção Binária

- O algoritmo de inserção direta pode ser aperfeiçoado observando-se que a seqüência destino a[1], a[2], ..., a[i-1], na qual deve ser inserido o elemento **x**, já está ordenada
- Assim, pode-se utilizar um método mais rápido para determinar o ponto correto de inserção
- A escolha óbvia é a busca binária, que divide a seqüência destino no seu ponto central, continuando a divisão até encontrar o ponto correto de inserção

## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

| | m | i | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| a | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
| | L | R | | | | | | |



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

| | m | i | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| a | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
| | | R | | | | | | |
| | | L | | | | | | |



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

| | | m | i | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| a | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
| | L | | R | | | | | |



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

| | | m | | i | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| a | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
| | | L | R | | | | | |



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

| | | | j | | | | | |
|---|---|---|---|---|---|---|---|---|
| | m | | i | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| a | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
| | L | | | | | | | |
| | R | | | | | | | |



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

| | | | j | | | | | |
|---|---|---|---|---|---|---|---|---|
| | m | | i | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| a | 45 | 56 | 56 | 43 | 95 | 19 | 8 | 67 |
| | L | | | | | | | |
| | R | | | | | | | |

## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

j

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 45 | 45 | 56 | 43 | 95 | 19 | 8 | 67 |

L

R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 45 | 56 | 43 | 95 | 19 | 8 | 67 |

L

R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 45 | 56 | 43 | 95 | 19 | 8 | 67 |

L R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 45 | 56 | 43 | 95 | 19 | 8 | 67 |

L R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

j

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 45 | 56 | 43 | 95 | 19 | 8 | 67 |

R

L



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

j

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 45 | 56 | 56 | 95 | 19 | 8 | 67 |

R

L

## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 45 | 45 | 56 | 95 | 19 | 8 | 67 |

R

L



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |

R

L



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |

L R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |

L R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

j

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |

R

L



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |

R

L

## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

| | | m | | | | i | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| a | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |
| | L | | | | | R | | |

61

## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

| | | m | | | | i | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| a | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |
| | L | | R | | | | | |

62

## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

| | m | | | | | i | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| a | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |
| | L | R | | | | | | |

63

## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

| | | | | | | i | j | |
|---|---|---|---|---|---|---|---|---|
| | m | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| a | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |
| | | R | | | | | | |
| | | L | | | | | | |

64

## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

| | | | | | | i | j | |
|---|---|---|---|---|---|---|---|---|
| | m | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| a | 12 | 43 | 45 | 56 | 95 | 95 | 8 | 67 |
| | | R | | | | | | |
| | | L | | | | | | |

65

## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

| | | | | | | i | j | |
|---|---|---|---|---|---|---|---|---|
| | m | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| a | 12 | 43 | 45 | 56 | 56 | 95 | 8 | 67 |
| | | R | | | | | | |
| | | L | | | | | | |

66

## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

j

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 43 | 45 | 45 | 56 | 95 | 8 | 67 |

R
L



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 43 | 43 | 45 | 56 | 95 | 8 | 67 |

R
L



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 19 | 43 | 45 | 56 | 95 | 8 | 67 |

R
L



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 19 | 43 | 45 | 56 | 95 | 8 | 67 |

L R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 19 | 43 | 45 | 56 | 95 | 8 | 67 |

L R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 19 | 43 | 45 | 56 | 95 | 8 | 67 |

L R

## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

j

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 19 | 43 | 45 | 56 | 95 | 8 | 67 |

L

R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

j

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 19 | 43 | 45 | 56 | 95 | 95 | 67 |

L

R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

j

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 19 | 43 | 45 | 56 | 56 | 95 | 67 |

L

R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

j

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 19 | 43 | 45 | 45 | 56 | 95 | 67 |

L

R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

j

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 19 | 43 | 43 | 45 | 56 | 95 | 67 |

L

R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

j

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 19 | 19 | 43 | 45 | 56 | 95 | 67 |

L

R

## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 12 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |

L

R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |

L

R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |

L R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |

L R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |

L R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

j

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |

L

R

## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 95 |

L

R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

m i

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |

L

R



## Inserção Binária

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

N = 8

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |

Vetor ordenado



## Inserção Binária: Análise

- A posição correta para a inserção é encontrada quando **L** = **R**. Assim, o intervalo de busca ao final do algoritmo deve ser de comprimento unitário e isso significa que a operação de bissecção deverá ser aplicada log(i) vezes sobre um intervalo de comprimento **i**. Assim:

$$C = \sum_{i=1}^{N} \lceil \log_2(i) \rceil \cong \int_1^N \log_2(x)dx = N(\log_2(N) - c) + c = O(N \log_2 N)$$

- onde c = $\log_2(e)$ = 1/ln(2) = 1.44269...
- O número de comparações é independente da ordem inicial dos elementos
  - Entretanto, devido ao truncamento inerente à operação de divisão envolvida na bissecção do intervalo de busca, o número exato de comparações necessárias para a ordenação de **i** elementos pode ser até uma unidade maior que o esperado
  - Essa diferença é tal que as posições de inserção próximas da extremidade **superior** do vetor são, em média, localizadas um pouco mais rapidamente do que as que estão no outro extremo, favorecendo casos em que os elementos originais estão em ordem
  - O algoritmo exibe um **comportamento natural** (se os elementos estiverem em ordem será necessário o mínimo de comparações, mas se estiverem em ordem reversa, o número de comparações será máximo)



## Inserção Binária: Análise

- A melhoria obtida é referente apenas ao número de comparações, mas não ao número de movimentações
- Como, em geral, mover os elementos consome mais tempo do que comparar duas chaves, a melhoria obtida não é de modo algum drástica: o termo importante **M** é ainda da ordem de $\mathbf{N^2}$
- Esse método mostra que uma "melhoria óbvia", em geral, possui conseqüências menos drásticas do que se pode pensar à primeira vista
- No geral, a ordenação por inserção não parece ser um método adequado para uso em computadores digitais, pois a inserção de um elemento com o subseqüente deslocamento dos demais é anti-econômico
- Resultados melhores podem se obtidos nos métodos nos quais só ocorram movimentações de elementos unitários, o que nos leva à ordenação por seleção



## Exercício

```
for(i = 2; i <= N; i++)
{ x = a[i];
  L = 1;
  R = i;
  while(L < R)
  { m = (L + R) / 2;
    if(a[m] <= x)
      L = m + 1;
    else
      R = m;
  }
  for(j = i; j > R; j--)
    a[j] = a[j-1];
  a[R] = x;
}
```

- Utilizando o algoritmo de inserção binária, obtenha o número de comparações e movimentações em cada passo para os seguintes vetores
  - 45,56,12,43,95,19,8,67
  - 8,12,19,43,45,56,67,95
  - 95,67,56,45,43,19,12,8
  - 19,12,8,45,43,56,67,95

# Solução

| i | Ci | Mi | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 2 | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
| 3 | 2 | 4 | 12 | 45 | 56 | 43 | 95 | 19 | 8 | 67 |
| 4 | 2 | 4 | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |
| 5 | 2 | 2 | 12 | 43 | 45 | 56 | 95 | 19 | 8 | 67 |
| 6 | 3 | 6 | 12 | 19 | 43 | 45 | 56 | 95 | 8 | 67 |
| 7 | 3 | 8 | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |
| 8 | 3 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | 16 | 29 | | | | | | | | |

| i | Ci | Mi | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 2 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 3 | 1 | 2 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 4 | 2 | 2 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 5 | 2 | 2 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 6 | 2 | 2 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 7 | 2 | 2 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 8 | 3 | 2 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | | 13 | 14 | | | | | | | |

| i | Ci | Mi | 19 | 12 | 8 | 45 | 43 | 56 | 67 | 95 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 3 | 12 | 19 | 8 | 45 | 43 | 56 | 67 | 95 |
| 3 | 2 | 4 | 8 | 12 | 19 | 45 | 43 | 56 | 67 | 95 |
| 4 | 2 | 2 | 8 | 12 | 19 | 45 | 43 | 56 | 67 | 95 |
| 5 | 2 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 6 | 2 | 2 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 7 | 2 | 2 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 8 | 3 | 2 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | 14 | 18 | | | | | | | | |

| i | Ci | Mi | 95 | 67 | 56 | 45 | 43 | 19 | 12 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 3 | 67 | 95 | 56 | 45 | 43 | 19 | 12 | 8 |
| 3 | 2 | 4 | 56 | 67 | 95 | 45 | 43 | 19 | 12 | 8 |
| 4 | 2 | 5 | 45 | 56 | 67 | 95 | 43 | 19 | 12 | 8 |
| 5 | 3 | 6 | 43 | 45 | 56 | 67 | 95 | 19 | 12 | 8 |
| 6 | 3 | 7 | 19 | 43 | 45 | 56 | 67 | 95 | 12 | 8 |
| 7 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 | 8 |
| 8 | 3 | 9 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | | 17 | 42 | | | | | | | |



# Seleção Direta

1. Selecionar o elemento que apresenta a chave de menor valor;
2. Trocá-lo com o primeiro elemento do vetor;
3. Repetir estas operações, envolvendo agora apenas os **N**-1 elementos restantes, depois os **N**-2 elementos, etc., até restar um só elemento, o maior deles.



# Seleção Direta

1. Selecionar o elemento que apresenta a chave de menor valor;
2. Trocá-lo com o primeiro elemento do vetor;
3. Repetir estas operações, envolvendo agora apenas os **N**-1 elementos restantes, depois os **N**-2 elementos, etc., até restar um só elemento, o maior deles.

```
for(i = 1; i <= N-1; i++)
{ indice_menor = índice do menor elemento de
    a[i], a[i+1], ..., a[N];
  trocar a[i] com a[indice_menor];
}
```



# Seleção Direta

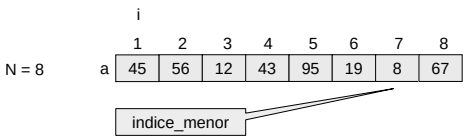
```
for(i = 1; i <= N-1; i++)
{ indice_menor = índice do menor elemento de
  a[i], a[i+1], ..., a[N];
  trocar a[i] com a[indice_menor];
}
```

| Vetor inicial | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
|---|---|---|---|---|---|---|---|---|
| i = 1 | 8 | 56 | 12 | 43 | 95 | 19 | 45 | 67 |
| i = 2 | 8 | 12 | 56 | 43 | 95 | 19 | 45 | 67 |
| i = 3 | 8 | 12 | 19 | 43 | 95 | 56 | 45 | 67 |
| i = 4 | 8 | 12 | 19 | 43 | 95 | 56 | 45 | 67 |
| i = 5 | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |
| i = 6 | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |
| i = 7 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |



# Seleção Direta

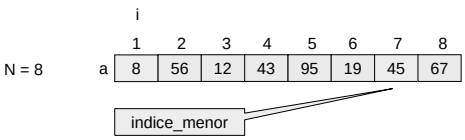
```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```





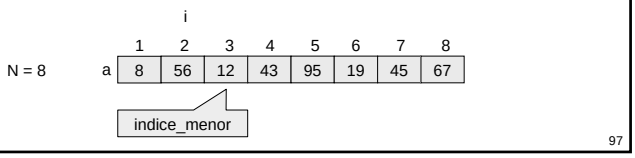

# Seleção Direta

```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```



## Seleção Direta

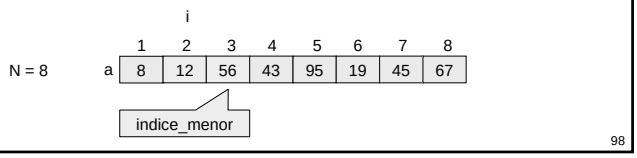
```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```






## Seleção Direta

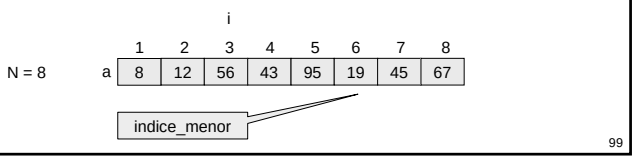
```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```






## Seleção Direta

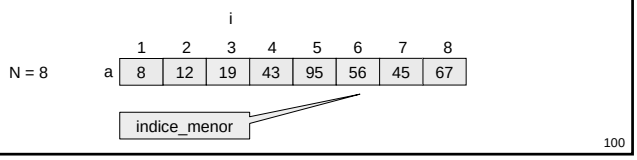
```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```






## Seleção Direta

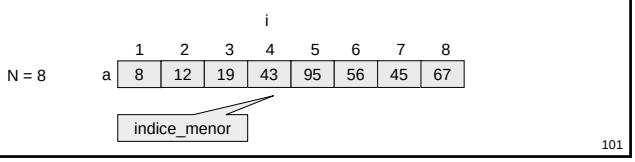
```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```






## Seleção Direta

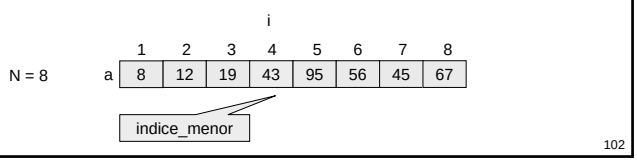
```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```





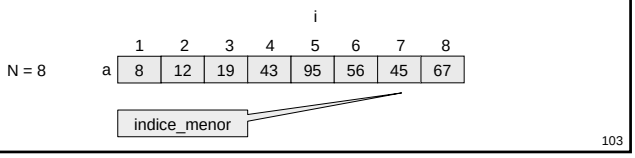

## Seleção Direta

```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```



## Seleção Direta

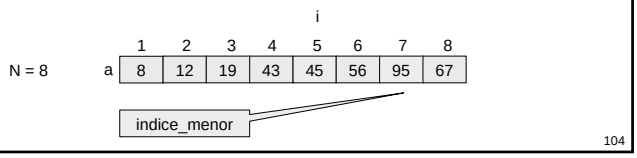
```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```






## Seleção Direta

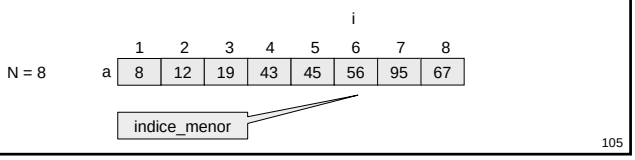
```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```






## Seleção Direta

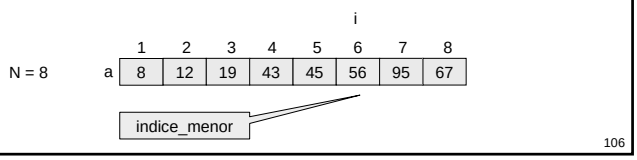
```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```






## Seleção Direta

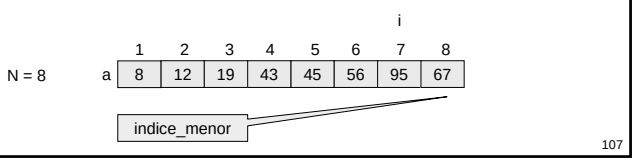
```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```






## Seleção Direta

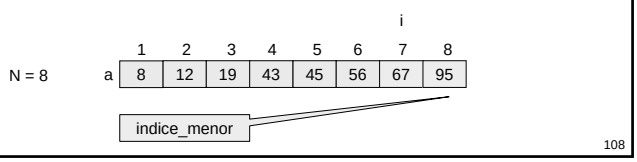
```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```






## Seleção Direta

```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```



# Seleção Direta

```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```

N = 8

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| a | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |

Vetor ordenado



# Seleção Direta: Análise

- O número **C** de comparações das chaves é independente da ordem inicial das mesmas
- Assim, esse método apresenta comportamento menos natural que o da inserção direta

$$C = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} 1 = \sum_{i=1}^{N-1} N - i = \frac{N^2 - N}{2} = O(N^2)$$

$$M = \sum_{i=1}^{N-1} 3 = 3(N-1) = O(N)$$



# Exercício

- Utilizando o algoritmo de seleção direta, obtenha o número de comparações e movimentações em cada passo para os seguintes vetores
  - 45,56,12,43,95,19,8,67
  - 8,12,19,43,45,56,67,95
  - 95,67,56,45,43,19,12,8
  - 19,12,8,45,43,56,67,95

```
for(i = 1; i <= N-1; i++)
{ indice_menor = i;
  for(j = i+1; j <= N; j++)
    if(a[j] < a[indice_menor])
      indice_menor = j;
  x = a[i];
  a[i] = a[indice_menor];
  a[indice_menor] = x;
}
```



# Solução

| i | Ci | Mi | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 3 | 8 | 56 | 12 | 43 | 95 | 19 | 45 | 67 |
| 2 | 6 | 3 | 8 | 12 | 56 | 43 | 95 | 19 | 45 | 67 |
| 3 | 5 | 3 | 8 | 12 | 19 | 43 | 95 | 56 | 45 | 67 |
| 4 | 4 | 3 | 8 | 12 | 19 | 43 | 95 | 56 | 45 | 67 |
| 5 | 3 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |
| 6 | 2 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |
| 7 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | 28 | 21 | | | | | | | | |

| i | Ci | Mi | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 2 | 6 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 3 | 5 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 4 | 4 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 5 | 3 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 6 | 2 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 7 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | 28 | 21 | | | | | | | | |

| i | Ci | Mi | 95 | 67 | 56 | 45 | 43 | 19 | 12 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 3 | 8 | 67 | 56 | 45 | 43 | 19 | 12 | 95 |
| 2 | 6 | 3 | 8 | 12 | 56 | 45 | 43 | 19 | 67 | 95 |
| 3 | 5 | 3 | 8 | 12 | 19 | 45 | 43 | 56 | 67 | 95 |
| 4 | 4 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 5 | 3 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 6 | 2 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 7 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | 28 | 21 | | | | | | | | |

| i | Ci | Mi | 19 | 12 | 8 | 45 | 43 | 56 | 67 | 95 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 3 | 8 | 12 | 19 | 45 | 43 | 56 | 67 | 95 |
| 2 | 6 | 3 | 8 | 12 | 19 | 45 | 43 | 56 | 67 | 95 |
| 3 | 5 | 3 | 8 | 12 | 19 | 45 | 43 | 56 | 67 | 95 |
| 4 | 4 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 5 | 3 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 6 | 2 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 7 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | 28 | 21 | | | | | | | | |



# Bubblesort

- É um método em que a permutação entre dois elementos é a principal característica do processo
- Como no método de seleção direta, efetuam-se varreduras repetidas sobre o vetor, deslocando-se, a cada passo para a sua extremidade esquerda, o menor dos elementos do conjunto que restou
- Se o vetor for visto na posição vertical ao invés da horizontal, os elementos podem ser comparados a bolhas em um tanque de água, com densidades proporcionais ao valor das respectivas chaves
- Assim, cada varredura efetuada sobre o vetor resulta na ascensão de uma bolha para o seu nível apropriado, de acordo com sua densidade



# Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | a |
|---|---|
| 1 | 45 |
| 2 | 56 |
| 3 | 12 |
| 4 | 43 |
| 5 | 95 |
| 6 | 19 |
| 7 | 8 |
| 8 | 67 |

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| i | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 43 | |
| | 5 | 95 | |
| | 6 | 19 | |
| | 7 | 8 | j-1 |
| | 8 | 67 | j |

115

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| i | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 43 | |
| | 5 | 95 | |
| | 6 | 19 | j-1 |
| | 7 | 8 | j |
| | 8 | 67 | |

116

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| i | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 43 | |
| | 5 | 95 | |
| | 6 | 8 | j-1 |
| | 7 | 19 | j |
| | 8 | 67 | |

117

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| i | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 43 | |
| | 5 | 95 | j-1 |
| | 6 | 8 | j |
| | 7 | 19 | |
| | 8 | 67 | |

118

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| i | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 43 | |
| | 5 | 8 | j-1 |
| | 6 | 95 | j |
| | 7 | 19 | |
| | 8 | 67 | |

119

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| i | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 43 | j-1 |
| | 5 | 8 | j |
| | 6 | 95 | |
| | 7 | 19 | |
| | 8 | 67 | |

120

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| i | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 8 | j-1 |
| | 5 | 43 | j |
| | 6 | 95 | |
| | 7 | 19 | |
| | 8 | 67 | |

121

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| i | 2 | 56 | |
| | 3 | 12 | j-1 |
| | 4 | 8 | j |
| | 5 | 43 | |
| | 6 | 95 | |
| | 7 | 19 | |
| | 8 | 67 | |

122

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| i | 2 | 56 | |
| | 3 | 8 | j-1 |
| | 4 | 12 | j |
| | 5 | 43 | |
| | 6 | 95 | |
| | 7 | 19 | |
| | 8 | 67 | |

123

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| i | 2 | 56 | j-1 |
| | 3 | 8 | j |
| | 4 | 12 | |
| | 5 | 43 | |
| | 6 | 95 | |
| | 7 | 19 | |
| | 8 | 67 | |

124

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| i | 2 | 8 | j-1 |
| | 3 | 56 | j |
| | 4 | 12 | |
| | 5 | 43 | |
| | 6 | 95 | |
| | 7 | 19 | |
| | 8 | 67 | |

125

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | j-1 |
| i | 2 | 8 | j |
| | 3 | 56 | |
| | 4 | 12 | |
| | 5 | 43 | |
| | 6 | 95 | |
| | 7 | 19 | |
| | 8 | 67 | |

126

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

| | | N = 8 a | |
|---|---|---|---|
| | 1 | 8 | j-1 |
| i | 2 | 45 | j |
| | 3 | 56 | |
| | 4 | 12 | |
| | 5 | 43 | |
| | 6 | 95 | |
| | 7 | 19 | |
| | 8 | 67 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

| | | N = 8 a |
|---|---|---|
| | 1 | 8 |
| i | 2 | 45 |
| | 3 | 56 |
| | 4 | 12 |
| | 5 | 43 |
| | 6 | 95 |
| | 7 | 19 |
| | 8 | 67 |

Término da primeira passagem (i=2). Note que o elemento 8, mais "leve" (menor valor) encontra-se na extremidade superior do vetor (ou extremidade esquerda se visto na horizontal)



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

| | | N = 8 a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| i | 3 | 56 | |
| | 4 | 12 | |
| | 5 | 43 | |
| | 6 | 95 | |
| | 7 | 19 | j-1 |
| | 8 | 67 | j |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

| | | N = 8 a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| i | 3 | 56 | |
| | 4 | 12 | |
| | 5 | 43 | |
| | 6 | 95 | j-1 |
| | 7 | 19 | j |
| | 8 | 67 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

| | | N = 8 a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| i | 3 | 56 | |
| | 4 | 12 | |
| | 5 | 43 | |
| | 6 | 19 | j-1 |
| | 7 | 95 | j |
| | 8 | 67 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

| | | N = 8 a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| i | 3 | 56 | |
| | 4 | 12 | |
| | 5 | 43 | j-1 |
| | 6 | 19 | j |
| | 7 | 95 | |
| | 8 | 67 | |

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| i | 3 | 56 | |
| | 4 | 12 | |
| | 5 | 19 | j-1 |
| | 6 | 43 | j |
| | 7 | 95 | |
| | 8 | 67 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| i | 3 | 56 | |
| | 4 | 12 | j-1 |
| | 5 | 19 | j |
| | 6 | 43 | |
| | 7 | 95 | |
| | 8 | 67 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| i | 3 | 56 | j-1 |
| | 4 | 12 | j |
| | 5 | 19 | |
| | 6 | 43 | |
| | 7 | 95 | |
| | 8 | 67 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| i | 3 | 12 | j-1 |
| | 4 | 56 | j |
| | 5 | 19 | |
| | 6 | 43 | |
| | 7 | 95 | |
| | 8 | 67 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | j-1 |
| i | 3 | 12 | j |
| | 4 | 56 | |
| | 5 | 19 | |
| | 6 | 43 | |
| | 7 | 95 | |
| | 8 | 67 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | j-1 |
| i | 3 | 45 | j |
| | 4 | 56 | |
| | 5 | 19 | |
| | 6 | 43 | |
| | 7 | 95 | |
| | 8 | 67 | |

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

| N = 8 | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| i | 3 | 45 | |
| | 4 | 56 | |
| | 5 | 19 | |
| | 6 | 43 | |
| | 7 | 95 | |
| | 8 | 67 | |

Término da segunda passagem (i=3)



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

| N = 8 | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 45 | |
| i | 4 | 56 | |
| | 5 | 19 | |
| | 6 | 43 | |
| | 7 | 95 | j -1 |
| | 8 | 67 | j |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

| N = 8 | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 45 | |
| i | 4 | 56 | |
| | 5 | 19 | |
| | 6 | 43 | |
| | 7 | 67 | j -1 |
| | 8 | 95 | j |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

| N = 8 | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 45 | |
| i | 4 | 56 | |
| | 5 | 19 | |
| | 6 | 43 | j -1 |
| | 7 | 67 | j |
| | 8 | 95 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

| N = 8 | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 45 | |
| i | 4 | 56 | |
| | 5 | 19 | j -1 |
| | 6 | 43 | j |
| | 7 | 67 | |
| | 8 | 95 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

| N = 8 | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 45 | |
| i | 4 | 56 | j -1 |
| | 5 | 19 | j |
| | 6 | 43 | |
| | 7 | 67 | |
| | 8 | 95 | |

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 45 | |
| i | 4 | 19 | j -1 |
| | 5 | 56 | j |
| | 6 | 43 | |
| | 7 | 67 | |
| | 8 | 95 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 45 | j -1 |
| i | 4 | 19 | j |
| | 5 | 56 | |
| | 6 | 43 | |
| | 7 | 67 | |
| | 8 | 95 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | j -1 |
| i | 4 | 45 | j |
| | 5 | 56 | |
| | 6 | 43 | |
| | 7 | 67 | |
| | 8 | 95 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a |
|---|---|---|
| | 1 | 8 |
| | 2 | 12 |
| | 3 | 19 |
| i | 4 | 45 |
| | 5 | 56 |
| | 6 | 43 |
| | 7 | 67 |
| | 8 | 95 |

Término da passagem i=4



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | |
| | 4 | 45 | |
| i | 5 | 56 | |
| | 6 | 43 | |
| | 7 | 67 | j -1 |
| | 8 | 95 | j |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | |
| | 4 | 45 | |
| i | 5 | 56 | |
| | 6 | 43 | j -1 |
| | 7 | 67 | j |
| | 8 | 95 | |

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | |
| | 4 | 45 | |
| i | 5 | 56 | j-1 |
| | 6 | 43 | j |
| | 7 | 67 | |
| | 8 | 95 | |

151

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | |
| | 4 | 45 | |
| i | 5 | 43 | j-1 |
| | 6 | 56 | j |
| | 7 | 67 | |
| | 8 | 95 | |

152

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | |
| | 4 | 45 | j-1 |
| i | 5 | 43 | j |
| | 6 | 56 | |
| | 7 | 67 | |
| | 8 | 95 | |

153

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | |
| | 4 | 43 | j-1 |
| i | 5 | 45 | j |
| | 6 | 56 | |
| | 7 | 67 | |
| | 8 | 95 | |

154

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | |
| | 4 | 43 | |
| i | 5 | 45 | |
| | 6 | 56 | |
| | 7 | 67 | |
| | 8 | 95 | |

Término da passagem i=5

155

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | |
| | 4 | 43 | |
| | 5 | 45 | |
| i | 6 | 56 | |
| | 7 | 67 | j-1 |
| | 8 | 95 | j |

156

## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | |
| | 4 | 43 | |
| | 5 | 45 | |
| i | 6 | 56 | j-1 |
| | 7 | 67 | j |
| | 8 | 95 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | |
| | 4 | 43 | |
| | 5 | 45 | j-1 |
| i | 6 | 56 | j |
| | 7 | 67 | |
| | 8 | 95 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | |
| | 4 | 43 | |
| | 5 | 45 | |
| i | 6 | 56 | |
| | 7 | 67 | |
| | 8 | 95 | |

Término da passagem i=6.
Note que não houve permutação
de elementos nesta passagem.



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | |
| | 4 | 43 | |
| | 5 | 45 | |
| | 6 | 56 | |
| i | 7 | 67 | j-1 |
| | 8 | 95 | j |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | |
| | 3 | 19 | |
| | 4 | 43 | |
| | 5 | 45 | |
| | 6 | 56 | j-1 |
| i | 7 | 67 | j |
| | 8 | 95 | |



## Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | | a |
|---|---|---|
| | 1 | 8 |
| | 2 | 12 |
| | 3 | 19 |
| | 4 | 43 |
| | 5 | 45 |
| | 6 | 56 |
| i | 7 | 67 |
| | 8 | 95 |

Término da passagem i=7.
Note que não houve permutação
de elementos nesta passagem.

# Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | a | |
|---|---|---|
| 1 | 8 | |
| 2 | 12 | |
| 3 | 19 | |
| 4 | 43 | |
| 5 | 45 | |
| 6 | 56 | |
| 7 | 67 | j -1 |
| i 8 | 95 | j |



# Bubblesort

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

N = 8

| | a |
|---|---|
| 1 | 8 |
| 2 | 12 |
| 3 | 19 |
| 4 | 43 |
| 5 | 45 |
| 6 | 56 |
| 7 | 67 |
| i 8 | 95 |

Término da passagem i=8.
Note que não houve permutação de elementos nesta passagem.

Vetor ordenado



# Bubblesort: Análise

- Os números **C** de comparações das chaves e **M** de movimentos são

$$C = \sum_{i=2}^{N}\sum_{j=i}^{N} 1 = \sum_{i=2}^{N} N - i + 1 = \frac{N^2 - N}{2} = O(N^2)$$

$$M_{\text{mín}} = 0 = O(1) \qquad M_{\text{méd}} = \frac{3}{2}C = O(N^2) \qquad M_{\text{máx}} = 3C = O(N^2)$$



# Exercício

```
for(i = 2; i <= N; i++)
  for(j = N; j >= i; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
    }
```

- Utilizando o algoritmo de borbulhamento, obtenha o número de comparações e movimentações em cada passo para os seguintes vetores
  - 45,56,12,43,95,19,8,67
  - 8,12,19,43,45,56,67,95
  - 95,67,56,45,43,19,12,8
  - 19,12,8,45,43,56,67,95



# Solução

| i | Ci | Mi | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 7 | 18 | 8 | 45 | 56 | 12 | 43 | 95 | 19 | 67 |
| 3 | 6 | 12 | 8 | 12 | 45 | 56 | 19 | 43 | 95 | 67 |
| 4 | 5 | 9 | 8 | 12 | 19 | 45 | 56 | 43 | 67 | 95 |
| 5 | 4 | 6 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 6 | 3 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 7 | 2 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 8 | 1 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | | 28 | 45 | | | | | | | |

| i | Ci | Mi | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 7 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 3 | 6 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 4 | 5 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 5 | 4 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 6 | 3 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 7 | 2 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 8 | 1 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | | 28 | 0 | | | | | | | |

| i | Ci | Mi | 19 | 12 | 8 | 45 | 43 | 56 | 67 | 95 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 7 | 9 | 8 | 19 | 12 | 43 | 45 | 56 | 67 | 95 |
| 3 | 6 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 4 | 5 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 5 | 4 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 6 | 3 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 7 | 2 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 8 | 1 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | | 28 | 12 | | | | | | | |

| i | Ci | Mi | 95 | 67 | 56 | 45 | 43 | 19 | 12 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 7 | 21 | 8 | 95 | 67 | 56 | 45 | 43 | 19 | 12 |
| 3 | 6 | 18 | 8 | 12 | 95 | 67 | 56 | 45 | 43 | 19 |
| 4 | 5 | 15 | 8 | 12 | 19 | 95 | 67 | 56 | 45 | 43 |
| 5 | 4 | 12 | 8 | 12 | 19 | 43 | 95 | 67 | 56 | 45 |
| 6 | 3 | 9 | 8 | 12 | 19 | 43 | 45 | 95 | 67 | 56 |
| 7 | 2 | 6 | 8 | 12 | 19 | 43 | 45 | 56 | 95 | 67 |
| 8 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | | 28 | 84 | | | | | | | |



# Bubblesort: Aperfeiçoamentos

- No vetor exemplo, pode-se observar que os três últimos passos do algoritmo não afetaram a ordem dos elementos do vetor, pois estes já se encontravam ordenados
- Uma técnica para melhorar o algoritmo Bubblesort consiste em manter uma indicação informando se houve ou não a ocorrência de uma permutação, para determinar antecipadamente o término do algoritmo
- Entretanto, mesmo essa melhoria pode ser por sua vez aperfeiçoada, guardando-se não a simples informação da ocorrência de uma permutação, mas a posição (índice) do vetor em que ocorreu a última permutação realizada

# Bubblesort: Aperfeiçoamentos

- Assimetria: uma bolha única, colocada de modo incorreto, na extremidade mais densa do vetor, cujos demais elementos estejam ordenados, será posicionada corretamente em um único passo, mas um elemento incorretamente posicionado na extremidade menos densa irá deslocar-se de apenas uma posição por vez em direção à sua correta posição. Por exemplo, o vetor

  12 19 43 45 56 67 95 8

- é ordenado pelo método Bubblesort aperfeiçoado em um único passo, mas o vetor

  95 8 12 19 43 45 56 67

- requer sete passos para a sua ordenação. Esta assimetria não natural sugere uma terceira melhoria: alternar a direção dos sucessivos passos de ordenação: ***shakersort***

169

# Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a |
|---|---|---|
| | 1 | 45 |
| | 2 | 56 |
| | 3 | 12 |
| | 4 | 43 |
| | 5 | 95 |
| | 6 | 19 |
| | 7 | 8 |
| | 8 | 67 |

170

# Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 43 | |
| | 5 | 95 | |
| | 6 | 19 | |
| | 7 | 8 | j -1 |
| k R | 8 | 67 | j |

171

# Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| L | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 43 | |
| | 5 | 95 | |
| | 6 | 19 | j -1 |
| | 7 | 8 | j |
| k R | 8 | 67 | |

172

# Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| L | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 43 | |
| | 5 | 95 | |
| | 6 | 8 | j -1 |
| k | 7 | 19 | j |
| R | 8 | 67 | |

173

# Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| L | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 43 | |
| | 5 | 95 | j -1 |
| | 6 | 8 | j |
| k | 7 | 19 | |
| R | 8 | 67 | |

174

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| L | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 43 | |
| | 5 | 8 | j -1 |
| k | 6 | 95 | j |
| | 7 | 19 | |
| R | 8 | 67 | |

175

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| L | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 43 | j -1 |
| | 5 | 8 | j |
| k | 6 | 95 | |
| | 7 | 19 | |
| R | 8 | 67 | |

176

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| L | 2 | 56 | |
| | 3 | 12 | |
| | 4 | 8 | j -1 |
| k | 5 | 43 | j |
| | 6 | 95 | |
| | 7 | 19 | |
| R | 8 | 67 | |

177

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| L | 2 | 56 | |
| | 3 | 12 | j -1 |
| | 4 | 8 | j |
| k | 5 | 43 | |
| | 6 | 95 | |
| | 7 | 19 | |
| R | 8 | 67 | |

178

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| L | 2 | 56 | |
| | 3 | 8 | j -1 |
| k | 4 | 12 | j |
| | 5 | 43 | |
| | 6 | 95 | |
| | 7 | 19 | |
| R | 8 | 67 | |

179

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 45 | |
| L | 2 | 56 | j -1 |
| | 3 | 8 | j |
| k | 4 | 12 | |
| | 5 | 43 | |
| | 6 | 95 | |
| | 7 | 19 | |
| R | 8 | 67 | |

180

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | | 1 | 45 | |
| | L | 2 | 8 | j -1 |
| k | | 3 | 56 | j |
| | | 4 | 12 | |
| | | 5 | 43 | |
| | | 6 | 95 | |
| | | 7 | 19 | |
| | R | 8 | 67 | |

181

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | | 1 | 45 | j -1 |
| | L | 2 | 8 | j |
| k | | 3 | 56 | |
| | | 4 | 12 | |
| | | 5 | 43 | |
| | | 6 | 95 | |
| | | 7 | 19 | |
| | R | 8 | 67 | |

182

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | | 1 | 8 | j -1 |
| k | L | 2 | 45 | j |
| | | 3 | 56 | |
| | | 4 | 12 | |
| | | 5 | 43 | |
| | | 6 | 95 | |
| | | 7 | 19 | |
| | R | 8 | 67 | |

183

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a |
|---|---|---|---|
| | | 1 | 8 |
| k | L | 2 | 45 |
| | | 3 | 56 |
| | | 4 | 12 |
| | | 5 | 43 |
| | | 6 | 95 |
| | | 7 | 19 |
| | R | 8 | 67 |

Término da 1ª "subida"

184

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | | 1 | 8 | |
| k | | 2 | 45 | j -1 |
| | L | 3 | 56 | j |
| | | 4 | 12 | |
| | | 5 | 43 | |
| | | 6 | 95 | |
| | | 7 | 19 | |
| | R | 8 | 67 | |

Início da 1ª "descida"

185

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | | 1 | 8 | |
| k | | 2 | 45 | |
| | L | 3 | 56 | j -1 |
| | | 4 | 12 | j |
| | | 5 | 43 | |
| | | 6 | 95 | |
| | | 7 | 19 | |
| | R | 8 | 67 | |

186

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| N = 8 | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| L | 3 | 12 | j-1 |
| k | 4 | 56 | j |
| | 5 | 43 | |
| | 6 | 95 | |
| | 7 | 19 | |
| R | 8 | 67 | |

187

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| N = 8 | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| L | 3 | 12 | |
| k | 4 | 56 | j-1 |
| | 5 | 43 | j |
| | 6 | 95 | |
| | 7 | 19 | |
| R | 8 | 67 | |

188

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| N = 8 | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| L | 3 | 12 | |
| | 4 | 43 | j-1 |
| k | 5 | 56 | j |
| | 6 | 95 | |
| | 7 | 19 | |
| R | 8 | 67 | |

189

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| N = 8 | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| L | 3 | 12 | |
| | 4 | 43 | |
| k | 5 | 56 | j-1 |
| | 6 | 95 | j |
| | 7 | 19 | |
| R | 8 | 67 | |

190

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| N = 8 | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| L | 3 | 12 | |
| | 4 | 43 | |
| | 5 | 56 | |
| k | 6 | 95 | j-1 |
| | 7 | 19 | j |
| R | 8 | 67 | |

191

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| N = 8 | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| L | 3 | 12 | |
| | 4 | 43 | |
| | 5 | 56 | |
| | 6 | 19 | j-1 |
| k | 7 | 95 | j |
| R | 8 | 67 | |

192

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | | 1 | 8 | |
| | | 2 | 45 | |
| | L | 3 | 12 | |
| | | 4 | 43 | |
| | | 5 | 56 | |
| | | 6 | 19 | |
| k | | 7 | 95 | j -1 |
| | R | 8 | 67 | j |

193

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | | 1 | 8 | |
| | | 2 | 45 | |
| | L | 3 | 12 | |
| | | 4 | 43 | |
| | | 5 | 56 | |
| | | 6 | 19 | |
| | | 7 | 67 | j -1 |
| k | R | 8 | 95 | j |

194

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a |
|---|---|---|---|
| | | 1 | 8 |
| | | 2 | 45 |
| | L | 3 | 12 |
| | | 4 | 43 |
| | | 5 | 56 |
| | | 6 | 19 |
| | R | 7 | 67 |
| k | | 8 | 95 |

Término da 1ª "descida".
Término da primeira passagem
(subida e descida de bolhas)

195

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | | 1 | 8 | |
| | | 2 | 45 | |
| | L | 3 | 12 | |
| | | 4 | 43 | |
| | | 5 | 56 | |
| | | 6 | 19 | j -1 |
| | R | 7 | 67 | j |
| k | | 8 | 95 | |

Início da 2ª "subida", pois
L <= R

196

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | | 1 | 8 | |
| | | 2 | 45 | |
| | L | 3 | 12 | |
| | | 4 | 43 | |
| | | 5 | 56 | j -1 |
| | | 6 | 19 | j |
| | R | 7 | 67 | |
| k | | 8 | 95 | |

197

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | | 1 | 8 | |
| | | 2 | 45 | |
| | L | 3 | 12 | |
| | | 4 | 43 | |
| | | 5 | 19 | j -1 |
| k | | 6 | 56 | j |
| | R | 7 | 67 | |
| | | 8 | 95 | |

198

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| L | 3 | 12 | |
| | 4 | 43 | j -1 |
| | 5 | 19 | j |
| k | 6 | 56 | |
| R | 7 | 67 | |
| | 8 | 95 | |



## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| L | 3 | 12 | |
| | 4 | 19 | j -1 |
| k | 5 | 43 | j |
| | 6 | 56 | |
| R | 7 | 67 | |
| | 8 | 95 | |



## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | |
| L | 3 | 12 | j -1 |
| | 4 | 19 | j |
| k | 5 | 43 | |
| | 6 | 56 | |
| R | 7 | 67 | |
| | 8 | 95 | |



## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 45 | j -1 |
| L | 3 | 12 | j |
| | 4 | 19 | |
| k | 5 | 43 | |
| | 6 | 56 | |
| R | 7 | 67 | |
| | 8 | 95 | |



## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a | |
|---|---|---|---|
| | 1 | 8 | |
| | 2 | 12 | j -1 |
| k L | 3 | 45 | j |
| | 4 | 19 | |
| | 5 | 43 | |
| | 6 | 56 | |
| R | 7 | 67 | |
| | 8 | 95 | |



## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

N = 8

| | | a |
|---|---|---|
| | 1 | 8 |
| | 2 | 12 |
| k L | 3 | 45 |
| | 4 | 19 |
| | 5 | 43 |
| | 6 | 56 |
| R | 7 | 67 |
| | 8 | 95 |

Término da 2ª "subida"

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | 1 | 8 | | |
| | 2 | 12 | | |
| k | 3 | 45 | | j -1 |
| L | 4 | 19 | | j |
| | 5 | 43 | | |
| | 6 | 56 | | |
| R | 7 | 67 | | |
| | 8 | 95 | | |

Início da 2ª "descida"



## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | 1 | 8 | | |
| | 2 | 12 | | |
| | 3 | 19 | | j -1 |
| k L | 4 | 45 | | j |
| | 5 | 43 | | |
| | 6 | 56 | | |
| R | 7 | 67 | | |
| | 8 | 95 | | |



## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | 1 | 8 | | |
| | 2 | 12 | | |
| | 3 | 19 | | |
| k L | 4 | 45 | | j -1 |
| | 5 | 43 | | j |
| | 6 | 56 | | |
| R | 7 | 67 | | |
| | 8 | 95 | | |



## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | 1 | 8 | | |
| | 2 | 12 | | |
| | 3 | 19 | | |
| L | 4 | 43 | | j -1 |
| k | 5 | 45 | | j |
| | 6 | 56 | | |
| R | 7 | 67 | | |
| | 8 | 95 | | |



## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | 1 | 8 | | |
| | 2 | 12 | | |
| | 3 | 19 | | |
| L | 4 | 43 | | |
| k | 5 | 45 | | j -1 |
| | 6 | 56 | | j |
| R | 7 | 67 | | |
| | 8 | 95 | | |



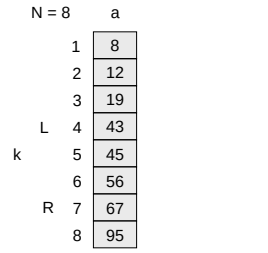
## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

| | | N = 8 | a | |
|---|---|---|---|---|
| | 1 | 8 | | |
| | 2 | 12 | | |
| | 3 | 19 | | |
| L | 4 | 43 | | |
| k | 5 | 45 | | |
| | 6 | 56 | | j -1 |
| R | 7 | 67 | | j |
| | 8 | 95 | | |

## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

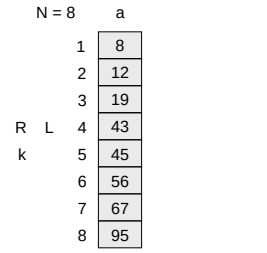



Término da 2ª "descida".



## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

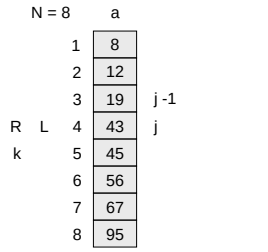



Início da 3ª "subida", pois
L <= R



## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

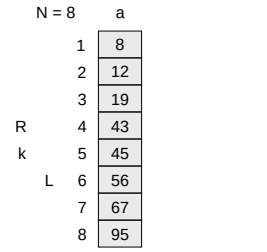



Término da 3ª "descida".



## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

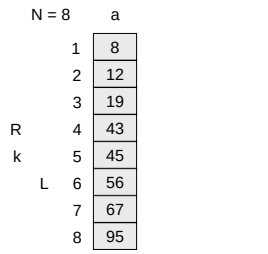



Início da 3ª "descida".



## Shakersort

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```




Término da 3ª "descida".
Note que o laço de descida não é executado nessa passagem.
Como L > R, Vetor Ordenado



## Shakersort: Análise

- A análise do Shakersort é complexa
- O número mínimo de comparações é $C_{mín}$=N-1
- O número médio de comparações é proporcional a $C_{méd}$=($N^2$-N(K+ln(N)))/2, onde *K* é uma constante
- Entretanto, nota-se que todas as melhorias não afetam o número de movimentações: elas apenas reduzem o número de testes redundantes
- Como a movimentação de dois elementos é uma operação mais onerosa do que a comparação de chaves, estas otimizações operam no algoritmo um ganho menor do que se poderia esperar

# Exercício

```
L = 2; R = N; k = N;
do
{ for(j = R; j >= L; j--)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  L = k + 1;
  for(j = L; j <= R; j++)
    if(a[j-1] > a[j])
    { x = a[j-1];
      a[j-1] = a[j];
      a[j] = x;
      k = j;
    }
  R = k - 1;
} while (L <= R);
```

- Utilizando o algoritmo de agitação, obtenha o número de comparações e movimentações em cada passo para os seguintes vetores
  - 45,56,12,43,95,19,8,67
  - 8,12,19,43,45,56,67,95
  - 95,67,56,45,43,19,12,8
  - 19,12,8,45,43,56,67,95



# Solução

| L | R | Ci | Mi | 45 | 56 | 12 | 43 | 95 | 19 | 8 | 67 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 13 | 30 | 8 | 45 | 12 | 43 | 56 | 19 | 67 | 95 |
| 4 | 4 | 9 | 15 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| 6 | 4 | 1 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | | 23 | 45 | | | | | | | | |

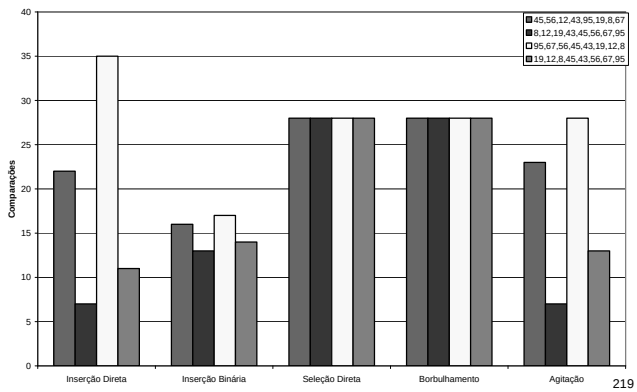
| L | R | Ci | Mi | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 7 | 7 | 0 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | | 7 | 0 | | | | | | | | |

| L | R | Ci | Mi | 19 | 12 | 8 | 45 | 43 | 56 | 67 | 95 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 2 | 13 | 12 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | | 13 | 12 | | | | | | | | |

| L | R | Ci | Mi | 95 | 67 | 56 | 45 | 43 | 19 | 12 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 7 | 13 | 39 | 8 | 67 | 56 | 45 | 43 | 19 | 12 | 95 |
| 4 | 6 | 9 | 27 | 8 | 12 | 56 | 45 | 43 | 19 | 67 | 95 |
| 5 | 5 | 5 | 15 | 8 | 12 | 19 | 45 | 43 | 56 | 67 | 95 |
| 6 | 4 | 1 | 3 | 8 | 12 | 19 | 43 | 45 | 56 | 67 | 95 |
| | | 28 | 84 | | | | | | | | |

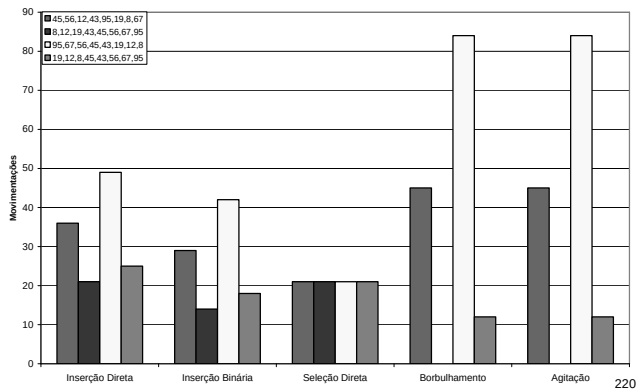


# Quadro Geral: Comparações






# Quadro Geral: Movimentações






# Resumo

- Em geral, a atividade de ordenação é o processo de rearranjo de um certo conjunto de objetos (elementos) de acordo com um critério (ordem) específico
- O objetivo da ordenação é facilitar a localização dos membros de um conjunto de objetos
- Nesta aula foram vistos alguns métodos de ordenação (diretos); entretanto existem muitos outros, cada um apresentando vantagens e desvantagens em relação aos demais
- Cabe ao programador selecionar qual o método de ordenação mais adequado para cada aplicação