



ESTRUTURAS DE DADOS E ALGORITMOS

ALGORITMOS DE ORDENAÇÃO POR COMPARAÇÃO - I

Adalberto Cajueiro

Departamento de Sistemas e Computação

Universidade Federal de Campina Grande



CONTEXTO

Busca de **Passagens**

Origem:

cpv

Destino:

gru

Data de Ida:

Data de Volta:

Adultos:

1

Crianças (2-11):

Crianças

Preço

Horário

Lista

Galeria

Guias de Compras relacionados

Produtos 1 - 50 de **859**

Comparar

Ordenar por:

Melhores produtos

Melhores produtos

Menor Preço

Maior Preço


Mais Ofertados

Mais Visitados

Menor Tempo Restante

em Vista

☐



Apple Iphone 8gb Versão 1.1.4 Desbloqueado Pronta Entrega !

Nova Versão 1.1.4 Desbloqueado E Menu Em Português. Sedex Grátis !!!


Qualif. vendedor: 3702 (99% +)

Localização: **Paraná**

Produto: **Novo**

Pontuação do produto: ★★★★★ [Ver opiniões](#)

☐



iPhone 8 Gigas Totalmente Desbloqueado Pague Em Até 12 X


Versão 1,1,4 Em Portugues Todas Operadoras

Qualif. vendedor: 6022 (97% +)

Localização: **São Paulo**

Produto: **Novo**

Pontuação do produto: ★★★★★ [Ver opiniões](#)

 **Compre Já!**

R\$ 1290.00

Ofertas: 471

Visitas: 53477

Finaliza em: 16d 10h

[Adicionar em Produtos em Vista](#)

ORDENACAO

- Uma das tarefas mais fundamentais e extensivamente usadas na computação:
 - Bancos de dados, compiladores, interpretadores, sistemas operacionais, etc.
- Ordenacao é o processo de arranjar um conjunto de informacoes semelhantes em ordem crescente ou decrescente
- Diversos algoritmos
 - Ordenacao por comparacao
 - Ordenacao em tempo linear

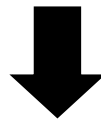
ALGORITMO IN-PLACE

- A ordenação é feita no **mesmo local** onde os dados são armazenados. No máximo precisa de uma memória extra (no máximo é constante)

ALGORITMO ESTÁVEL

- Um algoritmo de ordenação é estável se dados dois elementos R e S com a mesma chave, se R aparece antes de S na lista original, R aparecerá antes de S na lista ordenada final
- É importante quando tivermos dados satélites.
- Exemplo:

10(Carlos) 2(Débora) 7(João) 7(José)



2(Débora) 7(João) 7(José) 10(Carlos)



BUBBLE SORT

6



BUBBLE SORT

- O **bubble sort**, ou ordenação por flutuação (literalmente "por bolha"), é um algoritmo de ordenação dos mais simples. A ideia é percorrer o vetor diversas vezes, a cada passagem fazendo **flutuar para o topo o maior elemento da sequência**.
- As trocas acontecem como elemento subsequente
- Exemplo:

[8,4,7,1]	[4,7,1,8]
[4,8,7,1]	[4,1,7,8]
[4,7,8,1]	[4,1,7,8]
[4,7,1,8]	[1,4,7,8]
[4,7,1,8]	[1,4,7,8]

BUBBLE SORT

- Exercício:
 - Ordene a lista a seguir utilizando o bubble sort
 - 21, 23, 2, 34, 245, 33, 66

BUBBLE SORT

- Como seria o algoritmo do Bubble Sort?

```
procedure bubbleSort( A : list of sortable items )  
  n = length(A)  
  for each i in 1 to n – 1 inclusive {  
    for each j in 1 to n-i inclusive{  
      if (A[j] > A[j+1]){  
        swap(A[j], A[j+1])  
      }  
    }  
  }  
end procedure  
procedure swap(A[i],A[j])  
  int temp = A[i];  
  A[i] = A[j];  
  A[j] = temp;  
end procedure
```

BUBBLE SORT

- Qual o tempo do Bubble Sort no pior caso?

```
procedure bubbleSort( A : list of sortable items )
```

```
  n = length(A)
```

```
  for each i in 1 to n - 1 inclusive {
```

```
    for each j in 1 to n-i inclusive{
```

```
      if (A[j] > A[j+1]){
```

```
        swap(A[j], A[j+1])
```

```
      }
```

```
    }
```

```
  }
```

```
end procedure
```

```
procedure swap(A[i],A[j])
```

```
  int temp = A[i];
```

```
  A[i] = A[j];
```

```
  A[j] = temp;
```

```
end procedure
```

n repeticoes

(n - 1)

(n - 2)

(n - 3)

...

0

(n - 1)

(n - 2)

(n - 3)

...

(n - n)

$n^2 - S_n$

$n^2 - n(n+1)/2$

$(n^2 - 1)/2 = \Theta(n^2)$

BUBBLE SORT

- Qual o tempo do Bubble Sort no pior caso?

```
procedure bubbleSort( A : list of sortable items )
```

```
  n = length(A)
```

```
  for each i in 1 to n - 1 inclusive {
```

```
    for each j in 1 to n-i inclusive{
```

```
      if (A[j] > A[j+1]){
```

```
        swap(A[j], A[j+1])
```

```
      }
```

```
    }
```

```
  }
```

```
end procedure
```

```
procedure swap(A[i],A[j])
```

```
  int temp = A[i];
```

```
  A[i] = A[j];
```

```
  A[j] = temp;
```

```
end procedure
```

(n - 1)

(n - 2)

(n - 3)

...

0

n repeticoes

(n - 1)

(n - 2)

(n - 3)

...

(n - n)

$n^2 - S_n$

$n^2 - n(n+1)/2$

$(n^2 - 1)/2 = \Theta(n^2)$

O que acontece no melhor caso?

BUBBLE SORT

- O que esse algoritmo faz?

```
procedure XXXXXX( A : list of sortable items )  
  n = length(A)  
  for each i in 1 to n-1 inclusive {  
    for each j in n-1 to i inclusive{  
      if A[j-1] > A[j] {  
        swap(A[j-1], A[j])  
      }  
    }  
  }  
end procedure
```

BUBBLE SORT

- O que esse algoritmo faz?

```
procedure YYYYYYYY( A : list of sortable items )  
  swapped = true;  
  n = length(A)  
  while swapped {  
    swapped = false;  
    for each i in 1 to n-2 inclusive {  
      if(A[i] > A[i+1]){  
        swap(A[i],A[i+1])  
        swapped = true  
      }  
    }  
  }  
end procedure
```

CARACTERÍSTICAS

- Fácil de implementar
- Fácil de entender
- In place: não precisa de espaço extra
- Stable: preserva a ordem
- Pode ser útil para valores de n pequenos
- Número alto
 - comparações
 - Swap (trocas) $O(n^2)$
 - Muito ruim na prática

“Bubble sort seems to have
nothing to recommend it.”

[Donald Knuth]





SELECTION SORT

15



SELECTION SORT

- Lógica do **selection sort**:
 - Encontre o **menor elemento** da lista
 - **Troque** com o primeiro elemento da lista
 - Faça o processo **novamente** para o restante da lista, começando da próxima posição
- Exemplo:

[8,4,7,**1**]

[**1**,**4**,7,8]

[**1**,**4**,7,8]

[**1**,**4**,**7**,8]

[**1**,**4**,**7**,**8**]

[**1**,**4**,**7**,**8**]

SELECTION SORT

- Exercício:
 - Ordene a lista a seguir utilizando o selection sort
 - 21, 23, 2, 34, 245, 33, 66

SELECTION SORT

- Como seria o algoritmo do Selection Sort?

```
SELECTION-SORT(A)
  n = length(A)
  for i ← 1 to n-1 {
    min ← i
    for j ← (i + 1) to n {
      if A[j] < A[min] {
        min ← j
      }
    }
    swap (A[i], A[min])
  }
}
```

SELECTION SORT

- Qual o tempo do Selection Sort no pior caso?

```
SELECTION-SORT(A)
  n = length(A)
  for i ← 1 to n-1 {
    min ← i
    for j ← (i + 1) to n {
      if A[j] < A[min] {
        min ← j
      }
    }
    swap (A[i], A[min])
  }
}
```

SELECTION SORT

- Qual o tempo do Selection Sort no pior caso?

SELECTION-SORT(A)

n = length(A)

for i ← 1 to n-1 {

min ← i

for j ← (i + 1) to n {

if A[j] < A[min] {

min ← j

}

}

swap (A[i], A[min])

}

}

n repeticoes

(n - 1)

(n - 2)

(n - 3)

...

(n - n)

$n^2 - S_n$

$n^2 - n(n+1)/2$

$(n^2 - 1)/2 = \Theta(n^2)$

SELECTION SORT

- O que esse algoritmo faz?

XXXXXXXXXX-SORT(A)

 n = length(A)

 for i ← n downto 2 {

 max ← i

 for j ← 1 to i {

 if A[j] > A[max] {

 max ← j

 }

 }

 swap (A[i], A[max])

 }

}

CARACTERÍSTICAS

- Fácil de implementar
- Fácil de entender
- In place: não precisa de espaço extra
- Stable
 - Depende de como é feito o swap
- Pode ser útil para valores de n pequenos
- Melhor que o bubble sort (tem menos trocas) mas perde para o insertion sort
 - Menos swap (trocas)



INSERTION SORT

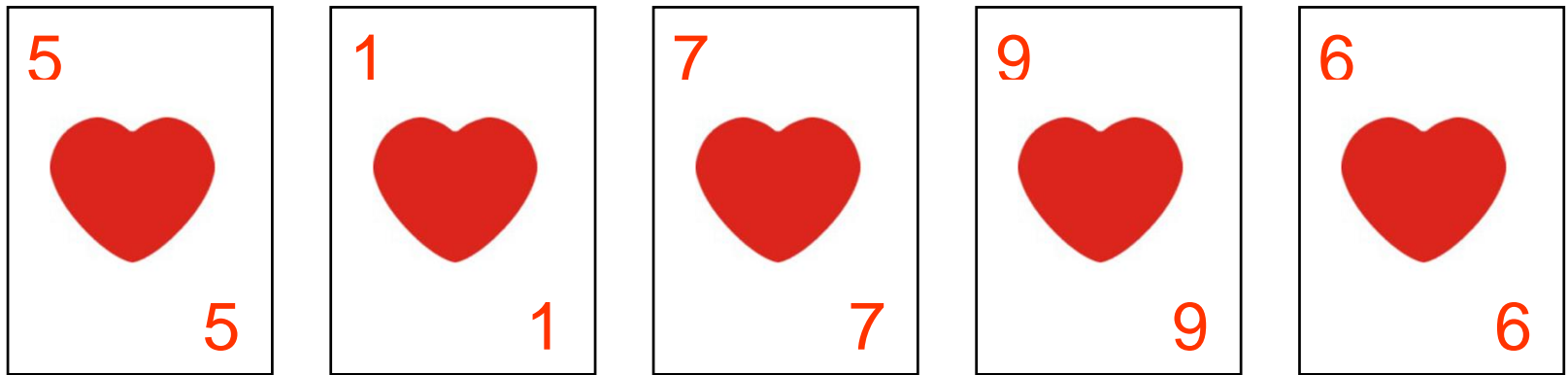
23



INSERTION SORT

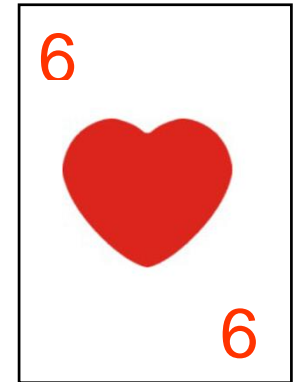
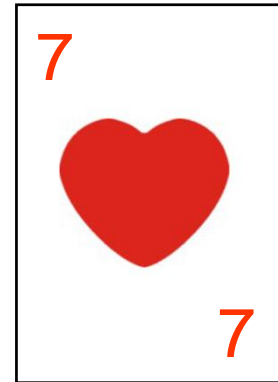
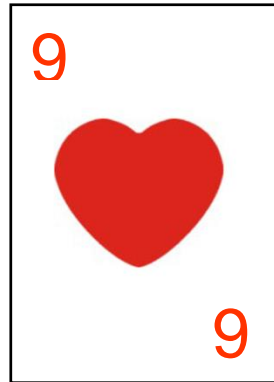
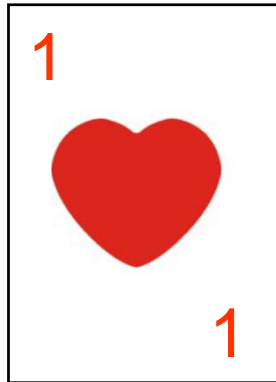
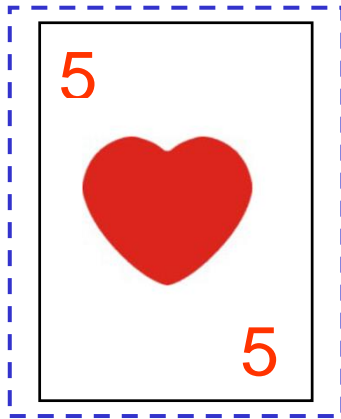
- Lógica do **insertion sort**: similar a ordenar um baralho com a seguinte idéia:
 - Pegue uma carta com a mão direita e coloque na mão esquerda
 - Faça isso para as demais cartas colocando-as de forma ordenada na mão esquerda.

INSERTION SORT

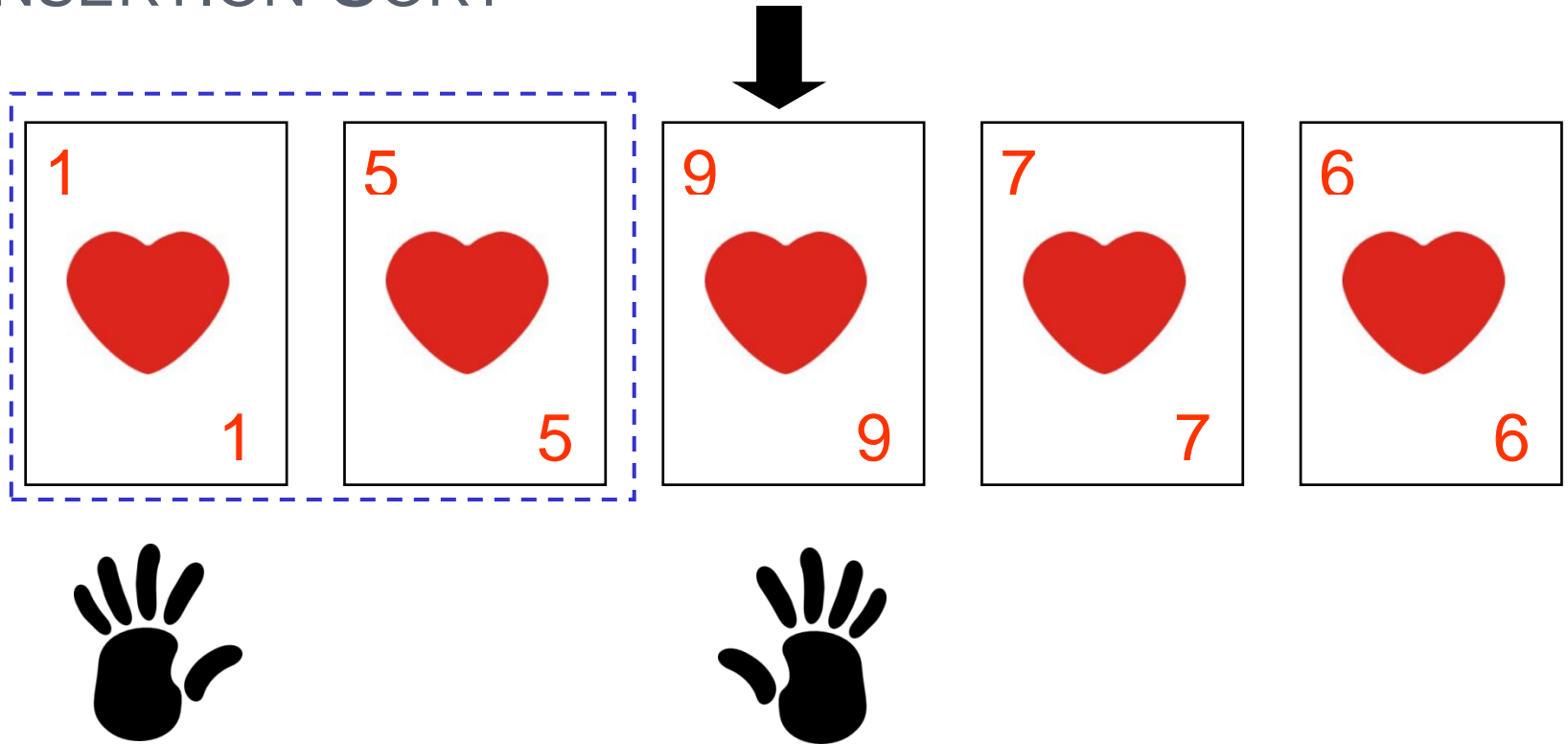


Como **ordenar** as cartas acima?

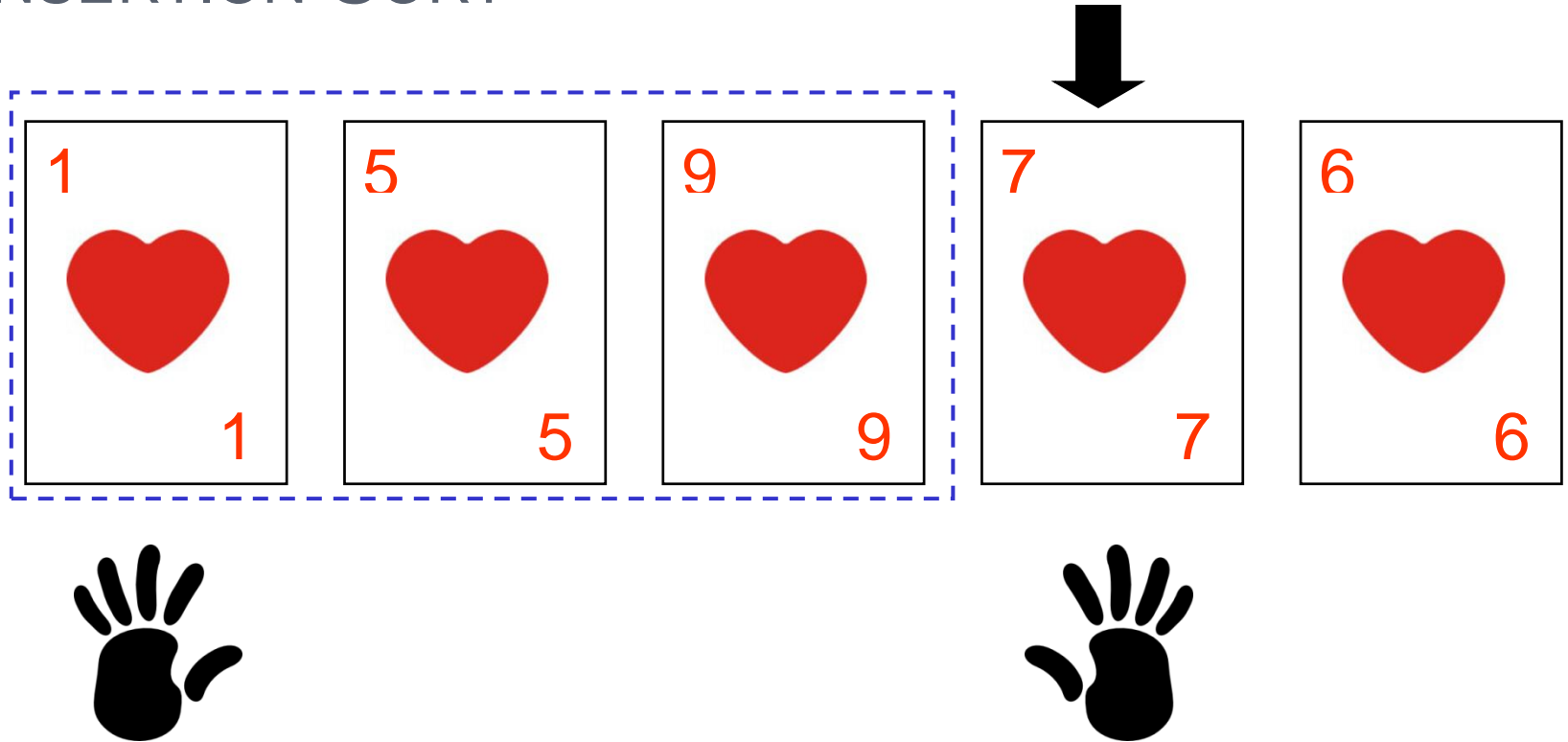
INSERTION SORT



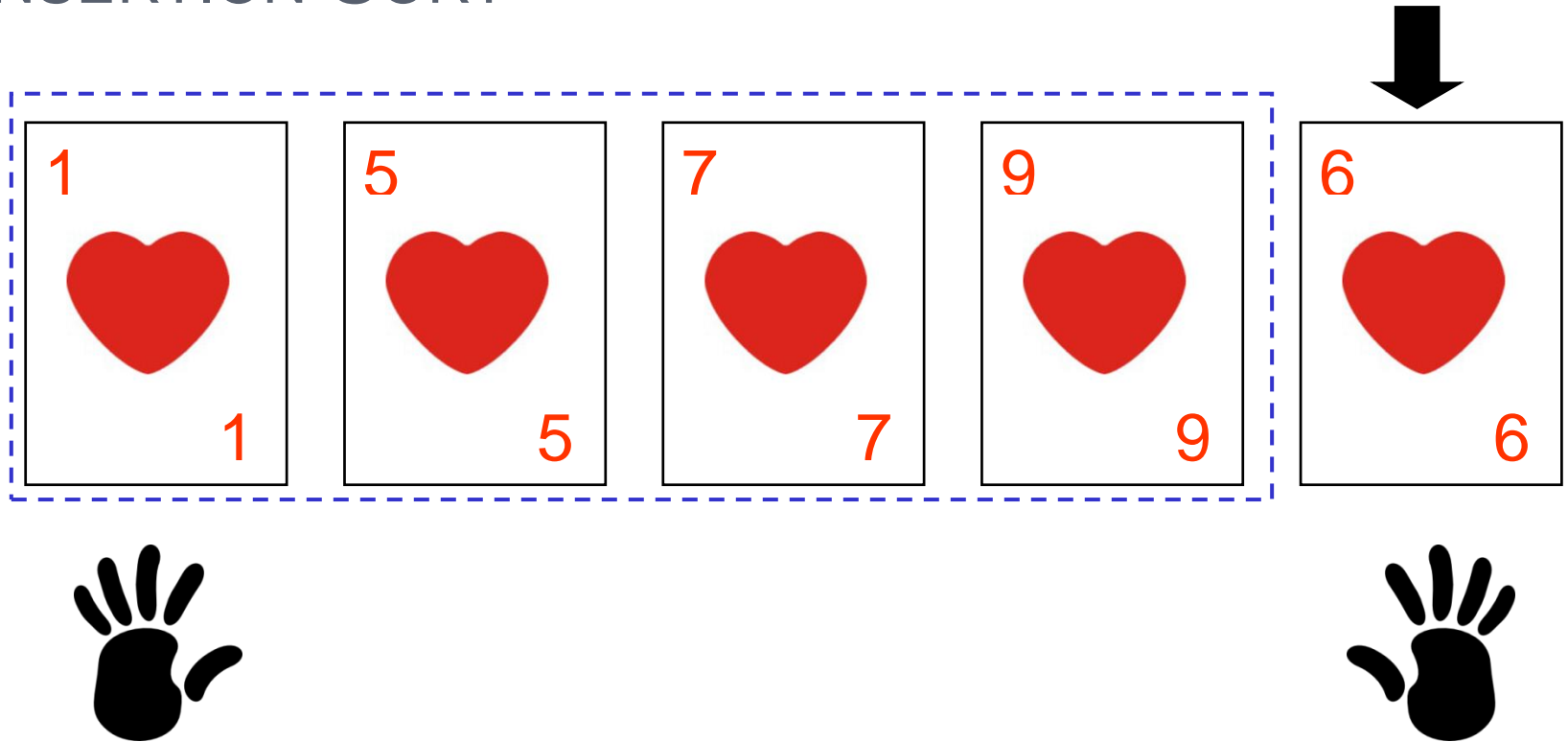
INSERTION SORT



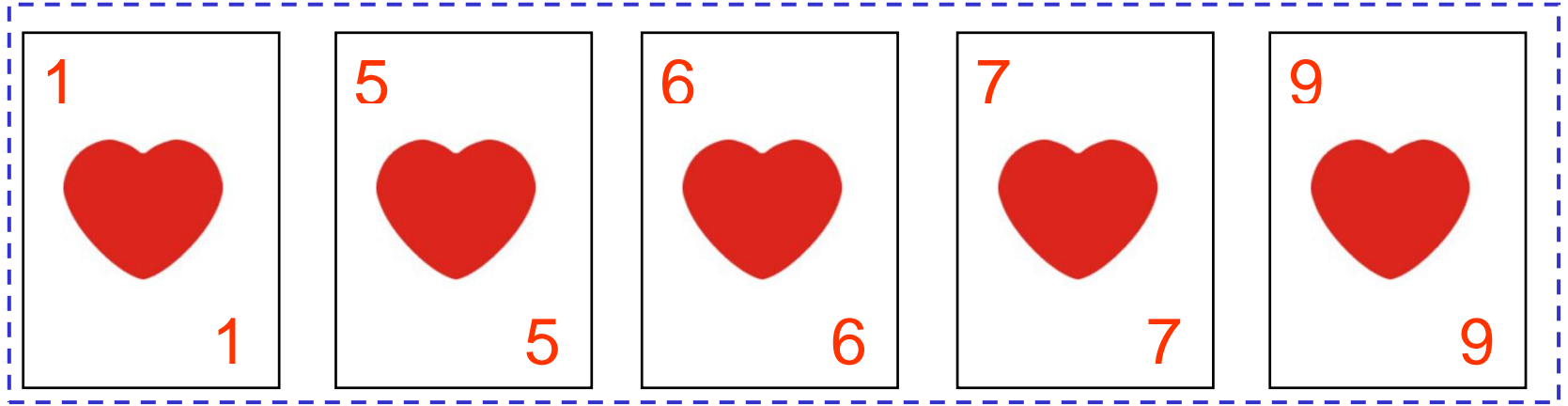
INSERTION SORT



INSERTION SORT



INSERTION SORT



INSERTION SORT

- Exercício:
 - Ordene a lista a seguir utilizando o insertion sort
 - 21, 23, 2, 34, 245, 33, 66

INSERTION SORT

- Como seria o algoritmo do Insertion Sort?

INSERTION-SORT(A, n)

for $j \leftarrow 2$ **to** n **do**

$key \leftarrow A[j]$

$i \leftarrow j - 1$

while $i > 0$ **and** $A[i] > key$ **do**

$A[i+1] \leftarrow A[i]$

$i \leftarrow i - 1$

$A[i+1] = key$

INSERTION SORT

- Como seria o algoritmo do Insertion Sort?

INSERTION-SORT(A, n)

for $j \leftarrow 2$ **to** n **do**

$\text{key} \leftarrow A[j]$

$i \leftarrow j - 1$

while $i > 0$ **and** $A[i] > \text{key}$ **do**

$A[i+1] \leftarrow A[i]$

$i \leftarrow i - 1$

$A[i+1] = \text{key}$

n repeticoes

(1)

(2)

(3)

...

(n)

$$\frac{n(n+1)}{2}$$

$$(n^2 + 1) / 2 = \Theta(n^2)$$

INSERTION SORT

- O que esse algoritmo faz?

```
endOfOrderedList = 1;
while(endOfOrderedList < length(A)){
  int next = A[endOfOrderedList + 1];
  for each j in endOfOrderedList downto 1 {
    if (next < A[j]) {
      swap(A[j],A[j+1]);
    }
  }
  endOfOrderedList++;
}
```

CARACTERÍSTICAS

- Fácil de implementar
- Fácil de entender
- In place: não precisa de espaço extra
- Stable
- Muita escrita (troca) e menos comparações
- Melhor Caso
 - $O(n)$
- Caso Médio
 - $O(n^2/4)$
- Eficiente para valores de n pequenos
- Melhor do que bubble e selection sort



OUTROS ALGORITMOS DE ORDENAÇÃO POR COMPARAÇÃO

36



GNOME SORT

- Inventor: Hamid Sarbazi-Azad (2000)
- Idéia:
 - Adota um pivot (que tenha anterior).
 - Olha para o proximo
 - Se o pivot e o proximo estão na ordem correta entao incrementa o pivot.
 - Se eles nao estao na ordem correta entao troca eles e decrementa o pivot.
 - Condições: se nao existe anterior ao pivot entao anda para frente (ao invés de decrementar). Se nao tem proximo entao termina.

GNOME SORT

- Como seria o algoritmo?

GNOME SORT

- Como seria o algoritmo?

```
procedure gnomeSort(a[])  
  pos := 1 //começando com 1  
  while pos < length(a) //enquanto nao atinge o ultimo elemento  
    if (a[pos] >= a[pos-1]) //se esta na ordem correta  
      pos := pos + 1  
    else  
      swap (a[pos],a[pos-1])  
      if (pos > 1) //se tem anterior  
        pos := pos - 1  
      else //nao tem anterior deve andar para frente  
        pos := pos + 1  
      end if  
    end if  
  end while  
end procedure
```

GNOME SORT

○ Análise

- Pior caso = $O(n^2)$
- Melhor caso = $O(n)$
- Caso médio = $O(n^2)$

COMB SORT

- Variação do bubble sort
- Inventor Wlodzimierz Dobosiewicz (1980)
- Focado em melhorar a entrada do bubble sort antes de aplica-lo (bubble compara elementos de distancia 1)
- Idéia:
 - A distancia começa por $gap = tamanho/fator$ ($fator=1.25$)
 - A entrada é ordenada com as trocas considerando elementos distanciados por gap
 - gap é atualizado ($gap = gap/fator$) até 1
 - Quando $gap=1$ combo sort continua até o array estar todo ordenado

COMB SORT

- Como seria o algoritmo?

COMB SORT

- Como seria o algoritmo?

```
function combsort(array input)
  gap := input.size //initialize gap size
  loop until gap = 1 and swaps = 0
    gap := gap / 1.25
    if gap < 1 then gap := 1
    i := 0
    swaps := 0
    loop until i + gap >= input.size
      if input[i] > input[i+gap] //faz trocas considerando distancia = gap
        swap(input[i], input[i+gap])
        swaps := 1 // ocorreu troca
      end if
      i := i + 1
    end loop
  end loop
end function
```

COMB SORT

○ Análise

- Pior caso = $O(n^2)$
- Melhor caso = $O(n)$
- Caso médio = $O(n \cdot \log n)$

QUESTÕES DE IMPLEMENTAÇÃO

- Como desenvolver/adaptar os algoritmos de ordenação vistos para ordenar tipos genéricos?
- Ao invés de trabalhar com inteiros precisamos trabalhar com um tipo T.

QUESTÕES DE IMPLEMENTAÇÃO

- Exemplo: bubblesort

```
procedure bubbleSort( A : list of sortable items )
```

```
  n = length(A)
```

```
  for each i in 1 to n – 1 inclusive {
```

```
    for each j in 1 to n-i inclusive{
```

```
      if (A[j] > A[j+1]){
```

```
        swap(A,j,j+1)
```

```
      }
```

```
    }
```

```
  }
```

```
end procedure
```

```
procedure swap(A: list of sortable items, int i, int j)
```

```
  int temp = A[i];
```

```
  A[i] = A[j];
```

```
  A[j] = temp;
```

```
end procedure
```

QUESTÕES DE IMPLEMENTAÇÃO

- Exemplo: bubblesort

```
procedure bubbleSort( A : list of Comparable items )  
  n = length(A)  
  for each i in 1 to n – 1 inclusive {  
    for each j in 1 to n-i inclusive{  
      if (A[j] > A[j+1]){  
        swap(A,j,j+1)  
      }  
    }  
  }  
end procedure  
procedure swap(A:list of Comparable items, int i, int j)  
  int temp = A[i];  
  A[i] = A[j];  
  A[j] = temp;  
end procedure
```

QUESTÕES DE IMPLEMENTAÇÃO

- Exemplo: bubblesort

```
procedure bubbleSort( A : list of Comparable items )  
  n = length(A)  
  for each i in 1 to n – 1 inclusive {  
    for each j in 1 to n-i inclusive{  
      if ( A[j].compareTo(A[j+1]) > 0 ){  
        swap(A,j,j+1)  
      }  
    }  
  }  
end procedure  
procedure swap(A:list of Comparable items, int i, int j)  
  int temp = A[i];  
  A[i] = A[j];  
  A[j] = temp;  
end procedure
```


QUESTÕES DE IMPLEMENTAÇÃO

- Exemplo: bubblesort
 - Como implementar o bubblesort considerando apenas um “pedaço” do vetor a ser ordenado?

procedure bubbleSort(A : list of **Comparable** items)

```
n = length(A)
for each i in 1 to n – 1 inclusive {
  for each j in 1 to n-i inclusive{
    if (A[j].compareTo(A[j+1]) > 0){
      swap(A,j,j+1)
    }
  }
}
end procedure
```

QUESTÕES DE IMPLEMENTAÇÃO

○ Exemplo: bubblesort

- Como implementar o bubblesort considerando apenas um “pedaço” do vetor a ser ordenado?

```
procedure bubbleSort( A : list of Comparable items,  
                     int leftIndex, int rightIndex )
```

```
  for each i in leftIndex to rightIndex inclusive {  
    for each j in 1 to rightIndex inclusive{  
      if ( A[j].compareTo(A[j+1]) > 0 ){  
        swap(A,j,j+1)  
      }  
    }  
  }  
end procedure
```