BubbleSort vs InsertionSort

Igor Machado Coelho

Prova Didática para o Concurso de Prof. Substituto
Instituto de Computação – Universidade Federal Fluminense (UFF)

22 de Janeiro de 2015

Instituto de computação

Conteúdo da Aula

Por que ordenar?

- A ordenação é uma operação natural do coditiano

Por que ordenar?

- A ordenação é uma operação natural do coditiano
- Ordenar para encontrar mais facilmente (busca)

Por que ordenar?

- A ordenação é uma operação natural do coditiano
- Ordenar para encontrar mais facilmente (busca)
- Ordenar para facilitar tarefa futura (intercalação)

Por que ordenar?

- A ordenação é uma operação natural do coditiano
- Ordenar para encontrar mais facilmente (busca)
- Ordenar para facilitar tarefa futura (intercalação)
- Ordenar para exibição e interpretação de dados mais fácil.

Ordenação

- Ordenação de vetores (possível também em listas, etc)

Ordenação

- Ordenação de vetores (possível também em listas, etc)
- Operação de comparação bem definida

Ordenação

- Ordenação de vetores (possível também em listas, etc)
- Operação de comparação bem definida
- Entrada: vetor com n elementos a serem ordenados

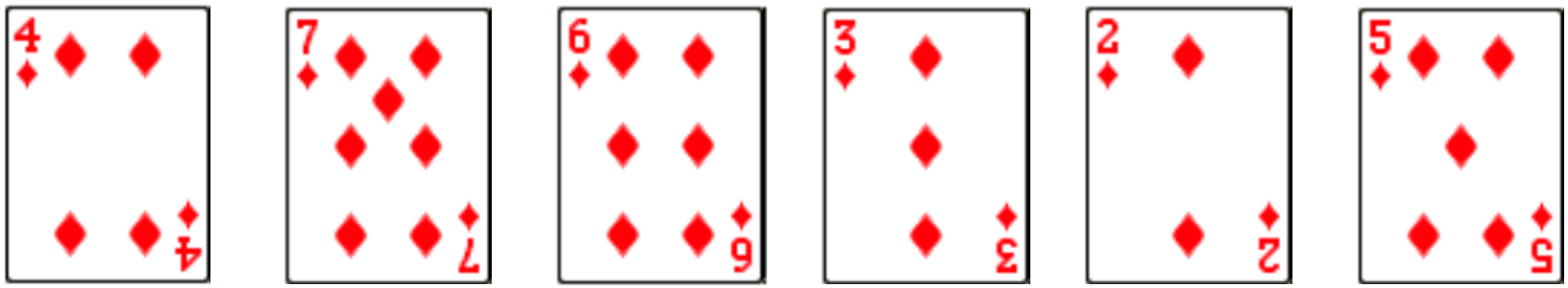
Ordenação

- Ordenação de vetores (possível também em listas, etc)
- Operação de comparação bem definida
- Entrada: vetor com n elementos a serem ordenados
- Saída: vetor com os elementos ordenados

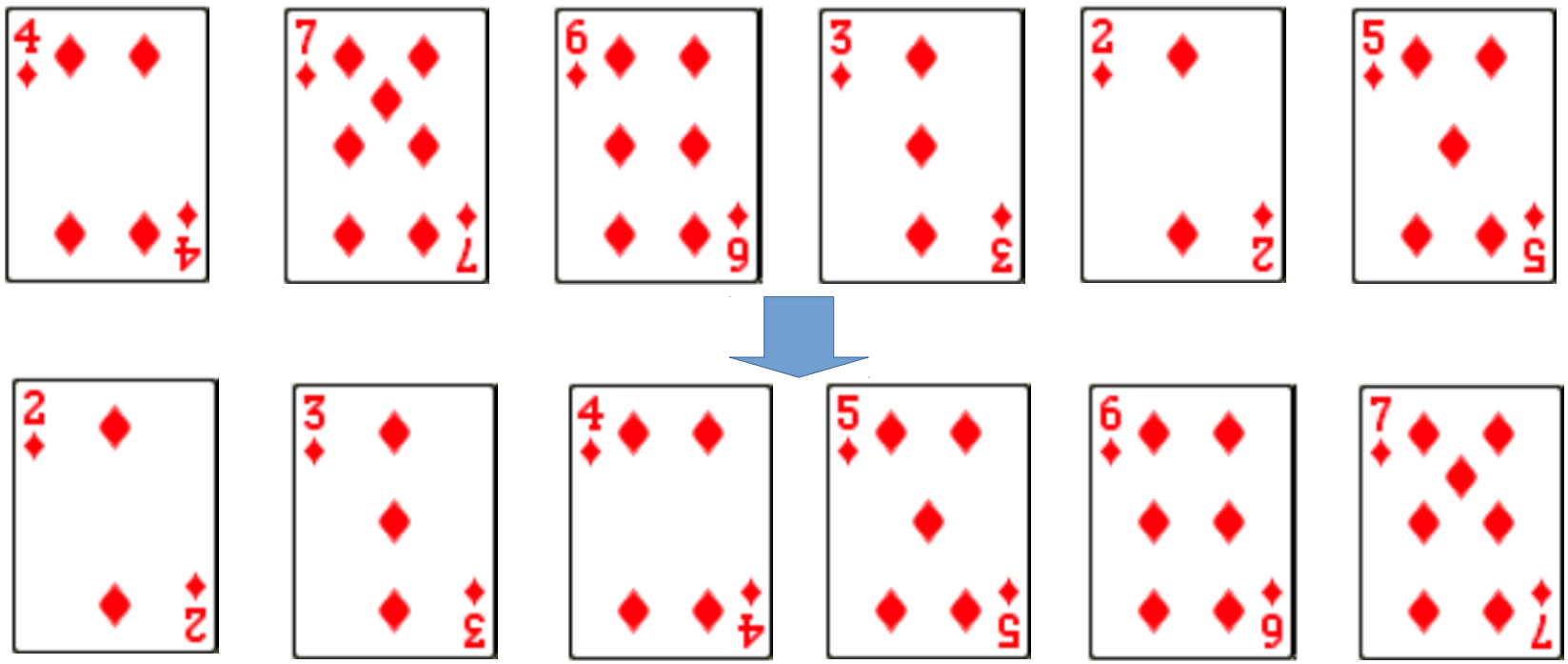
Ordenação

- Ordenação de vetores (possível também em listas, etc)
- Operação de comparação bem definida
- Entrada: vetor com n elementos a serem ordenados
- Saída: vetor com os elementos ordenados
- Ordem crescente ou decrescente

Objetivo



Objetivo



Ordenação

- Algoritmos simples de implementar (maior número de comparações): bolha, inserção, seleção, ...

Ordenação

- Algoritmos simples de implementar (maior número de comparações): bolha, inserção, seleção, ...
- Algoritmos mais eficientes (baixo número de comparações aprox. $n \log n$): *mergesort*, *quicksort*, ...

Ordenação

- Algoritmos simples de implementar (maior número de comparações): bolha, inserção, seleção, ...
- Algoritmos mais eficientes (baixo número de comparações aprox. $n \log n$): *mergesort*, *quicksort*, ...
- Extensões dos algoritmos básicos podem ser muito boas na prática!

Inserção e Bolha

- Ordenação Bolha e por Inserção são técnicas clássicas

Inserção e Bolha

- Ordenação Bolha e por Inserção são técnicas clássicas
- Bolha: elementos “caminham” como uma bolha até o topo

Inserção e Bolha

- Ordenação Bolha e por Inserção são técnicas clássicas
- Bolha: elementos "caminham" como uma bolha até o topo

- Inserção: forma comum de se ordenar cartas (como no pôquer)

Ordenação por Inserção

Ideia

1. Percorrer o vetor da esquerda para a direita;

Ordenação por Inserção

Ideia

1. Percorrer o vetor da esquerda para a direita;
2. tomar o primeiro elemento à direita e armazená-lo;

Ordenação por Inserção

Ideia

1. Percorrer o vetor da esquerda para a direita;
2. tomar o primeiro elemento à direita e armazená-lo;
3. inserí-lo na posição correta no subvetor à esquerda.

Ordenação por Inserção

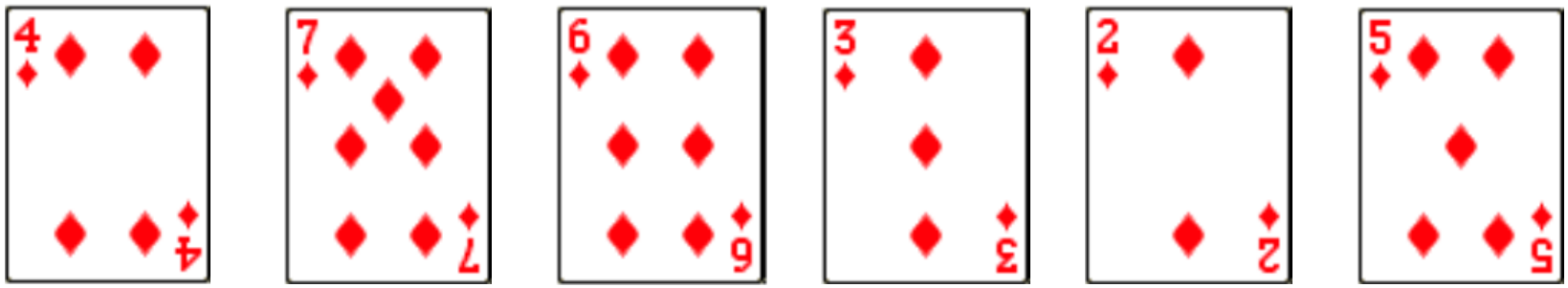
Ideia

1. Percorrer o vetor da esquerda para a direita;
2. tomar o primeiro elemento à direita e armazená-lo;
3. inserí-lo na posição correta no subvetor à esquerda.

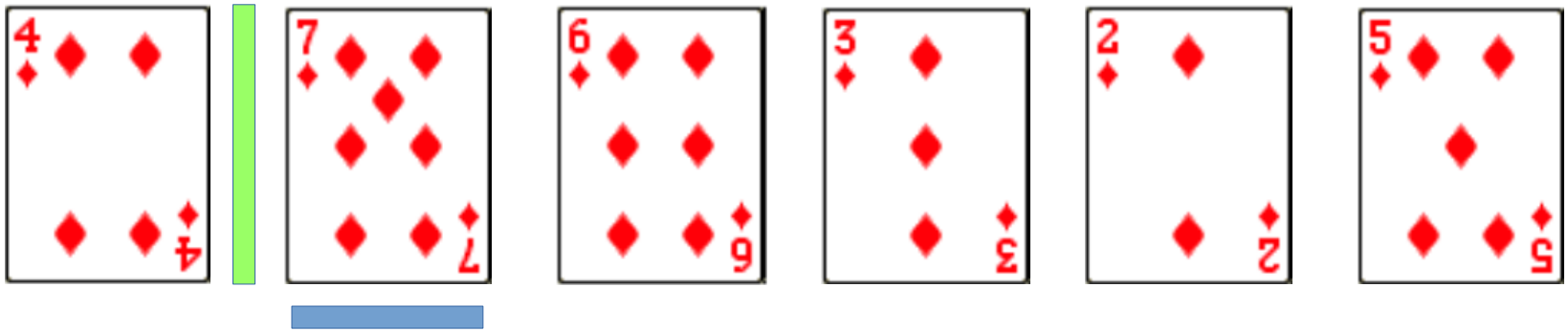
O passo 3 envolve realocações sucessivas para uma estrutura de vetor.

Ordenação por Inserção

Exemplo

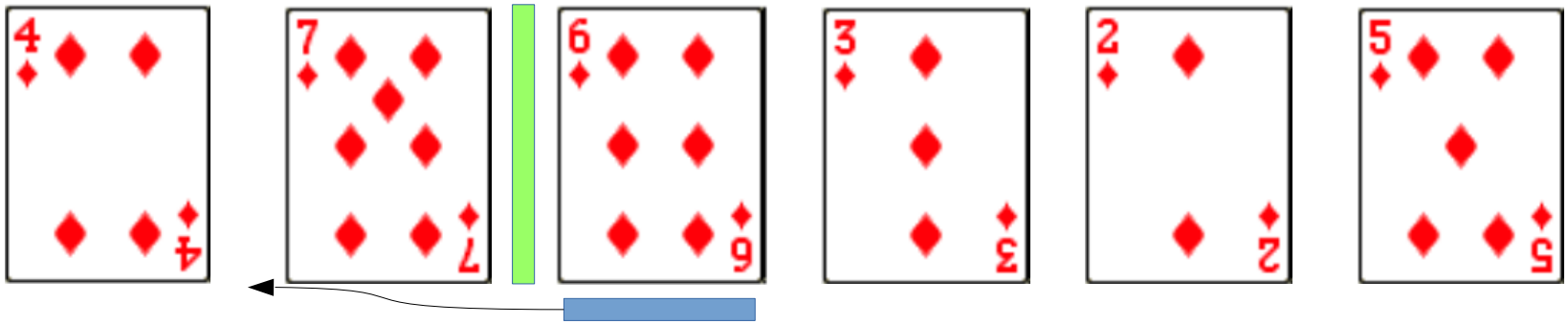


Exemplo



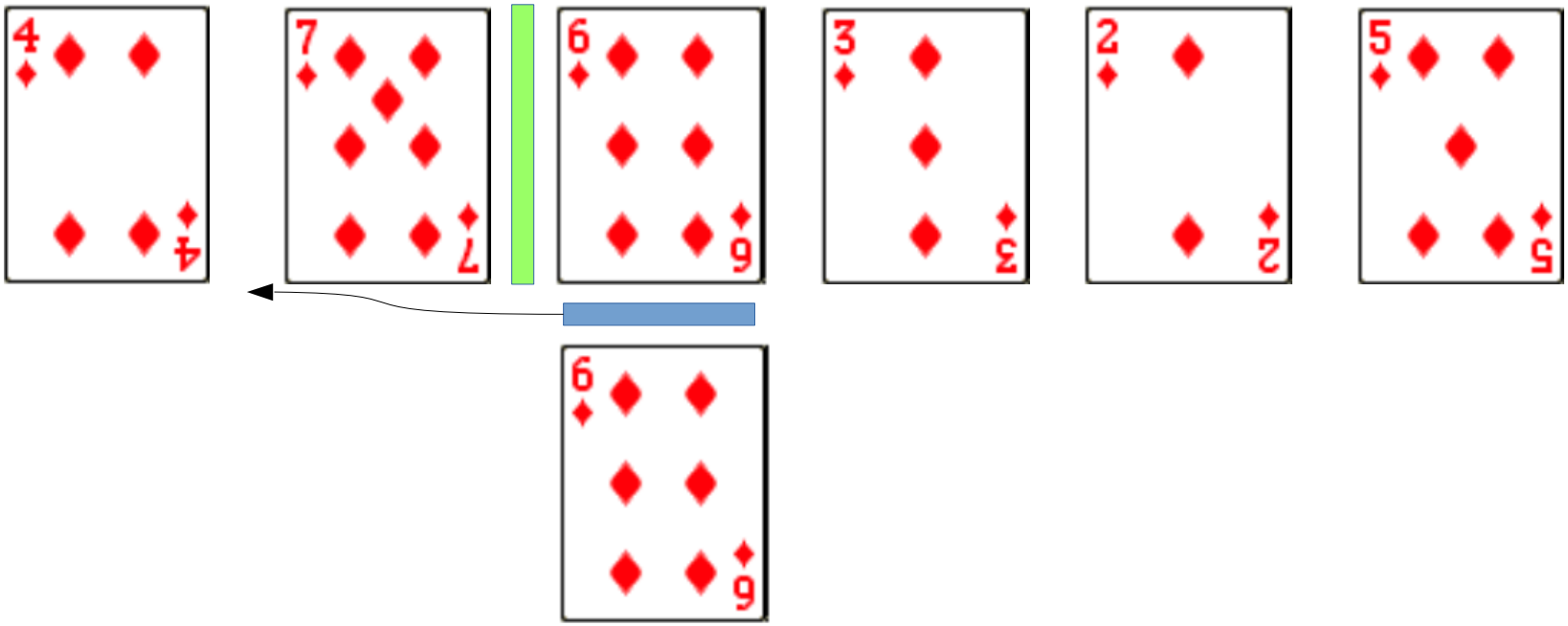
Ordenação por Inserção

Exemplo



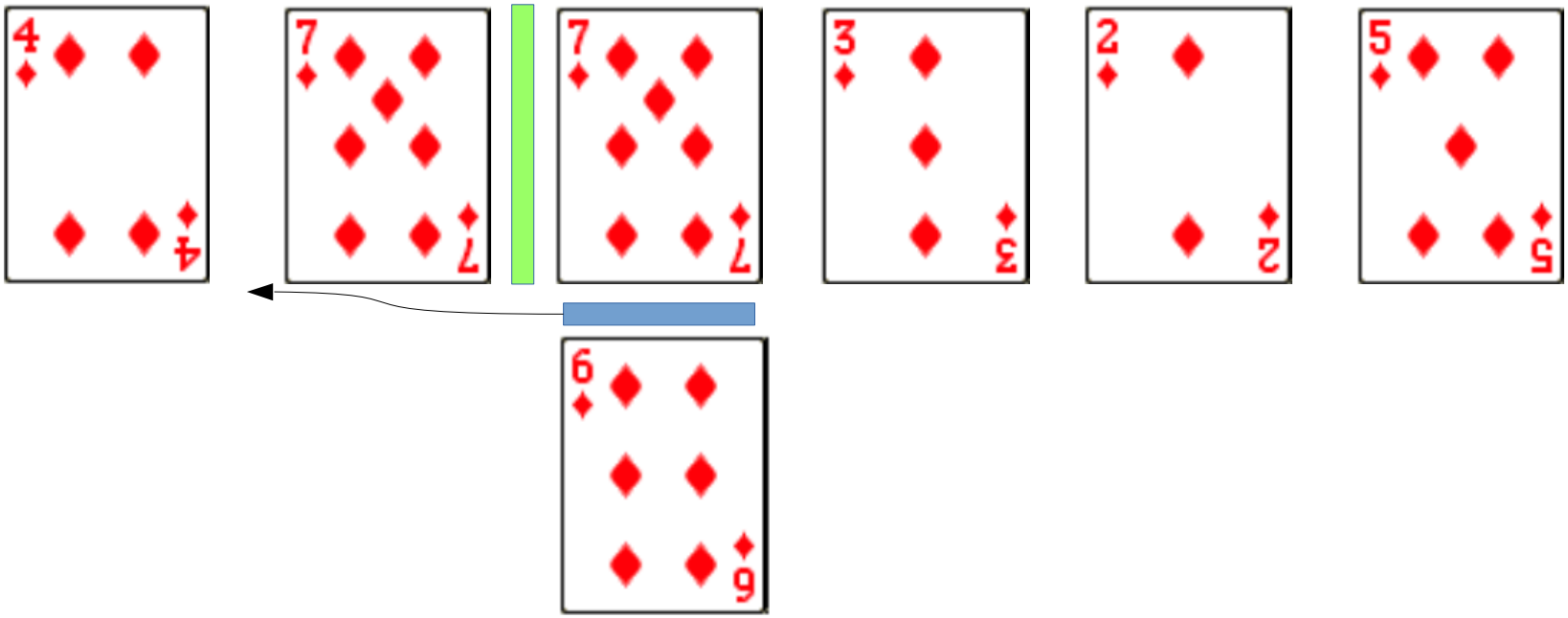
Ordenação por Inserção

Exemplo



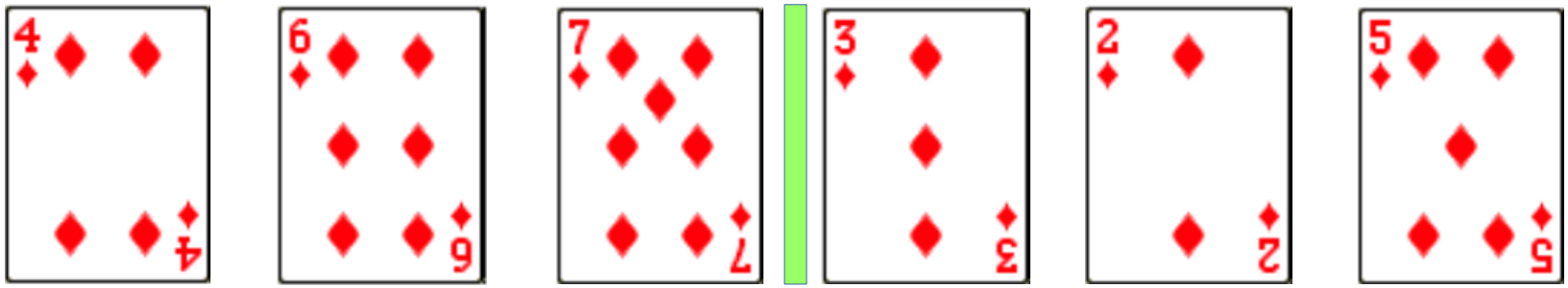
Ordenação por Inserção

Exemplo



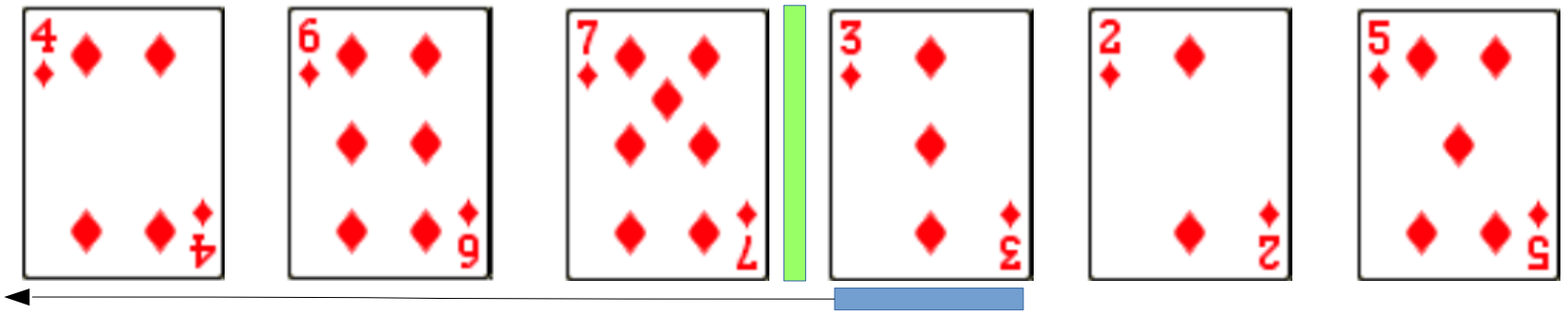
Ordenação por Inserção

Exemplo



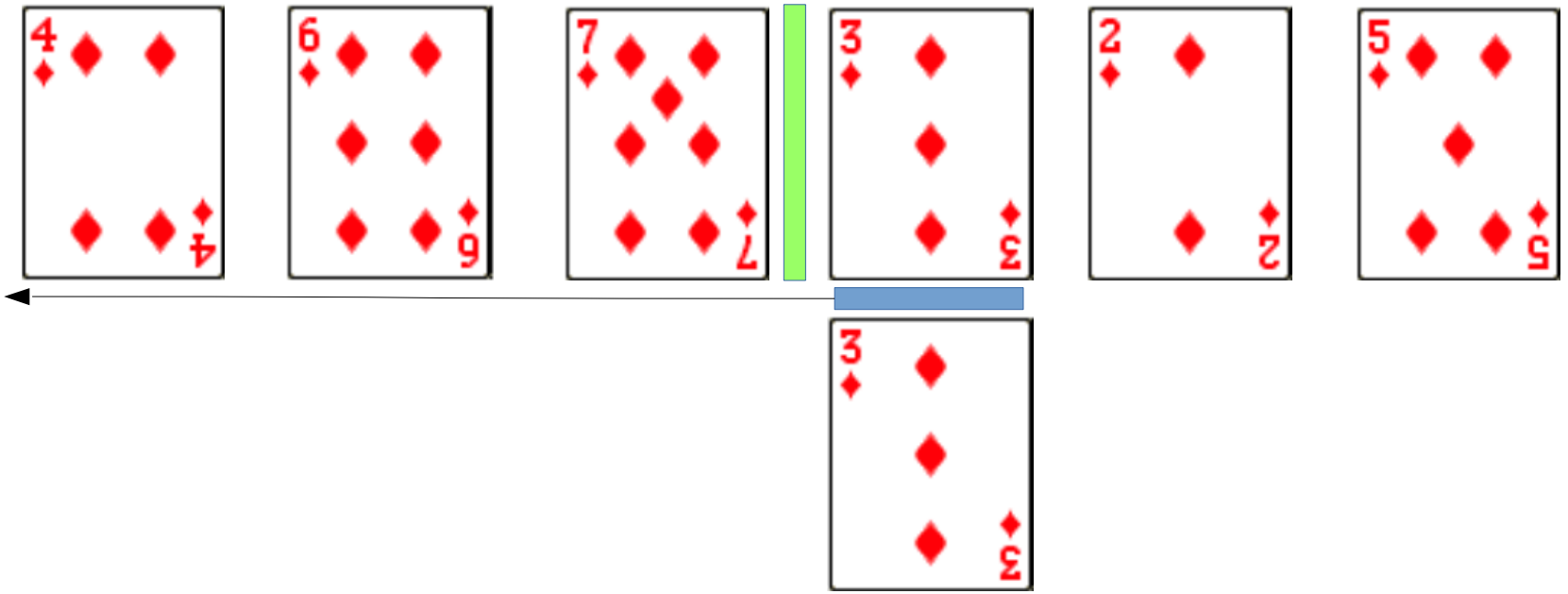
Ordenação por Inserção

Exemplo



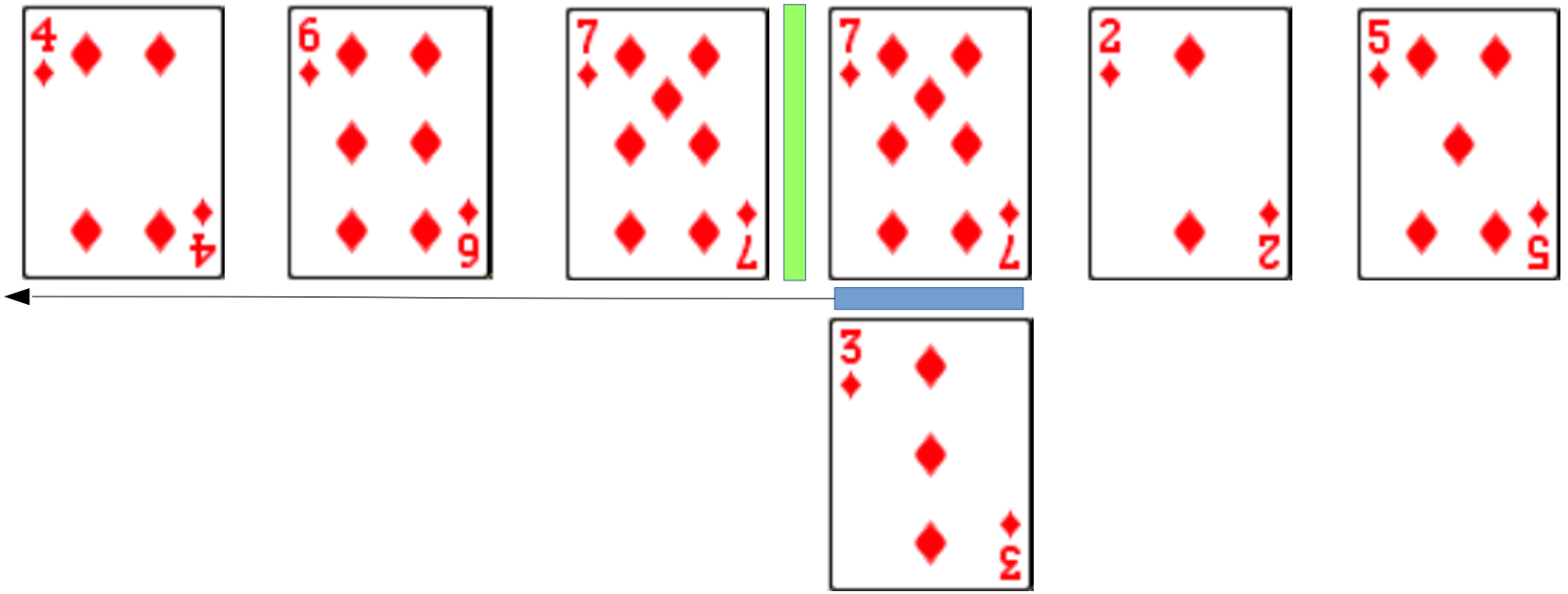
Ordenação por Inserção

Exemplo



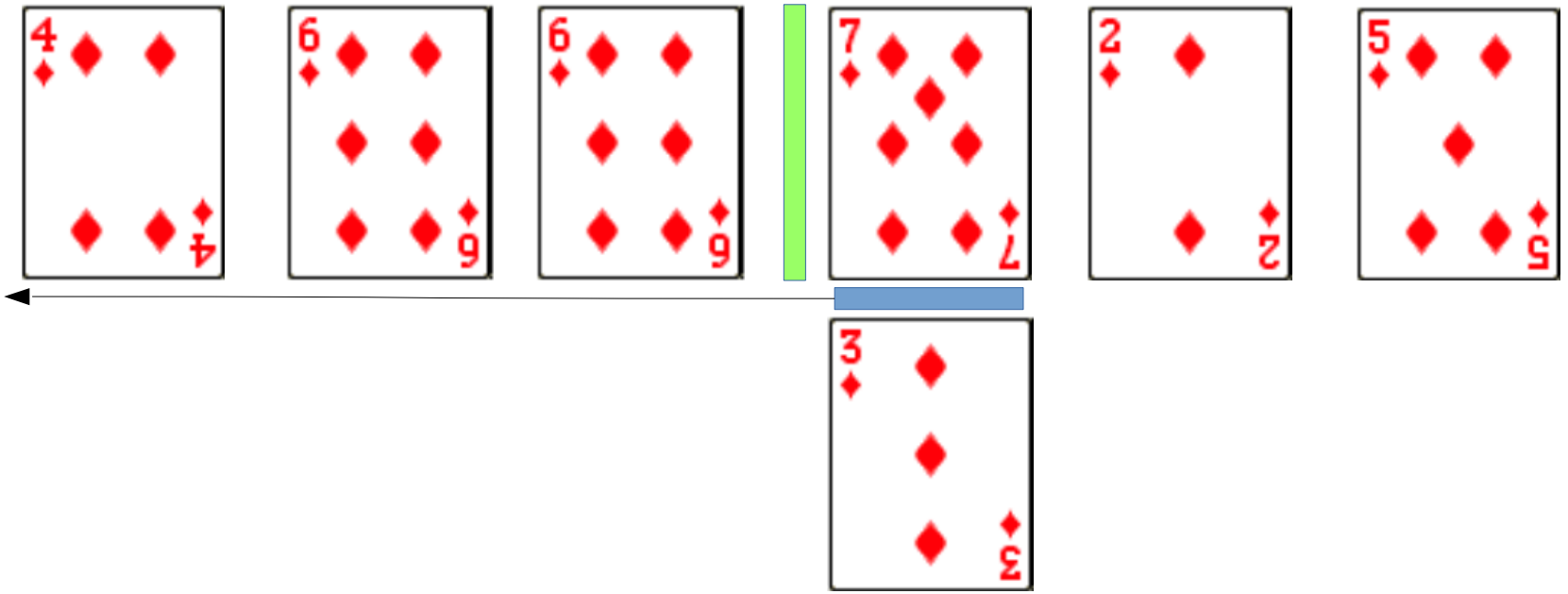
Ordenação por Inserção

Exemplo



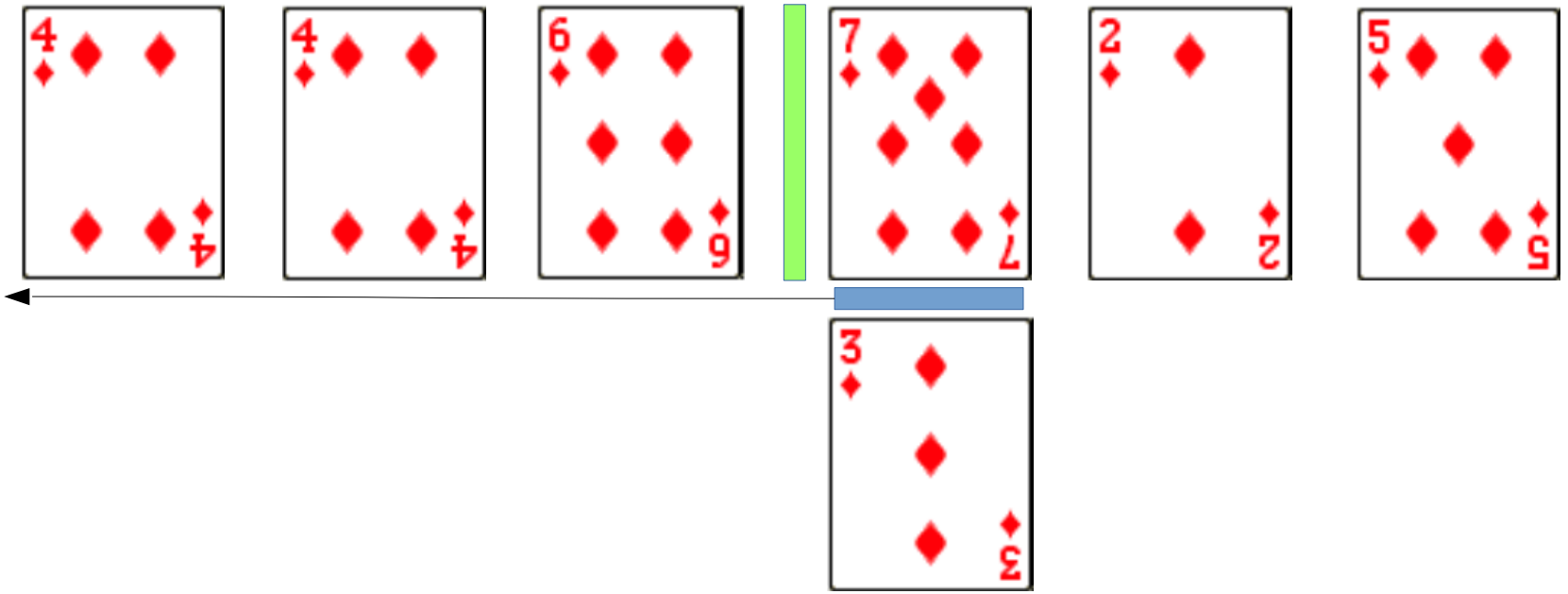
Ordenação por Inserção

Exemplo

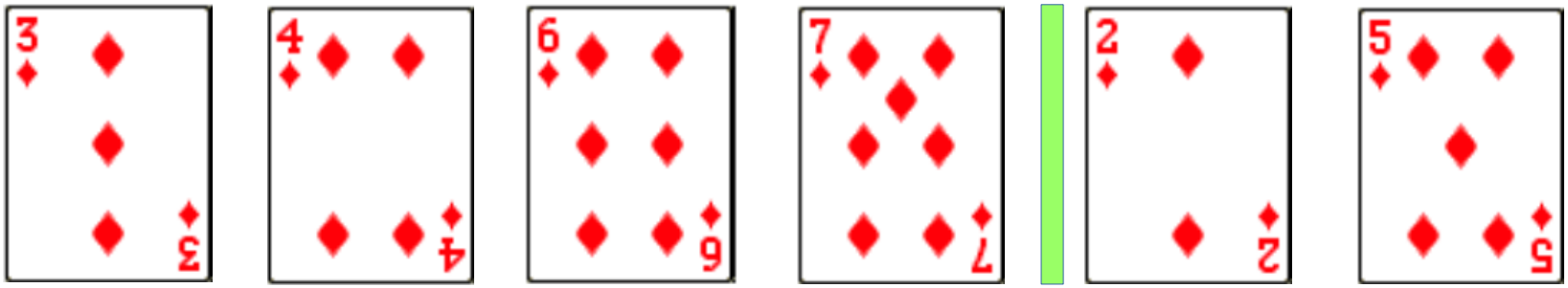


Ordenação por Inserção

Exemplo

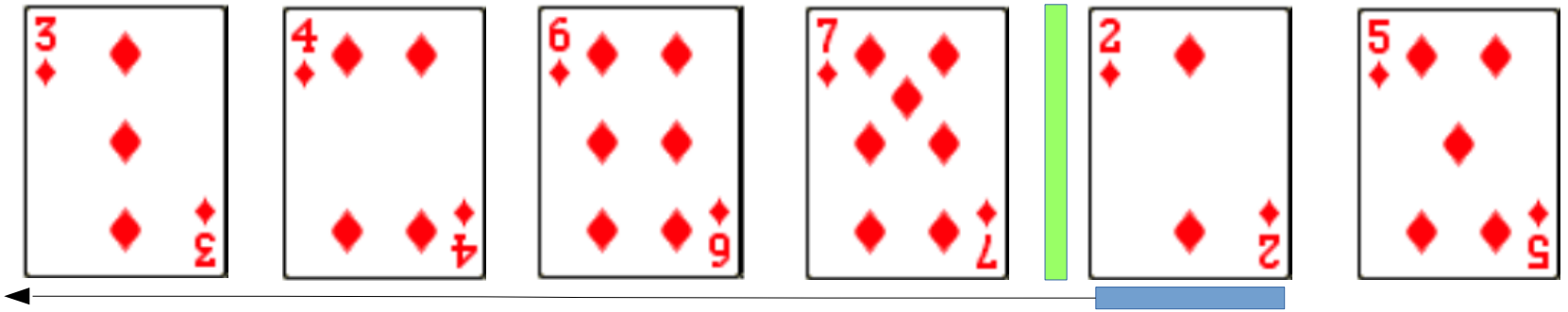


Exemplo



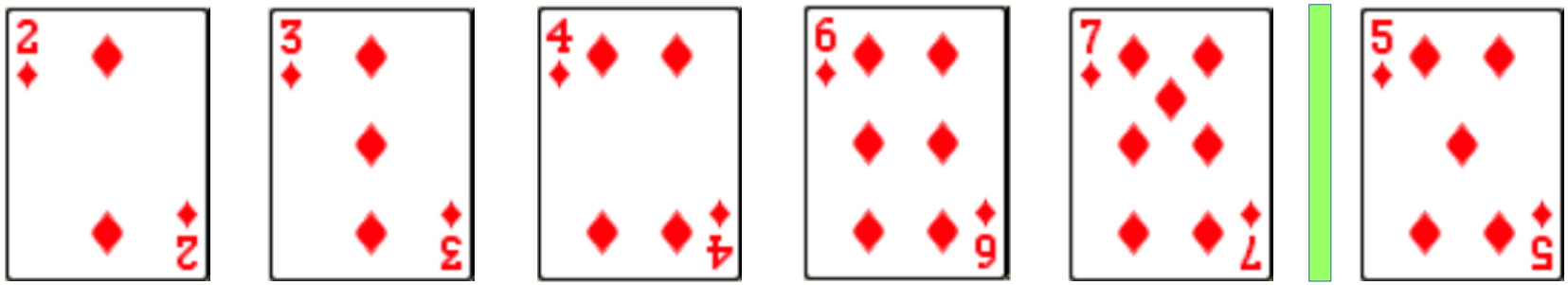
Ordenação por Inserção

Exemplo



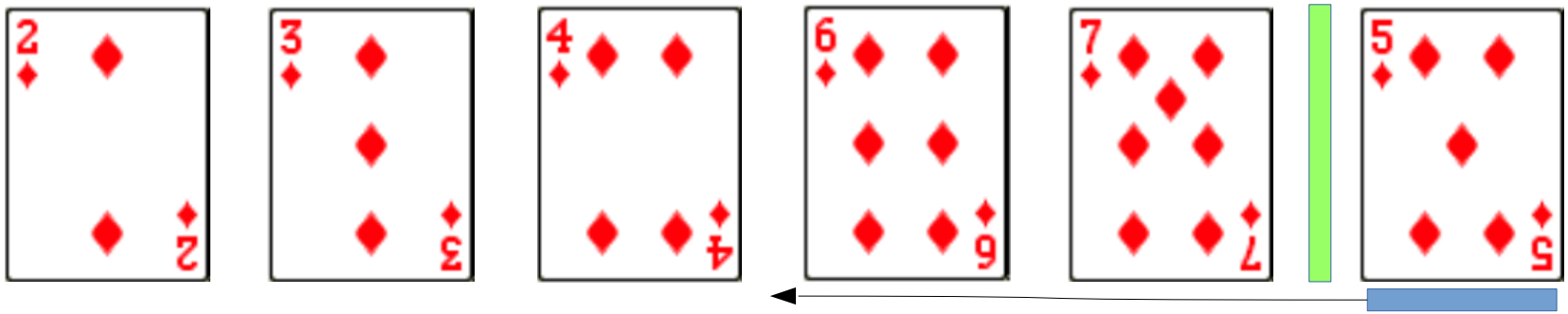
Ordenação por Inserção

Exemplo



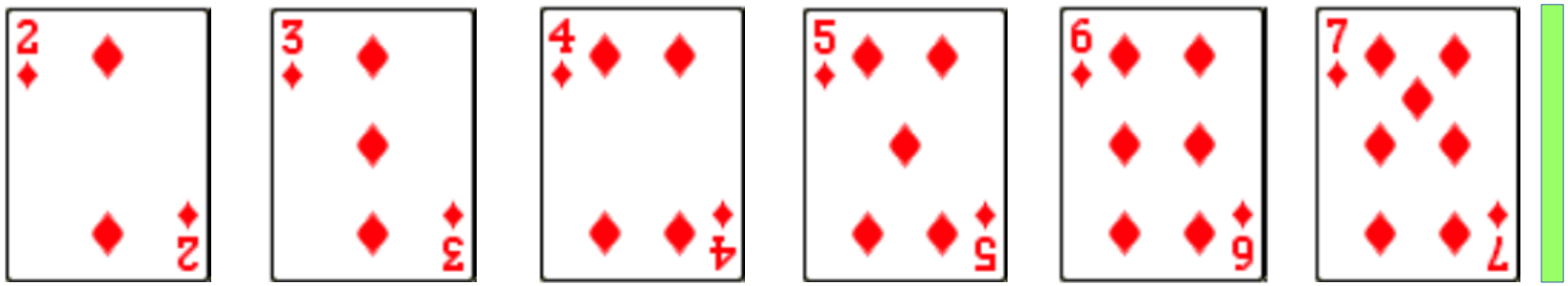
Ordenação por Inserção

Exemplo



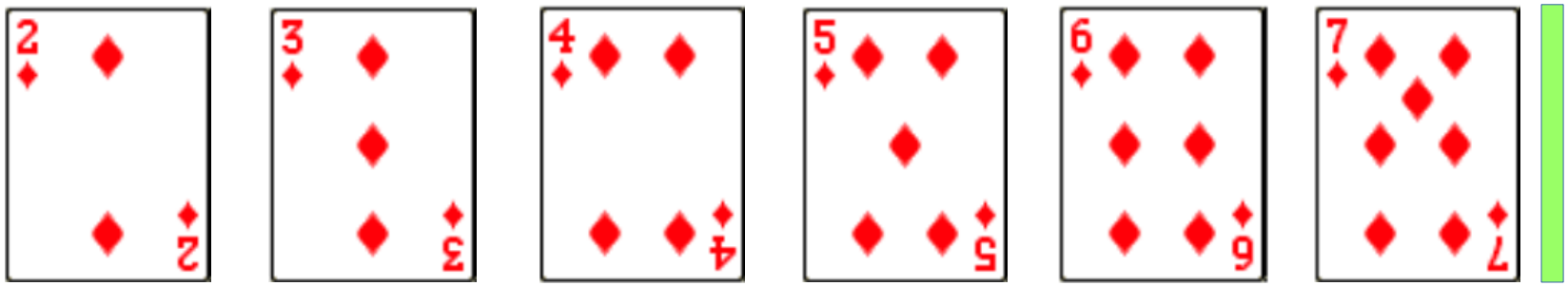
Ordenação por Inserção

Exemplo



Ordenação por Inserção

Exemplo



Ordenação por Inserção

Código em C

```
int i, j, t;
for (j = 1 ; j <= n - 1; j++) {
  i = j;

  while ( i > 0 && array[i] < array[i-1]) {
    t          = array[i];
    array[i]   = array[i-1];
    array[i-1] = t;

    i--;
  }
}
```

Ordenação por Inserção

Análise

1. Percorrer o vetor da esquerda para a direita; $O(n)$

Ordenação por Inserção

Análise

1. Percorrer o vetor da esquerda para a direita; $O(n)$
2. tomar o primeiro elemento à direita e armazená-lo; $O(1)$

Ordenação por Inserção

Análise

1. Percorrer o vetor da esquerda para a direita; $O(n)$
2. tomar o primeiro elemento à direita e armazená-lo; $O(1)$
3. inserí-lo na posição correta no subvetor à esquerda. $O(n)$

Ordenação por Inserção

Análise

1. Percorrer o vetor da esquerda para a direita; $O(n)$
2. tomar o primeiro elemento à direita e armazená-lo; $O(1)$
3. inserí-lo na posição correta no subvetor à esquerda. $O(n)$

Complexidade: $O(n^2)$

Ordenação Bolha

Ideia

1. Percorrer o vetor da esquerda para a direita;

Ordenação Bolha

Ideia

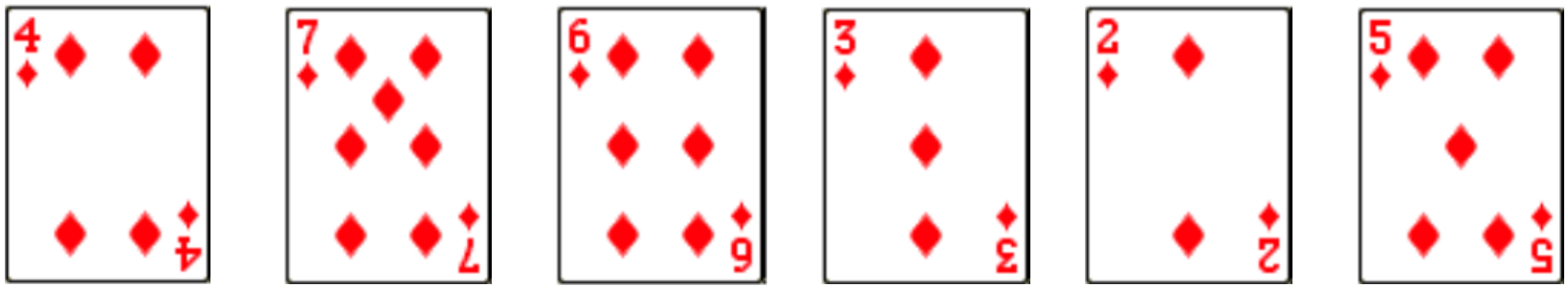
1. Percorrer o vetor da esquerda para a direita;
2. a cada passo, ordenar o elemento atual e o próximo à direita;

Ordenação Bolha

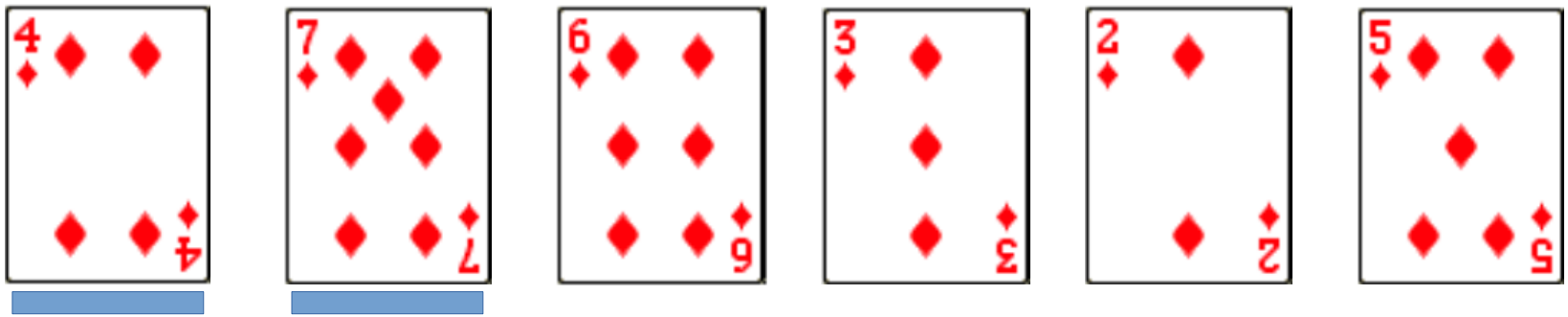
Ideia

1. Percorrer o vetor da esquerda para a direita;
2. a cada passo, ordenar o elemento atual e o próximo à direita;
3. repetir o processo até que o vetor esteja ordenado.

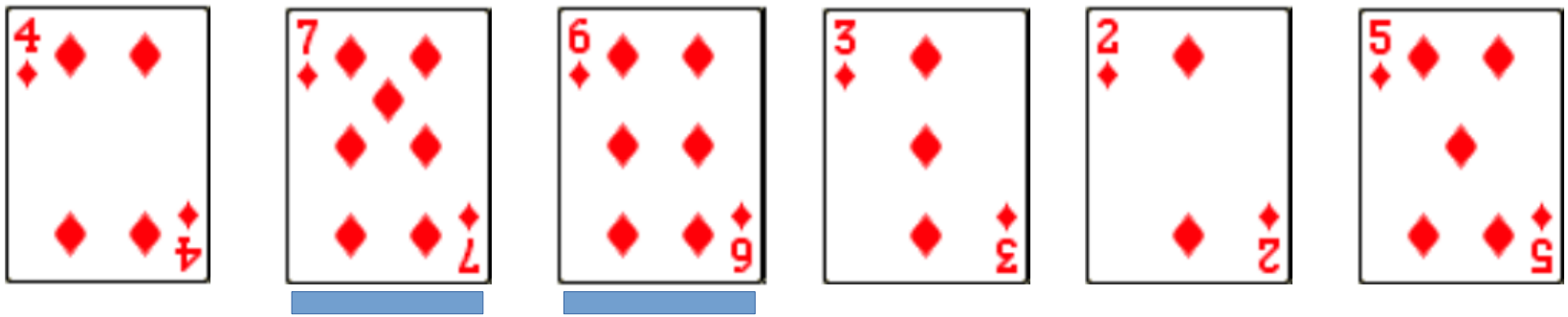
Exemplo



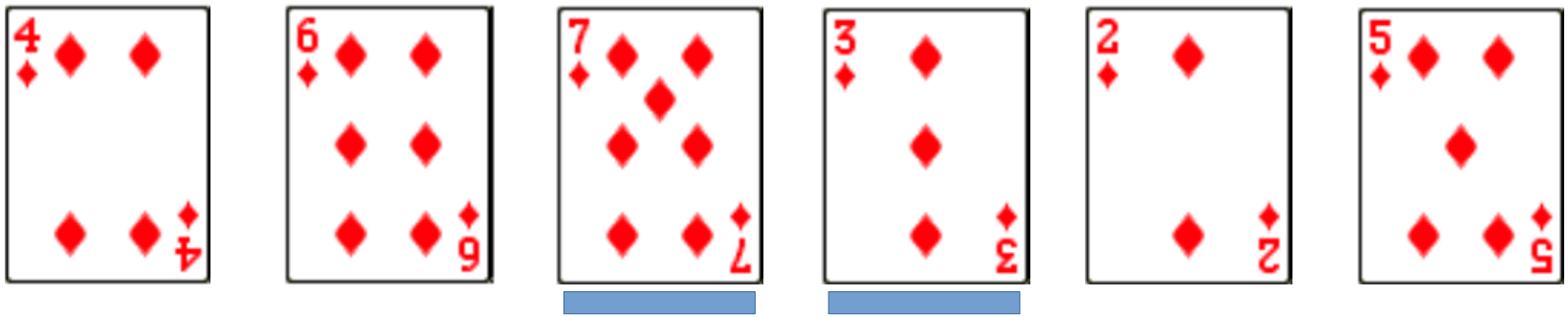
Exemplo



Exemplo

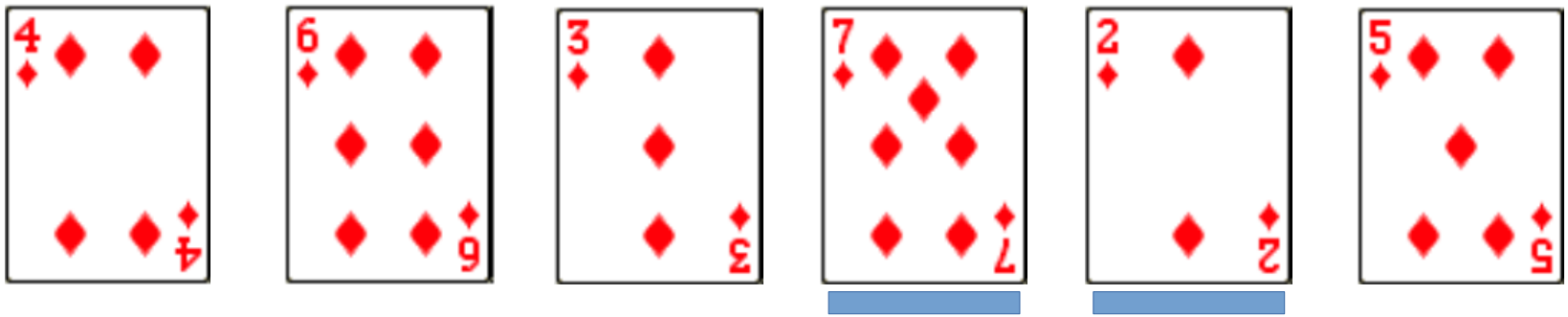


Exemplo

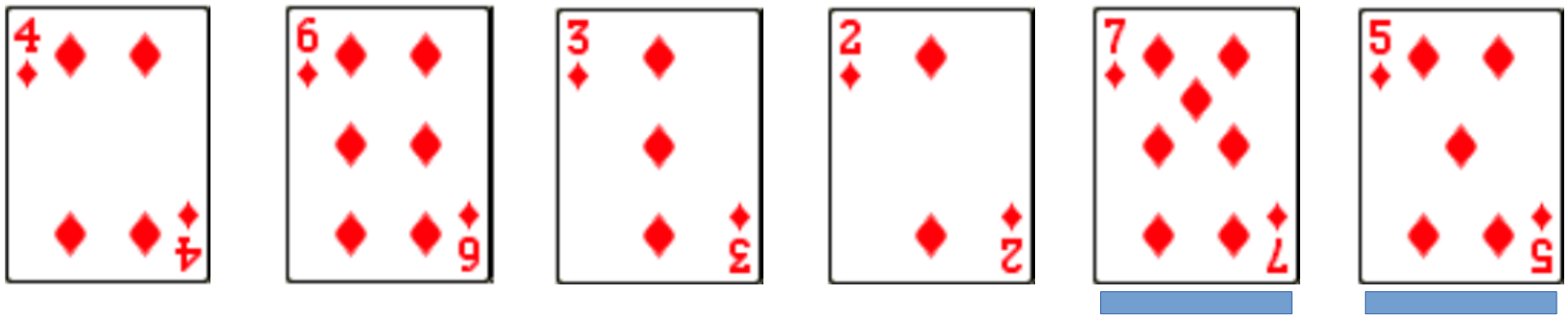


Ordenação Bolha

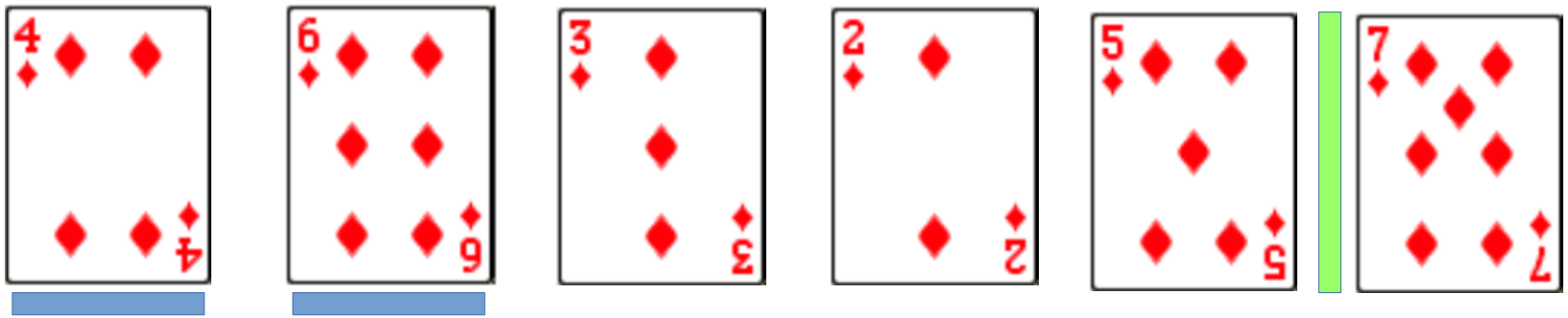
Exemplo



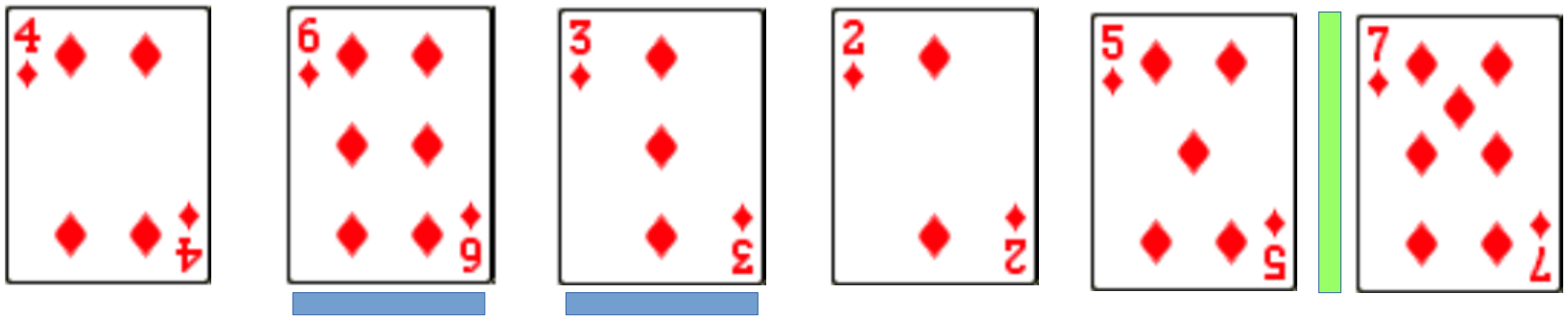
Exemplo



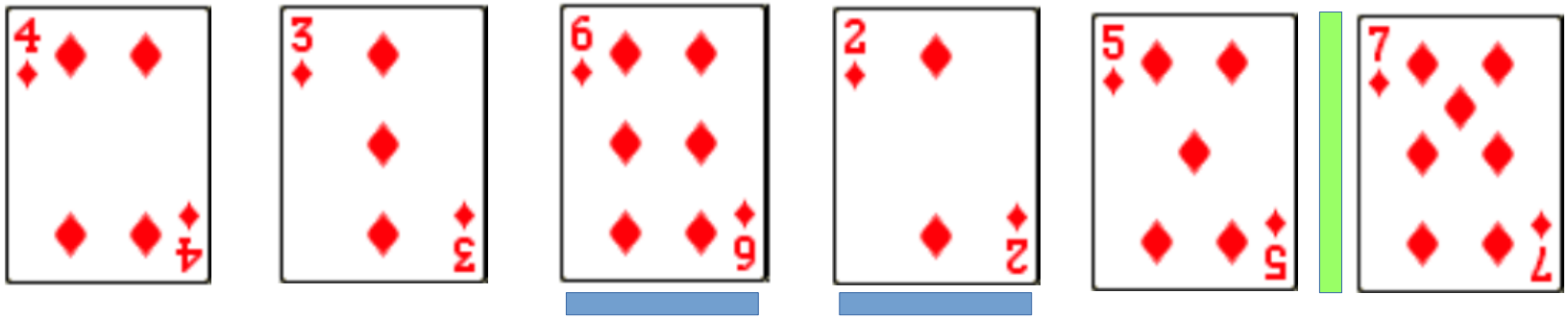
Exemplo



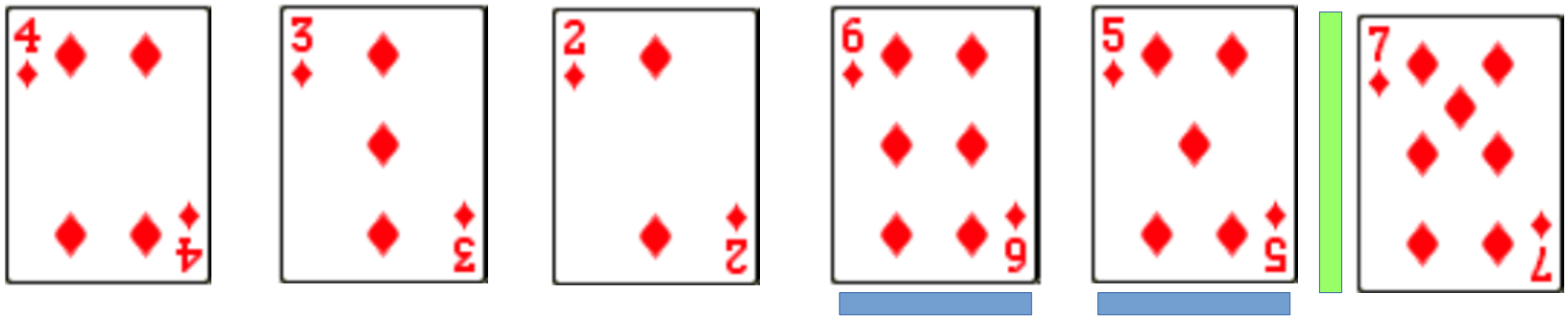
Exemplo



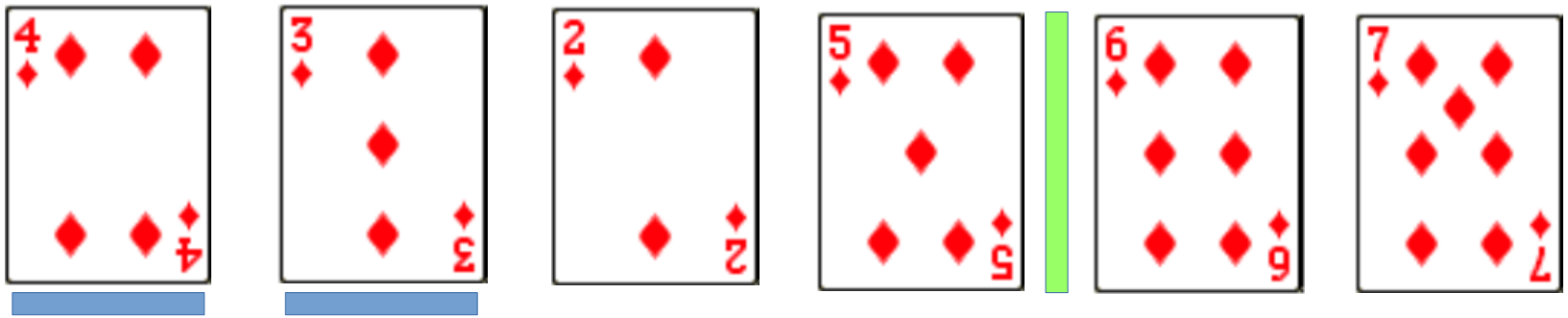
Exemplo



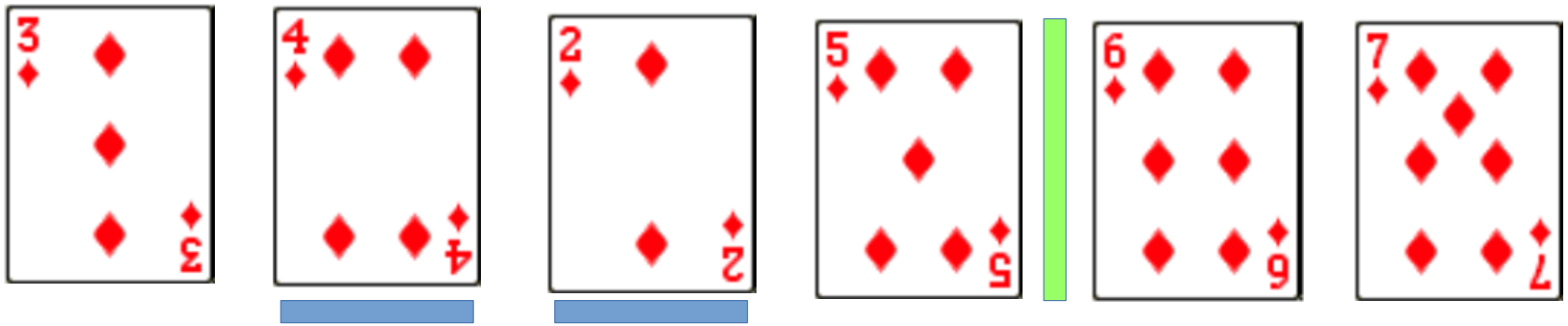
Exemplo



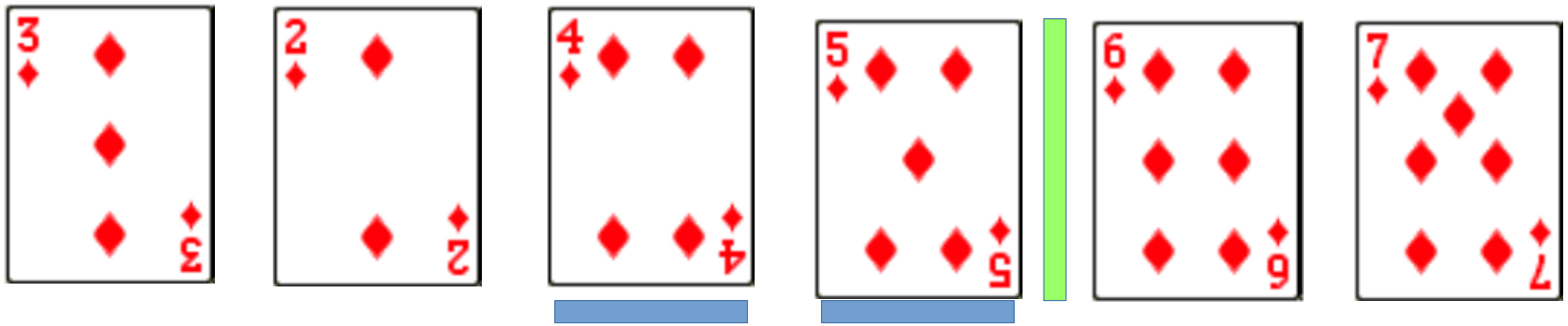
Exemplo



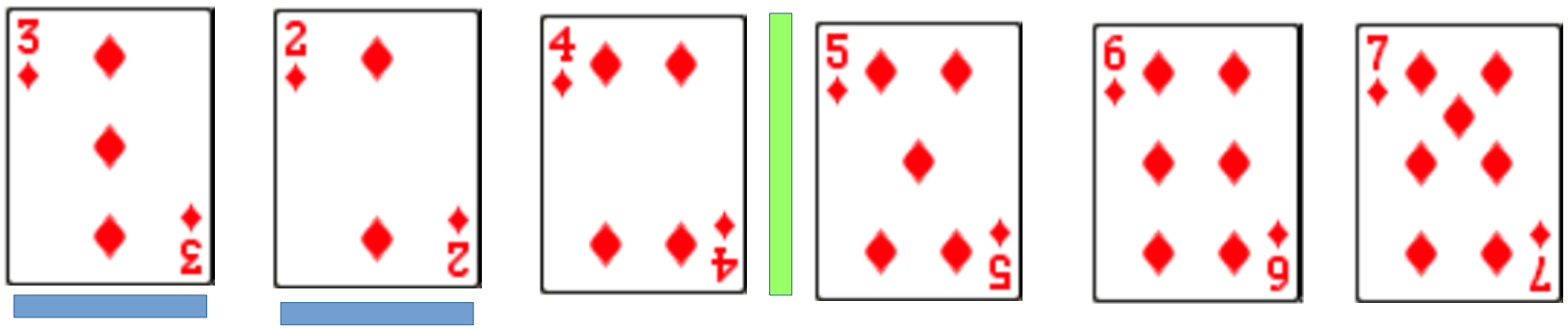
Exemplo



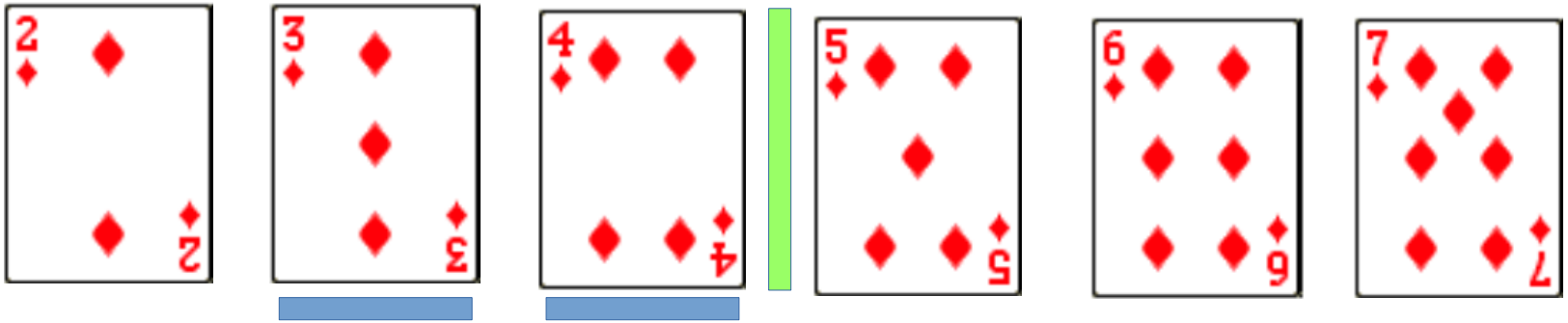
Exemplo



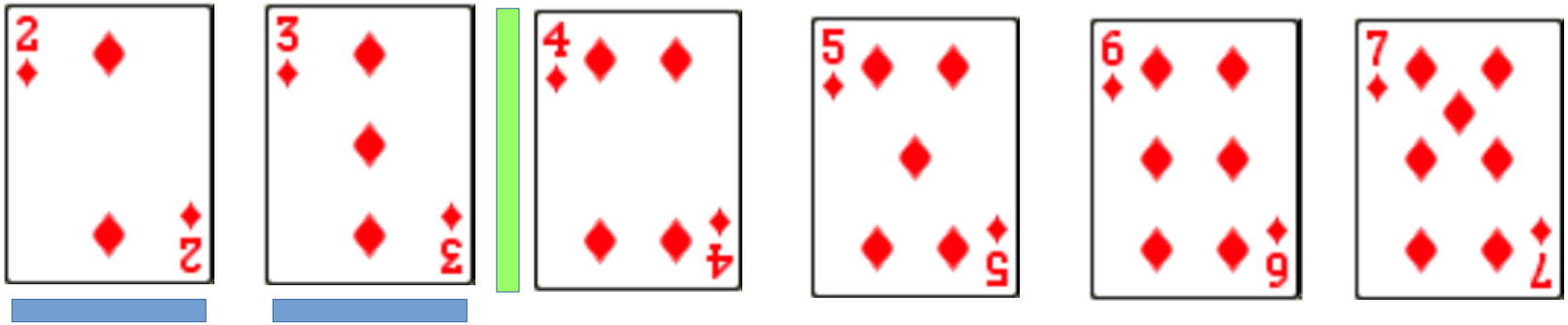
Exemplo



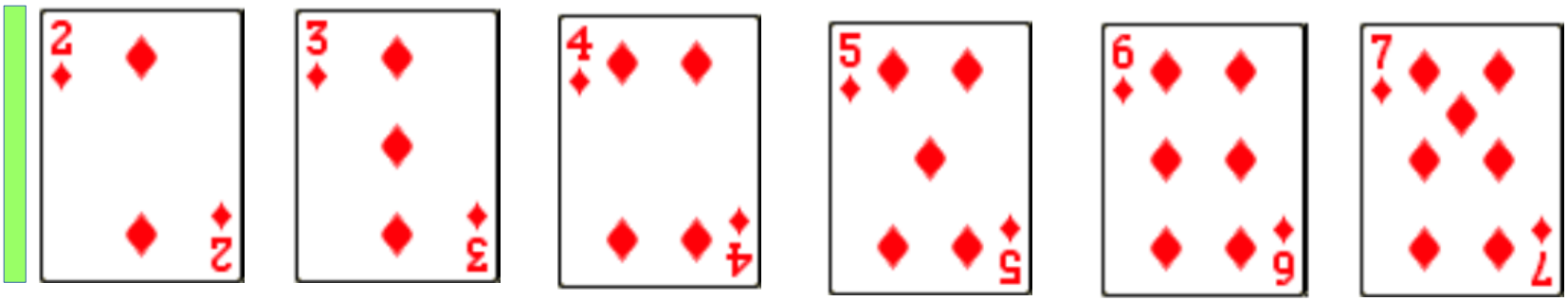
Exemplo



Exemplo



Exemplo



Ordenação Bolha

Código em C

```
int i, j, t;
for( i = 0; i < n-1 ;  i++ )
{
    for( j=0; j < n-i+1; j++ )
    {
        if( array[j] > array[j+1] )
        {
            t          = array[j];
            array[j]   = array[j+1];
            array[j+1] = t;
        }
    }
}
```

Análise

1. Percorrer o vetor da esquerda para a direita; $O(n)$

Ordenação Bolha

Análise

1. Percorrer o vetor da esquerda para a direita; $O(n)$
2. a cada passo, ordenar o elemento atual e o próximo à direita; $O(1)$

Ordenação Bolha

Análise

1. Percorrer o vetor da esquerda para a direita; $O(n)$
2. a cada passo, ordenar o elemento atual e o próximo à direita; $O(1)$
3. repetir o processo até que o vetor esteja ordenado. $O(n)$

Ordenação Bolha

Análise

1. Percorrer o vetor da esquerda para a direita; $O(n)$
2. a cada passo, ordenar o elemento atual e o próximo à direita; $O(1)$
3. repetir o processo até que o vetor esteja ordenado. $O(n)$

Complexidade: $O(n^2)$ e $\Omega(n^2)$

Ordenação Bolha

Código em C (melhorado)

```
int i, j, t, melhorou;

for( i = 0; i < n-1 ;  i++ )
{
    melhorou = 0;
    for( j=0; j < n-i+1; j++ )
    {
        if( array[j] > array[j+1] )
        {
            t          = array[j];
            array[j]   = array[j+1];
            array[j+1] = t;
            melhorou = 1;
        }
    }
    if(!melhorou)
        break;
}
```

Ordenação Bolha

Análise

1. Percorrer o vetor da esquerda para a direita; $O(n)$
2. a cada passo, ordenar o elemento atual e o próximo à direita; $O(1)$
3. repetir o processo até que o vetor esteja ordenado. $O(n)$

Complexidade: $O(n^2)$ e $\Omega(n)$

Ordenação

- vetores com dados complexos (strings, structs, etc)

```
struct NotaAluno
{
    char* nome;
    int nota1;
    int nota2;
    float media;
};
```

Ordenação Bolha

Código em C

```
NotaAluno array[1000];
int i, j; NotaAluno t;
for( i = 0; i < n-1 ; i++)
    for( j=0; j < n-i+1; j++) {
        if( array[j].media > array[j+1].media ) {
            t          = array[j];
            array[j]   = array[j+1];
            array[j+1] = t;
        }
    }
```

Código em C

```
NotaAluno array[1000];
int i, j; NotaAluno t;
for( i = 0; i < n-1 ; i++)
    for( j=0; j < n-i+1; j++) {
        if( array[j].media > array[j+1].media ) {
            t          = array[j];
            array[j]   = array[j+1];
            array[j+1] = t;
        }
    }
```

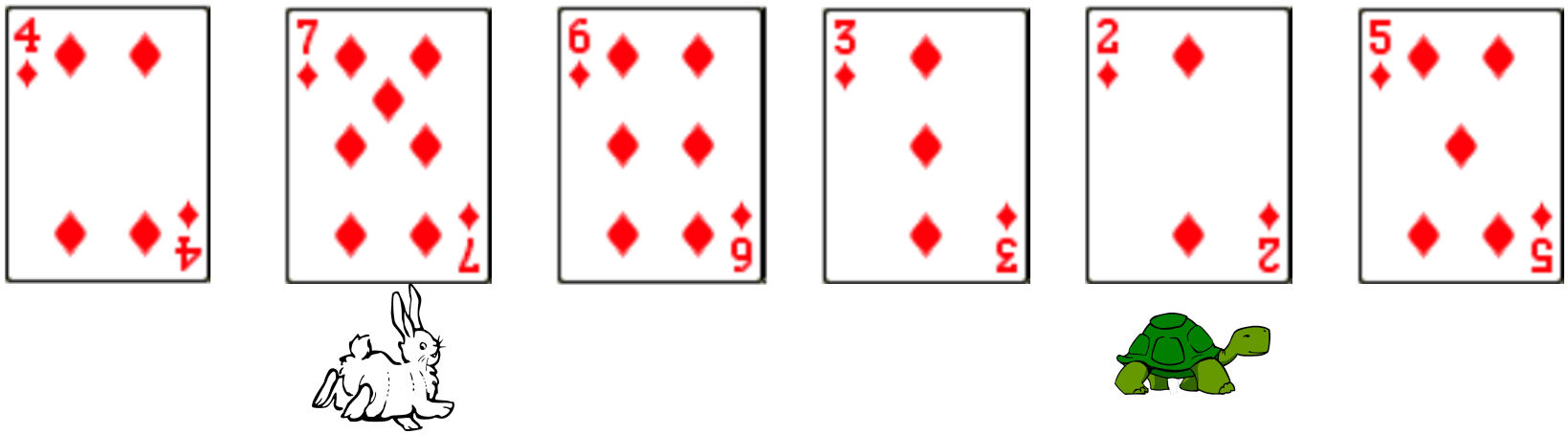
- como não perder eficiência nas cópias?

Ordenação Bolha

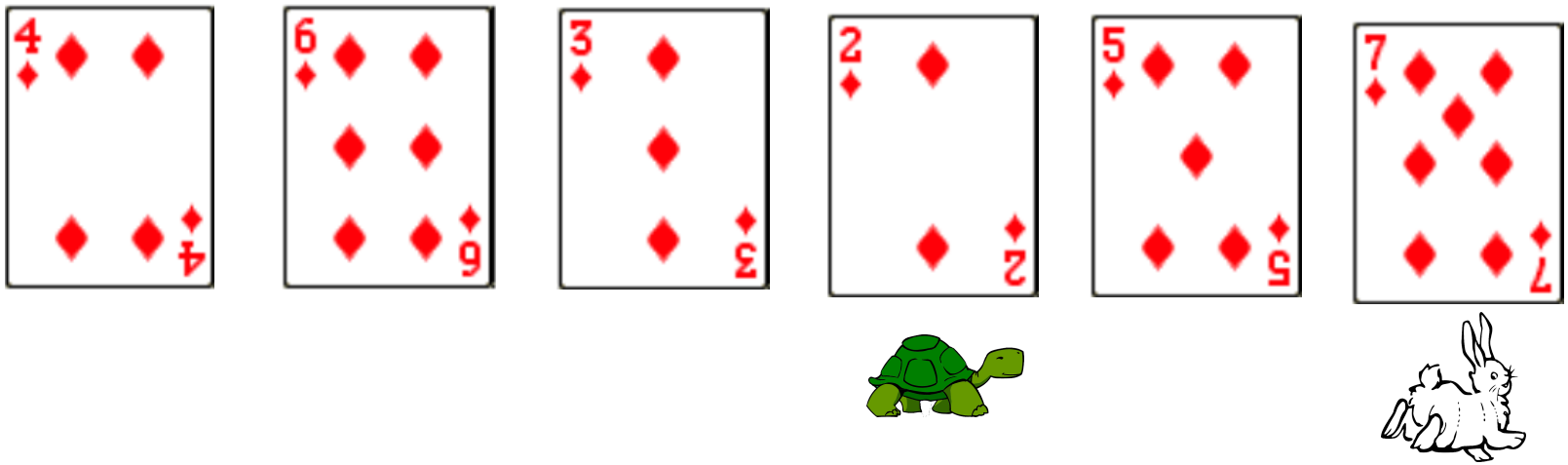
Código em C

```
NotaAluno* array[1000];
// inicializa valores dos alunos
int i, j; NotaAluno* t;
for( i = 0; i < n-1 ; i++)
    for( j=0; j < n-i+1; j++) {
        if(array[j]->media > array[j+1]->media) {
            t          = array[j];
            array[j]   = array[j+1];
            array[j+1] = t;
        }
    }
```

Problema da Bolha



Problema da Bolha



Melhorias

- Presença de tartarugas atrapalham algoritmo (maior número de iterações);

Ordenação Bolha

Melhorias

- Presença de tartarugas atrapalham algoritmo (maior número de iterações);
- *Cocktail Sort* tenta "consertar" o problema. Como?

Melhorias

- Presença de tartarugas atrapalham algoritmo (maior número de iterações);
- *Cocktail Sort* tenta "consertar" o problema. Como?
- Percorrendo vetor nas duas direções!

Ordenação Bolha

Melhorias

- Presença de tartarugas atrapalham algoritmo (maior número de iterações);
- *Cocktail Sort* tenta "consertar" o problema. Como?
- Percorrendo vetor nas duas direções!
- Complexidade fica igual...

Ideia

Extensão da Ordenação Bolha

1. **Definir um coeficiente de encolhimento (shrink) estritamente maior que** 1 **(ex.:** $1,3$**) e um gap inicial (ex.:** $n/1,3$**);**

Ideia

Extensão da Ordenação Bolha

1. **Definir um coeficiente de encolhimento (shrink) estritamente maior que** 1 **(ex.:** $1,3$**) e um gap inicial (ex.:** $n/1,3$**)**;
2. Percorrer o vetor da esquerda para a direita;

Ideia

Extensão da Ordenação Bolha

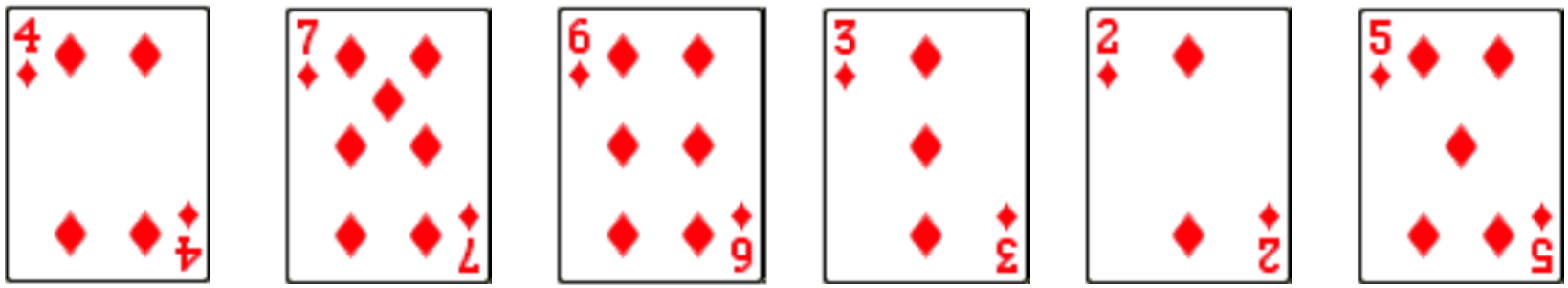
1. **Definir um coeficiente de encolhimento (shrink) estritamente maior que** 1 **(ex.:** $1,3$**) e um gap inicial (ex.:** $n/1,3$**)**;
2. Percorrer o vetor da esquerda para a direita;
3. a cada passo, ordenar o elemento atual e ~~o próximo~~ **algum outro** à direita, **de acordo com o gap atual**;

Ideia

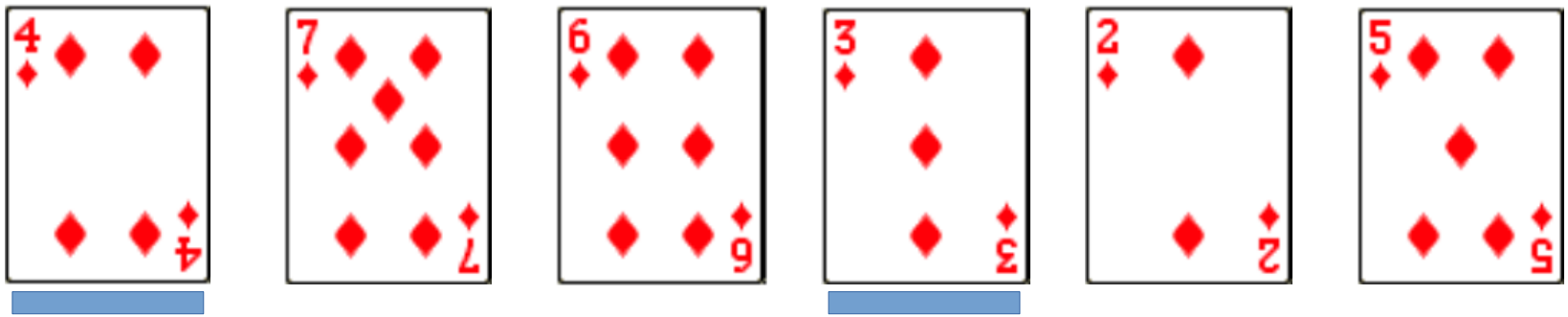
Extensão da Ordenação Bolha

1. **Definir um coeficiente de encolhimento (shrink) estritamente maior que** 1 **(ex.:** $1,3$**) e um gap inicial (ex.:** $n/1,3$**)**;
2. Percorrer o vetor da esquerda para a direita;
3. a cada passo, ordenar o elemento atual e ~~o próximo~~ **algum outro** à direita, **de acordo com o gap atual**;
4. repetir o processo **diminuindo o gap** até que o vetor esteja ordenado.

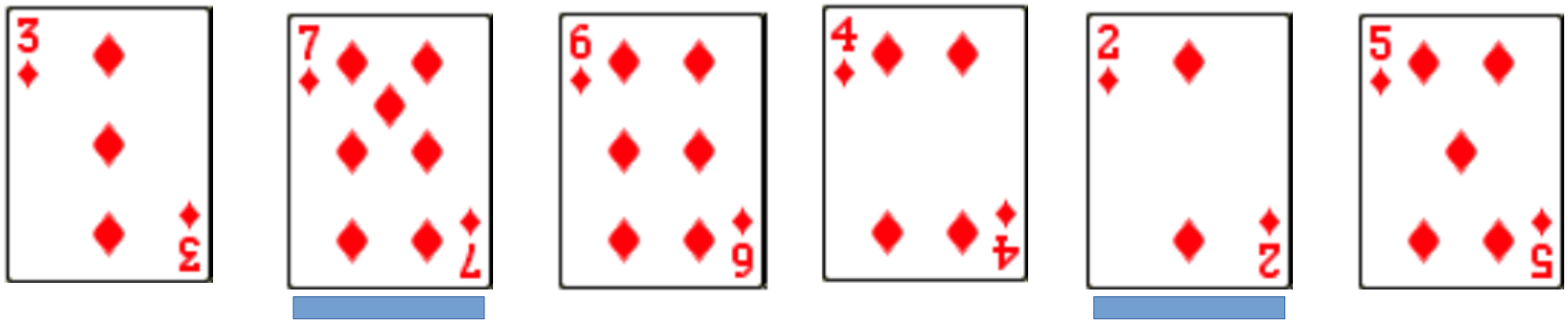
Exemplo



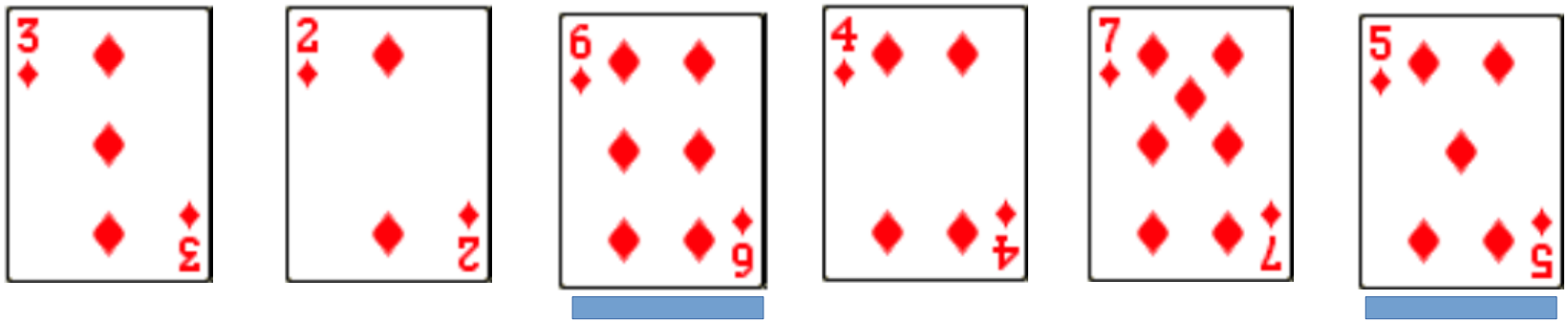
Exemplo (gap 3)



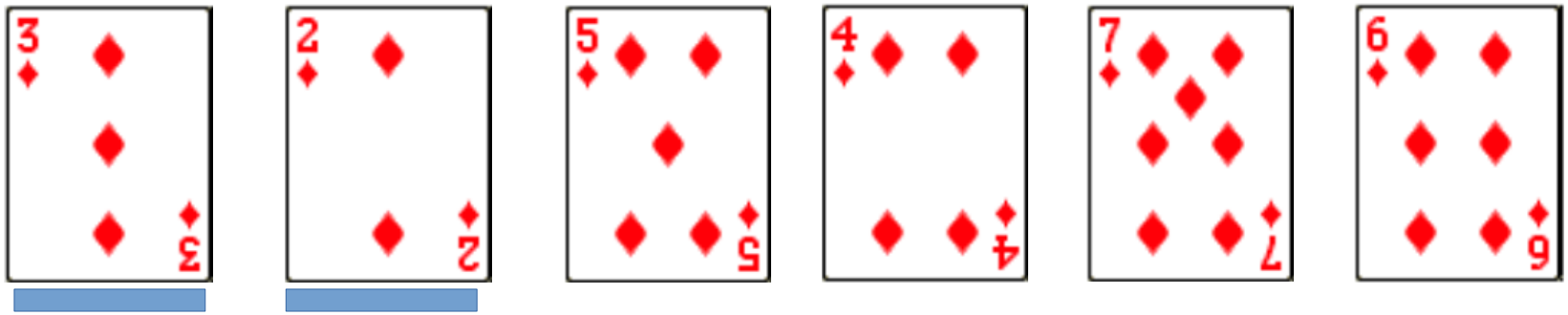
Exemplo (gap 3)



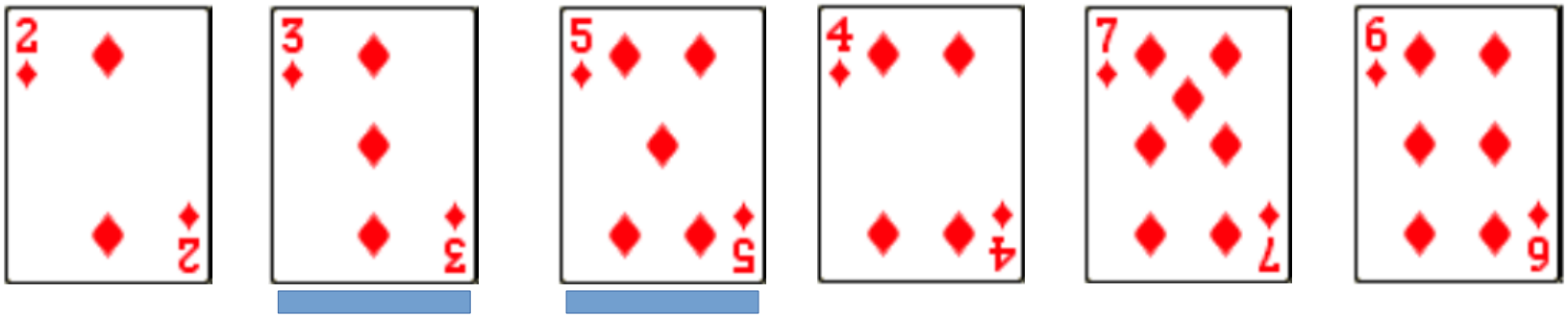
Exemplo (gap 3)



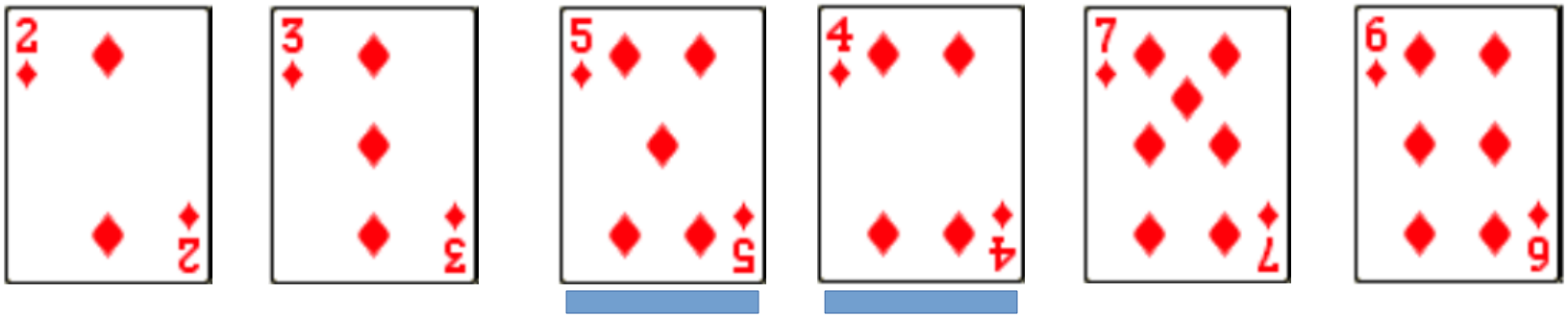
Exemplo (gap 1) – Bolha



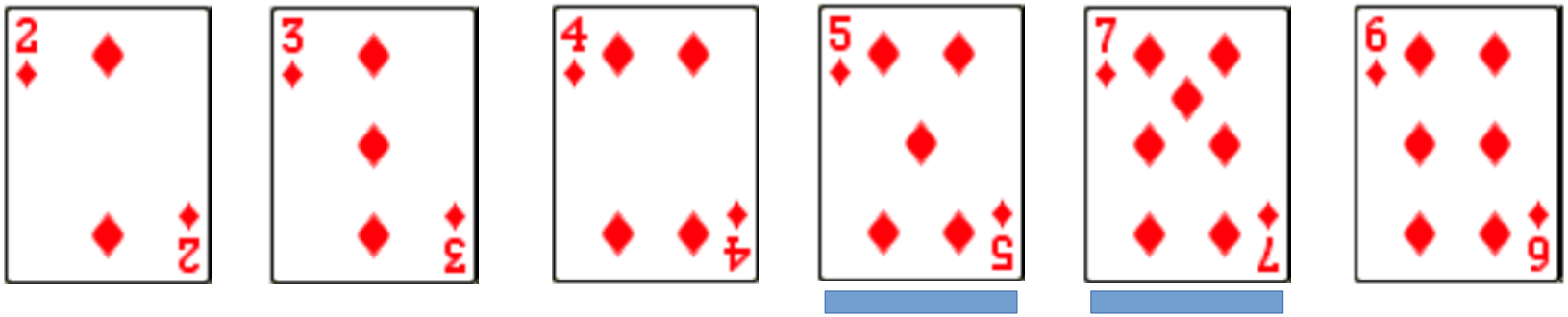
Exemplo (gap 1) – Bolha



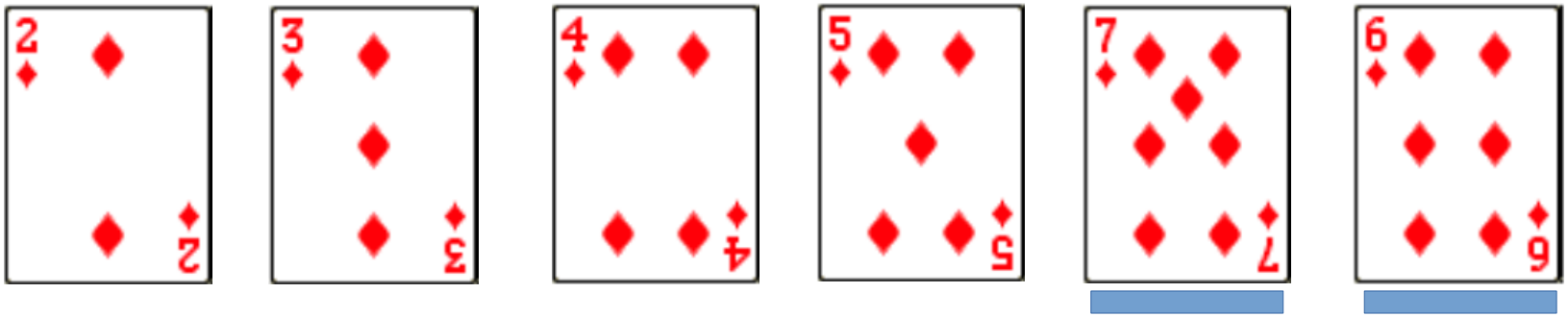
Exemplo (gap 1) – Bolha



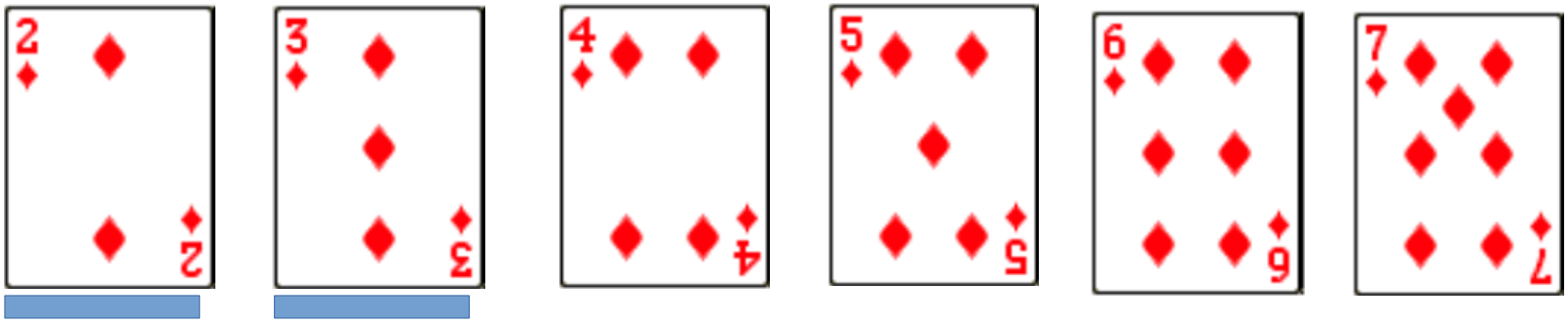
Exemplo (gap 1) – Bolha



Exemplo (gap 1) – Bolha



Exemplo (gap 1) – Bolha



Análise

- Complexidade igual à Bolha $O(n^2)$, $\Omega(n)$

Análise

- Complexidade igual à Bolha $O(n^2)$, $\Omega(n)$
- Fator de encolhimento empírico com bons resultados $(1, 3)$

Análise

- Complexidade igual à Bolha $O(n^2)$, $\Omega(n)$
- Fator de encolhimento empírico com bons resultados $(1, 3)$
- Performance comparável aos melhores ordenadores.

Ideia

Extensão da Ordenação por Inserção

1. **Definir uma sequencia de intervalos (gaps) para ordenação**;

Ideia

Extensão da Ordenação por Inserção

1. **Definir uma sequencia de intervalos (gaps) para ordenação**;
2. Percorrer o vetor da esquerda para a direita;

Ideia

Extensão da Ordenação por Inserção

1. **Definir uma sequencia de intervalos (gaps) para ordenação**;
2. Percorrer o vetor da esquerda para a direita;
3. a cada passo, considerar um subvetor com o elemento atual e todos dentro do intervalo corrente de ordenação **de acordo com o gap atual**;

Ideia

Extensão da Ordenação por Inserção

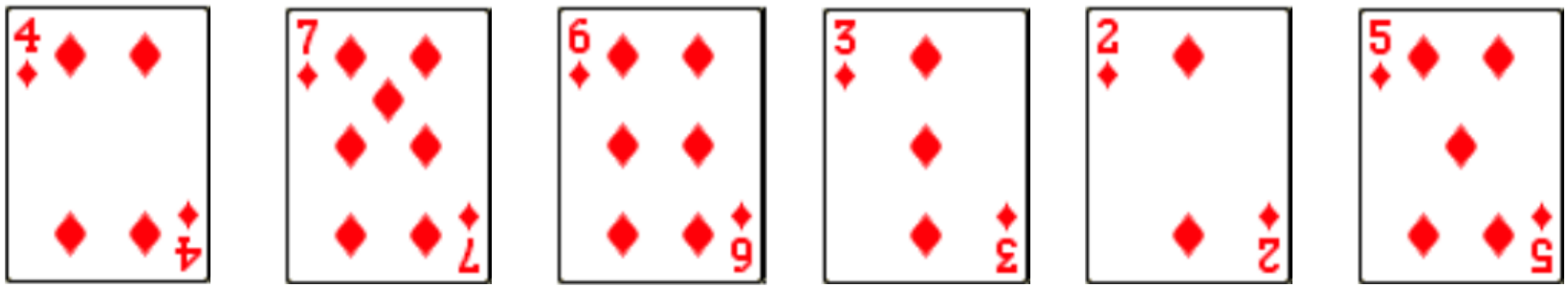
1. **Definir uma sequencia de intervalos (gaps) para ordenação**;
2. Percorrer o vetor da esquerda para a direita;
3. a cada passo, considerar um subvetor com o elemento atual e todos dentro do intervalo corrente de ordenação **de acordo com o gap atual**;
4. ordenar o subvetor utilizando inserções;

Ideia

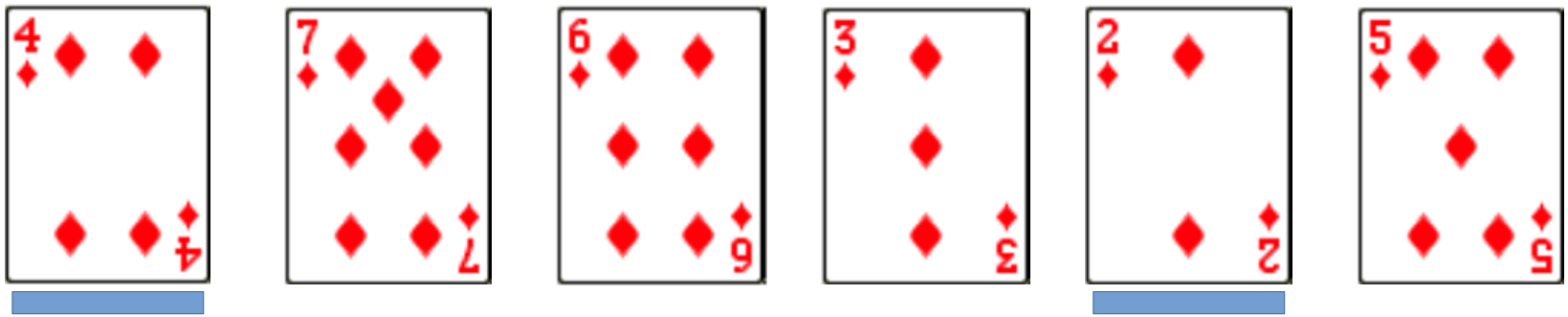
Extensão da Ordenação por Inserção

1. **Definir uma sequencia de intervalos (gaps) para ordenação**;
2. Percorrer o vetor da esquerda para a direita;
3. a cada passo, considerar um subvetor com o elemento atual e todos dentro do intervalo corrente de ordenação **de acordo com o gap atual**;
4. ordenar o subvetor utilizando inserções;
5. repetir o processo **diminuindo o gap** até que o vetor esteja ordenado.

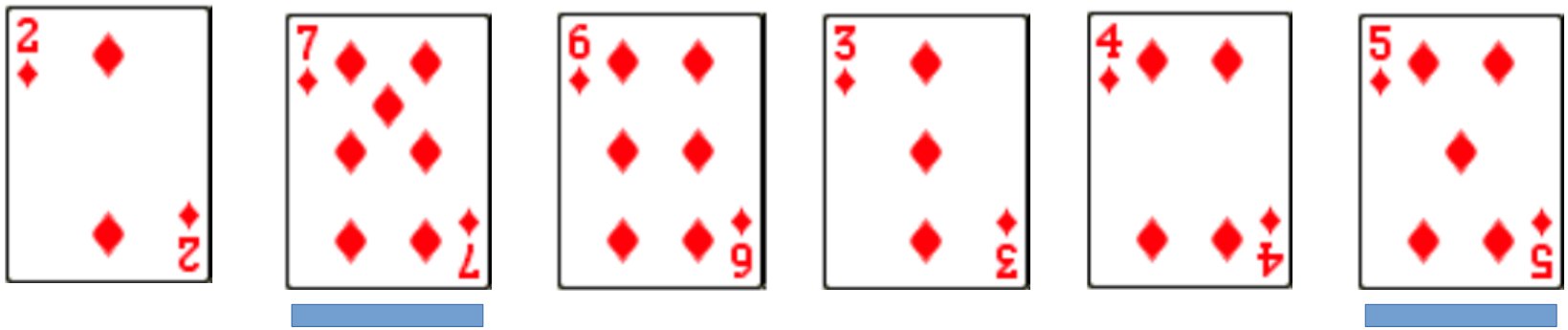
Exemplo



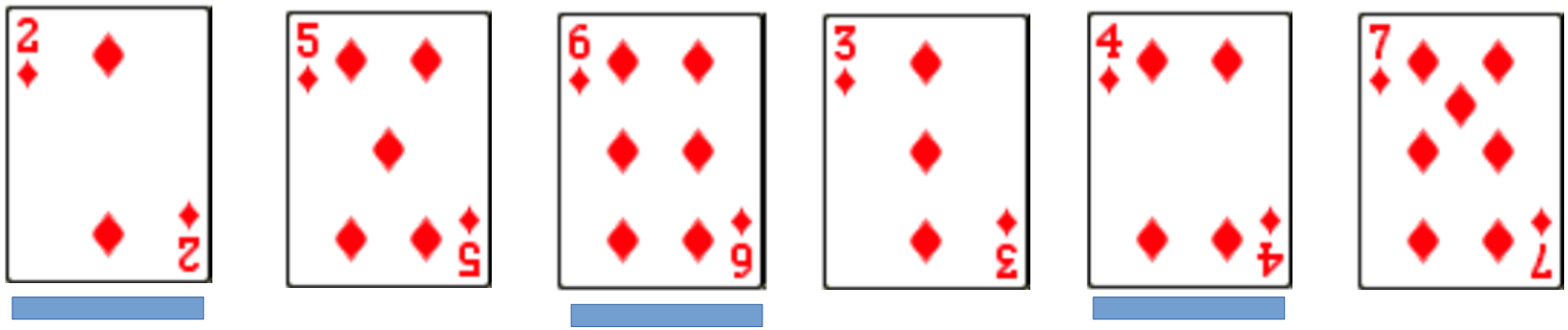
Exemplo (gap 4)



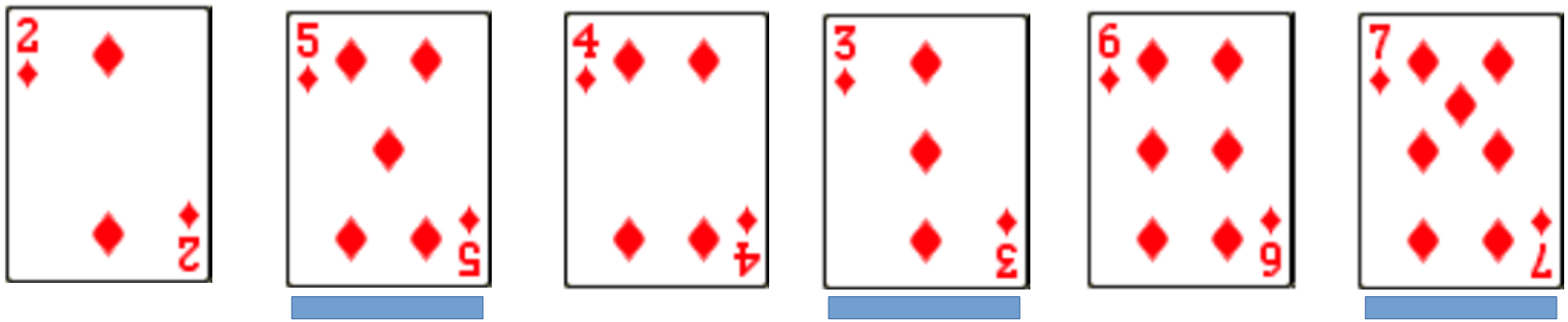
Exemplo (gap 4)



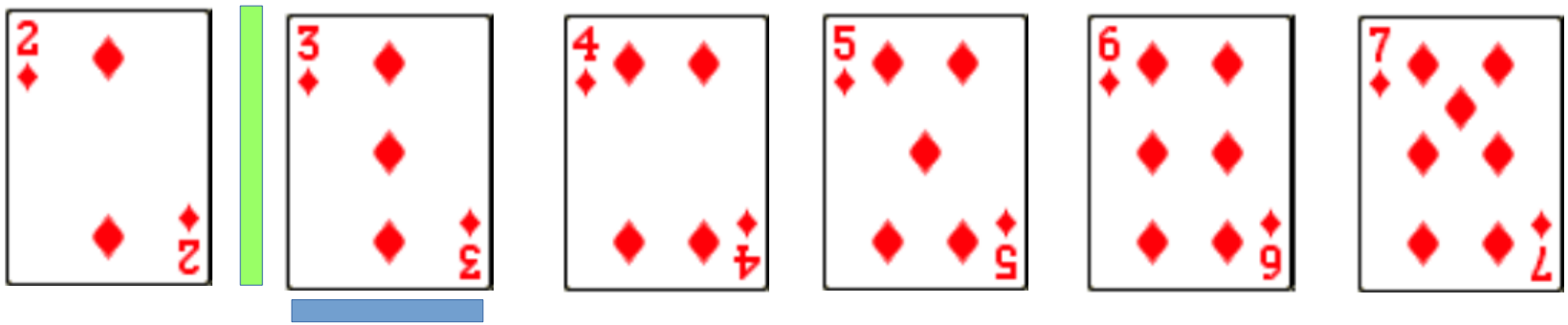
Exemplo (gap 2)



Exemplo (gap 2)



Exemplo (gap 1) - Ordenação por Inserção



Qual sequencia usar?

- Problema em aberto!

Qual sequencia usar?

- Problema em aberto!
- Complexidade depende da sequencia;

Qual sequencia usar?

- Problema em aberto!
- Complexidade depende da sequencia;
- Sequencia de divisões de potências de dois é $O(n^2)$ (artigo original);

Qual sequencia usar?

- Problema em aberto!
- Complexidade depende da sequencia;
- Sequencia de divisões de potências de dois é $O(n^2)$ (artigo original);
- Evoluiu para $O(n^{3/2})$, $O(nlog^2n)$, $O(n^{4/3})$,...;

Qual sequencia usar?

- Problema em aberto!
- Complexidade depende da sequencia;
- Sequencia de divisões de potências de dois é $O(n^2)$ (artigo original);
- Evoluiu para $O(n^{3/2})$, $O(nlog^2n)$, $O(n^{4/3})$,...;
- Atualmente (2001) a melhor sequencia é de Ciura: 1, 4, 10, 23, 57, 132, 301, 701;

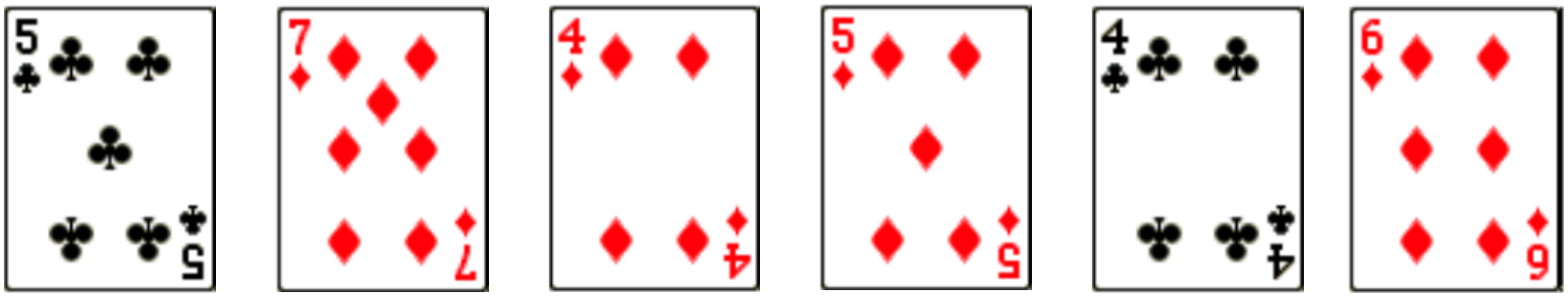
Qual sequencia usar?

- Problema em aberto!
- Complexidade depende da sequencia;
- Sequencia de divisões de potências de dois é $O(n^2)$ (artigo original);
- Evoluiu para $O(n^{3/2})$, $O(nlog^2n)$, $O(n^{4/3})$,...;
- Atualmente (2001) a melhor sequencia é de Ciura: 1, 4, 10, 23, 57, 132, 301, 701;
- $O(???)$!

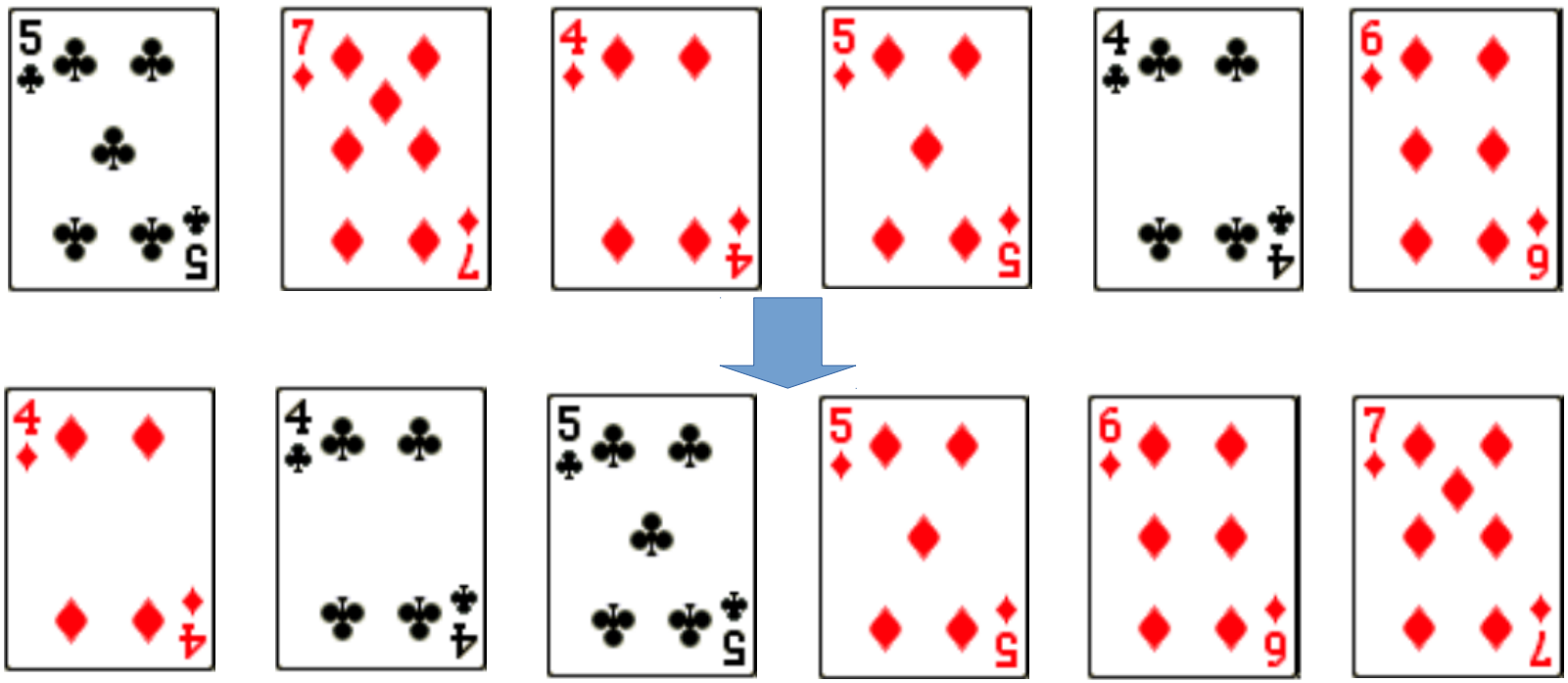
Qual sequencia usar?

- Problema em aberto!
- Complexidade depende da sequencia;
- Sequencia de divisões de potências de dois é $O(n^2)$ (artigo original);
- Evoluiu para $O(n^{3/2})$, $O(nlog^2n)$, $O(n^{4/3})$,...;
- Atualmente (2001) a melhor sequencia é de Ciura: 1, 4, 10, 23, 57, 132, 301, 701;
- $O(???)$!
- Performance muitas vezes equivalente aos melhores ordenadores.

Valores repetidos



Valores repetidos



Estabilidade

Quem é o que?

- Ordenação Bolha é estável;

Estabilidade

Quem é o que?

- Ordenação Bolha é estável;
- Ordenação por Inserção é estável;

Estabilidade

Quem é o que?

- Ordenação Bolha é estável;
- Ordenação por Inserção é estável;
- *Combsort* não é estável;

Quem é o que?

- Ordenação Bolha é estável;
- Ordenação por Inserção é estável;
- *Combsort* não é estável;
- *Shellsort* não é estável.

Exercício prático

Exercício prático: implementar e comparar os métodos para diversos vetores aleatórios (linguagem C ou outra preferida).

Instituto de computação

Referências

- Imagens gratuitas no pixabay
- Wikipedia (português) para *bubble sort* e *insertion sort*
- Wikipedia (inglês) para *bubble sort* e *insertion sort*