

Classificação e Pesquisa de Dados

Cristiano Santos

cristiano.santos@amf.edu.br

OBJETIVO

- Capacitar o aluno na **análise e seleção de algoritmos de classificação de dados e de pesquisa de dados**
- Específicos:
 - **Conhecer os principais conceitos e técnicas para implementação de algoritmos de classificação e pesquisa de dados existentes e comparar a sua eficácia em diferentes conjuntos de dados;**
 - **Compreender os aspectos introdutórios do processo de análise de complexidade de algoritmos com base nos algoritmos estudados na disciplina, sabendo determinar os que apresentam o melhor e o pior custo em termos de tempo e espaço;**
 - **Implementar algoritmos de manipulação de dados e analisar sua complexidade para solucionar problemas específicos.**

O que é?

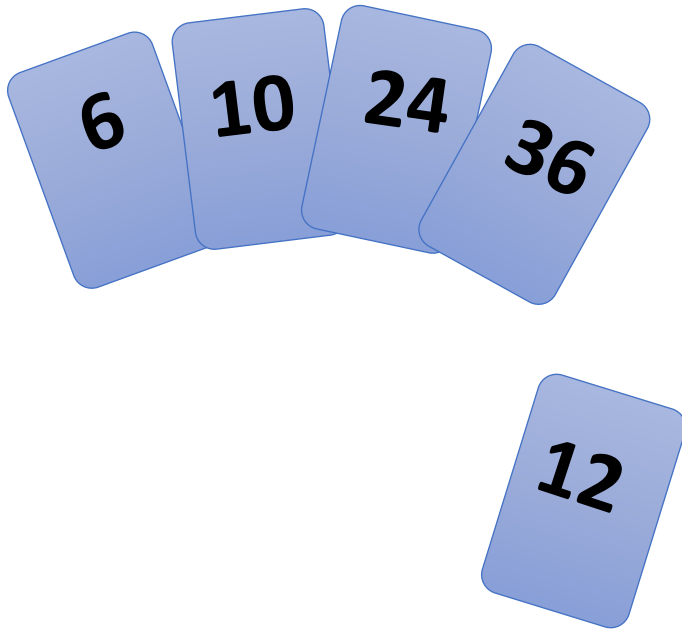
- Analise de algoritmos
 - Em ciência da computação, é a área de pesquisa cujo o foco são os algoritmos.
- Busca responder:
 - Podemos fazer algoritmos mais eficientes?

Ordenação (Sort)

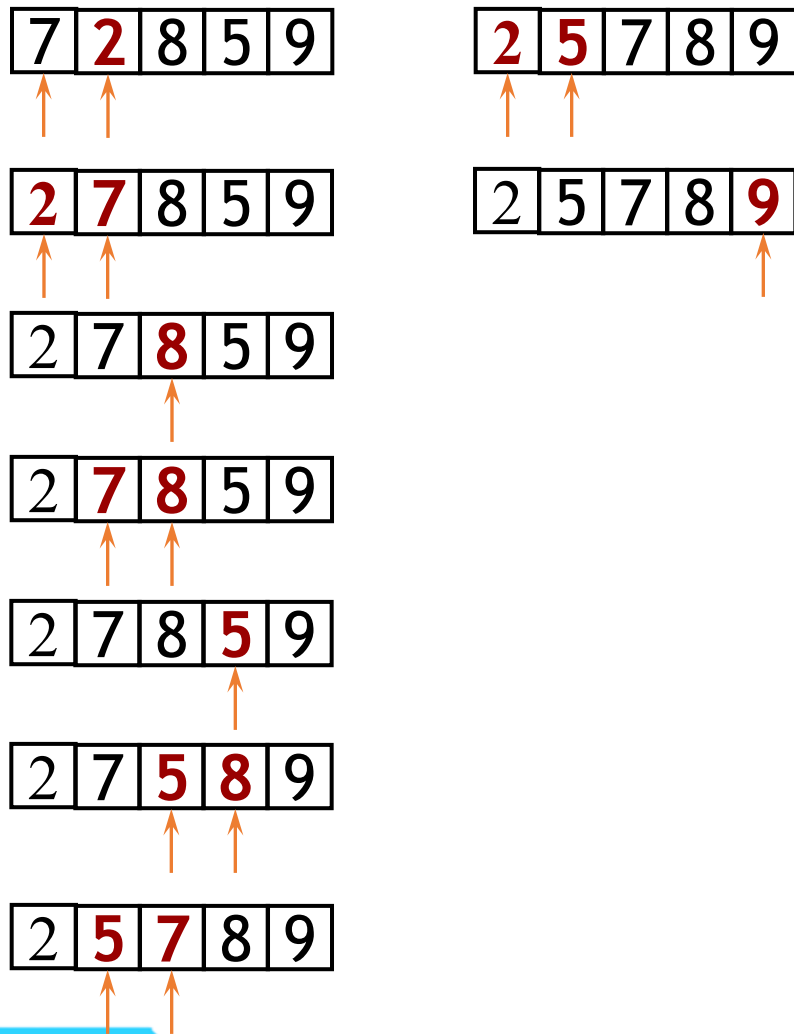
- Dado um conjunto de dados armazenados em ordem **aleatória**, reordena-los com base em uma **regra de precedência**
 - Qual a regra de precedência para:
 - Números
 - Nomes
 - Alunos

Método de Ordenação

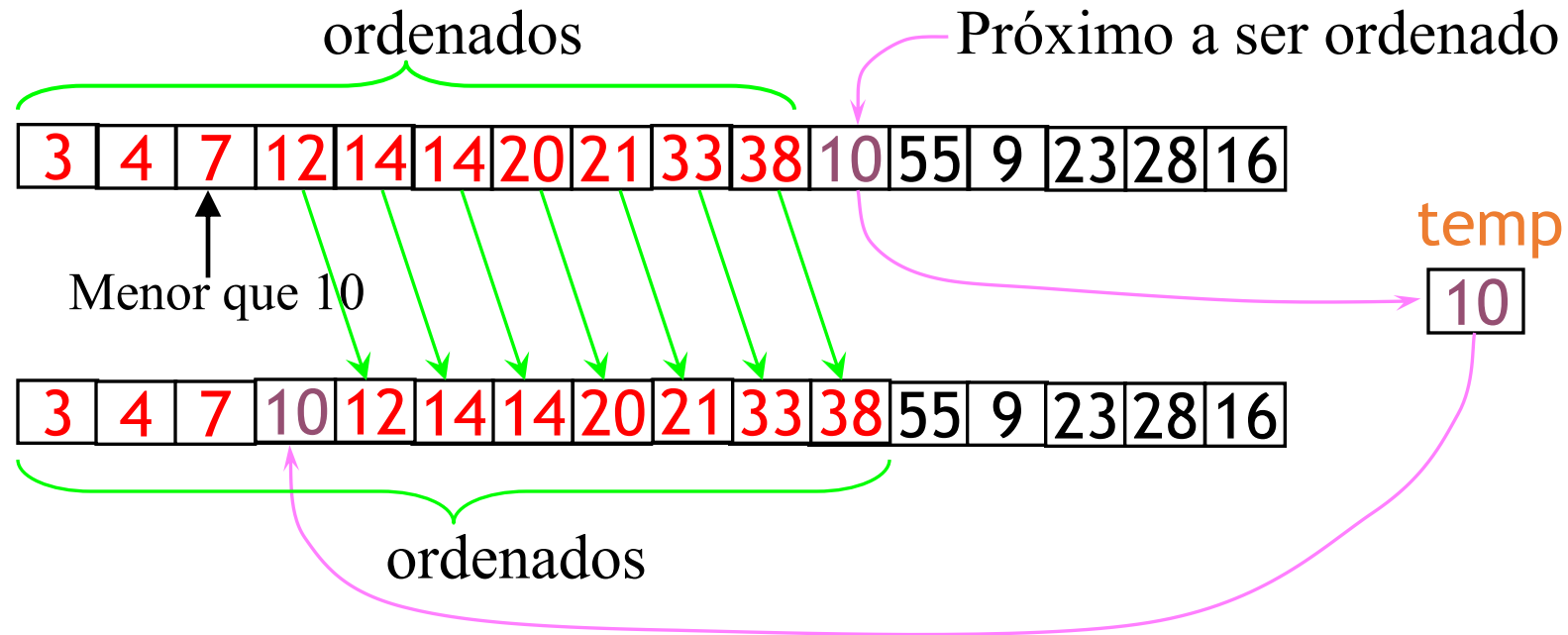
- Método do jogador de cartas



Ordenação (Exemplo)



Um passo de ordenação



Algoritmos e Complexidade

- Eficiência de um algoritmo
 - Complexidade de tempo: **quanto “tempo”** é necessário para computar o resultado para uma instância do problema de **tamanho n**
 - **Pior caso:** Considera-se a instância que faz o algoritmo funcionar mais lentamente
 - **Caso médio:** Considera-se todas as possíveis instâncias e mede-se o tempo médio
- Eficiência de uma estrutura de dados
 - Complexidade de **espaço**: quanto “espaço de memória/disco” é preciso para armazenar a estrutura (pior caso e caso médio)
- Complexidade de **espaço e tempo** estão frequentemente relacionadas

Algoritmo

```
algoritmoSort (int data[], int n) {  
    int tmp,i,j;  
    for (j=1; j<n; j++) {  
        i =j - 1;  
        tmp = data[j];  
        while ( (i>=0) && (tmp < data[i]) ) {  
            data[i+1] = data[i];  
            i--;  
        }  
        data[i+1] = tmp;  
    }  
}
```

Algoritmo

```
algoritmoSort (int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i =j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```

3	2	1	4	5
---	---	---	---	---

Algoritmo

```
algoritmoSort (int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```

3	2	1	4	5
---	---	---	---	---



Algoritmo

```
algoritmoSort (int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i =j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

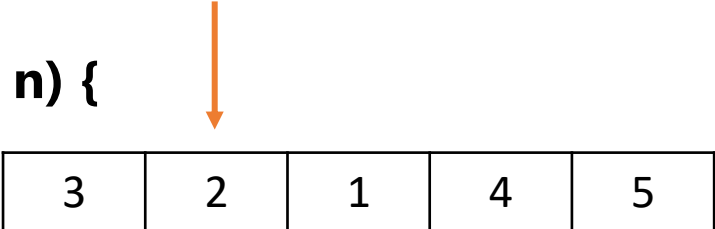
```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```



3	2	1	4	5
---	---	---	---	---

Algoritmo

```
algoritmoSort (int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i =j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

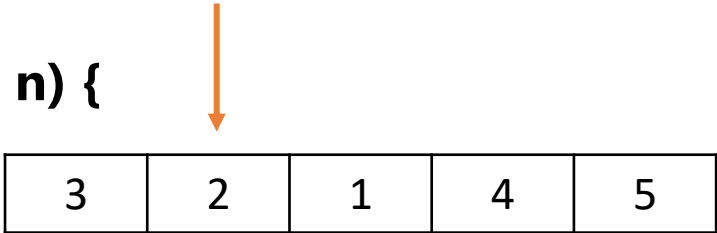
```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

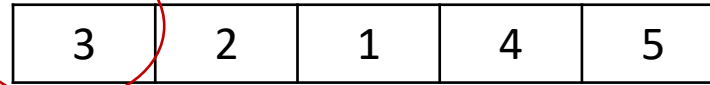
```
}
```



3	2	1	4	5
---	---	---	---	---

Algoritmo

```
algoritmoSort (int data[], int n) {  
    int tmp,i,j;  
    for (j=1; j<n; j++) {  
        i = j - 1;  
        tmp = data[j];  
        while ( (i>=0) && (tmp < data[i])) {  
            data[i+1] = data[i];  
            i--;  
        }  
        data[i+1] = tmp;  
    }  
}
```



Algoritmo

```
algoritmoSort (int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i =j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

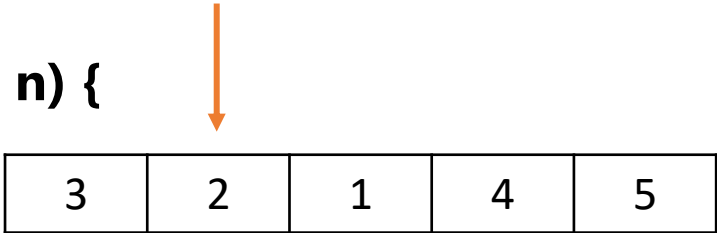
```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```



3	2	1	4	5
---	---	---	---	---

Algoritmo

```
algoritmoSort (int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i =j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```


```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```



3	3	1	4	5
---	---	---	---	---

Algoritmo

```
algoritmoSort (int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

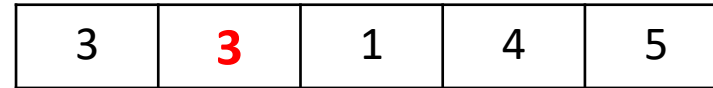
```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```



3	3	1	4	5
---	---	---	---	---

Algoritmo

```
algoritmoSort (int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i =j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

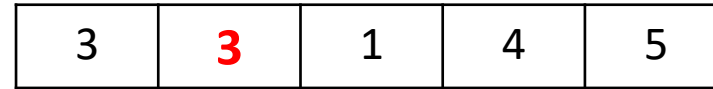
```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```



3	3	1	4	5
---	---	---	---	---

Algoritmo

```
algoritmoSort (int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

```
            i--;
```

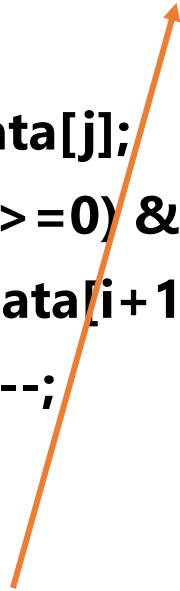
```
        }//while
```

```
        data[i+1] = tmp; // -1 + 1 = 0
```

```
    }//for
```

```
}
```

2	3	1	4	5
---	---	---	---	---



Estudo de caso

- Qual o melhor caso?
 - Os dados já estarem ordenados

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

- Qual o pior?
 - Os dados estarem na ordem inversa

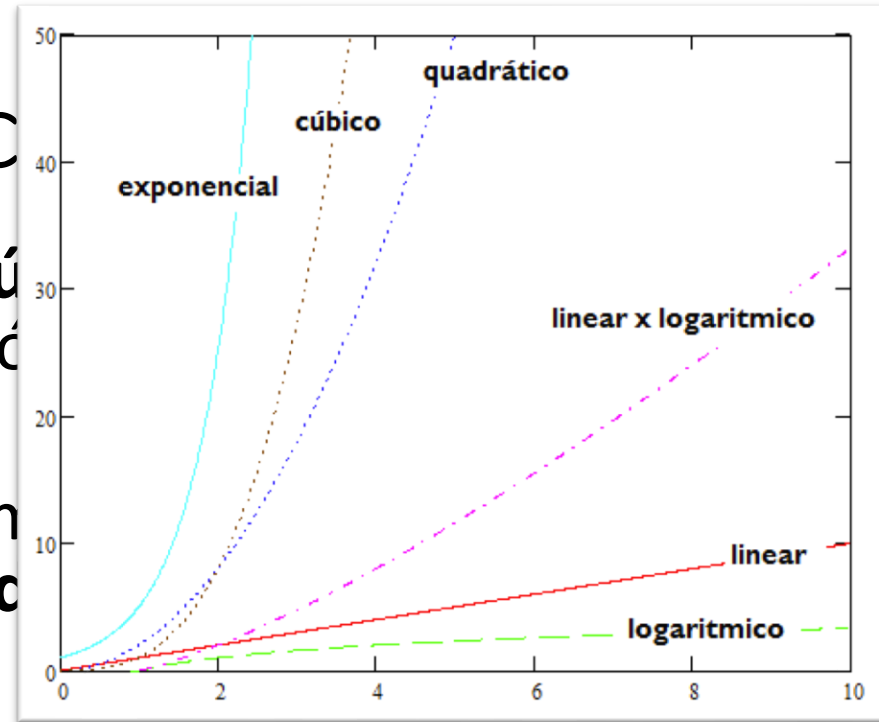
9	8	7	6	5	4	3	2	1
---	---	---	---	---	---	---	---	---

Algoritmos e Complexidade

- Eficiência medida objetivamente depende de:
 - Como o programador implementou o algoritmo/ED
 - Características do computador usado para fazer experimentos:
 - Velocidade da CPU
 - Capacidade e velocidade de acesso à memória primária / secundária
 - Etc
 - Linguagem / Compilador / Sistema Operacional / etc
- Portanto, a **medição formal** de complexidade tem que ser **subjetiva**, porém matematicamente **consistente**
⇒ Complexidade assintótica

Complexidade Assintótica

- Tempo / espaço medidos em **número** de instruções / “palavras” de memória / bytes
- Análise do algoritmo / E.D. **depende** do tamanho da **entrada** armazenados (n).
 - Ex.: $T(n) = 13n^3 + 2n^2 + 6n \log n$
- Percebe-se que à medida que n aumenta, o termo **cúbico** começa a **dominar**
- A **constante que multiplica o termo cúbico** tem relativamente a mesma **importância** que a velocidade da CPU / memória
- Diz-se que $T(n) \in O(n^3)$



Limites

- Definição:

$T(n) \in O(f(n))$ se existem constantes c e n_0 tais que
 $T(n) \leq c f(n)$ para todo $n \geq n_0$

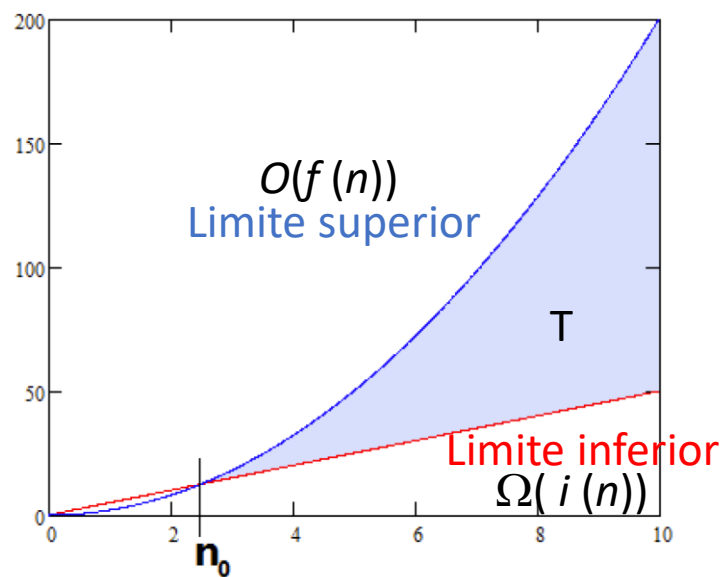


Figura 2. Algoritmo com limites assintóticos superior e inferior quadrático e linear, respectivamente. Estes dois limites correspondem ao melhor caso e ao pior caso, mas a área em azul compreende todos os outros possíveis casos.

Limites Justos

- Observe que se $T(n) \in O(n^3)$ então:
 - $T(n) \in O(n^4)$, $T(n) \in O(n^5)$, etc
- Analogamente, se $T(n) \in \Omega(n^3)$ então:
 - $T(n) \in \Omega(n^2)$, $T(n) \in \Omega(n)$, etc
- Se uma função $T(n)$ tem como limites superior e inferior a mesma função $f(n)$, então:
 - $f(n)$ é um **limite justo** de $T(n)$, ou $T(n) \in \Theta(f(n))$

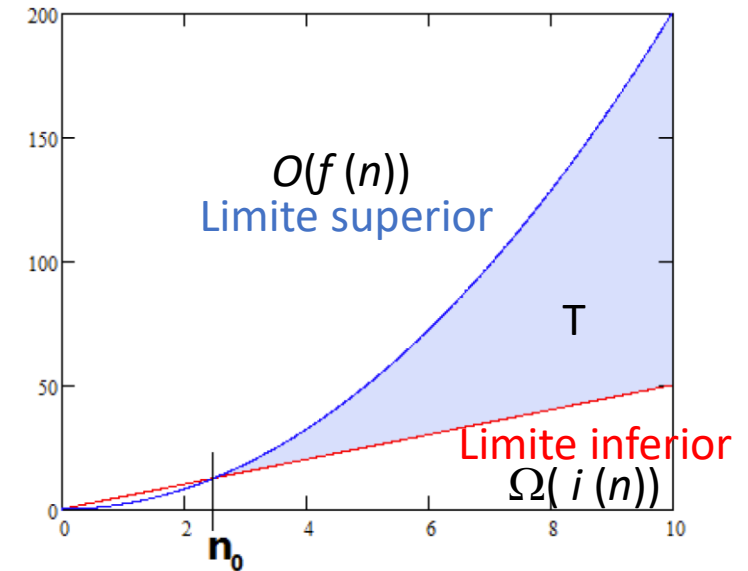


Figura 2. Algoritmo com limites assintóticos superior e inferior quadrático e linear, respectivamente. Estes dois limites correspondem ao melhor caso e ao pior caso, mas a área em azul compreende todos os outros possíveis casos.

Inventário de funções de complexidade

- $T(n) \in O(1)$: constante – mais rápido, impossível
- $T(n) \in O(\log \log n)$: super-rápido
- $T(n) \in O(\log n)$: logarítmico – muito bom
- $T(n) \in O(n)$: linear – é o melhor que se pode esperar se algo não pode ser determinado sem examinar toda a entrada
- $T(n) \in O(n \log n)$: limite de muitos problemas práticos, ex.: ordenar uma coleção de números
- $T(n) \in O(n^2)$: quadrático
- $T(n) \in O(n^k)$: polinomial – ok para n pequeno
- $T(n) \in O(k^n), O(n!), O(n^n)$: exponencial – evite!

Qual a complexidade do
insertion sort?

http://bigocheatsheet.com/

The screenshot shows a web browser window with the URL `http://bigocheatsheet.com/`. The page has a dark navigation bar with links: `Big-O Cheat Sheet`, `Data Structures`, `Sorting`, `Graphs`, `Heaps`, `Chart`, and `Comments`. The main content area has a heading `Know Thy Complexities!` followed by an introductory paragraph and a `Legend` section with color-coded boxes: `Excellent` (green), `Good` (yellow), `Fair` (orange), `Bad` (red), and `Horrible` (dark red). Below this is a section titled `Data Structure Operations` containing a table with 10 data structures and their time and space complexities.

Know Thy Complexities!

Hi there! This webpage covers the space and time Big-O complexities of common algorithms used in Computer Science. When preparing for technical interviews in the past, I found myself spending hours crawling the internet putting together the best, average, and worst case complexities for search and sorting algorithms so that I wouldn't be stumped when asked about them. Over the last few years, I've interviewed at several Silicon Valley startups, and also some bigger companies, like Yahoo, eBay, LinkedIn, and Google, and each time that I prepared for an interview, I thought to myself "Why hasn't someone created a nice Big-O cheat sheet?". So, to save all of you fine folks a ton of time, I went ahead and created one. Enjoy! - Eric

Legend

Excellent Good Fair Bad Horrible

Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Excellent	Good	Good	Good	Good	Good	Good	Good	Good
Stack	Good	Good	Good	Good	Good	Good	Good	Good	Good
Singly-Linked List	Good	Good	Good	Good	Good	Good	Good	Good	Good
Doubly-Linked List	Good	Good	Good	Good	Good	Good	Good	Good	Good
Skip List	Good	Good	Good	Good	Good	Good	Good	Good	Good
Hash Table	-	Good	Good	Good	-	Good	Good	Good	Good
Binary Search Tree	Good	Good	Good	Good	Good	Good	Good	Good	Good
Cartesian Tree	-	Good	Good	Good	-	Good	Good	Good	Good
B-Tree	Good	Good	Good	Good	Good	Good	Good	Good	Good
Red-Black Tree	Good	Good	Good	Good	Good	Good	Good	Good	Good
Splay Tree	-	Good	Good	Good	-	Good	Good	Good	Good

Complexidade por contagem

```
Insertionsort (int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i =j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}//Insertionsort
```

E agora?

E agora?

1

N+1

(1)N+1

(1+1)N+1

(T+1+1)N+1

((1)T+1+1)N+1

((1+1)T+1+1)N+1

((1+1)T+1+1+1)N+1

TN

Essa é a complexidade?

complexidade do algoritmo de cada caso

Qual o T em ambos os casos?

- Qual o melhor caso?
 - Os dados já estarem ordenados

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

$$T = 1 \longrightarrow 1N \longrightarrow \Omega(N)$$

- Qual o pior?
 - Os dados estarem na ordem inversa

9	8	7	6	5	4	3	2	1
---	---	---	---	---	---	---	---	---

$$T = N \longrightarrow NN \longrightarrow O(N^2)$$



Revisão:

Definição: Estruturas de dados

- Uma estrutura de dados **é um meio para armazenar e organizar dados com o objetivo de facilitar o acesso e as modificações.**
- **Nenhuma estrutura de dados única funciona bem para todos os propósitos**, e assim é importante conhecer os pontos fortes e as limitações de várias delas

Ordenação

- Vamos ordenar um lista de números

	 Insertion	 Selection	 Bubble	 Shell	 Merge	 Heap	 Quick	 Quick3
 Random								
 Nearly Sorted								
 Reversed								
 Few Unique								

Somente para números?

- Essas técnicas só funcionam para números?
 - Se definirmos uma função que diga a partir de dois elementos qual o maior, podemos aplicar esses algoritmos para diversos tipos de dados
 - Exemplos?
 - Alfabeto
 - Cerveja
 - ...

Tarefa:

1. Identificar qual algoritmo de ordenação apresentado em cada SLIDE e justificar
2. Descreva/pesquise o que é notação assintótica?
3. Identificar qual o **melhor e o pior caso** (pesquisar/encontrar **qual o melhor e o pior caso de cada algoritmo estudado**)
4. Qual dos algoritmos estudados está(ão) faltando nestes slides?



Qual é o Algoritmo que segue nos próximos?

Algoritmo 1

???

```
(int data[], int n) {  
    int tmp,i,j;  
    for (j=1; j<n; j++) {  
        i =j - 1;  
        tmp = data[j];  
        while ( (i>=0) && (tmp < data[i]) ) {  
            data[i+1] = data[i];  
            i--;  
        }  
        data[i+1] = tmp;  
    }  
}
```

Algoritmo 1

???

```
(int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```

3	2	1	4	5
---	---	---	---	---

Algoritmo 1

???

```
(int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```

3	2	1	4	5
---	---	---	---	---



Algoritmo 1

???

```
(int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

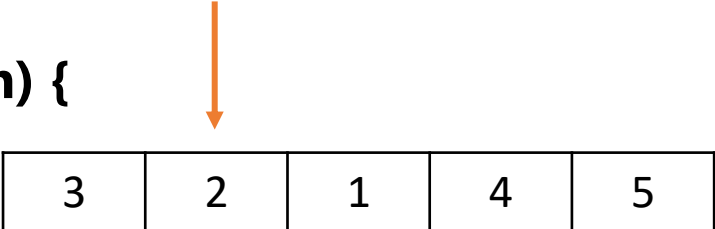
```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```



3	2	1	4	5
---	---	---	---	---

Algoritmo 1

???

```
(int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

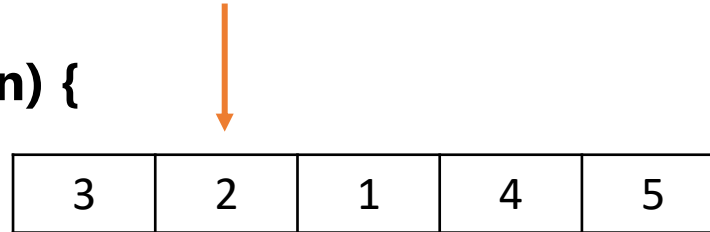
```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```



3	2	1	4	5
---	---	---	---	---

Algoritmo 1

???

```
(int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i])) {
```

```
            data[i+1] = data[i];
```

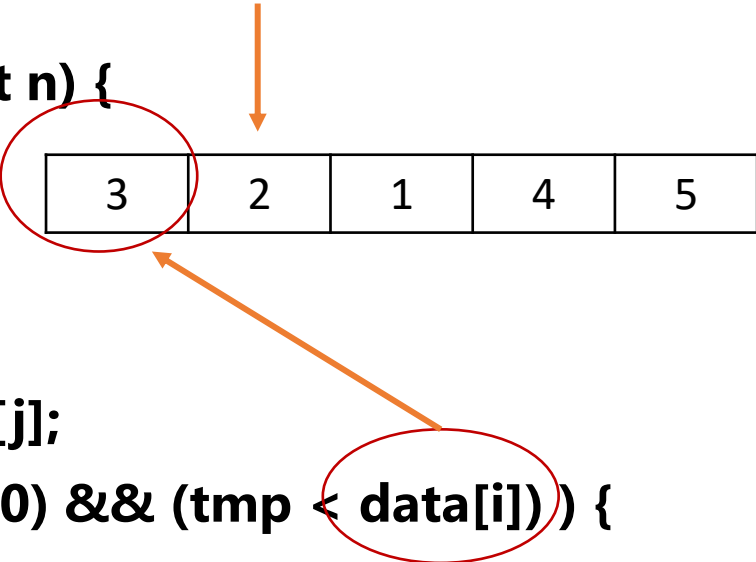
```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```



3	2	1	4	5
---	---	---	---	---

Algoritmo 1

???

```
(int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

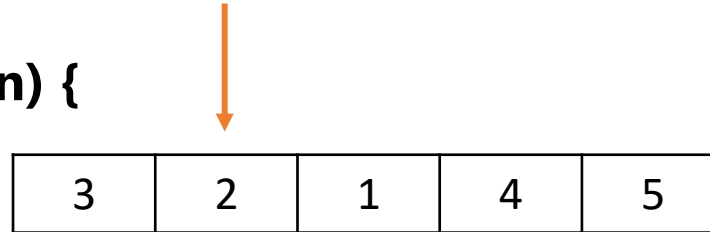
```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```



3	2	1	4	5
---	---	---	---	---

Algoritmo 1

???

```
(int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

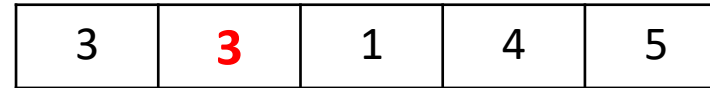
```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```



3	3	1	4	5
---	---	---	---	---

Algoritmo 1

???

```
(int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

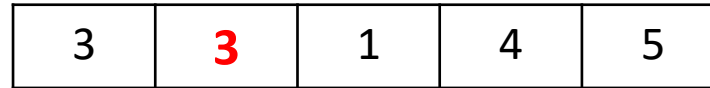
```
            i--;
```

```
        }//while
```

```
        data[i+1] = tmp;
```

```
    }//for
```

```
}
```



3	3	1	4	5
---	---	---	---	---

Algoritmo 1

???

```
(int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i >= 0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

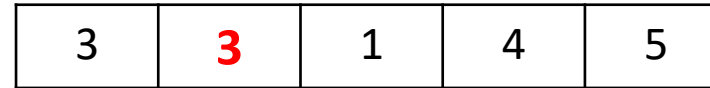
```
            i--;
```

```
        } //while
```

```
        data[i+1] = tmp;
```

```
    } //for
```

```
}
```



3	3	1	4	5
---	---	---	---	---

Algoritmo 1

???

```
(int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

```
            i--;
```

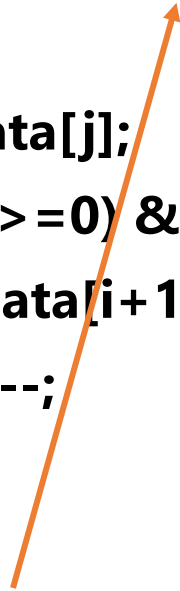
```
        } //while
```

```
        data[i+1] = tmp; // -1 + 1 = 0
```

```
    } //for
```

```
}
```

2	3	1	4	5
---	---	---	---	---



Algoritmo 1

???

```
(int data[], int n) {
```

```
    int tmp,i,j;
```

```
    for (j=1; j<n; j++) {
```

```
        i = j - 1;
```

```
        tmp = data[j];
```

```
        while ( (i>=0) && (tmp < data[i]) ) {
```

```
            data[i+1] = data[i];
```

```
            i--;
```

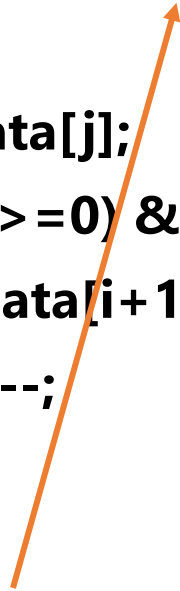
```
        } //while
```

```
        data[i+1] = tmp; // -1 + 1 = 0
```

```
    } //for
```

```
}
```

2	3	1	4	5
---	---	---	---	---





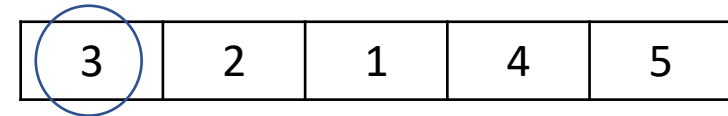
Qual é o Algoritmo que segue nos próximos?

Algoritmo 2

```
??? (int data[], int n) {  
    int menor, troca, i, j, menor_id;  
    for (i=0; i<n-1; i++) {  
        menor = data[i];  
        for (j=i+1; j<n; j++)  
            if (data[j] < menor)  
            {  
                menor = data[j];  
                menor_id = j;  
            }  
        troca = data[i];  
        data[i] = data[menor_id];  
        data[menor_id] = troca;  
    }  
}
```

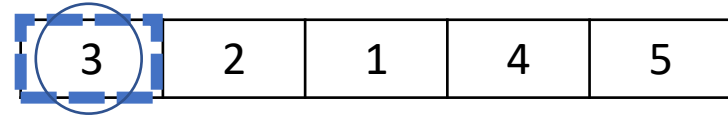

Algoritmo 2

```
??? (int data[], int n) {  
    int menor, troca, i, j, menor_id;  
    for (i=0; i<n-1; i++) {  
        menor = data[i];  
        for (j=i+1; j<n; j++)  
            if (data[j] < menor)  
            {  
                menor = data[j];  
                menor_id = j;  
            }  
        troca = data[i];  
        data[i] = data[menor_id];  
        data[menor_id] = troca;  
    }  
}
```



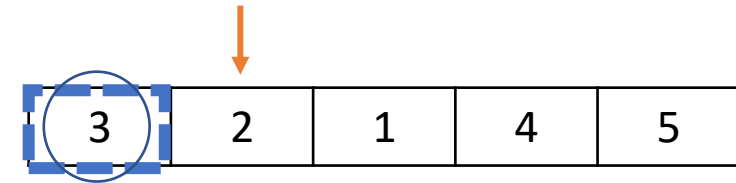
Algoritmo 2

```
??? (int data[], int n) {  
    int menor, troca, i, j, menor_id;  
    for (i=0; i<n-1; i++) {  
        menor = data[i];  
        for (j=i+1; j<n; j++)  
            if (data[j] < menor)  
            {  
                menor = data[j];  
                menor_id = j;  
            }  
        troca = data[i];  
        data[i] = data[menor_id];  
        data[menor_id] = troca;  
    }  
}
```



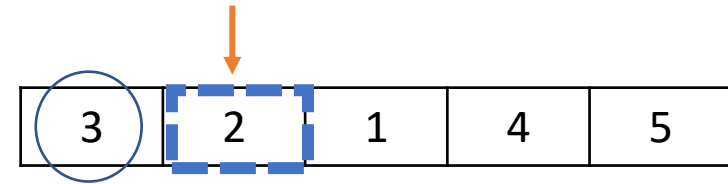
Algoritmo 2

```
??? (int data[], int n) {  
    int menor, troca, i, j, menor_id;  
    for (i=0; i<n-1; i++) {  
        menor = data[i];  
        for (j=i+1; j<n; j++)  
            if (data[j] < menor)  
            {  
                menor = data[j];  
                menor_id = j;  
            }  
        troca = data[i];  
        data[i] = data[menor_id];  
        data[menor_id] = troca;  
    }  
}
```



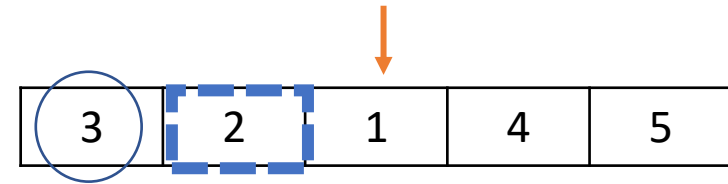
Algoritmo 2

```
??? (int data[], int n) {  
    int menor, troca, i, j, menor_id;  
    for (i=0; i<n-1; i++) {  
        menor = data[i];  
        for (j=i+1; j<n; j++)  
            if (data[j] < menor)  
            {  
                menor = data[j];  
                menor_id = j;  
            }  
        troca = data[i];  
        data[i] = data[menor_id];  
        data[menor_id] = troca;  
    }  
}
```



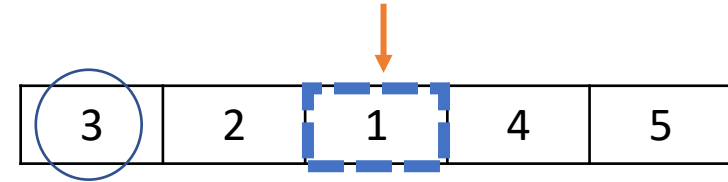
Algoritmo 2

```
??? (int data[], int n) {  
    int menor, troca, i, j, menor_id;  
    for (i=0; i<n-1; i++) {  
        menor = data[i];  
        for (j=i+1; j<n; j++)  
            if (data[j] < menor)  
            {  
                menor = data[j];  
                menor_id = j;  
            }  
        troca = data[i];  
        data[i] = data[menor_id];  
        data[menor_id] = troca;  
    }  
}
```



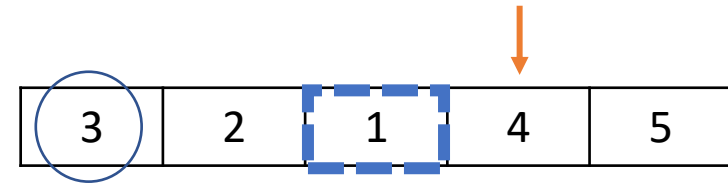
Algoritmo 2

```
??? (int data[], int n) {  
    int menor, troca, i, j, menor_id;  
    for (i=0; i<n-1; i++) {  
        menor = data[i];  
        for (j=i+1; j<n; j++)  
            if (data[j] < menor)  
            {  
                menor = data[j];  
                menor_id = j;  
            }  
        troca = data[i];  
        data[i] = data[menor_id];  
        data[menor_id] = troca;  
    }  
}
```



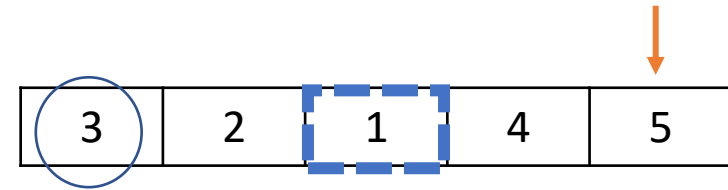
Algoritmo 2

```
??? (int data[], int n) {  
    int menor, troca, i, j, menor_id;  
    for (i=0; i<n-1; i++) {  
        menor = data[i];  
        for (j=i+1; j<n; j++)  
            if (data[j] < menor)  
            {  
                menor = data[j];  
                menor_id = j;  
            }  
        troca = data[i];  
        data[i] = data[menor_id];  
        data[menor_id] = troca;  
    }  
}
```



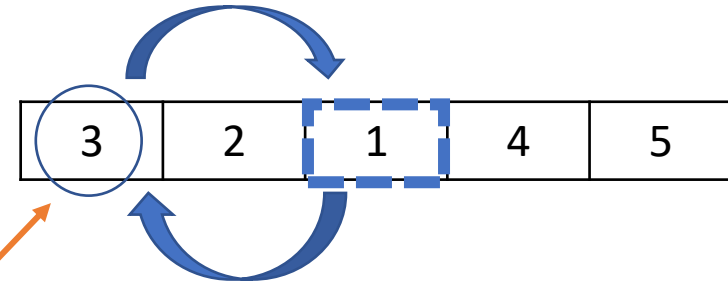
Algoritmo 2

```
??? (int data[], int n) {  
    int menor, troca, i, j, menor_id;  
    for (i=0; i<n-1; i++) {  
        menor = data[i];  
        for (j=i+1; j<n; j++)  
            if (data[j] < menor)  
            {  
                menor = data[j];  
                menor_id = j;  
            }  
        troca = data[i];  
        data[i] = data[menor_id];  
        data[menor_id] = troca;  
    }  
}
```



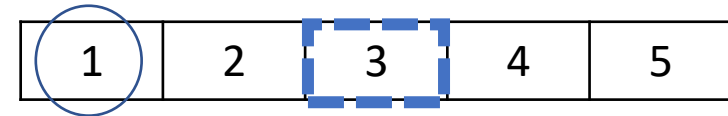
Algoritmo 2

```
??? (int data[], int n) {  
    int menor, troca, i, j, menor_id;  
    for (i=0; i<n-1; i++) {  
        menor = data[i];  
        for (j=i+1; j<n; j++)  
            if (data[j] < menor)  
            {  
                menor = data[j];  
                menor_id = j;  
            }  
        troca = data[i];  
        data[i] = data[menor_id];  
        data[menor_id] = troca;  
    }  
}
```



Algoritmo 2

```
??? (int data[], int n) {  
    int menor, troca, i, j, menor_id;  
    for (i=0; i<n-1; i++) {  
        menor = data[i];  
        for (j=i+1; j<n; j++)  
            if (data[j] < menor)  
            {  
                menor = data[j];  
                menor_id = j;  
            }  
        troca = data[i];  
        data[i] = data[menor_id];  
        data[menor_id] = troca;  
    }  
}
```





Qual é o Algoritmo que segue nos próximos?

Algoritmo 3

```
void ???(int* v, int n) {  
    int i, fim, aux;  
    for (fim = n-1; fim > 0; --fim) {  
        for (i = 0; i < fim; ++i) {  
            if (v[i] > v[i+1]) {  
                aux = v[i];  
                v[i] = v[i+1];  
                v[i+1] = aux;  
            }  
        }  
    }  
}
```

6 5 3 1 8 7 2 4

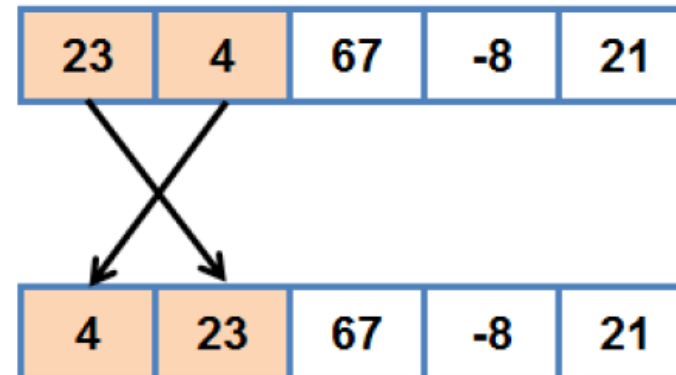


Qual é o Algoritmo que segue nos próximos?

Algoritmo 4

```
void ??? (int *V, int N) {  
    int i, continua, aux, fim = N;  
    do {  
        continua = 0;  
        for (i = 0; i < fim-1; i++) {  
            if (V[i] > V[i+1]) {  
                aux = V[i];  
                V[i] = V[i+1];  
                V[i+1] = aux;  
                continua = i;  
            }  
        }  
        fim--;  
    } while (continua != 0);  
}
```

Troca dois valores
consecutivos no vetor





Qual é o Algoritmo que segue nos próximos?

Algoritmo 5

```
void ???(int *v, int inicio, int fim){  
    int meio;  
    if (inicio < fim){  
        meio = floor((inicio + fim)/2);  
        ???(v, inicio, meio);  
        ???(v, meio+1, fim);  
        ???(v, inicio, meio, fim)  
    }  
}
```


Algoritmo 5

```
void ???(int *v, int inicio, int fim){
```

```
    int meio;
```

```
    if (inicio < fim){
```

```
        meio = floor((inicio + fim)/2);
```

```
        ???(v, inicio, meio);
```

```
        ???(v, meio+1, fim);
```

```
        ???(v, inicio, meio, fim)
```

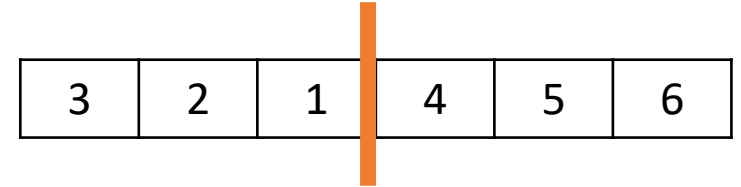
Divisão

Conquista

```
}
```

Algoritmo 5

```
void ??? (int *v, int inicio, int fim){  
    int meio;  
    if (inicio < fim){  
        meio = floor((inicio + fim)/2);  
        ??? (v, inicio, meio);  
        ??? (v, meio+1, fim);  
        ??? (v, inicio, meio, fim)  
    }  
}
```



Algoritmo 5

```
void ??? (int *v, int inicio, int fim){
```

```
    int meio;
```

```
    if (inicio < fim){
```

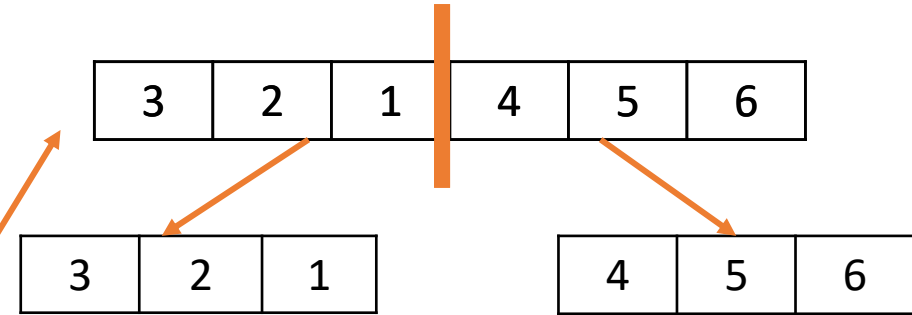
```
        meio = floor((inicio + fim)/2);
```

```
        ??? (v, inicio, meio);
```

```
        ??? (v, meio+1, fim);
```

```
        ??? (v, inicio, meio, fim)
```

```
}
```



Algoritmo 5

```
void ??? (int *v, int ini, int meio, int fim) {  
    int *temp, p1, p2, tam, i, j, k;  
    int fim1 = 0, fim2 = 0;  
    tam = fim - ini + 1;  
    p1 = ini;  
    p2 = meio + 1;  
    temp = (int *) malloc(tam * sizeof(int));  
    if(temp != NULL){  
        for(i = 0; i < tam; i++){  
            if(!fim1 && !fim2){  
                if(v[p1] < v[p2])  
                    temp[i] = v[p1++];  
                else  
                    temp[i] = v[p2++];  
  
                if(p1 > meio) fim1 = 1;  
                if(p2 > fim) fim2 = 1;  
            }else{  
                if (!fim1)  
                    temp[i] = v[p1++];  
                else  
                    temp[i] = v[p2++];  
            }  
        }  
        for(j = 0, k = ini; j < tam; j++, k++)  
            v[k] = temp[j];  
    }  
    free(temp);  
}
```



Qual é o Algoritmo que segue nos próximos?

Algoritmo 6

```
void ??? (int *V, int inicio, int fim) {  
    int pivo;  
    if (fim > inicio) {  
        pivo = particiona(V, inicio, fim);  
        quickSort(V, inicio, pivo-1);  
        quickSort(V, pivo+1, fim);  
    }  
}
```

Algoritmo 6

```
19 int ??? (int *V, int inicio, int final ) {
20     int esq, dir, pivo, aux;
21     esq = inicio;
22     dir = final;
23     pivo = V[inicio];
24     while(esq < dir) {
25         while(esq <= final && V[esq] <= pivo)
26             esq++;
27
28         while(dir >= 0 && V[dir] > pivo)
29             dir--;
30
31         if(esq < dir) {
32             aux = V[esq];
33             V[esq] = V[dir];
34             V[dir] = aux;
35         }
36     }
37     V[inicio] = V[dir];
38     V[dir] = pivo;
39     return dir;
40 }
```

Classificação e Pesquisa de Dados

Cristiano Santos

cristiano.santos@amf.edu.br