Algoritmos e Estruturas de Dados I

Profª Priscilla Abreu

2022.1

Algoritmos e Estruturas de Dados I

Roteiro da aula

- Correção de exercício
- Complexidade de algoritmos

Algoritmos e Estruturas de Dados I

Disponibilização de materiais e avisos:

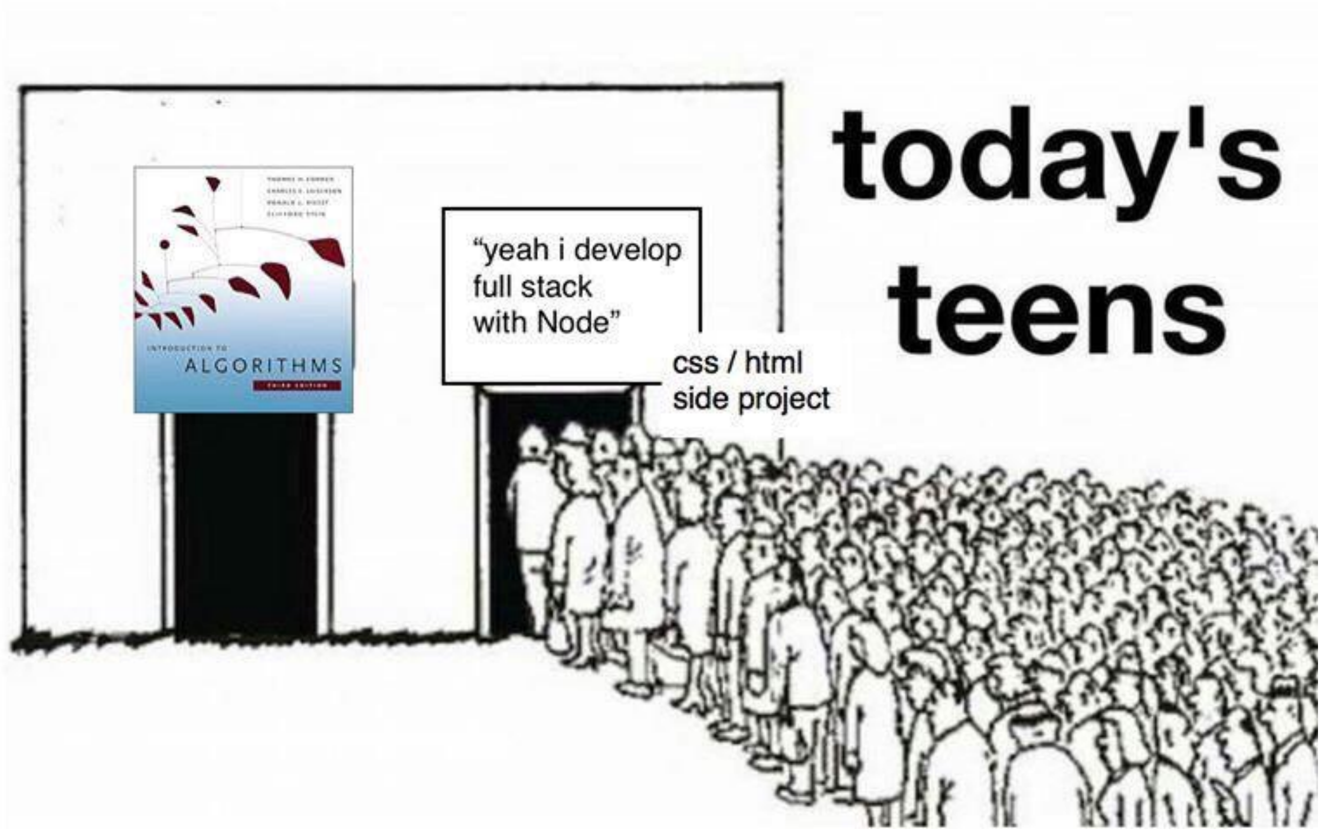
- Google Classroom:

vurjz3i

https://classroom.google.com/c/NDg4NDE4MDk5NTA4?cjc=vurjz3i

Complexidade de algoritmos

Algoritmos e Estruturas de Dados I



Algoritmos e Estruturas de Dados I

- **Algoritmos**
- **Correção e Análise**
- **Complexidade de um algoritmo**

Algoritmos e Estruturas de Dados I

- **Algoritmos**
  - processo sistemático (sequência de operações) para a resolução de um problema;
  - procedimento computacional bem definido que recebe um conjunto de valores de entrada e produz outro conjunto de valores de saída.
- **Correção e Análise**
- **Complexidade de um algoritmo**

Algoritmos e Estruturas de Dados I

- **Algoritmos**
  - processo sistemático (sequência de operações) para a resolução de um problema;
  - procedimento computacional bem definido que recebe um conjunto de valores de entrada e produz outro conjunto de valores de saída.
- **Correção e Análise**
  - Correção: verifica a exatidão do método empregado;
  - Análise: define parâmetros para avaliar a eficiência em termos de tempo de execução e memória ocupada.
- **Complexidade de um algoritmo**

Algoritmos e Estruturas de Dados I

- **Algoritmos**
  - processo sistemático (sequência de operações) para a resolução de um problema;
  - procedimento computacional bem definido que recebe um conjunto de valores de entrada e produz outro conjunto de valores de saída.
- **Correção e Análise**
  - Correção: verifica a exatidão do método empregado;
  - Análise: define parâmetros para avaliar a eficiência em termos de tempo de execução e memória ocupada.
- **Complexidade de um algoritmo**
  - É a quantidade de “trabalho” necessária para a sua execução, expressa em função do volume de dados de entrada;
  - “trabalho” -> memória e de tempo.

Algoritmos e Estruturas de Dados I

Estrutura de dados

- Para gerar uma saída, um **algoritmo** manipula dados a partir de um conjunto de entrada;
- Dados dispostos e manipulados de forma homogênea = Tipo Abstrato de Dados (TAD);
- **Tipo Abstrato de Dados**: formado por um conjunto de valores e funções que podem ser aplicadas sobre esses valores;
- **Estrutura de dados**: é uma representação do Tipo Abstrato de Dados.

Algoritmos e Estruturas de Dados I

Estrutura de dados

- Meio para o armazenamento e organização de dados visando facilitar o acesso e as modificações;
- As estruturas de dados diferem entre sim pela disposição e manipulação de seus dados;
- A escolha da estrutura de dados adequada depende do conhecimento dos algoritmos que a manipulam de forma eficiente.

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos: como avaliar?

- Espaço

  Quantidade de recursos (memória) utilizados.

- Tempo

  Tempo de execução do algoritmo ou o total de instruções executadas.

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos: como medir?

- Empiricamente
- Analiticamente

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos: como medir?

- Empiricamente
  - Organização de um conjunto de entradas para o algoritmo que contenham níveis diferentes de exigência de recursos;
  - Execução e medição do consumo do recurso;
  - Apresentação do resultado dos experimentos utilizando tabelas e gráficos;
  - Análise de funções que possam descrever o gráfico.

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos: como medir?

- Analiticamente

Estudo do algoritmo;

A complexidade será medida em função do tamanho dos dados de entrada do algoritmo;

Será obtida através de **métodos analíticos**, determinando uma **expressão matemática** que traduza o **comportamento de tempo** do algoritmo.

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos: como medir?

- Representaremos o tempo de execução de um algoritmo por uma função de custo T, considerando T(n) como a medida do tempo necessário para executar um algoritmo de tamanho *n*. T é denominada de **função de complexidade de tempo** do algoritmo.
- Se T representa a memória necessária para a execução de um algoritmo então T(n) é denominada **função de complexidade de espaço** do algoritmo.

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos: como medir?

- Consideraremos como **instruções** as operações aritméticas, de movimentação de dados e de controle;
- Além disso, o tempo de cada uma dessas operações será considerado com um **tempo constante** de execução;
- As operações mais **significativas** de um algoritmo são aquelas que contribuem para o seu tempo de execução;
- T(n) não representa de forma direta o tempo de execução, mas o número de vezes que uma operação significativa é executada.

Algoritmos e Estruturas de Dados I

Exemplo:

Implementar uma função para encontrar o menor valor de um vetor de inteiros com n elementos.

Algoritmos e Estruturas de Dados I

Exemplo linear

```
funçao encontraMenor( vetor[n]: inteiro, n: inteiro):inteiro
inicio
        i, menor: inteiro
        menor = vetor[0]
        para i:=1 até n-1 faça
                se (vetor[i] <  menor) então
                        menor := vetor[i]
        fim-para
        retorne menor
fim
```

```
int encontraMenor( int vetor[], int n){
        int i, menor;
        menor = vetor[0];
        for (i=1; i<n; i++){
            if (vetor[i] <  menor)
                menor := vetor[i];
        }
        return menor;
}
```

Algoritmos e Estruturas de Dados I

Exemplo linear

```
funçao encontraMenor( vetor[n]: inteiro, n: inteiro):inteiro
inicio
        i, menor: inteiro
        menor = vetor[0]
        para i:=1 até n-1 faça
                se (vetor[i] <  menor) então
                        menor := vetor[i]
        fim-para
        retorne menor
fim
```

Função de complexidade T(n):

número de comparações entre os elementos de vetor.

Serão realizadas n-1 comparações.

T(n) = n-1

Tempo de execução **uniforme**, dependente apenas do tamanho de entrada.

Algoritmos e Estruturas de Dados I

Exemplo linear

Considere a busca por um determinado valor em um vetor.

1	9	4	26	18	55		30
0	1	2	3	4	5	...	n

Algoritmos e Estruturas de Dados I

Exemplo linear

Considere a busca por um determinado valor em um vetor.

1	9	4	26	18	55		30
0	1	2	3	4	5	...	n

Para efetuar a busca é necessário comparar o valor desejado com cada elemento do vetor até que seja encontrado ou compare com todos os valores sem obter sucesso.

Algoritmos e Estruturas de Dados I

Exemplo linear

Considere a busca por um determinado valor em um vetor.

1	9	4	26	18	55		30
0	1	2	3	4	5	...	n

```
funcao busca(vetor[n]: inteiro, n:inteiro, valor: inteiro): inteiro
inicio
        i: inteiro
        para i:=0 até (n-1) faça
                se (vetor[i] = valor) então
                        retorne i
                fim-se
        fim-para
        retorne -1
fim
```

Algoritmos e Estruturas de Dados I

Exemplo linear: Considere a busca por um determinado valor em um vetor.

Esse algoritmo não se comporta de maneira uniforme. Temos três casos:

- Pior caso

  Maior tempo de execução sobre todas as entradas de tamanho n.

- Melhor caso

  Menor tempo de execução sobre todas as entradas de tamanho n.

- Caso médio

  Média dos tempos de execução do algoritmo sobre todas as entradas de tamanho n.

Algoritmos e Estruturas de Dados I

Exemplo linear: Considere a busca por um determinado valor em um vetor.

A função de complexidade de tempo T é obtida a partir do número de comparações efetuadas.

Assim:

- Melhor caso

  Quando o valor a ser procurado encontra-se na primeira posição do vetor.

Algoritmos e Estruturas de Dados I

Exemplo linear: Considere a busca por um determinado valor em um vetor.

A função de complexidade de tempo T é obtida a partir do número de comparações efetuadas.

Assim:

- Pior caso

  Quando o valor a ser procurado encontra-se na última posição do vetor ou não está armazenado no vetor.

Algoritmos e Estruturas de Dados I

Exemplo linear: Considere a busca por um determinado valor em um vetor.

A função de complexidade de tempo T é obtida a partir do número de comparações efetuadas.

Assim:

- Caso médio

  Leva em consideração uma distribuição de probabilidade, considerando p_i como a probabilidade de procurar o *i-ésimo* elemento do vetor e supondo que a probabilidade de encontrar cada elemento seja 1/n..

Algoritmos e Estruturas de Dados I

Exemplo Quadrático

Dadas duas matrizes A =(a_{ij}) e B = (b_{ij}), ambas nxn, determinar a matriz soma C = (c_{ij}).

```
funcao somaMatriz(m1: matriz, m2:matriz): matriz
inicio
    mSoma: matriz
    lin, col: inteiro
    para lin:=0 até (n-1) faça
        para col:=0 até (n-1) faça
            mSoma[lin][col] = m1[lin][col] + m2[lin][col]
        fim-para
    fim-para
    retorne -1
fim
```

Algoritmos e Estruturas de Dados I

Exemplo Quadrático

Dadas duas matrizes A =(a_{ij}) e B = (b_{ij}), ambas nxn, determinar a matriz soma C = (c_{ij}).

```
funcao somaMatriz(m1: matriz, m2:matriz): matriz
inicio
    mSoma: matriz
    lin, col: inteiro
    para lin:=0 até (n-1) faça
        para col:=0 até (n-1) faça
            mSoma[lin][col] = m1[lin][col] + m2[lin][col]
        fim-para
    fim-para
    retorne -1
fim
```

Tamanho da entrada: n^2
Tempo = total de passos = total de instruções executadas: n^2

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos

- Ordem de grandeza
- Operação dominante
- Pior caso
- Descarte de constantes aditivas e/ou multiplicativas
- Comportamento assintótico = entradas com tamanho suficientemente grande.

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos

- Escolha de algoritmos está relacionada ao desempenho sobre tamanhos de entrada grandes!
- Assim, estuda-se o comportamento assintótico da função de complexidade de tempo dos algoritmos: sua **eficiência assintótica**.
- O **comportamento assintótico** representa a **curva de crescimento** da função gerada pelo processo de análise de algoritmos.
- Foco de estudo: como o algoritmo se comporta à medida que o tamanho da entrada aumenta.

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos

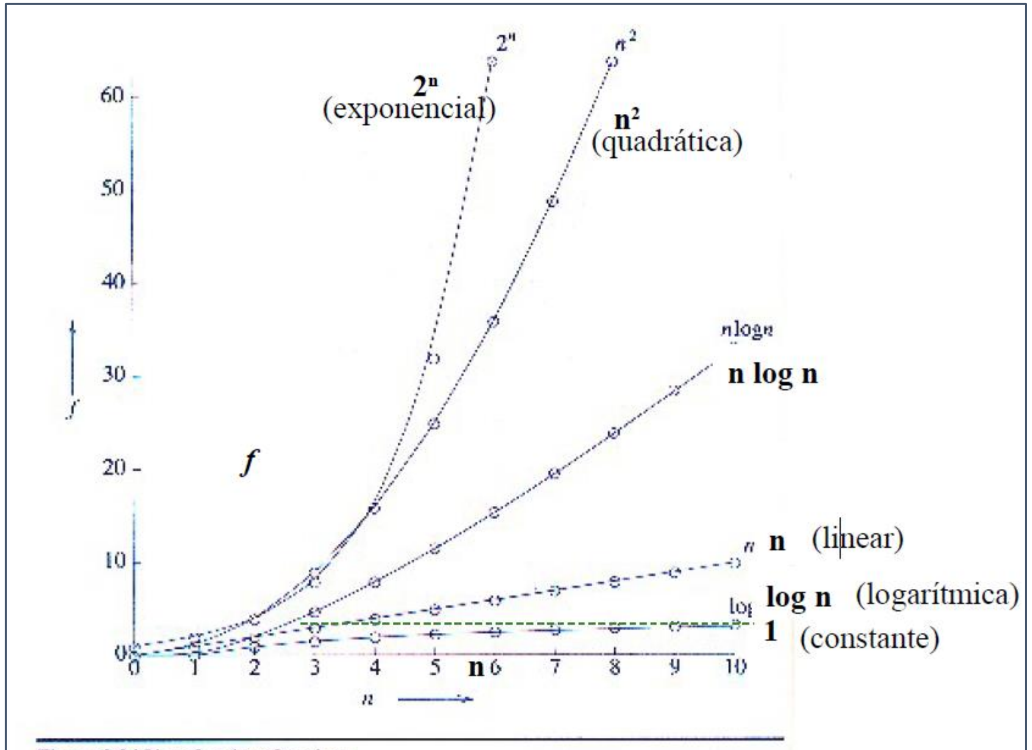
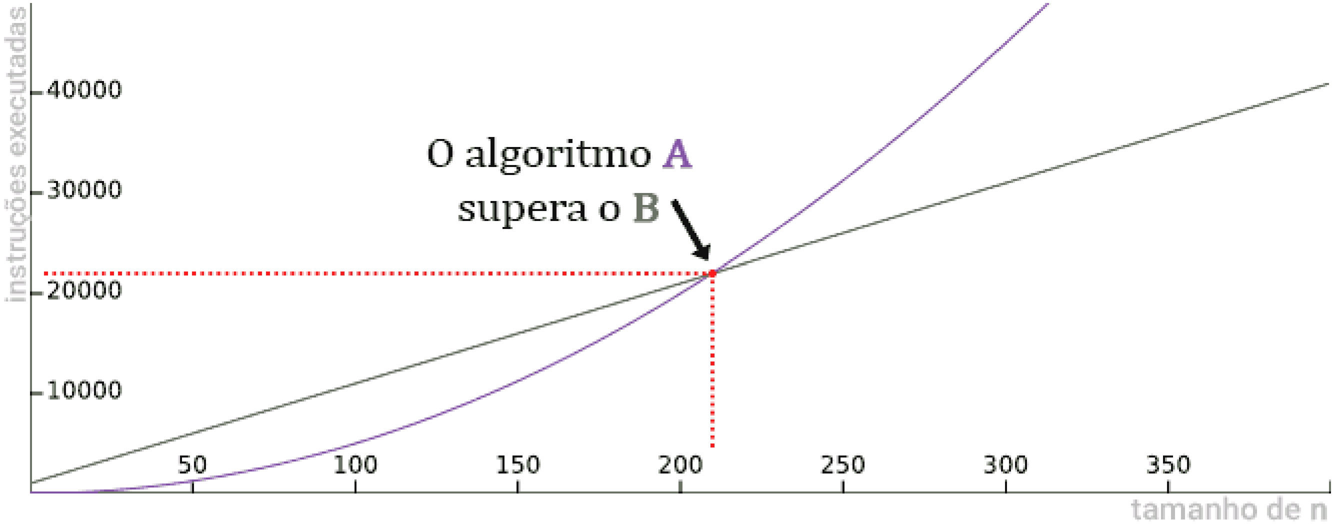


Figure 2.24 Plot of various functions

Fonte: Data Structures,Algorithms and Applications in C++, Sahni



Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos – notação assintótica

Notação utilizada para representar o comportamento assintótico das funções de complexidade de tempo dos algoritmos, bem como relacionar o comportamento das funções de complexidade de dois algoritmos.

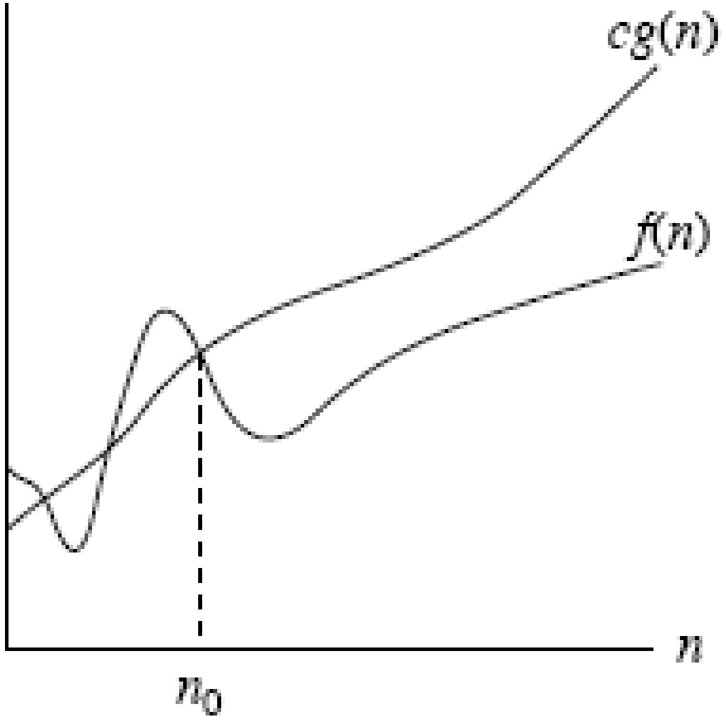
Consideraremos funções com domínio como o conjunto dos números naturais N = {0, 1, 2, ...}.

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos – notação assintótica

Definição: uma função g(n) domina assintoticamente outra função f(n) se existem duas constantes positivas c e n_0 tais que, para n >= n_0 temos que:

$$|f(n)| \leq c \times |g(n)|$$



Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos – notação assintótica

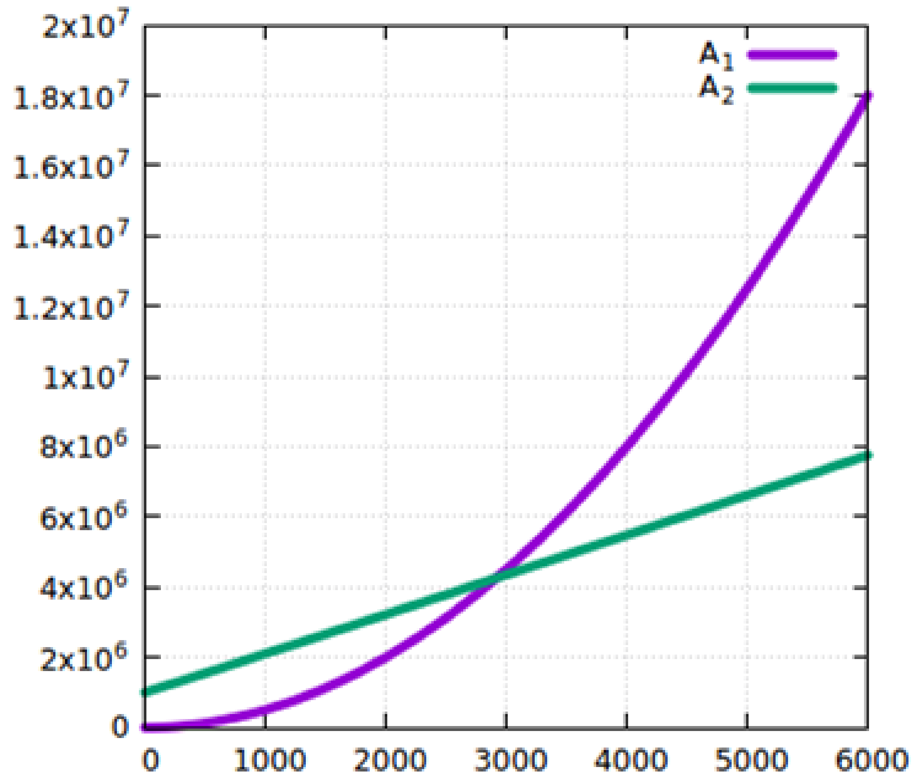
- Alguns tipos:
  - Notação O
  - Notação Ω
  - Notação θ

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos – notação assintótica

- Notação O

Exemplo: Considere dois algoritmos A1 e A2 que realizam o upload de arquivos para um servidor. Suponha que a complexidade de A1 seja O(n^2) e de A2 seja O(n). Considere o gráfico a seguir representando o comportamento desses algoritmos.



Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos – notação assintótica

- Notação O
  - À medida que o tamanho do arquivo aumenta, o algoritmo A1 cresce **exponencialmente** em número de operações.
  - Este algoritmo segue uma classe de funções que se comportam como uma exponencial quadrática.
  - Já o algoritmo A2 demonstra um crescimento linear, pois à medida que o tamanho do arquivo de entrada cresce, o numero de operações **cresce na mesma proporção**.

Algoritmos e Estruturas de Dados I

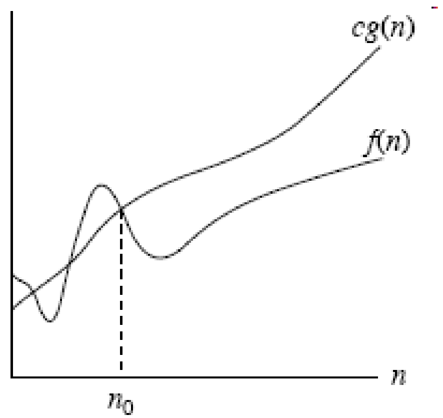
Complexidade de Algoritmos – notação assintótica

- Notação O

$$f(n) = O(g(n)),$$

se existirem uma constante c e um valor n_0 tal que

$$0 \leq f(n) \leq c \times g(n), \; \forall \; n \geq n_0$$



A notação O se refere ao **limite superior** de uma função. Ou seja, existe uma função à qual o algoritmo analisado não terá um ponto acima da curva desta função.

Isto é, a função f(n) cresce no máximo como a função g(n).

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos – notação assintótica

- Notação O

$$f = n^2 - 1 \rightarrow f = O(n^2)$$
$$f = n^2 - 1 \rightarrow f = O(n^3)$$
$$f = 403 \rightarrow$$
$$f = 3n + 5\log n + 2 \rightarrow$$
$$f = n^2 + n^3 \rightarrow$$
$$f = 7n^2 + 2^n \rightarrow$$
$$f = 2n + n.\log n + 3 \rightarrow$$
$$f = n^2 + n.\log n \rightarrow$$

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos – notação assintótica

- Notação O

$$f = n^2 - 1 \rightarrow f = O(n^2)$$

$$f = n^2 - 1 \rightarrow f = O(n^3)$$

$$f = 403 \rightarrow f = O(1)$$

$$f = 3n + 5\log n + 2 \rightarrow$$

$$f = n^2 + n^3 \rightarrow$$

$$f = 7n^2 + 2^n \rightarrow$$

$$f = 2n + n.\log n + 3 \rightarrow$$

$$f = n^2 + n.\log n \rightarrow$$

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos – notação assintótica

- Notação O

$$f = n^2 - 1 \rightarrow f = O(n^2)$$

$$f = n^2 - 1 \rightarrow f = O(n^3)$$

$$f = 403 \rightarrow f = O(1)$$

$$f = 3n + 5\log n + 2 \rightarrow f = O(n)$$

$$f = n^2 + n^3 \rightarrow$$

$$f = 7n^2 + 2^n \rightarrow$$

$$f = 2n + n.\log n + 3 \rightarrow$$

$$f = n^2 + n.\log n \rightarrow$$

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos – notação assintótica

- Notação O

$$f = n^2 - 1 \rightarrow f = O(n^2)$$

$$f = n^2 - 1 \rightarrow f = O(n^3)$$

$$f = 403 \rightarrow f = O(1)$$

$$f = 3n + 5\log n + 2 \rightarrow f = O(n)$$

$$f = n^2 + n^3 \rightarrow f = O(n^3)$$

$$f = 7n^2 + 2^n \rightarrow$$

$$f = 2n + n.\log n + 3 \rightarrow$$

$$f = n^2 + n.\log n \rightarrow$$

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos – notação assintótica

- Notação O

$$f = n^2 - 1 \rightarrow f = O(n^2)$$
$$f = n^2 - 1 \rightarrow f = O(n^3)$$
$$f = 403 \rightarrow f = O(1)$$
$$f = 3n + 5\log n + 2 \rightarrow f = O(n)$$
$$f = n^2 + n^3 \rightarrow f = O(n^3)$$
$$f = 7n^2 + 2^n \rightarrow f = O(2^n)$$
$$f = 2n + n.\log n + 3 \rightarrow$$
$$f = n^2 + n.\log n \rightarrow$$

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos – notação assintótica

- Notação O

$$f = n^2 - 1 \rightarrow f = O(n^2)$$
$$f = n^2 - 1 \rightarrow f = O(n^3)$$
$$f = 403 \rightarrow f = O(1)$$
$$f = 3n + 5\log n + 2 \rightarrow f = O(n)$$
$$f = n^2 + n^3 \rightarrow f = O(n^3)$$
$$f = 7n^2 + 2^n \rightarrow f = O(2^n)$$
$$f = 2n + n.\log n + 3 \rightarrow f = O(n.\log n)$$
$$f = n^2 + n.\log n \rightarrow$$

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos – notação assintótica

- Notação O

$$f = n^2 - 1 \rightarrow f = O(n^2)$$
$$f = n^2 - 1 \rightarrow f = O(n^3)$$
$$f = 403 \rightarrow f = O(1)$$
$$f = 3n + 5\log n + 2 \rightarrow f = O(n)$$
$$f = n^2 + n^3 \rightarrow f = O(n^3)$$
$$f = 7n^2 + 2^n \rightarrow f = O(2^n)$$
$$f = 2n + n.\log n + 3 \rightarrow f = O(n.\log n)$$
$$f = n^2 + n.\log n \rightarrow f = O(n^2)$$

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos – notação assintótica

- Notação O – Operações e propriedades

$$f(n) = O(f(n))$$

$$O(c.f(n)) = c.O(f(n)) = O(f(n)), \text{ c é uma constante}$$

$$O(f(n)) + O(f(n)) = O(f(n))$$

$$O(g(n) + f(n)) = O(f(n)) + O(g(n)) = O(\max(f(n),g(n)))$$

$$O(f(n).g(n)) = O(f(n)).O(g(n))$$

Algoritmos e Estruturas de Dados I

Complexidade de Algoritmos

Exercício

Construir um algoritmo para dada uma matriz nxn determinar o maior elemento da matriz. Calcular as complexidades de melhor e pior caso do seu algoritmo.

FIM