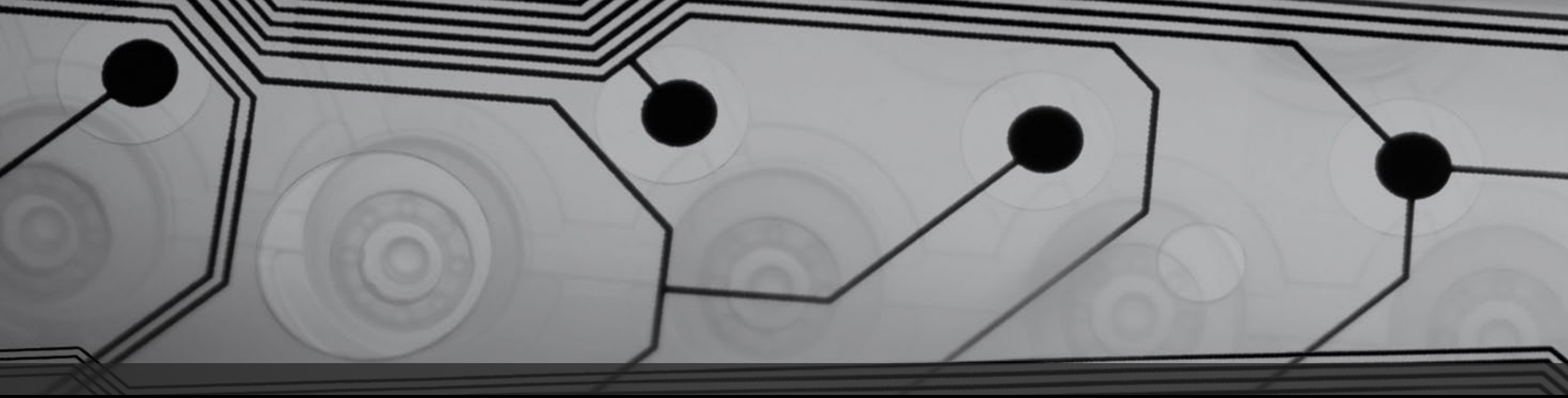


Analise de algoritmos

Notação Assintótica

Análise de algoritmos

- **Análise de algoritmo**
  - Tempo de processamento em função dos dados de entrada;
  - Espaço de memória total requerido para os dados;
  - Comprimento total do código;
  - Obtenção correta do resultado pretendido;
  - Robustez (como comporta-se com as entradas inválidas ou não previstas).
- Análise de algoritmos é medição de complexidade de algoritmo
  - Quantidade de "trabalho" necessária para a sua execução, expressa em função das **operações fundamentais,** as quais variam de acordo com o algoritmo, e em função do volume de dados.

Complexidade

- **Porquê o estudo da complexidade?**
  - Performance
    - Escolher entre vários algoritmos o mais eficiente para implementar;
    - Desenvolver novos algoritmos para problemas que já têm solução;
    - Desenvolver algoritmos mais eficientes (melhorar os algoritmos), devido ao aumento constante do "tamanho" dos problemas a serem resolvidos.
  - Complexidade computacional – torna possível determinar se a implementação de determinado algoritmo é viável.

Complexidade

- Tipos de complexidade
  - Espacial
    - Este tipo de complexidade representa, por exemplo, o espaço de memória usado para executar o algoritmo.
  - Temporal
    - Este tipo de complexidade é o mais usado podendo dividir-se em dois grupos:
      - Tempo (real) necessário à execução do algoritmo. (Como podemos medir?)
      - Número de instruções necessárias à execução.

Analise de algoritmos

- Medidas de análise
  - Devem ser independentes da tecnologia (hardware/software)
  - Modelos matemáticos simplificados baseados nos fatores relevantes:
    - Tempo de execução

    Uma função que relaciona o tempo de execução com o tamanho de entrada:

    $$T = F(n)$$

      - Conjunto de operações a serem executadas.
      - Custo associado à execução de cada operação.
    - Ocupação de espaço em memória.

Complexidade

- Exemplo
  - Sejam cinco (5) algoritmos A_1 a A_5 para resolver um mesmo problema, de complexidade diferentes. (Supomos que uma operação leva 1 ms para ser efetuada.)
  - $T_k(n)$ é a complexidade ou seja o número de operações que o algoritmo efetua para N entradas.

n	A1 T1(n) = n	A2 T2(n) = nlog n	A3 T3(n) = n^2	A4 T4(n) = n^3	A5 T5(n) = 2^n
16	0,016s	0,064s	0,256s	4s	1m4s
32	0,032s	0,16s	1s	33s	46 Dias
512	0,512s	9s	4m22s	1 Dia 13 h	10^{137} Séculos

Tempo necessário para o algoritmo em função de n entradas

Operações primitivas

- Atribuição de valores a variáveis
- Chamada de métodos
- Operações aritméticas
- Comparação de dois números
- Acesso a elementos de um array
- Seguir uma referência de objeto (acesso a objeto)
- Retorno de um método

A análise!

- Você acha importante essa análise?
- Por que?

Notação assintótica

- **Notação O** (big O)
  - Definição: Considere uma função f(n) não negativa para todos os inteiros n >= 0.

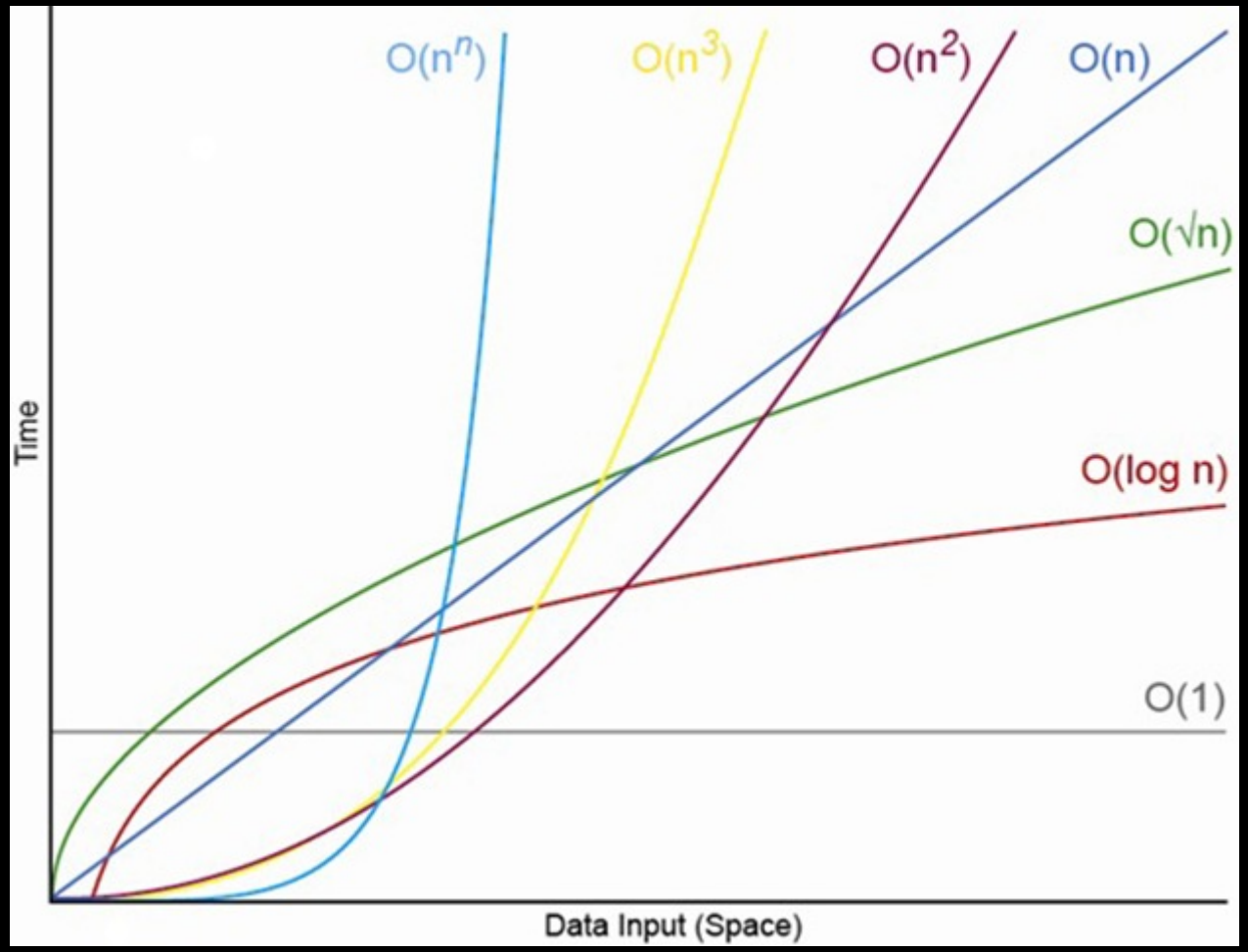
    Dizemos que "f(n) é O(g(n))" e escrevemos f(n) = O(g(n)), se existem um inteiro n_0 e uma constante c > 0, tais que para todo inteiro n >= n_0, f(n) <= cg(n)
  - Caracteriza o comportamento assintótico de uma função, estabelecendo um limite superior quanto à taxa de crescimento da função em relação ao crescimento de n.
  - Permite ignorar fatores constantes e termos de menor ordem, centrando-se nos componentes que mais afetam o crescimento de uma função.

Notação Assintótica

- Terminologia de classes mais comuns de funções:
  - Logarítmica – O(log n)
  - Linear – O(n)
  - Quadrática – O(n^2)
  - Polinomial – O(n^k), com k >= 1
  - Exponencial – O(a^n), com a > 1

Função	Designação
C	Constante
Log N	Logaritmo
log^2 N	Logaritmo Quadrado
N	Linear
N log N	N log N
N^2	Quadrática
N^3	Cúbica
2^n	Exponencial

Diagrama



Vídeo de ajuda