ITAÚ
LET'S CODE

AULA 08

Complexidade de Algoritmos

L >
< C

Introdução

- Quando falamos sobre a complexidade de um algoritmo estamos nos referindo à **quantidade de recursos** (tempo de processamento ou espaço de armazenamento) que precisa ser **consumida** ou alocada **para sua execução**.

Introdução

- Essa análise de complexidade é fundamental para que possamos projetar algoritmos eficientes e verificar a eficiência dos algoritmos que utilizamos. Além disso, para escolher a estrutura de dados mais adequada para cada problema que precisamos resolver, é necessário identificar quais operações essa estrutura fornece com máxima eficiência.

Do que depende?

DO QUE DEPENDE O TEMPO DE EXECUÇÃO DE UM ALGORITMO?

1. Da **velocidade** do computador;
2. Da **linguagem** de programação;
3. Do **compilador** que traduziu o programa para executar diretamente no computador.

Como analisar?

1. Vamos pensar no tempo de execução do algoritmo como uma função do tamanho da sua entrada.
2. Vamos pensar no tempo de execução do algoritmo como quão rápido essa função cresce dado o tamanho da entrada.

TAXA DE CRESCIMENTO DO TEMPO DE EXECUÇÃO!

Como analisar?

- O tamanho da entrada aqui é a quantidade de "coisas" que o algoritmo precisa processar.
- Fazemos a análise em função do tamanho da entrada, pois o que nos interessa é identificar como o algoritmo irá escalar.

Como analisar?

- Em outras palavras, queremos responder algo do tipo: **"Quando dobramos a entrada, o tempo de execução dobra? quadruplica?"**

- Enfim, **o que acontece com o tempo de execução a medida que o tamanho da entrada aumenta**.

Como analisar?

- Queremos uma medida que seja geral o suficiente para capturar o comportamento do algoritmo **independentemente do tipo de máquina** que o código irá executar.
- Se usássemos segundos ou minutos para medir o tempo, teríamos que fazer uma análise diferente para cada processador existente.

Como analisar?

- Por esse motivo, o que contamos é **a quantidade de operações básicas** que são executadas e cada uma destas instruções leva um tempo constante.

- **Como é feita essa contagem?**

Contagem de Instruções

- **Instruções simples:** são aquelas que podem ser executadas em linguagem de máquina. Diremos que medem **1 unidade de tempo**, ou simplesmente 1.

- **Instruções complexas:** **combinação** de instruções simples. Instruções de **controle de fluxo**. Em resumo, a soma de instruções simples.

Instruções Simples

CASOS

1. Atribuições de valores (de um modo geral)
2. Incremento de valores
3. Operações aritméticas mais complexas
4. Acesso ao valor de um elemento de um vetor
5. Expressões lógicas
6. Operações de leitura e escrita

Existe custo zero?

- **Sim! Estruturas de seleção possuem custo zero.** O que gera valor é a expressão agregada (comparação).

- **Declaração de variáveis** também têm custo zero. Como não trabalhamos com declaração de variáveis em Python, não iremos considerar esse caso aqui.

Exemplos

EXEMPLOS

```
a = 21
b = a + 11
x = x * x + (a - b)
```

Notação Assintótica

- Considere a seguinte situação:
- Você deseja ordenar algumas listas, e você tem duas opções de algoritmos para isso com as seguintes complexidades:

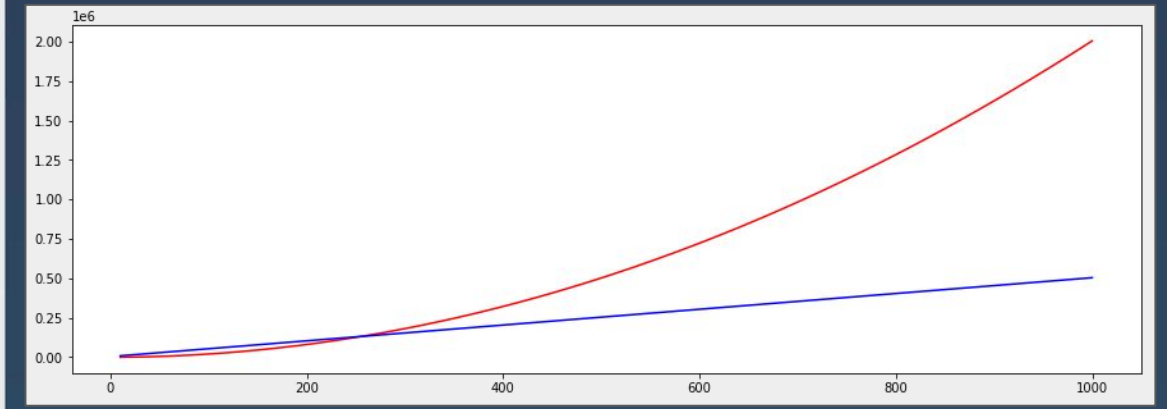
1. **$2n^2 + 5n$** operações
2. **$500n + 4000$** operações

Qual deles seria mais eficiente para ordenar uma lista com 10 itens?

E com 1000 itens?

Notação Assintótica

- **Qual você escolheria para utilizar em uma lista arbitrária (da qual você não saberia o tamanho previamente)?**



Notação Assintótica

- Trata-se de uma notação que nos permite **descartar os termos constantes** e menos significativos.

VAMOS À PRÁTICA!