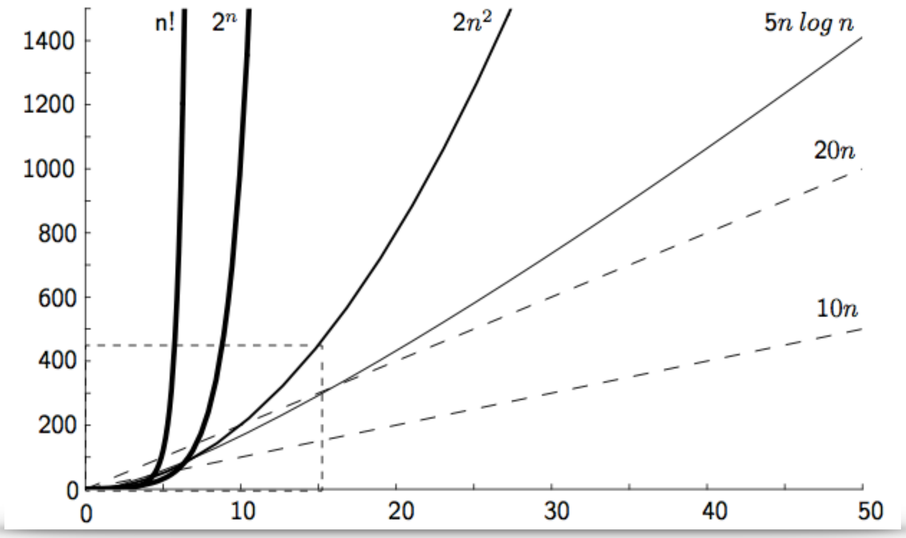
Profa. Dra. Raquel C. de Melo-Minardi
Departamento de Ciência da Computação
Instituto de Ciências Exatas
Universidade Federal de Minas Gerais



MÓDULO 3
COMPLEXIDADE DE ALGORITMOS
Introdução

ANALISE DE COMPLEXIDADE DE ALGORITMOS

- O projeto de algoritmos é fortemente influenciado pelo estudo de seus comportamentos
- Após entender perfeitamente o problema e suas possíveis soluções algorítmicas, o algoritmo projetado será implementado em uma linguagem de programação
- A escolha do algoritmo envolve considerações importantes sobre o tempo de execução e o espaço de memória ocupados
- Esse tipo de escolha envolve um processo que chamamos de análise de algoritmos

ANALISE DE COMPLEXIDADE DE ALGORITMOS

- Há dois tipos de análises bem distintas que podemos realizar
  - **Análise de um algoritmo em particular**
    - Qual o custo de um determinado algoritmo na solução de um problema?
    - Comumente, analisamos ou contamos número de vezes que cada trecho de um programa irá executar dependendo da entrada
    - Podemos ainda analisar o espaço em memória requerido também em função da entrada

ANALISE DE COMPLEXIDADE DE ALGORITMOS

- Há dois tipos de análises bem distintas que podemos realizar
  - **Análise de uma classe de algoritmos**
    - Qual o melhor algoritmo ou o algoritmo de menor custo para solução de um problema?
    - Normalmente, analisamos toda uma família de algoritmos para resolver um problema em particular visando escolher o melhor ou mais eficiente deles

LIMITES E OTIMALIDADE DE ALGORITMOS

- Esse último tipo de análise normalmente envolve a estimativa de **limites** a complexidade computacional da classe de algoritmos
  - Quando se consegue determinar o **menor custo possível** para resolver um determinado problema ou problemas de uma certa classe, temos a medida da dificuldade inerente a esse tipo particular de problema
  - Quando conseguimos **provar** que esse custo é o menor possível, dizemos que o algoritmo é **ótimo** para tal medida de custo
  - Comumente temos **inúmeros possíveis algoritmos** para resolver um mesmo problema e normalmente desejamos utilizar o **melhor** ou o **mais eficiente**

MEDINDO O CUSTO COMPUTACIONAL

- Mas como medir então esse custo computacional?
  - Pode ser medido de várias maneiras
    - **Tempo de execução**
      - Possíveis problemas desse tipo de abordagem empírica?
        - os resultados dependem do compilador utilizado que pode favorecer certas construções em detrimento de outras
        - os resultados dependem de hardware

- Como resolver esse problema?

SOLUÇAO

- Utilizar **modelos matemáticos** baseados em um **computador hipotético** idealizado
- É como se todos os programas rodassem em um mesmo computador pois o mesmo não influenciaria em nossa análise
- Define-se assim um conjunto de operações importantes a serem avaliadas e seu custo e ignora-se outro conjunto de instruções que não seriam tão significativas
  - Exemplo: em algoritmos que ordenam uma lista em ordem ascendente podemos considerar apenas as operações de comparar dois elementos e trocar esses elementos de posição e desconsiderar outras operações aritméticas, atribuições ou manipulações de índices, caso existam

FUNÇAO DE COMPLEXIDADE

- Esse modelo matemático que nos permite então medir o custo de execução de um algoritmo nada mais é do que uma **função matemática** que chamamos de **função de complexidade f(n)** em que medimos o tempo gasto para executar um algoritmo em um problema de tamanho *n*
  - Se *f(n)* é uma medida do tempo gasto para execução, essa função será denominada **função de complexidade de tempo** do algoritmo
  - Essa função **não representa o tempo de execução diretamente** mas o número de vezes que determinadas operações consideradas relevantes são executadas para uma determinada entrada
  - Se f(n) é uma medida da quantidade de memória gasta para execução, essa função será denominada **função de complexidade de espaço** do algoritmo

Desafio

1. Projete um algoritmo e implemente uma função em Python que receba como entrada uma lista e retorne o maior elemento desse conjunto.
2. Descubra quais as operações mais relevantes em termos de tempo de execução de seu programa.
3. Tente calcular a função de complexidade de tempo *f(n)* de seu programa.
4. Você acha que seu programa é ótimo?