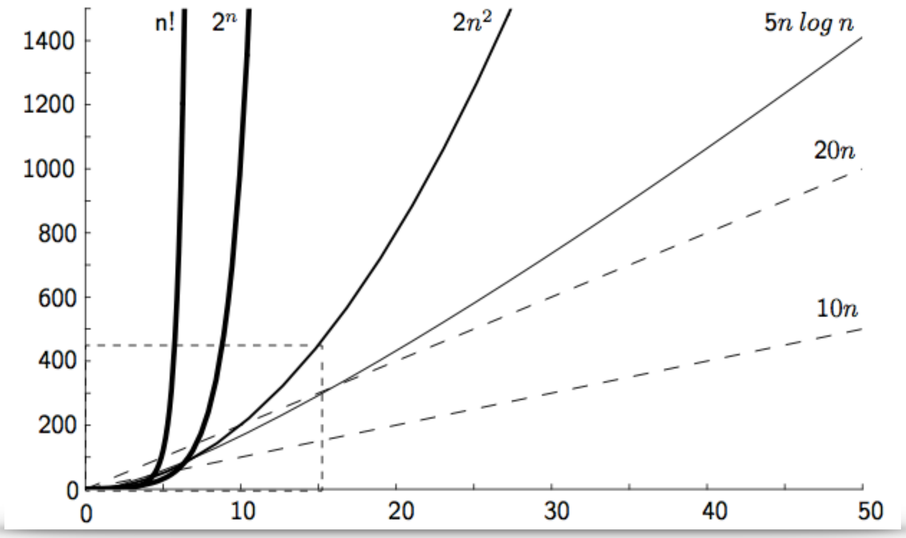
Profa. Dra. Raquel C. de Melo-Minardi
Departamento de Ciência da Computação
Instituto de Ciências Exatas
Universidade Federal de Minas Gerais



MÓDULO 3
COMPLEXIDADE DE ALGORITMOS
Função de complexidade – Parte I

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?

SOLUÇÃO - PARTE (1)

- Veja abaixo nossa proposta de solução para a parte (1) desse desafio

```
def maior(lista):
        maior = lista[0]
        for i in range(1, len(lista)):
                print(i)
                if (maior < lista[i]):
                        maior = lista[i]
        return maior
```

SOLUÇÃO – PARTE (2)

- Quanto à parte (2) do desafio, qual a operação mais relevante a ser executada nesse código?

- A **comparação** se cada elemento é maior que o maior elemento até o momento, ou seja, o comando condicional

- Note que quando tivermos uma lista de tamanho realmente grande, é essa operação que será significativa para o crescimento do tempo de execução já que as outras linhas do programa executarão poucas vezes

SOLUÇÃO – PARTE (3)

- Para obtermos f(n), que é parte (3) do desafio, basta contarmos **quantas vezes esse `if` será executado**

- Como o laço se repete para `i` entre *1* e a última posição válida da lista, podemos afirmar que o **`if`** será executado *n-1* vezes, ou seja

$$f(n) = n-1$$

SOLUÇÃO – PARTE (4)

- A parte (4) do desafio nos pergunta se esse algoritmo é ótimo
- Pode-se provar [Ziviani, 2004] que para encontrar o maior elemento em uma lista de *n* elementos, pelo menos *n-1* comparações são necessárias logo, nosso algoritmo é ótimo

[Ziviani, 2004] Ziviani, Nivio. Projeto de algoritmos: com implementações em Pascal e C. Vol. 2. Thomson, 2004.

Desafio

1. Projete um algoritmo e implemente uma função em Python que receba como entrada uma lista e retorne o **menor** e 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?

SOLUÇÃO - PARTE (1)

- Veja abaixo nossa proposta de solução para a parte (1) desse desafio

```
def maiorMenor1(lista):
        menor = maior = lista[0] # Nova variável menor
        for i in range(1, len(lista)):
                if (menor > lista[i]):
                        menor = lista[i]
                if (maior < lista[i]):
                        maior = lista[i]
        tupla = (menor, maior)
        return tupla
```

SOLUÇÃO – PARTE (1)

- Aproveitamos a função anterior e apenas adicionamos a variável `menor` que também foi inicializada com o valor da primeira posição da lista a princípio e posteriormente, dentro do laço, foi tendo seu valor alterado a medida que um novo menor valor era encontrado na lista

Desafio

1. Projete um algoritmo e implemente uma função em Python que receba como entrada uma lista e retorne o **menor** e 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?

SOLUÇÃO – PARTE (2)

- A operação mais relevante para a análise da complexidade desse algoritmo é o **comando condicional**

Desafio

1. Projete um algoritmo e implemente uma função em Python que receba como entrada uma lista e retorne o **menor** e 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?

SOLUÇÃO – PARTE (3)

- Contudo agora termos a execução desse comando condicional duas vezes, o que nos leva a ter como resposta à parte (3) do desafio

$$f(n)=2(n-1)$$

Desafio

1. Projete um algoritmo e implemente uma função em Python que receba como entrada uma lista e retorne o **menor** e 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?

SOLUÇÃO – PARTE (4)

- A resposta à parte (4) do desafio é não
  - Essa não é uma função ótima pois podemos encontrar formas mais eficientes de obter o menor e o maior elementos de uma lista sem ter de fazer *2(n-1)* comparações

- Mas como isso poderia ser feito? Alguma idéia? Você pode gastar um tempo refletindo a respeito desse código e em possíveis formas de melhorá-lo antes de passar a frente no estudo desse texto se desejar