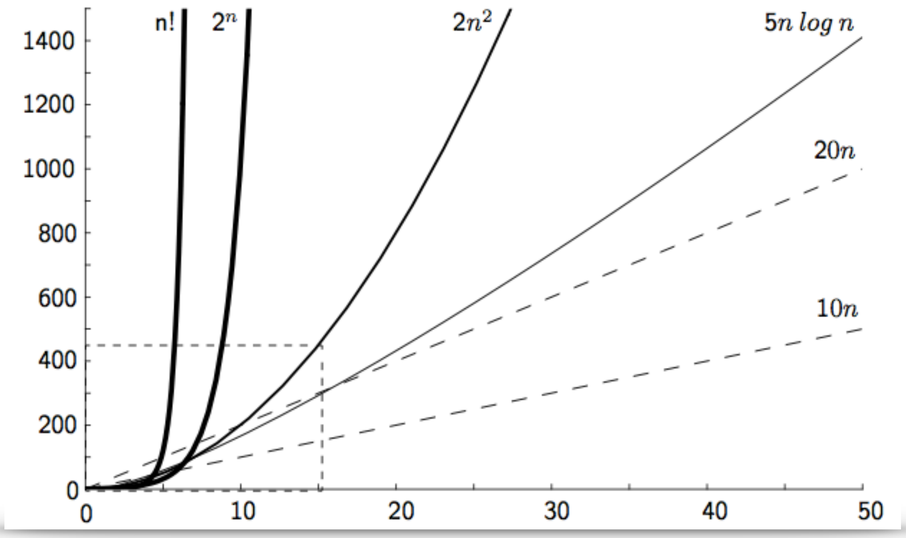
Profa. Dra. Raquel C. de Melo-Minardi
Departamento de Ciência da Computação
Instituto de Ciências Exatas
Universidade Federal de Minas Gerais



MÓDULO 3
COMPLEXIDADE DE ALGORITMOS
Função de complexidade – Parte IV

UM OUTRO EXEMPLO INTERESSANTE...

- Considere o problema de se **pesquisar registros** em um arquivo
  - Base de dados em geral em que os registros foram carregados previamente em um arranjo estando em **memória RAM**
  - Cada registro tem uma **chave única**, seu identificador, que é utilizada para encontrá-lo no arquivo
    - id do UniProt ou mesmo um id do PDB (Protein Data Bank)

Desafio

Dada uma chave qualquer, localizar, caso exista, o registro que contêm essa chave
Implemente esse problema como um desafio

SOLUÇÃO - PARTE (1)

- Vamos apresentar a solução mais ingênua para esse programa no código a seguir

```
def pesquisa(reg, lista):
        for i in range(0, len(lista)):
                if lista[i] == reg:
                        return i
        return -1
```

Desafio

Com que tipo de arranjo ocorre e quais as funções de complexidade desse algoritmo no:

- melhor caso
- pior caso
- caso médio

ANÁLISE DE COMPLEXIDADE

- Essa função recebe um registro `reg` e uma lista e pesquisa através de um laço que percorre a lista desde a posição *0* até a última posição *(n-1)* verificando se encontrou ou não o mesmo na lista

- No **melhor caso**, o elemento pesquisado seria o primeiro elemento da lista e apenas uma iteração do `for` seria executada visto que há um return dentro do `if` e sua função de complexidade seria

$$f(n) = 1$$

ANÁLISE DE COMPLEXIDADE

- No **pior caso**, o elemento procurado estaria na última posição da lista ou não seria encontrado na mesma
  - O laço seria executado até o final e a função retornaria “`return -1`" resultando na seguinte função de complexidade

$$f(n) = n$$

ANÁLISE DE COMPLEXIDADE

- Temos que considerar a baixa probabilidade dessa ocorrência
- O que seria então o caso médio ou caso mais esperado?
- Para essa análise é preciso conhecer a probabilidade de que cada registro fosse acessado
  - No contexto de bases de dados biológicas e públicas, é possível que os mantenedores tenham estatísticas de acesso a cada registro e isso poderia gerar uma probabilidade de que cada item fosse buscado por um usuário
  - Seja p_i, a probabilidade de que o i-ésimo registro seja procurado e, considerando que para encontrar o i-ésimo registro sejam necessárias i comparações, teremos

$$f(n) = 1 \times p_1 + 2 \times p_2 + 3 \times p_3 + \ldots + n \times p_n$$

ANÁLISE DE COMPLEXIDADE

- Assim, para calcular *f(n)*, seria necessário conhecer cada p_i. Vamos supor, em casos gerais, que todos os registros tenham a mesma probabilidade de serem procurados, então $p_i = 1/n$ para todo *i* entre *1* e *n*

$$f(n) = 1/n\ (1 + 2 + 3 + \ldots + n) = (n+1) / 2$$

- O **caso médio** é o valor médio entre o pior caso e o melhor caso quando considerados todos os registros equiprováveis de serem procurados

Desafio

1. Esse algoritmo é ótimo?
2. Se não, como ele poderia ser melhorado?
3. Tente implementar melhorias e demonstrar através da nova função de complexidade que ele é mais eficiente