# Análise de Complexidade em Algoritmos

Pedro Santos Costa
Eng. Comp Noturno 2º Período

Aanálise de algoritmos (ou análise de complexidade) é um mecanismo para entender e avaliar um algoritmo em relação aos critérios destacados, bem como saber aplica-los à problemas práticos. A complexidade de um algoritmo é analisada em termos de **tempo** e **espaço.**
O custo final de um algoritmo pode estar relacionado a diversos fatores: Tempo de execução, utilização de memória principal, utilização de disco, consumo de energia.

## Análise Matemática

A análise Matemática é de suma importância para a elaboração de um código, através dela é dada a projeção do consumo de hardware e tempo que um algoritmo tem de processamento. A partir dessa análise é definida qual a melhor opção e estratégia para tornar o código cada vez mais otimizado.
Comportamento assintótico representa o limite do comportamento do custo quando n cresce. A partir disso temos o **dominio assintótico:**

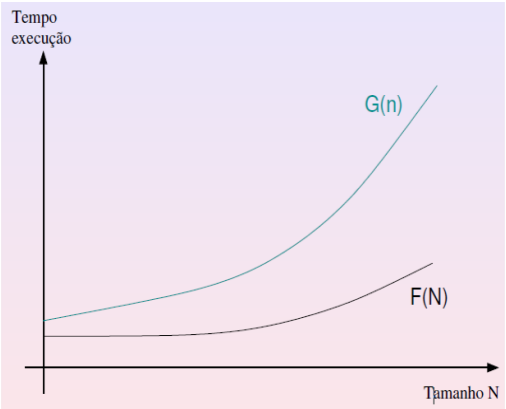
Tendo como exemplo as funções F e G:

Se F é sempre inferior a G, ou seja, o gráfico de F está sempre abaixo do gráfico de G, então escolhemos o algoritmo correspondente a F.

Se F as vezes é inferior a G, e vice-versa, e os gráficos de F e G se interceptam em um número infinito de pontos. Neste caso, temos um empate, ou seja, a função de custo não é suficiente para escolher entre os dois algoritmos.

## Comportamento assintótico

Se F as vezes é inferior a G, e vice-versa, e os gráficos de F e G se interceptam em um número finito de pontos. Desta forma, a partir de um certo valor n, F é sempre superior a G, ou é sempre inferior. Neste caso, escolhemos o algoritmo cuja função é inferior para grandes valores de n execução.



**Limites**

- Limite superior

Seja U (n) o tempo de execução de um algoritmo A (digamos), então g (n) é o Limite superior de A se houver duas constantes C e N tais que U (n) <= C * g (n) para n> N.

- Limite inferior

- Seja L (n) o tempo de execução de um algoritmo A (digamos), então g (n) é o **Limite Inferior** de A se houver duas constantes C e N tais que L (n)> = C * g (n) para n> N

**Tipos de análise assintótica**

- Notação O (big O)

Usamos a notação big-Θ para limitar de forma assintótica o crescimento do tempo de execução com fatores constantes acima e abaixo. Algumas vezes, vamos querer limitar somente acima. Seria conveniente ter uma forma de notação assintótica que diga "o tempo de execução cresce no máximo até um determinado valor, mas poderia crescer mais devagar". A notação "big-O" serve para essas ocasiões.

- Notação Θ (Theta)

Quando usamos a notação big-Θ estamos dizendo que temos um **limite assintoticamente restrito** no tempo de execução. "Assintoticamente" porque ele importa apenas para valores grandes de n. "Limite restrito" porque definimos o tempo de execução para um fator constante superior e inferior.

- Notação Ω (Ômega)

Algumas vezes, queremos dizer que um algoritmo leva *ao menos* uma certa quantidade de tempo, sem fornecer um limite superior. Usamos a notação Ω, que é a letra grega "omega". Se um tempo de execução é Ω(*f*(*n*)), então, para um n suficientemente grande, ele será ao menos *k* para uma constante k qualquer.

**Classes de Problemas**

| FUNÇÃO DE COMPLEXIDADE | n (tamanho do problema) | | |
|---|---|---|---|
| | 20 | 40 | 60 |
| n | 0.0002 s | 0.0004 s | 0.0006 s |
| $n \log_2 n$ | 0.0009 s | 0.0021 s | 0.0035 s |
| $n^2$ | 0.0040 s | 0.0160 s | 0.0360 s |
| $n^3$ | 0.0800 s | 0.6400 s | 2.1600 s |
| $2^n$ | 10.0000 s | 27 dias | 3660 séculos |
| $3^n$ | 580 minutos | 38550 séculos | $1.3*10^{14}$ séculos |

**Melhor caso, Pior caso e caso Médio**

- O de **melhor caso** corresponde ao de menor tempo de execução sobre todos os possíveis arranjos de entradas de tamanho n.
- O de **pior caso** corresponde ao de maior tempo de execução sobre todos os possíveis arranjos de entrada de tamanho n.
- O **caso médio**, ou **caso esperado**, corresponde à média dos tempos de execução de todas as entradas de tamanho n. É o mais difícil de ser obtido por depender de distribuições de probabilidade.