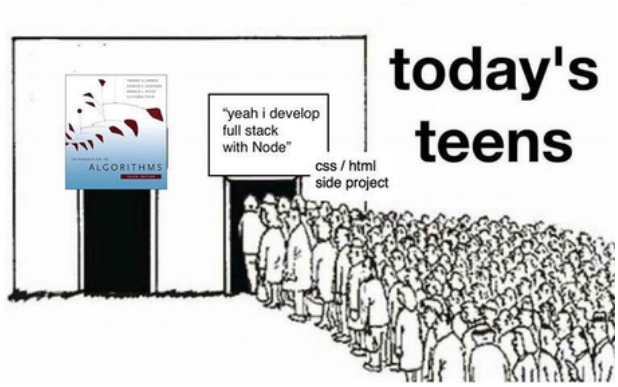
COMPLEXIDADE DE ALGORITMOS

YASMIN SOUZA CAMARGO

"ESSA SOLUÇÃO É SUFICIENTE?"
"A SOLUÇÃO É ESCALÁVEL OU LEGÍVEL?"

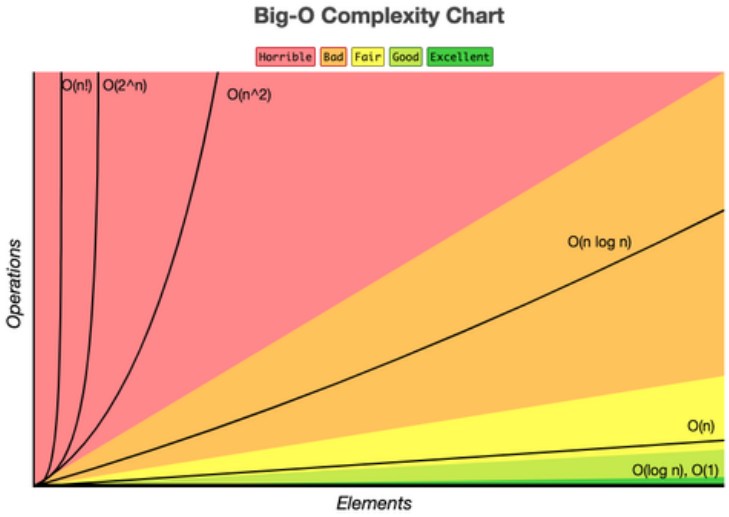
HTTPS://YOUTU.BE/NZFR4KC1IVU?T=437

- Relacionado com a quantidade de trabalho necessário para executar uma tarefa
- A complexidade de um algoritmo é analisada em termos de tempo e espaço
- A ideia é oferecer uma análise independente da máquina, compilador ou sistema



- O tempo que leva para um algoritmo ser executado é baseado no número de passos (Geralmente existe um N, que afeta o tempo de execução significativamente)
- Normalmente é diretamente proporcional ao tamanho dos dados a serem processados

Complexidade assintótica: quando o tamanho da entrada (n) tende a infinito, com isso, as constantes são ignoradas e apenas o componente mais significativo da função de complexidade é considerado



BASICAMENTE...

- Termos de menor grau são desprezados
- É analisado o pior caso (o maior número de operações usadas para qualquer entrada de tamanho n)

NOTAÇÃO O (BIG O NOTATION)

- A notação do O é sobre:

Limite superior (pior caso): Big O
Limite inferior (melhor caso): Big Omega (Ω)
Combinação de ambos: Big Theta (θ)

O(1) = O(yeah) - constante
O(log n) = O(nice) - logarítmica
O(n) = O(k) - linear
O(n log n) = O(k-ish) - logarítmica linear
O(n^2) = O(my) - quadrática
O(2^n) = O(no) - subexponencial
O(n^n) = O(fuck) - exponencial
O(n!) = O(mg!) - fatorial

COMPLEXIDADE CONSTANTE O(1)

- O número de operações não muda independente do tamanho da entrada (n).
- Exemplo: uma multiplicação, adição e divisão
- As instruções são executadas um número fixo de vezes
- Quanto maior a entrada fica, menos importantes se tornam as constantes
- As constantes de um algoritmo são normalmente ignoradas

COMPLEXIDADE LOGARÍTMICA O(LOG N)

- Diminui o N toda vez que um processamento é feito (transformam um problema em outros menores)
- Quando n é mil, log2n ≈ 10, quando n é 1 milhão, log2n ≈ 20
- Exemplo: busca binária

COMPLEXIDADE LINEAR O(N)

- É a melhor situação possível para um algoritmo que tem de processar/produzir n elementos de entrada/saída
- Quanto maior o input (n), maior o tempo de execução do algoritmo (maior o número de operações que serão feitas)
- Cada vez que n dobra de tamanho, o tempo de execução dobra.
- A a parte linear dominará sobre as constantes
- Exemplo: busca sequencial, calcular fatorial

COMPLEXIDADE LOGARÍTMICA LINEAR O(N LOG N)

- Típico em algoritmos que quebram um problema em outros menores, resolvem cada um deles independentemente e ajuntando as soluções depois.
- Quando n é 1 milhão, nlog2n é cerca de 20 milhões.
- Quando n é 2 milhões, nlog2n é cerca de 42 milhões, pouco mais do que o dobro

COMPLEXIDADE QUADRÁTICA O(N^2)

- Ocorrem quando os itens de dados são processados aos pares
- A complexidade quadrática desenha uma curva (parábola) em relação ao eixo de tempo para cada vez que N aumenta
- Quando n é mil, o número de operações é da ordem de 1 milhão
- Exemplo: dois for aninhado
- As complexidades quadrática e cúbica são chamadas de polinomiais
- Úteis para resolver problemas de tamanhos relativamente pequenos
- Exemplo: imprimir uma matriz

COMPLEXIDADE EXPONENCIAL O(2^N)

- Geralmente não são úteis sob o ponto de vista prático.
- Ocorrem na solução de problemas quando se usa força bruta para resolvê-los
- Quando n é 20, o tempo de execução é cerca de 1 milhão

COMPLEXIDADE FATORIAL O(N!)

- Geralmente ocorrem quando se usa força bruta para na solução do problema

REFERENCIAS:

https://homepages.dcc.ufmg.br/~chaimo/paa/Aula%202%20-%20Complexidade%20Assintotica.pdf

https://pt.stackoverflow.com/questions/33319/o-que-%C3%A9-a-complexidade-de-um-algoritmo

https://medium.com/nagoya-foundation/introdu%C3%A7%C3%A3o-%C3%A0-complexidade-de-algoritmos-4a9c237e4ecc