/*

TRABALHO TEORICO 5

PONTIFICIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

ALUNO: JULIA VELOSO DIAS ID: 1314675

PROFESSOR: MAX DO VAL MACHADO

CURSO: CIENCIA DA COMPUTAÇÃO // TURNO: MANHÃ // PERIDO: SEGUNDO

*/

RESUMO --- NOTAÇÕES $O$, $\Omega$ e $\Theta$

É usada, em ciência da computação para medir a quantidade de tempo de acordo com o número N de entrada, em matemática é o limite da expressão.

Usa-se a representação da letra O seguida do valor da taxa de crescimento, que dependendo do seu grau é como se comportará no gráfico.

Existem alguns níveis de complexidade:

O(1) - onde não depende do número de entradas, então não há diferença de crescimento

O(LOGN) - logaritmo, quando o número de operações é menor que o de itens

O(N) - linear, quando há proporcionalidade entre itens e operação

O(N LOGN) - quando log n é executada n vezes

O(N^2) - quando há duas repetições(loops) e processadas por pares, quando em números grandes, tendem a ser ruim pois gastam muito tempo

O(2^N) - há um aumento exponencial de acordo com o aumento do valor de n

O(N!) - cresce rapidamente mesmo com números menores

FONTE:

https://dev.to/danielle8farias/complexidade-de-algoritmos-notacao-big-o-26al

https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/Oh.html

```
//
for (int i = 3; i < n; i++){
a--;
}
```

N – 3 subtraçoes

O(n), Ω(n) e **Θ**(n)

```
//
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++){
a--;
}
}
```

6 vezes

O(1), Ω(1) e **Θ**(1)

```
//
int i = 1, b = 10;
while (i > 0){ //
b--;
i = i >> 1;
}
i = 0;
while (i < 15) {
b--;
i += 2;
}
```

Realizara 9 subtraçoes

O(1), Ω(1) e **Θ**(1)

```
//
for (int i = 0; i < n; i++) // n vezes
for (int j = 0; j < n - 3; j++) // n (de i true) - 3
a *= 2; // se ambos verdadeiros realiza n multiplicaçoes de A X 2
```

(n – 3) * n

n^2 - 3n

O(n^2), Ω(n^2) e Θ(n^2)

```
//
for (int i = n - 7; i >= 1; i--)
for (int j = 0; j < n; j++)
a *= 2;
```

(n – 7) * n

n^2 - 7n

O(n^2), Ω(n^2) e Θ(n^2)

```
//
for (int i = n; i > 0; i /= 2) //log 2
a *= 2;
```

Teto (arredondamento para cima) lg(n)

O(lg(n)), Ω(lg(n)) e Θ(lg(n))

```
//
for (int i = n+4; i > 0; i >>= 1)
a *= 2;
```

(n – 4) * n

n^2 - 4n

O(n^2), Ω(n^2) e Θ(n^2)

//

```
for (int i = n - 7; i >= 1; i--) // enquanto verdadeiro
for (int j = n - 7; j >= 1; j--)
a *= 2;
```

(n – 7) * n

n^2 – 7n

O(n^2), Ω(n^2) e Θ(n^2)

//

a) 3n + 2n^2  -- O(n^2), Ω(n^2) e Θ(n^2)

b) 5n + 4n^3 -- O(n^3), Ω(n^3) e Θ(n^3)

c) lg(n) + n -- O(lg(n)), Ω(lg(n)) e Θ(lg(n))

d) 2n^3 + 5 -- O(n^3), Ω(n^3) e Θ(n^3)

e) 9n^4 + 5n^2 + n/2 -- O(n^4), Ω(n^4) e Θ(n^4)

f) lg(n) + 5 lg(n) -- O(lg(n)), Ω(lg(n)) e Θ(lg(n))