Análise de Complexidade

1. O que significa dizer que uma função g(n) é O(f (n))?
g(n) é da ordem de no máximo f(n), f(n) domina g(n) assintoticamente, ou seja, para números grandes cf(n) será sempre maior que g(n), para alguma constante c.

2. Explique a diferença entre O(1) e O(2).
Algoritmos O(1) têm complexidade constate, eles têm o mesmo desempenho independente do valor de n; na análise de complexidade, *O*(2) não é uma notação comum e não representa uma complexidade específica.

3. Indique se as afirmativas a seguir são verdadeiras ou falsas:

(a) $2^{n+1} = O(2^n)$
verdadeira

(b) $2^{2n} = O(2^{2n})$
verdadeira

(c) *f(n) = O(u(n)) e g(n) = O(v(n))*→ $f(n) + g(n) = O(u(n)) + O(v(n))$
verdadeira

4. Se os algoritmos A e B levam tempo a(n) = n^2 − n + 549 e b(n) = 49n + 49.
a) a(n) = O(b(n))?
Não, b(n) não domina a(n) assintoticamente

b) b(n) = O(a(n))?
Sim, a(n) domina b(n) assintoticamente, pois um crescimento quadrático é considerado pior que um crescimento linear.

5. Considere um algoritmo para inserir um elemento em um arranjo ordenado de elementos.
a) Qual o número mínimo de passos para resolver este problema?
O número mínimo de passos para resolver este problema é 1, quando o elemento que está sendo inserido já está na posição correta no arranjo e não é necessário fazer nenhuma movimentação.

b) Qual o melhor caso?
Quando o elemento a ser inserido é menor que todos os outros elementos e tem que ser inserido na primeira posição.

c) Qual o pior caso?
Quando o elemento a ser inserido é maior que todos os outros elementos e tem que ser inserido na última posição.

d) Qual o caso médio?
Quando o elemento a ser inserido está entre a menor e a maior posição no arranjo ordenado.

6. Se os algoritmos A e B levam tempo a(n) = n^2 − 2 + 549 e b(n) = n^2 + 30n.
a) a(n) = O(b(n))?
Sim, b(n) domina assintoticamente a(n).

b) b(n) = O(a(n))?
Não, a(n) não domina assintoticamente b(n).
c) Estes dois algoritmos em algum momento (dependente do tamanho de n) irão dar o mesmo resultado? Se Sim para qual o valor de n?

$$n^2 - 2 + 549 = n^2 + 30n$$
$$547 = 30n$$
$$n = \frac{547}{30}$$
$$n = 18,23333...$$

d) Para quais valores A leva menos tempo para executar do que B?
Para n<18,23.

7. Qual a ordem de complexidade das sentenças abaixo:
a) $f(n) = 4n + n - 2 + 3$
$O(n)$

b) $g(n) = 4n^2 + 3n^3 + 2n - 2$
$O(n^3)$

c) $h(n) = 2^{2n} + 4n^3$
$O(2^{2n})$

d) $i(n) = 2^{5n} + 4n^2$
$O(2^{5n})$

8. Faça um método que receba um número inteiro n e efetue o número de multiplicações, pedido nos casos a seguir:
a) $5n + 4n^3$

```
int operacao (int n){
   int resultado=5*n+4*n*n*n;
   return resultado;
}
```

b) $9n^4 + 5n^2 + n/2$

```
float operacao (int n){
   float resultado=9*n*n*n*n + 5*n*n + (float)n/2;
   return resultado;
}
```

c) $4n^3 + 2$

```
int operacao (int n){
   int resultado = 4*n*n*n+2;
   return resultado;
}
```

d) $lg(n) + n^2$

```
#include <math.h>
float operacao (int n){
   float resultado=log2(n)+n*n;
   return resultado;
}
```

e) $3lg(n) + lg(n)$

```
#include <math.h>
float operacao (int n) {
   float resultado=3*log2(n)+log2(n);
   return resultado;
}
```

f) $2n + 2n^2 + lg(n)$

```
#include <math.h>
float operacao (int n){
   float resultado=2*n+2*n*n+log2(n);
   return resultado;
}
```

9. Apresente a função e a taxa de complexidade para as 3 notações vistas em sala, referente ao número de comparações, para o pior e melhor caso, para as opções a seguir:

(a)

```
void imprimirMaxMin(int array[], int n){
  int maximo, minimo;

  if (array[0]>array[1]){
    maximo=array[0];
    minimo=array[1];
  }
  else {
    maximo=array[1];
    minimo=array[0];
  }

  for (int i=2; i<n; i++){
    if (array[i]>maximo){
      maximo=array[i];
    }
```

```
    else if (array[i]<minimo){
      minimo=array[i];
    }
  }
}
```

Notação O(1):
Pior Caso: O primeiro elemento do array é maior que o segundo.
Melhor Caso: O primeiro elemento do array é menor que o segundo.

Notação O(n):
Pior Caso: Nenhum dos elementos é maior que o atual máximo e nenhum é menor que o atual mínimo.
Melhor Caso: O primeiro elemento é maior que o segundo.

Notação O(n^2):
A complexidade não é O(n^2) para nenhum caso nesta função.

(b)i=0;

```
while (i<n){
  i++:
  a--;
}

if (b>c){
  i--;
}
else {
  i--;
  a--;
}
```

A complexidade pode variar dependendo das características específicas dos valores e expressões utilizados no código.

(c)

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

Melhor caso: O melhor caso ocorre quando n é inicialmente menor ou igual a zero. A complexidade é O(1).

Pior caso: O pior caso ocorre quando n é grande e ambos os loops são executados completamente. A complexidade é O(n²).

Caso médio: O caso médio pode ser afetado pela distribuição de valores de n e o comportamento das variáveis a, b e c dentro do loop. A complexidade é O(n^2).

10. Apresente o tipo de crescimento que melhor caracteriza as funções abaixo:

	Constante	Linear	Polinomial	Exponencial
$3n$		x		
1	x			
$(3/2)n$		x		
$2n^3$			x	
2^n				x
$3n^2$			x	
1000	x			
$(3/2)^n$				x

11. Classifique as funções $f1(n) = n.lg(n)$, $f2(n) = lg(n)$, $f3(n) = 8n^2$, $f4(n) = 64$, $f5(n) = 6n^3$, $f6(n) = 8^{2n}$ e $f7(n) = 4n$ de acordo com o crescimento, do mais rápido para o mais lento.
f6>f5>f1>f3>f7>f2>f4.