Relatório Trabalho 1

Nome: Carlos Eduardo Santos Oliveira. (22111154-5)

Turma: 010

Introdução:

Para o desenvolvimento do desafio proposto no relatório foram criadas duas classes, que são a classe App e a classe Funções. O objetivo deste trabalho é implementar e analisar os algoritmos fornecidos no enunciado, e assim descobrir a complexidade do algoritmo e representar de forma gráfica o seu desempenho. Para o desenrolar do trabalho é válido relembrar as complexidades que estamos utilizando, começando da mais eficiente para a menos eficiente. Começamos pela complexidade Constante (1), seguida por Logaritmo (Log n), Linear (n), n Log n, Quadrática (n^2), Cúbica (n^3) e por fim Exponencial (x^n) a qual possui o pior desempenho.

Classe App:

```java
public static void main(String[] args) throws Exception{
    Funções func= new Funções();

    System.out.println(x:"Algoritmo 1:");
    for(int n=1; n<1000; n+=50) {
        int r = func.F1(n);
        System.out.println(n+";"+r);
    }

    System.out.println(x:"Algoritmo 2:");
    for(int n=1; n<1000; n+=50) {
        int r = func.F2(n);
        System.out.println(n+";"+r);
    }

    System.out.println(x:"Algoritmo 3:");
    for(int n=1; n<1000; n+=50) {
        int r = func.F3(n);
        System.out.println(n+";"+r);
    }

    System.out.println(x:"Algoritmo 4:");
    for(int n=1; n<50; n+=5) {
        int r = func.F4(n);
        System.out.println(n+";"+r);
    }

    System.out.println(x:"Algoritmo 5:");
    for(int n=1; n<200; n+=15) {
        int r = func.F5(n);
        System.out.println(n+";"+r);
```

A classe app foi implementada para chamar as funções da classe Função e imprimir os resultados da execução de cada algoritmo. Cada “for” executa pulando o valor de 50 em 50, excetos pelos códigos que tem um desempenho muito baixa, pois senão o tempo de execução seria alto demais.

Classe Funções:

```
public class Funções{

    public int F1(int nroOp){
        int i,j,k,res= 0;                                   //
        int cont_op= 0;                                     //
                                                            //
        for(i=1; i<=nroOp+1; i+=1){                         //
            for(j=1; j<=i*i; j+=i+1){                       //
                for(k=i/2; k<=nroOp+j; k+=2) {              //
                    res= res+nroOp-1;                       //
                    cont_op++;                              //
                }                                           //
            }                                               //
        }                                                   //
        return cont_op;                                     //
    }

    public int F2(int nroOp){                               //
        int i,j,k,res= 0;                                   //
        int cont_op= 0;                                     //
                                                            //
        for(i=nroOp; i<=nroOp; i+=i/2+1){                   //
            for(j=i/2; j<=i*i; j+=i+1){                     //
                for(k=nroOp; k<=2*nroOp; k+=i+1){           //
                    res = res + nroOp;                      //
                    cont_op++;                              //
                }                                           //
            }                                               //
        }                                                   //
        return cont_op;                                     //
    }
```

```
public int F3(int nroOp){
    int i,j,k,res= 0;                                       //
    int cont_op= 0;                                         //
                                                            //
    for(i=1; i<=nroOp*nroOp; i+=2){                         //
        for(j=i/2; j<=2*i; j+=i/2+1){                       //
            for(k=j+1; k<=nroOp+j; k+=k/2+1){               //
                res= res+Math.abs(j-i);                     //
                cont_op++;                                  //
            }                                               //
        }                                                   //
    }                                                       //
    return cont_op;                                         //
}

public int F4(int nroOp){
    int i,j,k,res= 0;                                       //
    int cont_op= 0;                                         //
                                                            //
    for(i=nroOp; i<=nroOp*nroOp; i+=2){                     //
        for(j=nroOp+1; j<=nroOp*nroOp; j+=2){               //
            for(k=j; k<=2*j; k+=2){                         //
                res= res+1;                                 //
                cont_op++;                                  //
            }                                               //
        }                                                   //
    }                                                       //
    return cont_op;                                         //
}
```

```java
public int F5(int nroOp){
    int i,j,k,res= 0;                                   //
    int cont_op= 0;                                     //
                                                        //
    for(i=1; i<=nroOp*nroOp; i+=1){                     //
        for(j=1; j<=i; j+=2){                           //
            for(k=nroOp+1; k<=2*i; k+=i*j){             //
                res= res+k+1;                           //
                cont_op++;                              //
            }                                           //
        }                                               //
    }                                                   //
    return cont_op;                                     //
}
```

A classe Funções possui a implementação dos algoritmos a serem analisados.

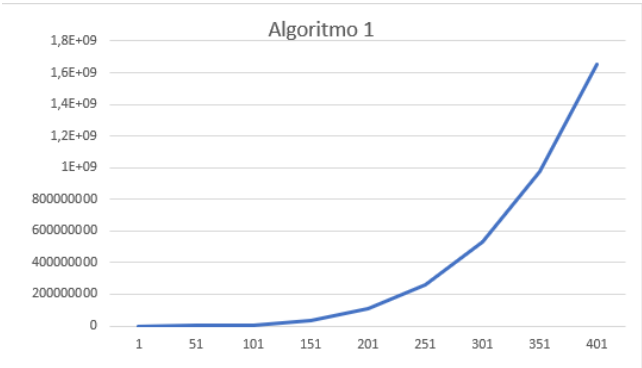
Análise dos Algoritmos:

Para analisar os algoritmos utilizaremos, a contagem de operações, o gráfico e uma equação que diz aproximadamente o valor do expoente da função.

Algoritmo 1:

O primeiro algoritmo possui complexidade Quadrática (n^4), cujo possui um desempenho bom inicialmente, porém conforme aumenta o número de operações, diminui o desempenho.

1	6
51	499343
101	7079956
151	34398018
201	106484906
251	256746693
301	527964856
351	972295368
401	1651269806



Cálculo do grau da função:

$$\frac{\log(1651269806)-\log(499343))}{\log(401)-\log(51)} \approx 3{,}92979$$

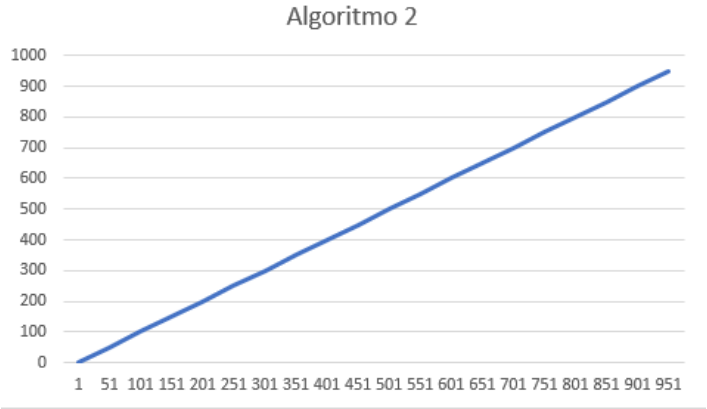
Portanto, $f(n)= n^4$.

Por mais que o gráfico seja similar com o de uma função n^2 o número de execuções é extremamente maior. Podemos observar o gigantesco salto nas operações, executando 1 vez a saída é 6, porém ao executar 51 vezes a saída aumenta milhares de vezes.

Algoritmo 2:

O segundo algoritmo é de complexidade linear (n), é uma função rápida e com ótimo desempenho.

1	1
51	50
101	100
151	150
201	200
251	250
301	300
351	350
401	400



Cálculo do grau da função:

$\frac{(\log(400)-\log(50))}{\log(401)-\log(51)} \approx 1{,}008$

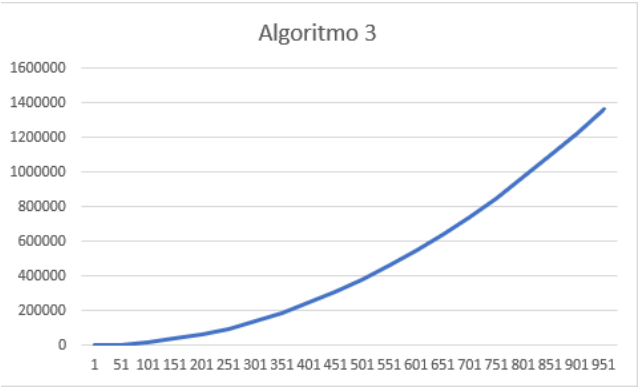
Portanto, f(n)= n.

De acordo com o gráfico e com a equação podemos afirmar que o algoritmo possui uma baixa complexidade e possui baixo custo de processamento.

Algoritmo 3:

O terceiro algoritmo possui complexidade Quadrática (n^2). Assim como o primeiro analisado, tem um desempenho bom no início, mas piora conforme executa mais vezes.

1	3
51	4222
101	15973
151	35225
201	61979
251	96236
301	137991
351	187246
401	244005



Cálculo do grau da função:

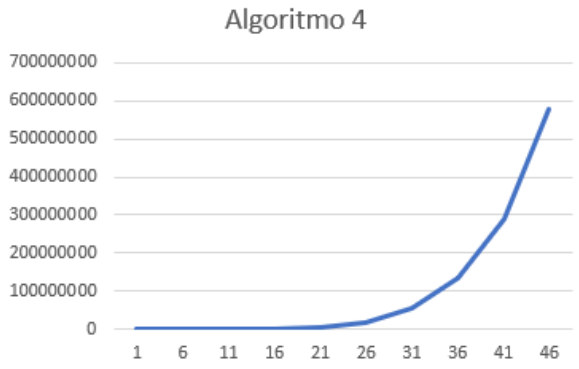
$\frac{(\log(244005)-\log(4222))}{\log(401)-\log(51)} \approx 1{,}96$

Portanto, f(n)= n^2.

Algoritmo 4:

O quarto algoritmo é de complexidade Cúbica (n^6), esse código possui um desempenho baixíssimo sendo apenas pior que a complexidade exponencial.

1	0
6	2640
11	104720
16	994620
21	5162115
26	18647200
31	53955810
36	132576255
41	290494430



Cálculo do grau da função:

$$\frac{(\log(290494430)-\log(2640))}{\log(41)-\log(6)} \approx 6{,}04$$

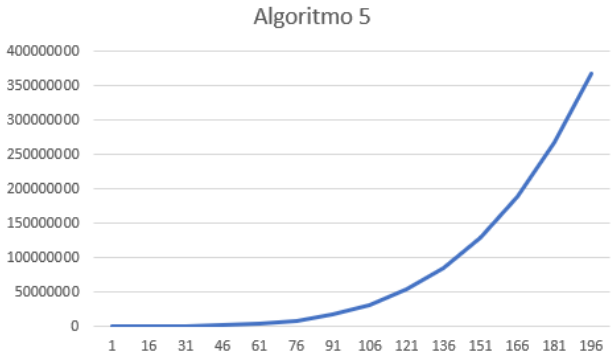
Portanto, $f(n)= n^6$.

Analisando a tabela e o gráfico vemos que o algoritmo é extremamente pesado e com um custo de processamento altíssimo, executando apenas 41 vezes a saída é de 290 milhões.

Algoritmo 5:

O quinto e último algoritmo é de complexidade Quadrática (n^4), assim como o primeiro e o terceiro.

1	1
16	16732
31	232227
46	1122348
61	3466741
76	8348752
91	17155542
106	31577943
121	53610631



Cálculo do grau da função:

$$\frac{(\log(53610631)-\log(16732))}{\log(121)-\log(16)} \approx 3{,}98$$

Portanto, $f(n)= n^4$.

Possui um baixo desempenho, ficando mais lento conforme executa mais vezes.

Conclusão:

Analisar um algoritmo pode ser uma tarefa complicada caso seja feita de forma profunda, pois muitos fatores devem ser levados em consideração. A fim de compreender o desempenho dos algoritmos desse trabalho observamos principalmente a tabela resultante da execução do mesmo, assim como o gráfico obtido da tabela e a equação do grau da função. É sempre interessante desenvolver códigos visando um bom desempenho para que futuramente não sejam necessárias muitas otimizações. Com esse trabalho aprendi que é importante tomar cuidado ao fazer códigos, pois dependendo da forma como se utiliza o “for”, você pode acabar fazendo seu código ter um desempenho extremamente baixo.