PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

BACHARELADO EM ENGENHARIA DE SOFTWARE

# Algoritmos e Estruturas de Dados I:

Trabalho 1

MARIA EDUARDA WENDEL MAIA

PORTO ALEGRE

2023

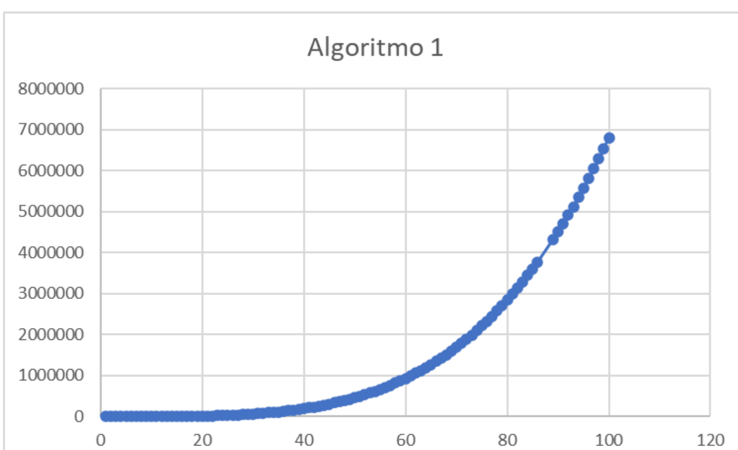
# Sumário:

## Algoritmo 1:

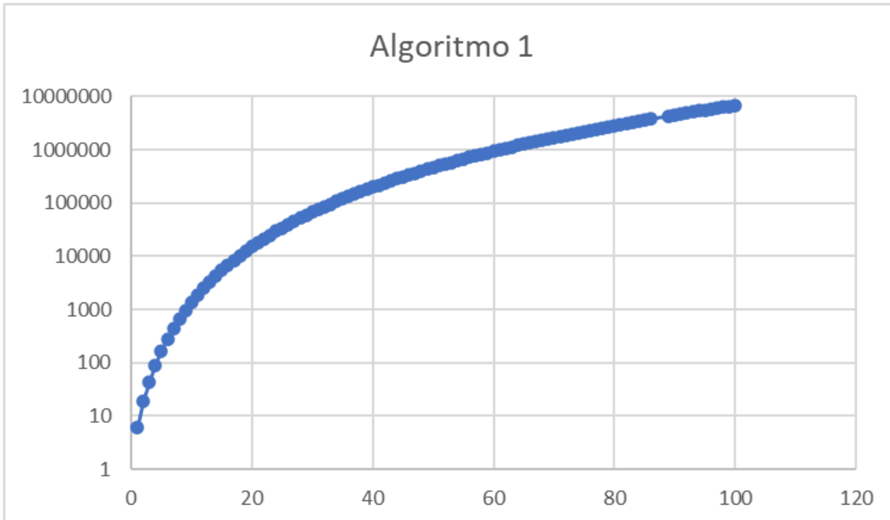
```
public class Main {
	public static void main (String[] args){
		for(int i = 1 ; i <= 100; i++){
			System.out.println(i + "\t" + f(i));
	}
}

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

- Desenvolvimento:

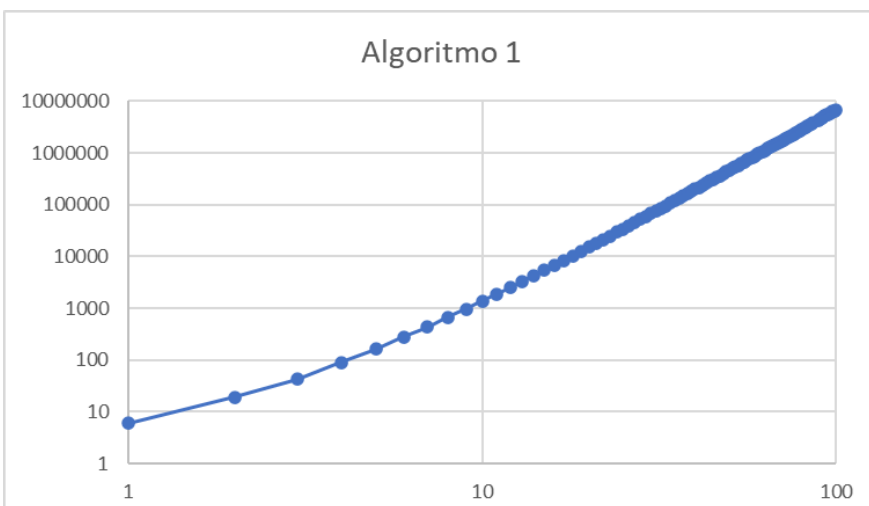
A planilha do Excel recebeu os dados resultantes da execução do código acima e, a partir desses dados, foi gerado o gráfico a seguir:



O resultado apresentado abaixo foi obtido após a aplicação da escala logarítmica no eixo y do gráfico clicando sobre o Eixo Vertical, selecionando formatar eixo e marcando a escala logarítmica na base 10:




e para obter a escala logarítmica no eixo x do gráfico cliquei sobre o Eixo Horizontal, selecionei formatar eixo e marquei a escala logarítmica na base 10:




E para obter a função e em qual grau ela cresce utilizei a fórmula:

b = (log(6.809.401) – log(6)) / (log(100) – log(1))

b = (log(6.809.401) – log(6)) / (log(100) – log(1))

b = (log10(6.809.401) – log10(6)) / (log10(100) – log10(1))

b = (6.832.508 - 0.778151) / (2 - 0)

b = 6.054357 / 2

b = 3.0271785

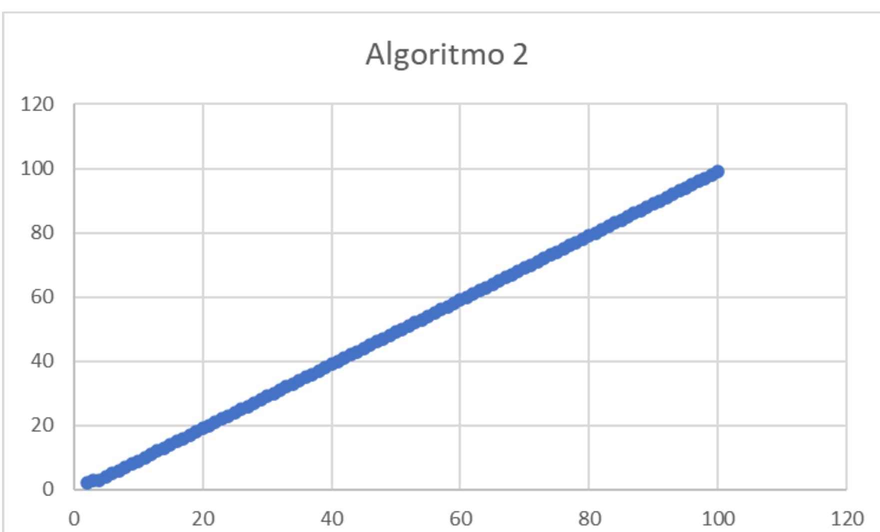
Portanto, o valor de b é aproximadamente 3.0271785 ou f(n) ≈ $n^{3,02}$, assim podemos concluir que esse algoritmo é polinomial pois a quantidade de operações que ele realiza é proporcional a um polinômio de grau 3 (função cúbica), e foram utilizados esses valores na fórmula pois a f(1)= 6 e no f(100)= 6.809.401. Pela função ser cúbica, acaba sendo considerada em alguns casos ineficiente para tamanhos de entradas grandes, pois o tempo de execução cresce rapidamente à medida que o tamanho da entrada aumenta, podendo resultar em tempos de execução muito longos para entradas maiores, tornando o algoritmo inviável em muitos casos.

## Algoritmo 2:

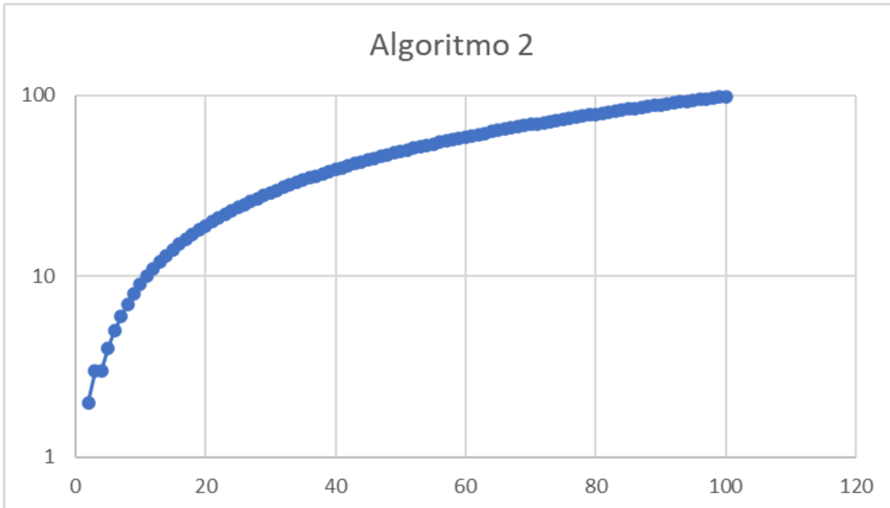
```
public class Main2 {
	public static void main (String[] args){
		for(int i = 2 ; i <= 100; i++){
		System.out.println(i + "\t" + f(i));
		}
	}

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

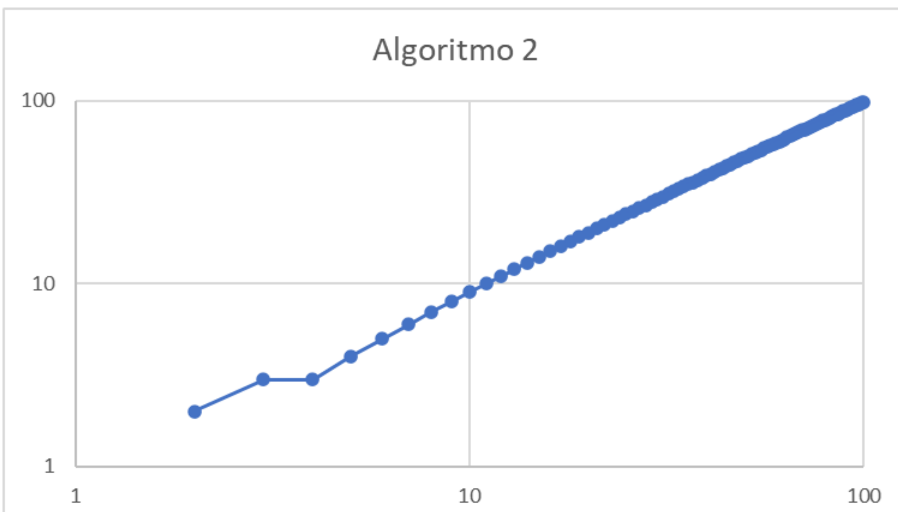
- Desenvolvimento:

A planilha do Excel recebeu os dados resultantes da execução do código acima e foi gerado o gráfico a seguir:



O resultado apresentado abaixo foi obtido após a aplicação da escala logarítmica na base 10 no eixo y e depois no eixo x, igual realizado no algoritmo 1:




E para obter a função e em qual grau ela cresce utilizei a fórmula:

b = (log(99) – log(2)) / (log(100) – log(2))

b = log(99/2) / log(100/2)

b = log(49.5) / log(50)

b =log(49.5) / log(50)

$$b = \log10(49.5) / \log10(50)$$

$$\log10(49.5) \approx 1.6946$$

$$\log10(50) \approx 1.69897$$

$$b \approx 1.6946 / 1.69897$$

$$b \approx 0.99744$$

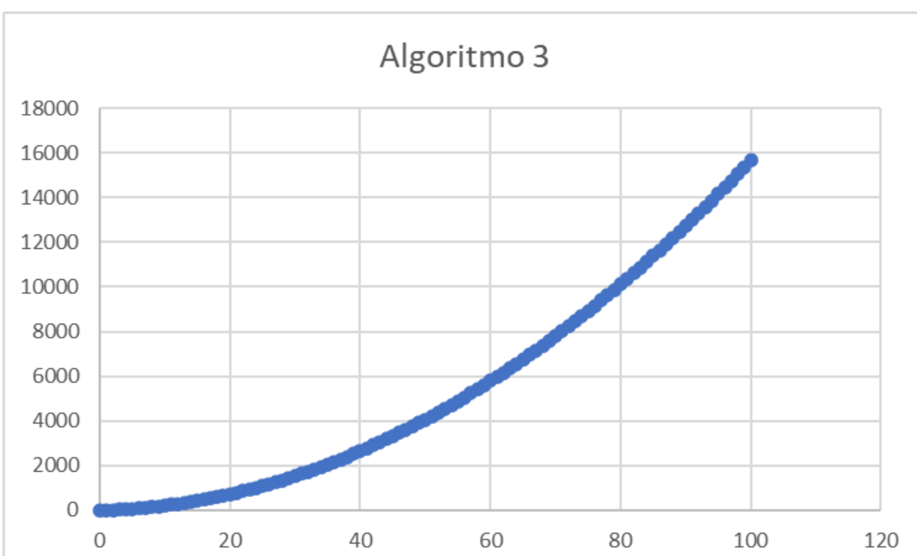
Portanto, o valor de b é aproximadamente 0.99744 sendo $f(n) \approx n^1$, é uma função linear pois a sua taxa de variação é sempre aproximadamente 1, e temos nela apenas o termo n, que é de primeiro grau. As funções lineares são algoritmos simples e eficientes para realizar tarefas que envolvem cálculos lineares e são muito rápidos para se executar. Foram utilizados esses valores na fórmula pois a função no 2 é igual a 2 e no 100 é igual a 99 ($f(2)= 2$ e no $f(100)= 99$).

## Algoritmo 3:

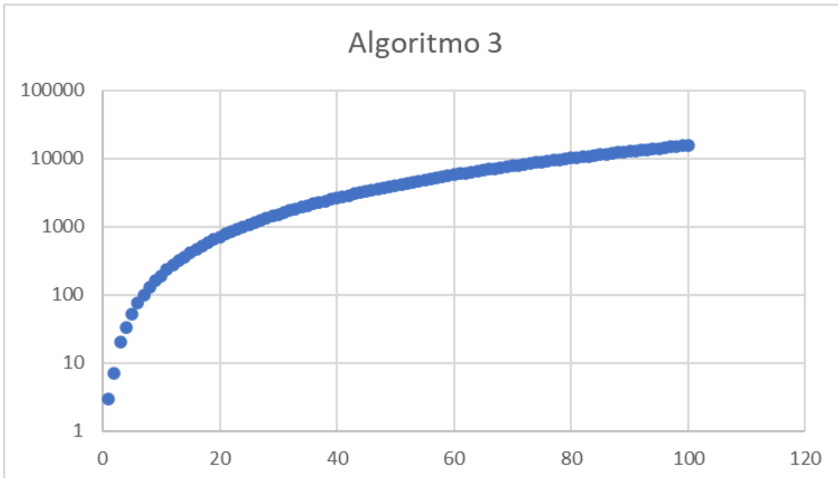
```
public class Main3 {
	public static void main (String[] args){
		for(int i = 0 ; i <= 100; i++){
		System.out.println(i + "\t" + f(i));
		}
	}

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

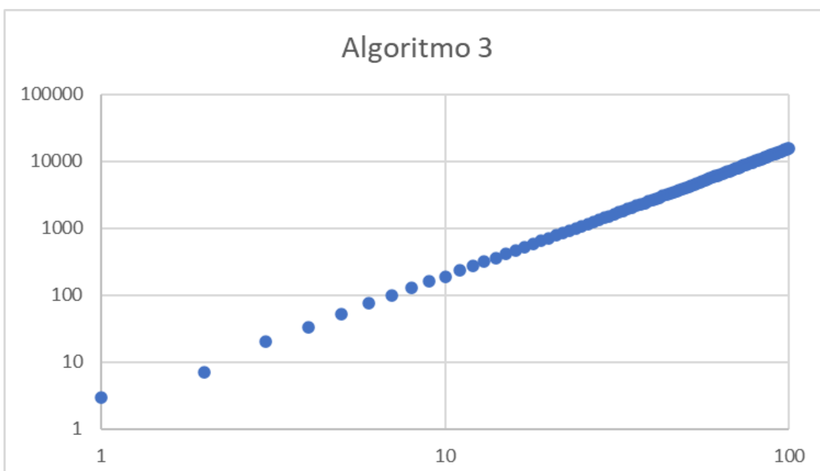
- Desenvolvimento:

A planilha do Excel recebeu os dados resultantes da execução do código acima e foi gerado o gráfico a seguir:



O resultado apresentado abaixo foi obtido após a aplicação da escala logarítmica na base 10 no eixo y e depois no eixo x, igual realizado no algoritmo 1 e 2:




E para obter a função e em qual grau ela cresce utilizei a fórmula:

b = (log(15662) – log(3)) / (log(100) – log(1))

b = (log(15662) - log(3)) / (log(100) - log(1))

b = log(15662/3) / log(100/1)

15662/3 = 5220.6667

100/1 = 100

b = log(5220.6667) / log(100)

b = 3.7185 / 2 b = 1.85925

b ≈ 1.85925.

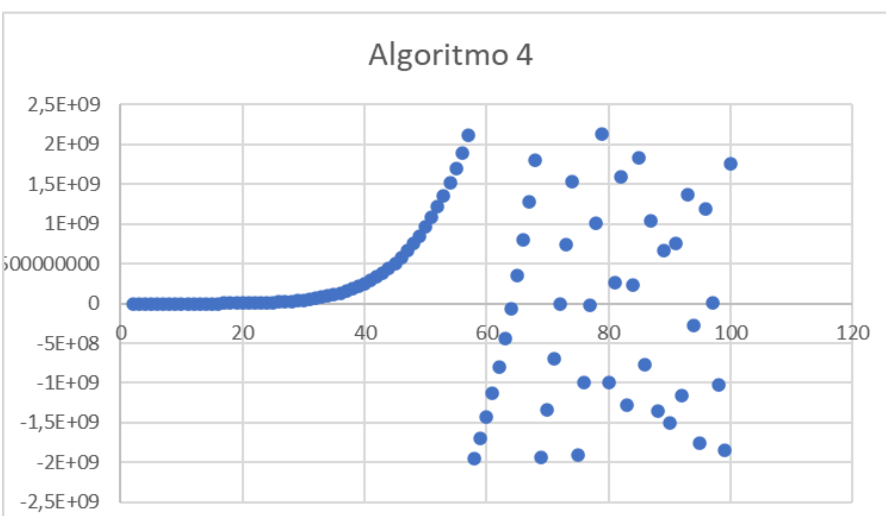
Portanto, o valor da função é f(n)≈ $n^{1,85}$, sendo assim ela é uma função linear que é igual a $n^1$, pois mesmo o expoente está mais perto de 2 ainda não cresce de maneira quadrática até virar dois. Algoritmos lineares são geralmente considerados eficientes, especialmente quando comparados a algoritmos com complexidade maior, à medida que o tamanho da entrada aumenta, o tempo de execução do algoritmo aumenta linearmente. Foram utilizados esses valores na fórmula pois a função no 1 é igual a 3 e no 100 é igual a 15.662 (f(1)= 3 e no f(100)= 15.662).

## Algoritmo 4:

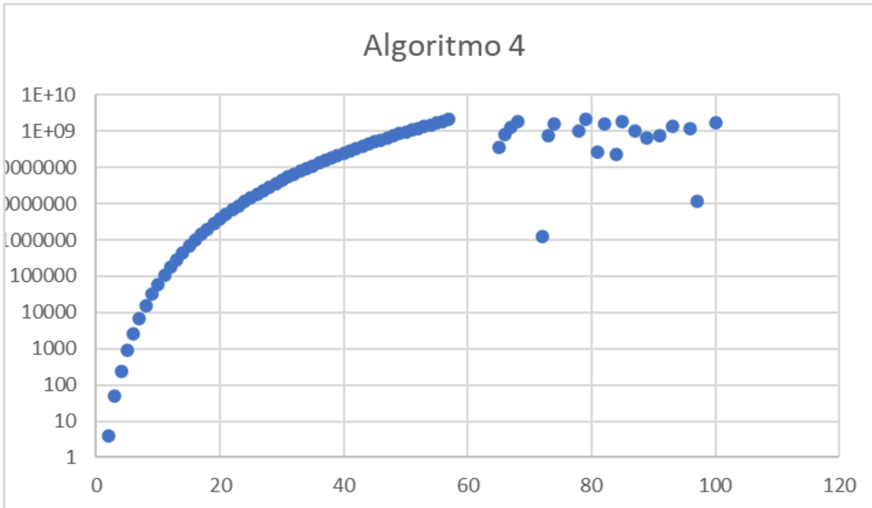
```
public class Main4 {
	public static void main (String[] args){
		for(int i = 2 ; i <= 100; i++){
		System.out.println(i + "\t" + f(i));
		}
	}

	public static int f( int n ) {
		int cont_op = 0;
		int i, j, k, res = 0;
		for( i = n; i <= n*n; i += 2 )
			for( j = n+1; j <= n*n; j += 2 )
			for( k = j; k <= 2*j; k += 2 ) {
				res = res + 1;
				cont_op++;
			}
		return cont_op;
	}
}
```

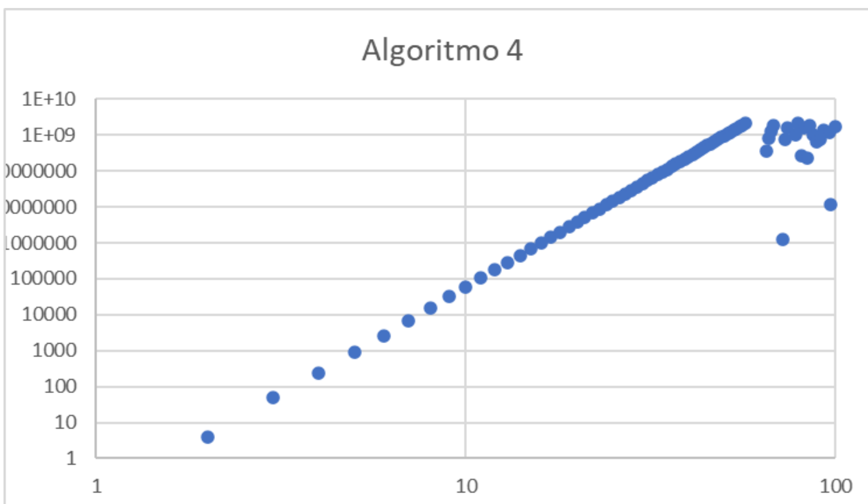
- Desenvolvimento:

A planilha do Excel recebeu os dados resultantes da execução do código acima e foi gerado o gráfico a seguir:



O resultado apresentado abaixo foi obtido após a aplicação da escala logarítmica na base 10 no eixo y e depois no eixo x, igual realizado no algoritmo 1,2 e 3:




E para obter a função e em qual grau ela cresce utilizei a fórmula:

b = (log(25014250) – log(1)) / (log(100) – log(1))

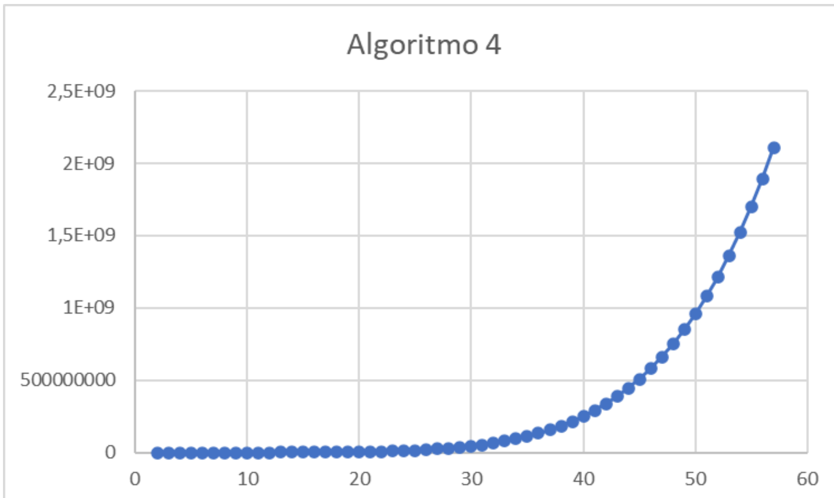
b = log(25014250) / log(100)

b = log10(25014250) / log10(100)

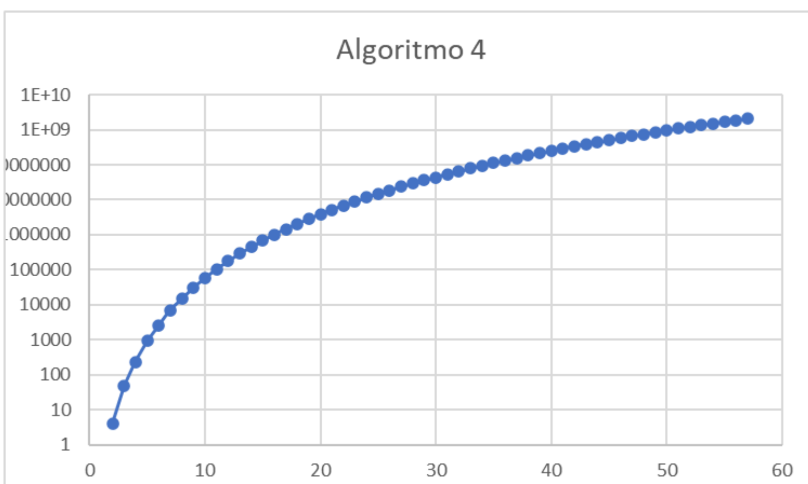
b = 7.3983 / 2

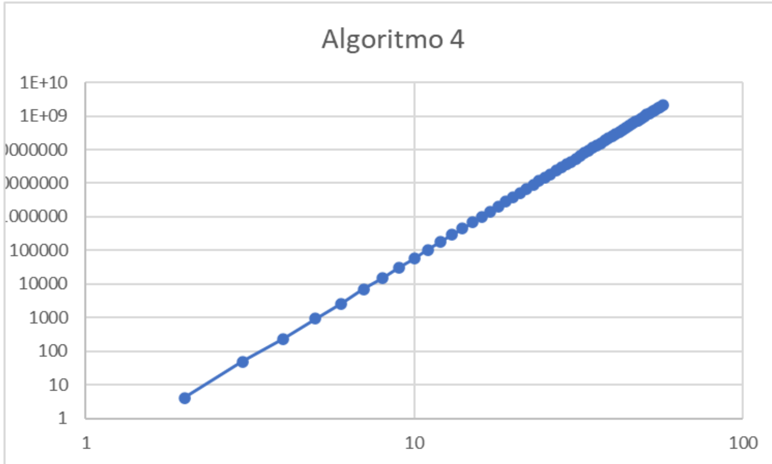
b = 3.69915

Depois realizei o gráfico até o 57 pois depois começam a aparecer os números negativos, gerando o seguinte gráfico:




e realizei a escala logarítmica na base 10 no eixo y e depois no eixo x:






E para obter a função e em qual grau ela cresce utilizei a fórmula:

b = (log(2109141930) – log(4)) / (log(57) – log(2))

b = log(2109141930/4) / log(57/2)

b = log(527285482.5) / log(28.5)

b = log(527285482.5) / log(28.5)

b = 8.725 / 1.454

b = 5.995.

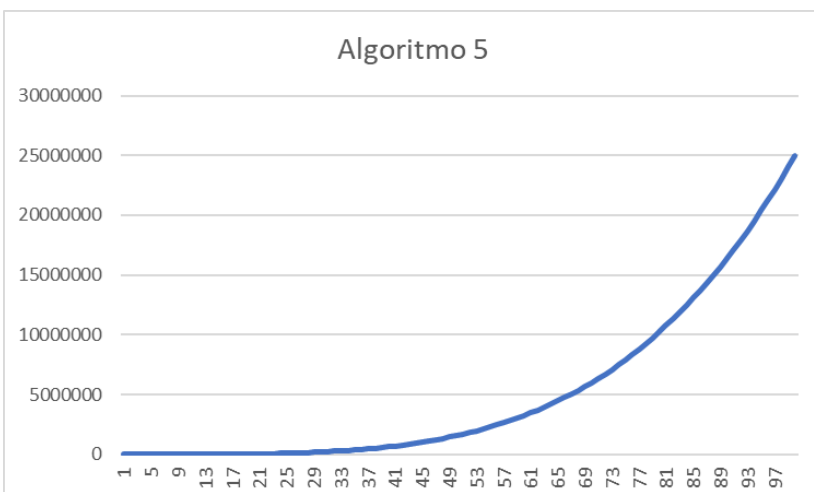
Portanto, o valor da função é f(n)≈ $n^6$ , sendo assim ela é uma função polinomial a $n^6$. Pela função ser elevada ao expoente 6, o desempenho do mesmo não é tão rápido e eficiente quanto o esperado, pois o tempo de execução cresce rapidamente à medida que o tamanho da entrada aumenta, podendo resultar em tempos de execução muito longos para entradas maiores, tornando o algoritmo inviável em muitos casos. Foram utilizados esses valores na fórmula pois a função no f(1) =0, então foi usado o f(2)=4 e a f(57)= 5.995.

## Algoritmo 5:

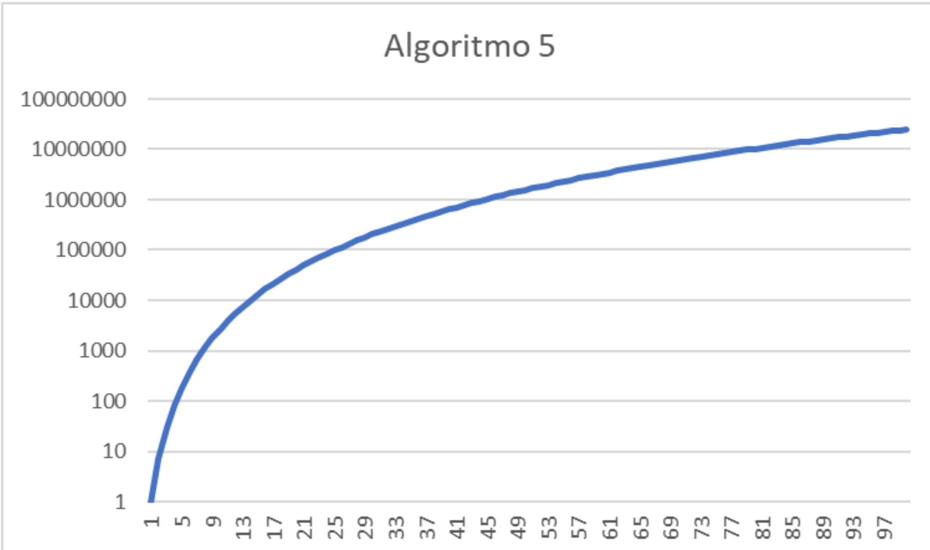
```
public class Main5 {
        public static void main (String[] args){
                for(int i = 1 ; i <= 100; i++){
                System.out.println(i + "\t" + f(i));
                }
        }

        public static int f( int n ) {
                int cont_op = 0;
                int i, j, k, res = 0;
                for( i = 1; i <= n*n; i += 1 )
                        for( j = 1; j <= i; j += 2 )
                        for( k = n+1; k <= 2*i; k += i*j ) {
                                res = res + k+1;
                                cont_op++;
                        }
                return cont_op;
        }
}
```

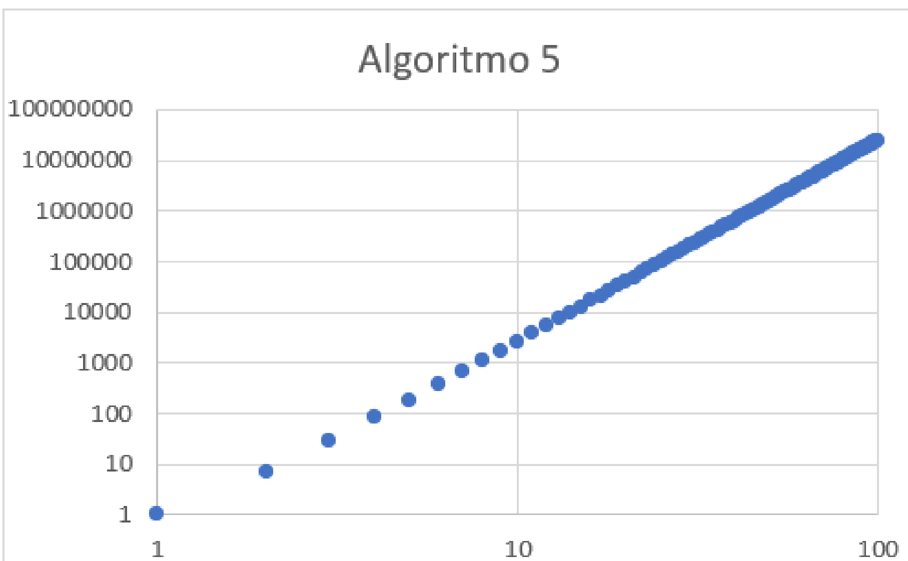
- Desenvolvimento:

A planilha do Excel recebeu os dados resultantes da execução do código acima e foi gerado o gráfico a seguir:



O resultado apresentado abaixo foi obtido após a aplicação da escala logarítmica na base 10 no eixo y e depois no eixo x, igual realizado no algoritmo 1,2, 3 e 4:




E para obter a função e em qual grau ela cresce utilizei a fórmula:

b = (log(25014250) – log(1)) / (log(100) – log(1))

b = log(25014250/1) / log(100/1)

25014250/1 = 25014250

100/1 = 100

b = log(25014250) / log(100)

b = 7.3983 / 2

b = 3.69915

Portanto, o valor da função é n elevado a 3,7, sendo assim ela é uma função cúbica, ele tem uma complexidade mais elevada, e em problemas com grandes conjuntos de dados, o tempo de execução do algoritmo elevado ao cubo pode se tornar inviável, levando a longos tempos de espera e possivelmente tornando o código inutilizável. Foram utilizados esses valores na fórmula pois a função no 1 é igual a 1 e no 100 é igual a 25.014.250 (f(1)= 1 e no f(100)= 25.014,250).