**Pontifícia Universidade Católica do Rio Grande do Sul**

**Algoritmos e Estruturas de Dados I**

**Engenharia de Software**

Mateus Campos Caçabuena

Relatório do Trabalho 1

Experimentos em Complexidade

Porto Alegre

2023

Algoritmo 1

// Código implementado para medir operações: ALGORITMOS 1, 3, 4 E 5 IGUAIS

public class exercicio1

{

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, 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) {

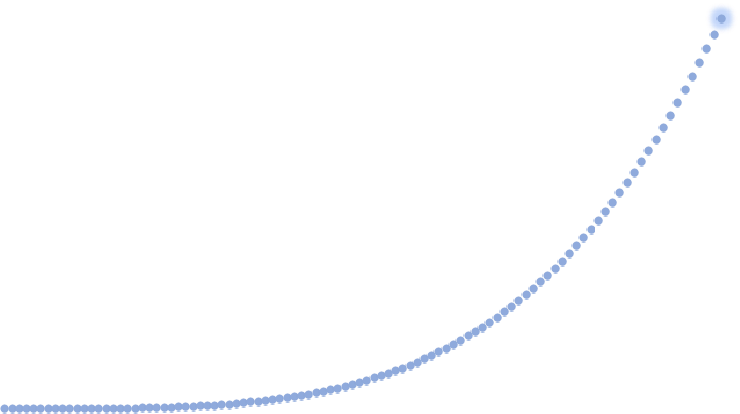
res = res + n-1; cont\_op++;

}

return cont\_op;

}

}



**Plotagem do número de chamadas da função**

8000000

7000000

6000000

5000000

4000000

3000000

2000000

1000000

0

0

20

40

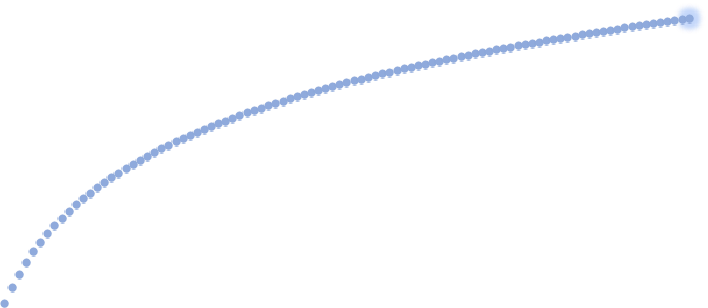
60

80

100

120

Agora, será feita a extração do logaritmo destes números para saber se a função é exponencial ou polinomial:



**Plotagem do logaritmo do número de**

**chamadas.**

10000000

1000000

100000

10000

1000

100

10

1

0

20

40

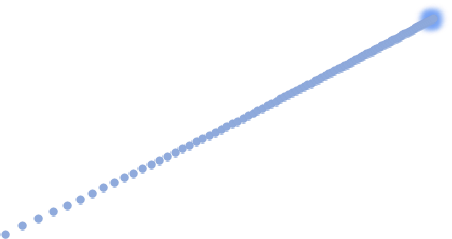
60

80

100

120

Como o gráfico apresenta uma curva logarítmica, então a próxima possibilidade é de uma função polinomial. A partir disto, é necessário “espremermos” também o eixo *x*, a fim de saber se trata de um polinômio:



**Plotagem do logaritmo do número de**

**operações.**

10000000

1000000

100000

10000

1000

100

10

1

1

10

100

Percebe-se que o gráfico apresentou uma quase reta, portanto, trata-se de uma função polinomial. A conta para descobrir o *b* e consequentemente o expoente de uma função polinomial é:

*f*(1) = 6

*f*(100) =6809401

*b* ≈ (*log*(6809401) − *log*(6)) / (*log*(100) − *log*(1)) = 3.0274788299

# Algoritmo 2

// Código implementado para medir operações:

public static void main(String[] args) throws Exception {

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 = 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;

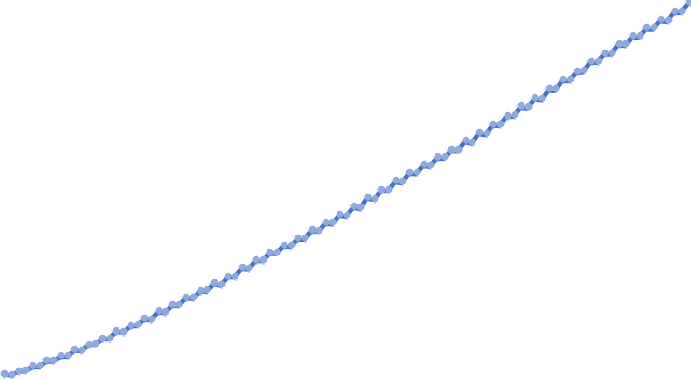
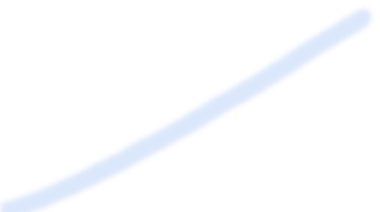
cont\_op++;

}

return cont\_op;

}

}



**Plotagem do número de chamadas da função.**

700

600

500

400

300

200

100

0

0

20

40

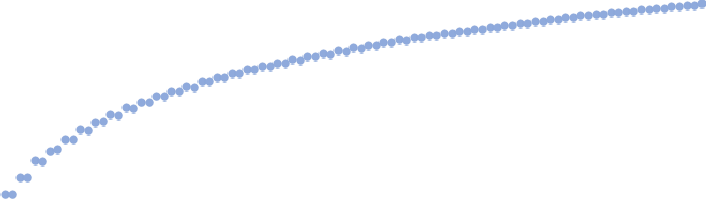
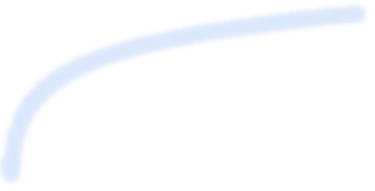
60

80

100

120

Agora, será feita a extração do logaritmo destes números para saber se a função é exponencial ou polinomial:



**Plotagem do logaritmo do número de**

**chamadas.**

1000

100

10

1

0

20

40

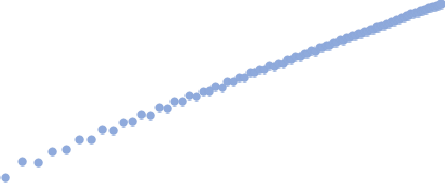
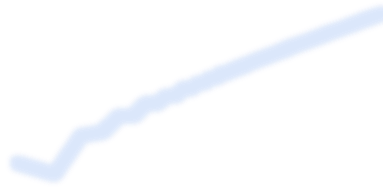
60

80

100

120

Como o gráfico apresenta uma curva logarítmica, então a próxima possibilidade é de uma função polinomial. A partir disto, é necessário “espremermos” também o eixo *x*, a fim de saber se trata de um polinômio:



**Plotagem do logaritmo do número de**

**chamadas.**

1000

100

10

1

1

10

100

Percebe-se que o gráfico apresentou uma quase reta, portanto, trata-se de uma função polinomial. A conta para descobrir o b e consequentemente o expoente de uma função polinomial é:

*f*(2) = 3

*f*(100) = 626

*b* ≈ (*log*(626) − *log*(3)) / (*log*(100) − *log*(2)) = 1.36521131778

# Algoritmo 3

// Código implementado para medir operações:

public class exercicio3{ public static void main (){

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, cont\_op = 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 + (j-i);

cont\_op++;

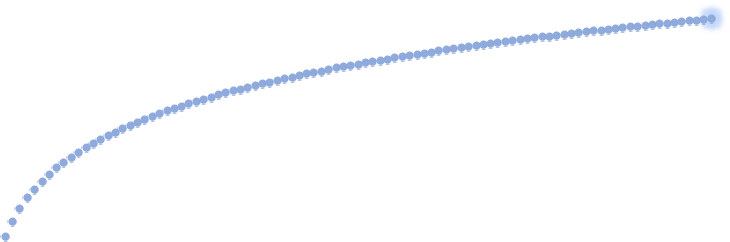
}

return cont\_op;

}

}

Agora, será feita a extração do logaritmo destes números para saber se a função é exponencial ou polinomial:



**Plotagem do logaritmo do número de**

**operações.**

100000

10000

1000

100

10

1

0

20

40

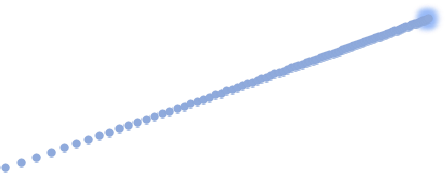
60

80

100

120

Como o gráfico apresenta uma curva logarítmica, então a próxima possibilidade é de uma função polinomial. A partir disto, é necessário “espremermos” também o eixo *x*, a fim de saber se trata de um polinômio:



**Plotagem do logaritmo do número de**

**operações.**

100000

10000

1000

100

10

1

1

10

100

Percebe-se que o gráfico apresentou uma quase reta, portanto, trata-se de uma função polinomial. A conta para descobrir o *b* e consequentemente o expoente de uma função polinomial é:

*f*(1) = 3

*f*(100) = 15662

*b* ≈ (*log*(15662) − *log*(3)) / (*log*(100) − *log*(1)) = 1.85886298246

# Algoritmo 4

// Código implementado para medir operações:

public class exercicio4{ public static void main (){

for (int i = 1; i <= 50; i++){ System.out.println(i + "\t" + f(i));

}

}

public static int f( int n ) {

int i, j, k, res = 0, cont\_op = 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 ) {

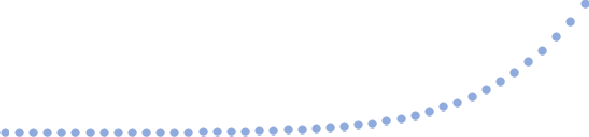
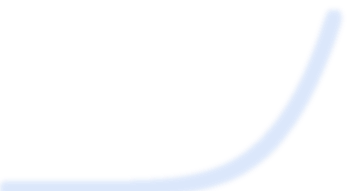
res = res + 1; cont\_op++;

}

return cont\_op;

}

}



**Plotagem do número de chamadas da função.**

1.2E+09

1E+09

800000000

600000000

400000000

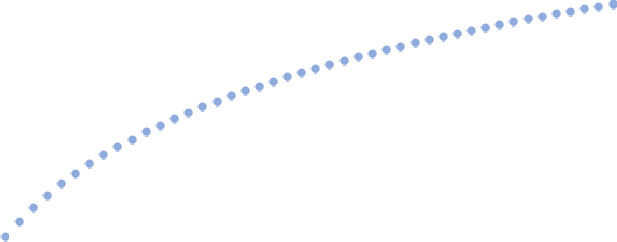
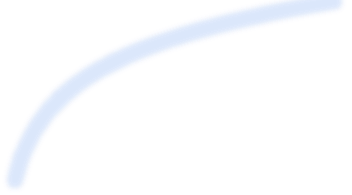
200000000

0

0 10 20 30 40 50

60

Agora, será feita a extração do logaritmo destes números para saber se a função é exponencial ou polinomial:



**Plotagem do logaritmo do número de**

**chamadas.**

1E+09

100000000

10000000

1000000

100000

10000

1000

100

10

1

0

10

20

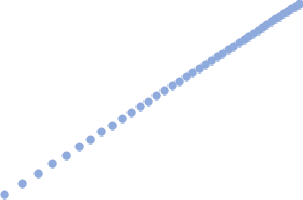
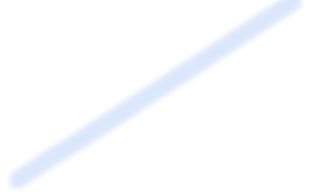
30

40

50

60

Como o gráfico apresenta uma curva logarítmica, então a próxima possibilidade é de uma função polinomial. A partir disto, é necessário “espremermos” também o eixo *x*, a fim de saber se trata de um polinômio:



**Plotagem do logaritmo do número de**

**chamadas.**

1E+09

100000000

10000000

1000000

100000

10000

1000

100

10

1

1

10

100

Percebe-se que o gráfico apresentou uma quase reta, portanto, trata-se de uma função polinomial. A conta para descobrir o *b* e consequentemente o expoente de uma função polinomial é:

*f*(1) = 0

*f*(50) =6809401

*b* ≈ *log*(958180300) / (*log*(50) − *log*(1)) = 5.28640718475

***f(n) ≈ n 5,28***

# Algoritmo 5

// Código implementado para medir operações:

public class exercicio5{ public static void main (){

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, cont\_op = 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;

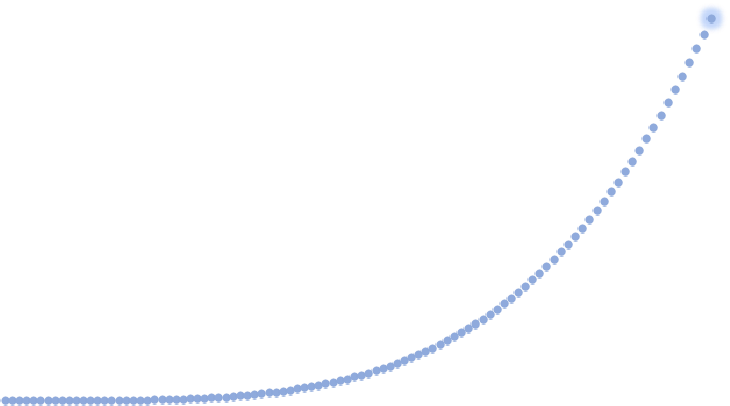
cont\_op++;

}

return cont\_op;

}

}



**Plotagem do número de chamadas da função.**

30000000

25000000

20000000

15000000

10000000

5000000

0

0

20

40

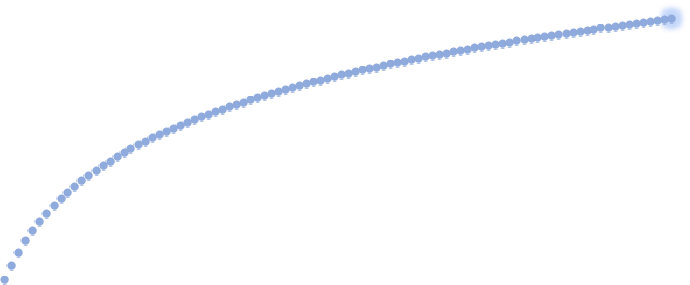
60

80

100

120

Agora, será feita a extração do logaritmo destes números para saber se a função é exponencial ou polinomial:



**Plotagem do logaritmo do número de**

**operações.**

100000000

10000000

1000000

100000

10000

1000

100

10

1

0

20

40

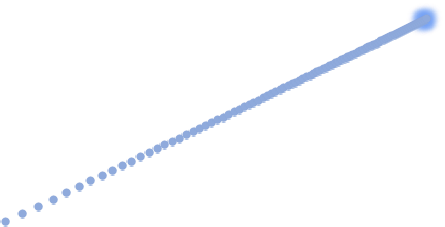
60

80

100

120

Como o gráfico apresenta uma curva logarítmica, então a próxima possibilidade é de uma função polinomial. A partir disto, é necessário “espremermos” também o eixo *x*, a fim de saber se trata de um polinômio:



**Plotagem do logaritmo do número de**

**operações.**

100000000

10000000

1000000

100000

10000

1000

100

10

1

1

10

100

Percebe-se que o gráfico apresentou uma quase reta, portanto, trata-se de uma função polinomial. A conta para descobrir o *b* e consequentemente o expoente de uma função polinomial é:

*f*(1) = 1

*f(*100) = 25014250

*b* ≈ (*log*(25014250) − *log*(1)) / (*log*(100) − *log*(1)) = 3.699093743

**Conclusão**

Após estudar e analisar todos os 6 algoritmos, percebe-se como uma simples alteração no código pode mudar drasticamente o custo, como é mostrado nos gráficos. O algoritmo que possui o menor custo de operações é o 2º, pois possui o menor número no expoente comparado às outras funções polinomiais. Enquanto isso, o algoritmo que possui o maior custo de operações é o 6º, pois cresce exponencialmente e, como foi visto no gráfico do algoritmo, a partir da 25a contagem ele começa a consumir operações de forma mais rápida que qualquer outro algoritmo anterior.

Voltando a citar o fato de pequenas alterações gerarem grandes diferenças, deve ser destacado a diferença gritante que há entre a quantia de operações do 2o e 3o algoritmo. Não, o 3o algoritmo não chega nem perto de ter a função que mais consome operações deste relatório, nem mesmo citando apenas as funções polinomiais. Mas possui a menor diferença no numeral do expoente do algoritmo que menos consome, já mostrando uma diferença enorme. Em 100 execuções, um consumiu 626 operações contra 15662 operações do outro e, detalhe: o expoente do 3º é “apenas” 0,49 maior que o expoente do 2º.