PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática
Exercícios sobre Contagem de Operações e Medição do Tempo de Execução de Algoritmos
Curso: Engenharia de Software
Disciplina: Algoritmos e Estruturas de Dados II
Professores: Eveline Alonso Veloso e João Caram
Aluno: Arthur Curi Kramberger(729488)

ALGORITMO A:

Algoritmo A

```java
public static int codigoA(int n){
      int b=0;
      for(int i = 0; i <= n; i+=2){
            b += 3;
      }
      return b;
}
```

1. **Código feito em *Java* para contagem de operações e tempo médio de execução**

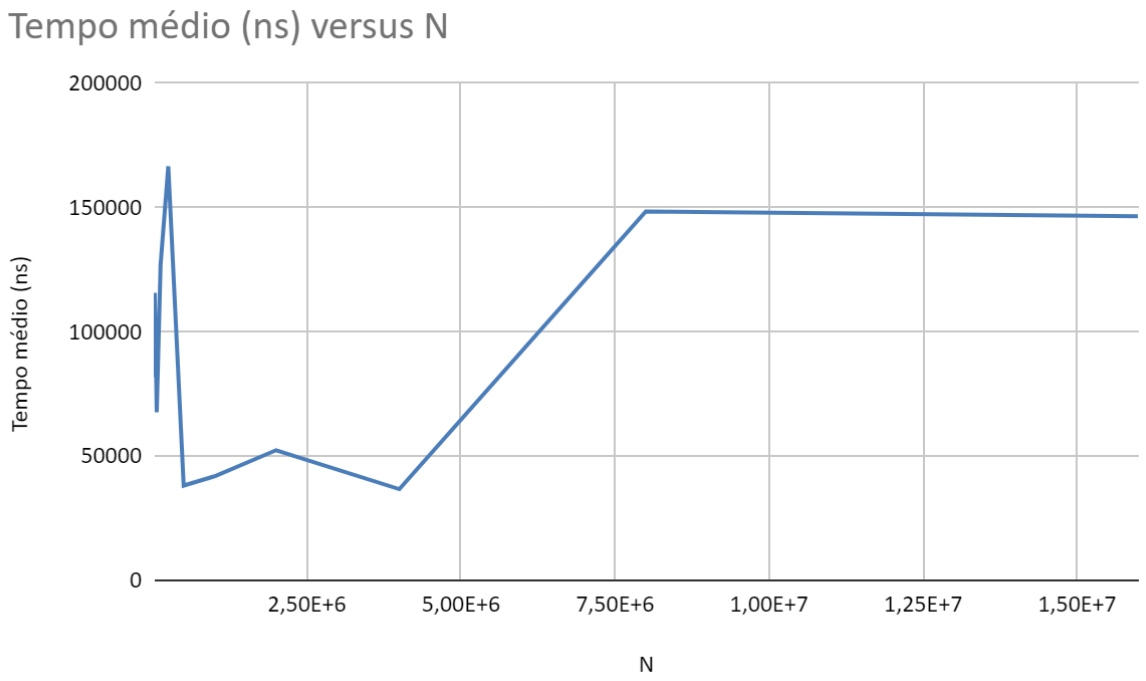
```java
import java.util.Arrays;

public class AppA {
    public static void main(String[] args) throws Exception {
        int n = 15625;
        final long LIM_TEMPO_NANOSEGUNDOS = 50_000_000_000L;

        for (int i = n; i <= 2000000000; i *= 2) {
            long[] tempos = new long[5];
            //as operacoes nao precisariam ser armazenadas em vetor,
pois as mesmas nao variam a resposta.
            //mantive assim apenas para realizar um "teste" que esta
comentado, porem o mesmo nao afeta o programa.
            long[] operacoes = new long[5];

            for (int j = 0; j < 5; j++) {

                long inicio = System.nanoTime();
                int numOp = codigoA(i);
                long fim = System.nanoTime();

                tempos[j] = fim - inicio;
                operacoes[j] = numOp;
```

```
            }
            Arrays.sort(tempos);
            long tempoMedio = (tempos[1] + tempos[2] + tempos[3]) / 3;
            //System.out.println("Tempo 1: " + tempos[1] + ", Tempo 2:
" + tempos[2] + ", Tempo 3: " + tempos[3]);
            //System.out.println("Operacoes 1: " + operacoes[1] + ",
Operacoes 2: " + operacoes[2] + ", Operacoes 3: " + operacoes[3]);
            System.out.println("N: " + i + ", Operações: " +
operacoes[0] + ", Tempo médio (ns): " + tempoMedio);

            if (tempoMedio > LIM_TEMPO_NANOSEGUNDOS) {
                System.out.println("Ultrapassou o tempo medio de 50s");
                break;
            }
        }
    }

    public static int codigoA(int n) {
        int operacoesA = 0;
        int b = 0;
        for (int i = 0; i <= n; i += 2) {
            operacoesA++;
            b += 3;
        }
        //retorno da contagem não da variavel b
        return operacoesA;
    }
}
```

2. Análise dos Resultados

N	Operações	Tempo médio (ns)
15625	7813	81800
31250	15626	115700
62500	31251	67666
125000	62501	127100
250000	125001	166600
500000	250001	38133
1000000	500001	41866
2000000	1000001	52366
4000000	2000001	36700
8000000	4000001	148466

16000000	8000001	146500
32000000	16000001	292933
64000000	32000001	591833
128000000	64000001	1171466
256000000	128000001	2442366
512000000	256000001	4685600
1024000000	512000001	9419966



3. Conclusão

Nessa tabela*(CÓDIGO A)* , dá pra ver que o tempo de execução tende a aumentar de forma linear conforme o valor de N cresce, com algumas flutuações que diminuem conforme N aumenta.

ALGORITMO B:

Algoritmo B

```
public static void codigoB(int[] vet){
      for (int i = 0; i < vet.length; i+=2){
            for(int j = i; j < (vet.length/2); j++){
                  vet[i] += vet[j];
            }
      }
}
```

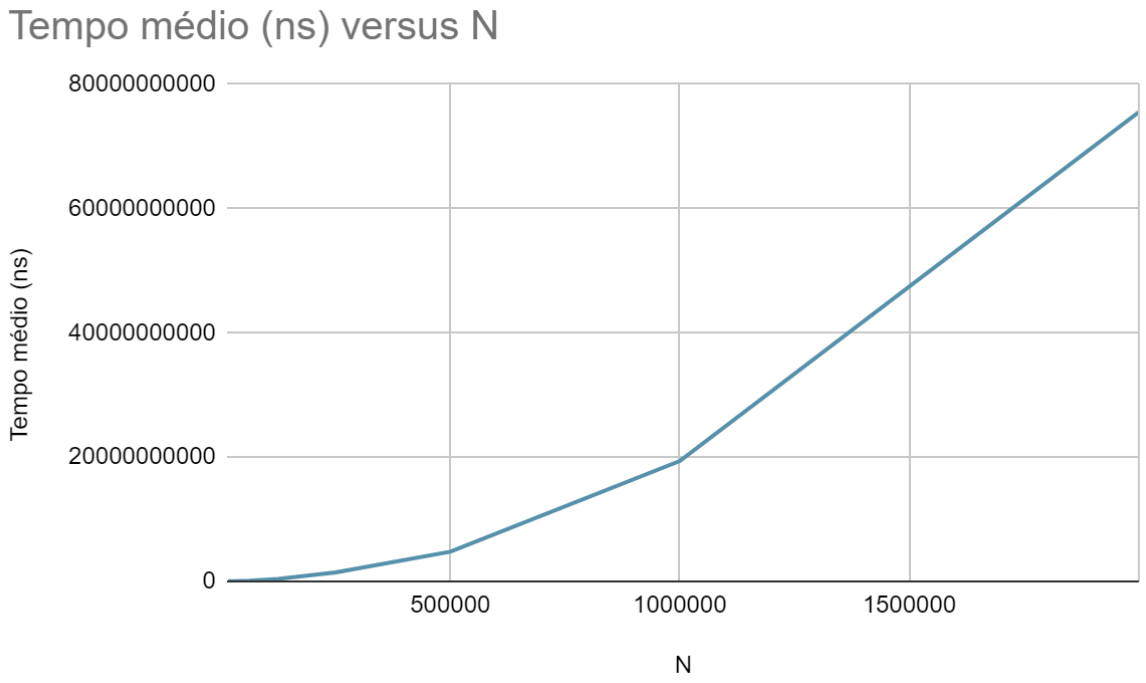
1. **Código feito em *Java* para contagem de operações e tempo médio de execução**

```java
import java.util.Arrays;

public class AppB {
    public static void main(String[] args) throws Exception {
        int n = 15625;
        final long LIM_TEMPO_NANOSEGUNDOS = 50_000_000_000L;

        for (int i = n; i <= 2000000000; i *= 2) {
            long[] tempos = new long[5];
            //as operacoes nao precisariam ser armazenadas em vetor, pois as mesmas nao variam a resposta.
            //mantive assim apenas para realizar um "teste" que esta comentado, porem o mesmo nao afeta o programa.
            long[] operacoes = new long[5];
            int[] vet = new int[i];

            for (int j = 0; j < 5; j++) {

                long inicio = System.nanoTime();
                int numOp = codigoB(vet);
                long fim = System.nanoTime();

                tempos[j] = fim - inicio;
                operacoes[j] = numOp;


            }
            Arrays.sort(tempos);
            long tempoMedio = (tempos[1] + tempos[2] + tempos[3]) / 3;
            //System.out.println("Tempo 1: " + tempos[1] + ", Tempo 2: " + tempos[2] + ", Tempo 3: " + tempos[3]);
            //System.out.println("Operacoes 1: " + operacoes[1] + ", Operacoes 2: " + operacoes[2] + ", Operacoes 3: " + operacoes[3]);
            System.out.println("N: " + i + ", Operações: " + operacoes[0] + ", Tempo médio (ns): " + tempoMedio);

            if (tempoMedio > LIM_TEMPO_NANOSEGUNDOS) {
                System.out.println("Ultrapassou o tempo medio de 50s");
                break;
            }
        }
    }
```

```java
    //troquei o void por int para retornar a contagem de operacoes
    public static int codigoB(int[] vet) {
        int operacoesB = 0;
        for (int i = 0; i < vet.length; i += 2) {
            for (int j = i; j < (vet.length / 2); j++) {
                vet[i] += vet[j];
                operacoesB++;
            }
        }
        //retorno da contagem
        return operacoesB;
    }
}
```

2. Análise dos Resultados

N	Operações	Tempo médio (ns)
15625	15260742	6000333
31250	61042969	23655200
62500	244156250	96016466
125000	976593750	400635266
250000	-388654796	1442765266
500000	-1554744184	4774518300
1000000	-1924259440	19369745633
2000000	892396832	75501242033



3. Conclusão

Nessa tabela*(CÓDIGO B)*. O número de operações começa a mostrar valores negativos, o que não deveria acontecer. Isso indica que provavelmente houve um overflow. Além disso, o tempo de execução cresce muito rápido chegando a encerrar o programa após passar do tempo limite estipulado.