N584 – Projeto e Análise de Algoritmos
Prof. Napoleão Nepomuceno

Atividade Computacional I

Exercício 1

Passo 1: Implementar o seguinte código em Java ou equivalente em outra linguagem de programação.

```
import java.util.Scanner;
public class Linear {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		System.out.print("Informe o valor de n: ");
		long n = scanner.nextLong();
		scanner.close();
		System.out.printf("%20s%30s%30s\n","n", "solucao", "tempo");
		double inicio = System.currentTimeMillis();
		double soma = 0;
		for (long i = 1; i <= n; i++) {
			soma = soma + i;
		}
		double fim = System.currentTimeMillis();
		double tempo = fim - inicio;
		System.out.printf("%20d%30.0f%30.0f\n", n, soma, tempo);
	}
}
```

Passo 2: Executar o código 3 vezes para cada valor de n = 1.000, 1.000.000, 1.000.000.000. Para cada execução, preencher o resultado na planilha disponibilizada.

Passo 3: Explicar porque o tempo de execução cresce à medida que o valor de n cresce.

Passo 4: Explicar porque o tempo de execução varia mesmo para valores de n iguais.

Passo 5: Tente executar o código para o valor de n = 1.000.000.000.000. O que acontece?

Passo 6: Implementar o seguinte código em Java ou equivalente em outra linguagem de programação.

```
import java.util.Scanner;
public class Constante {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		System.out.print("Informe o valor de n: ");
		long n = scanner.nextLong();
		scanner.close();
		System.out.printf("%20s%30s%30s\n","n", "solucao", "tempo");
		double inicio = System.currentTimeMillis();
		double soma = n;
		soma = soma * (n+1) / 2;
		double fim = System.currentTimeMillis();
		double tempo = fim - inicio;
		System.out.printf("%20d%30.0f%30.0f\n", n, soma, tempo);
	}
}
```

Passo 7: Executar o código para n = 1.000, 1.000.000, 1.000.000.000, 1.000.000.000.000. Para cada execução, preencher o resultado na planilha disponibilizada.

Passo 8: Explicar porque o tempo de execução não cresce mesmo quando o valor de n cresce.

Exercício 2

Passo 1: Implementar o seguinte código em Java ou equivalente em outra linguagem de programação e preencher o resultado na planilha disponibilizada.

```
public class Cogigo1 {
	public static void main(String[] args) {
		System.out.printf("%20s%30s\n","n", "tempo");
		for (int inst = 1; inst <= 50; inst++) {
			long n = 1000000 * inst;
			double inicio = System.currentTimeMillis();
			double soma = n;
			soma = soma * (n + 1) / 2;
			double fim = System.currentTimeMillis();
			double tempo = fim - inicio;
			System.out.printf("%20d%30.0f\n", n, tempo);
		}
	}
}
```

Passo 2: Implementar o seguinte código em Java ou equivalente em outra linguagem de programação e preencher o resultado na planilha disponibilizada.

```
public class Codigo2 {
	public static void main(String[] args) {
		System.out.printf("%20s%30s\n","n", "tempo");
		for (int inst = 1; inst <= 50; inst++) {
			long n = 1000000 * inst;
			double inicio = System.currentTimeMillis();
			double soma = 0;
			for (long i = 1; i <= n; i++) {
				soma = soma + i;
			}
			double fim = System.currentTimeMillis();
			double tempo = fim - inicio;
			System.out.printf("%20d%30.0f\n", n, tempo);
		}
	}
}
```

Passo 3: Implementar o seguinte código em Java ou equivalente em outra linguagem de programação e preencher o resultado na planilha disponibilizada.

```
public class Codigo3 {
	public static void main(String[] args) {
		System.out.printf("%20s%30s\n","n", "tempo");
		for (int inst = 1; inst <= 50; inst++) {
			long n = 1000 * inst;
			double inicio = System.currentTimeMillis();
			double soma;
			double total = 0;
			for (int j = 1; j <= n; j++) {
				soma = 0;
				for (long i = 1; i <= n; i++) {
					soma = soma + i;
				}
				total = total + soma;
			}
			soma = total / n;
			double fim = System.currentTimeMillis();
			double tempo = fim - inicio;
			System.out.printf("%20d%30.0f\n", n, tempo);
		}
	}
}
```

Passo 4: Descrever o comportamento de cada um dos gráficos obtidos. Verifique a relação entre as curvas obtidas e a quantidade de *loop for* aninhados de cada algoritmo.

Exercício 3

Passo 1: Implementar o seguinte código em Java ou equivalente em outra linguagem de programação.

```
import java.util.Scanner;
import java.util.Random;

public class InsertionSort {
	public static void main(String[] args) {
		int[] A;
		double inicio, fim, tempo;
		Scanner scanner = new Scanner(System.in);
		System.out.print("Informe o valor de n: ");
		int n = scanner.nextInt();
		scanner.close();
		System.out.printf("%20s%30s\n","n", "tempo");

		A = CriaVetorCrescente(n);
		inicio = System.currentTimeMillis();
		A = Ordenacao(A);
		fim = System.currentTimeMillis();
		tempo = fim - inicio;
		System.out.printf("%20d%30.0f\n", n, tempo);

		A = CriaVetorAleatorio(n);
		inicio = System.currentTimeMillis();
		A = Ordenacao(A);
		fim = System.currentTimeMillis();
		tempo = fim - inicio;
		System.out.printf("%20d%30.0f\n", n, tempo);

		A = CriaVetorDecrescente(n);
		inicio = System.currentTimeMillis();
		A = Ordenacao(A);
		fim = System.currentTimeMillis();
		tempo = fim - inicio;
		System.out.printf("%20d%30.0f\n", n, tempo);
	}

	static int[] CriaVetorCrescente (int n) {
		Random randomGenerator = new Random();
		int[] A = new int[n];
		for (int i = 1; i < n; i++) {
			A[i] = A[i-1] + randomGenerator.nextInt(10);
		}
		return A;
	}

	static int[] CriaVetorDecrescente (int n) {
		Random randomGenerator = new Random();
		int[] A = new int[n];
		A[0] = Integer.MAX_VALUE;
		for (int i = 1; i < n; i++) {
			A[i]= A[i-1] - randomGenerator.nextInt(10);
		}
		return A;
	}

	static int[] CriaVetorAleatorio (int n) {
		Random randomGenerator = new Random();
		int[] A = new int[n];
		for (int i = 0; i < n; i++) {
			A[i] = randomGenerator.nextInt(Integer.MAX_VALUE);
		}
		return A;
	}

	static void ImprimeVetor (int[] A) {
		for (int i = 0; i < A.length; i++) {
			System.out.println(A[i]);
		}
	}

	static int[] Ordenacao(int[] A) {
		for (int j = 1; j < A.length; j++) {
			int chave = A[j];
			int i = j - 1;
			while (i >= 0 && A[i] > chave) {
				A[i+1] = A[i];
				i--;
			}
			A[i+1] = chave;
		}
		return A;
	}

}
```

Passo 2: Executar o código para $n = 100.000$.

Passo 3: Explicar porque o tempo de execução é tão diferente, mesmo sabendo que os vetores são ordenados pelo mesmo método e que possuem o mesmo tamanho.