ECM-306 -Tópicos Avançados em Estrutura de Dados
Prof. Dr. Aparecido V. de Freitas – Atividade 3

1. Utilizando o **Modelo Simplificado de Knuth**, determine a quantidade de operações executadas na **linha 1** do algoritmo abaixo:

```
package maua;
import java.util.Scanner;
public class Atividade3_1 {
	public static void main(String[] args) {
		Scanner in = new Scanner (System.in);
		int n = in.nextInt();
		System.out.println(Func(n));
		in.close();
	}
	public static int Func(int n) {
		int i = 4;
		int m = 0;
		while (i <= n) {
			m = m + 1;		// Linha 1
			i  = i + 2;
		}
		return m;
	}
}
```

2. Utilizando o **Modelo Simplificado de Knuth**, determine a quantidade de operações executadas na **linha 1** do algoritmo abaixo:

```
package maua;
import java.util.Scanner;
public class Atividade3_2 {
	public static void main(String[] args) {
		Scanner in = new Scanner (System.in);
		int n = in.nextInt();
		System.out.println(Func(n));
		in.close();
	}
	public static int Func(int n) {
		int i = 1;
		int m = 0;
		while (i <= n) {
			m = m + 1;		// Linha 1
			i  = i * 2;
		}
		return m;
	}
}
```

ECM-306 -Tópicos Avançados em Estrutura de Dados
Prof. Dr. Aparecido V. de Freitas – Atividade 3

3. Utilizando o **Modelo Simplificado de Knuth**, determine a quantidade de operações executadas na **linha 1** do algoritmo abaixo:

```
package maua;
import java.util.Scanner;

public class Atividade3_3 {
	public static void main(String[] args) {
		Scanner in = new Scanner (System.in);
		int n = in.nextInt();
		System.out.println(Func(n));
		in.close();
	}
	public static int Func(int n) {
		int m = 0;
		for (int i=1; i <= n; i++)
			for (int j = 1; j <= n; j++ ) {
				m = m + 1;  // Linha 1
		}
		return m;
	}
}
```

4. Utilizando o **Modelo Simplificado de Knuth**, determine a quantidade de operações executadas na **linha 1** do algoritmo abaixo:

```
package maua;
import java.util.Scanner;

public class Atividade3_4 {
	public static void main(String[] args) {
		Scanner in = new Scanner (System.in);
		int n = in.nextInt();
		System.out.println(Func(n));
		in.close();
	}
	public static int Func(int n) {
		int m = 0;
		for (int i=2; i < n; i++)
			for (int j = 2; j < n; j++ ) {
				m = m + 1;  // Linha 1
		}
		return m;
	}
}
```

ECM-306 -Tópicos Avançados em Estrutura de Dados
Prof. Dr. Aparecido V. de Freitas – Atividade 3

5. Utilizando o **Modelo Simplificado de Knuth**, determine a quantidade de operações executadas na **linha 1** do algoritmo abaixo:

```
package maua;
import java.util.Scanner;

public class Atividade3_5 {
	public static void main(String[] args) {
		Scanner in = new Scanner (System.in);
		int n = in.nextInt();
		System.out.println(Func(n));
		in.close();
	}
	public static int Func(int n) {
		int m = 0;
		for (int i=1; i <= n; i++)
			for (int j = i; j <= n; j++ ) {
				m = m + 1;  // Linha 1
		}
		return m;
	}
}
```

6. Vamos supor que estamos comparando implementações de ordenação por inserção e ordenação por intercalação na mesma máquina. Para entradas de tamanho n, a ordenação por inserção é executada $8n^2$ etapas, enquanto a ordenação por intercalação é executada em $64n \ln n$ etapas. Para que valores de n a ordenação por inserção supera a ordenação por intercalação?

7. Qual é o menor valor de n tal que um algoritmo cujo tempo de execução é $100n^2$ funciona mais rápido que um algoritmo cujo tempo de execução é 2^n na mesma máquina?

8. Considere dois algoritmos **A** e **B** com complexidades respectivamente iguais a $\mathbf{128n^2}$ e $\mathbf{4n^3}$. Qual o maior valor de **n**, para o qual o algoritmo **B** é mais eficiente que o algoritmo **A**?

9. Considere dois computadores **C1** e **C2** que executam $\mathbf{10^8}$ e $\mathbf{10^{10}}$ operações por segundo e dois algoritmos de ordenação **A** e **B** que necessitam $\mathbf{5n^2}$ e $\mathbf{40n\log_{10}n}$ operações com entrada de tamanho **n**, respectivamente. Qual o tempo de execução de cada algoritmo em cada um dos computadores **C1** e **C2** para ordenar $\mathbf{10^8}$ elementos?

10. Um algoritmo tem complexidade $\mathbf{2^n}$. Num certo computador, num tempo t, o algoritmo resolve um problema de tamanho 25. Imagine agora que se tenha disponível um computador **100** vezes mais rápido. Qual o tamanho máximo de problema que o mesmo algoritmo resolve no mesmo tempo t no computador mais rápido ?