Lista 3 - Complexidade

ESTRUTURA DE DADOS I – Pedro Nuno Moura

1. Diga a ordem de complexidade de cada um dos trechos de código a seguir:

a.

```
int a = 0, b = 0;
for (i = 0; i < N; i++) {
    a = a + rand();
}
for (j = 0; j < M; j++) {
    b = b + rand();
}
```

b.

```
int a = 0;
for (i = 0; i < N; i++) {
    for (j = N; j > i; j--) {
        a = a + i + j;
    }
}
```

c.

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
    for (j = 2; j <= n; j = j * 2) {
        k = k + n / 2;
    }
}
```

2. Assuma que cada uma das expressões abaixo modele o tempo de processamento T(n) gasto por um algoritmo para resolver um problema de tamanho n. Diga o tempo dominante e a menor complexidade Big-Oh em cada equação.

	Expressão	Termo dominante	$\boldsymbol{O}()$
a	$5 + 0.001n^3 + 0.025n$		
b	$500n + 100n^{1.5} + 50n\log_{10} n$		
c	$0.3n + 5n^{1.5} + 2.5\ n^{1.75}$		
d	$n^2 \log_2 n + n(\log_2 n)^2$		
e	$n \log_3 n + n \log_2 n$		
f	$3 \log_8 n + \log_2 \log_2 \log_2 n$		
g	$100n + 0.01n^2$		
h	$0.01n + 100n^2$		
i	$2n + n^{0.5} + 0.5n^{1.25}$		
j	$0.01n \log_2 n + n(\log_2 n)^2$		
k	$100n \log_3 n + n^3 + 100n$		
l	$0.003 \log_4 n + \log_2 \log_2 n$		

3. A seguir estão três implementações com lógicas diferentes de um método que percorre um vetor até achar o valor desejado. Diga qual a complexidade de cada método e explique cada uma.

   i)

```
private int retornaDesejo(int[] vetor, int desejado){
        int contador = 0;
        while(contador <= vetor.length()-1){
                if(vetor[contador] == desejado){
                        return vetor[contador];
                }
                contador++;
        }
        return Integer.MIN_VALUE;
}
```

   ii)

```
private int retornaDesejo(int[] vetor, int desejado){
        for(int contador=0; contador <= vetor.length()-1; contador++){
                if(vetor[contador] == desejado){
                        return vetor[contador];
                }
        }
        return Integer.MIN_VALUE;
}
```

iii)

```
private int retornaDesejo(int[] vetor, int desejado){
        int low = vetor[0];
        int high = vetor.length();
        while (low <= high) {
            int mid = (low + high) / 2;
            if (vetor[mid] < desejado) {
                low = mid + 1;
            } else if (vetor[mid] > desejado) {
                high = mid - 1;
            } else if (vetor[mid] == desejado) {
                return vetor[mid];
            }
          }
        return Integer.MIN_VALUE;
}
```

4. Melhore a complexidade do método getPosicaoValor() da classe abaixo. Considere que o vetor está ordenado em ordem crescente.

```
public class Fila{
    private int[] fila;
    .
    .
    .
```

```
    public int getPosicaoValor(int x){
        for(int i = 0; i <= fila.length()-1; i++{
            if(fila[i] == x){  return  i; }
        }
        return Integer.MIN_VALUE;
    }
}
```

5. Implemente um método com a melhor complexidade possível, melhor tempo possível, para conseguir a quantidade de inteiros pares de uma pilha. Considere que a pilha em questão possui os métodos pop e push padrões e atributos padrões. Você possui liberdade para criar novos métodos e/ou atributos e alterar os métodos pop e push já existentes.