UTN SANTA FE

Diseño e Implementación

De Estructuras de Datos

Guía 09 – Complejidad

Contenido

Guía 08

Realizar un fork del proyecto https://github.com/died-utn/guia09.

Problema 01

Suponga que cada renglón de un arreglo A de ***NxN*** posee valores 1 y 0 de tal modo que en cualquier renglón de A todos los 1 van antes de que los 0.

Escriba un método que permita determinar el renglón de A que contiene la mayor cantidad de unos

- **Ejemplo: resultado i=2**

1	1	0	0	0
1	1	1	0	0
1	**1**	**1**	**1**	**0**
1	0	0	0	0
1	1	1	0	0

Escribir en java, dos soluciones, una en la clase Solucion1, y otra en la clase Solucion2 para este problema, una con complejidad O(n) y otra con complejidad mayor (por ejemplo $O(n^2)$).

Ejecutar pruebas y realizar un análisis empírico para verificar que los tiempos de ejecución crecen según la complejidad estimada.

Problema 02

Un arreglo A contiene n-1 enteros únicos en el intervalo [0, n-1]; esto es hay un número de este intervalo que no está en A.

- **Ejemplo**
  - Sea N= 5
  - El arreglo se puede completar con valores entre [0;(N-1)] → [0;4]
  - El arreglo tiene n-1 elementos. → 4

Valor	1	4	2	0
Indice	*0*	*1*	*2*	*3*

"El valor que falta es 3"

Escribir en java, dos soluciones, una en la clase Solucion1, y otra en la clase Solucion2 para este problema, una con complejidad O(n) y otra con complejidad mayor (por ejemplo ***O(n^2)***).

Ejecutar pruebas y realizar un análisis empírico

Problema 03

Dada una secuencia de enteros positivos y negativos encontrar cual es la subsecuencia de suma máxima y calcular el resultado de dicha suma

De esta manera debemos recorrer el arreglo de forma tal de encontrar un sub arreglo cuyo valor de la $\sum$ de Ai hasta Aj se máximo.

Ejemplo:

- [3,-4,-3,2,7,-4,3] -> {2,7} = 9
- [3,-4,-3,2,7,-1,3,-4] -> {2,7,-1,3} = 11

Escribir en java, dos soluciones, una en la clase Solucion1, y otra en la clase Solucion2 para este problema, una con complejidad O(n) y otra con complejidad mayor (por ejemplo ***O(n^2)***).

Ejecutar pruebas y realizar un análisis empírico

Problema 04

Determinar la complejidad de los siguientes métodos

```
public void m1(int n) {
	for(int i  =0;i<n;i++) {
		System.out.println(i);
	}
}
```

```
public void m2(int n) {
	for(int i  =0;i<n;i+=2) {
		System.out.println(i);
	}
}
```

```
public void m3(int n) {
	for(int i  =0;i<n*n;i++) {
		System.out.println(i);
	}
}
```

```
public void m4(int n) {
	for(int i  =0;i<n;i++) {
		for(int j  =0;j<i;j++) {
			System.out.println(i+ " _ "+ j);
		}
	}
}
```

```
public void m5(int n) {
	for(int i  =0;i<n*n;i++) {
		for(int j  =0;j<i;j++) {
			System.out.println(i+ " _ "+ j);
		}
	}
}
```

Problema 05

a. ¿Cuál es el propósito del siguiente algoritmo? Analizar el tiempo de ejecución en el peor caso y expresarlo en la notación "O".

```
public static long f1(int a, int n){
	long b = 1;
	for(int k = 0; k < n; k++){
		b = b * a;
	}
	return b;
}
```

b. ¿Cuál es el propósito del siguiente algoritmo? Analizar el tiempo de ejecución en el peor caso y expresarlo en la notación "O".

```
public static long f2(int a, int n){
	long b = 1;
	long c = a;
	for(int k = n; k > 0;){
		if(k%2 == 0) {
			k = k / 2;
			c = c * c;
		} else {
			k--;
			b = b * c;
		}
	}
	return b;
}
```