UNIVERSIDAD ICESI

Departamento de TIC

Computación y Estructuras Discretas I

Seguimiento sobre Análisis de Complejidad Temporal y Espacial

Objetivos

Unidad 3: Estructuras Discretas no recursivas y análisis de algoritmos

Calcular la complejidad temporal de algoritmos iterativos.
Calcular la complejidad espacial de algoritmos iterativos.

```
 2
 3⊕ import java.io.BufferedReader;
 8
 9  public class AlgoritmoX {
10      public static int resolver(int[] nums) {
11          int a = 0;
12          int b = 0;
13          for (int i=0;i<nums.length;i++) {
14              int c = nums[i];
15              int d = 0;
16              for (int j=i;j<nums.length;j++) {
17                  if(nums[j]==c) {
18                      d++;
19                  }
20              }
21              if(d>b) {
22                  b = d;
23                  a = c;
24              }
25          }
26          return a;
27      }
28
29      public static void main(String[] args)
30              throws IOException {
31          BufferedReader br = new BufferedReader(
32              new InputStreamReader(System.in));
33          BufferedWriter bw = new BufferedWriter(
34              new OutputStreamWriter(System.out));
35
36          String linea = br.readLine();
37          String[] vals = linea.split(" ");
38          int[] nums = new int[vals.length];
39          for (int i=0;i<vals.length;i++) {
40              nums[i] = Integer.parseInt(vals[i]);
41          }
42
43          int x = resolver(nums);
44          bw.write(x+"\n");
45
46          br.close();
47          bw.close();
48      }
49  }
```

Lleve a cabo el análisis de complejidad temporal y espacial del algoritmo **resolver**. Para la complejidad temporal dibuje líneas horizontales entre cada instrucción del algoritmo a lo ancho de la hoja y trace una línea vertical formando una columna a la derecha del código de forma que pueda escribir en cada celda de esa columna la cantidad de veces que se repite la línea. Para la complejidad espacial realice el inventario de las variables utilizadas (entrada, auxiliares y salida) **[70pts]**. Para la complejidad temporal sume los valores de la columna y obtenga una función del tiempo de ejecución en términos del **tamaño de la entrada** usando notación Big O. Sume ordenadamente los valores de la columna *# de veces que se repite la instrucción* para obtener una función con los valores tan precisos como sea posible. Para la espacial sume los resultados de su inventario obtenga una función del espacio en términos del **tamaño de la entrada** usando notación Big O **[30pts]**.

```java
Public class AlgoritmoX {
  public static int resolver(int[] nums) {
      int a = 0;                        // (1 vez)
      int b = 0;                        // (1 vez)
      for (int i=0; i<nums.length; i++) {   // (n veces)
          int c = nums[i];              // (n veces)
          int d = 0;                    // (n veces)
          for (int j=i; j<nums.length; j++) {  // (n-i veces en cada iteración de i)
              if (nums[j] == c) {       // (n-i veces en cada iteración de i)
                  d++;                  // (se ejecuta cuando la condición es verdadera)
              }
          }
          if (d > b) {                  // (n veces)
              b = d;                    // (se ejecuta cuando d > b)
              a = c;                    // (se ejecuta cuando d > b)
          }
      }
      return a;                         // (1 vez)
  }
}
```

Análisis de complejidad temporal

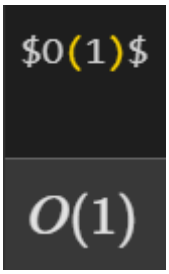
Instrucción	Costo	Veces que se repite
int a = 0;	O(1)	1 vez
int b = 0;	O(1)	1 vez
for (int i=0; i<nums.length; i++)	O(n)	n vez
int c = nums[i];	O(1)	n vez
int d = 0;	O(1)	n vez
for (int j=i; j<nums.length; j++)	O(n)	(n-i) veces por i
if (nums[j] == c)	O(1)	(n-i) veces por i
d++;	O(1)	cuando nums[j] == c
if (d > b)	O(1)	n veces
b = d;	O(1)	cuando d > b
a = c;	O(1)	cuando d > b
return a;	O(1)	1 vez

```
$n + (n-1) + (n-2) + \dots + 1 = \frac{n(n+1)}{2} = O(n^2)$
```

$$n + (n-1) + (n-2) + \cdots + 1 = \frac{n(n+1)}{2} = O(n^2)$$

Análisis de complejidad Espacial

Tipo	Variable	Tamaño de un valor atómico	Cantidad de valores atómicos
Entrada	nums	32 bits	n
Auxiliar	b	32 bits	1
Auxiliar	c	32 bits	1
Auxiliar	d	32 bits	1
Auxiliar	i	32 bits	1
Auxiliar	j	32 bits	1
Salida	a	32 bits	1



Análisis Completo

Variables	Auxiliares	Entradas	Salidas	Complejidad Temporal	Complejidad Espacial
nums	b, c, d, i, j	nums[ ]	a	$O(n^2)$	O(1)