

ITC-ADA-C1-2023: Assignment #1

Luis Ballado

luis.ballado@cinvestav.mx

CINVESTAV UNIDAD TAMAULIPAS — January 13, 2023

1 Diseñe un algoritmo para encontrar todos los elementos comunes en dos listas ordenadas de números



Info: Se hace la prueba con los siguientes valores

2, 5, 5, 5 y 2, 2, 3, 5, 5, 7

Pregunta 1

¿Cuál es el número máximo de comparaciones que realiza su algoritmo en función de las longitudes de las listas (m y n) respectivamente?

El número máximo de comparaciones viene dado por la búsqueda de los elementos del primer arreglo con el segundo, es decir aplicando la regla del producto $m \times n$, pero se puede reducir con los elementos repetidos ó con el manejo de un índice que nos indique la posición en que se quedó para retomarla en las proximas vueltas, pero de esa manera dejaríamos de fuera si el último elemento a buscar se encuentra en la primera posición

1.1 Pseudocodigo

Algorithm 1: Encontrar elementos iguales

entrada: $Arreglo_1, Arreglo_2$

salida : $Arreglo_3$ (arreglo con repetidos)

$CalcularLongitud(arreglo_1)$

$CalcularLongitud(arreglo_2)$

$lastIndex = 0$

for $i \leftarrow 0$ hasta $longitud(arreglo_1)$ **do**

for $j \leftarrow 0$ hasta $longitud(arreglo_2)$ **do**

if $arreglo_1[i] == arreglo_2[lastIndex]$ **then**

$arreglo_3.agregar(a[i])$

break

$lastIndex = i + 1$

1.2 Implementación

```
tarea1.cpp

1  #include <iostream>
2  #include <vector>
3
4  //Complejidad funcion principal O(n^2)
5  //por el doble for que recorre los arreglos
6  int main(){
7
8      std::vector<int> a;
9      a.push_back(2);
10     a.push_back(5);
11     a.push_back(5);
12     a.push_back(5);
13
14     std::vector<int> b;
15     b.push_back(2);
16     b.push_back(2);
17     b.push_back(3);
18     b.push_back(5);
19     b.push_back(5);
20     b.push_back(7);
21
22     //vector de resultados
23     std::vector<int> arr;
24
25     int last_index = 0;
26
27     //Usando fuerza bruta // O(n^2)
28     for (int i = 0; i < a.size(); i++){ // O(n)
29         for (int j = i; j < b.size(); j++){ // O(n)
30
31             //std::cout << a[i] << "<-a comparacion b->"<< b[
32             last_index] << "\n";
33
34             if (a[i] == b[last_index]){
35                 arr.push_back(a[i]);
36                 break; // romper ciclo cuando sean iguales
37             }
38
39             last_index = i+1; //indice auxiliar para avanzar
40         }
41     }
42
43     //-----
44     // Imprimir resultado
45     //-----
46
47     std::cout << "El resultado es: \n"; // 1
48     for(int i = 0; i<arr.size(); i++){ // n
49         std::cout << arr[i] << "\n";
50     }
51
52 }
```

ver código en github

Ejecutar desde una terminal

Command Line

```
$ g++ -o ./tarea1 ./tarea1.cpp
$ ./tarea1
```

1.3 Análisis de complejidad del mejor y peor caso



MEJOR CASO: encontrar todos los elementos del primer arreglo en el segundo arreglo de forma secuencial, sin el análisis de números que pertenezcan al primer arreglo ya que estarían ordenados $O(n)$



PEOR CASO: analizar todas las posibilidades y no encontrar similitudes $O(n^2)$