## 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 busqueda 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 dejariamos 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)

Calcular Longitud(arreglo_1)

Calcular Longitud(arreglo_2)

last Index = 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[last Index] then
| arreglo_3.agregar(a[i])
| break
| last Index = i + 1
```

#### 1.2 Implementación

```
tarea1.cpp
1 #include <iostream>
2 #include <vector>
4 //Complejidad funcion principal O(n^2)
5 //por el doble for que recorre los arreglos
6 int main(){
   std::vector<int> a;
   a.push_back(2);
   a.push_back(5);
   a.push_back(5);
   a.push_back(5);
   std::vector<int> b;
   b.push_back(2);
   b.push_back(2);
   b.push_back(3);
   b.push_back(5);
   b.push_back(5);
   b.push_back(7);
   //vector de resultados
   std::vector<int>arr;
   int last_index = 0;
                                           // O(n^2)
// O(n)
   //Usando fuerza bruta
   for (int i = 0; i < a.size(); i++){</pre>
     for (int j = i; j < b.size(); j++){</pre>
                                           // O(n)
       last_index] << "\n";</pre>
       if (a[i] == b[last_index]){
        arr.push_back(a[i]);
         break; // romper ciclo cuando sean iguales
       last_index = i+1; //indice auxiliar para avanzar
     }
   }
   // Imprimir resultado
   std::cout << "El resultado es: \n";</pre>
                                          // 1
   for(int i = 0; i < arr.size(); i++){</pre>
                                            // n
     std::cout << arr[i] << "\n";
```

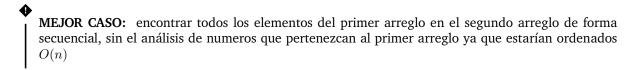
ver código en github

Ejecutar desde una terminal

```
Command Line

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

### 1.3 Análisis de complejidad del mejor y peor caso



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