ITC-ADA-C1-2023: Assignment #1

Luis Ballado
luis.ballado@cinvestav.mx

CINVESTAV UNIDAD TAMAULIPAS — January 12, 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 x n, pero se puede reducir con los elementos repetidos y reducir las comparaciones

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)
for i ← 0 hasta longitud(arreglo_1) do
    for j ← 0 hasta longitud(arreglo_2) do
        if arreglo_1[i] == arreglo_2[j] then
            if !Existe(arr, arreglo_1[i]) then
                arreglo_3.agregar(a[i])
        break;
```

Algorithm 2: Función Existe

```
input  : arreglo_3, valor
output: verdadero ó falso (booleano si existe el elemento ó no)

Function Existe(arreglo_3, valor)
    for i ← 0 hasta longitud(arreglo_3) do
        if arreglo_3[i] == valor then
            return Verdadero
    return Falso
```

1.2 Implementación

tarea1.cpp

```cpp
#include <iostream>
#include <vector>

bool existe(std::vector<int> _vector_, int valor){
  for(int i = 0; i < _vector_.size(); i++){ // n
    if (_vector_[i] == valor){              // 1
      return true;
    }
  }
  return false;
}

//Complejidad funcion principal O(n^2)
int main(){

  int a[] = {2,5,5,5};                      // 1
  int b[] = {2,2,3,5,5,7};                  // 1

  //vector de resultados
  std::vector<int>arr;                      // 1

  //longitud array1
  int a_length = sizeof(a)/sizeof(a[0]);    // 1
  //longitud array2
  int b_length = sizeof(b)/sizeof(b[0]);    // 1

  //Usando fuerza bruta                     // O(n^2)
  for (int i = 0; i < a_length; i++){
    for (int j = 0; j < b_length; j++){
      if (a[i] == b[j]){                    // 1
        //revisar si ya existe en el array
        if (!existe(arr,a[i])){             // 1
          arr.push_back(a[i]);
        }
        break;
      }
    }
  }

  // Imprimir resultado
  std::cout << "El resultado es: \n";       // 1
  for(int i = 0; i<arr.size(); i++){        // n
    std::cout << arr[i] << "\n";            // 1
  }
}
```

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 analisis de numeros que pertenezcan al primer arreglo

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