UNITED INTERNATIONAL UNIVERSITY
Department of Computer Science and Engineering (CSE)

Course Title: Data Structure & Algorithm Lab Lab II **Course Code: CSE2218**
Trimester & Year: Spring 2022 **Section: C** **Credit Hours: 1.0** **AZ**

ASSIGNMENT 02: Greedy Algorithm

Q1: Kruskal Algorithm Implementaion

Implement Kruskal's algorithm using the following pseudocode as discussed in the Class Lecture and show its simulation (Deposit the simulation as a separate pdf named *1.pdf*)

```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort the edges of G.E into nondecreasing order by weight w
5  for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```

Q2: MATCH THE STRINGS

Problem

You are given two binary strings S and T of the same length N. Your task is to make both the strings equal. Perform the following operation on string S:

Select any substring of S then flip all 1's to 0 and flip all 0's to 1 of that substring.

Print the minimum number of operations you have to perform to make them equal.

Input format

- The first line contains a single integer N (1≤N≤1e5) size of each string
- The second line contains a string S of length N.
- The third line contains a string T of length N.

Output format

Print a single integer denoting the minimum number of operations required to make both the strings equal.

Constraints

1≤N≤1e5

Sample Input	Sample Output
3 100 111	1

UNITED INTERNATIONAL UNIVERSITY
Department of Computer Science and Engineering (CSE)

Q3: FARIA'S PLAYTIME

14

Problem

Faria lives in the Purana Polton Colony. The colony has **N** houses numbered from 1 to **N**. There are **M** bidirectional roads in the colony for travelling between houses. There might be multiple roads between two houses.

Faria lives in the house with index 3. She has friends in all houses of the colony. She is always wanting to visit them to play. But her mom is really strict. She only allows her to go out for **K** units of time. This time includes the time taken to go to her friend's house, play with the friend and time taken to come back home.

You are given **Q** queries. In each query, Faria wants to go to his friend in house **A**, given **K** units of time. Help her find the maximum time that he can play. If **K** units of time is not sufficient to visit his friend and return back, Faria will not go and playing time will be zero.

Input:

- First line contains an integer **T**. **T** test cases follow.
- First line of each test case contains two space-separated integers **N**, **M**
- Next **M** lines contain three space-separated integers **X**, **Y** and **C**, denoting that there is Bidirectional road between house **X** and house **Y** with cost **C**.
- Next line contains an integer **Q**
- Following **Q** lines describe the queries. Each query contains two space-separated integers **A** and **K**.

Output
Print the answer to each query in a new line.

Sample Input	Sample Output
1 5 5 1 2 2 2 3 1 3 4 5 4 5 1 1 4 1 3 4 20 5 21 1 5	12 11 0

Constraints:
$1 \le \mathbf{T} \le 10$
$1 \le \mathbf{N}, \mathbf{Q} \le 10^4$
$1 \le \mathbf{M} \le 10^5$
$1 \le \mathbf{X}, \mathbf{Y}, \mathbf{A} \le \mathbf{N}$
$1 \le \mathbf{K} \le 10^4$
$1 \le \mathbf{C} \le 1000$

UNITED INTERNATIONAL UNIVERSITY
Department of Computer Science and Engineering (CSE)

Q4: Burst The Balloons

There are some spherical balloons taped onto a flat wall that represents the XY-plane. The balloons are represented as a 2D integer array of points where `points[i] = [`x_{start}`,` x_{end}`]` denotes a balloon whose horizontal diameter stretches between x_{start} and x_{end}. You do not know the exact y-coordinates of the balloons.

Arrows can be shot up directly vertically (in the positive y-direction) from different points along the x-axis. A balloon with x_{start} and x_{end} is burst by an arrow shot at x if $x_{start} <= x <= x_{end}$. There is no limit to the number of arrows that can be shot. A shot arrow keeps traveling up infinitely, bursting any balloons in its path.
Given the array points, return the *minimum number of arrows* that must be shot to burst all balloons.

Example 1:

Input:	**Output:**
`[[10,16],[2,8],[1,6],[7,12]]`	`2`

Explanation: The balloons can be burst by 2 arrows:

- Shoot an arrow at x = 6, bursting the balloons [2,8] and [1,6].
- Shoot an arrow at x = 11, bursting the balloons [10,16] and [7,12].

Example 2:

Input:	**Output:**
`[[1,2],[3,4],[5,6],[7,8]]`	`4`

Explanation: One arrow needs to be shot for each balloon for a total of 4 arrows.

Constraints:

- $1 <= points.length <= 10^5$
- `points[i].length == 2`
- $-2^{31} <= x_{start} < x_{end} <= 2^{31} - 1$