Assignment Solutions : Graph - 6 DSU

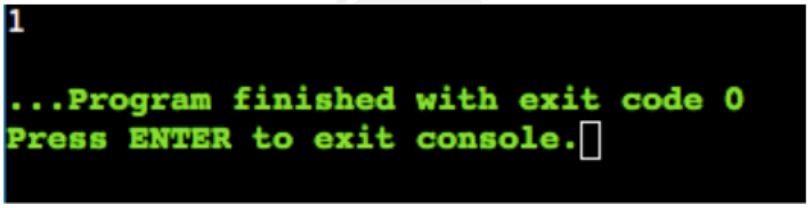
Q1 There are n computers numbered from 0 to n - 1 connected by ethernet cables connections forming a network where connections[i] = [ai, bi] represents a connection between computers ai and bi. Any computer can reach any other computer directly or indirectly through the network.

You are given initial computer network connections. You can extract certain cables between two directly connected computers, and place them between any pair of disconnected computers to make them directly connected.

Return the minimum number of times you need to do this in order to make all the computers connected. If it is not possible, return -1.

Code link: https://pastebin.com/BiqJuzct

Output:



Explanation :

- **The find function is a recursive function that performs path compression while finding the parent of a set to which element x belongs. It uses the par vector to store the parent of each element.**
- **The uni function performs the union of two sets to which elements x and y belong. It uses the rank vector to store the rank (or size) of each set. By performing union by rank, we optimize the DSU operations.required operations.**

- **The makeConnected function takes the number of computers n and the connections as input. It initializes the par vector to store the parent of each element and the rank vector to store the rank of each set.**
- **This loop iterates through the connections. If the two computers of a connection already belong to the same set (i.e., they have the same parent), it means the cable is redundant. Otherwise, it performs the union of the two sets using the uni function.**
- **After processing all the connections, it counts the number of disjoint sets. The variable required stores the number of sets minus one. If the number of required sets is greater than the number of redundant cables, it means it is not possible to connect all the computers, so it returns -1. Otherwise, it returns the number of**

Q2 On a 2D plane, we place n stones at some integer coordinate points. Each coordinate point may have at most one stone.

A stone can be removed if it shares either the same row or the same column as another stone that has not been removed.

Given an array stones of length n where stones[i] = [xi, yi] represents the location of the ith stone, return the largest possible number of stones that can be removed.

Code link: https://pastebin.com/a8mzNnU8

Explanation :

- **The find function is a recursive function that performs path compression while finding the parent of a set to which element x belongs. It uses the par vector to store the parent of each element.**
- **The uni function performs the union of two sets to which elements x and y belong. It uses the rank vector to store the rank (or size) of each set. By performing union by rank, we optimize the DSU operations.**
- **The removeStones function takes a vector of stone coordinates as input. It initializes the par vector to store the parent of each stone and the rank vector to store the rank of each set. It also creates two unordered maps, xCoord and yCoord, to store the mapping of x and y coordinates to stone indices.**
- **This loop processes each stone. It checks if the stone's x-coordinate or y-coordinate already exists in the corresponding unordered map. If both coordinates exist, it performs unions of the stone's index with the indices stored in the maps. If only one coordinate exists, it performs a union of the stone's index with the index stored in the map**


5
...Program finished with exit code 0
Press ENTER to exit console.