COP 3502 Fall 2023 Sections 1, 4 Recitation Program #2

Sky Islands

For each recitation program, in order to get full credit, you must submit your solution to open.kattis.com and get your solution accepted on all test cases. In addition, each one will have some separate requirements to fulfill based on your code. When submitting your work to Webcourses, please carefully read the corresponding directions document before submitting all of your files.

NOTE: Over the course of the semester, you must submit TWO out of the four recitation programs. It is expected that while you are in recitation, you start working on each of them. But, afterwards, you can choose which two to finish up.

Background Information

The intended algorithm taught in class required to solve this problem is very similar to floodfill. In a floodfill, the recursive function takes in an (x, y) coordinate where the fill is to occur, and if possible, recursively fills into adjacent squares.

In this problem, instead of the layout being a 2D grid, its ***N*** locations, where two locations are neighbors if they are connected by a bridge. Thus, instead of recursively "going to" 4 or 8 potential neighboring squares, here you can go to any number of neighboring squares, provided they are connected to the current location by a bridge. This means that inside the recursive function, there will be a loop that potentially runs *N* times.

Implementation Recommendation

There are many, many bridges (almost a million) potentially in the input, but there can never be more than 900^2 unique bridges (actually 450 x 899 to be exact.) The easiest way to store the bridges is to create a int 2D array of size N by N, where bridge[i][j] = 1 if there is a bridge from location i to location j, and where bridge[i][j] = 0 if there is no such bridge. Remember that bridges go both ways, so in the input, if there is a bridge from 1 to 2, that means there is also a bridge from 2 to 1.

Implementation Requirements

Dynamically allocated memory must be used to store which locations are connected, using the recommendation given above.

Although this problem can be solved non-recursively in several ways (both a breadth first search or use of a disjoint set can solve this problem without recursion), in order to get full credit **you must use recursion, preferably as specified in the background information section.**

What to Submit

Please submit the following:

1) Your source file, skyislands.c.

2) A screenshot of your solution's accepted status on Kattis.