CS29003 Algorithms Laboratory

Assignment No: 10

Last date of submission: 09–April–2019

Minimum Spanning Trees

In this assignment, you implement a second algorithm to create randomly generated bhulbhulaiyas. Consider again an $m \times n$ grid of square cells, and its adjacency graph G (two cells are adjacent if and only if they share an edge). Assign random weights to the edges of G. Compute a minimum spanning tree (MST) of G with respect to the assigned weights. This MST (instead of a random DFS tree as done in Assignment 9) will now define the bhulbhulaiya.

Part 1: Use the same data structure to represent a bhulbhulaiya as in Assignment 9. Copy the functions *initbhul* and *prnbhul* from this earlier assignment.

Part 2: Implement a function *MSTbhul* in order to generate an MST-based random bhulbhulaiya. For constructing an MST, implement Kruskal's algorithm. Assign random weights to the edges, and quick sort the list of edges with respect to these weights. Implement your own quick sort function. For implementing Kruskal's algorithm, you need a disjoint-sets data structure. You may use your implementation of this data structure in Assignment 8 with relevant modifications.

Notice that you can generate a random permutation of the edges, and pretend that this is the sorted list of edges. This is probabilistically equivalent, but we want you to make a complete implementation of Kruskal's algorithm. So use the generate-weights-and-sort method explained above.

Part 3: Rewrite the function *findrani* to locate the path from a source cell S to a (different) target cell T. In Assignment 9, you used the rooted DFS tree (stored using parent pointers) to find this path. Kruskal's algorithm, however, does not immediately give you a rooted tree, so you need to do something extra for the function *findrani*.

The *main*() function

- Insert the line `srand((unsigned int)time(NULL));` at the beginning of your *main* function.
- Read m and n from the user.
- Initialize the bhulbhulaiya, and print it (with all the walls in place).
- Call *MSTbhul* to generate an MST-based bhulbhulaiya. Print it.
- Generate two different random cells S and T. Find and print the S,T-path in the bhulbhulaiya.

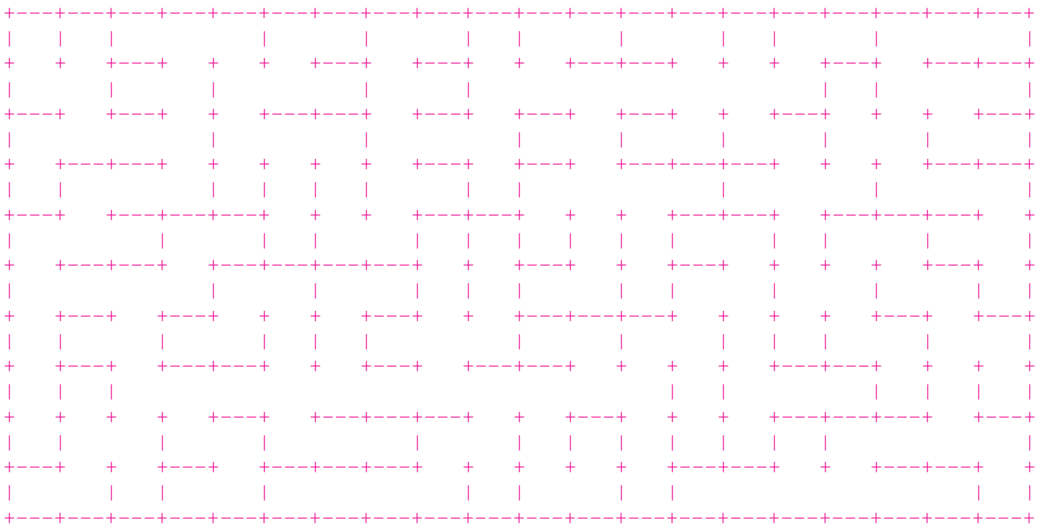
Submit a single C/C++ source file. Do not use global/static variables.

Sample output

```
m = 10
n = 20

+++ Initial bhulbhulaiya
```

```
+++ Random bhulbhulaiya generated
```



```
+++ Path from S = (4,14) to T = (9,17)
```