Question 2:

A simple algorithm for maze generation is to start, apart from entry and exit points, with all walls present and randomly knock down walls until the entry and exit points are connected. Write a C++ program to implement this algorithm for an arbitrary-sized maze – test with a 50 by 88 rectangular maze.

Using the disjoint set and Kruskal algorithm, we create a 2D array with each cell storing a pair of (w,h) coordinates. Assigning the entry and exit points. Run the Kruskal algorithm, every time the algorithms pick a random pair that was not connected to the same root parent, it will knock out a random wall around that cell.

Data structure:

Maze :

Pair<int,int> parent[w,h];

Pair<pair<int, int> , pair<int, int>> unblockedWall;

Pseudo-code:

Find : this function will recursively look for the parent most root. (find where the cell can go to another cell and return the ended cell)

|  |
| --- |
| Function find (int x, int y):  If (x,y) != parent[x,y]  Return find(parent[x,y])  Return parent[x,y] |

Union : This function is replacing 2 pairs to one parent. (checking if 2 cell can connect to a path, if not, we make it connect)

|  |
| --- |
| Function union (pair x, pair y):  Parent1 = find(x)  Parent2 = find(y)  If parent1 == parent2:  Return;  Let parent1 = parent2 |

MazeGeneration: This function will apply Kruskal Algorithm to create a map:

|  |
| --- |
| Function genrate (pair start, pair goal):  While(parent of start != parent of goal):  randomCell : pick a random cell  neighbor : find a random neighbor of this cell  if (this cell have different parent with the neighbor):  union(randomCell, neighbor)  unblockWall.add(the wall connect the cell with its neighbor) |

To print the maze, generate a maze template that does not have any path. Delete the unblocked wall.

Here is an example of a 30x20:

A picture containing diagram

Description automatically generated

There are more example of the bigger maze in the 5588.txt, 100vs100, and 200vs200 files