Drew Pulliam – DTP180003

CS3345.0U1

Assignment 8

1. x

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| start | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| Union(find(0),find(1)) | -2 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| union(find(2),find(3)) | -2 | 0 | -2 | 2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| union(find(4),find(5)) | -2 | 0 | -2 | 2 | -2 | 4 | -1 | -1 | -1 | -1 | -1 |
| union(find(4),find(6)) | -2 | 0 | -2 | 2 | -3 | 4 | 4 | -1 | -1 | -1 | -1 |
| union(find(7),find(8)) | -2 | 0 | -2 | 2 | -3 | 4 | 4 | -2 | 7 | -1 | -1 |
| union(find(7),find(9)) | -2 | 0 | -2 | 2 | -3 | 4 | 4 | -3 | 7 | 7 | -1 |
| union(find(7),find(10)) | -2 | 0 | -2 | 2 | -3 | 4 | 4 | -4 | 7 | 7 | 7 |
| union(find(1),find(5)) | 4 | 0 | -2 | 2 | -5 | 4 | 4 | -4 | 7 | 7 | 7 |
| union(find(3),find(9)) | 4 | 0 | 7 | 2 | -5 | 4 | 4 | -6 | 7 | 7 | 7 |
| union(find(1),find(3)) | 4 | 0 | 7 | 2 | 7 | 4 | 4 | -11 | 7 | 7 | 7 |

------------------------------------------------------------------------------------------------------------------------------------------

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 4 | 0 | 7 | 2 | 7 | 4 | 4 | -11 | 7 | 7 | 7 |

------------------------------------------------------------------------------------------------------------------------------------------

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| start | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| Union(find(0),find(1)) | -2 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| union(find(2),find(3)) | -2 | 0 | -2 | 2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| union(find(1),find(3)) | -3 | 0 | 0 | 2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| union(find(4),find(5)) | -3 | 0 | 0 | 2 | -2 | 4 | -1 | -1 | -1 | -1 | -1 |
| union(find(4),find(3)) | -3 | 0 | 0 | 2 | 0 | 4 | -1 | -1 | -1 | -1 | -1 |
| union(find(6),find(7)) | -3 | 0 | 0 | 2 | 0 | 4 | -2 | 6 | -1 | -1 | -1 |
| union(find(8),find(9)) | -3 | 0 | 0 | 2 | 0 | 4 | -2 | 6 | -2 | 8 | -1 |
| union(find(9),find(10)) | -3 | 0 | 0 | 2 | 0 | 4 | -2 | 6 | -2 | 8 | 8 |
| union(find(7),find(10)) | -3 | 0 | 0 | 2 | 0 | 4 | 8 | 6 | -3 | 8 | 8 |
| union(find(10),find(5)) | 8 | 0 | 0 | 2 | 0 | 4 | 8 | 6 | -4 | 8 | 8 |

------------------------------------------------------------------------------------------------------------------------------------------

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | 0 | 0 | 2 | 0 | 4 | 8 | 6 | -4 | 8 | 8 |

------------------------------------------------------------------------------------------------------------------------------------------

1. Path Compression

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| start | -1 | 0 | 0 | 2 | 2 | 1 | 1 | 5 | 5 | 7 |
| Find(9) | -1 | 0 | 0 | 2 | 2 | 0 | 1 | 0 | 5 | 0 |

------------------------------------------------------------------------------------------------------------------------------------------

1. To create a maze, you first start with a beginning and an end cell (for the example, top left and bottom right). Then you have a loop that runs until all cells are connected (in the same set). Inside the body of the loop you repeatedly select a random cell, and a random wall of that cell. Determine if that cell is already connected (in the set). If it is not connected, “knock down” the wall that was randomly selected, and then union the cell with the cell adjacent to it (the wall that was just knocked down). After the loop finishes, all cells are connected, meaning there must be at least one possible path from the beginning to the end of the maze.

------------------------------------------------------------------------------------------------------------------------------------------

1. The union data structure is useful because it allows the code to know when to stop (all cells are connected). Without it, checking each loop iteration would be more expensive. The find command is also useful because each loop iteration you can simply call union(find(randomly selected cell), find(adjacent cell)). This allows the union to add your new cell onto the root of the set that it just merged with.

------------------------------------------------------------------------------------------------------------------------------------------

1. I think that the simplest way for the program to know which walls have been removed would be to use a separate data structure. A two-dimensional Boolean array (with true if the wall exists, and false if it has been removed) would probably be the simplest option.

------------------------------------------------------------------------------------------------------------------------------------------