CS319 Project

Alan Flynn

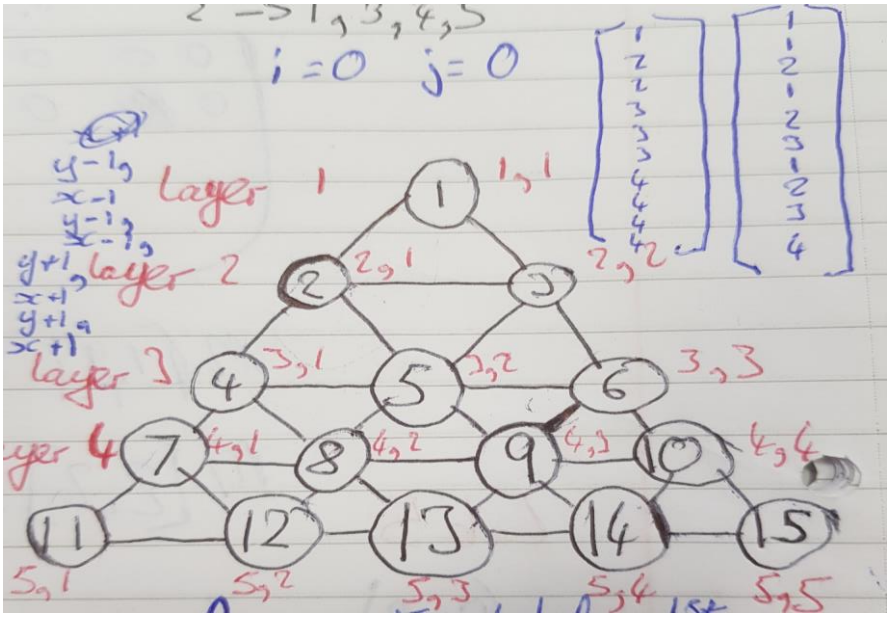
ID: 16331333

My project involved taking creating a matrix to represent the connections between the nodes of a triangular grid graph and then to test whether all the nodes were within 6 degrees of separation of one another. If they weren't I then had to keep adding in random edges until they were.

My initial issue was creating the initial matrix or triplet to hold the connections between the nodes.

I initially tried to do this by creating an array that would hold a stack of stacks holding the index of each value (e.g. [[1],[2,3][4,5,6] …. etc.) but programming this in c++ proved dificult

So I then decided to make a list of coordinates for each of the nodes such as in the diagram {(1,1),(2,1),(2,2),(3,1),(3,2),(3,3) etc.}



I did this with two nested for loops having the x value stay stagnant until the y value counted up from 0 until it was the same as the x value, then the x value added one and it started again. I held these values in vectors labelled x_values and y_values

Once I had their coordinates I could use those to get the connections. On a triangular grid graph each node can be connected to 6 possible nodes if such nodes exist. These are the nodes to the left, above and left, above and right, right, below and right and below and left. A good example would be position 5 in the diagram (Coordinates 3,2) which is connected in all six locations. I did this using two nested for loops checking which of the six nodes with the given coordinates exist and then updating the connections matrix/triplet matrix to include these

Once I had the connections matrix I updated it with the Floyd algorithm to show the degrees of separation for each. It was for this reason that I stopped using the sparse triplet matrix as the only zero values would be the diagonal since every other value would represent the degrees of separation of each node

After this I used a for loop to check if any values in the distance matrix were greater than 6 and if so I would then run through the while loop in order to keep adding in random edges and updating the distance matrix until all nodes were within 6 degrees of one another. A problem I encountered here was implementing the rand() method as initially it kept giving the same sequence of seemingly random numbers but I fixed this by adding in the srand() function before the for loop.