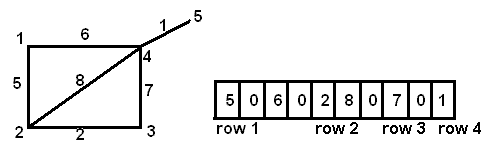
EECS 560 Lab 6 Week of February 29

Due to the symmetry in an undirected graph, it is only necessary to store the edge weights in the upper triangle of the adjacency matrix (above the diagonal). For example, if you want to find the weight of the edge (5, 3), due to symmetry you can get this information by looking at the weight of (3, 5). For this lab you will randomly generate connected graphs and store their information in a one dimensional array corresponding to the upper triangle of the adjacency matrix. This will require you to determine the index of the first element of the second row, third row, etc. Since the graphs don’t have loops, the main diagonal which would be all 0s is not included in the one dimensional array. Consider the example below:



Note that the first row has four values, the second row has three values, etc. Write out the adjacency matrix to be sure you understand how this is represented.

Random graph generation

In case you have forgotten, review how to generate random numbers as done in lab 3 In generating random weighted graphs you will first generate a random number to determine if an edge is present, and if so randomly generate a weight for that edge. Use the following methodology:

1. Prompt the user for the number of vertices in the graph and the seed for the random number generator. (This is to make it easier to grade the program.)

2. The graphs you generate will have edge density of 60%. Generate a random number between 0 and 1. If that number is less than or equal to 0.6 then there will be an edge. In this case generate an integer edge weight between 1 and 20 and put that weight into the one dimensional array. If the number generated is larger than 0.6 put a 0 into the array to indicate there is no edge between those two vertices.

3. If there are n vertices in the graph, use the following order for edge generation:

(1, 2), (1, 3), …, (1, n), (2, 3), (2, 4), …, (2, n), …, (n-1, n). That is, you will be filling the cells of the array in left to right order. (1-based indexing makes this a bit easier than 0-based.) Remember that not all cells will contain nonzero edge weights since if the first random number generated is greater than 0.6 no edge is present.

4. Now, since all graphs are to be connected you must determine if the graph you generated is connected. To do this you will need to use a depth first search. This will require that you have a visited array to indicate if you have seen a vertex. (See the dfs template at the top of page 420 in the text.) When your search stops, if there are any unvisited vertices it indicates the graph is disconnected. In this case, put in an edge between the last vertex visited and the first unvisited vertex. Do this by generating the edge weight and putting it into the array. Then resume the depth-first search. Continue in this way until all vertices have been visited.

5. Print out the array corresponding to your graph.

To turn in:

As usual turn in all code that you write. In addition, randomly generate connected graphs of two sizes: 8 vertices and 20 vertices. Print out the arrays that contain the edge weights, and for the 8 vertex graph, turn in a drawing of the graph (hand-drawn is fine).

All labs should be turned in by midnight on Saturday.

Will not receive credit if have a two dimensional array in your program.

Note: cannot do timing tests by just combining the other data files.