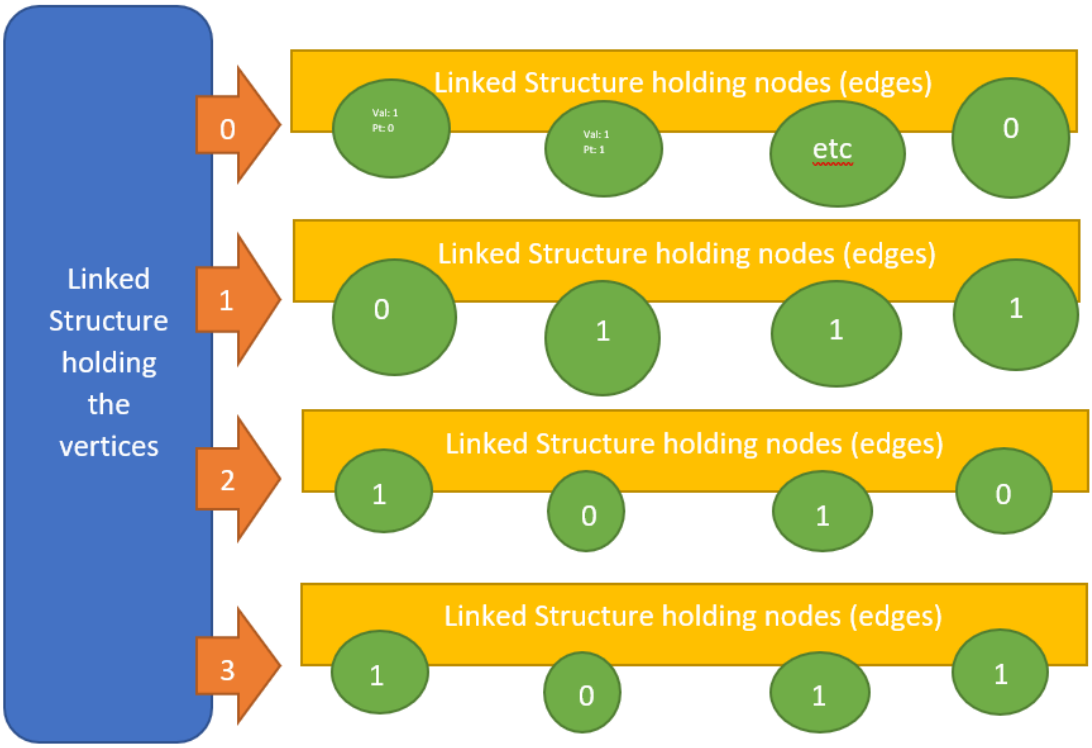
Lab 3

Arta Seyedian

Program

This program is the same program as the one developed in lab 2, with the caveat that the adjacency matrix has been read in as a linked structure, as opposed to an array matrix as in lab 2. Two new classes have been made for this lab: Node and LinkedStruct. These objects are linked objects which contain a next pointer. As with the previous lab, the input file is broken up line-by-line and integer-by-integer to be processed by the program. However, each binary '1' or '0' is, instead of being stored as elements in a matrix, stored as a binary value in each node. The node also stores a variable by the name of nextVert, which is a numerical value that corresponds to the number of the vertex in the graph. The basic idea behind the linked structure of this program is below:



The nodes were implemented in a 'hybrid' fashion. Each one of these nodes only contain two variables: binary value and next pointer. The next pointers point to the vertices, which also hold their own nodes. The recursive method from the previous lab is still used, but what is different is that the node's own next pointer is submitted as the next vertex in each successive recursive layer. Therefore, the computation of the paths is done with the next pointers. The path already traversed is kept track of using stacks.

To me, this was the simplest possible modification of lab 2 that would be permissible but still used pointers. As you can observe, the arrays which back the linked structures and vertices are initialized but not given a size until the dimensions of the adjacency matrix are read. While there is still no dynamic allocation, the arrays are only ever as large as required by the dimensions. The LinkedStructs are originally the holders of the nodes, but the nodes are passed onto the vertices for computation, as with the previous lab with the use of the 2-dimensional adjacency matrix. Both the vertices and the LinkedStructs have arrays of Nodes which act as connections from that object. Above all, the guiding principles in my design were simplicity and meeting the requirements. Alternatively, I could have contained references to the actual vertex objects instead of simply the integer value which corresponds to that vertex in the array in the graph object, but in a hybrid implementation, which is what I very intentionally sought after, that is not necessary.