1. We have focused primarily on *time complexity* in this course, but when choosing data structures, space complexity is often as important of a constraint. Given an adjacency matrix, what is the 'space complexity' in Big-O. That is, given *n* nodes, how much space (i.e. memory) would I need to represent all of the relationships given. Explain your response.

   The space complexity of adjacency matrix given n nodes is O(n^2). Because we need to represent connection between every other node. When one node is the source node, we check all the other destination. There will be n connection to check. And then we check every n nodes. So the total space complexity is n * n = n ^ 2.

2. Will it ever make sense for ROWS != COLUMNS in an adjacency matrix? That is, if we want to model relationships between every node in a graph, must rows always equal the number of columns in an adjacency matrix? Explain why or why not.

   If ROWS != COLUMNS for an adjacency matrix, we do not represent every possible connections between every node.

3. Explain the difference between the method we've used in adjacencymatrix.c and the method to dynamically allocate a 2D array that you chose in Part 3. Describe the method you've chosen for malloc'ing a 2D array, and why it is necessary.

   I will choose method1. Method 1 is allocating the memory of the value of Rows * Columns which is the how many spaces that required to store the array. This method is straightforward.