Homework 5 - Problem 2

This algorithm runs through a set graph and tells you how many edges you would need to connect to make every vertex connected

```
Let int V = num Vertices
Let set Adj be a graph of all the values (2D Array of Values and their pairs)

DisjointComponents:
    boolean[] visited = new boolean[V];
    Let set visited = new set of booleans of size V
    Let disjoint = 0

    for every value < V in visited:
        If value was visited:
                DFS(v, visited)
                disjoint++
    return disjoint

DFS:
    Given v is the vertex number
    Given set visited (from above)

    Set value in visited at v to true;
    For every value w in V (from above):
        If value in adjacency list == 1 and hasn't been visited in visited:
                DFS(w, visited)

main:
    Given number of vertices v
    Given number of edges e

    For every edge inputted, let t1 be vertex 1, and t2 be vertex 2
        Set ADJ to all the incoming values, and set the values to one at:

    DisjointComponents()
```

This works because:
It creates a 2D array of values that all have each other vertices' they have an edge with. For example if the input was:
3 1
1 2

Then the 2D array would look like this:
[0] - [nothing]
[1] - [2]
[2] - [1]
[3] - [nothing]
Because of this, we can check for each one in the array that has [nothing] and return that count, adding an edge for every vertex with [nothing] and using some simple math to remove 0.

Time Complexity: O(m+n)

It's very simple and only has 1 loop per, it only checks the M or the N of each. Plus the "visited checker", makes it easy to tell whether it's been visited or not.