# Algorithm Analysis

For our project we decided to use a greedy DFS recursive  search to find the best path from the source to the sink. We pass into our DFS function the random board array that was generated along with the source indexes, and the list of visited nodes.

The first thing that the algorithm checks is if the indexes passed in are the sink indexes and returns true if it is. Next it is going to check if the indexes are outside the board and returns false if it does since we don't want any nodes outside the board. Another check that it does is check if we are at or above max capacity for the edges. After that it does another check and checks to make sure that we didn't already visit this node. This way the algorithm doesn't go into loops going back to that same node.

After we do all the basis step checking we mark the current node as visited and start the process of which node it can go to is the next best node. We did this by checking all the nodes it can possibly go to and calculate all the flows. Once it has done that it will find the current flow of the path ((node1_value + node2_value) / 2) and determine which is smaller. The smaller of the 2 will be our maximum flow. Once all these operations are completed it will call DFS again and calculate the next node it should go to.

Once the algorithm reached the sink, all the function calls will percolate upwards and the graph will be drawn from the sink to the source since the function is recursive. Once all the drawing is done the program will update the html accordingly with the current flow and the number of used and unused edges. One thing that can be improved with this algorithm is while there are still extra edges to be used try and run the algorithm again to see if it can reach the sink in another path to increase the max flow of the graph.