**Result File**

**Purpose if the task:**

In this task we will implement a function to create random graphs and in addiction we will build auxiliary functions that will help to explore properties of the graphs.

**Functions i built:**

**vector<vector<int>> build\_random\_graph(int v, double p) -**

This function get a number of nodes and a probability number and return a random graph.

The function does this by raffle a random number and if it is equal to the probability set, there will be a edge between two nodes and we will add the nodes to the adjacency vector if each nide respectively.

**int is\_isolated(vector<vector<int>>& graph) -**

This function gets a random number graph, checks if there is a single node by running on the node vector, and checks if its adjacency vector is empty.

Returns 0 if none exists, returns 1 if exists.

**int BFSconnectivity(vector<vector<int>>& graph, int startingVertex) -**

This function gets a random graph, an initial node and checks if the graph is connected.

This function does this by the BFS algorithm and checks only if it has visited all node of the graph, if it has visited all the nodes if will return 1 otherwise 0.

**int connectivity(vector<vector<int>>& graph) -**

This function gets a random graph, checks if it is a connected graph.

This function checks whether there is a single node if there is a single node it return 0 I.e. the graph is not connected graph, if it does not exist the function check by one run of BFS algorithm whether the graph is connected by the BFSconnectivitty.

**int BFSdiameter(vector<vector<int>>& graph, int startingVertex) -**

This function gets a random graph, and an source node, checks what the diameter of the graph is from source node. Return the greatest distance.

**int diameter(vector<vector <int>>& graph -**

Receives a random graph, checks by the connectivity function whether the graph is connected, if not returns –1 ie diameter is equal to infinity, if the graph is a connected graph the function returns the diameter of the graph by running BFSdiameter for each node in graph. The function returns the diameter of the graph.

**void writeToFile(double\* threshold1, double\* threshold2, double\* threshold3,int\* countConnected, int\* countDiameter, int\* countIsolated) -**

This function gets a arrays of probabilities according to the attributes and result arrays of each attribute respectively. Opens a CSV file for each attribute and inserts the values into it.

**Main function -**

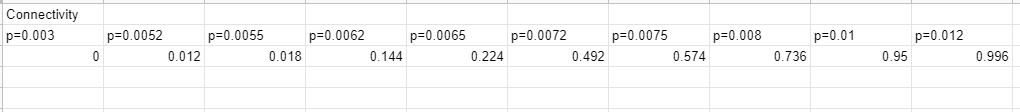
In the main function i made a permanent array for each argument, an array for each count of graphs that hold the argument according to the defined probability.

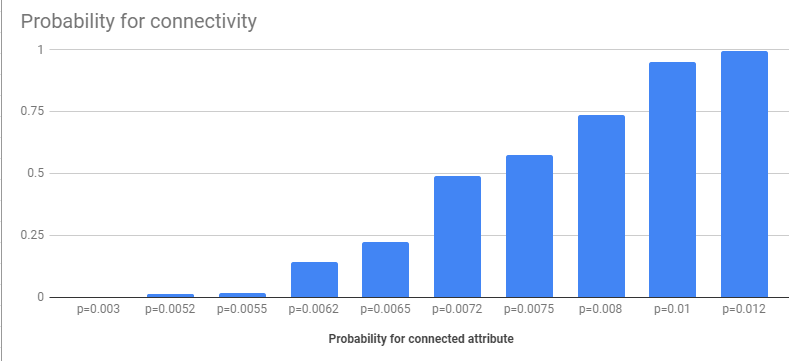
For each claim a loop that goes through all the probabilities and checks whether the claim holds, finally a function call to write a CSV file for each claim.

**Conclusions from the study:**

**Claim 1 – is the graph is a connected graph?**

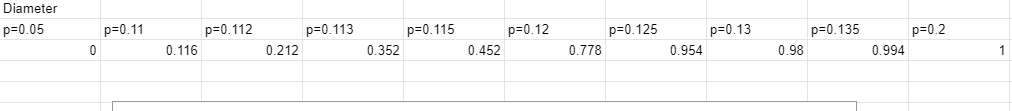
The claim is true if we note that the smaller probability than the threshold, the lower probability that the connected graph is.

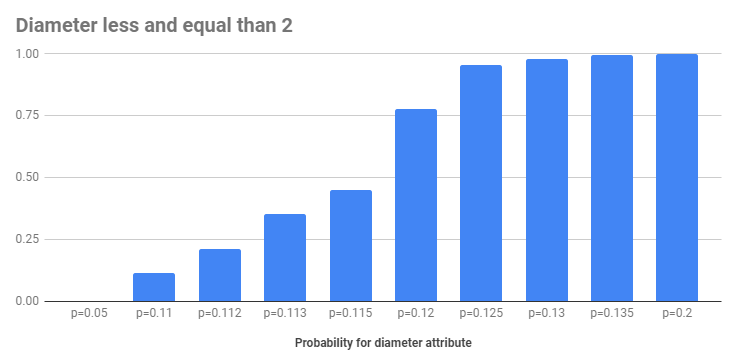




**Claim 2 – is the graph diameter equat to 2?**

The claim is true if we note that the smaller the probability than the threshold, the lower the probability that the graph diameter will be equat to 2, and the higher the probability that the threshold, the higher the probability of a small diameter equal to 2.





**Claim 3 – is there a single node in the graph?**

The claim is true if we notice that the smaller the probability than the threshold, the higher the probability that the graph will be an isolated node, and the higher the probability than the threshold, the lower the probability of an isolated node.

