2. Describe (do not implement) how you would update the above implemented `random_graph` method to generate a graph $G = (V, E)$ that does not contain a negative-weight cycle. You are given a function that can determine whether or not an edge completes a negatice-weight cycle.

After generating the random_graph. Checking whether or not edge completes a negative weight cycle.
Given a weighted directed random_graph G = (V, E) with weight function ***w: E →R,***
Given the function **negative_cycle_found()** that determine whether or not an edge completes a negative-weight cycle.

For inputting the edge in the graph using **m_edge matrix.**
if this edge creating cycle (checking using the function **negative_cycle_found()**)
then delete this edge
and increment the counter by 1.
else add the edge into the graph.

Whenever an edge is added to the graph it will check if it forms negative cycle or not. If any negative cycle is formed, it will remove that edge and start further with the other edges.

3. Implement the Bellman-Ford algorithm. What is the running time for Bellman-Ford using an adjacency matrix representation?

For using adjacency matrix in the Bellman Ford, it will take O (n^2) time per iteration for a total runtime of O (n^3).