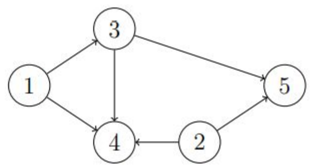
DSA Homework 7

Katie Foster

Part 1



To create a valid topological ordering, the algorithm must go through the graph, using a recursive implementation of depth-first search to check if a topological ordering exists by checking for cycles, and adding nodes to the topological order stack as it goes.

Non recursive part:
- Mark all nodes as unvisited
- Create stack topologicalOrder
- For node in graph:
    - If unvisited:
        - topologicalOrder = TopologicalOrdering(node, topologicalOrder, [])
- Return topologicalOrder

TopologicalOrdering(StartingNode, topologicalOrder, keepingTrackList):
- Add StartingNode to the keepingTrackList
- For node in StartingNode's neighbors
    - If node is in keepingTrackList:
        - Return None (because we have a cycle)
    - topologicalOrder= TopologicalOrdering(node, topologicalOrder, keepingTrackList)
    -
- Add StartingNode to topologicalOrder
- Mark StartingNode as visited
- Return topologicalOrder

Part 3

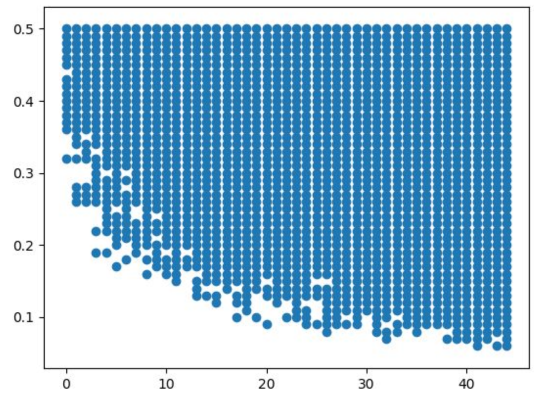


Figure 1: Scatterplot of number of nodes vs probability at which all instances of graph form one component

This is a scatterplot in which each dot represents an instance of probability (on the y axis), and number of graph nodes (on the x axis), for which all 10 tested graphs formed a single component. The graph shows that as the number of nodes increases, the probability that is required for each pair of nodes to be connected decreases.