Homework 7: Graph Traversal and Components

Data Structures and Algorithms Spring 2020

Sparsh Bansal

1 Question 1

It is given to us that we need to order the nodes in a directed graph G = (V, A) such that for all directed arcs (u, v), u comes before v in the ordering.

I am using the Depth-First Search Technique to implement this algorithm. I will be using a hash map (Visited) and a stack (Sorted). This algorithm will begin sorting by randomly picking one node from the graph. Then, it would search along the depth of any one of its neighbors, until it reaches a node without any outgoing edges or a visited node. The important addition to this search is that it temporarily stores the node in consideration as it checks an outgoing neighbor. In the first case, it marks this node as visited in Visited, and stacks it onto Sorted. In the second case, it marks the incoming node as visited in Visited, and stacks it onto Sorted. This process is called recursively until all nodes in Visited are marked at visited. Once recursion has ended, the elements are popped out of the stack and printed in a topologically sorted order.

The base case for the recursion would be that no nodes are available in Visited, marked as not-visited in order for depth-first search to occur. To check if the given directed graph is cyclic or acyclic, we can use a check at each of the recursion steps. If the algorithm finds the starting node in the depth, then it has successfully found a cycle. Correspondingly, a topological order for this graph would not exist, since it is not possible to satisfy the condition that for all directed arcs (u, v), where u comes before v. The pseudocode for this algorithm is given below.

```
Visited -> {Node : Not Visited}
Sorted -> []

func Visit(n):
    # Ensures that recursion reaches an end
    if n is visited according to Visited:
        exit
    # Ensures that the the graph is acyclic
    if n has a temporary mark:
        notify the caller that the given graph is cyclic at n

    # Ensures that the the graph is acyclic
    assign a temporary mark to n

    # Recursion occurs here - Depth First Search Algorithm
    for each outgoing_neighbor of n:
        Visit(outgoing_neighbor)
```

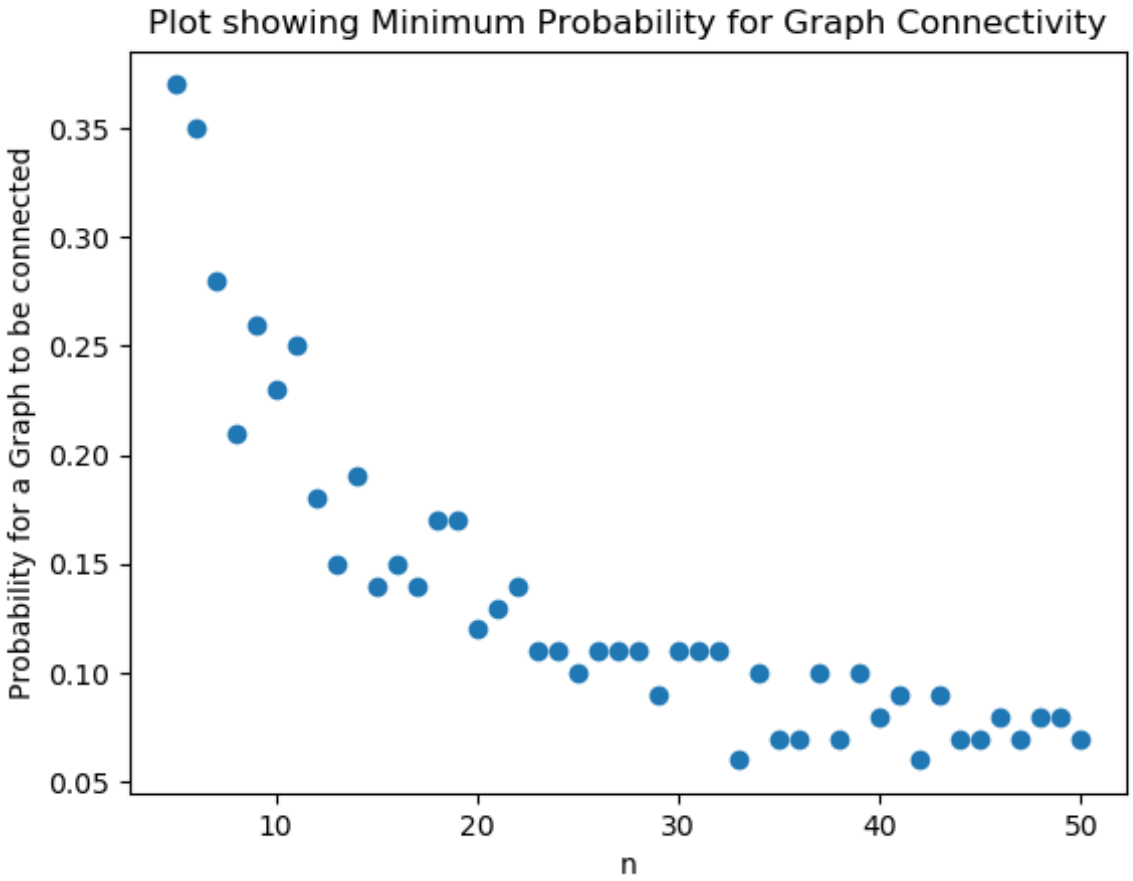
```
        mark n as visited in Visited
        delete the temporary mark
        stack n onto Sorted

    # Unvisited nodes are chosen and visited - correspondingly,
    # their depths are searched for
    While there is an unvisited node in Visited:
        Choose an unvisited node n
        Visit(n)

    # Generates an output containing topologically sorted
    # graph nodes
    for each node in Sorted:
        pop(node)
        print(node)
```

2 Question 2

Program in a separate file

3 Question 3

3.1 Plot



3.2 Observations

We can observe that the factors on the x and y axes vary inversely in this graph. Graphs with higher number of nodes have a higher chance of having a disconnect, hence low probability for showing fully connectedness, relative to graphs with lower number of nodes. Additionally, the spread of values in terms of the number of nodes with respect to probability is higher for lower probabilities (for probability = 0.07 - n ∼32-52) than high probabilities (for probability = 0.25 - n ∼8-12)