Project-2 CS101

Neredumalli Likhit-2023CSB1139

April 2024

Graph making

We conducted an experiment which involved student noting impressions of each other. This impression dataset was given to us in the form of an excel file. For converting this into a directed graph, I took the help of pandas and networkx modules. Furthermore, for converting the nodes into proper student id's, I sliced the email address string to 1st 11 letters and sliced the impression columns' elements/strings consisting of names to last 11 letters . I had to convert them into uppercase if required using .upper Then, edges were added from email address column node to impressions column nodes. Thus forming a graph with 143 nodes and 3438 edges.

Q1)

In question 1 we had to perform a random walk on the graph, and find the top leader based on random walk points. For this I defined a random walk function which returns random walk points of each node. In this function, a node is first selected at random and then it's RW points are incremented to 1(which were initialized to zero).Next, it checks the out nodes of this random node, pick one ut node at random among them and it's RW ponts are similarly incremented to 1.I've also included the case of out nodes being zero, in which function does teleportation and picks another node at random. This process continues for 100000 iterations and then the function returns RW points. I sorted these nodes based on RW points using another function.This one created a dictionary with nodes of graph as key and it's corresponding RW points as values.Then I sorted the nodes into a list based on their RW points from the dictionary. Upon running the program, It turns that the top leader is 2023CSB1091.

Q3)

For this question, I came up with 2 problems.

3a)

Find the number of transitive triangular relations in the graph?

Ans: for this I created a function, which checks that if an edge exists between a node in the graph and the neighbour's neighbour of that node.If so it increments the count to 1.It does so for all nodes. At the end, I had to divide the count by 2, to avoid double count(there can also be a case where node3 is found first, then we go to node 2, then to node 1 , and find edge exists b/w node3 and node1, but we already the counted it for the case when we went node1 first and so on) The answer is 15139

3b)

Upon picking an edge,how probable is it that edge turns out to be unidirectional?
Ans: For this I created two functions. The first function, counts the number of unidirectional edges by picking a node's edge, then checking if reverse edge involving those nodes exists in the edges of the graph.So, now we have no. of unidirectional edges in graph. Second function, finds the probabibilty by dividing the undirectional edges count by dividing it by total no. of edges in graph. The answer is 0.29959

Q2)

In this question, I found the missing links based on the idea that, if two nodes in the graph have many common neighbours, but there's no edge between these two nodes, then a missing link must exist between these two nodes.For this I first created a matrix similar to an adjacency matrix, but instead of zeros and ones, it will contain the no. of common neighbours between the two nodes(by checking intersection of neighbours of node1 and node2) Then I defined a missing links function.There must be threshold of common neighbours to have,for checking the existence of missing link.I assumed this to be 10 in the program, but this could be anything other than zero.So this function checks that if an edge doesn't exist between two nodes and if their common neighbours are greater than 10, add it to missing links list.I then printed the missing links and number of missing links. The no. of missing links between nodes having common neighbours greater than 10 is 393.