Indian Institute of Technology, Madras
Department of Applied Mechanics & Biomedical Engineering
Computational Tools: Algorithms, Data Structures and Programs - ID6105

Assignment set - IV

To be submitted by: Tuesday 14$^{\text{th}}$ October, 2025.
The problem appearing in blue are optional for extra marks.

1 Problem 1

Preliminaries: Consider a graph $G = (V, E)$, show the following,

(a) Show that the number of vertices of odd degree in a finite graph is even.

(b) Show that every graph contains two vertices of equal degree.

(c) Show that for a simple connected graph with n number of vertices and e number of edges, $(n-1) \leq e \leq (n+n^2)/2$.

(d) For the above question, show that the lower limit corresponds to a tree (with a root node and a unique path between any two vertices) while the upper limit corresponds to a complete graph.

(e) Show that if a simple graph has more than $(n-1)(n-2)/2$ edges, then it must be connected. (Hint: Use induction).

(f) Given an arbitrary simple planar graph with n number of vertices and e number of edges, show that the maximum number of edges, M, that can be added to the graph, subject to it remaining planar is given by $M = 3n - e - 6$.

---

2 Problem 2

Write a python code to compute the adjacency matrix given a simple graph $G = (V, E)$, with V being a list of vertices and E being a list representing the pairs of nodes that are connected. Also write a code which does the opposite, which is, given an adjacency matrix output the graph structure as $G = (V, E)$. Test the program on the following set of standard graphs,

(a) Complete graphs: K_4, K_5, K_6, K_{10}.

(b) Heawood graph shown in Fig. 1.

---

3 Problem 3

Download the data for anonymised, small-scale Facebook ego-networks from (https://snap.stanford.edu/data/) and test the '6 degree of separation' by computing the following,

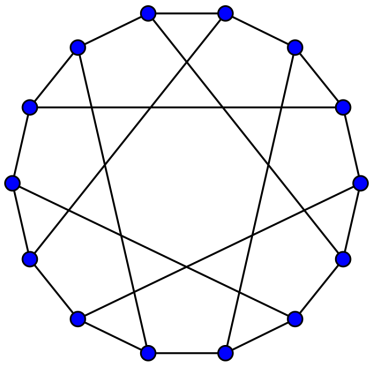

Figure 1: Heawood graph

(a) Find the average degree of vertices.

(b) Take 2 vertices at random and compute the shortest-path (number of connections). Repeat this to get the average separation.

(c) Plot also the distribution (histogram) of the shortest path to see how many pairs are distance $1, 2, 3 \ldots$.

(d) Find the mean, standard deviation and median of the distribution.

---

4 Problem 4

Connectedness is an important measure for understanding the reliability and effectiveness of a graph, and is used in many different applications. Here we aim to compute the connectedness of a Erdos-Renyi graph and show that it goes through a phase transition.

Write a code to construct an Erdos-Renyi graph $G(n, p)$, where n denotes the number of nodes and p denotes the probability that two nodes are connected by an edge. Choose the values of n to be $n \in [100, 200, \ldots, 1000]$ and p to be in the range of $p \in [0.5 \log n/n, 2 \log n/n]$.

(a) Use BFS or DFS to see if the graph is connected or not.

(b) Taking 30 different graphs (random generation using random seed), for each p, n find the average number of graphs which are connected.

(c) Explain the behaviour as p is varied.

(d) Explain the behaviour as n is varied.

Now let us change the problem to explore connectedness using the idea of percolation. Percolation indicates whether information can flow from one end to another (useful for disease modelling/flow etc). Consider a lattice $N \times N$, with each site being either open with probability p or closed with probability $1 - p$.

(e) Write a code to set up and visualise the grid.

(f) Use BFS or DFS to check if the grid is percolatable, that is, starting from any open sites at the top row can you find a path to the bottom row only through open cells. If such a path exists then the grid is percolatable, if no such path exists the grid is not percolatable.

(g) Taking $N = 100$, and p to be in the range $[0.4, 0.7]$. Plot the average number of grids which are percolatable as a function of p.

(h) Explain the behaviour of the above curve.

---

5 Problem 5

Consider the data set shared as 'Edges.txt' which has the information regarding connections on a semiconductor chip (the first two columns indicate the nodes that are connected by an edge, and the last column the weight). The graph has 2903 nodes.

(a) Check if the graph is connected, if yes proceed to the next question, if not add find the largest connected component and proceed. (For largest connected component: find the subgraph which has the maximum number of nodes in it).

(b) Write a program to find the minimal spanning tree for the graph.

(c) Write a program to find the minimal distance to all nodes, from the node which has the highest degree.

(d) Taking the weights to represent the capacity to cool, starting from the node with the highest degree, find the maximal flow that can be passed through the network till the farthest node which has a degree 3. Hint: You will have to make the graph directed for this exercise, easiest way is to make each connections to represent both forward and backward connection with same capacity as before.

---

6 Problem 6

Repeat the above set of problems by taking data from the OpenStreetMap, which is also available through the osmnx pacakge in python. Choose the nearest city to your hometown and redo the analysis.
