Total: 24 points

EE134: Graph Theory in Engineering

Midterm Solutions

Problem 1 (5 points) : For the graph in Figure 1, answer the questions that follow:

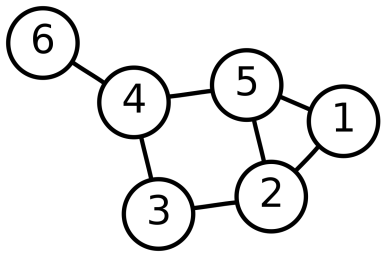

Figure 1: Graph for Problem 1.

(a) (1 point) Does the graph have a Hamiltonian path? Explain in a sentence why (or) why not. $6 \to 4 \to 5 \to 1 \to 2 \to 3$ is a Hamiltonian path.

Common Mistakes:

- We are interested in finding Hamiltonian path and not the Hamiltonian circuit. The given graph does not have a Hamiltonian circuit, but it has a Hamiltonian path.
- Application of Dirac theorem (every vertex having degree more than n/2 is Hamiltonian) is wrong here, because this theorem does not say that if we do not have such vertices, we will not have Hamiltonian circuit. For example consider a graph with 6 vertices connected in a ring. Here every vertex will have degree 2 which is less than $n/2$, but still we have Hamiltonian Circuit. Moreover, this theorem talks about Hamiltonian Circuit and not Hamiltonian path.

(b) (1 point) What is the diameter of the graph? Largest distance between any two vertices is 3, which is the diameter.

Common Mistakes:

- Some people wrote it as 4. Note that the distance between two adjacent nodes is 1 and not 2. The distance between any pair of nodes is the the minimum number of edges we need to take in reaching from one node to the other. And the diameter is maximum such distance among all pair of nodes.

(c) (1 point) Does the graph have an Eulerian circuit? Explain in a sentence why (or) why not. No, since there are vertices of odd degree.

(d) (2 points) Is it possible to add edges to this graph, so that the resulting graph has an Eulerian trail? Explain if yes or no, and if yes, what is the smallest number of edges you would need to add. Adding edge $(2, 4)$ gives us following Eulerian trail: $6 \to 4 \to 5 \to 1 \to 2 \to 3 \to \mathbf{4} \to \mathbf{2} \to 5$. And one is the smallest number of edge we will have to add as there are four vertices of odd degree and for a trail only two vertices can be of odd degrees.

Common Mistakes:

- Some people answered that minimum two edges are required as there are 4 nodes of odd degrees. Here we are asking for only a trail and not a circuit, so two nodes can be have odd degrees and we just need one edge to make other two nodes' degrees even.

Problem 2 (11 points) : Formulate each of the following problems as graph theory problems. In each case, you need to specify what the vertices and the edges of the graph are, and what problem it maps to. (You just need to map these problems to some graph theory problems- you do not need to solve them, or prove that there exists a solution.)

(a) (3 points) In computational biochemistry, there are some situations where conflicts between sequences in a sample can be resolved by removing some of the sequences. Assume that you are given a set of sequences, and you are also given that some of them cannot appear at the same time in the sample, i.e., there are pairwise conflicts between sequences. Your goal is to remove as few sequences as possible, so that there are no conflicts remaining. Formulate this as a graph theory problem.

- vertices: each unique sequence is a vertex.
- edges: an edge (i, j) exists if sequences corresponding to vertices i and j conflict.
- problem it maps to: finding the minimum vertex cover.

- mapping details: a vertex cover is a set of vertices such that all edges in the graph are incident on the set. For a set of sequences to have no conflict, there must be no edges between them. Thus finding the minimum vertex cover and removing all those vertices removes all conflicts and this is the least number of vertices we need to remove.

  Common Mistakes:

  - Greedily removing nodes one by one may not be optimal as we might end up removing more nodes compare to what Vertex Cover problem would have returned.

(b) (4 points) Assume that we have n computing nodes that we want to use to perform m tasks ($n \geq m$). To do so efficiently, we are going to use distributed computing: we will ask a (potentially different) subset of the nodes to work in parallel for each task. We thus divide the n computing nodes into m groups, where each group is responsible for one task. Each node may take part in more than one groups. When the computation is complete, we want to collect the computation outcome of each task to one (master) node from the corresponding group. For a given allocation of nodes into m groups - one for each task, can we select one master node from each group, so that no two groups have the same master node? Express this as a graph theory problem.

- vertices: one vertex corresponding to each computing node, and one vertex corresponding to each task.
- edges: an edge exists between a computing node vertex and a task if the vertex is assigned for that particular task.
- problem it maps to: Finding a perfect matching.
- mapping details: Such a mapping makes the graph bipartite with the computing nodes on the left and the task nodes on the right. Finding a perfect matching that saturates all vertices corresponding to tasks gives a unique computing node corresponding to each task; assign this to be the master node for the task (group).

  Common Mistakes:

  - If we consider only computing nodes as vertices of the graph, and add edges if two nodes are part of task, it does not account for which task they have been grouped into. And similarly if they have been into multiple task, they simple graphs can not handle this. So bipartite graph having computing nodes on left and tasks on right is required for this problem.

(c) (4 points) Consider a city where all roads can be used in both directions, and where we always have intersections of 4 roads (we have a roundabout whenever there is an intersection). The mayor would like to make all roads one way. Could you propose a way to do that, so that in each roundabout, we have exactly the same number of one way roads entering and exiting?

- vertices: Each intersection is a vertex.
- edges: Each road is an edge.
- problem it maps to: Finding an Eulerian circuit.
- mapping details: Each vertex in the graph has a degree of 4, and hence an Eulerian circuit exists. Finding an Eulerian circuit corresponds to a particular assignment of direction for each edge. Furthermore, each vertex would have 2 incoming edges and 2 outgoing edges corresponding to the Eulerian circuit.

  Common Mistakes:

  – Hamiltonian cycles based approach do not work here because they might not exists and we are okay to visit a vertex twice as long as we give every road (edge) a direction.

Problem 3 (8 points) The following two questions are not related.

(a) (4 points) Let T be a tree on n vertices that has no vertex of degree 2. Show that T has more than $n/2$ leaves. Suppose there are L leaves in the tree. Since the total number of nodes is n, the tree will have $n-L$ non-leaf nodes. In this tree all leaves have degree 1 and other non-leaf nodes will have degree at least 3, as there are no nodes of degree 2.

Thus, the sum of all the degrees is,

$$\sum_{i=1}^{n} d_i \geq L + 3(n - L).$$

Also, the sum of all the degrees is equal to two times the number of edges, and the

number of edges in a tree with n nodes are $n-1$, we will have:

$$\begin{aligned} 2(n-1) &= 2|\mathcal{E}| \\ &= \sum_{i=1}^{n} d_i \\ &\geq L + 3(n-L) \\ \implies 2n-2 &\geq L + 3n - 3L \\ \implies 2L &\geq n+1 \\ \implies L &> \frac{n}{2}. \end{aligned}$$

Common Mistakes:

- Writing two or three examples of such graphs and showing that in those examples we have more than $n/2$ leaves does not solve the problem. We need to show that no matter what tree we have, we will always have more than $n/2$ leaves.
- Mathematical Induction does not work here. Because a tree of this property need not to come from a tree with similar property with one less vertex. Also, making a tree with $n+1$ nodes from a tree of n vertices by adding one extra node does not capture general trees with $n+1$ vertices. So that is also wrong.
- Saying that a node will have more than or equal to two children, does not directly prove that number of leaves are more than $n/2$. It just says that number of nodes in next level is more than number of internal nodes of previous level.
- Saying number of leaves are exponentially increasing or rapidly increasing does not prove that they will be more than $n/2$.
- A root with degree one is also a leaf node.

(b) (4 points) Consider a bipartite graph that has an equal number of vertices on each side, and is a tree. Prove that we can find at most one perfect matching.

Solution 1: Consider a leaf of this tree. Since it is connected to only one node, in the perfect matching we will have to pick this edge. Now, we remove these two nodes from the tree as this is the only possible match for these two nodes, and pick another leaf node from the remaining forest. Note that remaining graph will not have cycles as we are only removing some nodes from the original tree. Continuing this process can result in exactly one matching. Note that, after removing a pair of nodes, the resulting

graph can become such that there might not exist any matching at all. Therefore, at most one matching could exist.

Solution 2: If they are two perfect matchings, then you pick all the edges which are not common and then alternating between them you can get a cycle which violates the property that we have a tree.

Common Mistakes:

- Two matchings need not to be disjoint. Only a difference of one edge will make them different.
- Two matching differing in one edge in a bipartite graph can not always be made into two matching differing everywhere.