ACM Summer School 2019

Graph Theory and Graph Algorithms

Problem sheet

Date: 19.06.2019 **Week:** 1 **Day:** 3

Topics: Shortest Paths, Matchings, BFS/DFS.

1. How would you modify BFS to find the shorest path in a graph in which all edge weights are positive integers?

2. You are given a set of cities along with the pattern of the highways between them, in the form of an undirected graph $G = (V, E)$. Each stretch of a highway $e \in E$ connects two cities and you know its length l_e in kilometers. You want to get from city s to city t. However, there is a problem, your car can hold enough gas to cover L kilometers. There are gas stations in each city but not in between them. Therefore you can take a route if every one of its edges has length $\leq L$.

   (a) Given this limitation on your car's fuel tank capacity, show how to determine in linear time whether there is a feasible route from s to t.

   (b) You are now planning to buy a new car, you want to know the minimum fuel tank capacity needed to travel from s to t. Give an $O((n + m)logn))$ time algorithm for the problem.

3. We are given a directed graph $G(V, E)$ on which each edge $(u, v) \in E$ has an associated value $r(u, v)$, which is a real number in the range $0 \leq r(u, v) \leq 1$ that represents the reliability of a communication channel from vertex u to vertex v. We interpret $r(u, v)$ as the probability that the channel from u to v will not fail, and we assume that these probabilities are independent. Give an efficient algorithm to find the most reliable path between two given vertices.
   Hint: Convert the problem to a shortest path problem on the same graph with modified edge weights.

4. The longest path problem is the problem of finding a simple path of maximal length in a graph; in other words, among all possible simple paths in the graph, the problem is to find the longest one. For an unweighted graph, it suffices to find the longest path in terms of the number of edges; for a weighted graph, one must use the edge weights instead. The problem, unfortunately does not have optimal substructure property. Fortunately, the longest path in a DAG does have optimal substructure, which allows us to solve in polynomial time. Design an algorithm for finding the longest path in a DAG. Suggested reading: http://www.mathcs.emory.edu/ cheung/Courses/171/Syllabus/11-Graph/Docs/longest-path-in-dag.pdf

5. Give a linear time algorithm to find a maximum matching in a tree.

6. Consider the game *Slither* by W. Anderson. Given a simple undirected graph $G = (V, E)$, two players take turns in selecting an edge e with the following rule: the edge e was not yet selected and the set of selected edges including the edge e represent a simple path. The player who cannot choose an edge according to the rules loses.

We will say that a player-i has a winning strategy if player-i can force a win for himself irrespective of the moves of the player-j. Show that if G admits a perfect matching, then player-1 has a winning strategy.

7. Design an algorithm to check if the vertex connectivity of the graph is 2. Recall that the vertex connectivity is the minimum number of vertices to remove so that some pair of vertices get disconnected.

8. Consider the following variant of the Dijkstra's algorithm: The distance estimate of all the vertices is changed in each iteration. Recall that the distance estimate of a vertex v_j is $relax[v_j] = \min_u(d(s,u) + w(u,v_j))$. The minimum is taken over all vertices in the graph G. After how many iterations can this update step terminate? How does the algorithm work when there is a negative weight cycle, and when there is no negative weight cycle?

9. Give a linear time algorithm to find a pair of vertices whose distance is the the largest among all vertex pairs in a given tree.