

TUTORIAL SHEET 13

1. You are given a directed graph $G = (V, E)$. A vertex v is called nice if there is a directed path from v to each of the other vertices in G . Suppose we run DFS on G starting from some vertex s and let w be the vertex with the highest finish time $F[w]$. Prove that if G has a nice vertex, then w is a nice vertex.
2. Let G be a directed graph. Let C_1, C_2, \dots, C_k denote the strongly connected components of G (each C_i is a subset of vertices). We now construct another directed graph H as follows: there are k vertices in H , denoted v_1, \dots, v_k . There is a directed edge (v_i, v_j) in H if and only if there is an edge in G from a vertex in C_i to a vertex in C_j . Prove that H is a DAG. Give a linear time algorithm to construct H .
3. You are given a directed acyclic graph. Each edge e has a length ℓ_e . The length of a path is the total length of the edges in it. Given a vertex s , give a linear time algorithm to find the longest path that starts from s .
4. A bipartite graph is an undirected graph whose vertex set can be partitioned into two sets A and B such that each edge in the graph goes between a vertex in A and a vertex in B (no edges exist between vertices in the same set). Give an $O(|V| + |E|)$ time algorithm that takes an input graph represented as an adjacency list, and decides if the graph is bipartite.
5. You are given an undirected graph G in the adjacency list data-structure. We would like to find the shortest cycle in the graph (length of a cycle is defined as the number of edges in it).
 - You try the following DFS based algorithm: construct the DFS tree (assume it is connected, otherwise we can run this on each connected component). For each back edge, find the length of the corresponding cycle and report the shortest cycle. Give an example where this algorithm fails to find the shortest cycle.
 - Give a BFS based algorithm to find the shortest cycle. What is the running time of your algorithm?
6. Given an undirected graph G and two vertices s and t , we know that BFS can be used to find the shortest path from s to t . We say that there is a *unique* shortest path from s to t if there is exactly one shortest path from s to t . How would you modify BFS to check if there is a unique shortest path from s to t ? The running time of your algorithm should still be linear.
7. We say that a vertex v in a connected undirected graph G is a cut vertex if removal of v from G disconnects the graph. Suppose we run DFS from a vertex s and let T be

the DFS tree. Let the children of v in T be w_1, \dots, w_r . Show that v is not a cut vertex iff the following condition is satisfied for each child w_i of v : there is an edge from a descendant of w_i to an ancestor of v in the graph G .