Week 5 Applied Sheet

Objectives: The applied sessions, in general, give practice in problem solving, in analysis of algorithms and data structures, and in mathematics and logic useful in the above.

Instructions to the class: You should actively participate in the class.

Instructions to Tutors: The purpose of the applied class is not to solve the practical exercises! The purpose is to check answers, and to discuss particular sticking points, not to simply make answers available.

Supplementary problems: The supplementary problems provide additional practice for you to complete after your applied class, or as pre-exam revision. Problems that are marked as **(Advanced)** difficulty are beyond the difficulty that you would be expected to complete in the exam, but are nonetheless useful practice problems as they will teach you skills and concepts that you can apply to other problems.

Problems

Problem 1. Devise an algorithm for determining whether a given undirected graph is two-colourable. A graph is two-colourable if each vertex can be assigned a colour, black or white, such that no two adjacent vertices are the same colour. Your algorithm should run in $O(V+E)$ time. Write pseudocode for your algorithm.

Problem 2. Describe an algorithm for counting the number of valid two colourings of a given undirected graph.

Problem 3. This problem is about cycle finding as discussed in the course notes.

(a) Explain using an example why the algorithm given for finding cycles in an undirected graph does not work when applied to a directed graph.

(b) Describe an algorithm based on depth-first search that determines whether a given directed graph contains any cycles. Your algorithm should run in $O(V+E)$ time. Write pseudocode for your algorithm.

Problem 4. Describe an algorithm for finding the **shortest** cycle in an unweighted, directed graph. A shortest cycle is a cycle with the minimum possible number of edges. You may need more than linear time to solve this one. Write pseudocode for your algorithm.

Problem 5. In this question we consider a variant of the single-source shortest path problem, the *multi-source shortest path* problem. In this problem, we are given an unweighted graph and a set of many source vertices. We wish to find for every vertex v in the graph, the minimum distance to any one of the source vertices. Formally, given the sources $s_1, s_2, ...s_k$, we wish to find for every vertex v

$$\mathrm{d}[v] = \min_{1\le i\le k} \mathrm{dist}(v, s_i).$$

Describe how to solve this problem using a modification to breadth-first search. Your algorithm should run in $O(V+E)$ time.

Problem 6. Write pseudocode for an algorithm that counts the number of connected components in an undirected graph that are cycles. A cycle is a non-empty sequence of edges $(u_1, u_2), (u_2, u_3), ...(u_k, u_1)$ such that u_1, u_2, $\ldots$, u_k are all distinct vertices.

Supplementary Problems

Problem 7. Write pseudocode for a non-recursive implementation of depth-first search that uses a stack instead of recursion.

Problem 8. Argue that the algorithm given in the course notes for detecting whether an undirected graph contains a cycle actually runs in $O(V)$ time, not $O(V+E)$ time, i.e. its complexity is independent of $|E|$.

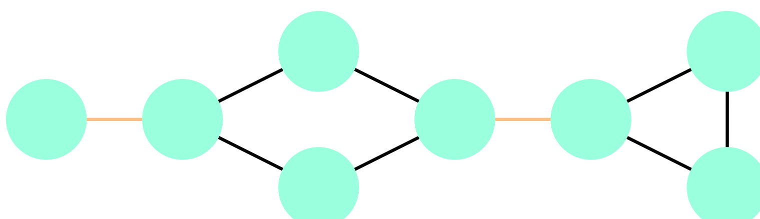
Problem 9. Consider a directed acyclic graph representing the hierarchical structure of n employees at a company. Each employee may have one or many employees as their superior. The company has decided to give raises to m of the top employees. Unfortunately, you are not sure exactly how the company decides who is considered the top employees, but you do know for sure that a person will not receive a raise unless all of their superiors do.

Describe an algorithm that given the company DAG and the value of m, determines which employees are guaranteed to receive a raise, and which are guaranteed to not receive a raise. Your algorithm should run in $O(V^2+VE)$ time.

Problem 10. A Hamiltonian path in a graph $G=(V,E)$ is a path in G that visits every vertex $v \in V$ exactly once. On general graphs, computing Hamiltonian paths is NP-Hard. Describe an algorithm that finds a Hamiltonian path in a directed acyclic graph in $O(V+E)$ time or reports that one does not exist.

Problem 11. (Advanced) Given a directed graph G in adjacency matrix form, determine whether it contains a "*universal sink*" vertex. A universal sink is a vertex v with in-degree $|V|-1$ and out-degree 0, ie. a vertex such that every other vertex in the graph has an edge to v, but v has no edge to any other vertex. **Your algorithm should run in $O(V)$ time.** Note that this means that you cannot read the entire adjacency matrix and meet the required complexity.

Problem 12. (Advanced) This problem is about determining another interesting property of a graph G, namely its *bridges*. A bridge is an edge that if removed from the graph would increase the number of connected components in the graph. For example, in the following graph, the bridges are highlighted:



(a) Prove that an edge is a bridge if and only if it does not lie on any simple cycle in G.

(b) Suppose that we number the vertices of G in the order that they are visited by depth-first search. Denote this quantity by $\texttt{dfs\_ord}[v]$ for each vertex v. Define the following quantity $\texttt{low\_link}[v]$ for each vertex v such that

$$\texttt{low\_link}[v] = \min \begin{cases} \texttt{dfs\_ord}[v], \\ \texttt{dfs\_ord}[u] : \text{for any vertex } u \text{ reachable from } v \text{ via unused edges after visiting } v \end{cases}$$

Explain how the quantity `low_link` can be computed in $O(V+E)$ time.

(c) Explain how the quantities `dfs_num` and `low_link` can be used to determine which edges are bridges.

(d) Write pseudocode that uses the above facts to implement an algorithm based on depth-first search for determining the bridges of a given undirected graph in $O(V+E)$ time.