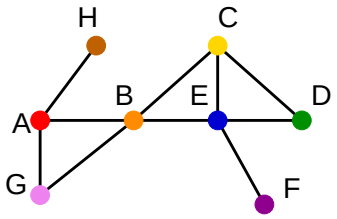
Problem Set 7

Data Structures and Algorithms, Fall 2023

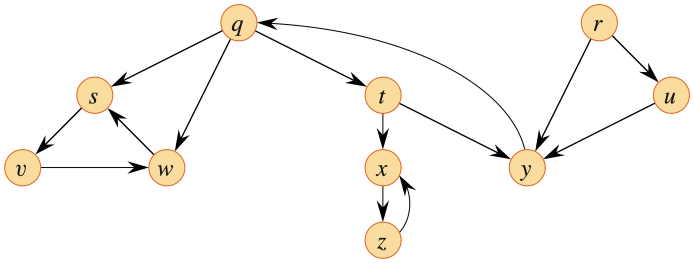
Due: November 23, in class or mail to `ai-dsalg-ps@chaodong.me`.

Problem 1

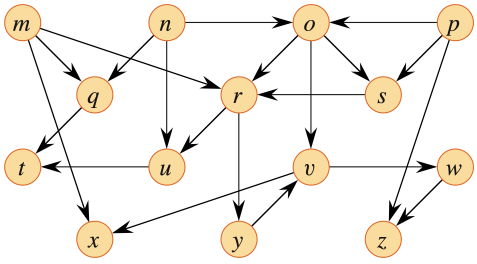
(a) Perform a BFS traversal on the following graph. Whenever there is a choice of vertices, pick the one that is alphabetically first. Show the *distance* value and the *parent* value for each vertex.



(b) Perform a DFS traversal on the following graph. Whenever there is a choice of vertices, pick the one that is alphabetically first. Show the classification of edges, and give the start/finish times of vertices.



(c) Run the DFS-based topological sorting algorithm on the following graph. Whenever there is a choice of vertices, pick the one that is alphabetically first. Show the final ordering.



Bonus Problem

The *diameter* of a tree $T = (V, E)$ is defined as $\max\{dist(u, v) : u, v \in V\}$, that is, the largest of all shortest-path distances in the tree. Devise an algorithm with $O(|V| + |E|)$ runtime to compute the diameter of a tree. You need to *prove* the correctness of your algorithm.

Problem 2

(a) Prove or disprove: There exists a directed graph $G = (V, E)$ in which some vertex $u \in V$ has both incoming and outgoing edges, such that some execution of DFS on G results in a DFS forest in which u ends up in a tree alone.

(b) Prove or disprove: Given an undirected graph $G = (V, E)$ using adjacency list representation, there exists an algorithm that can determine whether G contains a cycle in $O(|V|)$ time. That is, the runtime of the algorithm is independent of $|E|$.

(c) Read Section 22.5 of CLRS, pay attention to the algorithm for computing SCC as it is slightly different from the one introduced in class. Now, consider a modified version of the SCC algorithm introduced in CLRS: the modified algorithm uses the original (instead of the transpose) graph in the second depth-first search and scans the vertices in order of increasing finishing times. Is the modified algorithm correct? You need to prove your answer.

Problem Group I [Pick any two of the following three problems, but you get extra credit if you solve all of them.]

Problem I.1

Let $G = (V, E)$ be a connected undirected graph. Suppose we start with two coins on two arbitrarily chosen vertices of G, and we want to move the coins so that they lie on the same vertex using as few moves as possible. At every step, *each* coin *must* move to an adjacent vertex.

(a) Describe an algorithm to compute the minimum number of steps to reach a configuration where both coins are on the same vertex, or to report correctly that no such configuration is reachable.

(b) Now suppose there are three coins. Describe an algorithm to compute the minimum number of steps to reach a configuration where all three coins are on the same vertex, or to report correctly that no such configuration is reachable.

(c) Finally, suppose there are forty-two coins. Describe an algorithm to determine whether it is possible to move all 42 coins to the same vertex. Again, every coin must move at every step.

For full credit, each of your algorithm should run in $O(|V| + |E|)$ time.

Problem I.2

A *key vertex* of a connected undirected graph $G = (V, E)$ is a vertex whose deletion disconnects G.

(a) Devise an $O(|V| + |E|)$ time algorithm for finding a vertex that is *not* a key vertex.

(b) Devise an $O(|V| + |E|)$ time algorithm that finds a deletion order for the $|V|$ vertices such that no deletion disconnects the graph.

Problem I.3

In the 2SAT problem, you are given a set of *clauses*, where each clause is the disjunction (OR) of two *literals* (a literal is a Boolean variable or the negation of a Boolean variable). You are looking for a way to assign a value `true` or `false` to each of the variables so that all clauses are satisfied — that is, there is at least one true literal in each clause. For example, here's an instance of 2SAT:

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (\overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee x_4)$$

This instance has a satisfying assignment: set x_1, x_2, x_3, and x_4 to `true`, `false`, `false`, and `true`, respectively. The purpose of this problem is to lead you to a way of solving 2SAT efficiently by reducing it to the problem of finding the strongly connected components of a directed graph. Given an instance I of 2SAT with n variables and m clauses, construct a directed graph $G_I = (V, E)$ as follows.

- G_I has $2n$ nodes, one for each variable and its negation.
- G_I has $2m$ edges: for each clause $(\alpha \vee \beta)$ of I (where α, β are literals), G_I has an edge from the negation of α to β, and one from the negation of β to α.

(a) Carry out this construction for the instance of 2SAT given above.

(b) Prove that if G_I has a strongly connected component containing both x and $\overline{x}$ for some variable x, then I has no satisfying assignment.

(c) Now prove the converse of (b): if none of G_I's strongly connected components contain both a literal and its negation, then the instance I must be satisfiable.

(d) Describe a linear-time (that is, $O(|V| + |E|)$ time) algorithm for solving 2SAT.

Problem Group II [Pick any two of the following three problems, but you get extra credit if you solve all of them.]

Problem II.1

(a) Prove or disprove: Consider an undirected connected graph G, if an edge of G is contained in some minimum spanning tree of G, then it is a light edge crossing some cut of G.

(b) Prove or disprove: Consider an undirected connected graph G, if for every cut of G there is a unique light edge crossing the cut, then G has a unique minimum spanning tree.

(c) Prove or disprove: Consider an undirected connected graph G, if in G there are two edges with equal weights, then G has at least two distinct minimum spanning trees.

Problem II.2

Consider the problem of finding a minimum-weight connected sub-graph $T = (V' \subseteq V, E' \subseteq E)$ of a weighted connected graph $G = (V, E)$. The weight of T is the sum of all the edge weights in T.

(a) Why is this problem not just the minimum spanning tree problem?

(b) Given G, devise an efficient algorithm to compute the minimum-weight connected subset T.

Problem II.3

Let $G = (V, E)$ be an undirected connected graph whose weight function is $w : E \to \mathbb{R}$. Assume all edge weights are distinct. We define a second-best minimum spanning tree as follows. Let $\mathcal{T}$ be the set of all spanning trees of G, and let T be a minimum spanning tree of G. Then a second-best minimum spanning tree is a spanning tree T' such that $w(T') = \min\{w(T'') : T'' \in \mathcal{T} - \{T\}\}$.

(a) Show that the minimum spanning tree is unique, but that the second-best minimum spanning tree need not be unique.

(b) Let T be the minimum spanning tree of G. Prove that G contains some edge $(u, v) \in E(T)$ and some edge $(x, y) \notin E(T)$ such that $(E(T) - \{(u, v)\}) \cup \{(x, y)\}$ is the edge set of a second-best minimum spanning tree of G.

(c) Now let T be any spanning tree of G and, for any two vertices $u, v \in V$, let $max[u, v]$ denote an edge of maximum weight on the unique simple path between u and v in T. Devise an $O(|V|^2)$-time algorithm that, given T, computes $max[u, v]$ for all $u, v \in V$.

(d) Devise an $O(|V|^2)$-time algorithm that computes a second-best minimum spanning tree of G.