Practice Problem Set – DFS, MST, Shortest Path

1. Describe an algorithm to compute the reversal rev(G) of a directed graph in O(V + E) time.
2. Prove that for every directed graph G, the strong component graph scc(G) is acyclic.
3. Prove that scc(rev(G)) = rev(scc(G)) for every directed graph G.
4. Fix an arbitrary directed graph G. For any vertex v of G, let S(v) denote the strong component of G that contains v. For all vertices u and v of G, prove that u can reach v in G if and only if S(u) can reach S(v) in scc(G).
5. Suppose we are given a directed acyclic graph G with a unique source s and a unique sink t. A vertex $v \notin \{s, t\}$ is called an (s, t)-cut vertex if every path from s to t passes through v, or equivalently, if deleting v makes t unreachable from s. Describe and analyze an algorithm to find every (s, t)-cut vertex in G.

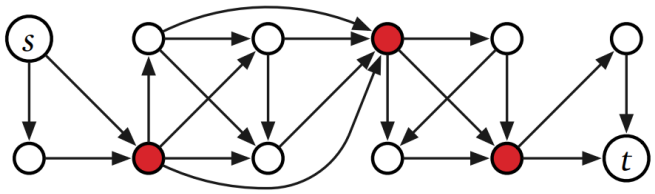

Figure 1: A directed acyclic graph with three (s, t)-cut vertices.

6. The transitive closure G^T of a directed graph G is a directed graph with the same vertices as G, that contains any edge u→v if and only if there is a directed path from u to v in G. A transitive reduction of G is a graph with the smallest possible number of edges whose transitive closure is G^T. The same graph may have several transitive reductions.
   a) Describe an efficient algorithm to compute the transitive closure of a given directed graph.
   b) Prove that a directed graph G has a unique transitive reduction if and only if G is acyclic.
   c) Describe an efficient algorithm to compute a transitive reduction of a given directed graph.

7. You have a collection of n lock-boxes and m gold keys. Each key unlocks at most one box. However, each box might be unlocked by one key, by multiple keys, or by no keys at all. There are only two ways to open each box once it is locked: Unlock it properly (which requires having one matching key in your hand), or smash it to bits with a hammer.

   Your baby brother, who loves playing with shiny objects, has somehow managed to lock all your keys inside the boxes! Luckily, your home security system recorded everything, so you know exactly which keys (if any) are inside each box. You need to get all the keys back out of the boxes, because they are made of gold. Clearly you have to smash at least one box.

   a) Your baby brother has found the hammer and is eagerly eyeing one of the boxes. Describe and analyze an algorithm to determine if it is possible to retrieve all the keys without smashing any box except the one your brother has chosen.

   b) Describe and analyze an algorithm to compute the minimum number of boxes that must be smashed to retrieve all the keys.

8. Suppose we are given both an undirected graph G with weighted edges and a minimum spanning tree T of G.

   a) Describe an algorithm to update the minimum spanning tree when the weight of a single edge e is decreased.

   b) Describe an algorithm to update the minimum spanning tree when the weight of a single edge e is increased.

   In both cases, the input to your algorithm is the edge e and its new weight; your algorithms should modify T so that it is still a minimum spanning tree. [Hint: Consider the cases $e \in T$ and $e \notin T$ separately.]

9. Describe and analyze an algorithm to find the second smallest spanning tree of a given graph G, that is, the spanning tree of G with smallest total weight except for the minimum spanning tree.

10. Describe and analyze an efficient algorithm to compute, given a weighted undirected graph G and an integer k, the k spanning trees of G with smallest weight.

11. Suppose we are given a directed graph G with weighted edges and two vertices s and t.

    a) Describe and analyze an algorithm to find the shortest path from s to t when exactly one edge in G has negative weight. [Hint: Modify Dijkstra's algorithm. Or don't.]

    b) Describe and analyze an algorithm to find the shortest path from s to t when exactly k edges in G have negative weight. How does the running time of your algorithm depend on k?

12. For any edge e in any graph G, let $G \setminus e$ denote the graph obtained by deleting e from G. Suppose we are given a graph G and two vertices s and t. The replacement paths problem asks us to compute the shortest-path distance from s to t in $G \setminus e$, for every edge e of G. The output is an array of E distances, one for each edge of G.

a) Suppose G is a directed graph, and the shortest path from vertex s to vertex t passes through every vertex of G. Describe an algorithm to solve this special case of the replacement paths problem in O(E log V) time.

b) Describe an algorithm to solve the replacement paths problem for arbitrary undirected graphs in O(E log V) time.

In both subproblems, you may assume that all edge weights are non-negative. [Hint: If we delete an edge of the original shortest path, how do the old and new shortest paths overlap?]

13. Suppose you are given a directed graph G in which every edge has negative weight, and a source vertex s. Describe and analyze an efficient algorithm that computes the shortest-path distances from s to every other vertex in G. Specifically, for every vertex t:

    - If t is not reachable from s, your algorithm should report dist(t) = ∞.
    - If G has a cycle that is reachable from s, and t is reachable from that cycle, then the shortest-path distance from s to t is not well-defined, because there are paths (formally, walks) from s to t of arbitrarily large negative length. In this case, your algorithm should report dist(t) = -∞.
    - If neither of the two previous conditions applies, your algorithm should report the correct shortest-path distance from s to t.

14. Describe and analyze an algorithm to determine the number of shortest paths from a source vertex s to a target vertex t in an arbitrary directed graph G with weighted edges. You may assume that all edge weights are positive and that all necessary arithmetic operations can be performed in O(1) time. [Hint: Compute shortest path distances from s to every other vertex. Throw away all edges that cannot be part of a shortest path from s to another vertex. What's left?]

15. There are n galaxies connected by m intergalactic teleport-ways. Each teleport-way joins two galaxies and can be traversed in both directions. Also, each teleport-way e has an associated cost of c(e) dollars, where c(e) is a positive integer. A teleport-way can be used multiple times, but the toll must be paid every time it is used.

    Judy wants to travel from galaxy s to galaxy t as cheaply as possible. However, she wants the total cost to be a multiple of five dollars, because carrying small change is not pleasant either.

    a) Describe and analyze an algorithm to compute the minimum total cost of traveling from galaxy s to galaxy t, subject to the restriction that the total cost is a multiple of five dollars.

    b) Solve part (a), but now assume that Judy has a coupon that allows her to use exactly one teleport-way for free.

16. After moving to a new city, you decide to choose a walking route from your home to your new office. To get a good daily workout, your route must consist of an uphill path

(for exercise) followed by a downhill path (to cool down), or just an uphill path, or just a downhill path. (You'll walk the same path home, so you'll get exercise one way or the other.) But you also want the shortest path that satisfies these conditions, so that you actually get to work on time.

Your input consists of an undirected graph G, whose vertices represent intersections and whose edges represent road segments, along with a start vertex s and a target vertex t. Every vertex v has an associated value h(v), which is the height of that intersection above sea level, and each edge uv has an associated value ℓ(uv), which is the length of that road segment.

a) Describe and analyze an algorithm to find the shortest uphill–downhill walk from s to t. Assume all vertex heights are distinct.

b) and analyze an algorithm to find the shortest "uphill then downhill" walk from s to t; you may use flat edges in both the "uphill" and "downhill" portions of your walk.

c) Finally, suppose you discover that there is no path from s to t with the structure you want. Describe an algorithm to find a path from s to t that alternates between "uphill" and "downhill" subpaths as few times as possible, and has minimum length among all such paths.

17. The first morning after returning from a glorious spring break, Alice wakes to discover that her car won't start, so she has to get to her classes at Sham-Poobanana University by public transit. She has a complete transit schedule for Poobanana County. The bus routes are represented in the schedule by a directed graph G, whose vertices represent bus stops and whose edges represent bus routes between those stops. For each edge u$\to$v, the schedule records three positive real numbers:

- ℓ(u$\to$v) is the length of the bus ride from stop u to stop v (in minutes)
- f(u$\to$v) is the first time (in minutes past 12am) that a bus leaves stop u for stop v.
- Δ(u$\to$v) is the time between successive departures from stop u to stop v (in minutes).

Thus, the first bus for this route leaves u at time f(u$\to$v) and arrives at v at time(u$\to$v)+ℓ(u$\to$v), the second bus leaves u at time f(u$\to$v)+Δ(u$\to$v) and arrives at v at time f(u$\to$v)+Δ(u$\to$v)+ℓ(u$\to$v), the third bus leaves u at time f(u*to*v)+2$\cdot\Delta$(u$\to$v) and arrives at v at time f(u$\to$v)+2$\cdot\Delta$(u$\to$v)+ℓ(u$\to$v), and so on.

Alice wants to leaves from stop s (her home) at a certain time and arrive at stop t (The See-Bull Center) as quickly as possible. If Alice arrives at a stop on one bus at the exact time that another bus is scheduled to leave, she can catch the second bus. Because she's a student at SPU, Alice can ride the bus for free, so she doesn't care how many times she has to change buses.

Describe and analyze an algorithm to find the earliest time Alice can reach her destination. Your input consists of the directed graph G = (V, E), the vertices s and t, the values ℓ(e), f(e), Δ(e) for each edge e $\in$ E, and Alice's starting time (in minutes past 12am).

[Hint: In this rare instance, it may be easier to modify the algorithm, instead of modifying the input graph.]