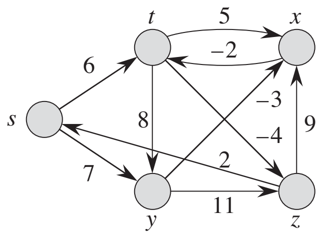
Problem Set 12

Data Structures and Algorithms, Fall 2021

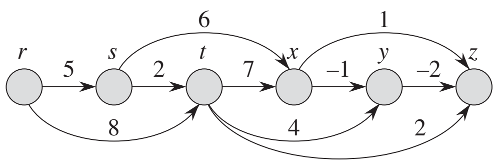
Due: December 24, in class.

Problem 1

(a) Run the Bellman-Ford algorithm on the following directed graph, using vertex z as the source. In each pass, `Update` edges in the order $(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$, and show the $dist$ and $parent$ values after each pass.

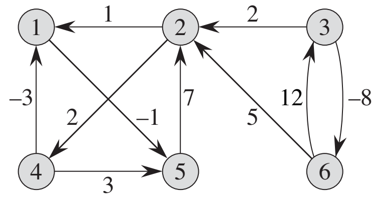


(b) Run the `DAGSSSP` algorithm on the following directed graph, using vertex r as the source. You need to show the $dist$ and $parent$ values after each iteration of the outer loop.



Problem 2

(a) Run the Floyd-Warshall algorithm on the following weighted, directed graph. Show the $n \times n$ matrix $dist[\cdot, \cdot, r]$ after each outermost loop. (That, for each $r \in [1, n]$, show the $dist$ matrix.)
(b) Use Johnson's algorithm to find the shortest paths between all pairs of vertices in the following graph. Show the final shortest distance value between all pairs of vertices, the h value for each vertex, and the updated edge weight $\hat{w}(u, v)$ for each edge (u, v).



Problem 3

The PERT chart formulation discussed in class is somewhat unnatural. In a more natural structure, vertices would represent jobs and edges would represent sequencing constraints; that is, edge (u, v) would indicate that job u must be performed before job v. We would then assign weights to vertices instead of edges. In this problem, you are given a DAG graph $G = (V, E)$ with a weight function $w : V \to \mathbb{R}^+$. Assume G has a unique source s and a unique sink t. For each of the following tasks, design an $O(|V| + |E|)$ time algorithm:

(a) Compute the total number of paths in G.

(b) For each vertex $v \in V$, the earliest starting time of the job represented by v, assuming the job represented by s starts at time 0.

(c) For each vertex $v \in V$, the latest starting time of the job represented by v, without affecting the project's total duration.

Problem 4

Suppose you are given a directed graph G in which *every edge has negative weight*, and a source vertex s. Design an 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 $dist(s, t) = \infty$.
- 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 from s to t of arbitrarily large negative length. In this case, your algorithm should report $dist(s, t) = -\infty$.
- If neither of the two previous conditions applies, your algorithm should report the correct shortest-path distance from s to t.

Try to devise the fastest algorithm you can come up with. You need to describe your algorithm, briefly argue its correctness, and analyze its running time.

Problem 5

You are the owner of a steamship that can ply between a group of port cities V. You make money at each port: a visit to city i earns you a profit of p_i dollars. Meanwhile, the transportation cost from port i to port j is $c_{ij} > 0$. You want to find a cyclic route in which the ratio of profit to cost is maximized. To this end, consider a directed graph $G = (V, E)$ whose nodes are ports, and which has edges between each pair of ports. For any cycle C in this graph, the profit-to-cost ratio is

$$r(C) = \frac{\sum_{(i,j)\in C} p_j}{\sum_{(i,j)\in C} c_{ij}}$$

Let r^* be the maximum ratio achievable by a simple cycle. One way to determine r^* is by binary search: by first guessing some ratio r, and then testing whether it is too large or too small. Consider any positive $r > 0$. Give each edge (i, j) a weight of $w_{ij} = r \cdot c_{ij} - p_j$.

(a) Prove that if there is a cycle of negative weight, then $r < r^*$.

(b) Prove that if all cycles in the graph have strictly positive weight, then $r > r^*$.

(c) Design an algorithm that takes as input a desired accuracy $\epsilon > 0$ and returns a simple cycle C for which $r(C) \geq r^* - \epsilon$. Remember to briefly argue the correctness of your algorithm and analyze its running time in terms of $|V|$, ϵ, and $R = \max_{(i,j)\in E}(p_j/c_{ij})$.

Problem 6

You've just accepted a job from Elon Musk, delivering burritos from San Francisco to New York City. You get to drive a Burrito-Delivery Vehicle through Elon's new Transcontinental Underground Burrito-Delivery Tube, which runs in a direct line between these two cities. Your Burrito-Delivery Vehicle runs on single-use batteries, which must be replaced after at most 100 miles. The actual fuel is virtually free, but the batteries are expensive and fragile, and therefore must be installed only by official members of the Transcontinental Underground Burrito-Delivery Vehicle Battery-Replacement Technicians' Union. Thus, even if you replace your battery early, you must still pay full price for each new battery to be installed. Moreover, your Vehicle is too small to carry more than one battery at a time.

There are several fueling stations along the Tube; each station charges a different price for installing a new battery. Before you start your trip, you carefully print the Wikipedia page listing the locations and prices of every fueling station along the Tube. Given this information, how do you decide the best places to stop for fuel?

More formally, suppose you are given two arrays $D[1 \cdot n]$ and $C[1 \cdots n]$, where $D[i]$ is the distance from the start of the Tube to the i-th station, and $C[i]$ is the cost to replace your battery at the i-th station. Assume that your trip starts and ends at fueling stations (so $D[1] = 0$ and $D[n]$ is the total length of your trip), and that your car starts with an empty battery (so you must install a new battery at station 1).

(a) Describe a greedy algorithm to find the minimum number of refueling stops needed to complete your trip. You do not need to prove that your algorithm is correct, but you need to analyze its running time.

(b) But what you really want to minimize is the total cost of travel. Show that your greedy algorithm in part (a) does not produce an optimal solution when extended to this setting.

(c) Describe an algorithm to compute the locations of the fuel stations you should stop at to minimize the total cost of travel. You do not need to prove that your algorithm is correct, but you need to analyze its running time.

Problem 7

Assume that you have a subroutine `IsWord` that takes an array of characters as input and returns `True` iff that string is a "word". Design an algorithm that, given an array $A[1 \cdots n]$ of characters, compute the number of partitions of A into words. You need to give the pseudocode of your algorithm, and analyze the runtime of your algorithm by bounding the number of calls to `IsWord`. For example, given the string ARTISTOIL, your algorithm should return 2, for the partitions ARTIST·OIL and ART·IS·TOIL.