Introduction to Algorithms: 6.006
Massachusetts Institute of Technology — April 9, 2021
Instructors: Mauricio Karchmer, Anand Natarajan, Nir Shavit — Problem Set 7

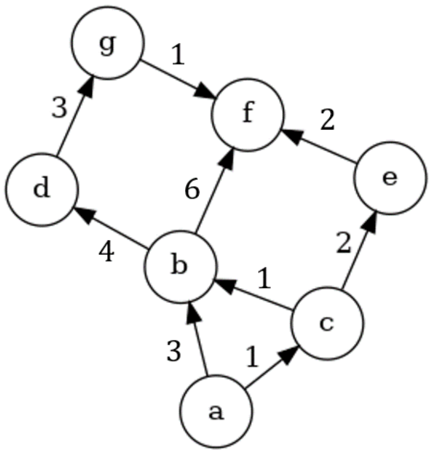
Problem Set 7

Instructions
All parts are due on April 15, 2021 at 10PM. Aim for concise solutions; convoluted and obtuse descriptions might receive low marks, even when they are correct. Solutions should be submitted on Gradescope, and any code should be submitted for automated checking on `https://alg.mit.edu`.

Please take note of the new collaboration policy as described in `https://canvas.mit.edu/courses/7477/discussion_topics/76837`. Unless explicitly stated otherwise, to get full credit you must prove the runtime and correctness of any algorithm you give.

Problem 7-1. [20 points] **Shortest Paths Practice**

Consider the weighted, directed, acyclic graph $G = (V, E, w)$ below, which has nonnegative edge weights.



(1) [14 points] Run both DAG relaxation (i.e. the optimization of Bellman-Ford for DAGs) and Dijkstra on graph G, starting from vertex a. For each algorithm, give a list of the edges that the algorithm relaxes, in the order it relaxes them (**relaxation order**). If there is any ambiguity in which node to process next, use alphabetical order to break the tie.

Solution:

Dijkstra: $[(a, b), (a, c), (c, b), (c, e), (b, d), (b, f), (e, f), (d, g), (g, f)]$

DAG Relaxation: Either of the following is okay:

- $[(a, b), (a, c), (c, b), (c, e), (b, d), (b, f), (d, g), (e, f), (g, f)]$ (alphabetically first topological sort order)

- $[(a,b),(a,c),(c,b),(c,e),(e,f),(b,d),(b,f),(d,g),(g,f)]$ (topological sort order derived from using dfs with alphabetical order at each step)

Rubric:

- 7 for DAG relaxation order
- 7 for Dijkstra relaxation order
- -1 for each mistake (addition, omission, inversion)
- Partial credit may be awarded.

(2) [6 points] List the shortest distance $\delta(a, v)$ for each $v \in V$.

Solution:

v	a	b	c	d	e	f	g
$\delta(a,v)$	0	2	1	6	3	5	9

Rubric:

- 6 for correct distances
- Partial credit may be awarded.

Problem 7-2. [20 points] **Dijkstra with Negative Weights**

Alyssa and Ben are each given an assignment to implement an efficient algorithm to solve the Single-Source Shortest-Path (**SSSP**) problem on some directed weighted graphs that might include negative weights but no negative cycles. Each of Alyssa and Ben is interested in applying their algorithms to different families of graphs.

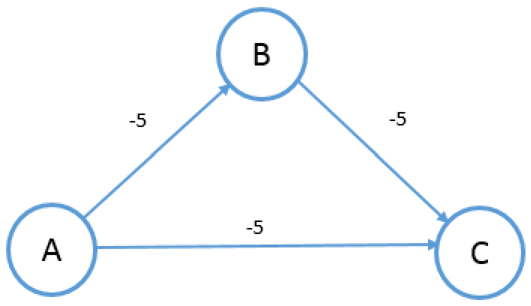
(1) [10 points] Ben just learned that DIJKSTRA doesn't work on general graphs with negative weights. However, after some careful observation, Ben concludes that, for the graphs he is interested in, none of the negative weights are smaller than -10. He then proposes the following graph transformation before applying DIJKSTRA:

For original graph $G = (V, E)$, construct a new graph G' with exactly the same vertices and edges. Then, add 10 to all the edge weights. That is,

$$w'(u, v) = w(u, v) + 10, \ \ \forall (u, v) \in E$$

Ben claims that, after performing this graph transformation and running DIJKSTRA on the new graph, the resulting paths will be shortest paths for the original graph as well. If Ben is correct, briefly explain why this graph transformation works. If Ben is incorrect, provide a counterexample.

Solution: Ben's algorithm is not guaranteed to work. Consider the following counterexample:



Before adding 10 to all edges, the shortest path is $A \to B \to C$. After adding 10, all the edge weights become 5, hence making $A \to C$ the shortest path, which is an incorrect result.

Rubric:

- 2 for a correct answer.
- 8 for a valid counterexample.
- Partial credit may be awarded.

(2) [10 points] Alyssa is also looking at graphs with special properties. In her case, she knows that all her graphs have at most 100 negative edges. Further, she knows that there are no cycles formed by the negative edges. Unfortunately, the runtime offered by Bellman-Ford is not good enough for her. In a moment of brilliance she thinks of a way to adapt Dijsktra's algorithm for her graphs. After initialization, rather than making one pass of Dijkstra's algorithm (the code inside the `while` loop), she figures that making 100 passes will allow her to compute shortest path on her special graphs.

Help Alyssa prove correctness of her algorithm. *Hint:* Try making an inductive argument with the proof presented in lecture as the base case.

Solution: Assume that we are computing distances form a node u.

By induction on k, the number of passes, we prove that, after the k-th pass, all distances realized by paths using at most $(k-1)$ negative edges are computed correctly. That is, if for a node v, there is a shortest path from u to v using at most $(k-1)$ edges, then $d(v) = \delta(v)$. Note that we don't make any claims about nodes for which all shortest paths to it form u contain more than $k-1$ negative edges.

The base case follows the proof of Dijkstra's algorithm, restricting our attention to positive edges.

For the induction step, we assume that all distances with shortest paths using at most $k-1$ negative edges are computed correctly and prove that, after one more pass, the same holds for all distances using k negative edges. Following the proof for Dijkstra's algorithm, we show that, when a vertex v, with a shortest path to u using at most k edges is placed into the set S, $d[v] = \delta[v]$. We prove this by contradiction. Let v be the first node for which the claim fails and let's look at the state just before v is placed into S. Pick a shortest path from u to v using k negative edges. We follow this shortest path in reverse, and stop before we either go to a node in S, or we cross a negative edge. Call this last node y (which can be the same as v) and call the node on the other side of the edge x (which can be the same as u). We have $y \notin S$ and $\delta[y] \leq \delta[v]$ given that all edges from y to v have positive weight. Once again, as in the original proof, we claim that $d[y] = \delta[y]$. We have two cases: i) $x \in S$ and, like in the original proof, we know that the edge (x, y) was relaxed when x was placed into S and so $d[y] = \delta[y]$; (ii) the edge (x, y) has a negative weight and so the path from u to x has at most $(k-1)$ negative edges. By induction, $d[x] = \delta[x]$ and, furthermore, when x was processed in the previous iteration, the edge (x, y) was relaxed and so $d[y] \leq \delta[x] + w(x, y) = \delta[y]$. The proof follows now like in the regular proof: we have that $d[v] \leq d[y] = \delta[y] \leq \delta[v] \leq d[v]$ and so $d[v] = \delta[v]$, a contradiction.

Rubric:

- 2 correct induction assumption.
- 4 for correct base case.
- 4 for correct induction step.
- Partial credit may be awarded.

Problem 7-3. [20 points] **Iakesi**

After dreaming of fantasy worlds, Sam has woken up to find himself transported to the world of Iakesi. Wanting to return to his original world, Sam learns that there is a famous magician in the city of Iris, who may be able to help Sam. After discovering he can now use magic, Sam has decided to become an adventurer in the hopes of meeting this magician. There are n cities and m one-way roads (in this world if you try to walk down a road in the wrong direction then the city at the other end will not let you in) connecting the cities. Sam is currently in the city of Aquilegia.

(1) [10 points] Each road r_{ab} from city a to city b takes d_{ab} days to travel. Thinking he needs to prove himself to the magician to get his help, Sam first wants to help defeat k demon commanders before he meets the magician. every city has either 0 or 1 demon commanders currently terrorizing it. In $O(km + knlog(kn))$ time, determine the quickest route that Sam can take to defeat k commanders and then end in Iris.

Solution: For each city a, build k nodes labelled $(a, 0), (a, 1), ..., (a, k)$. For each outgoing road from city a to city b, if a has a commander then create an edge from (a, i) to $(b, i+1)$ for all $0 \leq i \leq k-1$, and an edge from (a, k) to (b, k). If a does not have a commander, then create an edge from (a, i) to (b, i) for all $0 \leq i \leq k$. Run Dijkstra's algorithm to find the shortest path from (Aquilegia, 0) to (Iris, k). This graph has $O(kn)$ nodes and $O(km)$ edges, so the runtime is $O(km + knlog(kn))$.

(2) [10 points] Sam learns that the magician does not care about demon commanders, but can be persuaded with gold coins. Adventurers can earn gold pieces by accepting requests to slay various types of monsters. To maximize his efficiency, Sam decides that he will only slay monsters that are on the road he is on. Sam maps out which kinds of monsters live near each road (and how much gold he can make from those monsters), as well as the price to stay at the inn in each city. The inns are used to dealing with poor adventurers, and thus have devised a credit system where adventurers may pay all of their inn fees at the very end of their journey. Specifically, if he travels from city a to city b on road r_{ab}, the most amount of gold pieces he can make is denoted by p_{ab} and he will incur a cost of c_b, which he will need to pay once he makes it to Iris. Note that the monsters respawn relatively quickly, so if Sam ever repeats a road r_{ab} he can still accept the same request as before and once again make p_{ab}. He is unsure of how much the magician will charge him, but he assumes that 500 gold pieces should be enough. In $O(nm)$ time, determine if there is a path of roads that Sam can take to end at Iris with at least 500 gold pieces (after paying the inn fees) if he starts his journey with 0.

Solution: Build a graph with cities as nodes and roads as directed edges. Assign each edge e from city a to city b the weight $-r_{ab} + c_b$. Run Bellman-Ford on this graph and see if there is a path to Iris with a weight of -500 or less. If there is a negative weight cycle, then since the graph is strongly connected (clarified on Piazza), there must be a path to Iris with a weight of -500 so we automatically return true.

Note that if we do not assume the graph is strongly connected, then if we detect any negative weight cycles then we need to determine if it is possible to get from the cycle to Iris. We can identify a node in the negative weight cycle by starting from the node whose distance was still able to be updated in the last loop of Bellman-Ford (the node that tells us there is a negative weight cycle) and following the parent pointers until we repeat a node. We then run BFS or DFS from this node to see if we can reach Iris from the negative weight cycle, and if so return true. If Iris is not reachable from a negative weight cycle, then the shortest distance from running Bellman-Ford will still represent the negative of the maximum amount of gold that Sam can make before reaching Iris.

Problem 7-4. [40 points] **When to start?**

As a busy MIT student, you're hoping to schedule n problem sets to complete in the coming week, by determining a start time s_i at which you will start each assignment. Due to some assignments depending on each other, some restrictions will require you to start some assignments a certain amount before or after another assignment. These restrictions take the form $s_i - s_j \leq t_k$ for some time t_k, and there are m of them. Times (both the start times s_i and restriction times t_k) may be negative.

(1) [3 points] During your first week of classes, you have 4 assignments, with 5 restrictions

$$\begin{aligned} s_2 - s_1 &\leq -4 \\ s_3 - s_1 &\leq -15 \\ s_1 - s_4 &\leq 7 \\ s_4 - s_3 &\leq 10 \\ s_3 - s_2 &\leq -8. \end{aligned}$$

Find an assignment of all four s_i that satisfies these restrictions. (*Hint:* Start with $s_1 = 0$. What upper bounds for other s_i does this give us?)

Solution: One solution is $s_1 = 0$, $s_2 = -4$, $s_3 = -15$, and $s_4 = -5$.

(To find this solution, we can proceed as the hint does, beginning with $s_1 = 0$. This implies $s_2 \leq -4$ and $s_3 \leq -15$; for now, let's set these as equalities and see if it works. This setting of s_3 implies $s_4 \leq -5$. Setting this as an equality too, we see that this satisfies all restrictions.)

Rubric:

- 3 for a correct solution.
- Partial 1 if all but one equation is satisfied.

(2) [3 points] In your second week of classes, you have 4 assignments, and only 4 restrictions

$$\begin{aligned} s_1 - s_2 &\leq 7 \\ s_2 - s_3 &\leq -2 \\ s_3 - s_4 &\leq -10 \\ s_4 - s_1 &\leq 3. \end{aligned}$$

Argue that there is no choice of s_i that satisfies these restrictions. Your solution should be brief and algebraic in nature.

Solution: Summing up the 4 inequalities, we find that the left side simplifies to 0 while the right side sums to -2; $0 \leq -2$ is a contradiction, so this is impossible.

Rubric:

- 3 for a correct explanation.

(3) [5 points] Explain how a set of assignments and restrictions can be represented using a graph. You need not prove anything about your representation, but you must use this representation in the following part.

Solution: We begin with n, labeled $\{1, 2, ..., i\}$ vertices, corresponding to the n assignment start times. Then, for each restriction of the form $s_i - s_j \leq t_k$, we create an edge *from* the vertex corresponding to s_j *to* the vertex corresponding to s_i, with weight t_k.

Rubric:

- 3 for associating s_i to vertices and restrictions to edges.
- 2 for edges being in the correct direction.

(4) [9 points] Provide an algorithm that either finds a choice of starting times for your assignments, or states that none exists. Your algorithm should run in $O(mn)$ time and use part (c). Explaining the correctness of your algorithm is of particular importance for this part.

Solution:

We slightly modify the graph from the previous part to include a "source node," which will be connected to every other vertex with an edge weight of 0. We run Bellman-Ford on this modified graph, from the new source node. If, at the conclusion of Bellman-Ford, we detect no negative weight cycles (i.e. all edges are properly relaxed after $n-1$ rounds of relaxations), the distance estimates δ_i correspond to an assignment of s_i that works. Otherwise, Bellman-Ford has detected a negative weight cycle, and we return that there is no solution.

Correctness: If all edges are correctly relaxed, the distances δ_i satisfy $\delta_i \leq \delta_j + d(j, i)$ for each edge (j, i). By the way we've constructed the graph edges, this is equivalent to saying $\delta_i - \delta_j \leq t_k$ for every restriction.

If a cycle has a negative weight, this will result in a contradiction just like the one we saw in part (b): adding up the equations in a cycle results in all s_i cancelling out, leaving an equation of the form $0 \leq k$ for some negative number k.

Runtime: The number of edges is $m + n$, and the number of vertices is $n + 1$. Using Bellman-Ford, we get a runtime of $O((m+n)(n+1)) = O(mn)$, as desired.

Note: In order to deal with start times that are not part of any restrictions (i.e. when $m = o(n)$), note that we can simply remove any vertex in the graph that has no incoming or outgoing edges - in this case, there are no restrictions on the corresponding start time, so it can be arbitrarily chosen. In this way, we can ensure that $m = \Omega(n')$, where n' is the number of start times that are involved in at least one restriction, so the Bellman-Ford runtime of $O((m + n')(n' + 1) = O(mn' + n'^2) = O(mn') = O(mn)$, as desired.

Rubric:

- 3 points for correct algorithm using Bellman-Ford and a super-node.
- 3 points for justifying that the δ_i give a valid assignment of s_i.
- 2 points for justifying that a negative weight cycle leads to no solution.
- 1 for justifying runtime.

(5) [20 points] Code the function `find_start_times` described in the previous part. Download a template and submit your code at `https://alg.mit.edu/spring21/PS7`.

Please submit a screenshot of your report that includes your `alg.mit.edu` student ID (top right corner) and the percentage of tests passed to Gradescope.

Solution: Solution.