Problem Set 5

This problem set is divided into two parts: Part A problems are theory questions, and Part B problems are programming tasks.

Part A questions are due **Wednesday, April 21st** at **11:59PM**.

Part B questions are due **Friday, April 23rd** at **11:59PM**.

Solutions should be turned in through the course website in PDF form using LaTeX or scanned handwritten solutions.
A template for writing up solutions in LaTeX is available on the course website.
Remember, your goal is to communicate. Full credit will be given only to the correct solution which is described clearly. Convoluted and obtuse descriptions might receive low marks, even when they are correct. Also, aim for concise solutions, as it will save you time spent on write-ups, and also help you conceptualize the key idea of the problem.

Part A: Due Wednesday, April 21st

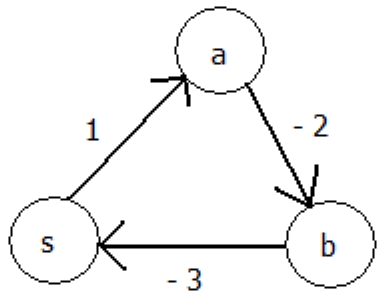
1. **(20 points)** Graph Examples

   (a) **(6 points)** Give an example of a graph that has some shortest path from vertex s to vertex v, such that for *all* non-zero (positive or negative) real c, if c is added to the weights of all the edges, that path is no longer a shortest path from s to v.

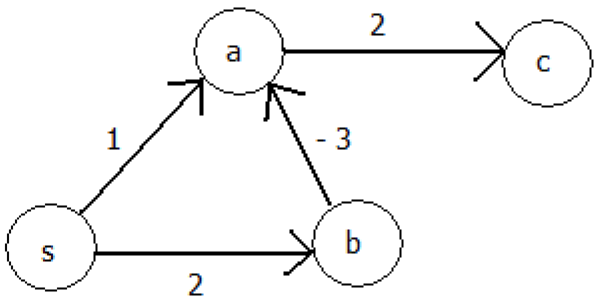
   Solution: The graph should have three different paths from s to v, each with equal total weight but with different numbers of edges. Call them p_1, p_2, and p_3, and assume p_1 has the smallest number of edges and p_3 has the greatest number of edges. Then p_2 is currently a shortest path from s to v, but if all the edge weights are increased by a constant c, then the total weight of p_1 increases less than the total weight of p_2 since it has less edges, so p_2 is no longer a shortest path. Similarly, if all the edge weights are decreased by a constant c, then the total weight of p_3 decreases by more than the total weight of p_2, since it has more edges, so p_2 is no longer a shortest path.

   (b) **(6 points)** Give an example of a graph with negative-weight edges for which Dijkstra's algorithm gives incorrect answers.

   Solution: Dijkstra's will give incorrect answers if a node u is extracted from Q before $d[u]$ is equal to $\delta(s, u)$, and as a result the d's of the nodes adjacent to it will not be updated to the correct shortest distances, because the edges from u are only relaxed once – when u is extracted. There are two examples of such graphs below.



Example 1



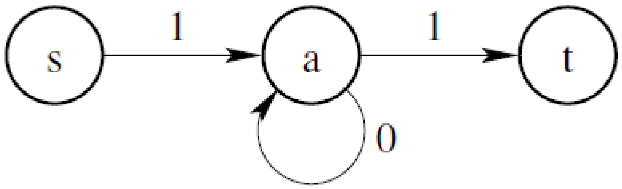
Example 2

In Example 1, there is a negative cycle reachable from s. Running Dijkstra's on the graph will return $d[s] = -4$, $d[a] = 1$, and $d[b] = -1$. But these are incorrect because if we keep relaxing edges, the distances will decrease; there is no shortest distance to any node in this graph.

In Example 2, the nodes are extracted in the order s, a, b, c. When a is extracted and the edge (a, c) is relaxed, $d[c]$ becomes 3. Later, by relaxing the edge from b to a, the distance to a becomes -1. But a has already been extracted, so the edge from a to c is not relaxed again, and $d[c]$ remains 3, when $\delta(s, c) = 1$.

(c) **(8 points)** Suppose that the RELAX function is changed so that it updates if $d[v] \geq d[u] + w(u, v)$ (instead of strictly greater than). Give an example of a graph with no negative-weight cycles such that if Bellman-Ford is run on this graph with the modified RELAX function, it will not return the correct π outputs.

Solution: The parent pointers may not lead back to the source node if a zero-length cycle exists.
In the example below, relaxing the (s, a) edge will set $d[a] = 1$ and $\pi[a] = s$. Then relaxing the (a, a) edge will set $d[a] = 1$ and $\pi[a] = a$. Following the π pointers from t will no longer give a path to s, so the algorithm is incorrect.



2. **(15 points)** Even-Length Paths

An even-length path is a path traversing an even number of edges. Describe a modified version of Dijkstra's algorithm that finds the shortest even-length path in a graph $G = (V, E)$ from a given start vertex s to all vertices $t \in V$. The graph has non-negative edge weights. Your solution should have the same asymptotic running time as Dijkstra's algorithm. (HINT: try solving the problem by constructing a graph G' that is somehow related to G, running Dijkstra's algorithm on G', and projecting the results back onto G.)

Solution: Construct a new, bipartite graph G' as follows: for each vertex $v \in V$, create *two* vertices, v_0 and v_1, in G'. For each edge $(u, v) \in E$, create *two* edges, (u_0, v_1) and (u_1, v_0) in G'. Then run Dijkstra on G', with start vertex s_0.

All paths in G' ending at a vertex v_0 have an even number of edges, and all paths ending at a vertex v_1 have an odd number of edges. (Traversing an edge means switching between the set of "even vertices" and the set of "odd vertices".) Thus, the shortest even-length path to a vertex t in G can be found by determining the shortest path from s_0 to t_0 in G', then discarding the subscripts.

3. **(15 points)** Shortest Paths with Negative-Weight Cycles

   $G = (V, E)$ is a directed graph that contains at least one negative-weight cycle. Give an $O(VE)$-time algorithm that labels each vertex v with the shortest-path distance from source vertex s to v, $-\infty$ if there the shortest-path distance is undefined because of a negative-weight cycle, and ∞ if v is not reachable from s.

Solution: There are two natural $O(VE)$-time algorithms for this problem. In general, the idea is to run Bellman-Ford to compute $d[v] = \delta(s, v)$ when $\delta(s, v) > -\infty$. Then, we need to find all vertices v reachable from a negative-weight cycle and set $d[v] = -\infty$.

1. After running Bellman-Ford, find all vertices with relaxable outgoing edges.
   Then run a DFS (or BFS) from each such vertex v, marking them with $d[v] = -\infty$. This runs in $O(VE)$ time ($O(E)$ per DFS). For practical efficiency, each successive DFS can backtrack when it hits a vertex visited by a previous DFS.

2. Just run Bellman-Ford again, but instead of relaxing relaxable edges, set d's to $-\infty$.