Problem 1. Prove the correctness of Dijkstra's algorithm (when the edges have non-negative weights).
Indicate where the assumption of non-negative edge weights is used.

Proof. We claim that for any vertex $v \in V$, when it is removed from S, we must have $dist(v) = spdist(s, v)$.

Proof by induction on the order of removal of vertices, denoted as $u_1, u_2, \dots, u_n$.

Inductive Hypothesis. When u_k is removed from S, we have $dist(u_i) = spdist(s, u_i)$ for every $1 \le i \le k$.

Base Case. The first vertex being removed, u_1, has $dist(u_1)$ initialized to zero, which is exactly $spdist(s, u_1)$. We have made use of the fact that no edge is negative, as such the sum of weights cannot be lower than zero forming a shorter path.

Inductive Step. Consider the moment when u_k is being removed. By the inductive hypothesis, $spdist(s, u_i) = dist(u_i)$ for every $1 \le i \le k-1$. Suppose for a contradiction that the shortest path from s to u_k is shorter than $dist(u_k)$, i.e. $spdist(s, u_k) < dist(u_k)$.

If a shortest path with length $spdist(s, u_k)$ from s to u_k involves entirely of vertices that have been removed, then immediately before we enter u_k, we must be on an edge (u_i, u_k) $(1 \le i \le k-1)$. Since $spdist(s, u_i) = dist(u_i)$, and we must have relaxed $dist(u_k)$ with the edge (u_i, u_k), then by the algorithm $dist(u_k) \le dist(s, u_i) + w_{u_i u_k}$, which must have already considered the path with length $spdist(s, u_k)$, implying that this case is impossible.

Hence, we must be able to identify a vertex z which has not been removed from S on the shortest path. Let y be the preceeding vertex on the path, which should have been removed. Clearly, $spdist(s, y) + w_{yz} \le spdist(s, u_k)$. By our former argument, we must have $spdist(s, y) + w_{yz} \ge dist(z)$ relaxed by y. Also, $dist(z) \ge dist(u_k)$, since z is relaxed by y and u_k is the vertex being removed before z. Combining these inequality, we have

$$dist(u_k) \le dist(z) \le dist(y) + w_{yz} \le spdist(s, u_k),$$

which is a contradiction. □

Problem 2. Consider a directed simple graph $G = (V, E)$ where each edge $e \in E$ has an arbitrary weight $w(e)$ (which can be negative). It is known that G does not have negative cycles. Prove: given any vertices $s, t \in V$, at least one shortest path from s to t is a simple path (i.e., no vertex appears twice on the path).

Proof. Let $\pi = (u_1, u_2, \dots, u_k)$ be a shortest path from s to t that uses the least number of edges. If π is not a simple path, then there must exist $1 \le i < j \le t$ with $u_i = u_j$. Thus the subpath $u_i, u_{i+1}, \dots, u_{j-1}, u_j$ is a cycle. Since the length of the any cycle is non-negative, we can remove the subpath and obtain: $u_1, \dots, u_i, u_{j+1}, u_{j+2}, t$, which is also a path from s to t. The new path must use strictly fewer edges, which contradicts the definition of π. □

Problem 3. Consider a simple acyclic directed graph $G = (V, E)$ where each edge $e \in E$ has an arbitrary weight $w(e)$ (which can be negative). Solve the SSSP problem on G in $O(|V| + |E|)$ time.

Solution. Define $\mathrm{IN}(v)$ as the set of in-neighbors of v, let

$$f(u) = \begin{cases} 0, & \text{if } s = v, \\ \infty, & \text{if } \mathrm{IN}(u) = \emptyset, \\ \min_{e=(v,u):v\in \mathrm{IN}(u)}\{f(v) + w(e)\}, & \text{otherwise.} \end{cases}$$

Compute the topological order of G in $O(|V| + |E|)$ time and compute $f(u)$ by dynamic programming. Then, the shortest path tree can be constructed in $O(|V| + |E|)$ time by the piggyback technique.

Problem 4. Let $G = (V, E)$ be a simple directed graph where the weight of an edge (u, v) is $w(u, v)$. Prove: the following algorithm correctly decides whether G has a negative cycle.

procedure NegativeCycleDetection(G)
  Pick an arbitrary vertex $s \in V$
  Initialize $dist(s) = 0$ and $dist(v) = \infty$ for every other vertex $v \in V$
  for $i \leftarrow 1$ **to** $|V| - 1$ **do**
    Relax all the edges in E
  end for
  for each edge $(u, v) \in E$ **do**
    if $dist(v) > dist(u) + w(u, v)$ **then**
      return “A negative cycle is found”
    end if
  end for
  return “No negative cycle exists”
end procedure

Proof. Two claims are required to prove this lemma.

Claim 1. If $dist(v) > dist(u) + w(u, v)$ holds for some (u, v), then there must be a negative cycle. This is because $dist(v)$ must already be the length of the shortest path after $|V| - 1$ rounds of edge relaxations if there is no negative cycle, which is guaranteed by Bellman-Ford’s algorithm.

Claim 2. If there is a negative cycle, then $dist(v) > dist(u) + w(u, v)$ holds for some (u, v). Consider the negative cycle $C = (v_1 - v_2 - \ldots - v_\ell - v_1)$. Hence,

$$w(v_\ell, v_1) + \sum_{i=1}^{\ell-1} w(v_i, v_{i+1}) < 0.$$

Assume the contrary that no (u, v) satisfies the inequality, then for every $1 \le i \le \ell-1$, $dist(v_{i+1}) \le dist(v_i) + w(v_i, v_{i+1})$, and $dist(v_1) \le dist(v_\ell) + w(v_\ell, v_1)$. This implies

$$\sum_{i=1}^{\ell} dist(v_i) \le \left(\sum_{i=1}^{\ell} dist(v_i)\right) + w(v_\ell, v_1) + \sum_{i=1}^{\ell-1} w(v_i, v_{i+1}) \Rightarrow 0 \le w(v_\ell, v_1) + \sum_{i=1}^{\ell-1} w(v_i, v_{i+1}),$$

which contradicts to the fact that C is a negative cycle. $\square$

Problem 5. Let $G = (V, E)$ be a simple directed graph where every edge (u, v) carries a weight $w(u, v)$, which can be negative. G has no negative cycles. Recall that Johnson's algorithm adds a vertex v_{dummy}, as well as some out-going edges of v_{dummy}, to G and computes the shortest path distance $spdist(v_{\text{dummy}}, v)$ from v_{dummy} to every vertex. Then, the weight of each edge (u, v) is modified to:

$$w'(u, v) = w(u, v) + spdist(v_{\text{dummy}}, u) - spdist(v_{\text{dummy}}, v).$$

Prove: $w'(u, v) \geq 0$.

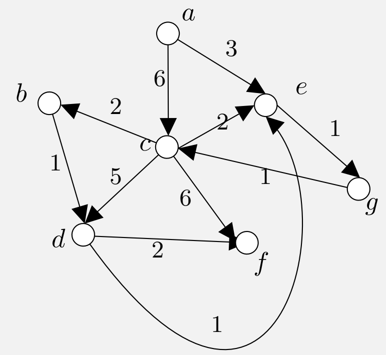
Proof. That is to prove $w(u, v) + spdist(v_{\text{dummy}}, u) \geq spdist(v_{\text{dummy}}, v)$. Clearly, the path formed by the shortest path from v_{dummy} to u and the edge (u, v) forms a possible path from v_{dummy} to v, and hence must be at least as long as the shortest path from v_{dummy} to v. □

Problem 6. Let $G = (V, E)$ be a simple directed graph where every edge (u, v) carries a non-negative weight $w(u, v)$. Apply Johnson's algorithm to compute a new weight $w'(u, v)$ for each edge $(u, v) \in E$. Prove: $w'(u, v) = w(u, v)$.

Proof. Since all the edges are non-negative, the shortest path length that starts from v_{dummy} and ends at any $v \in V$ cannot be less than zero. However, since we have added $|V|$ edges carrying the weight of 0 to all vertices $v \in V$, $spdist(v_{\text{dummy}}, v) = 0$.

Hence, $w'(u, v) = w(u, v) - h(u) + h(v) = w(u, v) - spdist(v_{\text{dummy}}, u) + spdist(v_{\text{dummy}}, v) = w(u, v)$. □

Problem 7. Consider the weighted directed graph below.



Run Dijkstra's algorithm starting from vertex a. Recall that the algorithm relaxes the outgoing edges of every other vertex in turn. Give the order of vertices by which the algorithm relaxes their edges.

Solution. The order is given by a, e, g, c, b, d, f.

Step 1

vertex v	$parent(v)$	$dist(v)$
a	nil	0
b	nil	∞
c	a	6
d	nil	∞
e	a	3
f	nil	∞
g	nil	∞

Step 2

vertex v	$parent(v)$	$dist(v)$
a	nil	0
b	nil	∞
c	a	6
d	nil	∞
e	a	3
f	nil	∞
g	e	4

Step 3

vertex v	$parent(v)$	$dist(v)$
a	nil	0
b	nil	∞
c	g	5
d	nil	∞
e	a	3
f	nil	∞
g	e	4

Step 4

vertex v	$parent(v)$	$dist(v)$
a	nil	0
b	c	7
c	g	5
d	c	10
e	a	3
f	c	11
g	e	4

Step 5

vertex v	$parent(v)$	$dist(v)$
a	nil	0
b	c	7
c	g	5
d	b	8
e	a	3
f	c	11
g	e	4

Step 6

vertex v	$parent(v)$	$dist(v)$
a	nil	0
b	c	7
c	g	5
d	b	8
e	a	3
f	d	10
g	e	4

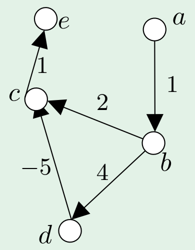
Problem 8. Consider a simple directed graph $G = (V, E)$ where each edge $(u, v) \in E$ carries a non-negative weight $w(u, v)$. Given two vertices $u, v \in V$, function $spdist(u, v)$ represents the shortest path distance from u to v. Given a vertex $v \in V$, denote by $\mathrm{IN}(v)$ the set of in-neighbors of v. Let s and t be two distinct vertices in G. Prove:

$$spdist(s,t) = \min_{v \in \mathrm{IN}(t)} \{spdist(s,v) + w(v,t)\}.$$

Proof. Consider the Dijkstra's Algorithm, where $dist(t)$ is only updated by a relaxation of some vertex v, which is an in-neighbour of t. Clearly, at any stage of the algorithm we have $dist(v) \geq spdist(s, v)$, and thus we always have $dist(v)+w(v, t) \geq spdist(s, v)+w(v, t) \geq dist(t) \geq spdist(s, t)$ when $dist(t)$ is being updated. $\square$

Problem 9. Give a counterexample to show that Dijkstra's algorithm does not work if edge weights can be negative.

Solution. Consider the SSSP instance with a being the source. Since e can only be relaxed by c and $dist(c)$ is only updated to -1 after the removal of d, $spdist(a, e) = 0$ cannot be correctly computed.



Problem 10. Consider the SSSP problem where every edge in the input graph $G = (V, E)$ has the same weight. Given two distinct vertices $s, t \in V$, describe an algorithm to find a shortest path from s to t in $O(|V| + |E|)$ time.

Solution. We replace the priority queue with a normal queue which supports push and pop operations in $O(1)$ time. Then, the Dijkstra's algorithm has a time complexity improved to $O(|V| + |E|)$.

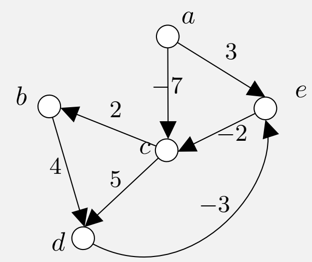
Lemma. For any k, let $(v_1, v_2, \ldots, v_k)$ be the elements of the queue at iteration k. At this iteration,

1. $dist(v_k) - dist(v_1) \leq 1$.
2. For any $i < j$, $dist(v_i) \leq dist(v_j)$.

Proof. (Base Case) Initially, the queue is empty, and hence the statements are vacuously true.

(Inductive Step) Consider the $(k+1)$-st iteration, where we remove the front element v_1 and the new front is v_2. Then, neighbours of v_1 are enqueued. Consider any neighbour u, $dist(u)$ is set to $dist(v_1) + 1$. Also, $dist(v_2) \geq dist(v_1) = dist(u) - 1$. Hence, concluding with the inductive hypothesis, the statements also hold for the $(k+1)$-st iteration.

Problem 11. Consider the weighted directed graph $G = (V, E)$ below.



Set the source vertex to a and run Bellman-Ford's algorithm, which performs 4 rounds of edge relaxations. Show $dist(v)$ of every $v \in V$ after each round.

Solution. In each round, we relax by the order $(a, c), (a, e), (b, d), (c, b), (c, d), (d, e), (e, c)$.

Round 1

vertex v	$dist(v)$
a	0
b	5
c	−7
d	−2
e	−5

Round 2

vertex v	$dist(v)$
a	0
b	−5
c	−7
d	−2
e	−5

Round 3

vertex v	$dist(v)$
a	0
b	−5
c	−7
d	−2
e	−5

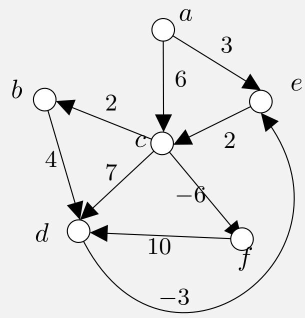
Round 4

vertex v	$dist(v)$
a	0
b	−5
c	−7
d	−2
e	−5

Problem 12. The Bellman-Ford algorithm presented in the lecture computes only the shortest-path distance from the source vertex s to every vertex. Extend the algorithm to output a shortest-path tree of s. The modified algorithm must still terminate in $O(|V||E|)$ time.

Solution. Compute $dist(u)$ for every $u \in V$ by Bellman-Ford algorithm in $O(|V| + |E|)$ time. Suppose that $dist(u)$ is updated by an edge (v, u), then let the assigned parent $p(u)$ be v. Scan the list of assigned parent and construct a graph T, this can be done in $O(|V| + |E|)$ time. We claim that T is a tree since any cycle must be non-negative. Then, there must exist an edge (u, v) on T such that $spdist(s, v) \leq spdist(s, u)$, while $spdist(s, v)$ has been attained before $dist(u)$ reaches $spdist(s, u)$, violating the defintion of how we relax $dist(v)$.

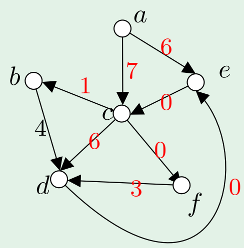
Problem 13. Consider the weighted directed graph $G = (V, E)$ below.



Suppose that we run Johnson's algorithm on G. Recall that the algorithm re-weights all the edges to make sure that every edge should carry a non-negative weight. Give all the edge weights after the re-weighting.

Solution.

vertex v	$h(v)$
a	0
b	0
c	-1
d	0
e	-3
f	-7

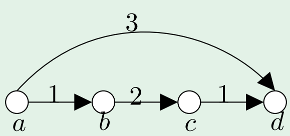


Problem 14. Prof. Goofy proposes to replace Johnson's re-weighting strategy with the following one:

- Find the smallest edge weight z in G.
- Re-weight each edge (u, v) in G by adding $-z$ to its weight, namely, (u, v) carries the weight $w(u, v) - z$ after the re-weighting.

Let G' be the resulting graph obtained by applying Prof. Goofy's strategy. Give an example to show that the strategy does not guarantee the following property: a path π from vertex u to v is a shortest path in G if and only if it is a shortest path in G'.

Solution. Consider the following instance, where the shortest path from a to d in G is 3 directly by the edge (a, d). However, in G', (a, d) carries the weight of 2, while $(a, b), (b, c), (c, d)$ have weights $0, 1, 0$, respectively, froming a path with length 1 shorter than directly from (a, d).



Problem 15. Let $G = (V, E)$ be a simple directed graph where each edge $(u, v) \in E$ carries a weight $w(u, v)$, which can be negative. Let $h : V \to Z$ be an arbitrary function (mapping each vertex in V to an integer). For each $(u, v) \in E$, define $w'(u, v) = w(u, v) + h(u) - h(v)$. Let $G' = (V, E)$ be the same graph as G, except that the edges are weighted using w'. Prove: G has a negative cycle if and only if G' does.

Proof. By definition, the reweighted edge carries the weight $w_{uv} + h(u) - h(v)$. Consider the equality

$$\sum_{i=1}^{k-1} w'_{v_i v_{i+1}} + w'_{v_k v_1} = \sum_{i=1}^{k-1} h(v_i) - h(v_{i+1}) + w_{v_i v_{i+1}} + h(v_k) - h(v_1) + w(v_k v_1) = \sum_{i=1}^{k-1} w_{v_i v_{i+1}} + w_{v_k v_1} < 0.$$

This effectively implies that cycle length is preserved throughout the transformation. Hence, a negative cycle $C = (v_1 - v_2 - \ldots - v_k - v_1)$ exists in G iff the same negative cycle $(v_1 - v_2 - \ldots - v_k - v_1)$ exists in G'.

□

Problem 16. Let $G = (V, E)$ be a simple directed graph where $V = \{1, 2, \ldots, n\}$. The transitive closure of G is an $n \times n$ matrix $\boldsymbol{M}$ where

$$M[i, j] = \begin{cases} 1, & \text{if vertex } i \text{ can reach vertex } j \text{ in } G, \\ 0, & \text{otherwise.} \end{cases}$$

Compute $\boldsymbol{M}$ in $O(|V|(|V| + |E|))$ time.

Solution. Set all the weights of the edges in G to 1, by traversing over the linked-list of G in $O(|V| + |E|)$ time.

For each vertex $v \in V$, set the source to v and run the SSSP algorithm on G by BFS. Each execution is done in $O(|V| + |E|)$ time. Consider $dist(v)$. If $dist(u) \neq \infty$, then set $M[v, u] = 1$, otherwise set $M[v, u] = 0$. This process costs $O(|V|)$ for each $v \in V$.

Hence, $\boldsymbol{M}$ can be computed in $O(|V|(|V| + |E|))$ time in total.