Problem 1 (20 points). For each of the following statements, either give a (short) proof to show that it is correct, or give a counter-example to show that it is not correct.

1. Let $G = (V, E)$ be a directed graph. Let $s \in V$. During a BFS run on G starting from s, vertex v is added to the queue after u. If $(u, v) \in E$, then we have $distance(s, u) < distance(s, v)$.

2. Let $G = (V, E)$ be a directed graph. Let $e = (u, v) \in E$. During a DFS run on G, edge e is a cross edge. Then we must have $post[u] < pre[v]$.

3. Let $G = (V, E)$ be a directed graph without negative cycles. Let $e \in E$ be on one shortest path from s to t. If the length of e is increased by 1, then the distance from s to t will also be increased by 1.

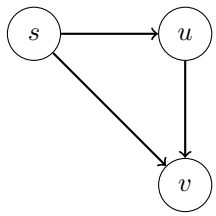
4. Let $G = (V, E)$ be a directed graph with positive edge length. If G satisfies that $|V| = |E|$, then we can always find an edge e in G such that increasing its length by 1 will not change the distance from s to any other vertex.

Solution.

Grading scheme: 2 points for identifying whether it is correct/incorrect; 3 for counterexample/proof.

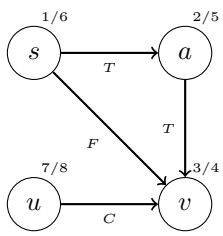
1. Incorrect.

   Counterexample: In the following graph, if we run a BFS starting from s, the order of vertices added to queue is s, u, v. So vertex v is added to the queue after u. Here $distance(s, u) = distance(s, v) = 1$, so $distance(s, u) \not< distance(s, v)$.



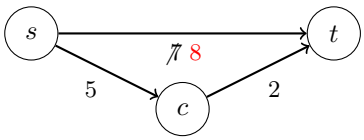
2. Incorrect.

   Counterexample: In the following graph, if we run DFS starting from s, we find that $u \to v$ is a cross edge. But $post[u] = 8 > pre[v] = 3$.



3. Incorrect.

   Counterexample: In the following graph, there are two shortest paths from s to t, and the length of each shortest path is 7. $e = (s, t)$ is an edge in the shortest path. Even if we increase its length by 1 (in figure, $l(s, t)$ is increased from 7 to 8), the distance from s to t remains the same as before. This is because the alternate path $s \to c \to t$ has a total length of $5 + 2 = 7$.



4. Correct.

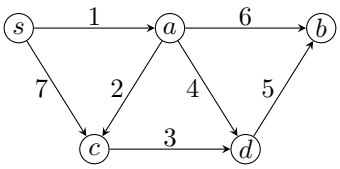
   Proof: We know that shortest path is a tree. Let m be the number of edges and n be the number of vertices in the shortest path tree T. From the property of tree, $m = n - 1$.

   If all the vertices are reachable from starting vertex s, then all the vertices will be included in T. Then $n = |V|$ and $m = n - 1 = |V| - 1 = |E| - 1$. So we have one less edge in the tree. If we increase the length of that edge by 1, the distance from s to any other vertex does not change.

   If some vertices are not reachable from s, then we will have less vertices in shortest path tree T, therefore $n < |V|$. From this we get, $n - 1 < |V| - 1$
   or, $m < |V| - 1$
   or, $m < |E| - 1$
   or, $m < |E|$.

   In both cases, we see that $m < |E|$. That means there is at least one edge in the graph that is not included in the shortest path tree. We can pick any edge that is not in the shortest path tree, and increase its length without affecting the distance of from s to any vertex other than s.

Problem 2 (10 points). Let $G = (V, E)$ be the following directed graph.



1. Give the order of vertices in which they are removed the priority queue when the Dijkstra's algorithm is applied on G with s as the starting vertex.

2. You are allowed to change the length of edge (a, d) but need to guarantee that, when the Dijkstra's algorithm is applied on the new graph (still with s as the starting vertex), the order of vertices being removed from the priority queue needs to be the same as before. Give the range (i.e., an interval) for possible new length of edge (a, d) so that the above constraint can be satisfied.

Solution:

1. The order of vertices in which they are removed the priority queue is s, a, c, d, b.

2. As the order of vertices being removed from the priority queue needs to be same as before, we need to guarantee $distance(s, c) < distance(s, d) < distance(s, b)$.

   Notice that $distance(s, c) = 3$ as changing $l(a, d)$ does not affect $distance(s, c)$.

   There are two edges leading to d. Therefore, we have $distance(s, d) = \min\{distance(s, a) + l(a, d), distance(s, c) + 3\} = \min\{1 + l(a, d), 6\}$.

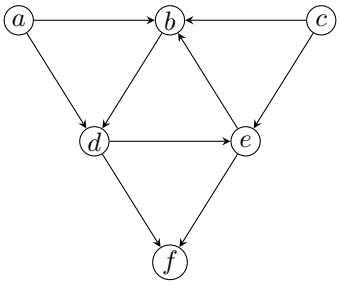
   There are two edges leading to b. Therefore, we have $distance(s, b) = \min\{distance(s, a) + 6, distance(s, d) + 5\} = \min\{7, distance(s, d) + 5\} = \min\{7, \min\{1 + l(a, d), 6\} + 5\} = \min\{7, 6 + l(a, d), 11\} = \min\{7, 6 + l(a, d)\}$.

Following $distance(s,c) < distance(s,d) < distance(s,b)$, we have

$$3 < \min\{1 + l(a,d), 6\} < \min\{7, 6 + l(a,d)\},$$

A simply mathematical analysis gives $l(a,d) > 2$, i.e., the range for $l(a,d)$ is $(2,\infty)$.

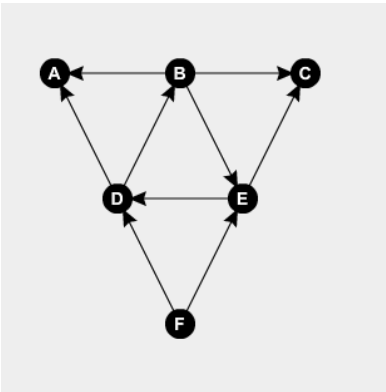
Problem 3 (15 points). Let $G = (V,E)$ be the following directed graph.



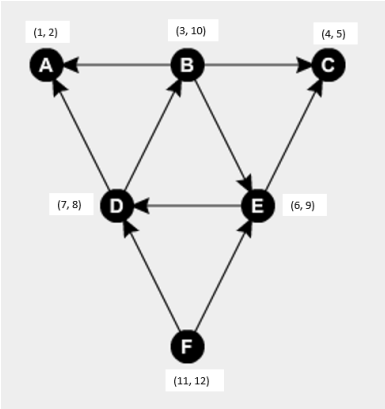
1. Draw the reverse graph G^R of G.
2. Run DFS on G^R to obtain a *post* number for each vertex. Assume that in the adjacency list representation of G, vertices are stored alphabetically, and in the list for each vertex, its adjacent vertices are also sorted alphabetically. In other words, the DFS algorithm needs to examine all vertices alphabetically, and when it traverses the adjacent vertices of a vertex, it also needs to examine them alphabetically. (NOTE: you only need to give the *post* number for each vertex; you do not need to illustrate the internal steps of running DFS on G^R.)
3. Run DFS on G by examining all vertices in the reverse order of their above *post* numbers to obtain all connected components of G. (NOTE: you only need to give the *cc* array, which gives the index of the connected components the corresponding vertex belongs to; you do not need to illustrate the internal steps of running DFS on G.)
4. Draw the meta graph G_M for G and give a linearization for G_M.

Solution.

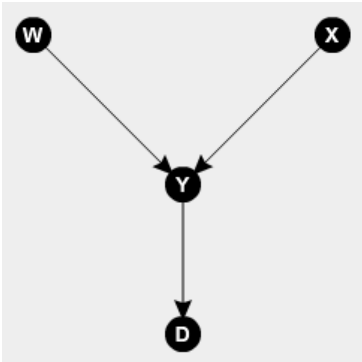
1. Graph G^R is given below:



2. The *post* number (together with the *pre* number) is given below:



3. The cc array is $cc[a,b,c,d,e,f] = [4,2,3,2,2,1]$.
4. The meta graph G_M is given below, where $W = \{a\}$, $X = \{c\}$, $Y = \{b,d,e\}$, and $D = \{f\}$. One possible linearization of G_M is W,X,Y,D.



Problem 4 (15 points). Let $G = (V,E)$ be a directed acyclic graph. Design an $O(|V|+|E|)$ time algorithm to decide whether there exists a path that goes through every vertex of G exactly once. Prove your algorithm is correct and explain why your algorithm runs in $O(|V|+|E|)$ time.

Solution.

Algorithm. Perform a linearization (topological sort) of G, and then check if successive vertices in the linearization are connected by an edge in the graph: if it is true for all pairs of successive vertices then there exists a path goes through all vertices exactly once (it is called a *Hamiltonian path*); otherwise these does not exists such path.

Proof. If there exists an edge connecting every pair of successive vertices then the linearization gives us such a Hamiltonian path. On the other side, if there exists a Hamiltonian path, then there exists a unique linearization of G, which is given by the Hamiltonian path, and consequently, every pair of successive vertices in this order are connected by an edge.

Running time. Linearization takes $O(|V|+|E|)$ time and checking all pairs also takes $O(|V|+|E|)$ time. Therefore the total running time is $O(|V|+|E|)$.

Problem 5 (20 points). Let $G = (V,E)$ be a directed graph. Let $s \in V$. Let $l(e)$ be the edge length of $e \in E$ (it is possible that $l(e) < 0$). Complete a dynamic programming algorithm for the single-source

shortest-path problem using this definition of subproblems: define $dist(v,k)$ be the length of the shortest path from s to v using exactly k edges, for any $v \in V$ and $0 \leq k < |V|$.

1. Give the recursion formula for updating $dist(v,k)$.
2. Give the initialization step of the dynamic programming algorithm using pseudo-code.
3. Give the iteration step of the dynamic programming algorithm using pseudo-code.
4. Suppose that graph G does not contain negative cycles. Give the termination step of the dynamic programming algorithm using pseudo-code.
5. Prove the following statement or give a counter-example: G does not contain negative cycle if and only if $dist(v,|V|-1) = dist(v,|V|)$ for all $v \in V$.

Solution.

1. $dist(v,k) = \min_{u \in V:(u,v) \in E} dist(u,k-1) + l(u,v)$
2. The pseudo-code for *initialization* is below:
   $dist(s,0) = 0$
   $dist(v,0) = \infty$, if $v \neq s$
3. The pseudo-code for *iteration* is below:
   for $k = 1$ to $|V|-1$:
       for $v \in V$:
           $dist(v,k) = \min_{u \in V:(u,v) \in E} dist(u,k-1) + l(u,v)$
       endfor
   endfor
4. The pseudo-code for *termination* is below:
   for $v \in V$:
       $distance(s,v) = \min_{k=0}^{|V|-1} dist(v,k)$
   endfor
5. *False.* Consider directed graph $G = (V,E)$ with $V = \{s,a\}$ and $E = \{(s,a)\}$ and $l(s,a) = 1$. When $k = 0$, we have $dist(s,0) = 0$ and $dist(a,0) = \infty$.
   When $k = 1$, we have $dist(s,1) = \infty$ and $dist(a,1) = 1$.
   When $k = 2$, we have $dist(s,2) = \infty$ and $dist(a,2) = \infty$.
   Therefore, we have that the $dist(a,1) \neq dist(a,2)$ but this graph does not contain negative cycles. (In fact, a graph with a single isolated vertex s is a counter-example.)

Problem 6 (20 points). Let $G = (V,E)$ be a directed graph with edge length $l(e) > 0$ for any $e \in E$. Let E' be another set of edges on V with edge length $l'(e') > 0$ for any $e' \in E'$. Let $s,t \in V$. Design an algorithm runs in $O(|V|^2 + |E'|)$ time to find an edge $e' \in E'$ whose addition to G will result in the maximum decrease of the distance from s to t. Explain why your algorithms runs in $O(|V|^2 + |E'|)$ time.

Solution. If the distance from s to t decreases with the addition of $e' = (u,v)$, the new shortest path from s to t will be the concatenation of the shortest path from s to u, the edge $e' = (u,v)$ and the shortest path from v to t. That is: $distance(s,t) = distance(s,u) + l'(e') + distance(v,t)$.

We can determine $distance(s,u)$ for all $u \in V$ by running Dijkstra's algorithm with s as the source vertex. To compute $distance(v,t)$ for all $v \in V$, consider the reverse graph G_R of G. Clearly, a shortest path from v to t in G will have a corresponding shortest path from t to v in G_R. Thus, we can run Dijkstra's

algorithm in G_R with t as the source vertex, which will give us $distance(v,t)$ for all $v \in V$. (This idea is similar to Problem 3 in Homework 6).

After that we iterate over all $e' = (u,v) \in E'$ and find the edge that minimize $distance(s,u) + l'(e') + distance(v,t)$. The total running time is twice Dijkstra plus $O(E')$. If Dijkstra is implemented by array, the total running time would be $O(|V|^2 + |E'|)$.