Quiz #2
Introduction to Algorithms/Algorithms 1
600.363/463

April 7th, 9:00-10:15am

Ethics Statement

I agree to complete this exam without unauthorized assistance from any person, materials, or device.

Signature Date

Problem 1 (20 points)

Define a strongly connected component of a directed graph G.

Let $G = (V, E)$. Let $C \subseteq V$ be any set of vertices. C is a strongly connected component if an only if it is a maximal set of vertices with the property:

$$\forall v_1, v_2 \in C, \quad v_1 \text{ is reachable from } v_2 \text{ and } v_2 \text{ is reachable from } v_1 \qquad (1)$$

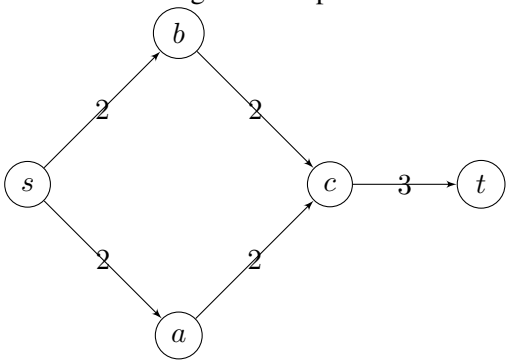
Being maximal means that there is no $v_3 \in \{G \backslash C\}$ such that $\{C + v_3\}$ has property (1).

Problem 2 (20 points)

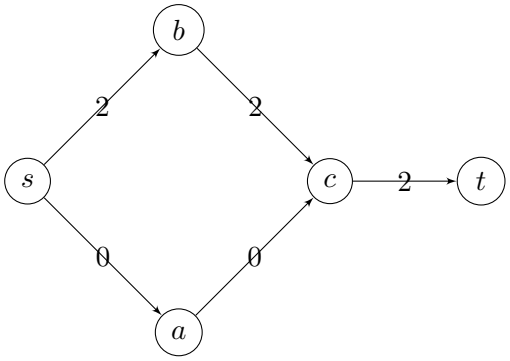
2.1 (10 points)

You are given a flow network represented by a directed graph $G = (V, E)$. Suppose that $f : V \times V \to \mathbb{R}$ and $g : V \times V \to \mathbb{R}$ are valid flow functions. For each of the next two statements write if it is correct or not. A counter-example or short explanation will be sufficient.

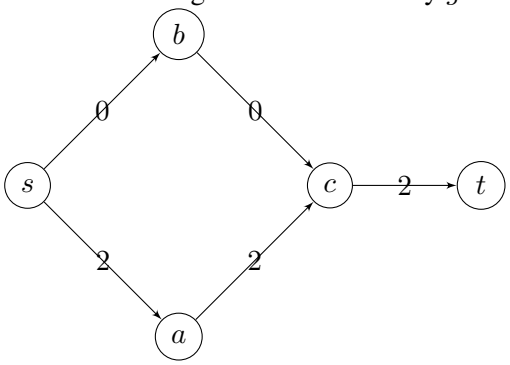
1. Let $h : V \times V \to \mathbb{R}$ be defined for all u and v as $h(u, v) = \min(f(u, v), g(u, v))$. h is a valid flow function.
   Soln. False. Counterexample:
   Let the following be the capacities of our network:



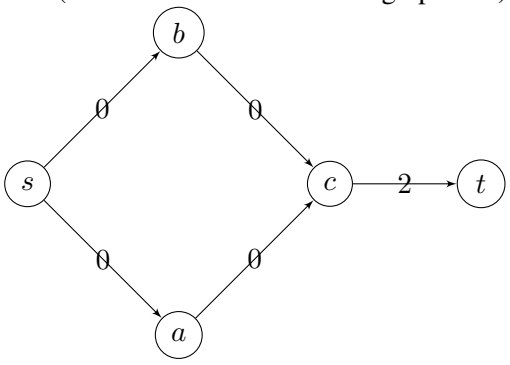
   Let the following be flows defined by f:



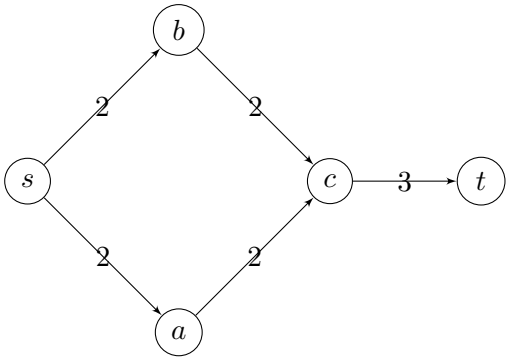
Let the following be flows defined by g:



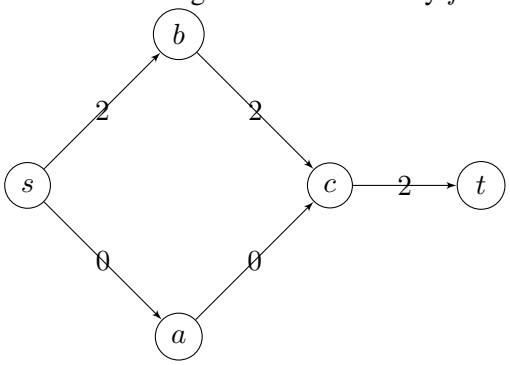
In this case, $\min(f(u,v), g(u,v))$ gives us the following invalid flow function (flow is not conserved through point c):



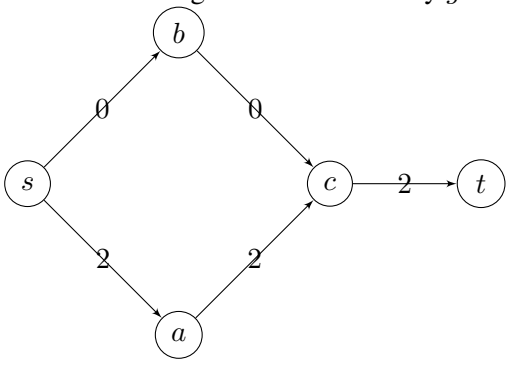
2. Let $h : V \times V \to \mathbb{R}$ be defined for all u and v as $h(u,v) = \max(f(u,v), g(u,v))$. h is a valid flow function.
   Soln. False. Counterexample:
   Let the following be the capacities of our network:



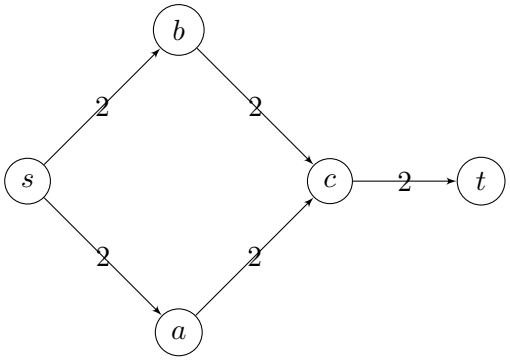
Let the following be flows defined by f:



Let the following be flows defined by g:



In this case, $\max\left(f(u,v), g(u,v)\right)$ gives us the following invalid flow function (flow is not conserved through point c):



2.2 (10 points)

1. Let T be a minimum spanning tree of a weighted graph G. Construct a new graph G' by adding a weight of k to every edge of G. Do the edges of T form a minimum spanning tree of G'? A counter-example or short explanation will be sufficient.

   Soln. True. Let $G = (V, E)$. Every spanning tree of G must have exactly $|V| - 1$ edges, by definition. Let T_i be the i^{th} spanning tree of G. Let w be the function that describes the weight of trees and edges in G. The total weight of T_i is

   $$w(T_i) = \sum_{e \in T_i} w(e) \tag{2}$$

   In G', each edge increased in weight by k. Let w' be the function that describes the weight of trees and edges in G'. It follows that the weight of T_i in G' is

   $$w'(T_i) = \sum_{e \in T_i} (w(e) + k) = \sum_{e \in T_i} w(e) + \sum_{e \in T_i} k \tag{3}$$

   which simplifies to:

   $$w'(T_i) = k(|V| - 1) + \sum_{e \in T_i} w(e) \tag{4}$$

   Now, consider T, the minimum spanning tree of G. By the properties of the minimum spanning tree, we know that

   $$w(T) \leq w(T_i) \tag{5}$$

   and therefore

   $$\sum_{e \in T} w(e) \leq \sum_{e \in T_i} w(e) \tag{6}$$

We can add $k(|V|-1)$ to each side to get

$$k(|V|-1)+\sum_{e\in T} w(e) \leq k(|V|-1)+\sum_{e\in T_i} w(e) \tag{7}$$

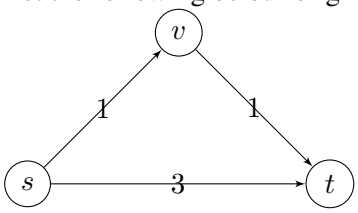
Using equation (4) above, this simplifies to

$$w'(T) \leq w'(T_i) \tag{8}$$

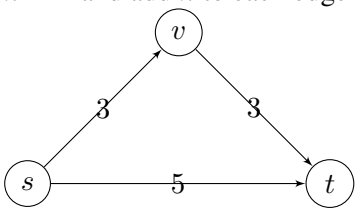
In other words, if T is an MST in G, then T is an MST in G'.

2. Let $P = \{s, ..., t\}$ describe a shortest weighted path between vertices s and t of a weighted graph G. Construct a new graph G' by adding a weight of k to every edge of G. Does P describe a shortest path from s to t in G'? A counter-example or short explanation will be sufficient.
Soln. False. Counterexample:

Let the following be our original graph:



The shortest s, t path in this graph is $s \to v \to t$ with weight 2. Now let $k = 2$ and add k to each edge in G, giving us this graph:



The shortest s, t path in this graph is $s \to t$ with weight 5.

Problem 3 (60 points)

You are given a **weighted directed acyclic** graph $G = (V, E)$ and two vertices s and t which belongs to G. Develop an algorithm which finds a shortest path from s to t or return 0 if there is no path from s to t. Your algorithm should work in time $O(|V| + |E|)$. Prove correctness of the algorithm and provide running time analysis.

Let $w(u, v)$ be the weight of the edge from u to v.
SHORTEST-PATH(G, s, t):

1. Use **DFS** to topologically sort G.
2. **for** each vertex $v \in V$:

   $v.d = \infty$

   $v.\pi =$ **null**
3. $s.d = 0$
4. **for** each vertex $u \in V$, taken in topologically sorted order:

   for each vertex $v \in V$ that is adjacent to u:

   if $v.d > u.d + w(u, v)$:

   $v.d = u.d + w(u, v)$

   $v.\pi = u$
5. PRINT-PATH(G, s, t)

PRINT-PATH(G, s, t):

1. **if** $s == t$:

   print s
2. **else if** $v.\pi ==$ **null**:

   print no path from s to t exists
3. **else**

   PRINT-PATH($G, s, t.\pi$)

   print t

CORRECTNESS
From textbook:
We first show that $v.d = \delta(s, v)$ for all vertices $v \in V$ at termination. If v is not reachable from s, then $v.d = \delta(s, v) = \infty$ by the no-path property. Now suppose that v is reachable from s, so that there is a shortest path $p = (v_0, v_1, \ldots, v_k)$ where $v_0 = s$ and $v_k = v$. Because we process the vertices in topologically sorted order,

we relax the edges on p in the order (v_0, v_1), (v_1, v_2), $\ldots$, (v_{k-1}, v_k). The path-relaxation property implies that $v_i.d = \delta(s, v_i)$ at termination for $i = 0, 1, \ldots, k$. Finally, by the predecessor-subgraph property, G_π is a shortest-paths tree.

No-path property If there is no path from s to v, then we always have $v.d = \delta(s, v) = \infty$.

Path-relaxation property If $p = (v_0, v_1, \ldots, v_k)$ is a shortest path from $s = v_0$ to v_k, and we relax (examine) the edges of p in the order $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$, then $v_k.d = \delta(s, v_k)$. This property holds regardless of any other relaxation steps that occur, even if they are intermixed with relaxations of the edges of p.

Predecessor-subgraph property Once $v.d = \delta(s, v)$ for all $v \in V$, the predecessor subgraph is a shortest-paths tree rooted at s.

RUNTIME
Step (1) is $O(|V| + |E|)$. Step (2) is $(O|V|)$. Step (3) is constant time. Step (4) is proportional in time to the total number of edges, because each edge is examined at most twice (once when each of its endpoints are examined). Therefore step (4) is $O(|V| + |E|)$. Step 5 is $O(|E|)$. The sum of these steps is $O(|V| + |E|)$ as desired.

Problem 4 (50 points)

Devise the following algorithm.
Input:

- weighted graph $G = (V, E)$,
- T — minimum spaning tree of G,
- edge $e \in E$ such as e is not in T.

Output:

- the smallest change in the weight of e, that would cause T not to be an MST of G anymore.

Prove correctness and provide running time analysis. Full score will be given for the algorithm that works in time $O(|V| + |E|)$.

ALGORITHM
Let $w(e)$ be the weight of e.
Let $e = (x, y)$, that is, x and y are the endpoints of e.

SMALLEST-CHANGE(G, T, e):

1. Perform a **BFS** in T from source y and find a path to x. Call this path p.
2. Scan the edges of p and select the the edge of maximum weight. Call this edge e'.
3. **return** $w(e') - w(e) - 1$

CORRECTNESS

There must be a path $p : y \to x$ in T because T is fully connected and contains all of the same vertices as G. The concatenation of p and e forms a cycle. Call this cycle C. It must be true that the maximum weight edge in any cycle is not part of any MST. Therefore, if e is not the maximum weight edge in C, then some other edge, $e' \in T$, is the maximum weight edge. This makes T an invalid MST. Therefore, T is an invalid MST if the following statement holds:

$$w(e) < w(e') \tag{9}$$

We are making a change of size ϵ, so we want to find the smallest value of ϵ for which the following is true:

$$w(e) + \epsilon < w(e') \tag{10}$$

$$\epsilon < w(e') - w(e) \tag{11}$$

This value is:

$$\epsilon = w(e') - w(e) - 1 \tag{12}$$

RUNTIME

Step (1) takes $O(|V| + |E|)$. Step (2) takes $O(|E|)$. Step (3) is constant time. The sum of these 3 steps is $O(|V| + |E|)$ as desired.