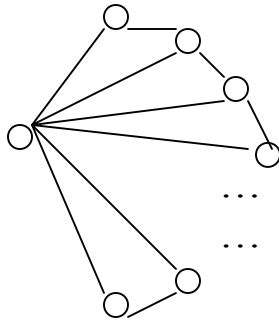


Solution Set for Homework #1

Problem 1

Here's a worst-case example for an n -node graph:

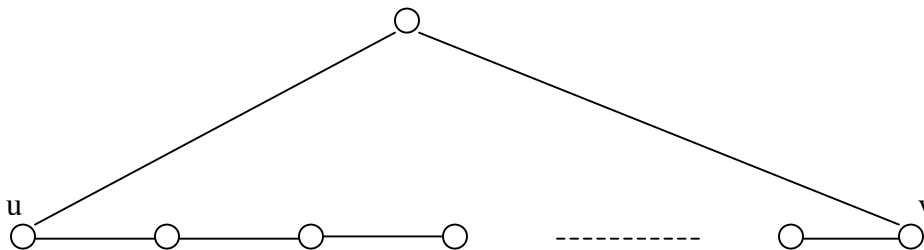


Suppose the cost of each node is 1. Then we get $d=2$ and $d'=n-2$.

Note: You will get full credit if the family of graphs you give satisfies $d=O(1)$ and $d'=\Omega(n)$.

Problem 2

Recall an “overcharge” is a payment to a node that is (nontrivially) higher than the cost incurred by the node. We can construct the following worst-case example for an n -node graph:



Suppose that the cost of the node at the top is L , where $L > n$, and that the cost of each other node is 1. Then the payment to each node on the u - v path is $L-(n-4)$, *i.e.*, the overcharge of each vertex is $L-(n-3)$. Therefore the overall overcharge of the u - v path is $(n-3)(L-(n-3))$. For any n , the overcharge can be arbitrarily large if we make L sufficiently large.

Note: You will get full credit if every member of the family of graphs you give can have an arbitrarily large overcharge.

Problem 3

Reasons include, but are not limited to:

- In practice, a router chooses only the next hop, not the whole path. More generally, a distributed computational model is needed for a realistic study of interdomain routing, but Roughgarden-Tardos uses a centralized computational model.
- Roughgarden-Tardos deals with splittable flows. In practice, IP flows generally are not split; each flow is sent along a single path from source to destination.
- Latency functions may be unknown and/or constantly changing.

Problem 4

When G is directed, V_y is an MST with destination y but not one with source y . As explained in Section 7 of the Hershberger-Suri paper [HS], computing the set Y in the directed case requires first reversing the direction of every edge in E . Even after the shortest-path trees X and Y have been computed, finding the x - y distance omitting each edge on $\text{path}(x,y)$ is still more complicated than in the undirected case, because Lemma 1 does not hold for directed networks. See Figure 5 of [HS] for a counterexample.

Problem 5

First, a couple of notational changes: Denote the block number of vertex v by $b(v)$. Denote the left (respectively, right) vertex of edge e by $l(e)$ (respectively, $r(e)$).

For an undirected network, the preprocessing steps are:

- Compute the shortest path from x to v , for each $v \in V$, by building the shortest-path tree X .
- Compute the shortest path from y to v , for each $v \in V$, by building the shortest-path tree Y .
- Compute $b(v)$, for each $v \in V$, using a depth-first traversal of X .
- For each $e \in E$, if $b(l(e)) < b(r(e))$, put e into $L[b(l(e))]$ and $R[b(r(e))]$.

The time complexity is $O(n \log n + m)$.

For a directed network, the preprocessing steps are:

- Compute the shortest path from x to v , for each $v \in V$, by building the shortest-path tree X .
- Compute the shortest path from y to v , for each $v \in V$, by building the shortest-path tree Y .
- Compute $b(v)$, for each $v \in V$, using a depth-first traversal of X .
- Compute $\text{minblock}(v)$ for each $v \in V$, using a preorder traversal of Y .

- For each $e \in E$, if $b(l(e)) < \text{minblock}(r(e))$, put e into $L[b(l(e))]$ and $R[\text{minblock}(r(e))]$.

The time complexity is also $O(n \log n + m)$.

See Equation (4) on page 9 of [HS] for the (directed-network case) formula for w .

Problem 6

Let the graph be G . For each edge, define its cost as its (reported) weight. Denote the MST of G by MST_G . (If MSTs are not unique, we can fix a deterministic procedure for choosing one.) The mechanism computes MST_G based on the reported weights. The payment to edge e is

$$p^e = \sum_{e' \in \text{MST}_{G-e}} c_{e'} - \sum_{e' \in \text{MST}_G \text{ \& } e' \neq e} c_{e'}$$

where c_e is the cost reported by e .

This mechanism belongs to the VCG family, and thus it is strategyproof.

Problem 7

Centralized Case:

Note that we can compute MST_G with Kruskal's algorithm. Hence, the only nontrivial computational task is to compute MST_{G-e} efficiently, for each e in MST_G .

Denote the set of vertices by V . If we delete e from MST_G , V breaks into two subsets, V_1 and V_2 . Let E_{V_1, V_2} denote the set of edges between V_1 and V_2 . Then $\text{MST}_{G-e} = \text{MST}_G - \{e\} \cup \{e'\}$, where e' is a min-cost edge in E_{V_1, V_2} . Suppose that we have a way to find $c_{e'}$ and that the edges have been sorted by cost. Then we can use binary search to find e' in time $O(\log m)$.

But how can we find $c_{e'}$? Note that $c_{e'} - c_e$ is an upper bound on how much the cost of e can increase without changing the MST. The problem of finding such bounds (for *all* edges) is called "sensitivity analysis." Dixon, Rauch, and Tarjan provide an $O(m)$ -time algorithm for sensitivity analysis in the following paper:

B. Dixon, M. Rauch, and R. Tarjan. "Verification and Sensitivity Analysis of Minimum Spanning Trees in Linear Time," *SIAM Journal on Computing* **21** (1992), 1184-1192.

Because the number of edges in MST_G is $n-1$, the overall time complexity is $O(n \log m)$.

Distributed Case:

Distributed algorithms for MST construction and sensitivity analysis are still an active area of research; the purpose of this HW question was basically to get you to think about the problem. If you are interested and would like to look into it more deeply, see, *e.g.*,

S. Kutten and A. Porat, "Maintenance of a Spanning Tree in a Dynamic Network," in Proceedings of DISC '99. <http://iew3.technion.ac.il/~kutten/kporat99.ps>