

Problem 1: Consider an undirected graph $G = (V, E)$ with distinguished vertices s, t .

(a)[5 marks] Express the $s - t$ max flow problem as a linear program. Assume that all edge capacities are 1.

Assume E contains (i, j) and (j, i) for every undirected edge (i, j) .

$0 \leq f_{ij} \leq 1$ for all $(i, j) \in E$. 2 marks

$\sum_{j|(i,j) \in E} f_{ij} - \sum_{j|(j,i) \in E} f_{ji} = 0$ for $i \neq s, t$. 2 marks

$\max \sum_{j|(s,j) \in E} f_{sj} - \sum_{j|(j,s) \in E} f_{js}$ 1 mark

(b)[8 marks] Consider the problem of finding a shortest path from s to t . Express this as an ILP (integer linear program). Hint: Modify the flow formulation above so that the solution to the ILP will have unit flow in one of the shortest paths.

The first two sets of inequalities remain. 1 mark

Require that unit flow goes out of s , and unit enters t . 2 marks.

Also require f_{ij} to be 0 or 1. 1 mark

Finally, minimize $\sum_{i,j} f_{ij}$. 4 marks

(c)[12 marks] Suppose we are given an additional distinguished vertex u . Write an ILP which finds the smallest subgraph connecting s, t, u . Hint: This is a bit hard. Following the ideas of part (b), first get an ILP which checks whether there exists a subgraph connecting together s, t, u .

Write the constraints of the ILP for finding a shortest s, t path. Write the constraints of the ILP for finding a shortest s, u path. 5 marks.

Let f, g denote the flow variables in the two cases. Define variables h_{ij} , adding constraints $h_{ij} \geq f_{ij}$ and $h_{ij} \geq g_{ij}$. Minimize $\sum_{i,j} h_{ij}$. 7 marks.

Quite likely, there will be no partial credit.

Problem 2: In the Steiner tree problem you are given an undirected graph $G = (V, E)$ and a subset $W \subseteq V$ of vertices. You are required to find a tree t such that (1) t is a subgraph of G , (2) t contains W . Such a tree is called a Steiner tree for W . The size of a Steiner tree is the number of edges in it. You are required to construct a Steiner tree of minimum size. (As an example, note that part (c) of the previous problem was about finding a Steiner tree for $W = \{s, t, u\}$.)

(a)[5 marks] Give a decision version of this problem.

Input: $G=(V,E), W, k$. k is the target number of edges. Output: true iff such a tree with at most k edges exists. 5 marks, no partial credit.

(b)[5 marks] Show that the decision version is in NP.

$B(V,E,W,k,T$: subset of edges){Checks that T forms a tree, 1 mark and it has at most k edges, 2 marks and it spans W . 2 marks}

(c)[15 marks] Show that this problem is NP-hard. Hint: use a reduction from set cover.

Set cover instance is a bipartite graph S,U,M , where S is the set of subsets, U the universe, and M denotes membership.

Add a new node q with edges to S .

Demand Steiner tree for this graph with $W = \{q\} \cup U$. Target size = $k + |U|$. 6 marks.

SC has a solution $S' \Rightarrow$ pick edges to S' from q . Also pick those edges which certify the cover, i.e. Pick the first edge out of S' that covers each element of U . Thus we have a Steiner tree of $k + |U|$ edges. 3 marks.

ST has a solution $T \Rightarrow$ Consider the path from a vertex u in U to q . Suppose the path has edge length > 2 , and let s be the first vertex on the path starting from u , and u' the next vertex. Suppose we now remove the edge (s,u') and add (q,s) . The new tree T' has the same number of edges, and spans exactly the same set of vertices. But now u has a path of length 2 to q . In this way we can ensure that every u will have a path of length 2 to q . Now retain only the tree induced by these paths. This can only reduce the number of edges, and ensure that the degree of every u is exactly 1. Now the number of edges between S and U must be exactly $|U|$, and hence that between q and S at most k . Thus at most k vertices in S are in T . These k vertices correspond to the sets to select. 6 marks

Problem 3: In this we will develop an approximation algorithm for finding an approximate optimal Steiner tree. Let a circuit in G be called a Steiner circuit for W if it starts at some vertex in W , passes through all other vertices in W at least once and returns to the starting vertex. In this process, it may visit vertices of G not in W as many times as needed. The size of a Steiner circuit is the total number of edges traversed, with each edge counted as many times as it is traversed. Let C denote a smallest Steiner circuit for W . Let T denote an optimal Steiner tree. For an object X we will use $|X|$ to denote its size.

(a)[5 marks] Show that $2|T| \geq |C|$.

A dfs of a Steiner tree is also a Steiner circuit. – it visits every vertex traversing every edge twice. Thus an optimal Steiner circuit can only be smaller, if at all, than the dfs of any Steiner tree, including the optimal. 5 marks, no partial credit

Let H denote the metric closure of G , i.e. H is the complete (weighted) graph on V and each edge (u,v) in H has a length d_{uv} = the length of a shortest path between u,v in G . One such shortest path is designated as the image of the edge (u,v) of H .

(b)[5 marks] Draw the metric closure of the graph whose edge set is $\{(1,2), (2,3), (3,1), (3,4)\}$. Also draw the graph itself and mark the image of the edge $(1,4)$.

H: distances from 1,2 and 4: 2. 2 marks

From 3: 1 to all. 1 marks.

Image of $(1,4)$: path 1-3-4. 2 marks.

Consider now the subgraph H_W of H induced by W , i.e. include only the edges of H going between the vertices in W . Let M denote a minimum spanning tree of H_W , with $|M|$ = the sum the lengths of the edges in M . Let D be an optimal TSP tour of H_W , with $|D|$ = length of the tour. Consider appropriate images to show the following.

(c)[5 marks] Show that $|C| = |D|$.

If we traverse G in the order of the vertices in D using shortest paths, we get a circuit of the same size. Consider the optimal circuit, with the first occurrences of the vertices in W . The path connecting these occurrences must be the shortest, otherwise we could have shrunk it further. But then this corresponds to a tour in H_W .

(d)[5 marks] Show how to construct a Steiner tree T' for W in G such that $|M| \geq |T'|$.

Consider the graph obtained by taking the union of the images of the edges of M . This has size at most $|M|$. This graph spans W . Removing some edges we will get a steiner tree.

(e)[5 marks] Give a 2 approximation algorithm for finding a Steiner tree for W . Based on the previous parts, give a short proof of correctness.

Find an MST M for H_W . Find T' as above. 2 marks.

Note that $|D| \geq |M|$. 2 marks

Thus we have $2|T| \geq |C| = |D| \geq |M| \geq |T'|$. 1 mark.

Problem 4: Consider the following variation of the contention resolution problem discussed in class. In this all n processors start at the same time and execute the following code:

Repeat {Access the database with probability $1/n$. If success, stop.}

As before, a processor that accesses succeeds if no other processor accesses in that step. What we discussed in class had processors not stopping after the first success.

(a)[15 marks] Define the i th period to mean the steps after some $i - 1$ processors have succeeded and until the i th success (inclusive). What is the probability of success at each step of the i th period? Let T_i denote the number of steps in the i th period. Show that $E[T_i] = \theta(n/(n - i + 1))$

The probability of a success in the i th period is $(n - i + 1)(1 - p)^{n-i}p$, with $p = 1/n$. We know that $1/e \leq (1 - 1/n)^n \leq (1 - 1/n)^{n-i} \leq 1$. Hence the probability of a success in the i th period is $\theta((n - i + 1)/n)$.

Thus the expected number of steps needed for the period is $\theta(n/(n - i + 1))$.

5 marks for the probability expression. 5 marks for the algebra + 5 for expectation.

(b)[5 marks] What is the expected time for all processors to succeed once?

$\sum_{i=1}^n \theta(n/n - i + 1) = n\theta(1/n + 1/(n - 1) + \dots + 1/2 + 1) = \theta(n \ln n)$ 5 marks for $n \ln n$. No partial credit.

(c)[5 marks] Based on above, argue that expected time for all processors to succeed at least once, in the original problem, i.e. processors continue to attempt to access even after successes, is also $\Omega(n \ln n)$. In class we only showed (essentially) $O(n \ln n)$.

If all processors are active throughout, the processors that have not succeeded will have less probability of success because previously successful processors will also be contending. Hence each period will only be longer.

Problem 5: Consider the following modified algorithm in which the probability of access is increased with time, and as before, each processor stops after one success.

1. For $i = \log n$ down to $\log(n/\log n)$: (i th iteration is called **phase i**)

(a) For $j = 1$ to $10 \cdot 2^i$

Attempt to access with probability 2^{-i} , if successful, stop.

(b) end for

2. end for

(a)[5 marks] Call our algorithm *system A*. Suppose at the beginning of phase i we have $n_A \leq 2^i$ active processors. Consider a *system B* in which there are only 2^{i-1} processors, but the processors do not stop after the first success, they continue to attempt to access the database, as in the algorithm studied in class. Processors make accesses with probability 2^{-i} in both systems. Show that in each step that A has more active processors than B , A has higher probability of success.

The probability of success with j processors is $P_j = j(1-p)^{j-1}p$, for $p = 1/2^i$. $P_{j+1} = P_j \frac{j+1}{j} \frac{2^i-1}{2^i} \geq 1$, since $j+1 \leq n_A \leq 2^i$. Thus in each step the system with more processors (system A) has higher probability of success.

(b)[10 marks] Say phase i of A fails if the number of active processors remains above 2^{i-1} assuming it started no higher than 2^i . If phase i of A fails, what must happen in system B? Use this relationship to prove that phase i fails with probability at most $\exp(-2^i c)$ for some constant c . Hint: Steps of system B are independent and so easier to analyze.

While A has more processors than B, it has greater probability of success. If A fails, then it has greater number of processors than B, and it succeeded at most 2^{i-1} times. Thus B must also have succeeded at most 2^{i-1} times. So we simply estimate the probability that B succeeds so many times. 3 marks.

In B, success in each step is independent. Writing $m = 2^i$, the probability of success is $1/2(1 - 1/m)^{m/2} \geq 1/2e$. Thus the expected number of successes is at least $5m/e$. 3 marks.

The probability that there are at most $m/2$ successes is $\exp(-\epsilon^2 \mu/2)$, where $\mu \geq 5m/e$. $\mu(1 - \epsilon) = m/2$, i.e. $1 - \epsilon = e/10$, i.e. $\epsilon = 1 - e/10 = 0.73$. Thus the probability is $\exp(-0.73^2 5m/2/e) \leq \exp(-m/e)$. 4 marks.

(c)[5 marks] Show that with probability at least $1 - 1/n$ the number of active processors will reduce to at most $n/\log n$ at the end of the execution.

The probability of failure for some phase is $\sum_i \exp(-2^i/e) \leq (\log(n/\log n)) \exp(-n/e \log n)$

This is clearly smaller than $1/n$.

(d)[5 marks] What can you add to the algorithm so that in an additional $O(n)$ steps all active processors will have one success with probability at least $1 - 1/n$? Note that the overall time will thus be $O(n)$.

Now use the algorithm described earlier, or in class, whereby in $m \ln m$ steps all m processors finish. Choosing $m = n/\log n$ we get $m \ln m = (n/\log n) \ln(n/\log n) = O(n)$.

Suppose a random variable $X = X_1 + X_2 + \dots + X_n$, where X_i are *independent Bernoulli random variables* with $\Pr(X_i = 1) = p_i$. Let $\mu = E[X] = p_1 + p_2 + \dots + p_n$. Then,

$$\Pr(X \geq \beta\mu) \leq e^{(1 - \frac{1}{\beta} - \ln \beta)\beta\mu} \leq \left(\frac{\beta}{e}\right)^{-\beta\mu} \quad \beta > 0$$

$$\Pr(X \leq (1 - \epsilon)\mu) \leq e^{-\epsilon^2 \mu/2} \quad 0 < \epsilon$$