CS 373: Combinatorial Algorithms, Fall 2002
Homework 4, due Thursday, October 31, 2002 at 23:59.99

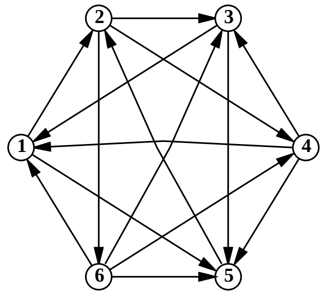
Name:	
Net ID:	Alias:
Name:	
Net ID:	Alias:
Name:	
Net ID:	Alias:
Undergrads	Grads

This homework is to be submitted in groups of up to three people. Graduate and undergraduate students are *not* allowed to work in the same group. Please indicate above whether you are undergraduate or graduate students. Only *one* submission per group will be accepted.

Required Problems

1. Tournament:
   A *tournament* is a directed graph with exactly one edge between every pair of vertices. (Think of the nodes as players in a round-robin tournament, where each edge points from the winner to the loser.) A *Hamiltonian path* is a sequence of directed edges, joined end to end, that visits every vertex exactly once.

   Prove that every tournament contains at least one Hamiltonian path.



A six-vertex tournament containing the Hamiltonian path $6 \to 4 \to 5 \to 2 \to 3 \to 1$.

2. Acrophobia:
   Consider a graph $G = (V, E)$ whose nodes are cities, and whose edges are roads connecting the cities. For each edge, the weight is assigned by h_e, the maximum altitude encountered when traversing the specified road. Between two cities s and t, we are interested in those paths whose maximum altitude is as low as possible. We will call a subgraph, G', of G an *acrophobic friendly subgraph*, if for any two nodes s and t the path of minimum altitude is always

included in the subgraph. For simplicity, assume that the maximum altitude encountered on each road is unique.

(a) Prove that every graph of n nodes has an acrophobic friendly subgraph that has only $n-1$ edges.

(b) Construct an algorithm to find an acrophobic friendly subgraph given a graph $G = (V, E)$.

3. Refer to the lecture notes on single-source shortest paths. The GenericSSSP algorithm described in class can be implemented using a stack for the 'bag'. Prove that the resulting algorithm, given a graph with n nodes as input, could perform $\Omega(2^n)$ relaxation steps before stopping. You need to describe, for any positive integer n, a specific weighted directed n-vertex graph that forces this exponential behavior. The easiest way to describe such a family of graphs is using an *algorithm*!

4. Neighbors:
Two spanning trees T and T' are defined as *neighbors* if T' can be obtained from T by swapping a single edge. More formally, there are two edges e and f such that T' is obtained from T by adding edge e and deleting edge f.

(a) Let T denote the minimum cost spanning tree and suppose that we want to find the second cheapest tree T' among all trees. Assuming unique costs for all edges, prove that T and T' are neighbors.

(b) Given a graph $G = (V, E)$, construct an algorithm to find the second cheapest tree, T'.

(c) Consider a graph, H, whose vertices are the spanning trees of the graph G. Two vertices are connected by an edge if and only if they are neighbors as previously defined. Prove that for any graph G this new graph H is connected.

5. Network Throughput:
Suppose you are given a graph of a (tremendously simplified) computer network $G = (V, E)$ such that a weight, b_e, is assigned to each edge representing the communication bandwidth of the specified channel in Kb/s and each node is assigned a value, l_v, representing the server latency measured in seconds/packet. Given a fixed packet size, and assuming all edge bandwidth values are a multiple of the packet size, your job is to build a system to decide which paths to route traffic between specfied servers.
More formally, a person wants to route traffic from server s to server t along the path of maximum throughput. Give an algorithm that will allow a network design engineer to choose an optimal path by which to route data traffic.

6. All-Pairs-Shortest-Path:
[This problem is required only for graduate students taking CS 373 for a full unit; anyone else can submit a solution for extra credit.]
Given an undirected, unweighted, connected graph $G = (V, E)$, we wish to solve the distance version of the all-pairs-shortest-path problem. The algorithm APD takes the $n \times n$ 0-1 adjacency matrix A and returns an $n \times n$ matrix D such that d_{ij} represents the shortest path between vertices i and j.

```
APD($A$)
    $Z \leftarrow A \cdot A$
    let $B$ be an $n \times n$ matrix, where $b_{ij} = 1$ iff $i \neq j$ and $(a_{ij} = 1$ or $z_{ij} > 0)$
    if $b_{ij} = 1$ for all $i \neq j$
        return $D \leftarrow 2B - A$
    $T \leftarrow APD(B)$
    $X \leftarrow T \cdot A$
    foreach $x_{ij}$
        if $x_{ij} \geq t_{ij} \cdot degree(j)$
            $d_{ij} \leftarrow 2t_{ij}$
        else
            $d_{ij} \leftarrow 2t_{ij} - 1$
    return $D$
```

(a) In the APD algorithm above, what do the matrices Z, B, T, and X represent? Justify your answers.

(b) Prove that the APD algorithm correctly computes the matrix of shortest path distances. In other words, prove that in the output matrix D, each entry d_{ij} represents the shortest path distance between node i and node j.

(c) Suppose we can multiply two $n \times n$ matrices in $M(n)$ time, where $M(n) = \Omega(n^2)$.[1] Prove that APD runs in $O(M(n) \log n)$ time.

[1] The matrix multiplication algorithm you already know runs in $O(n^3)$ time, but this is not the fastest known. The current record is $M(n) = O(n^{2.376})$, due to Don Coppersmith and Shmuel Winograd. Determinign the smallest possible value of $M(n)$ is a long-standing open problem.

Practice Problems

1. Makefiles:
   In order to facilitate recompiling programs from multiple source files when only a small number of files have been updated, there is a UNIX utility called 'make' that only recompiles those files that were changed after the most recent compilation, *and* any intermediate files in the compilation that depend on those that were changed. A Makefile is typically composed of a list of source files that must be compiled. Each of these source files is dependent on some of the other files which are listed. Thus a source file must be recompiled if a file on which it depends is changed.

   Assuming you have a list of which files have been recently changed, as well as a list for each source file of the files on which it depends, design an algorithm to recompile only those necessary. DO NOT worry about the details of parsing a Makefile.

2. The incidence matrix of an undirected graph $G = (V, E)$ is a $|V| \times |E|$ matrix $B = (b_{ij})$ such that
   $$b_{ij} = \begin{cases} 1 & \text{if vertex } v_i \text{ is an endpoint of edge } e_j \\ 0 & \text{otherwise} \end{cases}$$

   (a) Describe what all the entries of the matrix product BB^T represent (B^T is the matrix transpose). Justify.

   (b) Describe what all the entries of the matrix product B^TB represent. Justify.

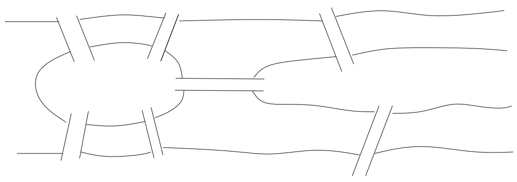
   *(c) Let $C = BB^T - 2A$. Let C' be C with the first row and column removed. Show that $\det C'$ is the number of spanning trees. (A is the adjacency matrix of G, with zeroes on the diagonal).

3. Reliable Network:
   Suppose you are given a graph of a computer network $G = (V, E)$ and a function $r(u, v)$ that gives a reliability value for every edge $(u, v) \in E$ such that $0 \le r(u, v) \le 1$. The reliability value gives the probability that the network connection corresponding to that edge will *not* fail. Describe and analyze an algorithm to find the most reliable path from a given source vertex s to a given target vertex t.

4. Aerophobia:
   After graduating you find a job with Aerophobes-R$'$-Us, the leading traveling agency for aerophobic people. Your job is to build a system to help customers plan airplane trips from one city to another. All of your customers are afraid of flying so the trip should be as short as possible.

   In other words, a person wants to fly from city A to city B in the shortest possible time. S/he turns to the traveling agent who knows all the departure and arrival times of all the flights on the planet. Give an algorithm that will allow the agent to choose an optimal route to minimize the total time in transit. Hint: rather than modify Dijkstra's algorithm, modify the data. The total transit time is from departure to arrival at the destination, so it will include layover time (time waiting for a connecting flight).

5. The Seven Bridges of Königsberg:
During the eighteenth century the city of Königsberg in East Prussia was divided into four sections by the Pregel river. Seven bridges connected these regions, as shown below. It was said that residents spent their Sunday walks trying to find a way to walk about the city so as to cross each bridge exactly once and then return to their starting point.



   (a) Show how the residents of the city could accomplish such a walk or prove no such walk exists.

   (b) Given any undirected graph $G = (V, E)$, give an algorithm that finds a cycle in the graph that visits every edge exactly once, or says that it can't be done.

6. Given an undirected graph $G = (V, E)$ with costs $c_e \geq 0$ on the edges $e \in E$ give an $O(|E|)$ time algorithm that tests if there is a minimum cost spanning tree that contains the edge e.

7. Combining Boruvka and Prim:
Give an algorithm that find the MST of a graph G in $O(m \log \log n)$ time by combining Boruvka's and Prim's algorithm.

8. Minimum Spanning Tree changes:
Suppose you have a graph G and an MST of that graph (i.e. the MST has already been constructed).

   (a) Give an algorithm to update the MST when an edge is added to G.

   (b) Give an algorithm to update the MST when an edge is deleted from G.

   (c) Give an algorithm to update the MST when a vertex (and possibly edges to it) is added to G.

9. Nesting Envelopes
You are given an unlimited number of each of n different types of envelopes. The dimensions of envelope type i are $x_i \times y_i$. In nesting envelopes inside one another, you can place envelope A inside envelope B if and only if the dimensions A are *strictly smaller* than the dimensions of B. Design and analyze an algorithm to determine the largest number of envelopes that can be nested inside one another.

10. $o(V^2)$ Adjacency Matrix Algorithms

   (a) Give an $O(V)$ algorithm to decide whether a directed graph contains a *sink* in an adjacency matrix representation. A sink is a vertex with in-degree $V - 1$.

(b) An undirected graph is a scorpion if it has a vertex of degree 1 (the sting) connected to a vertex of degree two (the tail) connected to a vertex of degree $V-2$ (the body) connected to the other $V-3$ vertices (the feet). Some of the feet may be connected to other feet.

Design an algorithm that decides whether a given adjacency matrix represents a scorpion by examining only $O(V)$ of the entries.

(c) Show that it is impossible to decide whether G has at least one edge in $O(V)$ time.

11. Shortest Cycle:
Given an **undirected** graph $G = (V, E)$, and a weight function $f : E \to \mathbf{R}$ on the **edges**, give an algorithm that finds (in time polynomial in V and E) a cycle of smallest weight in G.

12. Longest Simple Path:
Let graph $G = (V, E)$, $|V| = n$. A *simple path* of G, is a path that does not contain the same vertex twice. Use dynamic programming to design an algorithm (not polynomial time) to find a simple path of maximum length in G. Hint: It can be done in $O(n^c 2^n)$ time, for some constant c.

13. Minimum Spanning Tree:
Suppose all edge weights in a graph G are equal. Give an algorithm to compute an MST.

14. Transitive reduction:
Give an algorithm to construct a *transitive reduction* of a directed graph G, i.e. a graph G^{TR} with the fewest edges (but with the same vertices) such that there is a path from a to b in G iff there is also such a path in G^{TR}.

15. (a) What is $5^{2^{29^{5^0}+23^{4^1}+17^{3^2}+11^{2^3}+5^{1^4}}} \bmod 6$?

(b) What is the capital of Nebraska? Hint: It is not Omaha. It is named after a famous president of the United States that was not George Washington. The distance from the Earth to the Moon averages roughly 384,000 km.