CS570

Analysis of Algorithms

Summer 2008

Final Exam

Name: ______________________

Student ID: __________________

____4:00 - 5:40 Section　　　　____6:00 – 7:40 Section

	Maximum	Received
Problem 1	20	
Problem 2	16	
Problem 3	16	
Problem 4	16	
Problem 5	16	
Problem 6	16	
Total	100	

2 hr exam

Close book and notes

1) 20 pts
For each of the following statements, answer whether it is TRUE or FALSE, and briefly justify your answer.

a) If a connected undirected graph G has the same weights for every edge, then every spanning tree of G is a minimum spanning tree, but such a spanning tree cannot be found in linear time.

T

b) Given a flow network G and a maximum flow of G that has already been computed, one can compute a minimum cut of G in linear time.

F

c) The Ford-Fulkerson Algorithm finds a maximum flow of a unit-capacity flow network with n vertices and m edges in time O(mn) if one uses depth-first search to find an augmenting path in each iteration.

T

d) Unless P = NP, 3-SAT has no polynomial-time algorithm.

F

e) The problem of deciding whether a given flow f of a given flow network G is a maximum flow can be solved in linear time.

F

f) If a decision problem A is polynomial-time reducible to a decision problem B (i.e., $A \leq_p B$), and B is NP-complete, then A must be NP-complete.

F

g) If a decision problem B is polynomial-time reducible to a decision problem A (i.e., $B \leq_p A$), and B is NP-complete, then A must be NP-complete.

T

h) Integer max flow (where flows and capacities are integers) is polynomial time reducible to linear programming .

F

i) It has been proved that NP-complete problems cannot be solved in polynomial time.

F

j) NP is a class of problems for which we do not have polynomial time solutions.

T

2) 16 pts

Suppose that we are given a weighted undirected graph G = (V,E), a source s € V , a sink t € V , and a subset W of vertices V, and want to find a shortest path from s to t that must go through at least one vertex in W. Give an efficient algorithm that solves the problem by invoking any given single-source shortest path algorithm as a subroutine a small number (how many?) times. Show that your algorithm is correct.

Min (pathLength(s, w)+ pathLength(w, t)) where w belongs to W

pathLength(i, j) finds the shortest path between I and j. It can be implemented using any existing shortest path algorithm. It will be used 2*size(W) times

16 pts

Suppose that we have n files $F_1, F_2, ..., F_n$ of lengths $l_1, l_2, ... , l_n$ respectively, and want to concatenate them to produce a single file $F = F_1 \circ F_2 \circ ... \circ F_n$ of length $l_1 + l_2 + ... + l_n$. Suppose that the only basic operation available is one that concatenates two files. Moreover, the cost of this basic operation on two files of length l_i and l_j, is determined by a cost function $c(l_i,l_j)$ which depends only on the lengths of the two files. Design an efficient dynamic programming algorithm that given n files $F_1, F_2, ..., F_n$ and a cost function *c(.,.)*, computes a sequence of basic operations that optimizes total cost of concatenating the n input files.

The sub structure of this problem is the time to merge a set of files S. time(S) – the time needed to merge the files and the files in S should be kept for repeated use

```
if size (S) == 2:
    # S1 and S2 are the two files in S
    time (S) = c (length(S1), length(S2))
else:
    # f is a file in S
    # S-f : take f out from S
    time (S) = min (c (length(f), total length of all other files in S)+time (S-f)
```

Return time (S) where S contains all the files

4) 16 pts

Define the language

Double-SAT = { ψ: ψ is a Boolean formula with at least 2 distinct satisfying assignmentg}.

For instance, the formula ψ: (x v y v z) ^ (x′ v y′ V z′) ^ (x′ V y′ V z) is in Double-SAT, since the assignments (x = 1, y = 0, z = 0) and (x = 0, y = 1, z = 1) are two distinct assignments that both satisfy ψ.

Prove that Double-SAT is NP-Complete.

Various methods can be used for reducing SAT to Double-SAT. Since SAT is NP-Complete, Double-SAT is NP complete.

Following is an example for the reduction.

On input $\psi(x_1, \ldots, x_n)$:

1. Introduce a new variable y.
2. Output formula $\psi'(x_1, \ldots, x_n, y) = \psi(x_1, \ldots, x_n)$ ^ (y | not y).

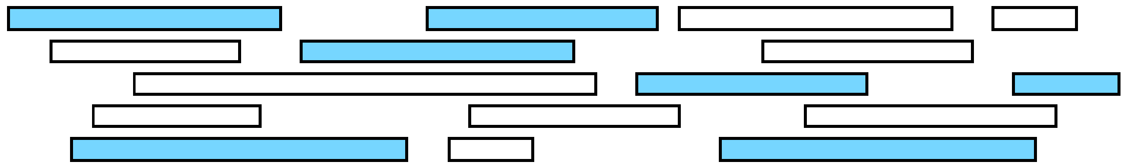
If $\psi(x_1, \ldots, x_n)$ belongs to SAT, then ψ ' has at least 1 satisfying assignment, and therefore $\psi'(x_1, \ldots, x_n, y)$ has at least 2 satisfying assignments as we can satisfy the new clause (y | not y) by assigning either

y = 1 or y = 0 to the new variable y, so $\psi'(x_1, \ldots, x_n, y)$ belongs to Double-SAT. On the other hand, if $\psi(x_1, \ldots, x_n)$ does not belong to SAT, then clearly $\psi'(x_1, \ldots, x_n, y) = \psi(x_1, \ldots, x_n)$ ^ (y | not y) has no satisfying assignment either, so $\psi'(x_1, \ldots, x_n, y)$ does not belong to Double-SAT. Therefore, SAT can be reduced to Double-SAT. Since the above reduction clearly can be done in P time, Double-SAT is NP-Complete.

5) 16 pts

Let X be a set of n intervals on the real line. A subset of intervals $Y \subset X$ is called a tiling path if the intervals in Y cover the intervals in X, that is, any real value that is contained in some interval in X is also contained in some interval in Y. The size of a tiling cover is just the number of intervals.

Describe and analyze an algorithm to compute the smallest tiling path of X as quickly as possible. Assume that your input consists of two arrays $X_L[1 .. n]$ and $X_R [1 .. n]$, representing the left and right endpoints of the intervals in X.


A set of intervals. The seven shaded intervals form a tiling path

Sort the intervals from left to right by the start position, if the start positions are same, then sort it from the left to right by the end position;

```
FOR (i=1;i<=n; i++) do
        Result[i] = positive unlimited ;
        FOR (j = 1;j < i; j ++) do
                IF (interval i overlap with interval j) then
                        IF (Result[i]> result[j]+1) then
                                Result[i] = result[j]+1;
Return result [n]
```

The complexity is O(n^2)

6) 16 pts
An edge of a flow network is called *critical* if decreasing the capacity of this edge results in a decrease in the maximum flow. Give an efficient algorithm that finds a critical edge in a network.

Find a min cut of the network which has the same capacity as the maximum flow. Every outgoing edge from the min cut is a critical edge

Additional Space

Additional Space