CS570
Analysis of Algorithms
Fall 2008
Final Exam

Name: _____

Student ID: _____

____Monday Section ____Wednesday Section ____Friday Section

	Maximum	Received
Problem 1	20	
Problem 2	10	
Problem 3	10	
Problem 4	20	
Problem 5	20	
Problem 6	10	
Problem 7	10	
Total	100	

2 hr exam
Close book and notes

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE]
If NP = P, then all problems in NP are NP hard

[FALSE]
L1 can be reduced to L2 in Polynomial time and L2 is in NP, then L1 is in NP

[FALSE]
The simplex method solves Linear Programming in polynomial time.

[FALSE]
Integer Programming is in P.

[FALSE]
If a linear time algorithm is found for the traveling salesman problem, then every problem in NP can be solved in linear time.

[TRUE]
If there exists a polynomial time 5-approximation algorithm for the general traveling salesman problem then 3-SAT can be solved in polynomial time.

[FALSE]
Consider an undirected graph G=(V, E). Suppose all edge weights are different. Then the longest edge cannot be in the minimum spanning tree.

[FALSE]
Given a set of demands D = *{dv}* on a directed graph *G(V,E)*, if the total demand over *V* is zero, then *G* has a feasible circulation with respect to *D.*

[TRUE]
For a connected graph G, the BFS tree, DFS tree, and MST all have the same number of edges.

[FALSE]
Dynamic programming sub-problems can overlap but divide and conquer sub-problems do not overlap, therefore these techniques cannot be combined in a single algorithm.

Grading Criteria: Pretty clear, each has two point. These T/F are designed and answered by Professor Shamsian

2) 10 pts
Demonstrate that an algorithm that consists of a polynomial number of calls to a polynomial time subroutine could run in exponential time.

Suggested Solution:
Suppose X takes an input size of n and returns an output size of n^2. You call X a polynomial number of times say n. If the size of the original input is n the size of the output will be n^{2n} which is exponential WRT n.

Grading Criteria: Rephrasing the question doesn't get that much. If you show you at least understood polynomial / exponential time, you get some credit.

3) 10 pts

Suppose that we are given a weighted, directed graph $G = (V, E)$ in which edges that leave the source vertex s may have negative weights, all other edge weights are nonnegative, and there are no negative-weight cycles. Argue that Dijkstra's algorithm correctly finds shortest paths from s in this graph.

Suggested solution :

```
DIJKSTRA(G,w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 S ← 0
3 Q ← V[G]
4 while Q ≠ 0
5       do u ← EXTRACT-MIN(Q)
6       S ← S ∪ {u}
7       for each vertex v ∈ Adj[u]
8             do RELAX(u, v,w)

RELAX(u, v,w)
1 if d[v] > d[u] + w(u, v)
2 then d[v] ← d[u] + w(u, v)
3 π[v]← u
```

old new

update with smaller

We have the algorithm as shown above.

We show that in each iteration of Dijkstra's algorithm, d[u]=δ(s,u) when u is added to S (the rest follows from the upper-bound property). Let $N^-(s)$ be the set of vertices leaving s, which have negative weights. We divide the proof to vertices in $N^-(s)$ and all the rest. Since there are no negative loops, the shortest path between s and all $u \in N^-(s)$, is through the edge connecting them to s, hence δ(s,u)=w(s,u), and it follows that after the first time the loop in lines 7,8 is executed d[u]= w(s,u)= δ(s,u) for all $u \in N^-(s)$ and by the upper bound property d[u]= δ(s,u) when u is added to S. Moreover it follows that $S= N^-(s)$ after $|N^-(s)|$ steps of the while loop in lines 4-8.

For the rest of the vertices we argue that the proof of Theorem 24-6 (***Theorem 24.6 - Correctness of Dijkstra's algorithm, Introduction to Algorithem by Cormen)*** holds since every
shortest path from s includes at most one negative edge (and if does it has to be the first one). To see why this is true assume otherwise, which mean that the path contains a loop, contradicting the property that shortest paths do not contain loops.

Grading Criteria : You need to argue, so bringing just one example shouldn't get you the whole credit , but it may did! If you showed you at least understood Dijkstra, you got some credit too.

4) 20 pts

Phantasy Airlines operates a cargo plane that can hold up to 30000 pounds of cargo occupying up to 20000 cubic feet. We have contracted to transport the following items.

Item type	Weight	Volume	Number	Cost if not carried
1	4000	1000	3	$800
2	800	1200	10	$150
3	2000	2200	4	$300
4	1500	500	5	$500

For example, we have contracted 10 items of type 2, each of which weighs 800 pounds and takes up to 1200 cubic feet of space. The last column refers to the cost of subcontracting shipment to another carrier.

For each pound we carry, the cost of flying the plane increases by 5 cents. Which items should we put in the plane, and which should we ship via other carrier, in order to have the lowest shipping cost? Formulate this problem as an integer programming problem.

Suggested Solution:

Assume the data in the table are per item.

Considering x_1, x_2, x_3, x_4 are the items that we will ship for types 1,2,3 and 4 respectively.

It is clear that :

$$0 \leq x_1 \leq 3 \quad \& \quad 0 \leq x_2 \leq 10 \quad \& \quad 0 \leq x_3 \leq 4 \quad \& \quad 0 \leq x_4 \leq 5$$

Weight constraint :

$$x_1 * 4000 + x_2 * 800 + x_3 * 2000 + x_4 * 1500 \leq 30{,}000$$

Volume constraint:

$$x_1 * 1000 + x_2 * 1200 + x_3 * 2200 + x_4 * 500 \leq 20{,}000$$

Cost of shipping :

$$C_{Ship} = C_A + (x_1 * 4000 + x_2 * 800 + x_3 * 2000 + x_4 * 1500) * \$0.05$$

Where C_A is the cost of operating empty cargo plane

Cost of sub-contracting

$$C_{Sub} = (3 - x_1) * 800 + (10 - x_2) * 150 + (4 - x_3) * 300 + (5 - x_{4)} * 500)$$

Out objective is to minimize $C_{Ship} + C_{Sub}$

Grading Criteria: Some credit will be deducted if you missed some optimization part. Making this simple question complicated may result deduction.

5) 20 pts

Suppose you are given a set of n integers each in the range 0…K. Give an efficient algorithm to partition these integers into two subsets such that the difference |S1-S2| is minimized, where S1 and S2 denote the sums of the elements in each of the two subsets.

Proposed Solution :

a[1]...a[m] = the set of integers

opt[i,k] = true if and only if it is possible to sum to k using elements 1...i

```
Initialize opt(i,k) = false for all i,k
for(i=1...n) {
  for(k=1..K) {
    if(opt(i,k) == true) {
    for(j=1..m) {
      opt(i,k + a[j]) = true;
    }
  }
}
for(k = floor(K/2)..1) {
  if(opt(n, k)) {
    Return |K - 2k|;   // Returns the difference. Partition can be found by back tracking
  }
}
```

exactly

假如能加到10了，然后还有2 3 5可以选，那么当然能精确加到12 13 15

假如有多个true,大的优先？？

6) 10 pts

Adrian has many, many love interests. Many, many, many love interests. The problem is, however, a lot of these individuals know each other, and the last thing our polyamorous TA wants is fighting between his hookups. So our resourceful TA draws a graph of all of his potential partners and draws an edge between them if they know each other. He wants at least *k* romantic involvements and none of them must know each other (have an edge between them). Can he create his LOVE-SET in polynomial time? Prove your answer.

Proposed Solution :

This is just Independent Set. Proof of NP-completeness was shown in class lecture. We show the correctness as follows: 1) Any Independent Set of size k on the graph will satisfy the requirement that no two love interests know each other (share an edge). 2) If there is a set of k love interests none of which know each other, then there must be a corresponding set of k vertices, such that no two share an edge (know each other).

7) 10 pts

An edge in a flow network is called a bottleneck if increasing its capacity increases the max flow in the network. Give an efficient algorithm for finding all the bottlenecks in the network.

Proposed Solution

Find max flow and the corresponding residual graph, then for each edge increase its capacity by one unit and see if you find a new path from s to t in the residual graph. (Of course you undo this increase before trying the next edge.) If the flow increases for any such edge then that edge must be a bottleneck.