CS570
Analysis of Algorithms
Summer 2011
Exam II

Name: _____

Student ID: _____

____Check if DEN student

	Maximum	Received
Problem 1	20	
Problem 2	20	
Problem 3	20	
Problem 4	20	
Problem 5	20	
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 except for the question at the bottom of the page.

[TRUE/FALSE]
Dynamic programming is a brute force method.

[TRUE/FALSE]
Max flow in a flow network with real valued capacities can be found in polynomial time.

[TRUE/FALSE]
The top down pass in dynamic programming produces the **value** of the optimal solution whereas the bottom up pass produces the actual solution.

[TRUE/FALSE]
If a dynamic programming solution is formulated correctly, then it can be solved in polynomial time.

[TRUE/FALSE]
A flow network can have many sources and sinks but only one super source and one super sink.

[TRUE/FALSE]
It is possible to have a max flow of zero in a flow network even if all edge capacities are greater than zero.

[TRUE/FALSE]
If the capacities of edges in a flow network G are changed and the total increase of all edge capacities is k, then the value of a maximum flow of G increases by at most k.

[TRUE/FALSE]
For a flow network G = (V, E) that can carry a flow of up to $k>0$ units, if the capacity of each edge in a flow network is multiplied by m, then the resulting flow network will have a max flow of value mk.

[TRUE/FALSE] with justification
Although the original Ford-Fulkerson algorithm may fail to terminate in the case where edge capacities are arbitrary real numbers, it can be slightly modified so that it is guaranteed to terminate in the case where edge capacities are rational numbers, regardless of how the augmenting paths are chosen. This is an immediate consequence of the statement in Question 3.

Justification:
For rational numbers we can always multiple edge capacities by their common denominator and convert the flow network into a integer max flow problem and solve it using algorithm given in 3.

2) 20 pts

Say you have a graph G and a collection of sources (s1, …sk) and sinks (t1, ….tk). You want to route a path from each source si to any one of the sink nodes, but you don't want the paths to share any edges, or share any sink nodes. Present a solution to this problem and describe how you determine whether a solution exits. Also provide complexity analysis.

Answer:

1) Add a super source S and a super sink T to the graph, add edge S->Si for i = 1 to k with capacity 1. Add edge ti->T for i = 1 to k with capacity 1.
2) Assign capacity 1 for all other edges.
3) Calculate max flow from S to T and see if it is equal to k
4) If yes, there exists a solution, otherwise no solution exitsts.

Proof:

Similar to problem 14 of homework 6 part (a).

Complexity: The maximum flow is no larger than k. Thus the running time should be O(k|E|).

3) 20 pts

a) Let G = (V,E) be a flow network with source s, sink t, and an integer capacity c(u,v) on each edge (u,v) in E. For a given number K, show that an augmenting path of capacity at least K can be found in O(E) time, if such a path exists.

1) Form a graph G' from G in which each edge has capacity K or greater. This takes time O(E).
If there is an augmenting path of capacity K in E, it is also the same path in E'. Find any path from s to t, since any path from s to t in E' is an augmenting path of capacity K. Finding this path takes at most |E'| edge traversals, where |E'| <= |E|. So the total time is O(E). See p. 660 of text.

(2) The capacity of an augmenting path is the minimum capacity of any edge on the path, so look for an augmentingpath whose edges all have capacity at least K. Do a BFS search to find the path, considering only edges with residual capacity of at least K. This takes O(V+E) = O(E) time, since |V| = O(E) is a flow network.

b) The following modification of Ford-Fulkerson-Method can be used to compute a maximum flow in G.

```
MAX-FLOW-BY-SCALING(G,s,t)
1 C← max (u,v) member-of E c(u,v)
2 initialize flow f to 0
3 K ← 2  ^ ( [lg C])
4 while K ≥ 1
5 do while there exists an augmenting path p of capacity at least K
6 do augment flow f along p
7 K ← K / 2
8 return f
```

Argue that MAX-FLOW-BY-SCALING returns a maximum flow.

MAX-FLOW-BY-SCALING uses the FORD-FULKERSON method. It repeatedly augments the flow along an augmenting path until there are no augmenting paths of capacity >= 1. Since all capacities are integers, and the capacity of an augmenting path is positivie, this means that there are no augmenting paths whatsoever in the residual graph Gf. Thus, by the max-flow min-cut theorem, case 2 no augmenting paths, MAX-FLOW-BY-SCALING returns a maximum flow.

4) 20 pts

Recall that the Bellman-Ford algorithm finds the shortest distance between every vertex in G to a given vertex t in O(nm) time. Provide an $O(n^3)$ algorithm to calculate **all** shortest path lengths between vertices of a graph, i.e. your solution should find the shortest distance between every two pair of vertices in the graph.

Answer:

First, let vertices of G be numbered 1 through *N*. Then let SP(*i*, *j*, *k*) denote the shortest possible path from *i* to *j* using vertices only from the set {1, 2, ..., *k*} as intermediate pointalong the way.

There are two candidates for each of these paths: either the true shortest path only uses vertices in the set {1, ..., *k*}; or there exists some path that goes from *i* to *k* + 1, then from *k* + 1 to *j* that is better. We know that the best path from *i* to *j* that only uses vertices 1 through *k* is defined by SP(*i*, *j*, *k*), and it is clear that if there were a better path from *i* to *k* + 1 to *j*, then the length of this path would be the concatenation of the shortest path from *i* to *k* + 1 (using vertices in {1, ..., *k*}) and the shortest path from *k* + 1 to *j* (also using vertices in {1, ..., *k*}).

Thus, we have SP(i, j, 0) = w(i, j) if vertex i and j are connected by an edge of weight w(i, j). Also, SP(i, j, k+1) = min{SP(i, j, k), SP(i, k+1, k) + SP(k+1, j, k)}

We can calculate SP for all i, j, k between 1 and n. This takes O(n^3).

5) 20 pts
Using dynamic programming, find the optimal order of matrix multiplication operations in the matrix multiplication chain $M_1M_2M_3M_4M_5$
Matrix dimensions are M_1 :5 × 10, M_2 : 10 × 3, M_3 :3 × 12, M_4 : 12 × 5, and M_5 :5 × 50

	1	2	3	4	5
1	0	150	330	405	1655
2		0	360	330	2430
3			0	180	930
4				0	3000
5					0

Answer: Let OPT[i, j] be the number of operations needed to compute Ai*...Aj. We have OPT[i, i]=0 and
OPT[i, j] = min{OPT[i, k]+OPT[k+1, j]+Ri*Ck*Cj} where Ri is the number of rows in Ai, Cj is the number of columns in Aj and k goes from i to j-1.
Following this, we can generate the matrix of OPT[i, j] as above.
The optimal solution should be ((A1A2)(A3A4))A5.