CS570
Analysis of Algorithms
Fall 2008
Exam II

Name: _____

Student ID: _____

____Monday Section ____Wednesday Section ____Friday Section

	Maximum	Received
Problem 1	20	
Problem 2	15	
Problem 3	15	
Problem 4	15	
Problem 5	20	
Problem 6	15	
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.

FALSE [TRUE/FALSE]
For every node in a network, the total flow into a node equals the total flow out of a node.

TRUE [TRUE/FALSE]
Ford Fulkerson works on both directed and undirected graphs.

Because at every augmentation step you just need to find a path form s to t. This can be done in both direct and undriected graphs.

NO [YES/NO]
Suppose you have designed an algorithm which solves a problem of size n by reducing it to a max flow problem that will be solved with *Ford Fulkerson*, however the edges can have capacities which are $O(2^n)$. Is this algorithm efficient?

YES [YES/NO]
Is it possible for a valid flow to have a flow cycle (that is, a directed cycle in the graph, such that every edge has positive flow)?

positive flow cycles don't cause any problems. The flow can still be valid.

FALSE[TRUE/FALSE]
Dynamic programming and divide and conquer are similar in that in each approach the sub-problems at each step are completely independent of one another.

FALSE [TRUE/FALSE]
Ford Fulkerson has pseudo-polynomial complexity, so any problem that can be reduced to Max Flow and solved using *Ford Fulkerson* will have pseudo-polynomial complexity.

For example the edge disjoint paths problem is solved using FF in polynomial time. This is because for some problems the capacity C becomes a function of n or m.

TRUE [TRUE/FALSE]
In a flow network, the value of flow from S to T can be higher than the number of edge disjoint paths from S to T.

FALSE [TRUE/FALSE]
Complexity of a dynamic programming algorithm is equal to the number of unique sub-problems in the solution space.

TRUE [TRUE/FALSE]
In *Ford-Fulkerson*'s algorithm, when finding an augmentation path one can use either BFS or DFS.

TRUE **[TRUE/FALSE]**
When finding the value of the optimal solution in a dynamic programming algorithm one must find values of optimal solutions for all of its sub-problems.

2) 15 pts
Given a sequence of n real numbers $A_1 \ldots A_n$, give an efficient algorithm to find a subsequence (not necessarily contiguous) of maximum length, such that the values in the subsequence form a strictly increasing sequence.

Let A[i] represent the i-th element in the sequence.

```
initialize OPT[i] = 1 for all i.
best = 1
for(i = 2..n) {
  for(j = 1..i-1) {
    if(A[j] < A[i]) {
      OPT[i] = max( OPT[i], OPT[j] + 1 )
    }
    best = max( best, OPT[i] )
  }
}
return best
```

The runtime of the above algorithm is O(n^2)

3) 15 pts

Suppose you are given a table with N x M cells, each having a certain quantity of apples. You start from the upper-left corner and end at the lower right corner. At each step you can go down or right one cell. Give an efficient algorithm to find the maximum number of apples you can collect.

Let the top-left corner be in row 1 column 1, and the bottom-right corner be in row n column m.

Let A[i,j] represent the number of apples at row i column j.

```
initialize OPT[i,j] = 0 for all i,j.
for(i = 1...n) {
  for(j = 1...m) {
    OPT[i,j] = A[i,j] + max( OPT[i-1,j], OPT[i,j-1] )
  }
}
return OPT[n,m]
```

The runtime of the above algorithm is O(nm).

4) 15 pts

Suppose that you are in charge of a large blood bank, and your job is to match donor blood with patients in need. There are *n* units of blood, and *m* patients each in need of one unit of blood. Let us assume that the only factor which matters is that the blood type be compatible according to the following rules:

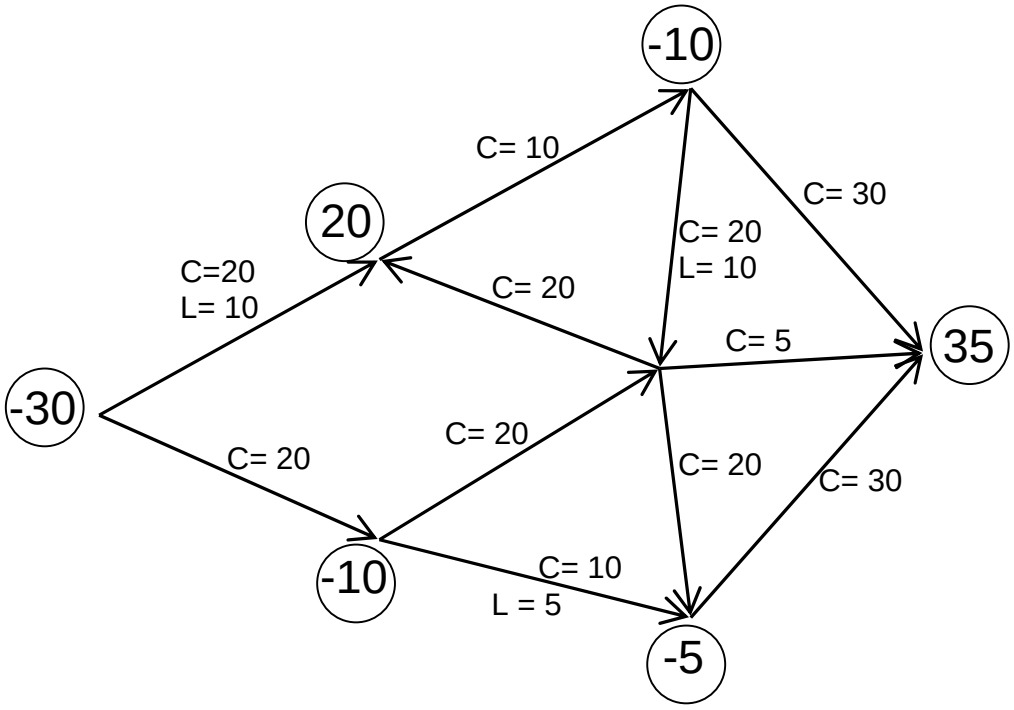
(a) Patient with type AB can receive types O, A, B, AB (universal recipient)
(b) Patient with type A can receive types O, A
(c) Patient with type B can receive types O, B
(d) Patient with type O can receive type O

Give a network flow algorithm to find the assignment such that the maximal number of patients receive blood, and prove its correctness.

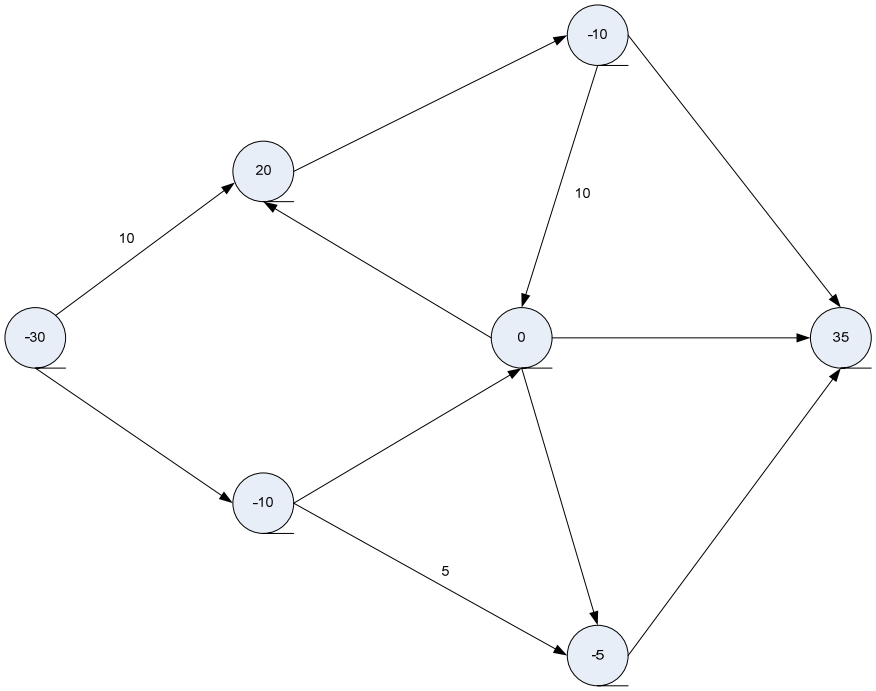
Given a set of n units of donor blood, and m patients in need of blood, each with some given blood type. We construct a network as follows. There is a source node, and a sink node, n donor nodes di, and m patient nodes pj. We connect the source to every donor node di with a capacity of 1. We connect each donor node di to every patient node pj which has blood type compatible with the donor's blood type, with a capacity of 1. We connect each patient node pj to the sink with capacity 1.

We then find the maximum flow in the network using Ford-Fulkerson. Patient j receives blood from donor i if and only if there is a flow of 1 from di to pj in the maximum flow. The total flow into each donor node di is bounded by 1, and the total flow out of each patient node pj is bounded by capacity 1, and so by conservation of flow, each patient and each donor can only give or receive 1 unit of blood. The types will be compatible by construction of the network.

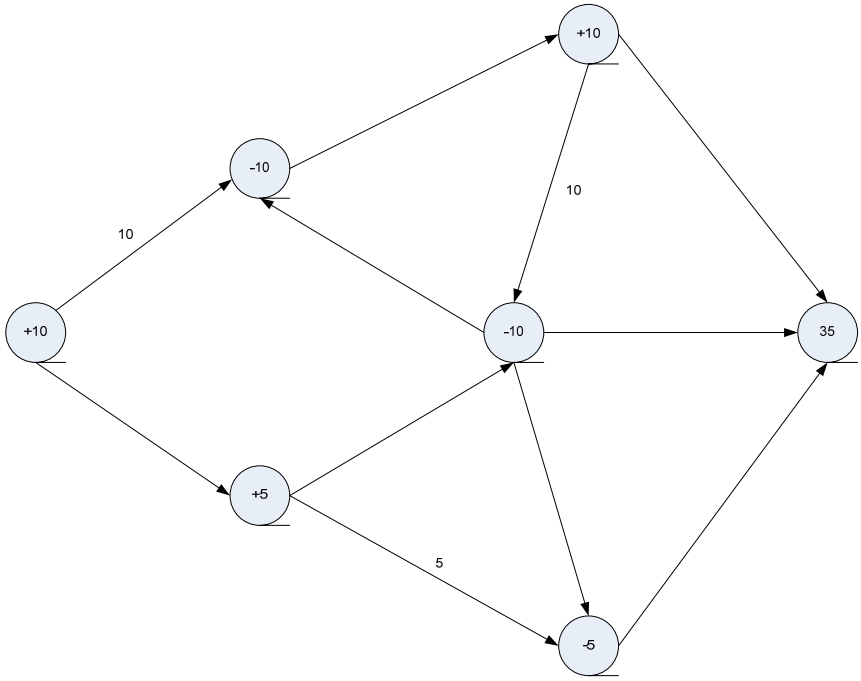
5) 20 pts
Given the graph below with demands/supplies as indicated below and edge capacities and possible lower bounds on flow marked up on each edge, find a feasible circulation. Show all your work



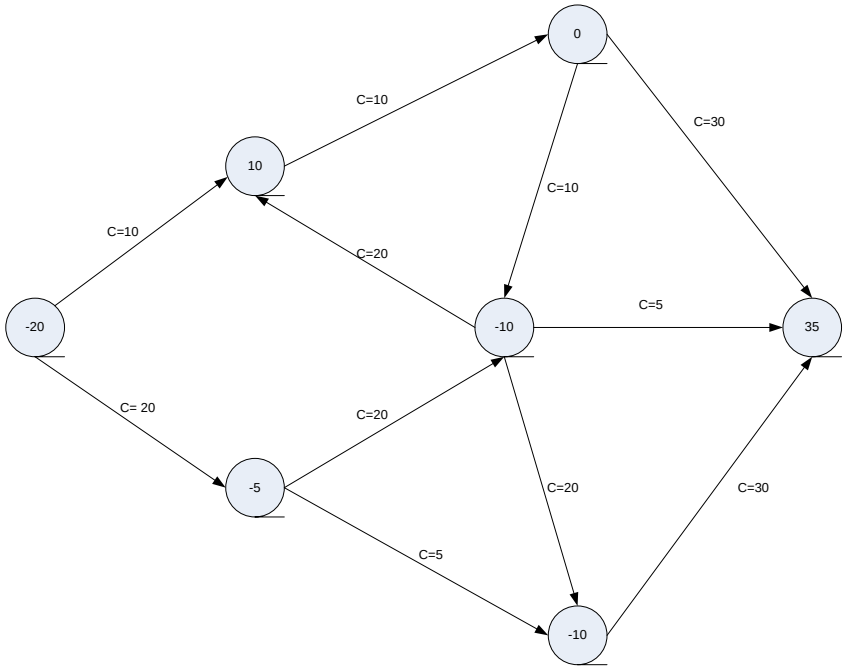
Solution to Q5
f0



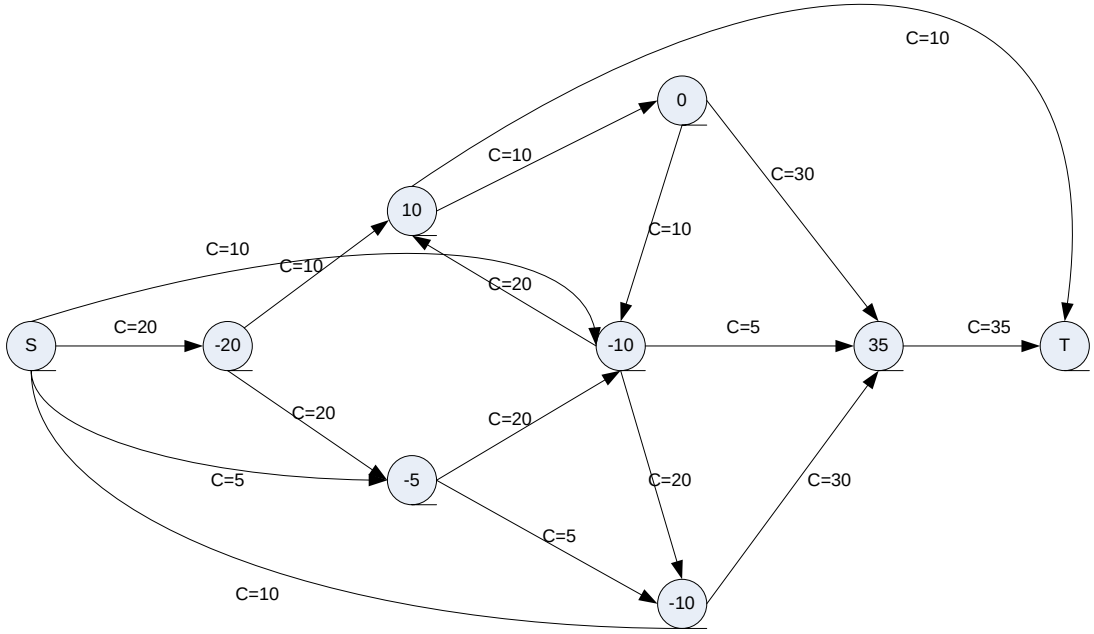
Lv



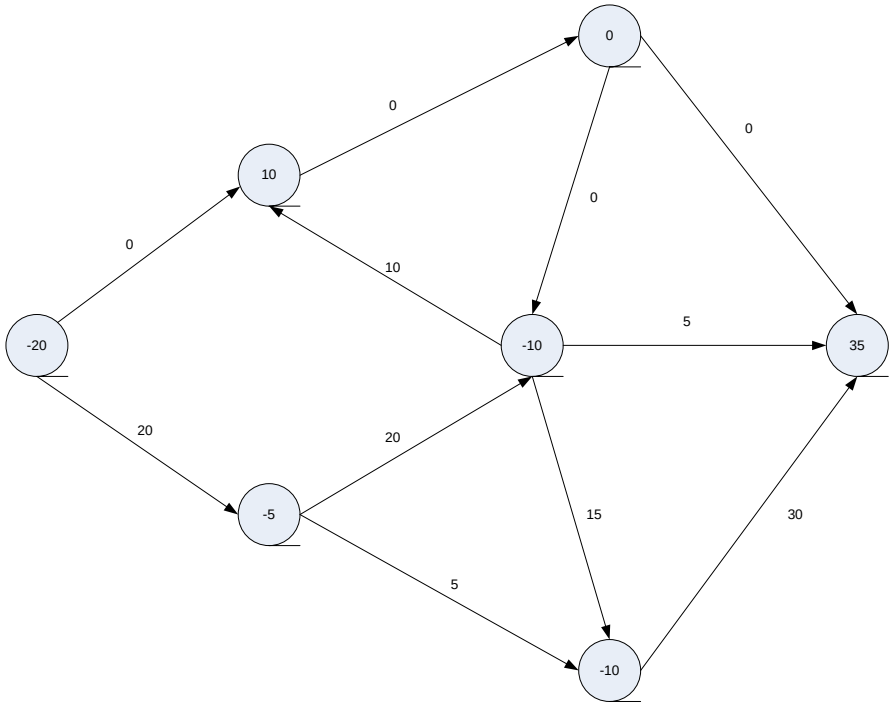
G’



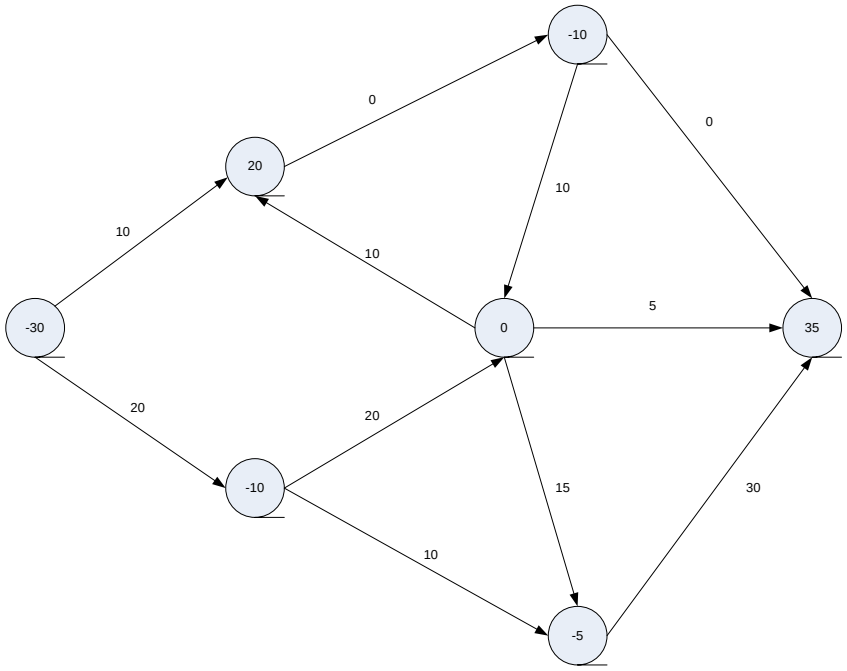
Convert G’ to a st network



f1



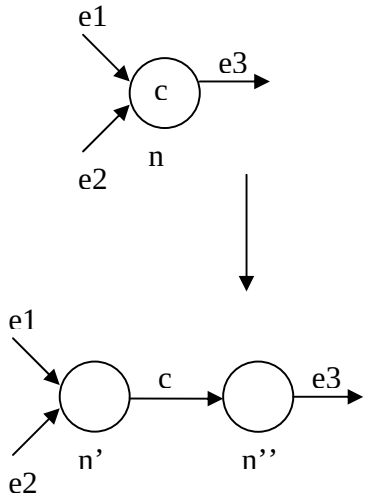
Circulation = f0+f1



6) 15 pts
Suppose that in addition to each arc having a capacity we also have a capacity on each node (thus if node i has capacity c_i then the maximum total flow which can enter or leave the node is c_i). Suppose you are given a flow network with capacities on both arcs and nodes. Describe how to find a maximum flow in such a network.

Solution:

Assume the initial flow network is G, for any node *n* with capacity *c*, decompose it into two nodes *n'* and *n''*, which is connected by the edge (*n'*, *n''*) with edge capacity *c*. Next, connect all edges into the node *n* in G to the node *n'*, and all edges out of the node n in G out of the node *n''*, as shown in the following figure.



After doing this for each node in G, we have a new flow network G'. Just run the standard network flow algorithms to find the maximal flow.