CS570
Analysis of Algorithms
Spring 2007
Exam 2

Name: _____
Student ID: _____

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

Note: The exam is closed book closed notes.

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

[TRUE/FALSE] True
Max flow problems can in general be solved using greedy techniques.

[TRUE/FALSE] False
If all edges have unique capacities, the network has a unique minimum cut.

[TRUE/FALSE] True
Flow f is maximum flow if and only if there are no augmenting paths.

[TRUE/FALSE] True
Suppose a maximum flow allocation is known. Increase the capacity of an edge by 1 unit. Then, updating a max flow can be easily done by finding an augmenting path in the residual flow graph.

[TRUE/FALSE] False
In order to apply divide & conquer algorithm, we must split the original problem into at least half the size.

[TRUE/FALSE] True
If all edge capacities in a graph are integer multiples of 5 then the maximum flow value is a multiple of 5.

[TRUE/FALSE] False
If all directed edges in a network have distinct capacities, then there is a unique maximum flow.

[TRUE/FALSE] True
Given a bipartite graph and a matching pairs, we can determine if the matching is maximum or not in O(V+E) time

[TRUE/FALSE] False
Maximum flow problem can be efficiently solved by dynamic programming

[TRUE/FALSE] True
The difference between dynamic programming and divide and conquer techniques is that in divide and conquer sub-problems are independent

2) 10pts
Give tight bound for the following recursion

a) $T(n)=T(n/2)+T(n/4)+n$

A simple method is that it is easy to easy that $T(n)=4n$ satisfy the recursive relation.
Hence $T(n)=O(n)$

b) $T(n) = 2T(n^{0.5})+\log n$

Let $n=2^k$ and $k=2^m$, we have $T(2^k)= 2T(2^{k/2})+k$ and $T(2^{k/2})= 2T(2^{k/4})+k/2$
Hence $T(2^k)=2(2T(2^{k/4})+k/2)+k=2^2\ T(2^{k/4})+2k=2^3\ T(2^{k/8})+3k=\ldots.=2^mT(1)+mk$
$=2^mT(1)+m2^m$. Note that $n=2^k$ and $k=2^m$, hence $m=\log\log n$. We have $T(n)=T(1)\log n+(\log\log n)*(\log n)$.
Hence $T(n)= (\log\log n)*(\log n)$

3) 20 pts

Let A = (a_0, a_1, . . . , a_{n-1}) be an array of n positive integers.

a- Present a divide-and-conquer algorithm which computes the sum of all the even integers a_i in A(1..n-1) where $a_i > a_{i-1}$. Solutions other than divide and conquer will receive no credit.

Algorithm: Compute-Even-Integers(l,r)
if r=l+1
    if((a_r%2=0) and ($a_r > a_l$))
        return a_r
    else
        return 0
else
    let m=$\left\lfloor \frac{l+r}{2} \right\rfloor$
    Let S= Compute-Even-Integers(l,m)+ Compute-Even-Integers(m,r)
    If a_m%2=0 and $a_m > a_{m-1}$
        S=S+ a_m

To computes the sum of all the even integers a_i in A(1..n-1) where $a_i > a_{i-1}$, invoke Compute-Even-Integers(0,n-1)

b- Show a recurrence which represents the number of additions required by your algorithm.

T(n)=2T(n/2)+c

c- Give a tight asymptotic bound for the number of additions required by your algorithm.
By master theorem, it is easy to see that $T(n)=O(n)$

d- Discuss how your approach would compare in practice to an iterative solution of the problem.
In practice, both methods are O(n) algorithm. Their complexity is the same.

4) 20 pts
Families 1…..N go out for dinner together. To increase their social interaction, no two members of the same family use the same table. Family j has a(j) members. There are M tables. Table j can seat b(j) people. Find a valid seating assignment if one exists.

We first construct a bipartite graph G whose node are the families f_i(i=1,…,N) and the tables t_j(j=1,…,M). We add edge e_{ij}=(f_i, t_j) in the graph for i=1,…,N and j=1,…,M and set e_{ij}=1. Then we add a course s and sink in G. For each family f_i, we add e=(s, f_i) in the graph and set c_e=a(i). For each stable t_j, we add e=(t_j, t) and set c_e=b(j) . After building the graph G for the original problem, we find the maximum s-t-flow value v in graph G by Fulkerson algorithm. If v is a(1)+…+a(N), then we make the seating assignment such that no two members of the same family use the same table, otherwise we can not.

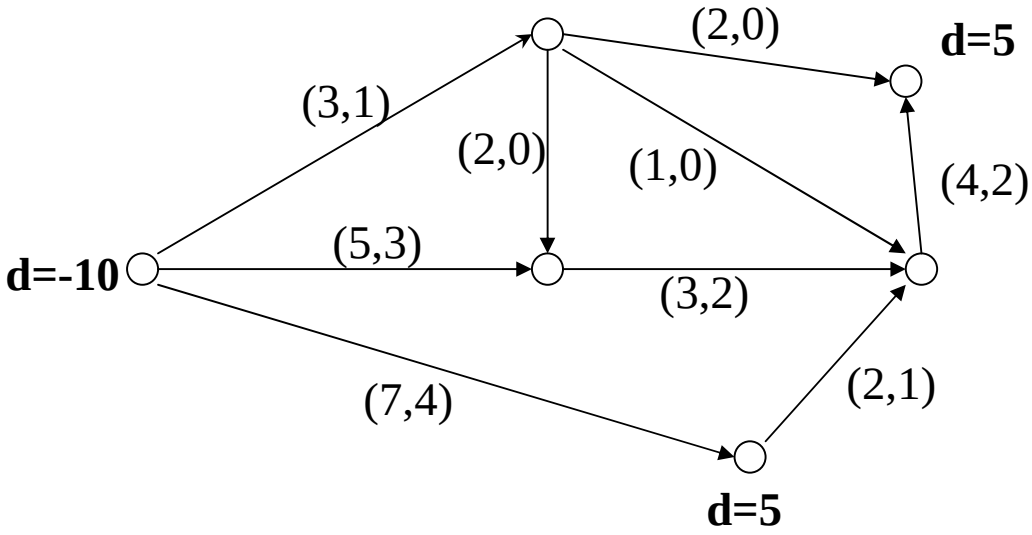
To prove the correctness of the algorithm, we prove that no two members of the same family use the same table if and only if the value of the maximum value of an s-t flow in G is a(1)+…+a(N):

First if no two members of the same family use the same table, it is easy to see that this flow meets all capacity constraints and has value a(1)+…+a(N).

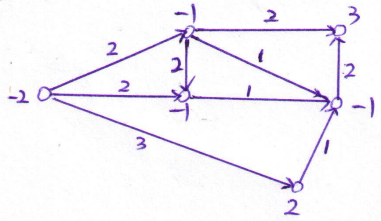
For the converse direction, assume that there is and s-t flow of value a(1)+…+a(N). The construction given in the algorithm makes sure that the there is at most one member in family j sitting in the table j as the capacity of the flow goes from f_i to c_j is 1. So we only need to make sure that all families are seated. Note that {s}, V-{s} is an s-t cut with value a(1)+…+a(N), so the flow must saturate all edges crossing the cut. Therefore, all families must be sitting in some tables. This completes the correctness proof.

Running time: The time complexity of construcing the graph is O(MN). The number of edges in the graph is MN+M+N, and v(f)= a(1)+…+a(N). Let T= a(1)+…+a(N), then the running time applying Ford-Fulkerson algorithm is O(MNT).

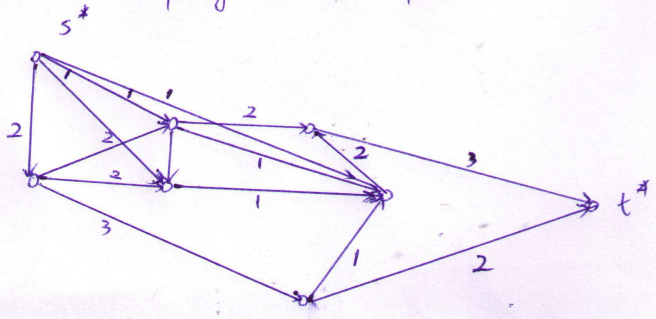
5) 20 pts
Determine if there is a feasible circulation in the below network. If so, determine the circulation, if not show where the bottlenecks are. The numbers in parentheses are (*lowerbound, upperbound*) on flow.



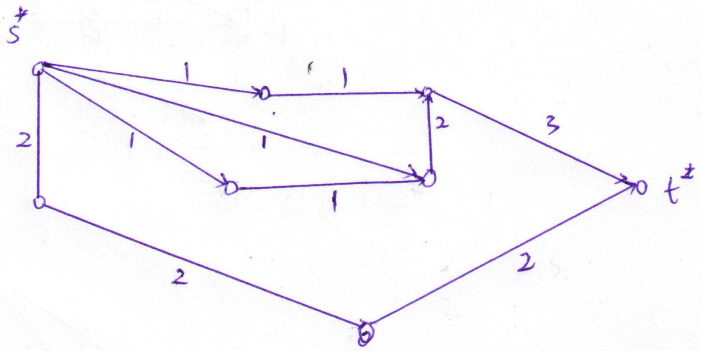
Step 1:



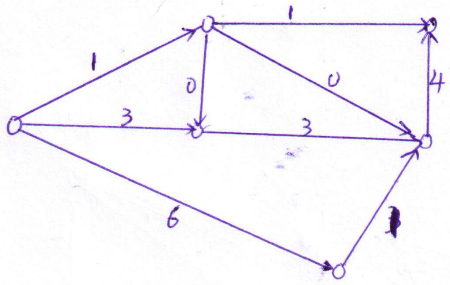
step 2: the ~~the~~ corresponding max-flow problem



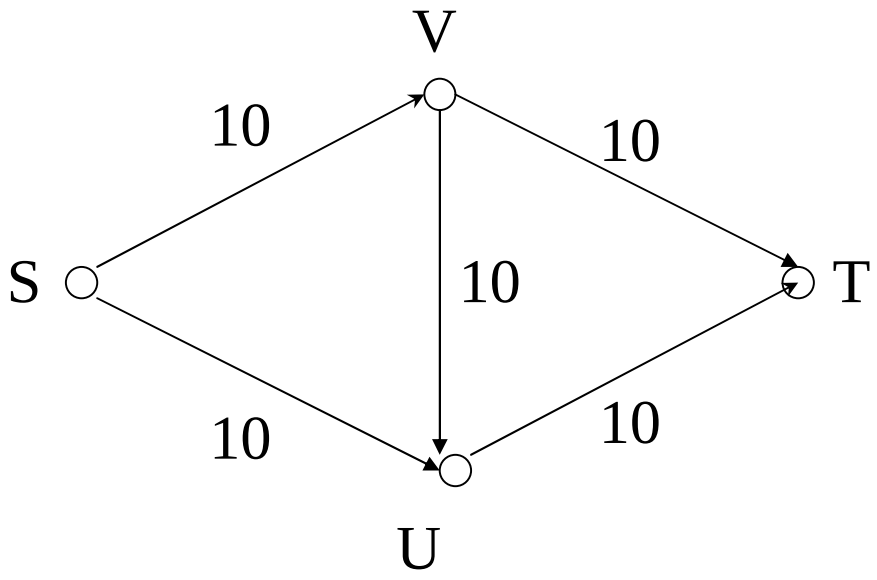
Step 3. Solving the max-flow problem and the result is as follows:



Step 4. The circulation is



6) 5 pts
List all minimum s-t cuts in the flow network pictured below. Capacity of every edge is equal to 10.



C1={{S},{V,U,T}}
C2={{S,U},{V,T}}
C3={{S,U,V},{T}}

7) 5 pts
Show an example of a graph in which poor choices of augmenting paths can lead to exactly C iterations of the Ford-Fulkerson algorithm before achieving max flow. (C is the sum of all edge capacities going out of the source and must be much greater than the number of edges in the network) Specifically, draw the network, mark up edge capacities, and list the sequence of augmenting paths.

Any example satisfying that the condition in this question is fine when you list the sequence of augmenting paths.