CS570

Analysis of Algorithms
Summer 2014
Exam II

Name: ______________________

Student ID: __________________

____On Campus ____DEN

	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

If a description to an algorithm is required please limit your description to within 150 words, anything beyond 150 words will not be considered.

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

[TRUE/FALSE] TRUE
A graph with unique edge weights will only have one minimum spanning tree

[TRUE/FALSE] FALSE
A graph with non-unique edge weights will have at least two minimum spanning trees

[TRUE/FALSE] FALSE
A flow network with unique edge capacities has a unique min cut

[TRUE/FALSE] FALSE
If a flow network with source s and sink t has a unique max flow from s to t, then it has a unique min-s-t cut.

[TRUE/FALSE] TRUE
Bellman-Ford algorithm can be used to find negative cost cycles in a graph.

[TRUE/FALSE] FALSE
In a divide and conquer algorithm, the complexity of divide step is always less than the complexity of the combine step

[TRUE/FALSE] TRUE
In a flow network with source s and sink t carrying a max flow, the amount of flow out of every s-t cut is identical.

[TRUE/FALSE] FALSE
Unlike dynamic programming, sub-problems in divide and conquer must have equal size

[TRUE/FALSE] TRUE
Unlike dynamic programming, sub-problems in divide and conquer are independent

[TRUE/FALSE] TRUE
If the edge weights of a weighted graph are doubled, then the number of minimum spanning trees of the graph remains unchanged

2) 20 pts

There are a series of volunteering activities lined up one after the other for the next year, $V_1, V_2, \dots, V_n$. For any i^{th} volunteering activity, you are given the happiness H_i associated with it. For any i^{th} activity, you are also given N_i, which is the number of immediately following volunteering activities that you cannot participate in if you participate in that i^{th} activity. Give an efficient dynamic programming solution to maximize the happiness. Also state the runtime of your algorithm.

.

For $1 <= i <= n$, let S_i denote a solution the maximum possible happiness for the set of events $\{V_i,....,V_n\}$ and let opt(i) denote its happiness.
If S_i chooses to volunteer for V_i, then it has to exclude V_{i+1} through V_{i+N_i}. In this case, its happiness is H_i plus the happiness it achieves for the set $\{V_{i+N_i+1},...,V_n\}$.
Since S_i is optimal for the set $\{V_i,..,V_n\}$, S_i restricted to the subset $\{V_{i+N_i+1},...,V_n\}$ has to be optimal. Thus in this case, opt(i) = H_i +opt($i + N_i + 1$).
If S_i chooses not to volunteer for V_i, then its happiness is the happiness it achieves for the set $\{V_{i+1},...,V_n\}$. Since S_i is optimal for the set $\{V_i,..,V_n\}$, S_i restricted to the subset $\{V_{i+1}...,V_n\}$ has to be optimal. Thus in this case, opt(i) = opt($i + 1$).
We thus have a recurrence for all i, opt(i)= max(opt($i + 1$), opt($i + N_i + 1$)+H_i).
(We understand that opt(index > n) = 0).
The boundary condition is that opt(n)=H_n and we start solving recurrence starting with opt(n), opt(n-1) and so on until opt(1).

3) 20 pts

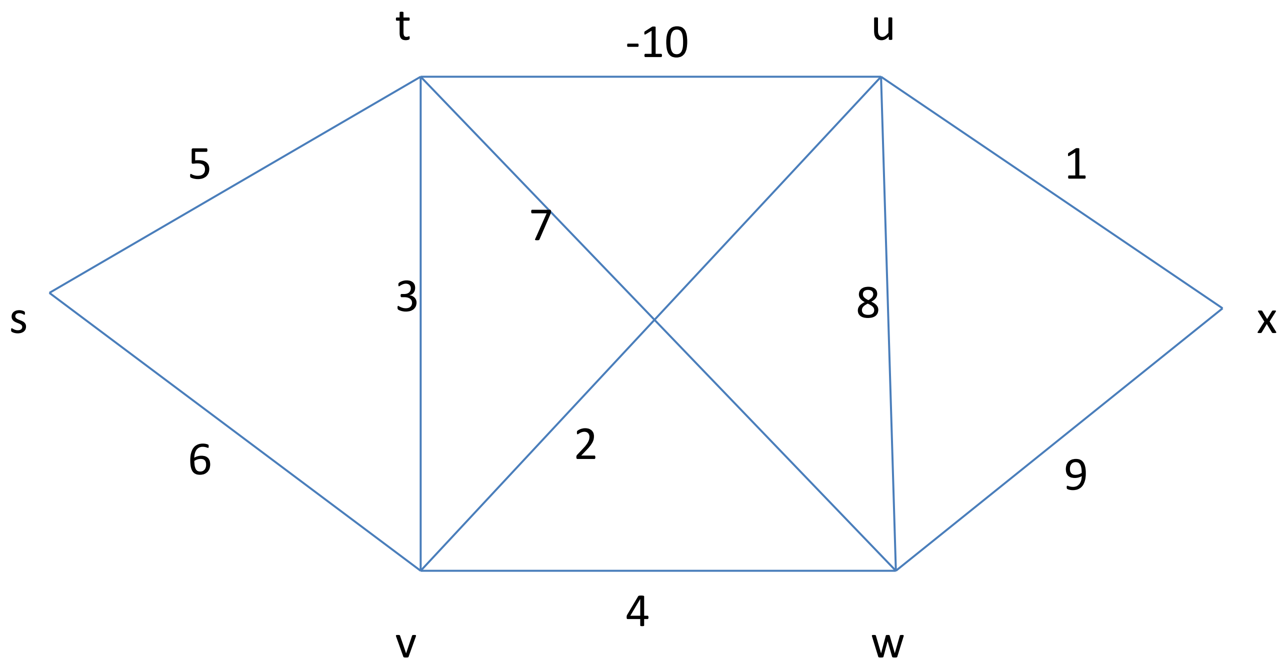
Two paths are said to be node-disjoint, if no vertex appears in both the paths. Given a graph G, with one vertex labelled as s and a different vertex labelled as t, give an efficient algorithm to find the maximum number of node-disjoint paths. State the run-time complexity of your algorithm and justify the correctness of your algorithm. Note: all paths will have s and t in common, but other than s and t they cannot have any other nodes in common.

Some of you answered the question considering G to be a directed graph and others considered G to be an undirected graph.

If you assume G to be directed, then the problem can be reduced to finding edge disjoint paths in directed graphs (which was solved in class) as follows. Create a new graph G' as follows. Replace every node v (that is neither s nor t) in G with two nodes v_{in} and v_{out}. Replace every edge (u, v) in G with an edge (u_{out}, v_{in}). Then there is a one to one correspondence between edge disjoint s-t paths in G' and node disjoint s-t paths in G (prove this !).

If you assume G to be undirected, then first make it directed by replacing each edge (u, v) with two directed edges (u, v) and (v, u) and then proceed as we did in the directed case to create G'. The proof in this case that there is a one to one correspondence between node disjoint s-t paths in G and directed edge disjoint paths in G' is a little more involved but follows from the section in the text about finding edge disjoint paths in undirected graphs.

4) 20 pts
Figure below shows an undirected graph and its edge costs.



a- Will Prim's algorithm correctly find the MST in this graph? Yes / No **Yes**

b- Will Kruskal's algorithm correctly find the MST in this graph? Yes / No **Yes**

c- Will the reverse delete algorithm correctly find the MST in this graph? Yes / No **Yes**

d- Can we use Bellman-Ford to find all shortest paths from every point in the graph to **S**? If yes, use Bellman ford to numerically calculate shortest paths from every node in the graph to **S.** If No, explain why. No, there is a negative weight cycle.

No, there is a negative cycle in the graph, t->u->v->t. So the shortest path from every node in the graph to S is not defined.

5) 20 pts

a- Which of the following techniques and algorithms can be classified as divide and conquer? (5 pts)

I. Binary search, **Yes**
II. Bellman-Ford running in a distributed environment as described in class, **No**
III. Kruskal's algorithm, **No**
IV. Strassen's method, **Yes**
V. Space efficient version of sequence alignment as described in class, **Yes**

b- A divide and conquer algorithm spends linear time to split a problem of size n into 4 equal subproblems of size n/2. It then solves the subproblems recursively. Finally, it spends n^2 log n time to combine all 4 sub-problems. Use the Master Theorem to find an upper bound to the cost of this algorithm. (5 pts)

T(n) = $4*T(\frac{n}{2})+\theta(n^2\log n)$

By case 2 in master theorem, i.e. If it is true, for some constant $k \geq 0$, that: f(n) = $\theta(n^c \log^k n)$ where c=$\log_b a$ then T(n) = $\theta(n^c \log^{k+1} n)$

T(n) = T($n^2\log^2$n)

c- Given a list P[1..n] that contains a stock's prices over the last n days, design a divide and conquer algorithm to determine if there are any three consecutive days during which the stock price increased (e.g. P[i] < P[i+1] < P[i+2]). What is the worst case running time of your solution?

```
Search(P, start, end)
{
    If(end-start == 0) return false;
    else if(end-start == 1)
    {
            If p[start] < p[start+1]<p[start+2] or p[start-1] < p[start]    <p[start+1]
then return true //Need to check boundary conditions, start >0 and end < n
            Else return false
    }
    Mid = (start+end)/2
    Return (Search(P,start,mid) or Search(P,mid+1,end))
}

Search(P,1,n)
```

Complexity: T(n) = 2T(n/2)+O(1), T(n) = O(n)