CS570
Analysis of Algorithms
Fall 2016
Exam III

Name: _____

Student ID: _____

Email Address:_____

_____Check if DEN Student

	Maximum	Received
Problem 1	20	
Problem 2	18	
Problem 3	18	
Problem 4	18	
Problem 5	18	
Problem 6	8	
Total	100	

Instructions:

1. This is a 2-hr exam. Closed book and notes
2. If a description to an algorithm or a proof is required please limit your description or proof to within 150 words, preferably not exceeding the space allotted for that question.
3. No space other than the pages in the exam booklet will be scanned for grading.
4. If you require an additional page for a question, you can use the extra page provided within this booklet. However please indicate clearly that you are continuing the solution on the additional page.

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

[TRUE] This is the definition of NP.
Every problem in NP has a polynomial time certifier.

[FALSE] Decision problems can be in P.
Every decision problem is in NP-complete.

[TRUE] If 3-SAT reduces to it, it's NP-Hard and we now it's NP so it's NPC.
An NP problem is NP-complete if 3-SAT reduces to it in polynomial time.

[FALSE] It will be O(m^2 + mn) and not linear.
If all edges in a graph have capacity 1, then Ford-Fulkerson runs in linear time.

[FALSE] consider a graph with edges (a,b) = 4, (b,c) = 3 and (a, c) = 5. The shortest path from a to c is 5 but (a, c) is not in the MST.
Let T be a minimum spanning tree of G. Then, for any pair of vertices s and t, the shortest s-t path in G is the path in T.

[TRUE] Master's Theorem
If the running time of a divide-and-conquer algorithm satisfies the recurrence $T(n) = 3\,T(n/2) + \Theta(n^2)$, then $T(n) = \Theta(n^2)$.

[FALSE] Consider the problem in the previous item.
In a divide and conquer solution, the sub-problems are disjoint and are of the same size.

[FALSE] Many of them are NP-Complete.
All Integer linear programming problems can be solved in polynomial time.

[TRUE]
If the linear program is feasible and bounded, then there exists an optimal solution.

[FALSE] Each specific operation can be larger than O(log n).
Suppose we have a data structure where the amortized running time of Insert and Delete is O(lg n). Then in any sequence of 2n calls to Insert and Delete, the worst-case running time for the nth call is O(lg n).

2) 18 pts

We are given an infinite array where the first n elements contain integers in sorted order and the rest of the elements are filled with ∞. We are not given the value of n. Describe an algorithm that takes an integer x as input and finds a position in the array containing x, if such a position exists, in O(lg n) time.

Solution:

- First determine the smallest power of 2 larger than n (8 points)

```
N=1
while A[i] ≠ ∞
    N = N*2
endwhile
```

N will be the smallest power of 2 larger than n

- Use binary search on A[1..N] to find x (8 points)

- Complexity: both components run in O(lg n) time

Common mistakes:

1. Without finding N, directly do binary search on A[1..n]
Note that n is not given. Cases like this get 4 points.

2. Without finding N, directly do binary search on A[1..2x+1]
x could be a negative integer; in this case, 2x+1 is also negative.
Cases like this get 8 points.

3. Time complexity is larger than O(log n) time. (e.g., linearly scan the array, or build a min heap first)
Cases like this get 0 points

3) 18 pts

For bit strings A = $a_1,..., a_m$, B = $b_1,..., b_n$ and C = $c_1,..., c_{m+n}$, we say that C is an interleaving of A and B if it can be obtained by interleaving the bits in A and B in a way that maintains the left-to-right order of the bits in A and B . For example if A = 101 and B = 01 then $a_1a_2b_1a_3b_2$ = 10011 is an interleaving of A and B, whereas 11010 is not. Give the most efficient algorithm you can to determine if C is an interleaving of A and B. Analyze the time complexity of your solution as a function of *m* and *n*.

Solution: The general form of the subproblem we solve will be: determine if $c_1,..., c_{i+j}$ is an interleaving of $a_1,..., a_i$ and $b_1,...,b_j$ for $0 \le i \le |A|$ and $0 \le j \le |B|$. Let c[i, j] be true if and only if $c_1,...,c_{i+j}$ is an interleaving of $a_1,..., a_i$ and $b_1,..., b_j$. We use the convention that if i = 0 then $a_i = \Phi$ (the empty string) and if j = 0 then $b_j = \Phi$. The subproblem c[i, j] can be recursively defined as shown (where c[|A|, |B|] gives the answer to the optimal problem):

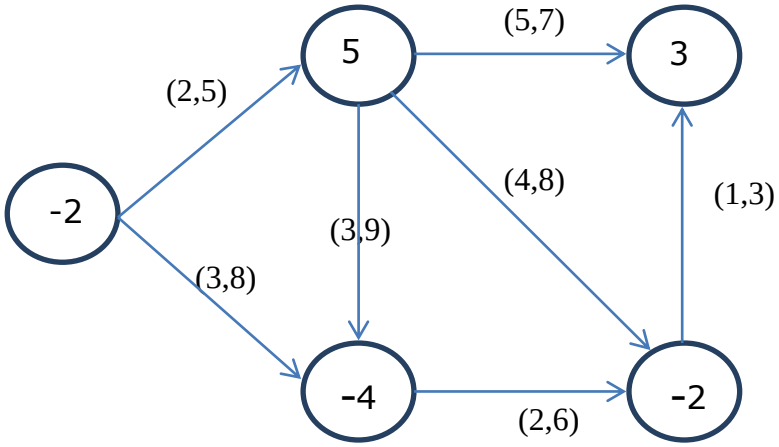
$$c[i,j] = \begin{cases} true, & if\ i = j = 0 \\ false, & if\ a_i \neq c_{i+j}\ \ and\ \ b_j \neq c_{i+j} \\ c[i-1,j], & if\ a_i = c_{i+j}\ \ and\ \ b_j \neq c_{i+j} \\ c[i,j-1], & if\ a_i \neq c_{i+j}\ \ and\ \ b_j = c_{i+j} \\ c[i-1,j] c[i,j-1], & if\ a_i = b_j = c_{i+j} \end{cases}$$

The time complexity is clearly O(|A||B|) since there are |A|x|B| subproblems each of which is solved in constant time. Finally, the c[i, j] matrix can be computed in row major order.

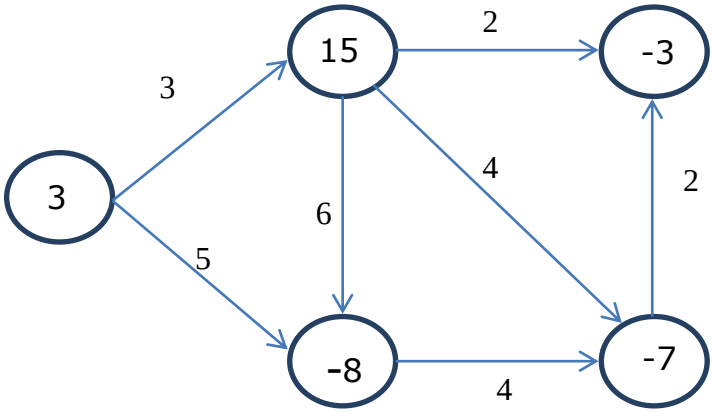
Proof of recurrence not asked for in the problem statement: *We now argue this recursive definition is correct. First the case where i = j = 0 is when both A and B are empty and then by definition C (which is also empty) is a valid interleaving of A and B. If $a_i \neq c_{i+j}$ and $b_j = c_{i+j}$ then there could only be a valid interleaving in which a_i appears last in the interleaving, and hence c[i, j] is true exactly when $c_1,..., c_{i+j-1}$ is a valid interleaving of $a_1,..., a_{i-1}$ and $b_1,..., b_j$ which is given by c[i-1, j]. Similarly, when $a_i \neq c_{i+j}$ and $b_j = c_{i+j}$ then c[i, j] = c[i-1, j].*
Finally, consider when $a_i = b_j = c_{i+j}$. In this case the interleaving (if it exists) must either end with b_i (in which case c[i-1, j] is true) or must end with b_i (in which case c[i, j-1] is true). Thus returning c[i-1, j] ∨ c[i, j-1] gives the correct answer.
Finally, since in all cases the value of c[i, j] comes directly from the answer to one of the subproblems, we have the optimal substructure property.

- *Recurrence and definition of OPT (12 pts)*
- *Algorithm (5 pts)*
- *Complexity (3 points)*

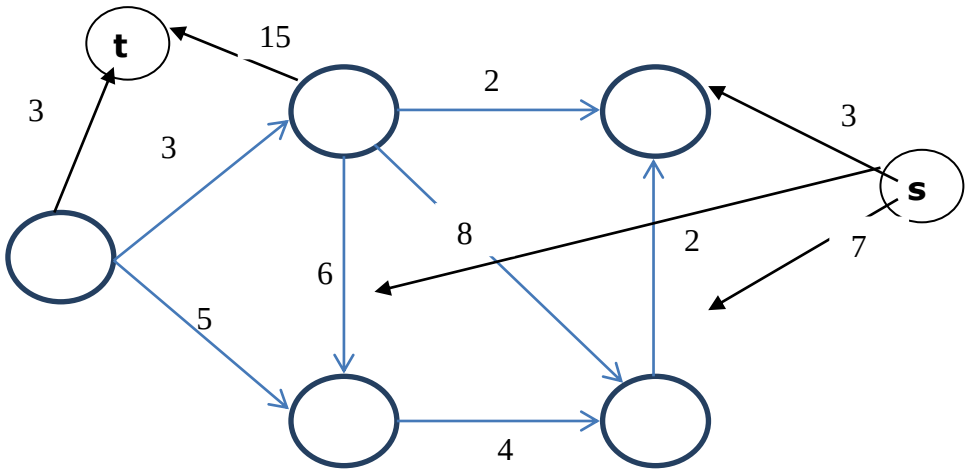
4) 18 pts
In the network below, the demand values are shown on vertices (supply value if negative). Lower bounds on flow and edge capacities are shown as (lower bound, capacity) for each edge. Determine if there is a feasible circulation in this graph. You need to show all your steps.



Solution.
Remove lower bounds by changing demands on each vertex



Reduce it to max-flow by adding s and t vertices.



Check if there is a s-t flow |f|= 18.
There is no such flow. It follows, no circulation.

6 points - for eliminating lower bounds and converting to problem with only demands
6 points - for eliminating demands and converting to max-flow problem
6 points - Checking if there exists a max-flow of value D

5) 18 pts

The Metric Traveling Salesman Problem (metric-TSP) is a restriction of the Traveling Salesman Problem, defined as follows. Given n cities $c_1, \ldots, c_n$ with inter-city distances $d(c_i, c_j)$ that form a metric:

$d(c_i,c_i) = 0$

$d(c_i, c_j) > 0$ for $i \neq j$

$d(c_i, c_j) + d(c_j, c_k) \geq d(c_i, c_k)$ for i, j, k in $\{1, \ldots, n\}$

Is there a closed tour that visits each city exactly once and covers a total distance at most C? Prove that metric-TSP is in NP-complete.

Solution.

1. It's easy to see that is in NP
2. Reduce form HC.

Given a graph G=(V,E) on which we want to solve the HAM-CYCLE problem. We will construct a complete G' with metric d as follows

d(u,v) = 0, if u = v
d(u,v) = 1, if (u,v) is in E
d(u,v) = 2, ow

Since triangle inequalities hold in G' (edges of any triangle will have sizes of 1 or 2), then $d(c_i, c_j) + d(c_j, c_k) \geq d(c_i, c_k)$ for i, j, k in $\{1, \ldots, n\}$

Let the bound C = |V|.
Claim: metric-TSP on G' is solvable if and only if the HC problem on G is solvable.
->) If there is a metric-TST of length |V|, then every distance between successive cities must be 1. Therefore these define as HC on G.
<-) If G has a hamiltonian cycle, then it defines a metric-TSP of size |V|.

6) 8 pts
Convert the following linear program

maximize ($3 x_1 + 8 x_2$)

Subject to
$x_1 + 4 x_2 \leq 20$
$x_1 + x_2 \geq 7$
$x_1 \geq -1$
$x_2 \leq 5$

to the standard form. You need to show all your steps.

x1 + 1 = z1 >= 0
x2 – 5 <= 0 so 5 – x2 = z2 >= 0

x1 + 4.x2 <= 0 → (z1 – 1) + 4(5 – z2) <= 20 → z1 – 4.z2 <= 1
x1 + x2 >= 7 → (z1 – 1) + (5 – z2) >= 7 → z2 – z1 <= -3

finally we get:

maximize 3.z1 – 8.z2 + 37
subject to:
z1 – 4.z2 <= 1
z2 – z2 <= -3
z1 >=0
z2 >= 0

Additional Space

Additional Space