CS570
Analysis of Algorithms
Fall 2016
Exam II

Name: ______________________

Student ID: __________________

Email Address:_________________

_____Check if DEN Student

	Maximum	Received
Problem 1	20	
Problem 2	16	
Problem 3	16	
Problem 4	16	
Problem 5	16	
Problem 6	16	
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.

[FALSE] The run-time is O(nW) so it's pseudopolynomial
0-1 knapsack problem can be solved using dynamic programming in polynomial time

[FALSE] If neither of the two nodes are reachable by the negative cycle, the distance between them is calculated accurately.
The Bellman-Ford algorithm always fails to find the shortest path between two nodes in a graph if there is a negative cycle present in the graph.

[TRUE] Value of max flow = capacity of min-cut (iterate over all edges of the min-cut)
Given the min-cut, we can find the value of max flow in O(|E|).

[TRUE] Set the penalties to 0 and reward of 1 for every match
The sequence alignment algorithm can be used to find the longest common subsequence between two given sequences.

[FALSE] You only have to do it once.
In dynamic programming you must calculate the optimal value of a sub-problem twice, once during the bottom up pass and once during the top down pass.

[TRUE] For any cut which is not a min-cut this is True.
Maximum value of an s-t flow could be less than the capacity of a given s-t cut in a flow network.

[FALSE] Consider a graph with 4 nodes s, a, b and t and E = { (s,a), (a, t), (s, b), (b, t)}. The increase in the flow would be f + 2.
Suppose the maximum (s, t)-flow of some graph has value f. Now we increase the capacity of every edge by 1. Then the maximum (s, t)-flow in this modified graph will have value at most f +1.

[FALSE] The Orlin algorithm runs in O(mn)
There are no known polynomial-time algorithms to solve maximum flow.

[FALSE] Edges having capacity 1 does not mean |f| = 1 so the Ford-Fulkerson is still going to be pseudo-polynomial.
If all edges in a graph have capacity 1, then Ford-Fulkerson runs in linear time.

[FALSE] Choice of augmenting path can still affect the number of iterations of the inner loop.
In the scaled version of the Ford Fulkerson algorithm, choice of augmenting paths cannot affect the number of iterations.

2) 16 pts

Given $N(G(V,E),s,t,c)$, a flow network with source s, sink t, and positive edge capacities c, we are asked to reduce the max flow by removing at most k edges. Prove or disprove the following statement:

Maximum reduction in flow can be achieved by finding a min cut in N, creating a sorted list of the edges going out of this cut in decreasing order and then removing the top k edges in this list.

The Statement is False. (8 points)

Proof (8 points). This requires a counter example where deleting the edges from the min-cut would not result in maximum reduction in max flow.

Many students tried to prove this statement. The statement is true only if all the edges have equal capacities. If the proof was correct based on that assumption, you were given 8 points.

3) 16 pts

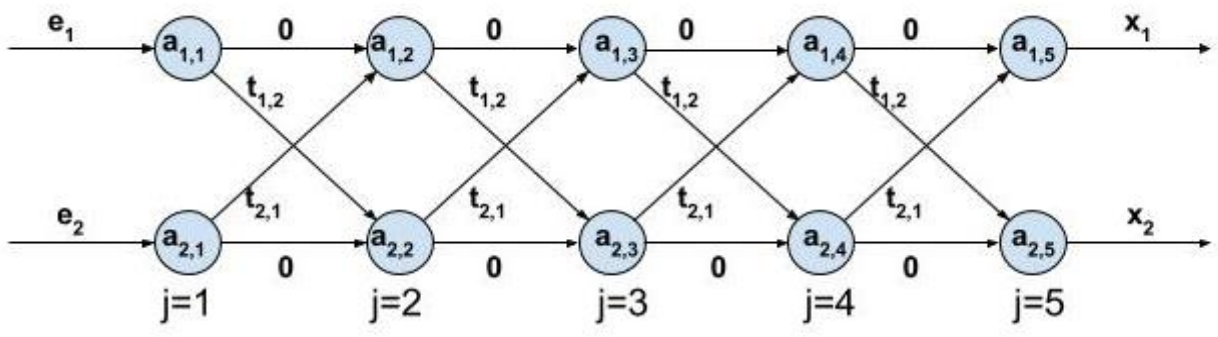
A car factory has k assembly lines, each with n stations. A station is denoted by $S_{i,j}$ where i indicates that the station is on the i-th assembly line (0<i<=k), and j indicates the station is the j-th station along the assembly line (0<j<=n). Each station is dedicated to some sort of work like engine fitting, body fitting, painting and so on. So, a car chassis must pass through each of the n stations in that order before exiting the factory. The time taken per station is denoted by $a_{i,j}$. Parallel stations of the k assembly lines perform the same task. After the car passes through station $S_{i,j}$, it will continue to station $S_{i,j+1}$ unless we decide to transfer it to another line. Continuing on the same line incurs no extra cost, but transferring from line x at station j-1 to line y at station j takes time $t_{x,y}$. Each assembly line takes an entry time e_i and exit time x_i which may be different for each of these k lines. Give an algorithm for computing the minimum time it will take to build a car chassis.

Below figure provides one example to explain the problem. The example shows k=2 assembly lines and 5 stations per line. To illustrate how to evaluate the cost, let's consider two cases: If we always use assembly line 1, the time cost will be:

$$e_1 + a_{1,1} + a_{1,2} + a_{1,3} + a_{1,4} + a_{1,5} + x_1.$$

If we use stations $s_{1,1} \to s_{2,2} \to s_{1,3} \to s_{2,4} \to s_{2,5}$, the time cost will be

$$e_1 + a_{1,1} + t_{1,2} + a_{2,2} + t_{2,1} + a_{1,3} + t_{1,2} + a_{2,4} + a_{2,5} + x_2.$$



a) Recursively define the value of an optimal solution (recurrence formula) (6 pts)

Dynamic programming Solution

Subproblem: minCost(i,j), if we use assembly line-i at station-j, the minimal cost so far.

Initialization: minCost(1,1) = e1+a11, minCost(2,1) = e2+a21, … minCost(k,1) = ek + ak1 (1 point)

Recurrence relation: $minCost(i,j) = \min_{(1\le l\le k)} minCost(l,j-1) + t_{l,i} + a_{i,j}$ (4 points)

$(t_{i,i} = 0$ in this formulation)

Final cost: minCost(i,exit) = minCost(i,n) + x_i   (1 point)

Attention: Some students may assume that the car can only pass to the station (i,j) from adjacent lines i-1 and i+1. In this case, you will have 3 points off.

b) Describe (using pseudocode) how the value of an optimal solution is obtained using iteration. Make sure you include any initialization required. (6 pts)

```
// initialization, boundary condition
For i=1:k
   minCost[i][1] = e[i]+a[i][1]            (1 point)

// recurrence
For t=2:n
   For i=1:k
      minCost[i][t] = minCost[1][t-1] + t[1][i]+a[i][t]
      for j=2:k
         minCost[i][t] = min(minCost[j][t-1] + t[j][i]+a[i][t], minCost[i][t])     (4 points)

for i=1:k
   minCost[i][n] = minCost[i][n]+x[i]

// final solution
return min( minCost[1][n], minCost[2][n], … minCost[k][n] )            (1 point)
```

Attention: some students may ignore the initialization.

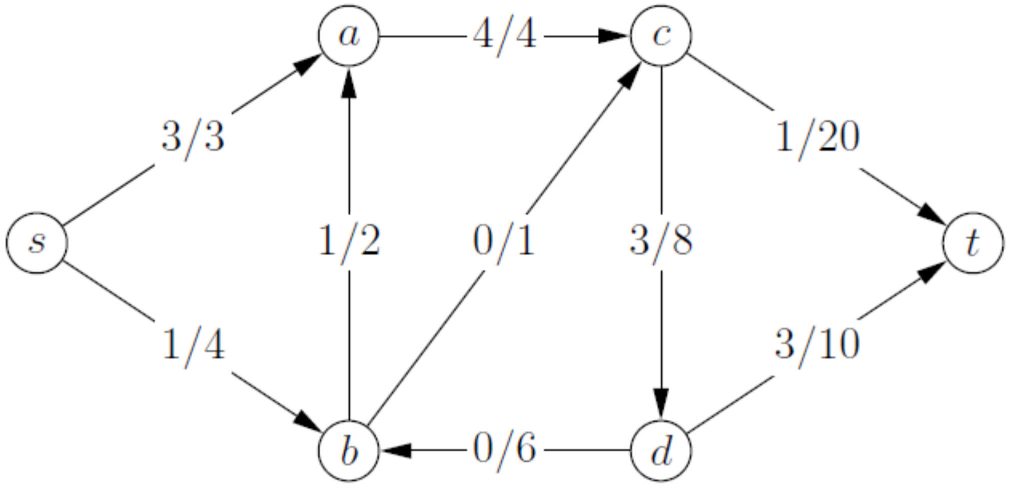
c) What is the complexity of your solution in part b? (4 pts)

From the recurrence loops, we can tell the complexity is $O(n\times(k^2))$

Attention: If you don't show your work process and your final solution is incorrect, you will have 4 points off.

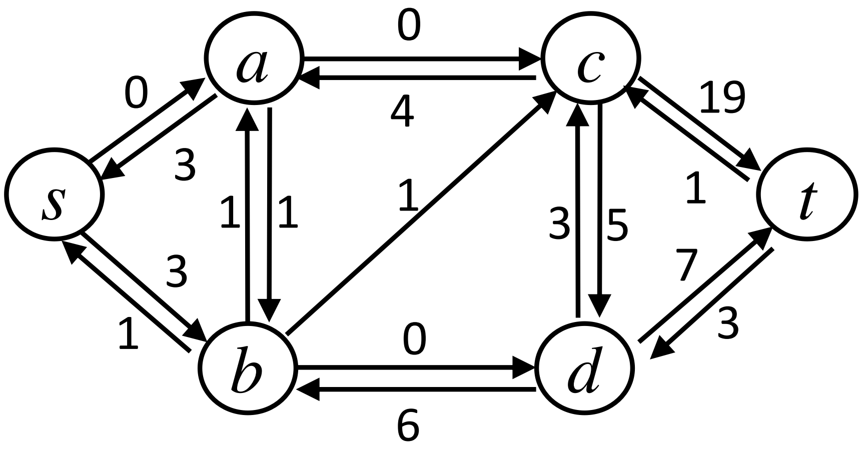
4) 16 pts

You are given a directed graph which after a few iterations of Ford-Fulkerson has the following flow. The labeling of edges indicate flow/capacity:



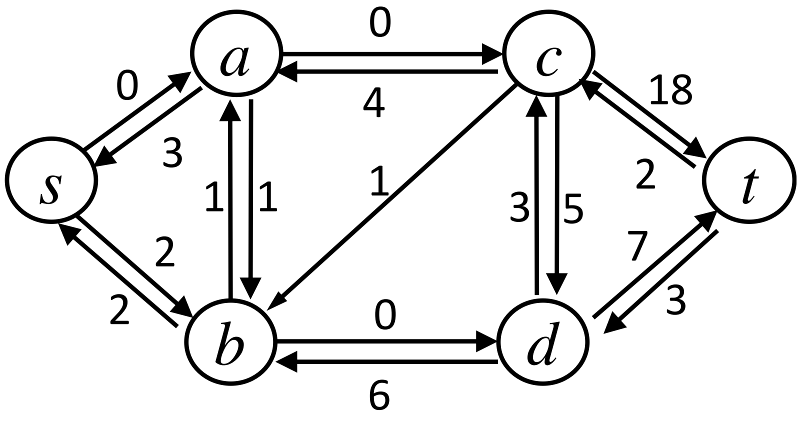
a) Draw the corresponding residual graph. (5 pts)

Each mistake in edge direction or value would cause one-point deduction.



b) Is this a max flow? If yes, indicate why. If no, find max flow. (6 pts)

No, since there is still path like s-b-c-t that can be augmented with flow 1 (3 points) The max flow is 4+1=5 that can be obtained from the residual graph shown below (3 points).



c) What is the min-cut? (5 pts)

Min-cut: ({s,a,b},{c,d,t}) (4 points). For the min-cut you need to show it on the graph or write it in the form (A,B) where A, B are the two parts of the original graph. The value of min-cut equals to $C_{ac}+C_{bc}=4+1=5$. (1 point)

Note that the other cuts like ({s,b},{a,c,d,t}) with value of 6 etc., have larger values than the min-cut with value of 5 and cannot be considered as min-cut for the network.

5) 16 pts

Given a rod of length n inches and an array of prices that contains prices of all pieces of size smaller than or equal to n. Determine the maximum value obtainable by cutting up the rod and selling the pieces. For example, if the length of the rod is 8 and the values of different pieces are given as in the following table, then the maximum obtainable value is 23 (by cutting the rod into two pieces of lengths 2 and 6). Notation: Price(i) is the price of a rod of length i

Length	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	18	17	20

And if the prices are as given in the following table, then the maximum obtainable value is 24 (by cutting the rod into eight pieces of length 1)

Length	1	2	3	4	5	6	7	8
Price	3	5	8	9	10	19	17	20

a) Recursively define the value of an optimal solution (recurrence formula) (6 pts)

The following four solutions are correct –

Soln 1:

OPT(i): best possible price obtained for a rod of length i

OPT(i) = max{p(j) + OPT(i-j)} for all j in {1..i}

Time complexity: O(n^2)

Here, the assumption is that the rod needs to be cut and cannot be left uncut.

Soln 2:

OPT(i): best possible price obtained for a rod of length i

OPT(i) = max{OPT(j) + OPT(i-j), p[j]} for all j in {1..i}

Time complexity: O(n^2)

Here, the assumption is that the rod need not be cut.

Soln 3:

OPT(i, j): best possible price obtained for a rod of length i with j cuts

OPT(i, j) = max{OPT(k, j-1) + p[i-k], OPT(i, j-1)} for all k in {1..i}

Time complexity: O(n^3)

4 points have been deducted for high time complexity.

Many students have used the above definition for OPT (i.e. j cuts as opposed to cuts 1..j) but have not used a third variable k. This is incorrect and marks have been deducted for this since we can have multiple cuts of same length.

Soln 4:

OPT(i, j): best possible price obtained for a rod of length i when the longest cut is no longer than j

$OPT(i, j) = \max\{OPT(i-j, j) + p[j], OPT(i, j-1)\}$

Time complexity: $O(n^2)$

Grading rubric:

(a) Definition of OPT: 2 marks
Recurrence: 3 marks
Final answer: 1 mark

(b) Initialization: 2 marks
Loop indices: 3 marks
Return final answer: 1 marks

(c) No partial marking
Note that no marks have been given for part c if the recurrence formed in part a was incorrect.

b) Describe (using pseudocode) how the value of an optimal solution is obtained using iteration. Make sure you include any initialization required. (6 pts)

Solution 1 can be written using iteration as

```
int cutRod(int price[], int n)
{
  int val[n+1];
  val[0] = 0;
  int i, j;

  // Build the table val[] in bottom up manner and return the last entry
  // from the table
  for (i = 1; i<=n; i++)
  {
     int max_val = INT_MIN;
     for (j = 1; j <=  i; j++)
       max_val = max(max_val, price[j] + val[i-j]);
```

```
        val[i] = max_val;
    }

    return val[n];
}
// end OR

return cutRod(arr, size)
```

c) What is the complexity of your solution in part b? (4 pts)

$O(n^2)$

6) 16 pts
There are n students in a class. We want to choose a subset of k students as a committee. There has to be m1 number of freshmen, m2 number of sophomores, m3 number of juniors, and m4 number of seniors in the committee. Each student is from one of k departments, where k = m1 +m2 +m3 +m4. Exactly one student from each department has to be chosen for the committee. We are given a list of students, their home departments, and their class (freshman, sophomore, junior, senior).
Describe an efficient algorithm based on network flow techniques to select who should be on the committee such that the above constraints are all satisfied.

a) Describe how to construct a flow network to solve this problem, including the description of nodes, edges, edge directions and their capacities. (10 pts)

b) Describe how the solution to the network flow problem you described in part a corresponds to the problem of determining who should be on the committee. (6 pts)

Solution: Consider the following graph, the source is connected to K departments with capacity one. Each department is connected to a subset of n students, that belongs to the department with capacity one. Each student is connected to the respective group (freshman, sophomore, junior, senior) with capacity one. Each group is connected to the sink with capacity m, for group I. The maximum flow in this problem finds the set of students to be included in the committee. The students' nodes that flow pass through them are shows who should be in the committee

a) The problem has two important constraints :

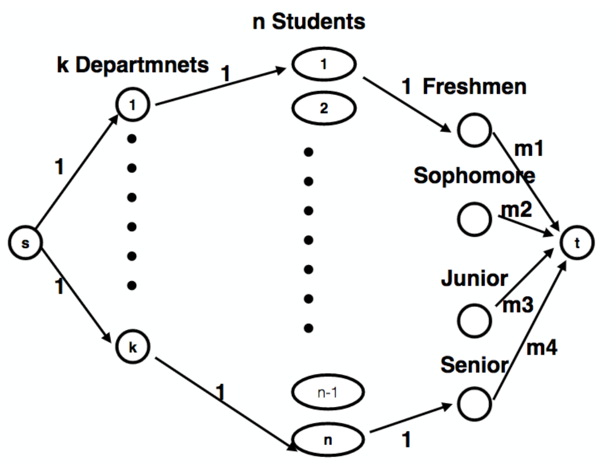
1. Exactly one student from each department must be selected.
   k nodes and k edges with correct capacity expected. (3 points)
2. Number of selected students from each class must be m1 for the freshmen, ..
   4 nodes for classes with capacity m1, m2, m3, m4 expected. (3 points)

For students you can either put some nodes or some edges, But if your graph doesn't give the correct answer you will not get point just for adding n nodes for students. (2 points)
The network flow graph gives the right set of students (2 points)

b) claim: Picking k students for the committee is possible iff The max flow of the given graph in part a is equal to k. (2 points)

Why your graph satisfies the constraint of the problem (2 points)
How the set of students can be determined based on you graph (2 points)

You can either explain these three question or prove the claim, and you will get 6 points.


n Students
k Departmnets
1
1
1
1
1
1
2
n-1
n
s
k
1 Freshmen
Sophomore
Junior
Senior
m1
m2
m3
m4
t
1