CS570

Analysis of Algorithms

Summer 2017

Exam II

Name: ______________________

Student ID: _________________

Email Address:_________________

_____Check if DEN Student

	Maximum	Received
Problem 1	20	
Problem 2	12	
Problem 3	10	
Problem 4	15	
Problem 5	13	
Problem 6	15	
Problem 7	15	
Total		

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]
Given the value of max flow, we can find a min-cut in linear time.

[TRUE]
Given a min-cut, we can find the value of max flow in linear time.

[FALSE]
The Ford-Fulkerson algorithm can compute the maximum flow in polynomial time.

[FALSE]
If all edges in a graph have capacity 1, then Ford-Fulkerson runs in linear time.

[TRUE]
If all of the edge capacities in a graph are an integer multiple of 3, then the value of the maximum flow will be a multiple of 3.

[TRUE]
The Floyd-Warhsall algorithm always detects if a graph has a negative cycle.

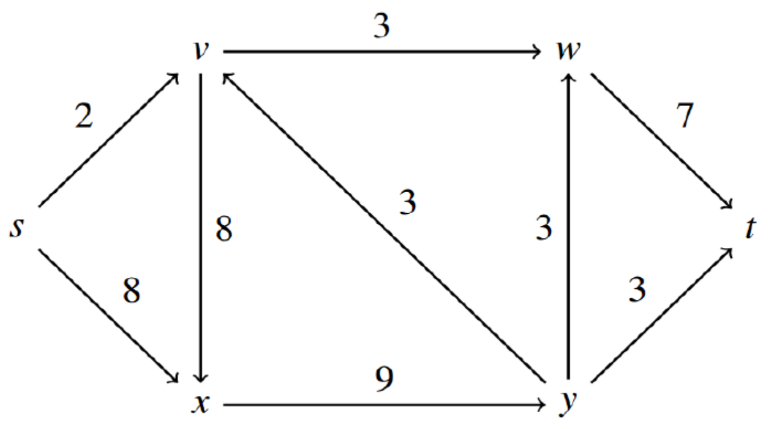
[TRUE]
Increasing the capacity of an edge that belongs to a min-cut in a flow network may not result in increasing the maximum flow.

[FALSE]
If a dynamic programming algorithm has *n* subproblems, then its running time complexity is *O(n)*.

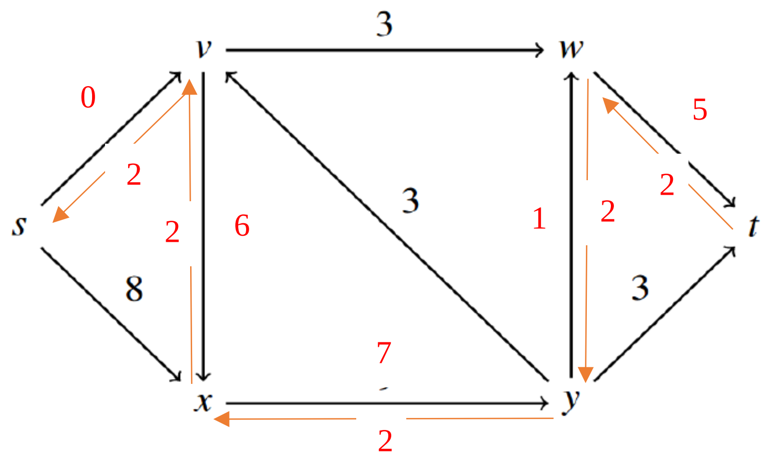
[TRUE]
It is possible for a dynamic programming algorithm to have exponential running time.

[FALSE]
We can use the Bellman-Ford algorithm for *undirected* graph with negative edge weights.

2) 12 pts.
You are given the following graph. Each edge is labeled with the capacity of that edge.



a) Draw the corresponding residual graph after sending as much flow as possible along the path $s \to v \to x \to y \to w \to t$. (4 pts)



b) Find the value of a max-flow. You do not need to demonstrate all steps. (4 pts)

Sol. 9

c) Find a min-cut partition? (4 pts)

Sol. A = {s, x, v, y}, B = {w, t}

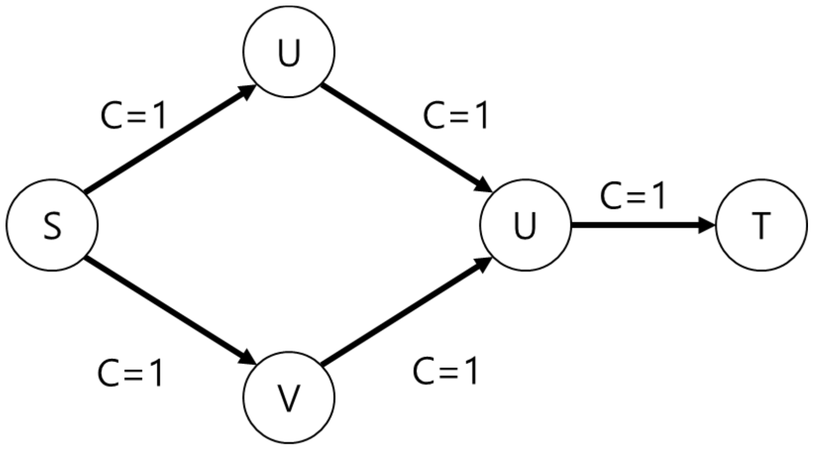
3) 10 pts.

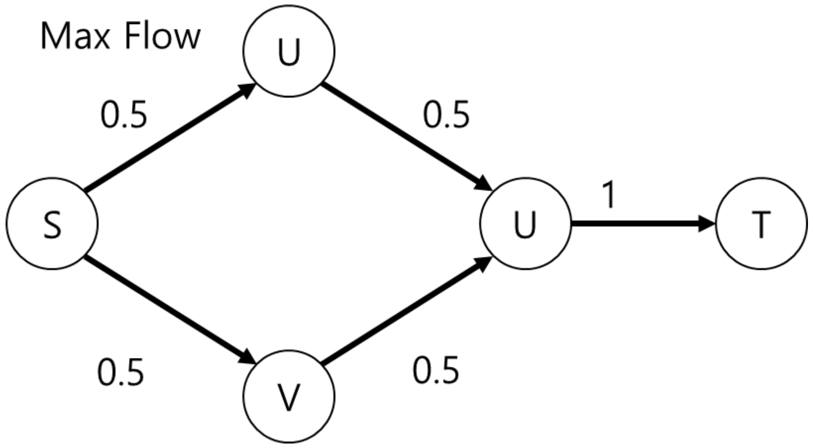
Given a flow network with the source *s* and the sink *t*, and positive integer edge capacities *c*. Prove or disprove the following statement:

A maximum flow in an integer capacity graph must have integral flow on each edge.

Hint: do not think of the Ford-Fulkerson algorithm.

Counterexample





Rubrics

A correct counter example = 10 points

Saying the statement is false, without right counter example (e.g. 0 is an integer so it will not count as a non-integral flow.) = 5 points

otherwise 0 points is given.

4) 15 pts.

There are three alternatives for solving a problem of size *n* using a divide-and-conquer technique:
1: You solve 3 subproblems of size *n*/2 with the cost for combining step Θ(n log n).
2: You solve 4 subproblems of size *n*/2 with the cost for combining step Θ(n^2).
3: You solve 5 subproblems of size *n*/2 with the cost for combining step Θ(n^2 log n).

Which alternative do you prefer and why?

1) Master theorem case 1

   T(n) = $\Theta(n^{log_2 3})$

2) Master theorem case 2,

   T(n) = Θ(n^2 log n)

3) Similar as 1), T(n) = $\Theta(n^{log_2 5})$

Therefore, 1) should be the case.

Rubrics
4 points for each complexity analysis
3 points for stating 1) is the case.
-2 points if Big O notation is used instead of Big Theta notation, or no notation.

5) 13 pts.

Consider an array *A* containing *n* distinct integers. We define a local minimum of *A* to be an *x* such that $x=A[i]$, for some $0<i<n$ with $A[i-1] > A[i]$ and $A[i]<A[i+1]$. For boundary elements, there is only one neighbor to consider. As an example, suppose $A = [10, 6, 4, 3, 12, 19, 18]$. Then *A* has two local minima: 3 and 18.

a) Describe a divide and conquer algorithm that find a local minimum. (9 pts.)

Let m=n/2 and examine if A[m] is local minimum:

1. A[m] is local minimum -> return it.
2. A[m] > A[m-1] -> the left half the array must contain a local minimum so do the same thing on the left half.
3. A[m] > A[m+1] -> the right half of the array must contain a local minimum so do the same thing on the right half.

Grading

Division

-3 pts if tries to find minimum values on both subproblems (We need only one minimum)

Combining step

-2pts for non-constant

b) Express a recurrence equation for the running time of your algorithm and solve the recurrence by the master theorem. (4 pts)

$T(n) = T(n/2) + \theta(1)$

by case 2 of the Masters Theorem, $T(n) = \theta(\log n)$

Grading rubric

Correct recursion: 2pt

Correct time complexity based on written recursion: 2pt

-1pt for using Big O notation or no notation for T(n)

Even if part a) searches for both side ($T(n) = 2T(n/2)$) or has wrong combining complexity, writing corresponding recursion and time complexity gets credit.

6) 15 pts.

You are given a piece of cloth of size $n \times n$, where n is positive integer. The goal is to cut the cloth into as many flags as possible. There are k possible flag types $F_1, F_2, \ldots, F_k$. Each flag type $F_j, j = 1, 2, \ldots, k$ is a rectangle with integral sides. There are two important constraints:

1) you can only make horizontal **or** vertical cut through the entire cloth.
2) the flags cannot be rotated.

Give a DP algorithm to find the maximum number of flags you can cut out of the given cloth. You may get many flags of the same type and none of the certain type.

a) Define (in plain English) subproblems to be solved. (5 pts)

Let $OPT[i, j]$ be the optimal number of flags we can cut from a piece of cloth of size $i \times j$.

b) Write the recurrence relation for subproblems. (7 pts)

a) $OPT[i, j] = 1$, *if $i \times j$ is a type, o.w. 0*

b) $OPT[i, j] = \underset{1 <= x < i}{max}(OPT[x, j] + OPT[i - x, j]$

c) $OPT[i, j] = \underset{1 <= y < j}{\max}(OPT[i, y] + OPT[i, j - y]$

OPT[i, j] is MAX of all three cases

d) Compute the runtime of the algorithm in terms of n and k. (3 pts)

$$O(n^3 + n^2k): O(n^2k)\ 1\ pts, O(n^3)\ 2\ pts$$

About wrong dynamic formula: Note that the maximum point given to the wrong or partially correct solurtion is 8. So please ask for regrading only if you think your answer is completely correct.

7) 15 pts.

A local charity wishes to organize a series of blind dates. There are n male and n female students, and each male student is to be paired with a female student each evening. Based on forms filled in by the participants, we know which pairs of students are compatible. Given this information, describe an algorithm to find the *maximum* number of rounds of blind dates that can be organized, subject to the following conditions: each round consists of exactly n dates, no student can participate in more than one date in the same round, no student encounters another student more than once, and no two-incompatible people are ever matched.

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)

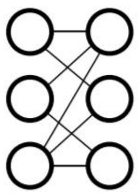
Sol.
(5 pts)
Consider a network $N = (V, E)$, where the source is connected to all the male students, all female students are connected to the sink, and we add directed edges from a male student to a female student if the pair is compatible.
On the edges between male and female students, we put capacity of one.

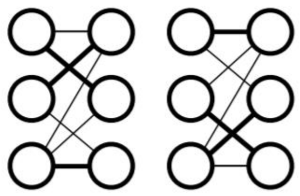
(5 pts)
On the edges connected to the sink, we put capacity of infinity.
On the edges connected to the source, we put capacity of k.

b) Describe how the solution to the network flow problem you described in part (a) corresponds to the problem of maximizing the revenue. (5 pts)

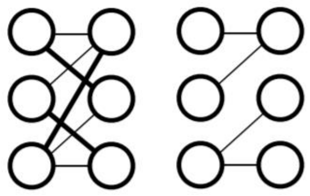
If we choose k and find the maximum flow, and the value of the maximum flow is kn, then we know that there are k perfect matchings in the problem. Therefore, we can start with k = 1 and increment k until the value of maximum flow is no longer kn.

Wrong Answer (10 pts)
1. we can think of a greedy algorithm that "peels off" perfect matchings one by one. Using any algorithm, either maximum bipartite matching algorithm using augmenting paths; or the above algorithm with k = 1, find a perfect matching on current graph. Then, delete all the edges in the matching, and repeat until there is no more perfect matching. The number of rounds this algorithm finds perfect matchings is the number of rounds we can have blind dates. This algorithm does not always give the right solution, and here is a counter example.





Two rounds of blind dates possible



Greedy algorithm might fail