

CS570
Analysis of Algorithms
Fall 2014
Exam II

Name: _____

Student ID: _____

Email: _____

Wednesday Evening Section

	Maximum	Received
Problem 1	20	
Problem 2	16	
Problem 3	16	
Problem 4	20	
Problem 5	16	
Problem 6	12	
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.

1) 20 pts

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

 [TRUE/FALSE]

If an iteration of the Ford-Fulkerson algorithm on a network places flow 1 through an edge (u, v) , then in every later iteration, the flow through (u, v) is at least 1.

 [TRUE/FALSE]

For the recursion $T(n) = 4T(n/3) + n$, the size of each subproblem at depth k of the recursion tree is $n/3^{k-1}$.

[TRUE/FALSE]

For any flow network G and any maximum flow on G , there is always an edge e such that increasing the capacity of e increases the maximum flow of the network.

[TRUE/FALSE]

The asymptotic bound for the recurrence $T(n) = 3T(n/9) + n$ is given by $\Theta(n^{1/2} \log n)$.

[TRUE/FALSE]

Any Dynamic Programming algorithm with n subproblems will run in $O(n)$ time.

[TRUE/FALSE]

A pseudo-polynomial time algorithm is always slower than a polynomial time algorithm.

[TRUE/FALSE]

The sequence alignment algorithm can be used to find the longest common subsequence between two given sequences.

[TRUE/FALSE]

If a dynamic programming solution is set up correctly, i.e. the recurrence equation is correct and each unique sub-problem is solved only once (memoization), then the resulting algorithm will always find the optimal solution in polynomial time.

[TRUE/FALSE]

For a divide and conquer algorithm, it is possible that the divide step takes longer to do than the combine step.

[TRUE/FALSE]

Maximum value of an $s - t$ flow could be less than the capacity of a given $s - t$ cut in a flow network.

2) 16 pts

Recall the Bellman-Ford algorithm described in class where we computed the shortest distance from all points in the graph to t . And recall that we were able to find all shortest distance to t with only $O(n)$ memory.

How would you extend the algorithm to compute both the shortest distance and to find the actual shortest paths from all points to t with only $O(n)$ memory?

3) 16 pts

During their studies, 7 friends (Alice, Bob, Carl, Dan, Emily, Frank, and Geoffrey) live together in a house. They agree that each of them has to cook dinner on exactly one day of the week. However, assigning the days turns out to be a bit tricky because each of the 7 students is unavailable on some of the days. Specifically, they are unavailable on the following days (1 = Monday, 2 = Tuesday, ..., 7 = Sunday):

- Alice: 2, 3, 4, 5
- Bob: 1, 2, 4, 7
- Carl: 3, 4, 6, 7
- Dan: 1, 2, 3, 5, 6
- Emily: 1, 3, 4, 5, 7
- Frank: 1, 2, 3, 5, 6
- Geoffrey: 1, 2, 5, 6

Transform the above problem into a maximum flow problem and draw the resulting flow network. If a solution exists, the flow network should indicate who will cook on each day; otherwise it must show that a feasible solution does not exist

4) 20 pts

Suppose that there are n asteroids that are headed for earth. Asteroid i will hit the earth in time t_i and cause damage d_i unless it is shattered before hitting the earth, by a laser beam of energy e_i . Engineers at NASA have designed a powerful laser weapon for this purpose. However, the laser weapon needs to charge for a duration ce before firing a beam of energy e . Can you design a dynamic programming based pseudo-polynomial time algorithm to decide on a firing schedule for the laser beam to minimize the damage to earth? Assume that the laser is initially uncharged and the quantities c, t_i, d_i, e_i are all positive integers. Analyze the running time of your algorithm. You should include a brief description/derivation of your recurrence relation. Description of recurrence relation = 8pts, Algorithm = 6pts, Run Time = 6pts

5) 16 pts

Consider a two-dimensional array $A[1:n, 1:n]$ of integers. In the array each row is sorted in ascending order and each column is also sorted in ascending order. Our goal is to determine if a given value x exists in the array.

- a. One way to do this is to call binary search on each row (alternately, on each column). What is the running time of this approach? [2 pts]
- b. Design another divide-and-conquer algorithm to solve this problem, and state the runtime of your algorithm. Your algorithm should take strictly less than $O(n^2)$ time to run, and should make use of the fact that each row and each column is in sorted order (i.e., don't just call binary search on each row or column). State the run-time complexity of your solution.

6) 12 pts

Consider a divide-and-conquer algorithm that splits the problem of size n into 4 sub-problems of size $n/2$. Assume that the divide step takes $O(n^2)$ to run and the combine step takes $O(n^2 \log n)$ to run on problem of size n . Use any method that you know of to come up with an upper bound (as tight as possible) on the cost of this algorithm.

Additional Space