

CSci 280, Spring 2009, Exam 1

This take-home exam is due **Friday, February 13**, at 4pm. You should submit your solution on paper, either handwritten or typed. E-mailed solutions will not be accepted.

Answer only four of the following five problems. If you answer all problems, I will grade only the first four. Each problem is worth 20 points, so your solutions will be graded out of 80 points.

The quality of writing and mathematical argument will be a central component in how your solutions are graded. When I return your graded solution, I will tell you what you should do to be eligible for a final 20 points. If your initial submission has no significant flaws, the final 20 points will require no additional work.

You should not obtain help from anybody or anything, except your instructor, your written class notes, the textbook (Dasgupta, Papadimitriou, & Vazirani, available on paper or on-line), and the course Web site.

1. Using induction, and without appealing to the Master Theorem, prove that $T(n) = n \log_2 n + n$ is a correct closed-form solution to the below recurrence, assuming that n is a power of 2.

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ 2T(\frac{n}{2}) + n & \text{if } n > 1 \end{cases}$$

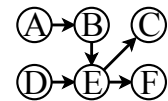
2. Suppose we have two arrays of n numbers, both known to be in increasing order. We want to find the median of the numbers in both arrays. For instance, if the arrays are $\langle 1, 2, 4 \rangle$ and $\langle 3, 5, 6 \rangle$, the median is 3.5: If you merge the arrays together, the middle two numbers are 3 and 4, which average to 3.5.

Describe an algorithm for this problem that takes $O(\log n)$ time. Argue that your algorithm is correct and that it takes the required amount of time.

3. Given an array a of numbers, we want to find the largest sum of any subsegment $a[i:j]$ of the array. For example, given the array $\langle -1, 4, -2, 3, -5 \rangle$, the largest sum is 5, achieved in this case by summing the middle three numbers.

Design a divide-and-conquer algorithm for this problem, and argue that it works correctly and that it takes $O(n \log n)$ time. (There is an $O(n)$ algorithm for this problem based on dynamic programming. However, your solution here should explicitly use the divide-and-conquer technique.)

4. Suppose we have a dag on n vertices in which there are never multiple paths from one vertex to another. This property is true of the dag at right, but if there were an edge from A to D, it would not be, since there would be two different paths from A to E (via B and via D).



Design an algorithm that computes the total number of valid topological sort orderings for such a dag. For the example dag, the output should be 6, since there are 6 valid orderings: ABDECF, ADBECF, DABECF, ABDEF C, ADBEFC, DABEFC. You should argue that your algorithm is correct and that it takes $O(n^2)$ or less time. (You can assume that multiplication and addition takes $O(1)$ time. That assumption isn't technically correct, because the numbers being added or multiplied can themselves have $O(n)$ digits.)

5. You're planning a marathon road trip, and you want to develop the cheapest plan for getting gasoline along the way. You know your vehicle holds up to 20 gallons of gasoline, and it gets 30 miles per gallon. You never stop unless your tank is below half full, and when you stop you always fill your tank — and you can't resist spending an extra \$2 on snacks with each stop.

Before you start, you get a listing of the n gas stations along the route. The listing specifies each station's location (the number of miles from Conway on the route) and the price per gallon of gasoline at that station. Using dynamic programming, design an algorithm that takes $O(n^2)$ time. Argue that your algorithm is correct and explain the role of dynamic programming in it; also, argue briefly that it meets the required time bound.