Problem 1

Does this path make me look fat? [10 points] (from Spring 2011 Final)

Consider a connected weighted directed graph \( G(V, E, w) \). Define the fatness of a path \( P \) to be the maximum weight of any edge in \( P \). Give an efficient algorithm that, given such a graph and two vertices \( s, t \in V \), finds the minimum possible fatness of a path from \( s \) to \( t \) in \( G \).
Problem 2

Passing an Evil Exam [25 points] (from Spring 2012 final)

Shaunak has designed the grading system for a new algorithms course, 6.666. Your task is to design an algorithm which computes the optimal test-taking strategy for Shaunak’s grading system.

In Shaunak’s proposal, an exam has \( n \) problems. Each problem \( i \) has a point value, \( v_i \), which is a positive number. All problems are optional. A student selects a subset \( S \) of the problems to attempt, and leaves all the other problems blank. If the student correctly answers all of the attempted problems, the student’s score is \( \sum_{i \in S} v_i \), the total point value of all attempted problems. However, if any of the answers are incorrect, the student’s score on the entire exam is 0.

Suppose that for each problem \( i \), you know the probability \( p_i \) of your answer being correct if you were to attempt that problem. The probabilities of successfully answering the attempted questions are independent: If you attempt a set \( S \) of problems, the probability of successfully solving all of those problems is \( \prod_{i \in S} p_i \). Your goal is to determine a set of problems \( S \) to attempt to maximize your expected score. Thus, you want to find a set \( S \) which maximizes:

\[
\left( \prod_{i \in S} p_i \right) \cdot \left( \sum_{i \in S} v_i \right)
\]

Throughout this problem, you may assume that arithmetic operations (+, -, *, and /) and comparisons on real numbers can be performed in constant time.

(a) [10 points] Suppose that \( v_i = 1 \) for all \( i \). Give an \( O(n \log n) \) algorithm to find an optimal subset \( S \) of problems. Briefly argue your algorithm’s correctness and analyze its running time.

Note that for full credit, you should be able to determine the actual subset \( S \), not just its expected value.

(b) [15 points] Now suppose that each \( v_i \) is a integer between 1 and \( C \). Give an \( O(Cn^2) \) algorithm to find an optimal subset \( S \) of problems. Briefly argue your algorithm’s correctness and analyze its running time.

Again, note that for full credit, you should be able to determine the actual subset \( S \), not just its expected value.
Problem 3

Guess Who? [10 points] (from Spring 2011 final)

Woody the woodcutter will cut a given log of wood, at any place you choose, for a price equal to the length of the given log. Suppose you have a log of length $L$, marked to be cut in $n$ different locations labeled 1, 2, . . . , $n$. For simplicity, let indices 0 and $n + 1$ denote the left and right endpoints of the original log of length $L$. Let $d_i$ denote the distance of mark $i$ from the left end of the log, and assume that $0 = d_0 < d_1 < d_2 < \ldots < d_n < d_{n+1} = L$. The wood-cutting problem is the problem of determining the sequence of cuts to the log that will cut the log at all the marked places and minimize your total payment. Give an efficient algorithm to solve this problem.
Problem 4

Sorting Fluff [20 points] (from Fall 2011 final)

In your latest dream, you find yourself in a prison in the sky. In order to be released, you must order \(N\) balls of fluff according to their weights. Fluff is really light, so weighing the balls requires great care. Your prison cell has the following instruments:

- A magic balance scale with 3 pans. When given 3 balls of fluff, the scale will point out the ball with the median weight. The scale only works reliably when each pan has exactly 1 ball of fluff in it. Let \(\text{MEDIAN}(x, y, z)\) be the result of weighing balls \(x, y\) and \(z\), which is the ball with the median weight. If \(\text{MEDIAN}(x, y, z) = y\), that means that either \(x < y < z\) or \(z < y < x\).

- A high-precision classical balance scale. This scale takes 2 balls of fluff, and points out which ball is lighter; however, because fluff is very light, the scale can only distinguish between the overall lightest and the overall heaviest balls of fluff. Comparing any other balls will not yield reliable results. Let \(\text{LIGHTEST}(a, b)\) be the result of weighing balls \(a\) and \(b\). If \(a\) is the lightest ball and \(b\) is the heaviest ball, \(\text{LIGHTEST}(a, b) = a\). Conversely, if \(a\) is the heaviest ball and \(b\) is the lightest ball, \(\text{LIGHTEST}(a, b) = b\). Otherwise, \(\text{LIGHTEST}(a, b)\)'s return value is unreliable.

On the bright side, you can assume that all \(N\) balls have different weights. Naturally, you want to sort the balls using as few weighings as possible, so you can escape your dream quickly and wake up before 4:30pm!

To ponder this challenge, you take a nap and enter a second dream within your first dream. In the second dream, a fairy shows you the lightest and the heaviest balls of fluff, but she doesn’t tell you which is which.

(a) [4 points] Given \(l\), the lightest ball \(l\) pointed out by the fairy, use \(O(1)\) calls to \(\text{MEDIAN}\) to implement \(\text{LIGHTER}(a, b)\), which returns \(\text{TRUE}\) if ball \(a\) is lighter than ball \(b\), and \(\text{FALSE}\) otherwise.

After waking up from your second dream and returning to the first dream, you realize that there is no fairy. Solve the problem parts below without the information that the fairy would have given you.

(b) [6 points] Give an algorithm that uses \(O(N)\) calls to \(\text{MEDIAN}\) to find the heaviest and lightest balls of fluff, without identifying which is the heaviest and which is the lightest.

(c) [2 points] Explain how the previous parts should be put together to sort the \(N\) balls of fluff using \(O(N \log N)\) calls to \(\text{MEDIAN}\) and \(O(1)\) calls to \(\text{LIGHTEST}\).

(d) [6 points] Argue that you need at least \(\Omega(N \log N)\) calls to \(\text{MEDIAN}\) to sort the \(N\) fluff balls.