1. Your favorite sawmill charges by the length to cut a piece of lumber. For example, to cut a 7 foot piece (the cut may be anywhere) costs 7 cents, and to cut a 10 foot piece costs 10 cents. As an example, suppose you bring them the following 29 foot board, with the cuts to be made as shown.

If you make the cuts in the (non-optimal) order cut 1, cut 2, cut 3, cut 4, your cost is $29 + 25 + 15 + 10 = 79$. An optimal order is cut 2, cut 1, cut 3, cut 4, with cost $29 + 14 + 15 + 10 = 68$.

At first glance, the following "solution" seems optimal: always make the next cut as close to the middle of some board as possible. Show that this does not always produce an optimal solution. Hint: make up a new example, don’t use the one here.

2. Design and analyze an efficient solution to solve the sawmill problem above. Hint: use dynamic programming.

3. Given a sorted array of distinct integers $A[1..n]$, determine whether or not there exists an index $i$ such that $A[i] = i$. Your algorithm should run in $O(\log n)$.

4. The following problem, called the supersink problem, was the first counterexample to the conjecture that any nontrivial problem on an arbitrary $n \times n$ binary matrix requires $\Omega(n^2)$ time. A supersink $s$ in a matrix is an index such that $A[s, y] = 0$ for all values of $y$, and $A[x, s] = 1$ for all $x \neq s$. Give an $O(n)$ algorithm to determine whether a binary matrix has a supersink.