Problem 1. Guess Who? (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 $1, 2, \ldots, n$. 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.

Solution: Dynamic programming. $c(i, j) = \min_{i<k<j}\{c(i, k) + c(k, j) + (d_j - d_i)\}$ where $c(i, j)$ is the min cost of cutting a log with left endpoint i and right endpoint j at all its marked locations. Start with $c(i, i+1)$ (consecutive cuts) and move outwards to $c(i, i+2)$, $c(i, i+3)$ until the maximal distance $c(1, n)$, which gives the optimal score for the whole wood. Remember pointers to k that gave max score at each step, and trace back pointers to construct optimal solution. Each iteration takes $O(n)$ (linear search between i and j) and there are $O(n^2)$ entries to fill (a triangle really, not a square). Greedy solutions that pick the maximum cut each time do not work. Similarly, heuristics like picking the point closest to the center do not work.

Problem 2. I Am Locutus of Borg, You Will Respond To My Questions [24 points]

Upon arrival at the planet Vertex T, you and Ensign Treaps are captured by the Borg. They promptly throw the ensign out of the airlock. You had better solve their problem, lest you share his fate.

The Vertex T parking lot is an $n \times n$ matrix A. There are already a number of spaceships parked in the lot. For $0 \leq i, j < n$, let $A[i][j] = 0$ if there is a ship occupying position (i, j), and 1 otherwise.

The Borg want to find the ***largest square parking space*** in which to park the Borg Cube. That is, find the largest k such that there exists a $k \times k$ square in A containing all ones and no zeros. In the example figure, the solution is 3, as illustrated by the 3×3 highlighted box.

	0	1	2	3	4
0	0	1	1	1	0
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	0	0	0
4	1	1	1	0	1

Describe an efficient algorithm that finds the size of the largest square parking space in A. Analyze the running time of your algorithm.

Hint: Call $A[0][0]$ be the *top-left* of the parking lot, and call $A[n-1][n-1]$ the *bottom-right*. Use dynamic programming, with the subproblem $S[i, j]$ being the side length of the largest square parking space whose bottom-right corner is at (i, j).

Solution: The base cases are the positions along the top and left sides of the parking lot. In each of these positions, you can fit a 1×1 square parking space if and only if the space is unoccupied. Thus, for the base cases ($i = 0$ or $j = 0$), we have $S[i, j] = A[i][j]$.

Now for the general case. Again, if $A[i][j] = 0$, then $S[i, j] = 0$, so we'll only consider the case where $A[i][j] = 1$. There is a parking space of size x with bottom-right corner at (i, j) if and only if there are three (overlapping) spaces of size $x - 1$ at each of the locations $(i - 1, j)$, $(i, j - 1)$, and $(i - 1, j - 1)$. Thus, the largest possible parking space is

$$S[i, j] = 1 + \min(\ S[i-1, j]\ ,\ S[i, j-1]\ ,\ S[i-1, j-1]\ )$$

The desired answer is $\max_{i,j} S[i, j]$, which requires $O(n^2)$ to compute.

There are n^2 subproblems, and each takes $O(1)$ time to solve, for a time of $O(n^2)$. This, added to the $O(n^2)$ to extract the desired answer, gives a total $O(n^2)$ running time, which is clearly optimal because all the data must be examined.