You are given an array A of n integers. Some of these elements are marked as *dodgy*, and you want to find the length of the longest increasing subsequence that include at most k dodgy elements. Given an array $A[1..n]$, a boolean array $\textsc{Dodgy}[1..n]$, and an integer k, describe an $O(kn^2)$ algorithm to compute the length of the longest increasing subsequence that contains at most k dodgy elements.

For example, if we have the array

$$A = [3^*, 1, 4^*, 1^*, 5, 9, 2^*, 6],$$

where all of the dodgy elements are marked by an asterisk $(*)$, then a longest increasing subsequence containing at most 3 dodgy elements is $3^*, 4^*, 5, 9$. Your algorithm should, then, return 4.

Solution. Set up a table to represent a Dynamic Programming table, say DP, with dimensions $n \times (n+1) \times (k+1)$, initialised all as 0. The first dimension will keep track of the index in A, the second will be the last element in the subsequence we are considering and the third will be the number of dodgy elements included. Now, for each element's index i in A, we must iterate through all of the possible values for the last element j (from 1 to n). For each amount of dodgy elements d (0 to k), we can fill out our table.

If $A[i] > j$, update $DP[i][j][d']$ (where $d' = d + 1$ if the element is dodgy, and d otherwise) to contain the maximum of it's current value and $DP[i'][j][d'] + 1$ for all $i' < i$. The largest value in the table will be length that we're looking for. Since we are checking each value in our array, we will have $O(kn^2)$.