You are given a long piece of stick of length L. You want to cut the stick into *exactly* n places along its length, where the ith place to cut occurs at position $A[i]$. Since sticks of larger size require more power, cutting a stick of length X requires X units of work.

(a) Given the stick of length L and the positions of the n places to cut $A[1..n]$, describe an $O(n^3)$ algorithm to compute the minimum number of units of work to cut the stick into n pieces.

(b) Describe an $O(n^2)$ algorithm to compute the minimum number of units of work to cut the stick into n pieces.

Hint. *What can we optimise here?*

Solution.

(a) We initialise a Dynamic Programming table DP, which will represent a $(n+2) \times (n+2)$ array. $DP[i][j]$ represents the minimum units of work required to cut the stick from position i to position j, where we also consider 0 and L as potential cutting positions. Clearly, for all $i = 0, \ldots, L$, we have $DP[i][i+1] = DP[i][i] = 0$. For each possible length, l, of the stick segment (2 to $n+1$), we run a second loop for each position i, we define $j = i + l - 1$, and then for each cut position k between i and j, we calculate the minimum work required, updating $DP[i][j]$ respectively. We are running three nested loops, so we have a time complexity of $O(n^3)$. Our final result will simply be $DP[0][n+1]$.

(b) To optimise our solution, we should consider the factthat the optimal way to cut a segment does not depend on the cuts made outside of that segment.