Qiyao Zhou

Z5379852

Question 1

From this problem, we can see that dynamic programming has a very important role in the question.

Subproblems： We can solve this problem by considering the subproblem span [i, j]: For every 0<=i<k<j<=n, span [i, j] = min (span [i, k] + span [k, j]) while span [i, j] = $(j-i)^2$ + max (A [i... j]).

Recurrence: For 0<=i< j<=n, k = argmin(span [i, j]). We can then continue the algorithm for both spans [i, k] and [k, j] until k=i+1 or j=k+1. And k is where we put pillars in the best solution.

Base case： no pillars, the total cost would be: span [0, n] = n^2 + max (A [1...n]).

Order of computation: when we get 2 subproblems spans [i, k] and [k, j], we can solve the subproblems span [i, k] first.

Final answer: The maximum total cost is span [0, n] at last. And k which we get in every recurrence is where we decide to put pillars in the best solution.

Time complexity: As for each span [i, j] we only compute once using dynamic programming, 0<=i< j<=n. So, we need to compute n^2 times. Every time we need o (1) so the overall time complexity is o (n^2).