Comp3121 assignment 4 Question 1
Chen En (z5335039)

Question 1 *Bridge Support*

[20 marks] The NSW government has decided to build a new bridge. The bridge is to be n meters long, where $A[i]$ is the maximum load on the bridge between $(i-1)$ meters and i meters.

The government have contracted you to complete this project, and you've decided to build the bridge in spans of integer length. The cost of building a span is given by the length of the span squared, plus the maximum load that the span needs to withstand. The total cost of building the bridge is the sum of the cost of all spans. More explicitly:

$$\texttt{total\_cost} = \sum_{s \in \text{spans}} (s_R - s_L)^2 + \max(A[(s_L+1)..s_R])$$

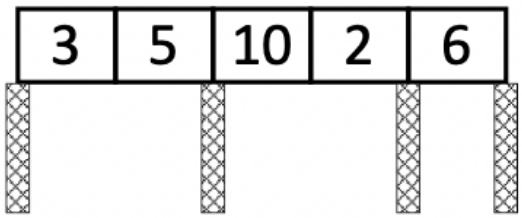
where s_L and s_R are the left and right endpoints respectively of the span s.

Your goal is to determine the **minimum** cost required to build a bridge that supports the load.

An example for the cost calculation: if $n = 5$ and $A = [3, 5, 10, 2, 6]$, and you decided to put pillars in at 2 meters and 4 meters, the total cost would be:

$\texttt{span\_[0,2]} = (2-0)^2 + \max(A[1..2]) = \$9$
$\texttt{span\_[2,4]} = (4-2)^2 + \max(A[3..4]) = \$14$
$\texttt{span\_[4,5]} = (5-4)^2 + \max(A[5..5]) = \$7$
$\texttt{total\_cost} = \$9 + \$14 + \$7 = \$30$

A depiction of this example is shown in image below.



1.1 **[14 marks]** Design an $O(n^3)$ algorithm which achieves your goal.

1.2 **[6 marks]** With an election looming, the NSW government has asked you to hurry up. Design an algorithm that achieves the goal in $O(n^2)$.

You may choose to skip Question 1.1, in which case your solution to Question 1.2 will also be submitted for Question 1.1.

Subproblems:
For each $0 \leq i \leq n$ let $Q(i)$ be the problem of determining $Opt(i)$, which is the minimum cost of the bridge with span $[0, i]$. (To generate $Opt(i)$ we need subproblem $Opt(i-1)$, and we can get those value from taking increasing order of i.)

Base case:
$Opt(0) = 0$ which presents the bridge with $span[0,0]$ the minimum cost is 0.

Recursion:

$$\boldsymbol{Opt(i) = min\left(Opt(m) + (i-m)^2 + T[m+1][i]\right) for\ 0 \leq m < i}$$

1.1
Given by the question, our goal is to find the minimum cost of span $[0, i]$. For each $0 \leq m \leq i$ that m is the closest pillar position to i. To build the bridge with $span\ [0, m]$ which costs $Opt(m)$. We can generate other cases by adding extra cost. That is, to calculate the cost that we extend the $span[0, m]$ to $span\ [0, i]$ we can present it as

$$Opt(i) = min(Opt(m) + (i-m)^2 + \max\ (A[(m+1) \ldots i])) \ for\ 0 \leq m < i\ .$$

1.2
We create a $n * n$ table $T[1 \ldots n][1 \ldots n]$ $that\ T[i][j]$ stores the maximum value of $A[i \ldots j]$. For each $1 \leq i \leq n$ and $i \leq j \leq n$, that i…j is the range that we are finding the maximum value. We first store the values that are diagonal on the graph,that is, $T[i][i] = A[i]$. Since there are n position, we iterate all possibility of each j to see which value on position j is lager than $\max\ (A[i \ldots m-1])$) . We store the larger one to the table, which can be presented as $T[i][j] = max(A[j], T[i][j-1])$. As a result, we get the recursion formula

$$Opt(i) = min\ (Opt(m) + (i-m)^2 + T[m+1][i])\ for\ 0 \leq m < i\ .$$

Answer: $Opt(n)$

Justification:
We prove the optimal subproblem by contradiction. For each $1 \leq k < i$ which k is the previous pillar before i. $Opt(i)$ is the minimum cost when the last pillar at position i. As the recursion is determined above, that $Opt(i) = min\ (Opt(k) + (i-k)^2 + T[k+1][i])$.
Suppose there exists another minimum cost $Opt'(k)$that is less than $Opt(k)$. That is, we can find another minimum cost by $Opt'(k)$ which contradicts our previous assumption that $Opt(i)$is the minimum cost.

Time complexity
As create an n*n table with n^2 elements that takes $O(n^2)$ since storing the value takes linear time, then we go through all n positions of m in each subproblem which takes $O(n)$ since we store the value into table, and to look up the value takes $O(1)$. There are total n subproblems. Thus, the time complexity is $O(2n^2) = O(n)$.