Algorithms
Dynamic Programming Project (100 pts total, 30 pts for this part)
Timber Problem

You are not allowed to use the internet or consult any external references. You may use lecture slides.

1 Problem Description

1.1 Introduction

The timber problem discussed in class and on the supplemental handout is briefly defined as follows: Given an array of n positive integers representing the length of segments that a tree will be split into, what is the maximum length of wood that you can leave the sawmill with if you can only take one segment at a time from the end of the log, alternating picks with a neighbor who is your intellectual equal.

1.2 Recurrence Relation

Recall from the handout that the recurrence relation for the timber problem is:

$$T(i,j) = \sum_{k=i}^{j} l_k - \min[T(i+1,j), T(i,j-1)]$$

$$\text{Base Case: } T(i,i) = l_i$$

2 Deliverables - Part 2 - Dynamic Programming

Please submit all deliverables as requested below for each question.

1. [10] Implement a dynamic programming algorithm (that uses a table to avoid recomputation) to compute the maximum sum of lengths that can be achieved. Either a top-down or bottom-up approach will be accepted, but note that a bottom-up approach is typically more efficient in practice and simplifies the process of determining the algorithm's asymptotic complexity. Submit your code to Gradescope for verification. You will have three opportunities (no extra credit) to submit your code without penalty. Subsequent submissions will incur a penalty of 10% per additional attempt.

2. [20] Derive the asymptotic complexity and provide an exact bound (i.e., use Θ) for the dynamic programming algorithm. Run the code from the previous algorithm on several different sizes of n to characterize the growth in runtime as n increases. Use a random number generator (RNG) to create the arrays. Provide a table or plot of your results, and compare/contrast the timing results with the derived asymptotic complexity (this means using a regression on the run-time results or using ratios to derive the apparent complexity from the runtimes for different sizes of n, and comparing this to the asymptotic complexity). Submit your response to this question to Gradescope.

   Include your code either as you derive its complexity or in an appendix. Note: only mention the dominating aspects that determine the overall complexity in your discussion.

Input Format
The input consists of two lines:

- The first line contains $1 \leq n \leq 2000$, the number of segments in the tree. ***n* can be odd or even.**
- The second line contains n space-separated integers giving the value of each tree segment from left to right. The value of each tree segment t will be in the range $1 \leq t \leq 1000$.

Output Format
The output consists of one line:

- On the first line, print the maximum sum of lengths that you can take.

For example, an input of

```
4
5 6 9 7
```

would have the following output:

```
14
```