Algorithms
Dynamic Programming Project (100 pts)
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 an even number, 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:

$$T(i,j) = \max\Big(l_i + \min\big[T(i+2,j), T(i+1,j-1)\big], l_j + \min\big[T(i+1,j-1), T(i,j-2)\big]\Big)$$

$$\text{Base Cases: } \begin{cases} T(i,j) = l_i \text{ when } j = i \\ T(i,j) = \max(l_i, l_j) \text{ when } j = i+1 \end{cases}$$

2 Deliverables

Please submit all of the items requested below in a single PDF file on Canvas.

1. [20] Write a recursive algorithm to solve the problem for the tree represented as [33, 28, 35, 25, 29, 34, 28, 32]. Implement the recurrence relation as-is, meaning your function should make two recursive calls to $T(i+1, j-1)$. *You may lose points if you do not make the redundant recursive call.* Including the initial call, how many calls are made to the function? Include your code in the report.
2. [10] Run the code from the previous algorithm on a few different sizes on 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 discuss.

3. [20] Implement a dynamic programming algorithm (that uses a table to avoid recomputation) to compute the maximum sum of lengths that can be achieved. Include code in your report.

4. [10] Run the code from the previous algorithm on a few different sizes of n to characterize the growth in runtime as n increases. Use a RNG to create the arrays. Provide a table or plot of your results and discuss.

5. [10] Implement a traceback step that identifies the order in which the segments are taken by both you and your neighbor (your choices are the 1st, 3rd, 5th, etc). Include code in your report.

   For consistency with our solutions, if presented with two choices that result in the same optimal outcome, choose the tree segment that is on the left/bottom side of the tree (the i segment, not the j segment). (Note that this does not necessarily mean that $l_i \geq l_j$.)

6. [20] Demonstrate that your code works correctly by showing its results on the following instance ($n = 20$):

   [33, 28, 35, 23, 23, 25, 37, 40, 42, 24, 38, 29, 22, 40, 36, 42, 39, 37, 45, 32]

   Output Format
   The output consists of two lines:

   - The first line prints maximum sum of lengths that you can take.
   - The second line prints the segment numbers in the order in which they are taken.

   For the example [5, 6, 9, 7] the output would look like:

   ```
   14
   1 2 3 4
   ```

7. [10] Demonstrate that your code works for larger values of n by correctly solving the instance of the problem in the Timber Verification quiz on Canvas. For this, you only need to give the maximum sum of lengths, not the traceback step.