Timber Problem

1 Problem Description

1.1 Introduction

A particularly valuable length of timber has fallen directly on your property line. As the lumber from the tree has spectacular potential for your handmade furniture business, you lay claim to it. However, your neighbor also sees great potential in the wood for their cabinetry business. Since you and your neighbor are arch-rivals, the case ends up in front of a judge in small claims court.

While examining the evidence, the honorable judge notices that the tree has an odd number of defects interspersed along its length at which the lumber will have to be cut regardless. These cuts will result in an even number of flawless sections of varying lengths. Thus, the judge orders the following resolution: You and your neighbor are to take the tree to the local sawmill, where you will each alternate turns claiming and cutting off either end of the log at the defects until you each have half of its n segments. Since you were first to lay claim to the timber, the judge grants you the first choice.

Your goal is to maximize the length of wood that you get to take from the sawmill by strategically picking which end of the tree to cut at each turn. *(Taking the longer of the two lengths at every turn does not guarantee you the maximum length.)*

1.2 Assumptions

1. The tree maintains a consistent girth along its entire length. You need only concern yourself with the length of each segment, not which end of the tree it comes from.
2. The length of each tree segment has been measured beforehand and all of the information is made available to both you and your neighbor. (For our purposes, this will be as an array, with index 1 representing the bottom of the tree and n the top.)
3. Your neighbor is also a Mines student and can be assumed to employ the best strategy in order to maximize their own amount of lumber.
4. Only one cut can be made in a given turn, i.e. you can only take a single segment and it must be at the end of the log in the current turn.
5. The number of segments in the tree, n, will always be even.

2 Recurrence Relation

2.1 Notation

An important step in dynamic programming is defining a system of notation that provides the necessary means to get you to a solution. In this problem, we'll need a way to label the segments of the log and their lengths, as well as a method of representing sub-problems. We'll do this in the following ways:

- Let the segments of the log be numbered from bottom to top as $1, 2, ..., n$.
- Let the two segments that are currently at the ends of the log be known as the i-th segment and the j-th segment, with the i-th being on the bottom-most end of the log and the j-th being at the top-most end. Thus, before the log is cut, $i = 1$ and $j = n$. Also, $i \leq j$ always.
- Let l_i denote the length of the i-th segment.
- Let $T(i, j)$ represent the maximum sum of lengths that you can achieve from the portion of the log that begins with the i-th segment and ends with the j-th. Thus, to solve the entire problem you need to find $T(1, n)$.

2.2 Recurrence

Solving the timber problem recursively requires the acknowledgement that in a given turn, you have two possible choices, the i-th (bottom) or the j-th (top) segment. The two choices reduce the problem into the following two sub-problems:

1. If you choose to take the i-th segment, you leave your neighbor with the choice of the $(i+1)$-th segment or the j-th segment.
2. If you choose the j-th segment, you leave your neighbor to choose between the i-th segment or the $(j-1)$-th segment.

Following your choice of segment, we can assume that your neighbor is going to take the next segment that will maximize their haul. However, since it is a zero-sum situation we can also look at their choice as attempting to minimize your remaining haul. Thus, your neighbor will look at the two sub-problems that result from the options that are available to them and allow you the minimum possible outcome. The choices available to them vary depending on your initial choice, so the recurrence relation for the timber problem is as follows:

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

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

3 Primer Questions

1. What situations do the two base cases represent?
2. How do we avoid adding your neighbor's lumber total to your own?
3. If n is guaranteed to be even, do we need both base cases? Why or why not?