You’re a lumberjack, and it’s OK..

Sam Khosravi

Spring Term 2022

Introduction

The goal of this task was to by using dynamic programming to when given a sequence of different lengths to find the best way to cut a log and reduce the cost of the cutting.

Split

In this task all code was given, except the code added below. The split basically is made to split our sequence in any way possible, and will return a list of all ways the log could be split in form of a tuple. Depending on the length of list we send into split, we will get a output of 2 to the power of n. So if our list consists of 1 and 2, we will get an output of 2 to the power of 2 which is 4 different tuples. For a list with 1, 2 and 3 we would have an output of 2 to the power of 3 which is a list of 8 tuples. This pattern continues for all lengths of lists.

```
def split(seq) do split(seq, 0, [], []) end
def split([], l, left, right) do
[{left, right, l}]
end
def split([s|rest], l, left, right) do
split(rest, l+s, [s|left], right) ++ split(rest, l+s, left, [s|right])
end
```

Dynamic vs Recursive

The code to calculate cost was given in the dynamic programming pdf, and will not be included here. What I basically did was to use the split function with small tweaks to make it calculate cost instead. This was implemented recursively and dynamically.

The recursive solution consisted of dividing the sequence by two followed by cutting the log by two again and lastly applying the function to make the cost of cutting the two logs as small as possible. The function is then remade to return tuples that give the amount of minimum cost and also the way to cut. To dynamically solve the task we would need a memory to store

sub parts of the task. As soon as a problem that has occured before occurs, we go to the memory and the problem is solved as done the first time it was encountered. The memory was updated and returned during every iteration together with the minimum cost.

The difference between the dynamic and recursive function are represented in the table:

elements	**Dynamic**	**Recursive**
6	0.66 s	0.01 s
8	2.9 s	0.39 s
10	102 s	188 s

Table 1: Difference between dynamic and recursive implementation of cost

Discussion

I felt like the implementation of a more effective memory was redundant to the task which is why I did not have any explanation on it. However, this was the absolutely coolest part of this lab, as I feel like we ”out-smarted” the computer. I however understand that this is only the case here because of the order of our numbers. Furthermore, I feel like I know more about dynamic programming, but I still don’t think I really know how powerful it really is. I had many moments where I felt stuck, and had a hard time continuing. For example, just understanding how the split would work was extremely confusing and took a lot of time, and as this was the part that took the longest for me, it is also the part I felt I needed to explain a bit, especially as the rest of the code in one way or another was derived from our split. Even though we had a few videos on dynamic programming, I missed the live lectures, which I feel is why I struggled so much. I did not find a lot on the internet so everything I had was the online lectures and my classmates, who were equally as confused.