Comp 526

Robert Johnson

Programming Assignment 1

The problem.

The bamboo cutting problem consists of bamboo shoots that grow at different rates per day. We would like to find a repeating sequence to cut these bamboo shoots in a way that minimizes the maximum height over all time with the restriction that we can only cut 1 bamboo shoot a day.

Due to already finding optimal solutions previously for this assignment I have been given a modified version which allows us to cut 2 bamboo shoots per day instead of 1.

My General Strategy

To find the optimal perque I created a separate program, this has been given in conjunction with the perque's required for the assignment.

The program works in the following way, we define a "max height restriction" before running the program, this is a value that we are sure can be achieved for example, in the integer bamboo shoot problem we can guarantee that we can find a ordering which is less than or equal to 24 using the rudimentary sequence (1,2)->(2,3)->(3,4)->(5,6)->(7,8).

We use this max height restriction in our program to reduce the search space in the following way. We conduct a normal brute force algorithm trying to find a non-repeating sequence that maintains our height restriction for a set number of cuts. For example we start by defining that we would like to find a non-repeating sequence of length 2000 that maintains our height restriction of 24. In a normal brute force approach we would have to test all 8^2000 combinations of cuts and take the minimum however we can reduce the amount of tests significantly by simply ensuring that we do not carry out tests on sequences which have already hit a max height above 24 in a previous time step.

For example in the cutting sequence (1,2)->(1,2)->(1,2)->(1,2) we have already passed our restriction of 24 having a max height of 32, we can state that any sequence following from this subsequence can never be less than 24 and therefore be discarded, we can continue our brute force approach searching (1,2)->(1,2)->(1,2)->(1,3) which also fails to maintain our restriction we can see that this vastly reduces our search space.

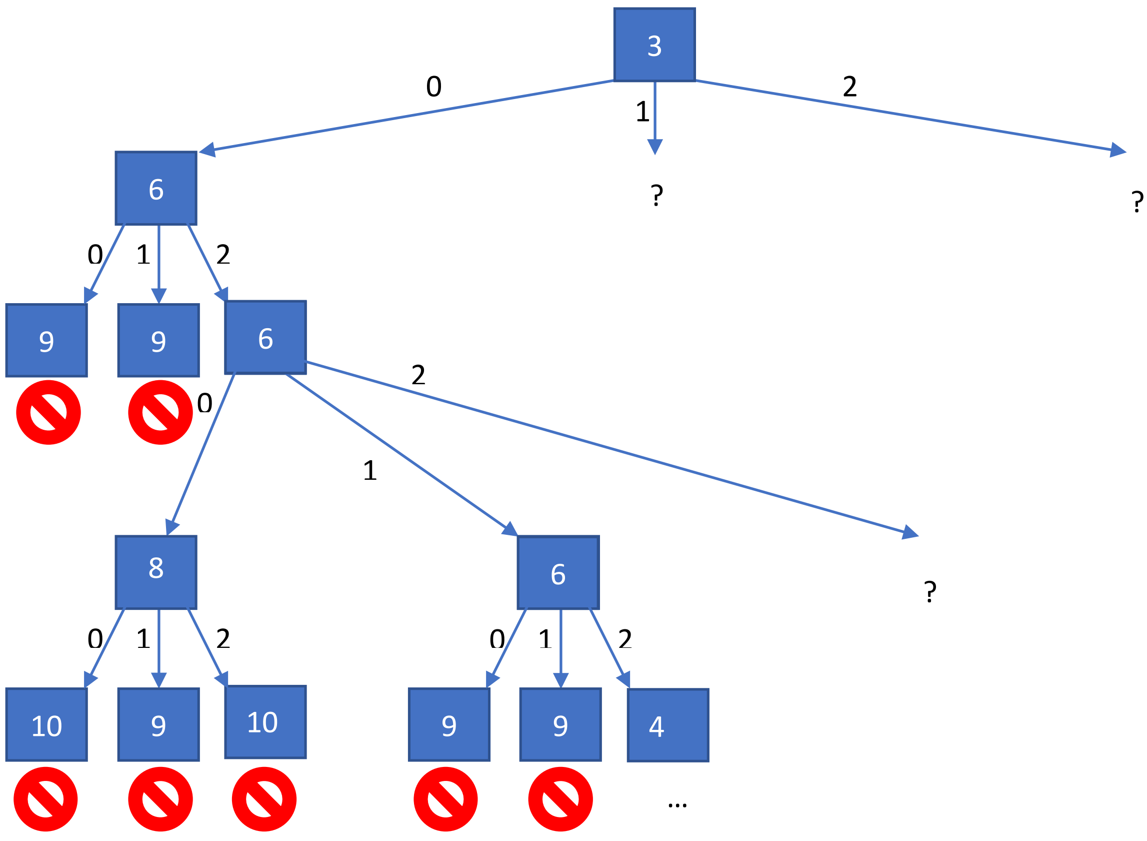
We can see that the sequence (1,2)->(1,2)->(1,2) is used in both examples we can therefore chain attempts by simply resetting our height values to their previous values at t-1 rather than calculating from the starting point and repeating the steps.

Once finding a non-repeating sequence that maintains our max height restriction we look for a subsequence that repeats within the main sequence, to aid in this it helps to know a few things about our non-repeating sequence first:

- Due to all bamboo shoots starting from a max height of 0, the first few iterations are almost trivial. To show this consider the max height restriction of 24 with cuts (a)->(b)->(c)->(7,8) we can see that no matter the cuts decided for a, b and c our height restriction will always be maintained by following this sequence by (5,6)->(3,4)->(1,2) which can then be repeated.
- Due to the program being stopped at a set time step it is not guaranteed that future cuts are able to be added to our sequence without breaking our height restriction.

Because of these two points we are able to remove the start and end of our sequence without having a major impact to our sequence as a whole. Additionally given a long enough non-repeating sequence with static rules for choosing the next bamboo shoot to cut our reduced sequence should contain multiple repeating subsequence within it, our program aims to find this repeating sequence. To guarantee optimality we can simply continue to reduce our max height restriction until we are unable to find non-repeating sequence that maintains it.

<u>Thinking of the problem as a tree given a max height restriction of 8 with only 3 bamboo shoots with growth rate {1,2,3} (edge weights indicate bamboo cut, node weights indicate current maxHeight) with only one bamboo shoot being cut a day.</u>



We test our paths using post order traversal cutting our tree if the node found is greater than our max height restriction. Although this is only testing using 1 bamboo shoot being cut a day we can see that this can be expanded to cutting 2 bamboo shoots by simply replacing edges to represent the cutting of two bamboo's instead of 1 ({0,1},{0,2},{1,2}).

My programmatic solution

```
Start from our bamboo shoots in T = 1
While(Iterations<maxIterations && lengthOfCuts < maxLengthOfSequence)
    Count how many bamboo shoots are above maxHeight
    If (Count>2)
        Reset to T-1 values
    Else If (Count == 2)
        If (we haven't previously cut at this T)
            Cut bamboo shoots that are above maxHeight
        Else
            Reset to T-1 values
    Else If (Count == 1)
        If (we haven't previously cut at this T)
            Find index of bamboo over maxHeight as i
            Set i to 0
            If I == 0
                Cut bamboo at Index 1 store index 1
            Else
                Cut bamboo at Index 0 store index 0
        Else
            Find index of bamboo over maxHeight as i
            Set i to 0
            Get previous index of last cut as j
            Set j to j + 1
            If(j==i)
                j = j + 1
            If(j>indexSize)
                Reset to T-1 values
            Else
                Cut bamboo at index j
    Else if (Count == 0)
        If (we haven't previously cut at this T)
            Cut 0 store index 0
            Cut 1 store index 1
        Else
            Get previous index as i
            Get previous index as j
            If(j+1>=indexSize)
                i++
                if(i==indexSize-1)
                    Reset to T-1 values
                Else
                    j = i + 1
            else
                j = j + 1;
                Cut i store index i
                Cut j store index j
```

Results

Integer Perque

Using the Strategy described above I managed to reduce my results to
Quotient= 1.2222222 MaxHeight= 22 HalfH= 18.0

Power Perque

Using the Strategy described above I managed to reduce my results to
Quotient= 1.0039216 MaxHeight= 128 HalfH= 127.5

Primes Perque

Using the Strategy described above I managed to reduce my results to
Quotient= 1.3246753 MaxHeight= 51 HalfH= 38.5