Rod cutting Algorithm

An example of Dynamic programming

Question-

Given an integer N denoting the Length of a line segment. you need to cut the line segment in such a way that the cut length of a line segment each time is integer either x , y or z. and after performing all cutting operation the total number of cutted segments must be maximum.

Approach-

First approach that may come to your mind could be that if we have to maximise the number of segments we can just choose the minimum one out of the three and see how many segments it can make. But there's a big problem with that approach. How would you know that the smallest out of x, y, z would be exactly divisible by n because if it's not than we can have a piece left which doesn't fit either of x, y or z. So now after some pondering you may realise that the only realistic way to find the maximum is to try everything.

We will first develop a recursive solution to this approach and then try to memoise it in order to make it run in polynomial time. If you remember solution of fibonacci with DP this will be very intuitive.

Let's make the base case first, if n==0 then the answer would be 0. If n becomes less than 0 then we won't want that case to be considered. So we'll start the code as (Note- I assume x, y and z to be global for this code.)

```
def rodCut(n,memo):
    if n==0:
        return 0
    if n<0:
        return -999999
```

We return -999999 so that if any n less than zero comes to the function it will automatically be disqualified from maximum race. Now just like fibonacci we can simply call the recursive solution with something like-

```
return max(rodCut(n-x)+1, rodCut(n-y)+1, rodCut(n-z)+1)
```

So now we can see that recursively we call the function for each one n-x, n-y and n-z. We add one because these cuts would add a segment in total number of segments.

Now we have reached to a solution but this is a brute force solution and runs in exponential time which is obviously not good and you may even be able to see that we are repeating a lot of steps. We are calculating the function for the same input a lot of times. So one smart thing to do could be to store the solution of inputs as we find them. So now let's make a list and we'll keep storing the solutions in that list.

```
memo=[-1 for i in range(n)]
```

Now that we have made the list we can start storing the solution in it.

```
def rodCut(n,memo):
    if n==0:
        return 0
    if n<0:
        return -999999
    if memo[n]!=-1:
        return memo[n]
     memo[n]=max(rodCut(n-x,memo)+1,rodCut(n-y,memo)
+1,rodCut(n-z,memo)+1)
     return memo[n]
```

This solution makes a lot of sense because It will run in O(n) time and give the most optimum solution.