Rod cutting problem

Given a rod of length n inches and an array of prices that includes prices of all pieces of size smaller than n. Determine the maximum value obtainable by cutting up the rod and selling the pieces. For example, if the length of the rod is 8 and the values of different pieces are given as the following, then the maximum obtainable value is 22 (by cutting in two pieces of lengths 2 and 6)

length | 1 2 3 4 5 6 7 8

--------------------------------------------

price | 1 5 8 9 10 17 17 20

And if the prices are as following, then the maximum obtainable value is 24 (by cutting in eight pieces of length 1)

length | 1 2 3 4 5 6 7 8

--------------------------------------------

price | 3 5 8 9 10 17 17 20

A naive solution to this problem is to generate all configurations of different pieces and find the highest-priced configuration. This solution is exponential in terms of time complexity. Let us see how this problem possesses both important properties of a Dynamic Programming (DP) Problem and can efficiently be solved using Dynamic Programming.  
**1) Optimal Substructure:**  
We can get the best price by making a cut at different positions and comparing the values obtained after a cut. We can recursively call the same function for a piece obtained after a cut.  
Let cutRod(n) be the required (best possible price) value for a rod of length n. cutRod(n) can be written as follows.  
cutRod(n) = max(price[i] + cutRod(n-i-1)) for all i in {0, 1 .. n-1}  
**2) Overlapping Subproblems**   
The following is a simple recursive implementation of the Rod Cutting problem.

The implementation simply follows the recursive structure mentioned above.

**Output**

Maximum Obtainable Value is 22

Considering the above implementation, the following is the recursion tree for a Rod of length 4.

cR() ---> cutRod()

cR(4)

/ /

/ /

cR(3) cR(2) cR(1) cR(0)

/ | / |

/ | / |

cR(2) cR(1) cR(0) cR(1) cR(0) cR(0)

/ | |

/ | |

cR(1) cR(0) cR(0) cR(0)

/

/

CR(0)

In the above partial recursion tree, cR(2) is solved twice. We can see that there are many subproblems that are solved again and again. Since the same subproblems are called again, this problem has the Overlapping Subproblems property. So the Rod Cutting problem has both properties (see [this](https://www.geeksforgeeks.org/overlapping-subproblems-property-in-dynamic-programming-dp-1/)and [this](https://www.geeksforgeeks.org/optimal-substructure-property-in-dynamic-programming-dp-2/)) of a dynamic programming problem. Like other typical [Dynamic Programming(DP) problems](https://www.geeksforgeeks.org/archives/tag/dynamic-programming), recomputations of the same subproblems can be avoided by constructing a temporary array val[] in a bottom-up manner.