Recursive Approach.

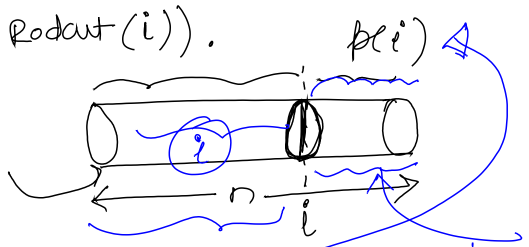
17/Feb/2022
12:51 PM

We have to generate the all configurations of different pieces, and find the highest priced configuration.

So, we can get the best price (maximum revenue) by making a cut at different positions and comparing the values obtained after a cut.

If Rodcut(n) is the function that gives the value of r(n), then

$$\text{Rodcut}(n) = \max(P(i) + \text{Rodcut}(i)).$$



So the Earning in Rodcut(n)

You earned profit p(i)

$$\text{Rodcut}(n) = \max\left(P(i) + \text{Rodcut}(n-i)\right)$$

Confusing, let's try to ~~Explain~~ understand this again

os the rod to be cut at ith index (meaning you are cutting rod (i) such that (n-i) is the one side length and other side length is i. Now also assume that (n-i) cut is equivalent to some piece length and assume that it has revenue like p(i). So here you can make relationship like

$$\boxed{\text{Roadcut}(n) = \max\left(\underline{p(i)} + \underbrace{\text{Rodcut}(i)}\right);}$$

This is fixed Entity

$$\begin{tikzcd}
 & \text{RoadCut}(n) \arrow[dl] \arrow[d] & & \\
P(1)+\text{Roadcut}(n-1) & P(2)+\text{Roadcut}(n-2) & \cdots & P(i)+\text{Roadcut}(n-\Sigma i)
\end{tikzcd}$$

$$\text{"} = (i)$$

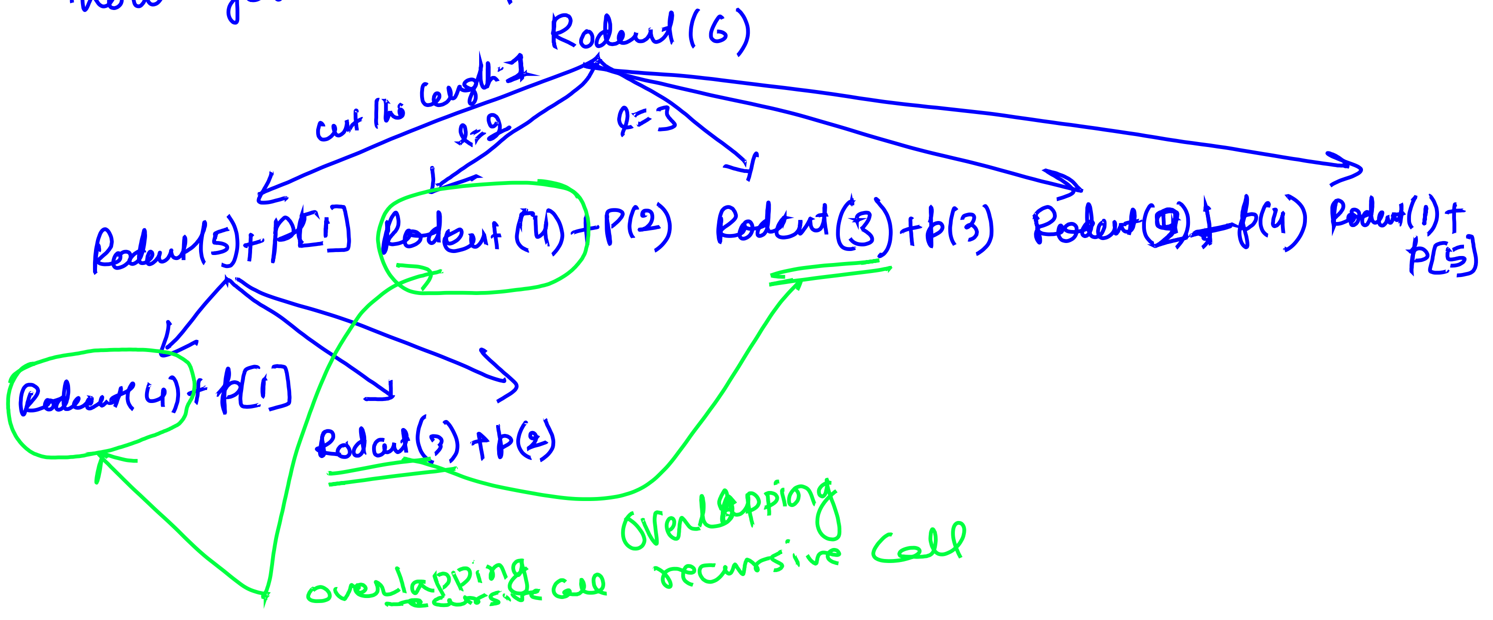
So, again this is optimization Problem and it may ~~[illegible]~~ be the candidate of DP (Memoization / Tabulation)

```
using namespace std;
//
int RodCut(int price [], int n){ //parameter as explained in main
{
    int max_revanue = INT_MIN;
    if(n<=0){
      return 0;
    }
    else{
      for(int i=1 ; i<=n ; i++){
        max_revanue = max(max_revanue, price(i)+ Rodcut(price,
                                                          (n-i)));
      }
    }
    return max_revanue;
}
//
int main(){
    int n=6;
    int price[n+1]={0,2,5,8,9,10,11};
                  //0 1 2 3 4 5  6(note)

    int answer = Rodcut(Price, n);   // Price is price table, n is rod length
    cout << "max revanue of length: << n << "for cutting =" << answer<<endl;

}
```

OK, so you done your recursive solution (~~brute force~~). What is wrong here? Offcourse this is not best solution, but how you will prove that this is not the best?



Let's write our code for memoization in recursion, to avoid the duplicate calculation which is currently happening in my code.

So i am just adding recursive method below for our understanding.

```
typedef vector<int> Memo;
 int rodCut( int price [] , int n, Memo &memo){
   if (memo[n] != -1){
     return memo[n];
   }
   if(n<=0) {
       return 0;        ] (A)
   }
   else{
(B)  -> for(int i=1 ; i<=n; ++i) {
       memo[n]= max( memo[n], p[i]+ rodCut(price, n-i));
     }
   return memo[n];
 }
```

Let's make our recursion solution into dp with tabulization approach.

```
int rodCut_dp(int price[] , int n) {
  Memo dp(n+1, -1);
   dp[0]=0; ] (A)
(B)-> for( i=1 ; i <=n ; i++){          <- rod length from 1 to n
     int best_price = INT_MIN;          <- cut possible on rod
(B)  for(int J=1 ;  j<=i ; j++){
       best_price = max (best_price, price[j]+ dp[i-j]);
     dp[i]= best_price;
  }
  return dp [n];
}
```