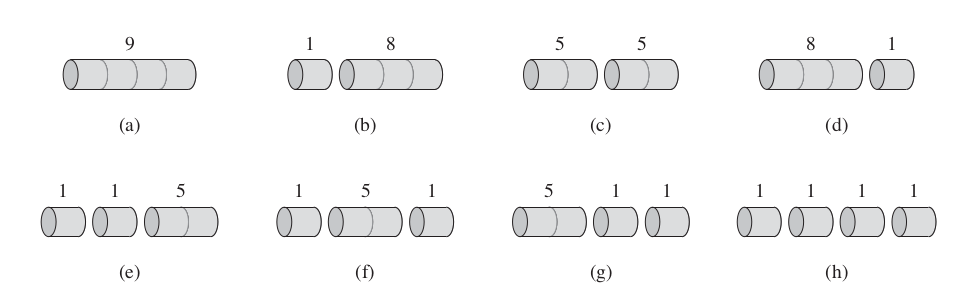
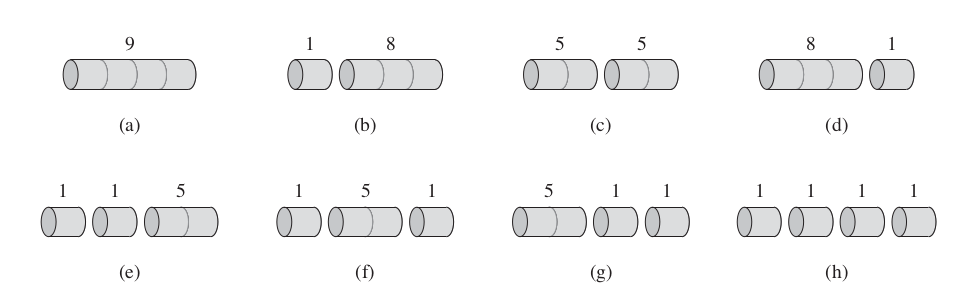
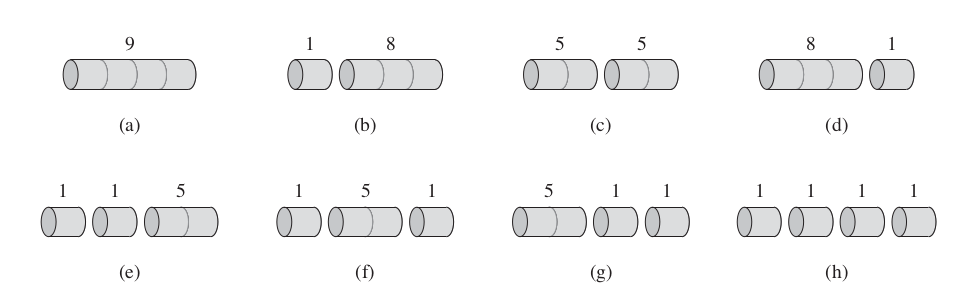
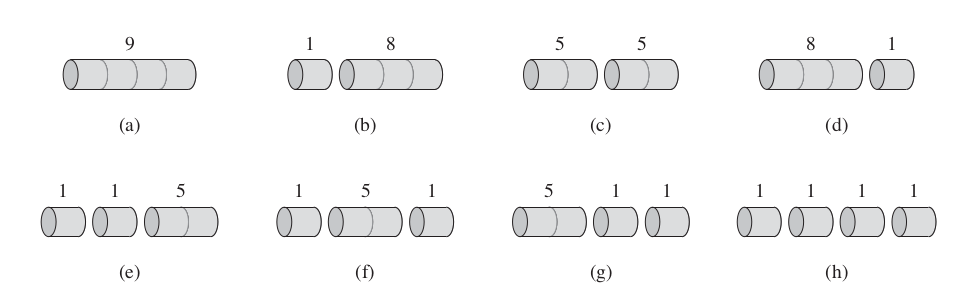
**Name: Mohamed Hazem**

**ID: 18107076**

**Rod Cutting**



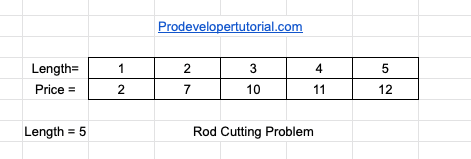
We have a rod length of 4



So now we need to get Minimum value for Maximum of rod

9+0+0+0 =9 8+1+0+0=9 5+1+1+1=8 5+5+0+0=10

Another example With DP Table



We have length of 5

1’s = 2, 2’s = 7, 3’s = 10, 4’s = 11, 5’s = 12

In our example, the length of the rod is 5.

If you give the whole rod, without cutting it, the profit is 12 rs.

But if you cut the rod at length 3 and length 2, the total profit will be 7 + 10 = 17. This gives us maximum profit.

We will use Brute Force Method

1 1 1 1 1 2 1 1 1 3 1 1 4 1 5

1 1 1 2 2 1 2 3 2

1 1 2 1 2 2 1

1 2 1 1 2 3

1 2 2

1 3 1

1 1 3

1 4

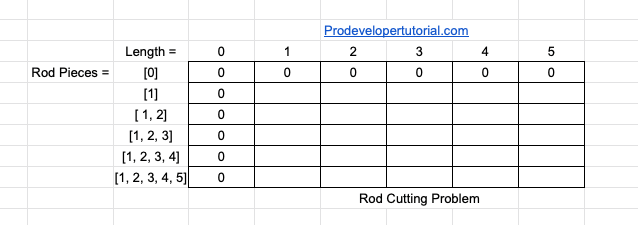
**From above we have total 16 different ways to cut the rod of length. Then we need to find the profit of all the combinations and find the max in them. Then we arrive at the result.**

Ways to cut is equal to W = (R.length-1)2

**Hence the time complexity at worst case will be O(2^n-1)**

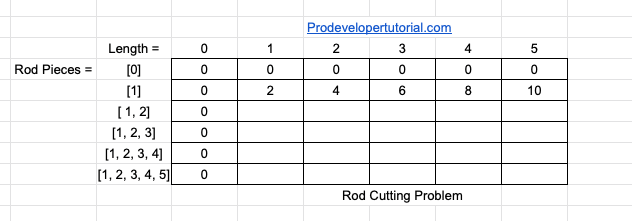
But We can reduce the time complexity to O(n^2) if we use DP.

When the rod length is 0, the profit will also be 0



It is important to start every DP matrix from ‘0’, so that you will be clear while solving the sub problem.

Going to the next row. “row 1”. We have rod piece of length 1. Hence for every varying length, we need to use combinations of rod piece 1 and calculate the profit.



For length 1 the profit will be 2.

For length 2 the profit will be 4.

For length 3 the profit will be 6.

For length 4 the profit will be 8.

For length 5 the profit will be 10.

Now for the next row, we have the rod piece combination of [1, 2]. Now we shall fill the array as below.

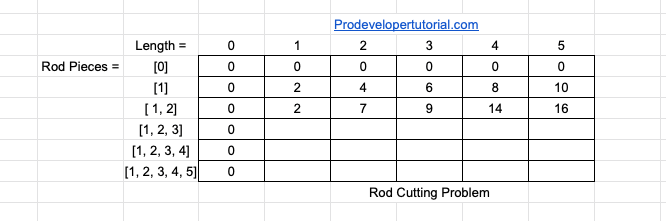
At length 1, you can use only 1 length. Hence profit is 2.

At length 2, you can have 2 choices. Either you use rod of length 1 twice or rod of length 2 once. As we need max profit, we calculate the profit in both of the cases.

1 piece + 1 piece = 2 + 2 = 4

Rod of length 2 = 7.

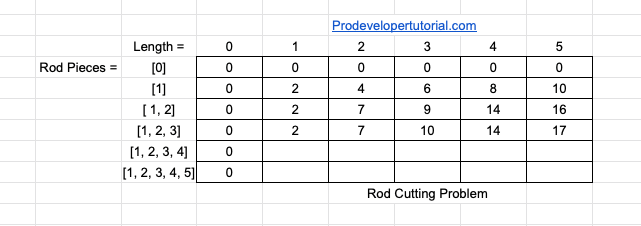
As we need maximum, we choose 7.

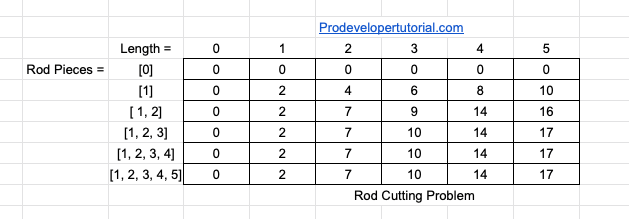


Similarly, for length 3, we can cut the rod for 1 length & 2 lengths. Hence profit is 2 + 7 = 9

Similarly, for length 4, we can cut the rod for 2 length & 2 lengths. Hence profit is 7 + 7 = 14

Similarly, for length 5, we can cut the rod for 2 length & 2 lengths & 1 length. Hence profit is 7 + 7 + 2 = 16





Another DP table for example of length 3

Values = 2 5 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Len | 0 | 1 | 2 | 3 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 |  |  |  |
| 1 2 | 0 |  |  |  |
| 1 2 3 | 0 |  |  |  |
| Len | 0 | 1 | 2 | 3 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 2 | 4 | 6 |
| 1 2 | 0 |  |  |  |
| 1 2 3 | 0 |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Len | 0 | 1 | 2 | 3 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 2 | 4 | 6 |
| 1 2 | 0 | 2 | 5 | 7 |
| 1 2 3 | 0 |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Len | 0 | 1 | 2 | 3 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 2 | 4 | 6 |
| 1 2 | 0 | 2 | 5 | 7 |
| 1 2 3 | 0 | 2 | 5 | 7 |

Max value can get is 7

## Algorithms

First Algorithm

Making a Cut at all positions

Comparing the values obtained after a cut

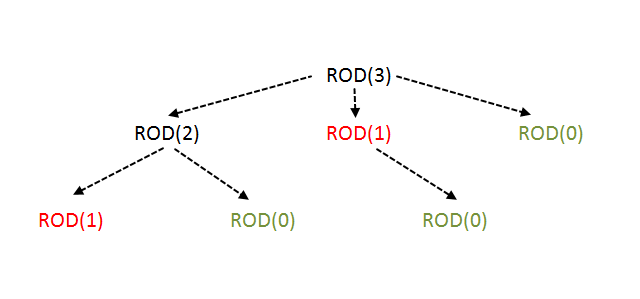
For all i in {0,1..n-1}

cutRod(n) = max(price[i]+cutRod(n-i-1))

this Recursive Solution has time Complexity of Exponential!

**Second Solution**

Overlapping subproblems



## Second Algorithm

For a given i

max\_val = Int\_main;

for(j=0;j<I;j++)

max\_val=max(max\_val,price[j]+val[i-j-1]);

val[i] = max\_val;

Implementation of the algorithm DP in Java With tracing comments & Tracing example at the end of the code

top-down recursive function Time complexity O(n^2)

Copy Start from here till

/\*

\*|\_\_\_\_\_\_Mohamed Hazem. .18107076\_\_\_\_\_\_|

#FIREMIDO#

\* and open the template in the editor.

\*/

package dynamiczz;

import java.util.\*;

public class Dynamiczz {

//top down recursive function Time complexity O(n^2);

/\* Returns the best obtainable price for a rod of length

n and price[] as prices of different pieces \*/

/\* Returns the best obtainable price for a rod of

length n and price[] as prices of different pieces \*/

static int cutRod(int price[],int n) //send Array and Size

{

int val[] = new int[n+1]; // create Temporary array which has same number of old array + 1

val[0] = 0; // make it's first element = 0 ( this the job of new array)

// Build the table val[] in bottom up manner and return

// the last entry from the table

for (int i = 1; i<=n; i++) // for loop on the arraies

{

int max\_val = Integer.MIN\_VALUE;

// Integer.MinValue stores the minimum possible value for any integer variable in Java.

for (int j = 0; j < i; j++)

max\_val = Math.max(max\_val, // Math.max returns the greater of two int values. in our case it's (max\_val and Price)

price[j] + val[i-j-1]);

//

val[i] = max\_val; // val[i] Storage new max\_val

}

return val[n]; // return array at element number N we send

}

/\* Driver program to test above functions \*/

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println("enter lenght of rod (N) ");

int N = sc.nextInt() ;

int arr[]= new int[N];

System.out.println("enter the (price) of it's elements in cumulative");

System.out.println("For example if rod = 3 then \nthe input be like [3 5 8] ");

int size = arr.length;

for(int i =0 ; i<size ; i ++)

{

arr[i]= sc.nextInt();

}

System.out.println("Maximum Obtainable Value is "+

cutRod(arr, size));

}

}

/\*

Exeplation

let say

N=3 [1,2,3]

| | |

cost = [3,5,8]

The Vel = [0,1,2,3] size[N+1]

Start of the process

price[j] + val[i-j-1]);

when i=1 (j=0 , price[0] + val[1-0-1=0] = 3+0 ) => max = 3 & vel[1] = 3

when i=2 (j=0 , price[0] + val[2-0-1=1] = 3+3 ) => max = 6 & vel[2] = 6

when i=2 (j=1 , price[1] + val[2-1-1=0] = 5+0 ) => max = 5 & vel[2] = 5

when i=3 (j=0 , price[0] + val[3-0-1=2] = 3+6 ) => max = 9 & vel[3] = 9

when i=3 (j=1 , price[1] + val[3-1-1=1] = 5+3 ) => ? 8>9 ! max = 9 & vel[3] = 9

when i=3 (j=2 , price[2] + val[3-2-1=0] = 8+0 ) => ? 8>9 ! max = 9 & vel[3] = 9

Maxmum value is 9

\*/

Here

To be easy to define Complexity Just look here

**for (i = 1; i<=n; i++) 🡺 N**

**{**

**int max\_val = INT\_MIN;**

**for (j = 0; j < i; j++)🡺 \*N**

So Total Complexity is O(N^2)

