Assignment 1

BHUKYA SIDDHU - EE18BTECH11004

Download all python and c codes from

https://github.com/Siddhu27/data_structures

and latex-tikz codes from

https://github.com/Siddhu27/data_structures

1 Problem

Define R_n to be the maximum amount earned by cutting a rod of length n meters into one or more pieces of integer length and selling them. For $i > 0$, let p[i] denote the selling price of a rod whose length is i meters. Consider the array of prices:

p[1]=1,p[2]=5,p[3]=8,p[4]=9,
p[5]=10,p[6]=17,p[7]=18.

Which of the following statements is/are correct about R_7 ?

A. $R_7 = 18$
B. $R_7 = 19$
C. R_7 is achieved by three different solutions.
D. R_7 cannot be achieved by a solution consisting of three pieces.

Solution :
Solutions for the given question are :
$R_7 = 18$
R_7 is achieved by three different solutions.

2 Explanation :

The idea to solve the problem is by dividing into smaller problems. Let the length of the rod is "i" meters then if the rod is cut into into two pieces at the position "j" ($j < i$) then the length of the two pieces will be j and i-j meters. Now by finding the maximum selling prices of the two pieces of rod will give the maximum selling price of the "i" length rod.To find maximum selling price of "j" meters length rod we need to find the maximum selling prices of the smaller pieces of the rod.So this can be written as recursive function given below :

$$\text{cutRod(n)} = \max(\text{price[i]} + \text{cutRod(n-i-1)}) \quad \text{for all i in } \{0, 1..n-1\}$$

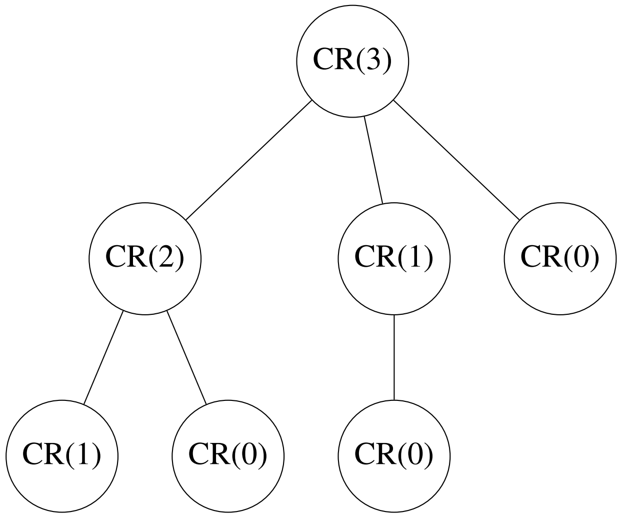
where cutRod(n) gives the required (best possible price) value for a rod of length n.

Following code gives the maximum selling price by selling the rod of n meters.

```
#include<stdio.h>
#include<limits.h>

// used to get the maximum of two integers
int max(int a, int b) { return (a > b)? a : b;}

/* Returns the best obtainable price for a rod of
     length n and
    price[] as prices of different pieces */
int cutRod(int price[], int n)
{
    if (n<=0)
    {
            return 0;
    }
    int max_val=INT_MIN;
    int i;
    for (i= 0; i < n; i++)
    {
            max_val = max(max_val, price[i] +
                cutRod(price,n-i-1));
    }

    return max_val;
}

int main()
{
     int arr[] = {1, 5, 8, 9, 10, 17, 18};
     int size = sizeof(arr)/sizeof(arr[0]);
     printf("Maximum Obtainable Value is %d",
          cutRod(arr, size));
    return 0;
}
```

From the above code we get $R_7 = 18$

The time complexity of the above code is $O(2^n)$ as n increases the time complexity also increases. So we have to choose other best solution.



The above tree diagram is for CutRod(3).Like this we can draw for cutRod(7).We can see that CR(5),CR(4),CR(3),CR(2),CR(1),CR(0) are being called again and again.Because of this the time comlexity of the code get increases. So now we store their values in an array and use them when they are needed.So this is a Dynamic Programming problem.The code for this approach is given below :

```
#include<stdio.h>
#include<limits.h>
//used to find the maximum vaule between two
    numbers
int max(int a, int b) { return (a > b)? a : b;}

/* Returns the best obtainable price for a rod of
    length n and
   price[] as prices of different pieces */
int cutRod(int price[], int n)
{
   int val[n+1];
   val[0] = 0;
   int i, j;

   // Build the table val[] in bottom up manner
       and return the last entry
   // from the table
   for (i = 1; i<=n; i++)
   {
       int max_val = INT_MIN;
       for (j = 0; j < i; j++)
         max_val = max(max_val, price[j] + val
             [i-j-1]);
       val[i] = max_val;
   }

   return val[n];
}

int main()
{
    int arr[] = {1, 5, 8, 9, 10, 17, 18};
    int size = sizeof(arr)/sizeof(arr[0]);
    printf("Maximum Obtainable Value is %d",
        cutRod(arr, size));
    return 0;
}
```

Now the function values which are called repeatedly are stored in an array and used when they are needed.Time complexity of this code is $O(n^2)$.

Therefore, we get $R_7 = 18$ and it can be achieved by solution consisting of three pieces
$p[2] + p[2] + p[3] = 5 + 5 + 8 = 18$