Dynamic Programming | Set 13 (Cutting a Rod)

Given a rod of length n inches and an array of prices that contains 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 length of the rod is 8 and the values of different pieces are given as 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
```

The naive solution for this problem is to generate all configurations of different pieces and find the highest priced configuration. This solution is exponential in term of time complexity. Let us see how this problem possesses both important properties of a Dynamic Programming (DP) Problem and can efficiently 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.

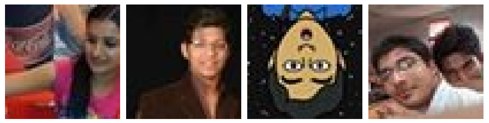
Let cutRoad(n) be the required (best possible price) value for a rod of lenght n. cutRod(n) can be

Google™ Custom Search

GeeksforGeeks

53,524 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms



written as following.

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

2) Overlapping Subproblems
Following is simple recursive implementation of the Rod Cutting problem. The implementation simply follows the recursive structure mentioned above.

```
// A Naive recursive solution for Rod cutting problem
#include<stdio.h>
#include<limits.h>

// A utility function 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;

   // Recursively cut the rod in different pieces and compare differen
   // configurations
   for (int i = 0; i<n; i++)
         max_val = max(max_val, price[i] + cutRod(price, n-i-1));

   return max_val;
}

/* Driver program to test above functions */
int main()
{
    int arr[] = {1, 5, 8, 9, 10, 17, 17, 20};
    int size = sizeof(arr)/sizeof(arr[0]);
    printf("Maximum Obtainable Value is %d\n", cutRod(arr, size));
    getchar();
    return 0;
}
```

Output:

```
Maximum Obtainable Value is 22
```

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

Considering the above implementation, following is 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)
```

In the above partial recursion tree, cR(2) is being solved twice. We can see that there are many subproblems which are solved again and again. Since same suproblems are called again, this problem has Overlapping Subprolems property. So the Rod Cutting problem has both properties (see this and this) of a dynamic programming problem. Like other typical Dynamic Programming(DP) problems, recomputations of same subproblems can be avoided by constructing a temporary array val[] in bottom up manner.

```
// A Dynamic Programming solution for Rod cutting problem
#include<stdio.h>
#include<limits.h>

// A utility function 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)
{
   int val[n+1];
   val[0] = 0;
   int i, j;

   // Build the table val[] in bottom up manner and return the last en
   // 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];
}

/* Driver program to test above functions */
int main()
{
    int arr[] = {1, 5, 8, 9, 10, 17, 17, 20};
    int size = sizeof(arr)/sizeof(arr[0]);
    printf("Maximum Obtainable Value is %d\n", cutRod(arr, size));
    getchar();
    return 0;
}
```

Output:

```
Maximum Obtainable Value is 22
```

Time Complexity of the above implementation is O(n^2) which is much better than the worst case time complexity of Naive Recursive implementation.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

705 Subscribe

Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 38 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 42 minutes ago

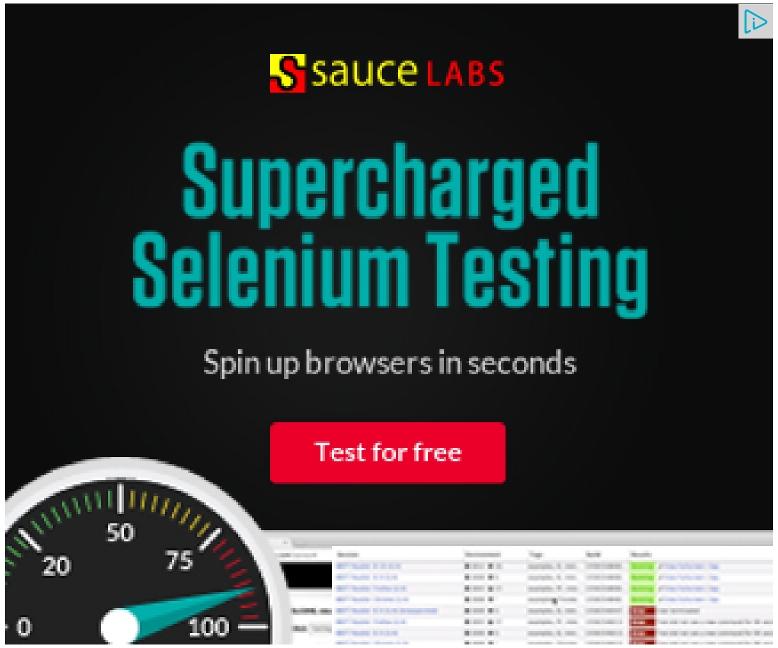
Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative



Related Tpoics:

- Backtracking | Set 8 (Solving Cryptarithmetic Puzzles)
- Tail Recursion
- Find if two rectangles overlap
- Analysis of Algorithm | Set 4 (Solving Recurrences)
- Print all possible paths from top left to bottom right of a mXn matrix
- Generate all unique partitions of an integer
- Russian Peasant Multiplication
- Closest Pair of Points | O(nlogn) Implementation

13 Tweet 0 1

Writing code in comment? Please use **ideone.com** and share the link here.

27 Comments **GeeksforGeeks**

Sort by Newest ▾

numbers also. We...

Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago

AdChoices

▶ C++ Code
▶ Java Source Code
▶ Programming C++

AdChoices

▶ Log Cutting
▶ Cutting Wood
▶ Java Algorithm

AdChoices

▶ Knapsack
▶ C Algorithm
▶ Log Me Int

Amit Kumar • a month ago

Good example. But I think something wrong with the output

length | 1 2 3 4 5 6 7 8

----

price | 1 5 8 9 10 17 17 20

For length 4, the price given by program is 10, which should be 9.

alext ➦ Amit Kumar • 6 days ago

why is that? cut length-4 rod into 2 pieces of length-2 rod, which is 10

prashant • 4 months ago

/*

int fun(int p[],int pl[],int n)

{

int k,max=0;

if(n==1)

return pl[0];

if(n==0)

return 0;

if(p[n]!=-1)

```
return p[n];

for(int i=1;i<=n;i++)
```

see more

arhtu • 5 months ago

```
public static int cutRod(int[] Price, int n, int[] OP)
{

int max_value = Price[n - 1];
//base
if (n == 1)
{
OP[0] = Price[0];
return Price[0];
}
//Just go thro half the length, eg for length=7 as(3+4) is the same as (4+3)
int toLength = n / 2;
for (int i = 0; i < toLength; i++){
if (OP[i] == 0) cutRod(Price, ((n - 1) - i), OP);
max_value = Math.Max(max_value, OP[i] + OP[(n - 2) - i]);//more efficient as y
sbuoptimal soulutions as we store it
}
OP[n - 1] = max_value;//store the suboptimal solutions
return max_value;
}
```

arhtu • 5 months ago

```
public static int cutRod(int[] Price, int n, int[] OP)

        {



int max_value = Price[n - 1];



if (n == 1)

            {

                OP[0] = Price[0];
```

see more

Sourin Sutradhar · 6 months ago

#include<iostream>
using namespace std;
int cutRod(int *arr, int size)
{
int max = 0;
if(size == 0)
return 0;
for(int i = 0; i < size; i++)
max = (arr[i]+cutRod(arr, size-i-1)>max)? arr[i]+cutRod(arr, size-i-1):max;
return max;

```
}
int main()
{
int arr[] = {1, 5, 8, 9, 10, 17, 17, 20};
int size = sizeof(arr)/sizeof(arr[0]);
cout<<"\nMaximum Obtainable Value is " << cutRod(arr, size)<<endl; return=
```

Vijay Apurva • 9 months ago

Shiwakant Bharti i have tested it . if we repeat inner loop till i then we swap eve
making inner loop till i/2+1

Shiwakant Bharti • 9 months ago

@Vijay: This looks correct to me. Not tested it though.

Anas Mourad • 9 months ago

Knapsack

Vijay Apurva • 9 months ago

we can do better.
we can reduce 2nd loop up to i/2 + 1.

```
#include<stdio.h>
int max(int a, int b).
{return (a > b)? a : b;}.

int cutRod(int arr[], int n){.
int temp[n], i,j;.
for(i=0;i<n;i++).
```

for(i=1;i<n;i++)
for(j=0;j<i/2+1;j++).
if(temp[i]<temp[j]+temp[i-j-1]).
temp[i]=temp[j]+temp[i-j-1];.

return temp[n-1];.
}

see more

Chaitanya • 10 months ago
In the given recurrence relation you have excluded the role of 'l' (lengths array)

There are two cases possible
(1) ith length cut can be done
(2) ith lenght cut can not be done

curRod(N, n) = Max {
cutRod(N-L[i], n-i-1) + P[i], --- (1)
cutRod(N, n-i-1) ---------------- (2)
}; 0 <= i < n

[sourcecode language="C"]
//p is price array, l is lengths array
int cutRod(int *p, int *l, int N, int n)
{
printf("R-%d--%d\n", N, n);
if(N <= 0)
return 0;

see more

Tushar Singhal • 10 months ago

in the above case why not max_val = max(max_val, price[j] + val[i-j-1]); be rep
val[j]+val[j-i]);.

max_val = max(max_val,

Sadiqin Ibnu Adam • 11 months ago

In UVa/ICPC problem, what code/id for this problem?

ganglu • a year ago

u can see a video explanation of the algorithm on the link below

http://www.youtube.com/watch?v...

2

kartheek • a year ago

/* 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;

// Recursively cut the rod in different pieces and compare different
// configurations
for (int i = 0; i<n; i++)
max_val = max(max_val, price[i] + cutRod(price+i, n-i-1));

return max_val;
}

kartheek • a year ago

```
 /* 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;

   // Recursively cut the rod in different pieces and compare differe
   // configurations
   for (int i = 0; i<n; i++)
         max_val = max(max_val, price[i] + cutRod(price+i, n-i-1));

   return max_val;
}
```

THe recursive function should work on the new array (price + i) which is a par
I'm wrong

gautamvs • 2 years ago

This is solution I have written:

```
 /* Paste your code here (You may delete these lines if not writing c
        public static int divideRod(int[] values, int[] answers, int 
                if(n==0) return 0;
                if(answers[n]!=-1) return answers[n];
                int max=-1, ans;
                for(int i=1; i<=n; i++){
                        ans=values[i]+divideRod(values, answers, n-i)
                        if(ans>max)
                                max=ans;
                }
                answers[n]=max;
                return max;
        }
```

see more

NIPUN • 2 years ago

```
 #include<stdio.h>
int cutRod(int price[],int n)
{
int max_val= max((n+1)*price[0],price[n]);
int i,max_sofar=0;

for(i=0;i<n;i++)
{
 max_sofar = price[i]+price[n-i-1];
    if(max_sofar >max_val)
        max_val = max_sofar ;
}
return max_val;
}
int max(int a, int b) { return (a > b)? a : b;}
int main()
{
    int arr[] = {2, 5, 8, 9, 10, 17, 17, 20};
    int size = sizeof(arr)/sizeof(arr[0]);
    printf("Maximum Obtainable Value is %d\n", cutRod(arr, size-1));
    getchar();
    return 0;
}
```

Mohit • 2 years ago

I think can be improved little bit.. below is the code of changed for loop...

```
 for (i = 1; i<=n; i++)
   {
       //int max_val = price[i];
       val[i] = price[i-1];
       for (j = 0; j < i/2; j++)
         val[i]= max(val[i], val[j] + val[i-j]);
       //val[i] = max_val;
   }
```

shiwakant.bharti ➦ Mohit • 9 months ago

Why do mohit(s) rock so much? Your code works awesomely by just a below.

```
           // Solves the max rod price for size = i
```

```
        // for actual item price we will need to use price|
        solutionDP[i] = price[i - 1];

        // Iterate over required pre-calculated solution fr
        for (j = 0; j <= i / 2; j++) {
            solutionDP[i] = Math.max(solutionDP[i], solutic
                    + solutionDP[i - j]);
        }
    }
    return solutionDP[n];
```

atul • 2 years ago

isn't it is similar to the 0-1 knapsack problem where ...
here size of knapsack is length of the rod and number of object available is 1 t
prices of each size wt.

```
/* Paste your code here (You may delete these lines if not writing co
```

nis • 2 years ago

Directly maps to knapsack.. here weight is length of Rod... and objects are sm
is weight of each obj and each piece price is each obj value.

Shahid • 2 years ago

Hi Kapil, I think in my solution either we are taking it full or we are not taking it s
knapsack problem. Pleas correct me if I am wrong .. thanks for your commen

shiv ➦ Shahid • 2 years ago

this will not work,here you are applying greedy approach, it will not wor
fails for 0-1 knapsack .

```
/* Paste your code here (You may delete these lines if not wr:
```

shahid Siddique • 2 years ago

length | 1 2 3 4 5 6 7 8
----
price | 1 5 8 9 10 17 17 20

DO it like this
1/1 = 1, 5/2 = 2.5, 8/3 = 2.66, 9/4 = 2.25, 10/5 = 2, 17/6 = 2.83, 17/7 = 2.42, 2C

Now highest is 2.83 we will take 2.83 which is piece 6 we left with 2 more piec
piece 2 is now more. so we will 5 which is piece 2.

I think I am right.. Thanks

kapil ➦ shahid Siddique • 2 years ago

That will not work always. The reason is same as why greedy algo doe

rahul • 2 years ago

great! a good example for beginners to understand DP.