Dynamic Programming | Set 9 (Binomial Coefficient)

Following are common definition of Binomial Coefficients.
1) A binomial coefficient C(n, k) can be defined as the coefficient of X^k in the expansion of (1 + X)^n.

2) A binomial coefficient C(n, k) also gives the number of ways, disregarding order, that k objects can be chosen from among n objects; more formally, the number of k-element subsets (or k-combinations) of an n-element set.

The Problem
Write a function that takes two parameters n and k and returns the value of Binomial Coefficient C(n, k). For example, your function should return 6 for n = 4 and k = 2, and it should return 10 for n = 5 and k = 2.

1) Optimal Substructure
The value of C(n, k) can recursively calculated using following standard formula for Binomial Cofficients.

```
  C(n, k) = C(n-1, k-1) + C(n-1, k)
  C(n, 0) = C(n, n) = 1
```

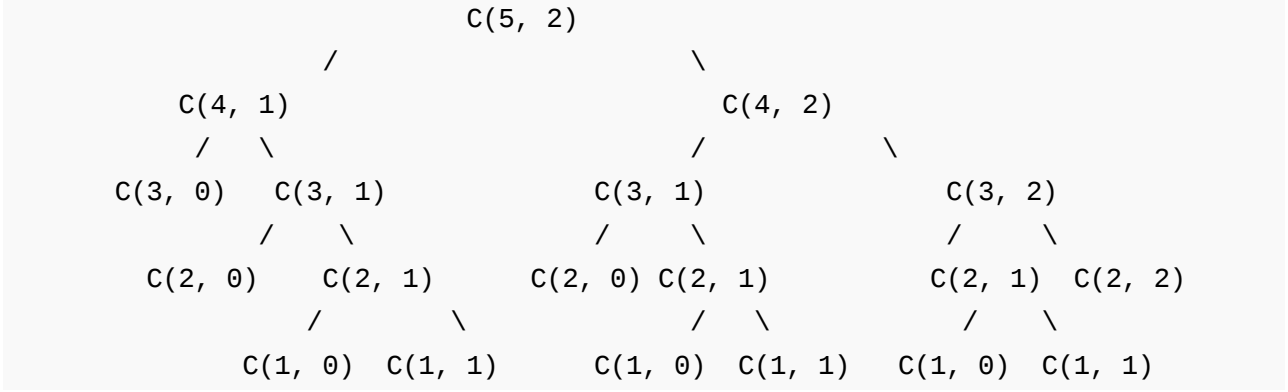
2) Overlapping Subproblems
Following is simple recursive implementation that simply follows the recursive structure mentioned above.

```
// A Naive Recursive Implementation
#include<stdio.h>

// Returns value of Binomial Coefficient C(n, k)
```


```
int binomialCoeff(int n, int k)
{
  // Base Cases
  if (k==0 || k==n)
    return 1;

  // Recur
  return  binomialCoeff(n-1, k-1) + binomialCoeff(n-1, k);
}
```

```
/* Drier program to test above function*/
int main()
{
    int n = 5, k = 2;
    printf("Value of C(%d, %d) is %d ", n, k, binomialCoeff(n, k));
    return 0;
}
```

It should be noted that the above function computes the same subproblems again and again. See the following recursion tree for n = 5 an k = 2. The function C(3, 1) is called two times. For large values of n, there will be many common subproblems.



Since same suproblems are called again, this problem has Overlapping Subprolems property. So the Binomial Coefficient 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 C[][] in bottom up manner. Following is Dynamic Programming based implementation.

```
// A Dynamic Programming based solution that uses table C[][] to calcu
// Binomial Coefficient
#include<stdio.h>
```

Geometric Algorithms

Popular Posts

```
// Prototype of a utility function that returns minimum of two integer
int min(int a, int b);

// Returns value of Binomial Coefficient C(n, k)
int binomialCoeff(int n, int k)
{
    int C[n+1][k+1];
    int i, j;

    // Caculate value of Binomial Coefficient in bottom up manner
    for (i = 0; i <= n; i++)
    {
        for (j = 0; j <= min(i, k); j++)
        {
            // Base Cases
            if (j == 0 || j == i)
                C[i][j] = 1;

            // Calculate value using previosly stored values
            else
                C[i][j] = C[i-1][j-1] + C[i-1][j];
        }
    }

    return C[n][k];
}

// A utility function to return minimum of two integers
int min(int a, int b)
{
    return (a<b)? a: b;
}

/* Drier program to test above function*/
int main()
{
    int n = 5, k = 2;
    printf ("Value of C(%d, %d) is %d ", n, k, binomialCoeff(n, k) );
    return 0;
}
```

Time Complexity: O(n*k)
Auxiliary Space: O(n*k)

Following is a space optimized version of the above code. The following code only uses O(k).

Thanks to AK for suggesting this method.

```
// A space optimized Dynamic Programming Solution
int binomialCoeff(int n, int k)
{
    int* C = (int*)calloc(k+1, sizeof(int));
    int i, j, res;

    C[0] = 1;

    for(i = 1; i <= n; i++)
    {
        for(j = min(i, k); j > 0; j--)
            C[j] = C[j] + C[j-1];
    }

    res = C[k];  // Store the result before freeing memory

    free(C);  // free dynamically allocated memory to avoid memory lea

    return res;
}
```

Time Complexity: O(n*k)
Auxiliary Space: O(k)

References:
http://www.csl.mtu.edu/cs4321/www/Lectures/Lecture%2015%20-%20Dynamic%20Programming%20Binomial%20Coefficients.htm

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

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 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

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

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

30 Comments GeeksforGeeks

Sort by Newest

Join the discussion

Abhinav Aggarwal • 5 months ago

O(k): Time Complexity
O(1): Space Complexity

nC1=n

and for any value

(n) C (r+1) = ((n) C (r))*(n-r)/(r+1)

Code: http://ideone.com/PLOKVD

Correct me if I am wrong.

Code_Addict • 5 months ago

Java version for naive recursive approach and DP (Bottom Up) :

http://ideone.com/ObA8PG

anonymous • 5 months ago

Why is the min(i,k) taken? I cant understand that part!
Please help.

Zain anonymous • a month ago

min(i,k) returns the value which is minimum i.e. if i<k then="" return=""

Ali Zain • a month ago

```
if(A<b) return="" a;="" else="" return="" b:="" <="" code=
```

Zain Zain • a month ago

IF (A<b) return="" a;="" else="" return="" b;="">

Rish • 9 months ago

I used this Identities involving binomial coefficients

c(n,k) = n/k * c(n-1,k-1)

```
#include <iostream>
using namespace std;

int c(int n, int k)
{
    if(k == 0)return 1;
    if(n <= k) return 0;

    return (n*c(n-1,k-1))/k;
}
int main()
{
    cout<<c(5,2); }="" <="" code="">
```

5

anshul35 • 11 months ago

I have tried to solve this in O(r) time. It is working fine for small no but giving ne

Please sm1 point out my mistake.

```
 #include<iostream>
using namespace::std;

long long int nCr(int n, int r)
{
        long long double res = 1;
        for(int i=1; i<=r; i++)
        {
                res *= float(n-r)/float(i) +1;
    }
    return res;
}

int main()
{
```

see more

anshul35 • 11 months ago

This code gives me negative results for even slightly big no like 80C60.
Why don't we use luca's theorem instead?

Jagat • a year ago

The equation
C(n, k) = C(n-1, k-1) + C(n-1, k)
has a nice intuitive interpretation.

To pick k elements from n elements [C(n, k)], you consider one element and e
elements, or you don't. If you do, you have to now choose k-1 elements from t

1)]; if you don't you need to choose k elements from the remaining n-1 elemen
QED.

ibn ➚ Jagat • 6 days ago

Thanks for pointing out the intutive.
Example:
Let choose 2 from the following set of 5 items { A, B, C, D, E}
If we choose the 1st item A, then the set contains just 4 items

{ B, C, D, E } and we need to choose 1 from this set.
If we don't choose A, we still take A out of the set. Thus, the remaining
we still need to choose 2 from this set. Either we decide to choose an
and smaller to the base case.

aman gupta • 2 years ago

@geeksforgeeks,
for n<k;
we should give answer as 0 instead of garbage value,
because number of ways to choose k items from n items for n<k is 0 only.
Your program is missing that case.
Same is for n=0 and k=non zero.

[sourcecode language="C"]
/* Paste your code here (You may delete these lines if not writing code) */

aman gupta ➚ aman gupta • 2 years ago

and we should use memoization approach here as it will take O(n+k) ti
please comment if i m wrong..

Ricky13 • 2 years ago

For the first DP approach Auxiliary Space should be O(n*k) instead of O(n^k).

GeeksforGeeks ➦ Ricky13 • 2 years ago

Thanks for pointing this out. There was a typo. We have corrected it nc

Sundar • 2 years ago

If you consider mathematically n(c, k) = (n*(n-1)*..*(n-(k-1)))/(k*(k-1)*...*1)

Here is the code

```
int  mathematicalWay(int n, int k) {
       int val = 1;
       int div = 1;
       int i;
       for (i = 1; i <= k; i++) {
              val = val * (n-(i-1));
              div = div * i;
       }
       return (val/div);
}
```

2

zyfo2 ➦ Sundar • a year ago

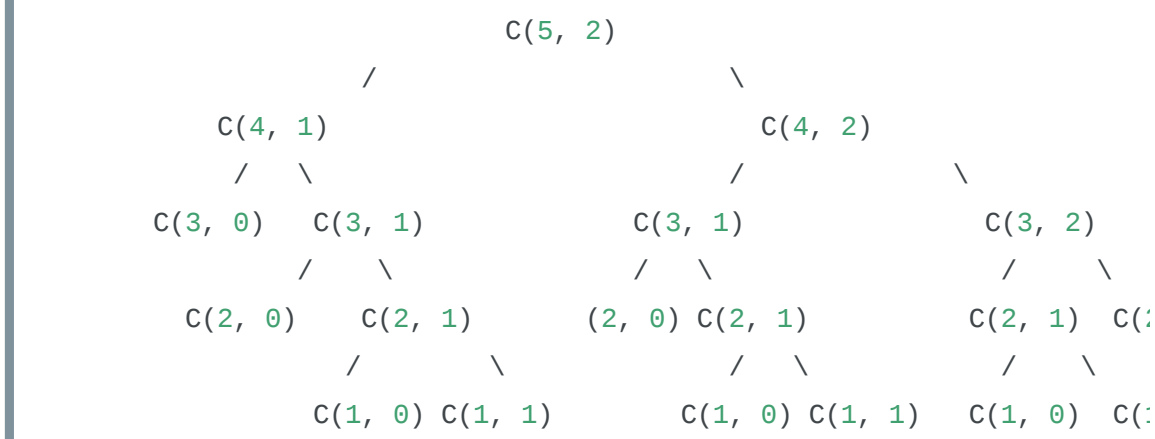
yeah, time O(k) and space O(1). definitely much better than DP

nirbhay • 2 years ago

```
#include
#include
void find(int ,int, float);
int main()
{
int a,b;
scanf("%d %d",&a,&b);
find(a,b,1);
}
void find(int a,int b,float sum)
{

if(b==1)
{
printf("%f",sum*a);
exit(0);
}
else
{
sum=sum*((float)a/b);
find(a-1,b-1,sum);
}
}
~
```

Sandeep Vasani • 2 years ago

```
Above recursive Tree for example C(5,2) is wrong it should be,
```



GeeksforGeeks ➦ Sandeep Vasani • 2 years ago

@Sandeep Vasani: Thanks for pointing this out. We have updated the

AK • 2 years ago

If you just want to find C[n][k], here is a simple O(n*k) time and O(n) space me

```
 int binomialCoeff(int n, int k)
{
    int* C = (int*)calloc(n+1,sizeof(int));
    int i, j;

    C[0] = 1;
    for(i = 1; i <= n; i++)
    {
        for(j = i; j > 0; j--)
            C[j] += C[j-1];
    }
    return C[k];
```

```
}
```

1

Nikhil Kumar ➚ AK • 4 months ago

@ak could u please explain how n why the above code works ?
m not able to get the idea behind the above algorithm....
i dont want to memorise this.. :/

1

kartik ➚ AK • 2 years ago

@AK: Thanks for suggesting a space optimized method. The time con
though. I think, the inner loop initialization statement can be modified to

```
 int binomialCoeff(int n, int k)
{
    int* C = (int*)calloc(n+1,sizeof(int));
    int i, j;

    C[0] = 1;
    for(i = 1; i <= n; i++)
    {
        for(j = k; j > 0; j--)
            C[j] += C[j-1];
    }
    return C[k];
}
```

Even after the loop initialization changes, this method seems be doing
given in the post doesn (k-1)k/2 + k(n-k) operations. Please correct me

shankar ➚ kartik • 2 years ago

@AK , Karthik Can You Explain this little bit more

C(n, k) = C(n-1, k-1) + C(n-1, k) ??

this recursion , please explain its meaning ?

```
/* Paste your code here (You may delete these lines if
```

GeeksforGeeks ➚ shankar • 2 years ago

@shankar: This follows the standard Binomial Coefficie

AK ➚ kartik • 2 years ago

Thats a very minor speed-up and depends on input k. You can

kartik ➚ AK • 2 years ago

Starting from min(i,k) makes sense. So the final code w

```
 int binomialCoeff(int n, int k)
{
    // Only O(k) space needed
    int* C = (int*)calloc(k+1,sizeof(int));
    int i, j;

    C[0] = 1;

    for(i = 1; i <= n; i++)
        for(j = min(i, k); j > 0; j--)
```

```
            C[j] += C[j-1];

    return C[k];
}
```

This DP method looks great. It uses O(k) space and sa
given in the post. We will add it to the original post. Thar

sachin kartik • 2 years ago

Why the inner loop is run backwards to 0 and not from (

```
/* Paste your code here (You may delete these li
```

GeeksforGeeks kartik • 2 years ago

@Frederic: Thanks for pointing this out. We have updat
leak.

Frederic kartik • 2 years ago

If you use calloc, you should call free(C) before returnin

Subscribe Add Disqus to your site