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.

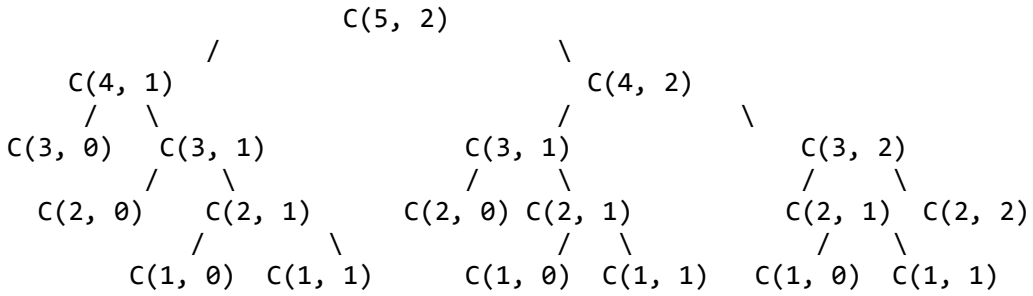
```
// A Naive Recursive Implementation
#include<stdio.h>

// 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 calculate th
```

```
// Binomial Coefficient
#include<stdio.h>

// Prototype of a utility function that returns minimum of two integers
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 leak

    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.

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Dynamic Programming

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

Join the discussion…

Akassh A Mishra • 25 days ago

Run & Space : O(n)
http://ideone.com/ZTE4qG

• Reply • Share ›

Guest • 25 days ago

Run : O(n), Space : O(n)
public static int nCk(int n, int k) {
List<integer> cList = new ArrayList<integer>();
cList.add(1);
for(int i=1; i<n 2+1="" &&="" i<="k;" ++i)="" clist.add((int)(clist.get(i-1)*((long)(n-i+1))="" i));="
for(int=" i="n/2+1;" i<="n" &&="" i<="k;" ++i)="" clist.add(clist.get(n-i))="" ;="" return="
clist.get(k);="" }="">

• Reply • Share ›

sk • a month ago

O(1) space and O(n-k) time complexity .
http://ideone.com/tI5tew

• Reply • Share ›

lucy • a month ago

@GeeksforGeeks i think in last method array C should initialize with zero initialy

• Reply • Share ›

cfh ➚ lucy • 6 hours ago

when an array is allocated using calloc() it is automatically initialized with 0.

• Reply • Share ›

Tuhin Bhattacharya • 5 months ago

```
#include<stdio.h>
int bin(n,k){
int i;
int c[k+1];
c[0] = 1;
for(i=1;i<=k;i++)
c[i] = ( c[i-1] * (n-i+1) )/ i;
return c[k];
}
int main(){
int n = 5, k = 2;
```

```
printf("%d\n",bin(n,k));
return 0;
}
```

• Reply • Share ›

Anam Amer • 7 months ago

can some1 plz explain this to me plz

((i+j)==k) for i=1 j=2 k=3

ans ((i+j)==k)==>1 this is whats written in my book n i have no idea whta this is

if i do it like normal math then 1+2=3 isnt tis what its supposed to be like

1 • Reply • Share ›

Aladeen ➚ Anam Amer • 6 months ago

Bro, == is not assignment operator instead its an comparator;
litrally it means if(sum of i&j is equal to k) then next statement....

• Reply • Share ›

Anam Amer ➚ Aladeen • 5 months ago

Ooh so its like i+j = k which is greater than 1 is this what it means

• Reply • Share ›

pritika • 7 months ago

Can anyone please explain how C[j] = C[j] + C[j-1]; is working in space optimized solution?

1 • Reply • Share ›

Anurag Singh ➚ pritika • 7 months ago

Recall Pascal's Triangle (here)
Pascal's triangle is a triangular array of the binomial coefficients.

1==========>> n = 1, C(1,0) = 1
1--1========>> n = 2, C(1,0) = 1, C(1,1) = 1
1--2--1======>> n = 2, C(2,0) = 1, C(2,1) = 2, C(2,2) = 1
1--3--3--1====>>n = 3, C(3,0) = 1, C(3,1) = 3, C(3,2) = 3, C(3,3)=1
1--4--6--4--1==>>n = 4, C(4,0) = 1, C(4,1) = 4, C(4,2) = 6, C(4,3)=4, C(4,4)=1
So here every loop on i, builds ith row of pascal triangle, using (i-1)th row

• Reply • Share ›

pritika ➚ Anurag Singh • 7 months ago

but we are saying c[j] = c[j] +c[j-1]..how could we use c[j] on rhs when we have to find it

• Reply • Share ›

Anurag Singh ➚ pritika • 7 months ago

At any time, every element of array C will have some value (ZERO or more) and in next iteration, value for those elements comes from previous iteration.
In statement,
C[j] = C[j] + C[j-1]
RHS side represent the value coming from previous iteration (A row of Pascal's triangle depends on previous row).
LHS side represents the value of current iteration which will be obtained by this statement.

Lets say we want to calculate C(4,3), i.e. n=4, k=3:
All elements of array C of size 4 (k+1) are initialized to ZERO while allocation (calloc).
i.e. C[0] = C[1] = C[2] = C[3] = C[4] = 0;
Then C[0] is set to 1
For i=1:
C[1] = C[1] + C[0] = 0 + 1 = 1 ==>> C(1,1) = 1
For i=2:

see more

2 ^ | v • Reply • Share ›

Vãîbhåv Joshî • 10 months ago

Bottom up DP solution of java

http://ideone.com/oN15fZ

^ | v • Reply • Share ›

Abhinav Aggarwal • a year 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.

2 ^ | v • Reply • Share ›

Adarsh Tadimari ➚ Abhinav Aggarwal • 10 months ago

Hey, you just need to add a test case (k == n). nCn is 1, but n will be returned.

• Reply • Share ›

Code_Addict • a year ago

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

http://ideone.com/ObA8PG

• Reply • Share ›

anonymous • a year ago

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

• Reply • Share ›

Zain anonymous • a year ago

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

• Reply • Share ›

Ali Zain • a year ago

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

• Reply • Share ›

Zain Zain • a year ago

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

• Reply • Share ›

Rish • 2 years 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="">
```

7 • Reply • Share ›

mathGeek ➦ Rish • a month ago

Thanks for the tip.
For the 2nd, it is a scrit inegality. Here is the correction :

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

return (n*binomial_coefficient(n-1,k-1))/k;
}
```

and the test log :
C(0,0) = 1
C(1,0) = 1 C(1,1) = 1
C(2,0) = 1 C(2,1) = 2 C(2,2) = 1
C(3,0) = 1 C(3,1) = 3 C(3,2) = 3 C(3,3) = 1
C(4,0) = 1 C(4,1) = 4 C(4,2) = 6 C(4,3) = 4 C(4,4) = 1

• Reply • Share ›

anshul35 • 2 years ago

I have tried to solve this in O(r) time. It is working fine for small no but giving negative results for even 450C350.

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;
}
```

see more

• Reply • Share ›

anshul35 • 2 years ago

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

• Reply • Share ›

lucy ➚ anshul35 • a month ago

can u explane luca's theorem

• Reply • Share ›

Jagat • 2 years 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 either include it in the k chosen elements, or you don't. If you do, you have to now choose k-1 elements from the remaining n-1 elements[C(n-1, k-1)]; if you don't you need to choose k elements from the remaining n-1 elements [C(n-1, k)].
QED.

2 • Reply • Share ›

ibn ➚ Jagat • a year 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 set contains 4 items { B, C, D, E } and we still need to choose 2 from this set. Either we decide to choose an item or not, the set would get smaller and smaller to the base case.

• Reply • Share ›

aman gupta • 3 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 • 3 years ago

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

Ricky13 • 3 years ago

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

GeeksforGeeks Ricky13 • 3 years ago

Thanks for pointing this out. There was a typo. We have corrected it now. Keep it up!!

Sundar • 3 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);
}
```

3

zyfo2 Sundar • 2 years ago

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

nirbhay • 3 years ago

#include
#include
void find(int ,int, float);
int main()

```
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);
}
}
~
```

• Reply • Share ›

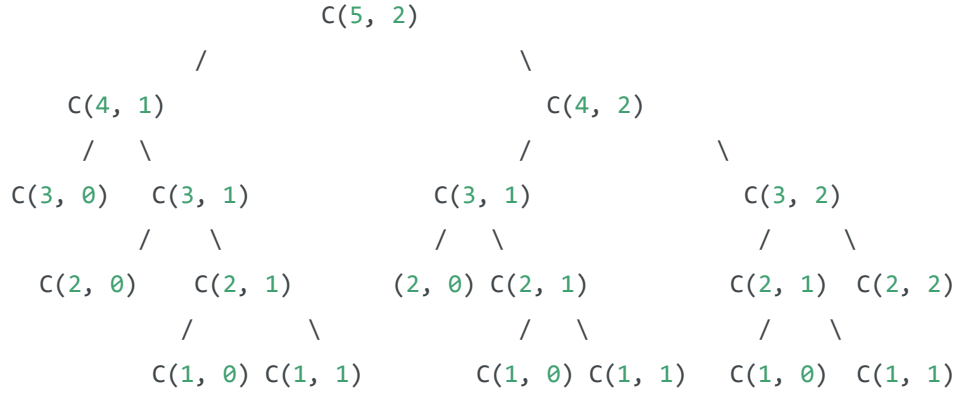
Aladeen ➚ nirbhay • 6 months ago

wats ur problem???

• Reply • Share ›

Sandeep Vasani • 3 years ago

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



• Reply • Share ›

GeeksforGeeks ➦ Sandeep Vasani • 3 years ago

@Sandeep Vasani: Thanks for pointing this out. We have updated the post with the correct recursion tree.

• Reply • Share ›

AK • 3 years ago

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

```
 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];
}
```

4 • Reply • Share ›

Nikhil Kumar ➦ AK • a year 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 • Reply • Share ›

Anurag Singh ➦ Nikhil Kumar • 9 months ago

It is using Pascal triangle approach
http://en.wikipedia.org/wiki/P...
But it is SC(n) and TC(n^2). Solution with same approach, posted by geeksforgeeks is optimized one with SC(k) and TC(n*k)

• Reply • Share ›

kartik ➦ AK • 3 years ago

@AK: Thanks for suggesting a space optimized method. The time complexity of this method is O(n^2) though. I think, the inner loop initialization statement can be modified to make it O(n*k).

```
 int binomialCoeff(int n, int k)
```

```
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 O(nk) operations. But, the method given in the post doesn (k-1)k/2 + k(n-k) operations. Please correct me if I am wrong.

• Reply • Share ›

shankar ➚ kartik • 3 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 not writing code) */
```

• Reply • Share ›

GeeksforGeeks ➚ shankar • 3 years ago

@shankar: This follows the standard Binomial Coefficient formula. Please see the wiki page.

• Reply • Share ›

AK ➚ kartik • 3 years ago

Thats a very minor speed-up and depends on input k. You can as well start j from min(i,k)

• Reply • Share ›

kartik ➚ AK • 3 years ago

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

```
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 same time complexity as the method given in the post. We will add it to the original post. Thanks for your time and effort. Keep it up!!

• Reply • Share ›

sachin kartik • 3 years ago

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

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

• Reply • Share ›

GeeksforGeeks kartik • 3 years ago

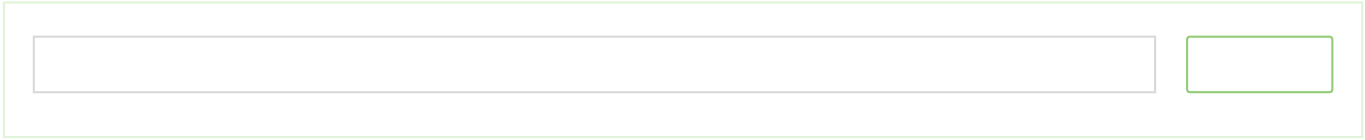
@Frederic: Thanks for pointing this out. We have updated the code to avoid memory leak.

• Reply • Share ›

Frederic kartik • 3 years ago

If you use calloc, you should call free(C) before returning; otherwise you have a leak.

• Reply • Share ›



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

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

- Follow @GeeksforGeeks

Recent Comments

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 1 hour ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 1 hour ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago