

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 Coefficients.

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

2) Overlapping Subproblems

Following is simple recursive implementation that simply follows the recursive structure mentioned above.

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

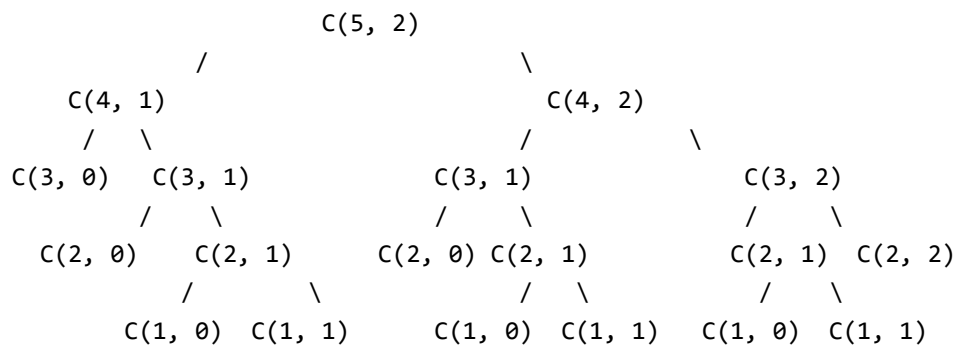
// Returns value of Binomial Coefficient C(n, k)
int binomialCoeff(int n, int k)
{
    // Base Cases
    if (k==0 || k==n)
        return 1;

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

/* Driver 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;
}
```

[Run on IDE](#)

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



Since same subproblems are called again, this problem has Overlapping Subproblems 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 the
// 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;

    // Calculate 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 previously 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;
}
```

[Run on IDE](#)

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;
}
```

[Run on IDE](#)

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.

55 Comments Category: Misc Tags: Dynamic Programming