Utilities For Programming by SUPERCOOL276

9th July 2016

Computing Binomial Coefficients : nCk

In mathematics, **binomial coefficients** are a family of positive integers [https://en.wikipedia.org/wiki/Integer] that occur as coefficients in the binomial theorem.

the binomial coefficient indexed by *n* and *k* is usually written $\binom{n}{k}$. It is the coefficient of the x^k term in the polynomial expansion of the binomial [https://en.wikipedia.org/wiki/Binomial_(polynomial)] power$(1 + x)^n$.

Implementing a computer program that computes value of binomial coefficient for a given value of **n** and **k** can be done using many different approaches .

Method 1 : FACTORIAL FORMULA

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} \quad \text{for } 0 \le k \le n,$$

Here is a C++ implementation of the idea.

```
#include<bits/stdc++.h>
using namespace std;

#define MOD 1000000007

typedef long long int lli;

lli nCk_factorial_formula(lli n,lli k){
 //Check Valid Input
 if(n < k){
  return -1;
```

```
  i--;
 }
 i = k;
 while(i>0){
  kf = kf * i;
  i--;
 }
 i = n-k;
 while(i>0){
  nkf = nkf * i;
  i--;
 }
 //Use formula
 lli answer = nf / (kf * nkf);
 return answer;
}
int main(){
 //Decalre variables n,k
 lli n,k;
 //input the value from user
 cin>>n>>k;
 //Factorial Formula
 cout<<nCk_factorial_formula(n,k);
 return 0;
}
```

Time Complexity : **O(n)**
Memory Complexity : **O(1)**

Remarks : This is not a suitable method for evaluating nCr due to **large values of n! ..** Moreover it might not give correct answer due to **overflow.**

Method 2 : RECURSIVE FORMULA

We use MOD = 1000000009 to avoid overflow.

Here is a C++ implementation of the idea.

```
#include<bits/stdc++.h>
using namespace std;

#define MOD 1000000007

typedef long long int lli;

lli nCk_recursive_formula(lli n,lli k){
 if(n<k){
  return 0;
 }
 else if(n==0||k==0||n==k){
  return 1;
 }
 else {
  //Use MOD to avoid Overflow
  return (nCk_recursive_formula(n-1,k-1)+nCk_recursive_formula(n-1,k))%MOD;
 }
}
int main(){
 //Decalre variables n,k
 lli n,k;
 //input the value from user
 cin>>n>>k;
 //Recursive Formula
 cout<<nCk_recursive_formula(n,k);
 return 0;
}
```

Method 3 : DYNAMIC PROGRAMMING

We can easily convert the recursive solution to both Top Down Dynamic Programming using Memoization to avoid repetitive computations.

We can also implement a bottom up approach.

We use MOD = 1000000009 to avoid overflow.

Here is a C++ implementation of the idea.

```
#include<bits/stdc++.h>
using namespace std;

#define MOD 1000000007

typedef long long int lli;
lli C[1001][1001];
void fill_matrix(int n){
 for(int i=0;i<=n;i++){
  C[i][0]=1;C[i][i]=1;
  for(int j=1;j<i;j++){
   C[i][j] = (C[i-1][j-1]+C[i-1][j])%MOD;
  }
 }
}
int main(){
 //Decalre variables n,k
 int n,k;
 //input the value from user
```

Utilities For Programming by SUPERCOOL276

search

Classic Flipcard Magazine Mosaic Sidebar Snapshot Timeslide

```
}
```

Time Complexity : **O(n^2)**
Memory Complexity : **O(n^2)** (We have taken array of const size but space requirement is **O(n^2)**)

Remarks : This is still not a suitable method for evaluating nCr due to **high time complexity and Memory Requirement.**

Method 4 : MULTIPLICATIVE FORMULA

$$\binom{n}{k} = \frac{n^{\underline{k}}}{k!} = \frac{n(n-1)(n-2)\cdots(n-(k-1))}{k(k-1)(k-2)\cdots 1} = \prod_{i=1}^{k} \frac{n-(k-i)}{i} = \prod_{i=1}^{k} \frac{n+1-i}{i},$$

This is one of the most efficient approach.

Here's the implementation in C++.

```
#include<bits/stdc++.h>
using namespace std;

#define MOD 1000000007

typedef long long int lli;
lli binomialCoeff(lli n, lli k)
{
    lli res = 1;

```

```
17     {
18         res *= (n - i);
19         res /= (i + 1);
20     }
21
22     return res;
23 }
24 int main(){
25  //Decalre variables n,k
26  int n,k;
27  //input the value from user
28  cin>>n>>k;
29  //Multiplicative Formula
30  cout<<binomialCoeff(n,k);
31  return 0;
32 }
```

Time Complexity : **O(k)**
Memory Complexity : **O(1)**

Remarks : This is a very efficient method for evaluating nCk.

Note : Since we have not taken care of **OVERFLOW** , this method would not show correct values beyond 60 as C(60,30) is nearly equal to 10^{18} which is highest value that can be stored in a 64 bit Variable (long long int in C++) .

To **avoid the overflow** , we can use MOD in line 18 but in line 19 since we are dividing , therefore we'll have to use Fermat's Little Theorem for taking MOD

```
(a / b) % p = ((a % p) * (b^(-1) % p)) % p

 For p = prime , b^(-1) % p = b^(p - 2) %  p .
```

This method is strictly for evaluating **C(n,k) % M**
Where n,k and M are given and **M is prime.**

We will use following results :-

factorial(n) = n * factorial(n-1) % M

invfactorial(n) = modular_inverse(factorial(n)) = (factorial(n) ^ (M-2)) % M

invfactorial(n-1) = (n)*(invfactorial(n)) % M

Derivation can be done using following :-
Modular Multiplicative Inverse for a prime M is in fact very simple. From Fermat's Little Theorem:

A^(M-1) % M = 1
Hence, A * A^(M-2) % M = 1
A^(-1) % M = A^(M-2) % M

For more information:-

Modular Multiplicative Inverse - Wikipedia [https://en.wikipedia.org/wiki/Modular_multiplicative_inverse]

Here is a C++ based implementation.

```
#include<bits/stdc++.h>
using namespace std;

#define MOD 1000000007

typedef long long int lli;
lli fact[100000 + 1], invfact[100000 + 1];
//Modular Exponentiation
```

```
   if (b & 1)
    result = (result * a) % MOD;
   b >>= 1;
   a = (a * a) % MOD;
  }
  return result;
 }
lli ncr(lli n, lli k)
{
 return (((fact[n] * invfact[k]) % MOD) * invfact[n - k]) % MOD;
}
void initialize(void)
{
 int i;
 fact[0] = 1;
 for (i = 1; i <= 100000; i++)
  fact[i] = (fact[i - 1] * i) % MOD;
 invfact[100000] = modpow(fact[100000], MOD - 2);
 for (i = 100000 - 1; i >= 0; i--)
  invfact[i] = (invfact[i + 1] * (i + 1)) % MOD;
}

int main(){
 //Decalre variables n,k
 int n,k;
 //input the value from user
 cin>>n>>k;
 //Precomputations
 initialize();
 //call ncr
 cout<<ncr(n,k);
 return 0;
}
```

Time Complexity : **O(n)**
Memory Complexity : **O(n)**

Utilities For Programming by SUPERCOOL276

search

Classic Flipcard Magazine Mosaic Sidebar Snapshot Timeslide

Posted 9th July 2016 by supercool276

0 Add a comment

8th July 2016

Alright! I always wondered what would I write about in my very first blog and then I thought why not share some **programming tips and utilities**.
So here it goes :-

I hope this blog gives you some helpful tips and **CODES**.

Posted 8th July 2016 by supercool276

0 Add a comment