

Implementation of nCr for big values of n

Samyak Ahuja

August 12, 2018

1 Introduction

nCr is the coefficient of the x^r term in the polynomial expansion of $(1+x)^n$ and is given by the formula

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$

To calculate nCr one can find the prime factors of $n!$, $r!$, and $(n-r)!$ and then cancel out the terms. So in essence all we have to do is find the powers of the prime factors of the above mentioned factorials and then subtract the powers of prime factors of $r!$ and $(n-r)!$ from that of $n!$

As an example

$$\begin{aligned}\binom{5}{3} &= \frac{5!}{3!2!} \\ &= \frac{5.4.3.2.1}{3.2.1 * 2.1} \\ &= \frac{2^3.3^1.5^1}{2^2.3^1} \\ &= 2^1.5^1\end{aligned}$$

One can verify last equation at [hackmath](#)

2 Proof of Correctness

How do we know that the answer obtained is correct? One can always calculate it but it isn't practical. On the other hand giving a rigorous proof requires knowledge of Number Theory.

Perhaps the best way of going about this is by proving that the algorithm results in an integer, that is no power of any prime factor is negative.

In nCr we have

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$

Now a key observation here is that the sum of the numbers that we are taking the factorial of in the denominator is exactly equal to the number in the numerator of which we are taking the factorial. Meaning $n = r + (n - r)$. So we can if prove generally that

$$\frac{(n_1 + \dots + n_k)!}{n_1! \dots n_k!}$$

is an integer(which can be proved by Lagrange's theorem) then we can conclude the nCr proof.

3 Code

```
#include <iostream>
#include <cmath>
#include <vector>

bool isPrime(int x){
    if(x == 2 || x == 3) return true;
    for(int i = 2; i * i <= x; i++){
        if(x % i == 0) return false;
        else if((i + 1) > sqrt(x)) return true;
    }
}

//occurence of f in n!
int fact(int n, int f){
    int sum = 0;
    while(n >= f){
        sum += (int) n / f;
        n = (int) n / f;
    }
    return sum;
}

int main(){
    int n,r,nr;
    std::cout<<"Enter n in nCr\t";
```

```

std::cin>>n;
std::cout<<"Enter r in nCr\t";
std::cin>>r;
nr = n - r;

if(n < 0 || r < 0){
    std::cout<<"Enter +ve integers\n";
    return 0;
}
if(r > n){
    std::cout<<"0\n";
    return 0;
}
if(r == 0 || n == r){
    std::cout<<"1\n";
    return 0;
}

//vectors for prime factors of n,r,and n-r
std::vector<int> nVec;
std::vector<int> rVec;
std::vector<int> nrVec;

std::vector<int> nFac(n);
std::vector<int> rFac(n);
std::vector<int> nrFac(n);

for(int i = 2; i <= n; i++){
    //check if i is prime
    if(!isPrime(i)) continue;
    nVec.push_back(i);
}

for(int i = 2; i <= r; i++){
    //check if i is prime
    if(!isPrime(i)) continue;
    rVec.push_back(i);
}

for(int i = 2; i <= nr; i++){
    //check if i is prime
    if(!isPrime(i)) continue;
    nrVec.push_back(i);
}

for(int i = 0; i < nVec.size(); i++){

```

```

        nFac[nVec[i] - 2] = fact(n, nVec[i]);
    }

    for(int i = 0; i < rVec.size(); i++){
        rFac[rVec[i] - 2] = fact(r, rVec[i]);
    }

    for(int i = 0; i < nrVec.size(); i++){
        nrFac[nrVec[i] - 2] = fact(nr, nrVec[i]);
    }

    bool show = false;

    for(int i = 0; i < nFac.size(); i++){
        nFac[i] = nFac[i] - rFac[i] - nrFac[i];
        if(nFac[i] != 0){
            if(show) std::cout<<" * ";
            std::cout<<i + 2<<"^"<<nFac[i];
            show = true;
        }
    }
    std::cout<<"\n";
    return 0;
}

```