## Binomial theorem/pascal's triangle

**(x+y)^n=sum(k=0 to n)(nCk)(X^(n-k))\*y^k**

pretty difficult to remember ryt?

Hence we can reduce this to **(n!)/k!\*(n-k)!**

now we will try code to implement this

1) Dynamic programming!

2) Recursion

1. Dynamic programming approach: 2D array or 1D array?

* We will start off with 2d array

**for(int j=0;j<=i && j<=n2;j++){**

**if(i==j || j==0)**

**{**

**arr[i][j]=1;**

**}**

**else{**

**arr[i][j]=arr[i-1][j-1]+arr[i-1][j];**

**}**

**}**

* We run the inner loop until j<i ie n>r and n<k we also initialize the array with with j=0 and i==j to 1 (initial setup) and compute the nCr values using previously computed values.
* using 1D array only retaining the final value required without storing subsequent results we get a final 1D array

**for(i=0;i<=n;i++)**

**{**

**for(j=mini(i,k);j>0;j--)**

**c[j] = c[j] + c[j-1];**

**}**

* We will run a loop to until we reach “n” and in the inner loop we initialize j to minimum of i and j,by doing so we are eliminating the extra computation that will be done even though it is not required.In c we have the output matrix with pascal's triangle calculated in an single 1D array

2) Recursion

* We compute nCk using recursive approach

**int nCr ( int n, int r ) {  
 if( n == r || r == 0 )   
 return 0;  
 return nCr( n-1, r-1 ) + nCr( n-1, r );  
}**

* The base condition is until n==r or r==0 because we cannot compute nCr when n<r
* We follow this approach when n value is small.ie if n value is big then the space occupied will be more as the stack grows exponentially.

Time complexity will be (2^k)n