Combinatorics, Probability and Expected Value

Youtube link :

https://www.youtube.com/watch?v=FwP-BAtoguM

Contents:

Combinatorics

Product Rule:
If a job A can be done in m ways and after it is done, another job B can be done in n ways, then total number of ways to do both A **AND** B is m*n ways.
(Independent jobs)
Eg. You need to form a team for ICPC of 1 mathematician, 1 programmer and 1 gamer. There are a total of 10 mathematicians, 15 programmers and 5 gamers in your college. In how many ways, can you form a team ?
- 10 ways to select a mathematician
- 15 ways to select a programmer
- 5 ways to select a gamer
Total ways = 10 x 15 x 5

Sum Rule

If a job A can be done in m ways and another job B can be done in n ways, then total number of ways to do both either A **OR** B is m+n ways.

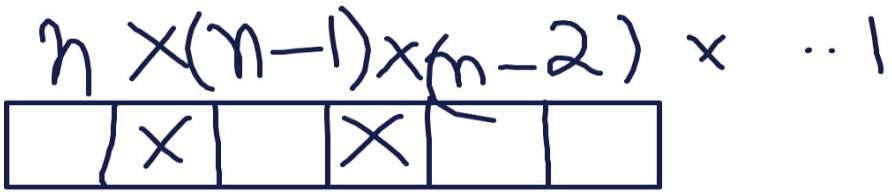
Eg. You need to form a team for ICPC of 1 mathematician or 1 programmer or 1 gamer. There are a total of 10 mathematicians, 15 programmers and 5 gamers in your college. In how many ways, can you form a team ?

- 10 ways to select a mathematician
- 15 ways to select a programmer
- 5 ways to select a gamer

Total ways = 10 + 15 + 5 = 30

Q. No. of ways to arrange n distinct objects.

n! = 1.2.3. .. (n-1)



Q. How many different arrangements of this set of letters?
AAABBBBCCC

(3 A's , 4 B's , 3 C's)
10!/(3! * 4! * 3!)

Permutation:

If n objects are present, we need to select any ordering of r objects. **Order matters.**

There are n * (n-1) * (n-2) *…. (n-r+1) = n! / (n-r)!
This is also called ${}^{n}P_{r}$

Combination:

If n objects are present, we need to select any r objects. **Order doesn't matter.**

${}^{n}C_{r} = n! /((n-r)! * (r)!)$

1. ${}^{n}C_{r} = {}^{n}C_{n-r}$
2. ${}^{n}C_{r} = {}^{n-1}C_{r-1} + {}^{n-1}C_{r}$
[Let's say there is 1 person called Jack in those n people. Either Jack will be there in team of r people or not]
(Pascal's Triangle property)

2. $(1+x)^{n} = \sum_{r=0}^{n} {}^{\wedge}(nCr)\, x^{r}$

3. $\sum_{r=0}^{n} {}^{\wedge}(nCr) = 2^{n}$

(Put x=1 in above equation)

4. $2n\ C\ n = \sum_{k=0}^{n} (nCk)^{2}$

Divide 2n people into 2 groups of n people each.

LHS = $\sum_{k=0}^{n}$ nC(k) * nC(n-k)

= RHS

Q. How to compute nCr mod m? (m is not necessarily prime)

Using DP - (Pascal's triangle property)

```
const int MAX = 1005;
int dp[MAX][MAX]; // initialise all values to -1 in main()
int m;

int nCr(int n, int r)
{
   if(r>n)
        return 0;
   if(r==0 || r==n)
        return 1;
   if(dp[n][r]!=-1)
        return dp[n][r];
    return dp[n][r]=(nCr(n-1,r) + nCr(n-1,r-1))%m
}
```

This would work only when n, r <=10^3.

Q. How to compute nCr mod m? (m is prime)

[n, r <= 10^5]

Using ${}^nC_r = n! /((n-r)! * (r)!)$

${}^nC_r$ mod m = (n! mod m) * inv((n-r)!) * inv((r)!)

For finding inverse, we can use Fermat's theorem

inv of n mod m = n^{m-2} mod m (Use binary exponentiation)

Time complexity: O(log m)
We need to pre-compute modulus of factorials of all numbers <= 10^5.
So, this method will work only when n,m<=10^6

Q. How to compute nCr mod m? (m is prime)
[n,r <= 10^{18}, m <= 10^6]

- By Lucas Theroem
Write n and r in base m
n = $n_{k-1}\ n_{k-2} n_{k-3} \ldots n_0$ [in base m]

r = $r_{k-1} r_{k-2} r_{k-3} \ldots r_0$ [in base m]
(Each digit < m)

nCr mod m = ((((n_{k-1} C r_{k-1} * n_{k-2} C r_{k-2}) mod m) * n_{k-r} C r_{k-r}) mod m ….

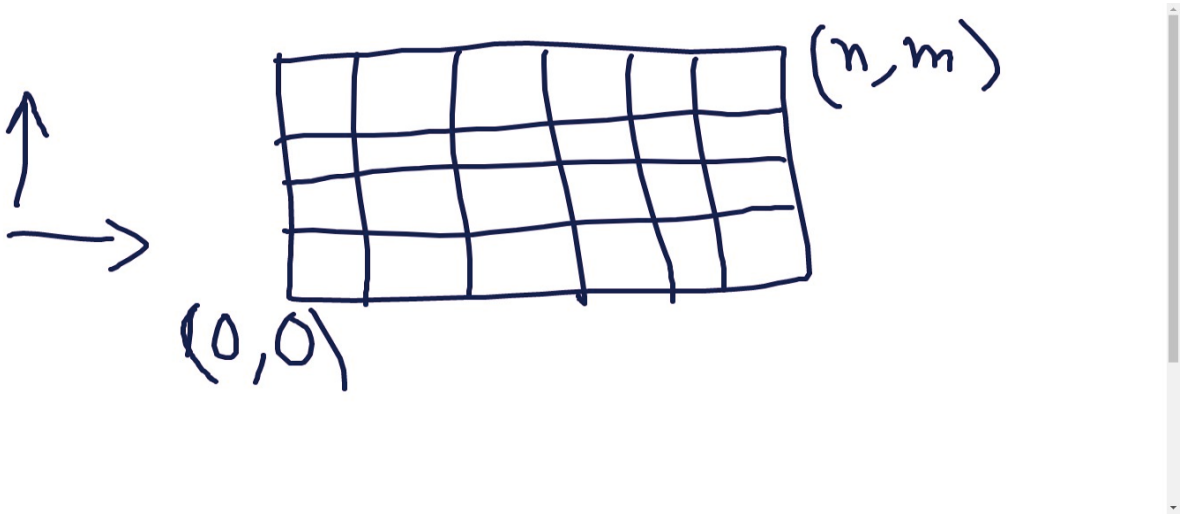
```
int nCrDigits(int n, int r)
{
// … Use any method for nCr of digits, as discussed above
// [Pascal's triangle or Modulo inverse method]
}

int nCr(int n, int r, int m)
{
    if(r==0)
     return 1;
   int dig_n = n%m;
   int dig_r = r%m;
```

```
  return (nCrDigits(dig_n,dig_r,m) * nCr(n/m, r/m, m))%m;
}
```

Time complexity: $O(\log_m n * \text{Time\_complexity(nCrDigits)})$

Try this problem:
https://www.hackerearth.com/problem/algorithm/binary-string-construction-c31f511d/description/
[First read about Catalan numbers in this doc, then try to attempt once again. If you are stuck, you can look at my submission given at the end of this doc]

Q. You are at origin (0,0). You need to reach (n,m) . You can move only in 2 directions - Up or Right .



In every possible path, you need to go n times Right (R) and m times up (U)
RRRR...RUU...U
No. of ways to reach (n,m)
= Number of permutations of the above string
= (n+m) ! / (n! m!)
= (n+m) C n

Q. [3-D version] You are at origin (0,0,0). You need to reach (n,m,p) . You can move only in 3 directions - Up or Right or Forward.

In every possible path, you need to go n times Right (R) and m times up (U) and p times forwards (F)
RRRR...RUU...UF….F
No. of ways to reach (n,m,p)
= Number of permutations of the above string
= (n+m+p) ! / (n! m! p!)

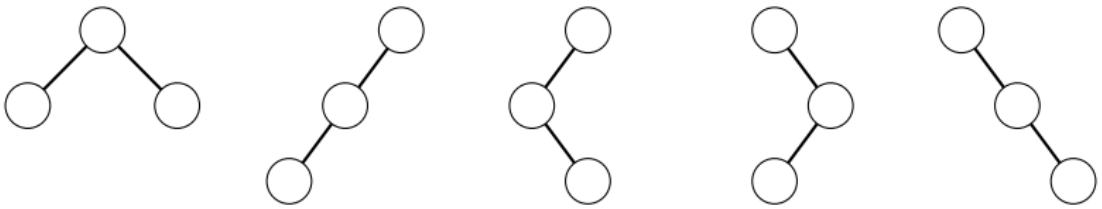
Q. You are at origin (0,0). You need to reach (n,n) . You can move only in 2 directions - Up or Right . But you can't cross the diagonal.
(You can touch the diagonal but you can't cross)
- Your path contains n R's and n U's
It is just a permutation of n R's and n U's
No. of total paths to reach (n,n) = (2n) C n
Now, invalid paths, in which we cross the diagonal are like:
- RUURRRUU
- RRUUURUR
For these paths, find the first point at which we are above the diagonal and reverse the direction of this path after that point [R becomes U and U becomes R].
- RUUUUURR (5 U's and 4 R's)
- RRUUUURU (5 U's and 4 R's)
All these paths would end up reaching at (n-1, n+1) and will have (n+1) U's and (n-1) R's
All the invalid paths would be 2n C (n+1)

Total paths which don't cross the diagonal (Dyck paths)
= Total paths to reach (n,n) - Invalid Paths
= (2n) C n - 2n C (n+1)

= ((2n) C n) / (n+1)
This is also called **catalan number.**

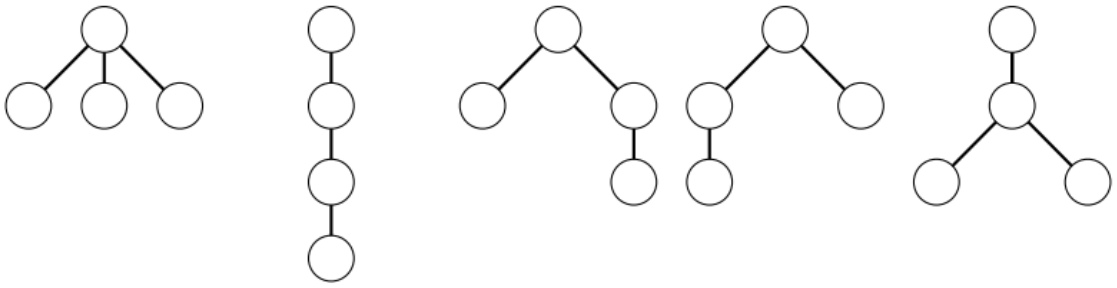
Number of strings of length 2*n having balanced parentheses
= ((2n) C n) / (n+1)
eg. (())() , ()(())
[A balanced parentheses expression is a string of 2n characters $s_1s_2 \ldots s_{2n}$ of letters (and), such that every prefix $s_1s_2 \ldots s_k$ contains at least as many opening parenthesis "(" as the closing parenthesis ")". Given such a string, like (()())(()) we can interpret it as a Dyck path, where (is a step to the right (R), and) is a step upwards (U). Then, the condition that the string is balanced is that, for every partial Dyck path, we have taken at least as many right steps as we have taken up steps. **This is equivalent to the Dyck path never crossing the diagonal,** giving the number of such parentheses expressions are thus also Cn]

No. of possible structures binary tree of n nodes
= ((2n) C n) / (n+1)
For eg. for n=3, you get the below structures:

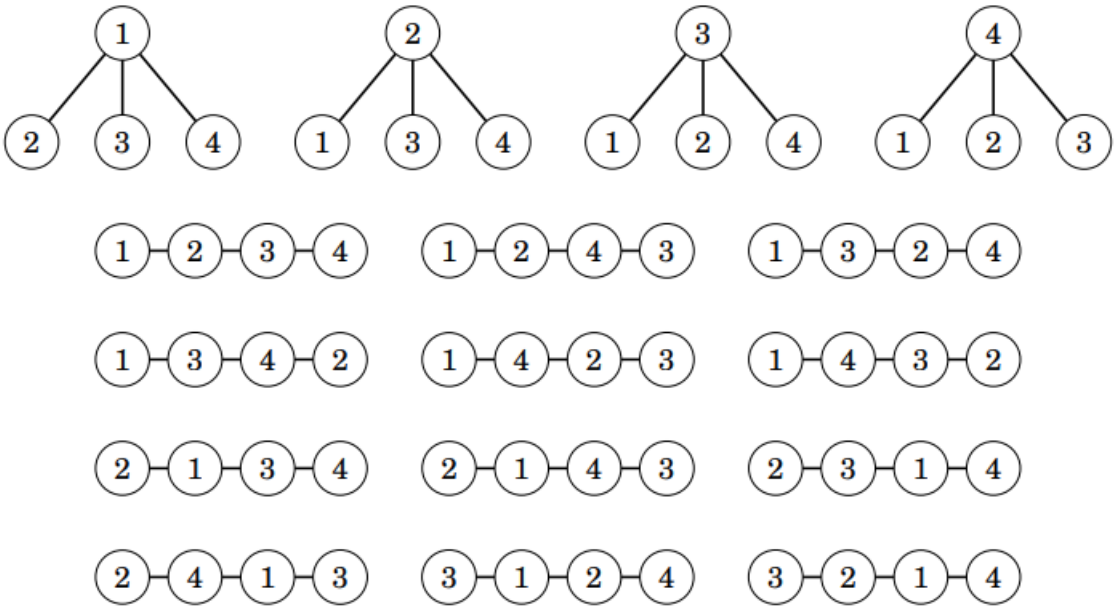


No. of possible structures of rooted trees of n nodes
= (n-1)th Catalan number
[Replace n by n-1 in ((2n) C n) / (n+1)]

For eg. n=4



- Total number of labelled trees of n nodes is $n^{(n-2)}$
(Cayley's formula)



- Number of possible spanning trees in a complete graph of n nodes is also $n^{(n-2)}$
[Cayley's formula]

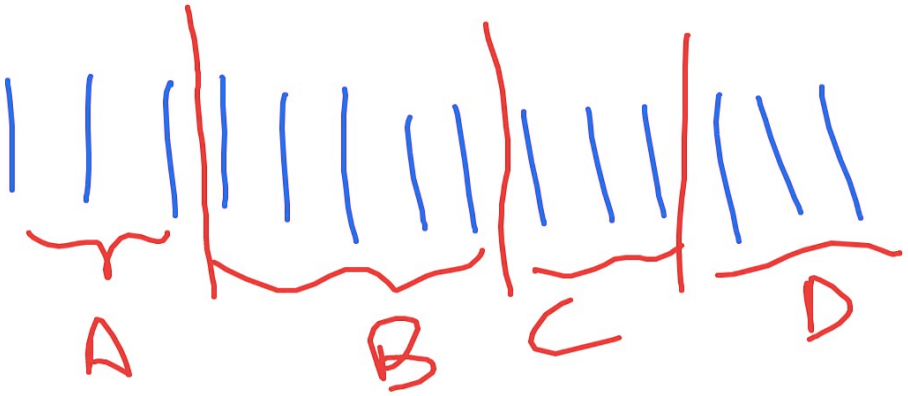
Q. If you need to distribute n similar candies to r people, in how many ways you can do it ?
Each person should get >=0 candies.
Output your answer modulo 10^9+7
[$n<=10^6$, $r<=10^6$]
In the form of equation, we can write as:

$$\sum_{i=1}^{r} a_i = n$$

where each a_i>=0

Consider r-1 sticks as dividers / partition

For now, let us say we have 14 candies and we are distributing to 4 people.



No. of ways = Permutations of 14 candies and 3 sticks
= (14+3)! / ((14)! 3!)

In general, no. of ways to distribute n candies among r people
= (n+(r-1))! / ((n!) (r-1)!)
= **(n+r-1) C (r-1)** [Try to remember yet]

Q. If you need to distribute n similar candies to r people, in how many ways you can do it ?
Each person should get >=c (at least c) candies.

In the form of equation, we can write as:

$$\sum_{i=1}^{r} a_i = n$$

Each a_i >=c

Replace b_i = a_i- c in above equation

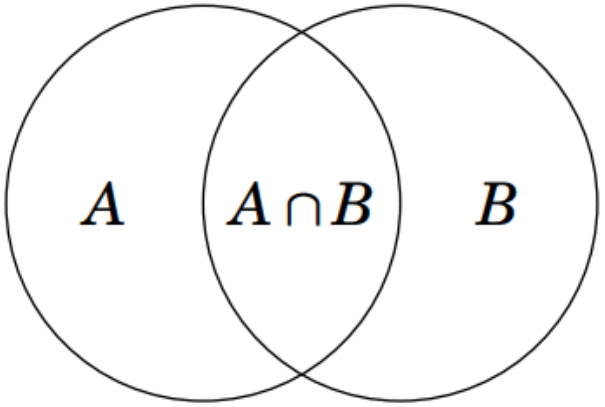
$$\sum_{i=1}^{r} b_i = n - \ r * c, [\ b_i >= 0\ ]$$

No. of ways = **(n - r*c + r - 1) C (r-1)**

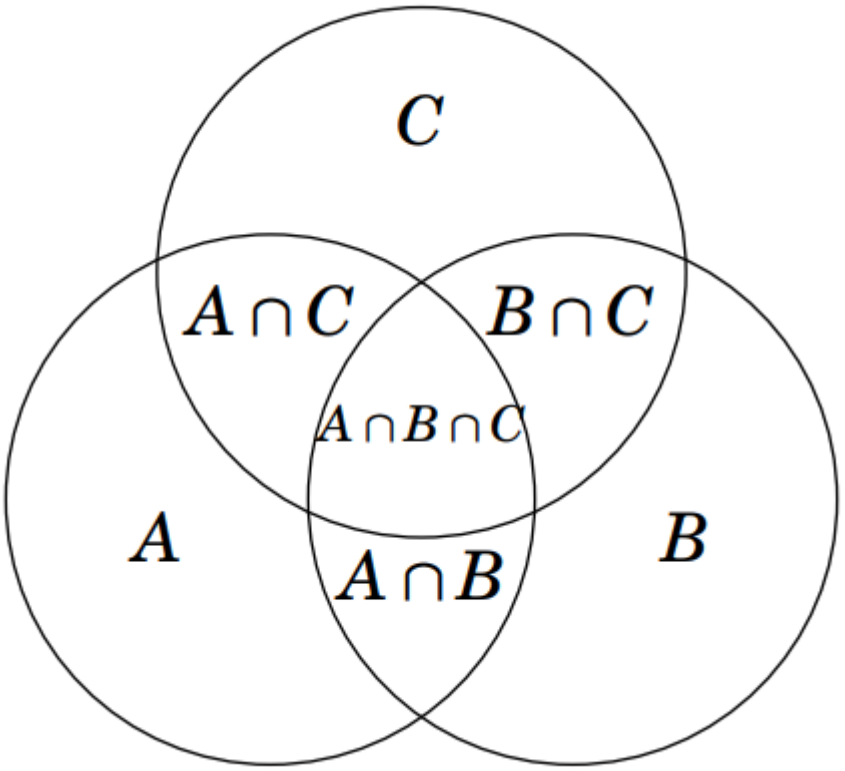
Q. https://atcoder.jp/contests/abc178/tasks/abc178_d

```
  int ans =0;
for(int len=1; len<= s/3; len++)
{
 ans+=nCr(s- len*3 + len-1, len-1)
}
// in nCr(n,r) , when n<0, return 0
```

Principle of Inclusion / Exclusion:



$$|A \cup B| = |A| + |B| - |A \cap B|,$$



$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

It is used when its easy to find intersection rather than union.

Derangements:

If you have n people, who have n houses. Find the number of ways to arrange these n people in n houses, so that no person reaches his own house.

n! - [nC1 * (n-1)! - nC2 * (n-2)! + nC3 * (n-3)! ….]

= n! - [(n)! / 1! - (n)!/2! + (n)!/3! - (n)!/4! …]

Some practice questions:
1. https://codingcompetitions.withgoogle.com/kickstart/round/000000000019ffc8/00000000002d8565#problem

2. https://atcoder.jp/contests/abc178/tasks/abc178_c
3. https://codeforces.com/problemset/problem/1475/E
4. https://leetcode.com/problems/number-of-ways-of-cutting-a-pizza/

Probability

P(Event E) = Favourable Outcomes for E / Total Outcomes

Q. Find the probability of drawing 3 cards with the same value from a shuffled deck of cards.

Method 1:
Total outcomes = 52 C 3
Favourable Outcomes = 13C1 * 4C3
Probaibility = (13C1 * 4C3) / (52 C 3)

Method 2:
Let's simulate the events.
1.(3/51).(2/50)

Q. You have 50 white balls and 50 black balls. 2 boxes are given. Put all the balls in boxes such that it gives maximum probability to select a white ball.

- If we put all white ball in 1 box and all black balls in 2nd box, probability of picking a 1 white ball = (½) * 1 + (½)* 0 = (½)

- If we put only 1 white ball in 1 box and all remaining 49 white balls and 50 black balls in box 2, probability of picking a 1 white ball = (½) * 1 + (½)* (49/99) = (½) + (49/198)
So, the maximum probability is (½) + (49/198) = `0.747`

Practice Problem:
https://codeforces.com/problemset/problem/1279/D

Expected Value

Expected value basically the value that is most likely to occur in a random experiment.

S={s1, s2…. Sn}
X(si) = xi
E(X) = summation (P(si) * xi) for every i

Q) You are rolling a 6-sided dice. What will be the expected value you'll get?

X={1,2,3,4,5,6}
E(X) = ⅙*(1+2+3+4+5+6) = 6*7/2/6 = 21/6 = 3.5

Q) You are rolling a 6-sided dice twice and your score will be the maximum of the values that you get in the two turns. What will be the expected value you'll get?

P(i) = probability that i is the maximum of two numbers
Every turn we can represent as (i,j)
For i to be the greater element, j<=i
Case 1: j=i, -> 1 way
Case 2: j<i, 2*(i-1) ways (i,j) and (j,i) both are different

P(i) = (2*i-1) / 36

E(X) = summation (2*i-1)/36*i
=1/36 * summation(2*i^2 - i) , i=1 to 6
= 1/36*(2*7*13 - 21) = 4.47222222222

Q) You are given a sequence of N distinct numbers. We have to choose one of 2^N - 1 non-empty subsets, uniformly at random. Find EV of the difference between the maximum and minimum element in the subset.

Lets sort the given array.
After sorting,
A1, A2, …. Ai,....., An

Let's consider an element Ai.
If in some nonempty subset, Ai is the maximum element, then its contribution will be Ai.
If in some nonempty subset, Ai is the minimum element, then its contribution will be -Ai.

Next step is,
Finding the probability that Ai is the maximum (or minimum) element in some chosen subset.
P_max(i) = ?
Any element with index < i can be chosen in the subset
2^(i-1)/(2^n -1)

P_min(i) = ?
Any element with index > i can be chosen in the subset
2^(n-i)/ (2*n - 1)

```
for(int i=1;i<=n;i++)
{
        Ans += 2^(i-1)/(2^n -1)*A[i];
        Ans -= 2^(n-i)/ (2*n - 1)*A[i];
}
```

Q) You are given beads of m colours. For each colour, there are infinite number of beads for each colour . You have to make a necklace of n beads using these beads. Find the expected number of distinct colours in the necklace.

P(i) = Probability that colour i is present in the necklace
Xi = 0 if colour i is not present and 1 if colour i is present

E(X) = summation (P(i)*1)

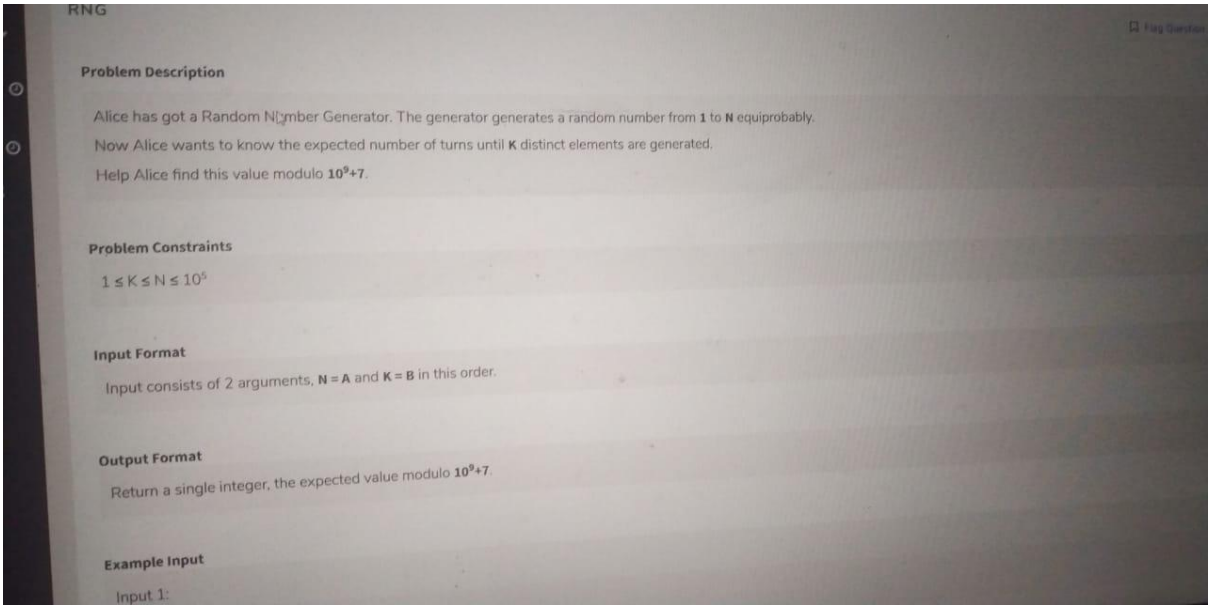
P(i) = same for all i

E(X) = m*P(i)
= m*(1- P’(i))

P’(i) = (m-1)^n / (m^n) // probability that colour i is not present

E(X) = m*(1- (m-1)^n / (m^n))

Q) You are given a random number generator that generates an integer from 1 to N uniformly at random at each turn. It will stop generating once k distinct elements have been generated. Find the expected number of turns.
Let k<=10^5
(Codenation coding round problem)


RNG

Problem Description

Alice has got a Random Number Generator. The generator generates a random number from **1** to **N** equiprobably.
Now Alice wants to know the expected number of turns until **K** distinct elements are generated.
Help Alice find this value modulo 10^9+7.

Problem Constraints

$1 \le K \le N \le 10^5$

Input Format

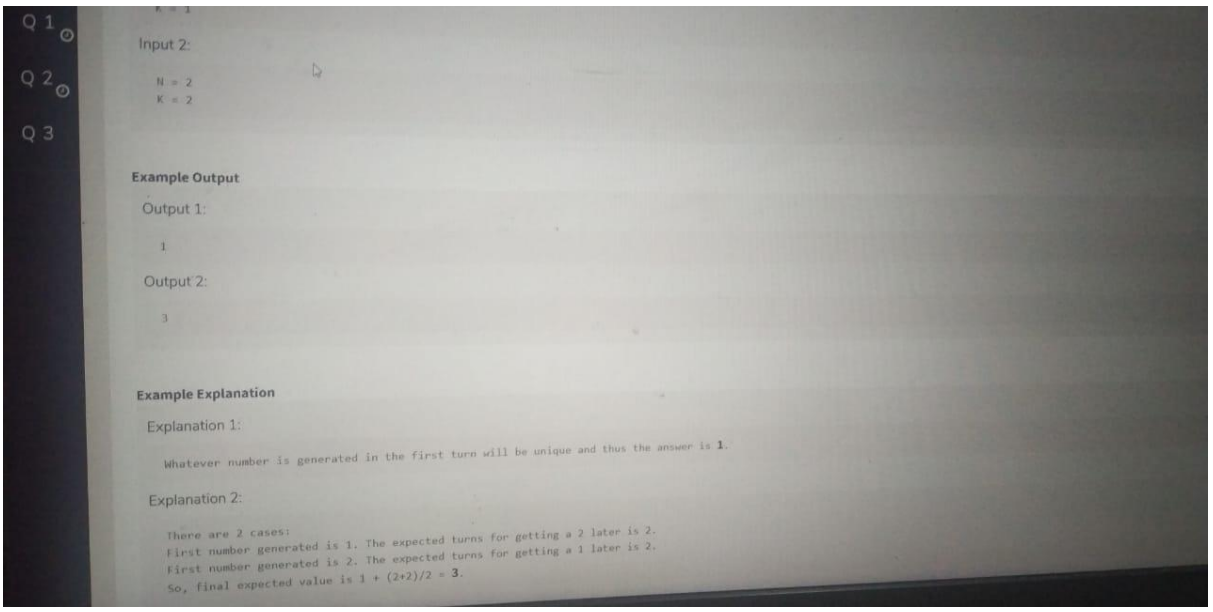
Input consists of 2 arguments, **N = A** and **K = B** in this order.

Output Format

Return a single integer, the expected value modulo 10^9+7.

Example Input

Input 1:



State:
Dp[i] = expected number of extra turns needed to reach **k** distinct elements such that we have **i** distinct elements right now.

Base case:
Dp[k] = ?
Dp[k] = 0 because we don't need any extra turns once we have k distinct elements

Answer = dp[0]

Transition :
Dp[i] :
Case 1: the chosen element is from one of the i distinct elements already chosen -> dp[i]
Case 2: the chosen element is not from one of the i distinct elements already chosen -> dp[i+1]

// 1,2….,i / some i elements
// after one turn where will you go? -> this will be your transition

Dp[i] = 1 + (i/n *dp[i] + (n-i)/n * dp[i+1])
Dp[i] (1-i/n) = (n-i)/n*dp[i+1]
Dp[i] ((n-i)/n) = (n-i)/n*dp[i+1] + n/n

Dp[i] = dp[i+1] + n/(n-i)

O(k) time complexity

My submission for Hackerearth Binary String Construction:
(Only if you are stuck on this problem)
https://csacademy.com/code/lFU1sELx/