CODING BLOCKS
Code Your Way To Success

COMPETITIVE PROGRAMMING LIVE BOOTCAMP {...}

Chapter-02 Mathematics for Competitive Coding

- *Prateek Narang, Founding Member & Mentor*

WARM UP
- *BIRTHDAY PARADOX !*

What is the minimum number of people that should be present in a room so that there's 50% chance of two people having same birthday ?

In a room of just __ people there's a 50-50 chance of two people having the same birthday. In a room of __ there's a 99.9% chance of two people matching.

HINT:
If there are two people in a room, Probability that two will have same birthday = 1/365 = 0.00274 = 0.274%

Probability that two will have different birthdays = 1 - (probability that two have same birthday) = 1 - 0.00274 = 0.9973 = 99.73%

SOLUTION:
23 for 50% probability
70 for 99.9% probability

CODE: http://cb.lk/code/BDAYP

TYPES OF PROBLEMS IN MATHEMATICS

- **Adhoc/Formula Based/Brute Force**
- **Big Integers**
- **Exponentiation**
- **Number Systems/Series**
- **Pigeonhole Principle**
- **Inclusion-Exclusion Principle**
- **Probability & Expectation**
- **Combinatorics**

ADHOC/BRUTE FORCE
These are relatively simpler problems based upon some formula or complete search.

Lotto (HackerBlocks)

In the German Lotto you have to select 6 numbers from the set {1,2,..,49}. A popular strategy to play Lotto - although it doesn't increase your chance of winning — is to select a subset *S* containing *k* (*k* > 6) of these 49 numbers, and then play several games with choosing numbers only from *S*.

For example, for *k* = 8 and *S* = *f*1; 2; 3; 5; 8; 13; 21; 34 there are 28 possible games: [1,2,3,5,8,13], [1,2,3,5,8,21], [1,2,3,5,8,34], [1,2,3,5,13,21], ..., [3,5,8,13,21,34].

Your job is to write a program that reads in the number *k* and the set *S* and then prints all possible games choosing numbers only from *S*.

BIG INTEGERS

In Java, Python it is easy to work with big integers but in C++ it's difficult because the ***long long int*** datatype can store only at max 18 digits.

So, for problems involving Big Numbers(containing 100's of digits) we either use ***Java Big Integer Class*** or Python or we use **Arrays in C++** ! Let us see one example.

Note: There is a BOOST C++ Library which allows us to work with big integers as well.

Computing Large Factorials in C++ !

The Java Big Integer Class !

In Java, the Big Integer class is very powerful and supports lots of operations on big numbers (having 100's of digits) like -

1) Modular Arithmetic
2) Base Conversion
3) GCD Calculation
4) Power Calculation
5) Prime Generation
6) Bit-masking, Bitwise Operations
7) Other Miscellaneous Tasks

It is important to learn about this class, to make our work easy in Programming Contests
Examples -

Example-1 Factorial of Big Number

C++ Code : http://cb.lk/code/CFACT
Java Code : http://cb.lk/code/JFACT
Python Code : http://cb.lk/code/PFACT

Example-2 Julka - Spoj

Problem - http://www.spoj.com/problems/JULKA/
Solution - **http://cb.lk/code/JULKA**

NUMBER SERIES & SEQUENCES

- Most of the sequences are based upon some formula or some recurrence.
- The sequence may contain - AP, GP, HP, Polynomial Sequence, Linear Recurrence etc.
- Other commonly used sequences are - Fibonacci Sequence, Binomial Series, Catalan Numbers etc.

OEIS - ONLINE SEQUENCE FINDER !!!

You can always refer https://oeis.org/ to find out any sequence and its formula.
So You only need to generate output for small inputs and then search for that sequences at OEIS (The Online Encyclopedia of Integer Sequences) ! Let's try to search for few examples -

Example 1 : 1 , 2, 6, 24
Example 2 : 1, 2, 6, 10
Example 3 : 1, 2, 5, 14, 42, 132, 429

BINOMIAL COEFFICIENTS

Binomial Coefficient ^nC_k denotes the number of ways of selecting k items from n items.

C(n,k) = n! / (n-r)! r!

Computing C(n,k) becomes difficult when n and k are large. We might prefer to use dynamic programming or Pascal's Triangle to Compute some are all values of C(n,k)

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

Using this formula we can also build the Pascal's Triangle in a bottom up way -

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
.
.
And so on, Binomial Coefficients are frequently used in problems involving Combinatorics.

CATALAN NUMBERS(Very Important Series)

How many ways are there to construct a Binary Search Tree with 'n' nodes numbered from 1 to N ?

Hint:
Make every possible ith node as the root node and recursively count for number of BST's it its left half and right-half. Do it for every value of i (1<=i<=n) and sum it up.

The formula generating after adding the series is the **nth Catalan Number !!**
It is defined using binomial coefficient notation ^nC_k as -

Cat(n) = $^{2n}C_n/(n+1)$

Cat(0) = 1

Using the above formula , the first few terms of series are -

1, 1, 2, 5, 14, 42, 132, 429, 1430

Another Recursive Formula is -

$C_0 = 1 \quad \text{and} \quad C_{n+1} = \sum_{i=0}^{n} C_i\,C_{n-i} \quad \text{for } n \geq 0;$

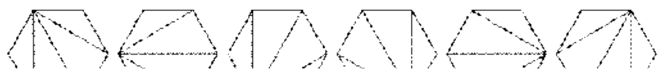
APPLICATIONS OF CATALAN NUMBERS

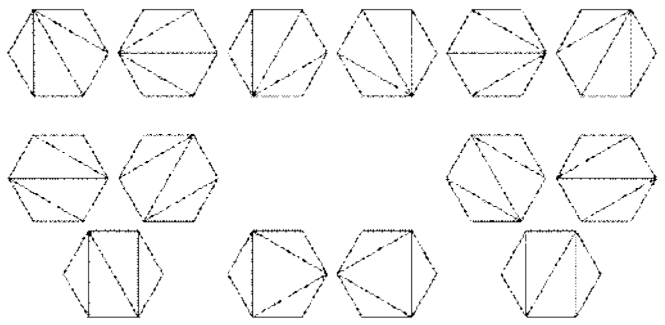
- Number of possible Binary Search Trees with n keys.

- Number of expressions containing n pairs of parentheses which are correctly matched. For n = 3, possible expressions are ((())), ()(()), ()()(), (())(), (()())

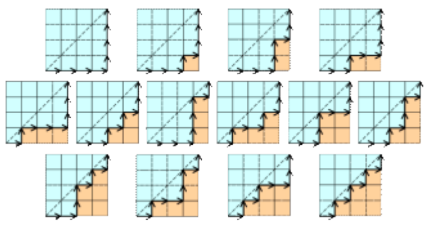
- Number of Ways n + 1 factors can be completely parenthesized, for e.g. N = 3 and 3 + 1 factors : {a,b,c,d}, we have: (ab)(cd), a(b(cd)), ((ab)c)d, (a(bc))(d) and a((bc)d).

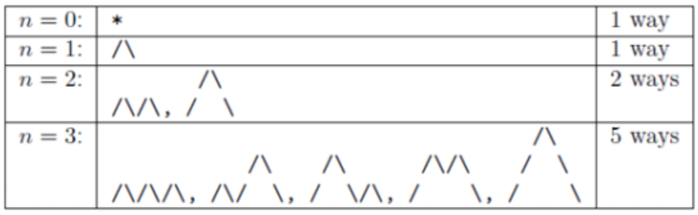
- Number of ways a convex polygon of n+2 sides can split into triangles by connecting vertices.



- 
- Number of different Unlabelled Binary Trees can be there with n nodes

- The number of paths with 2n steps on a rectangular grid from bottom left, i.e., (n-1, 0) to top right (0, n-1) that do not cross above the main diagonal.



- Number of ways to form a "mountain ranges" with n upstrokes and n down-strokes that all stay above the original line. The mountain range interpretation is that the mountains will never go below the horizon
- 

$n = 0$:	*	1 way
$n = 1$:	/\	1 way
$n = 2$:	/\ /\/\, / \	2 ways
$n = 3$:	/\ /\ /\ /\/\ / \ /\/\/\, /\/ \, / \/\, / \, / \	5 ways

Mountain Ranges

-

SOLVING LINEAR RECURRENCES

The problem is generally asking you the n-th term of a linear recurrence. It is possible to solve with dynamic programming if n is small, problem arises when n is very large.

Linear Recurrence

A linear recurrence relation is a function or a sequence such that each term is a linear combination of previous terms. Each term can be described as a function of the previous terms.

A famous example is the Fibonacci sequence: f(i) = f(i- 1) + f(i-2). Linear means that the previous terms in the definition are only multiplied by a constant (possibly zero) and nothing else. So, this sequence: f(i) = f(i-1) * f(i-2) is not a linear recurrence.

Problem

Given f, a function defined as a linear recurrence relation. Compute f(N). N may be very large.

How to Solve ?

Break the problem in four steps. Fibonacci sequence will be used as an example

Step 1. Determine K, the number of terms on which f(i) depends

More precisely, K is the minimum integer such that f(i) doesn't depend on f(i-M), for all M > K.

For Fibonacci sequence, because the relation is: f(i) = f(i-1) + f(i-2), therefore, K = 2.

In this way, be careful for missing terms though, for example, this sequence: f(i) = 2f(i-2) + f(i-4) has K = 4, because it can be rewritten explicitly as: f(i) = 0f(i-1) + 2f(i-2) + 0f(i-3) + 1f(i-4).

Step 2. Determine the F1 vector the initial values

If each term of a recurrence relation depends on K previous terms, then it must have the first K terms defined, otherwise the whole sequence is undefined. For Fibonacci sequence (K = 2), the well-known initial values are:

f(1) = 1
f(2) = 1
Note: We are indexing Fibonacci from 1, f(0) = 0.

We define a column vector Fi as a K x 1 matrix whose first row is f(i), second row is f(i+1), and so on, until K-th row is f(i+K-1). The initial values of f are given in column vector F1 that has values f(1) through f(K):

$$F_1 = \begin{bmatrix} f(1) \\ f(2) \\ \vdots \\ f(k) \end{bmatrix}$$

Step 3. Determine T, the transformation matrix. Construct a K x K matrix T, called transformation matrix, such that

$$TF_i = F_{i+1}$$

Suppose,

$$f(i) = c_1 f(i-1) + c_2 f(i-2) + c_3 f(i-3) + \ldots\ldots + c_k f(i-k)$$

$$f(i) = \sum_{j=1}^{k} c_j f(i-j)$$

Putting i = k + 1

$$f(k+1) = c_1 f(k) + c_2 f(k-1) + c_3 f(k-2) + \ldots + c_k f(1)$$

Hence, the transformation matrix is:

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ c_K & c_{K-1} & c_{K-2} & c_{K-3} & \cdots & c_1 \end{bmatrix}$$

Example For Fibonacci –
C1 = 1, C2 = 1

$$T = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

Step 4. Determine F(n)

$$F_2 = TF_1$$

$$F_3 = TF_2 = T^2 F_1$$

$$F_n = T^{n-1} F_1$$

Therefore, the original problem is now (almost) solved: compute FN as above, and then we can obtain f(N): it is exactly the first row of FN. In case of our Fibonacci sequence, the N-th term in Fibonacci sequence is the first row of:

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{N-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Use Fast Exponentiation!

To compute T^N-1 use exponentiation by squaring method that works in O(log N) time, with this recurrence:

- $A^p = A$, if $p = 1$,
- $A^p = A * A^{p-1}$, if p is odd
- $A^p = X^2$, where $X = A^{p/2}$, otherwise.

Multiplying two matrices takes ***O(K³)*** time using standard method, so the overall time complexity to solve a linear recurrence is ***O(K³ log N)***.

VARIATION

The recurrence relation may include a constant i.e., the function is of the form

$$f(i) = \sum_{j=1}^{k} c_j f(i-j) + d$$

In this variant, the F vector is enhanced to remember the value of d.
It is of size (K + 1) x 1 now -

$$F_i = \begin{bmatrix} f(i) \\ f(i+1) \\ \vdots \\ f(i+k-1) \\ d \end{bmatrix}$$

We now need to construct the T matrix, of size (K x 1)(K x 1) such that

T Fᵢ = Fᵢ₊₁

$$[T] \begin{bmatrix} f(i) \\ f(i+1) \\ \vdots \\ f(i+k-1) \\ d \end{bmatrix} = \begin{bmatrix} f(i+1) \\ f(i+2) \\ \vdots \\ f(i+k) \\ d \end{bmatrix}$$

Hence, the transformation matrix is -

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_K & c_{K-1} & c_{K-2} & c_{K-3} & \cdots & c_1 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

EXAMPLE

Write an efficient code to find the nth term of a Fibonacci sequence.

ANOTHER WAY TO COMPUTE Nth Fibonnaci Number in Log(N) without using Exponentitation !

- $f(2*k) = f(k)*f(k) + f(k-1)*f(k-1)$
- $f(2*k+1) = f(k)*f(k+1) + f(k-1)*f(k)$

Reference – http://codeforces.com/blog/entry/14516
Code – https://codingblocks.com/ide/#/s/2193

Complexity – LogN * (LogLogN) = Log(N) approx
Implementation – Simple Recursion

TIME TO TRY

1. Generate the Transformation matrix for the given sequence.

$$f(i) = 2f(i-1) + 3f(i-2) + 5$$

2. Generate the Transformation matrix for the given sequences and write code to compute nth term.

$$f(i) = f(i-1) + 2i^2 + 3i + 5$$

$$f(i) = f(i-1) + 2i^2 + 5$$

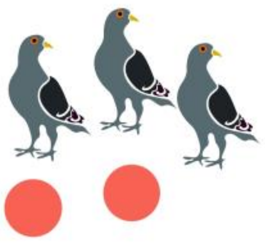
3. **Recursive Sequence Spoj (Spoj)**
   http://www.spoj.com/problems/SEQ/

4. **Recursive Sequence – Version-II (Spoj)**
   http://www.spoj.com/problems/SPP/

5. **Fast Ladders (HackerBlocks)**
   Given a ladder of containing N steps, a person standing at the foot of the ladder can take at max a jump of K steps at every point. Find out the number of ways to reach the top of that Ladder.

6. **Fast Tiling Problem (HackerBlocks)**
   Given n, and grid of size 4 x n, you have to find out the number of ways of filling the grid using 1 x 4 tiles.

PIGEONHOLE PRINCIPLE



The pigeonhole principle is a fairly simple idea to grasp. Say that you have 7 pigeons and 6 pigeonholes. So, now you decide to start putting the pigeons one by one into each pigeonhole.

|p| |p| |p| |p| |p| |p| |p|

So, now, you have one pigeon left, and you can put it into any of the pigeonholes.

|pp| |p| |p| |p| |p| |p| |p|

The point is that when the ***number of pigeons > no. of pigeonholes, there will be at least one pigeonhole with at least two pigeons.***

Hair counting problem
If the amount of hair is expressed in terms of the number of hair strands, the average human head has about 150,000 hair strands. It is safe to assume, then, that no human head has more than 1,000,000 strands of hair. Since the population of Delhi is more than 1,000,000 at least two people in Delhi have the same amount of hair.

EXAMPLES

1. **Divisible Subset(Codechef)**
   To find a non-empty subset of the given multiset with the sum of elements divisible by the size of original multiset.
   https://www.codechef.com/problems/DIVSUBS

   How to use Pigeonhole Principle ?

   $a \% N = x$
   $b \% N = x$
   Then, $(b-a) \% N = (b \% N - a \% N) = (x - x) = 0$

   Let's denote $a_1 + a_2 + \ldots + a_k$ by $\mathbf{b_k}$. So, we obtain:

   $b_0 = 0$
   $b_1 = a_1$
   $b_2 = a_1 + a_2$
   .
   . . .
   $b_n = a_1 + a_2 + a_3 + a_4 + \ldots\ldots + a_N$
   so, $a_L + a_{L+1} + \ldots\ldots + a_R = b_R - b_{L-1}$

   Therefore, if there are two values with equal residues modulo N among b0, b1, .., b_n then we take the first one for L-1 and the second one for R and the required subsegment is found.
   There are **N+1** values of bi and **N** possible residues for N. So, according to the pigeonhole principle the required subsegment will always exist.

2. **The Gray Similar Code(Codechef)**
   Given 'N' 64 bit integers such that any two successive numbers differ at exactly '1' bit. We have

to find out 4 integers such that their XOR is equal to 0.
https://www.codechef.com/problems/GRAYSC

Hint: If we take XOR of any two successive numbers, we will get a number with only 1 set bit and all others will be 0.

How to use Pigeonhole Principle ?

For N = 130, we have '65' pairs i.e. {X1,X2}, {X3,X4}, {X5,X6} ... {X129, X130}. But there exists only 64 possible position for the set bit '1', by pigeonhole principle at least two bits will be set at same positions say {Xi, Xi+1} and {Xj, Xj+1}. If we take xor of pair of these four numbers, we will get 0.
Thus, by pigeonhole principle for all n >= 130, we will always find 4 integers such that their XOR is 0. For n < 130, we can iterate for 3 values of A[i], A[j], A[k] and do a binary search to find 4th number which is (A[i] ^ A[j] ^ A[k])

3. **Holiday Accommodation (Spoj)**
Given a weighted tree, consider there are N people in N nodes. You have to rearrange these N people such that everyone is in a new node, and no node contains more than one person under the constraint that the distance travelled for each person must be maximized. There are N cities having N-1 highways connecting them.
http://www.spoj.com/problems/HOLI/

HINT:
In order to maximize cost:
- All edges will be used to travel around.
- We need to maximize the use of every edge used. Once we know how many time each edge is used, we can calculate the answer.

How to apply Pigeonhole principle ?

Now for any edge E_i, we can partition the whole tree into two subtrees, if one side has n nodes, the other side will have N - n nodes. Also, note that, min(n, N-n) people will be crossing the edge from each side. Because if more people cross the edge, then by pigeon-hole principle in one side, we will get more people than available node which is not allowed in the problem statement. So, E_i will be used a total of 2 * min(n, N-n) times.

$$\cos t = \sum 2^{*}\min(n_i, N - n_i)^{*} weight(E_i)$$

for every edge E_i

TRY IT YOURSEF !

1. **Divisible Subarrays(HackerBlocks)**
Find the number subarrays of the given multiset with the sum of elements divisible by the size of original multiset in linear time.

THE INCLUSION-EXCLUSION PRINCIPLE

Every group of objects(or set) A can be associated with a quantity - denoted |A| - called the number of elements in A or cardinality of A.

If X = A ∪ B and A ∩ B = ∅, then |X| = |A| + |B|.

If A and B are not disjoint, we get the simplest form of the Inclusion-Exclusion Principle:

|A ∪ B| = |A| + |B| - |A∩B|.

|A ∪ B ∪ C| = |A| + |B| + |C| - |A∩B| - |B∩C| - |A∩C| + |A∩B∩C|

In the more general case where there are n different sets Ai, the

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{i=1}^{n} |A_i| - \sum_{1\le i<j\le n} |A_i \cap A_j| + \sum_{1\le i<j<k\le n} |A_i \cap A_j \cap A_k| - \ldots + (-1)^{n-1} \left|\bigcap_{i=1}^{n} A_i\right|$$

2, 3, 5

2,3,5

2,

3,

5

2,3

3,5

2,5

$$\Rightarrow \frac{n}{2} + \frac{n}{3} + \frac{n}{5} - \frac{n}{2\cdot 3} - \frac{n}{3\cdot 5} - \frac{n}{2\cdot 5} + \boxed{\frac{n}{2\cdot 3\cdot 5}}$$

$\boxed{5}$

EXAMPLES

1. Prime Looking Numbers - HackerBlocks

How many number are there < 1000 such that they are Prime Looking i.e. **composite but not divisible by 2,3 or 5** (Ex- 49, 77, 91). Given that there are 168 primes upto 1000.
For any positive number N and m, the number of integers divisible by m which are less than N is **floor((N-1)/m)**

Solution:

divisible by 2 = floor(999/2) = 449
divisible by 3 = floor(999/3) = 333
divisible by 5 = floor(999/5) = 199
divisible by 2.3 = floor(999/6) = 166
divisible by 2.5 = floor(999/10) = 99
divisible by 3.5 = floor(999/15) = 66
divisible by 2.3.5 = floor(999/30) = 33

|2 ∪ 3 ∪ 5| = |2| + |3| + |5| - |2 ∩ 3| - |2 ∩ 5| - | 3 ∩ 5| + |2 ∩ 3 ∩ 5|
= 499 + 333 + 199 - 166 - 99 - 66 + 33
= 733

So, there exists 733 integers upto 1000 which have at-least 2,3 or 5 as divisor. This includes {2,3,5}.
Total number not having 2,3 or 5 as divisor = 999 - 733 = **266.**
Note that this set does not include 2,3 or 5.
Since there are 168 prime numbers upto 1000, but we have already excluded 2,3 and 5, number of prime looking numbers upto 1000 = 266 - 165 - 1(since 1 is neither prime nor composite) = **100**

INCLUSION - EXCLUSION USING BITMASKING !!!

2. Sereja & LCM - Codechef (Hard, Long Contest 9th Question)

We have to find the possible number of arrays: A[1], A[2], A[3],...,A[N] such that A[i] >=1 and A[i] <= M and **LCM(A[1], A[2], .., A[N])** is divisible by D. We have to find the sum of the answers with D = L, L+1, ...,R modulo $10^9 + 7$.

A/Q we have to find the number of array whose LCM is a multiple of a given number(say **'x'**).

Using negation **calculate the number of arrays whose LCM is not a multiple of x (say 'y')**.

Hence, ans = (possible array with m numbers) - y.

Note: The maximum value of the array elements can be 1000, the maximum number of distinct prime factors possible is **4** (2 * 3 * 5 * 7 * 11 > 1000).

Let the prime factors of **x** be **p,q,r,s**
x = (p^a) * (q^b) * (r^c) * (s^d)
p^a: P
q^b: Q
r^c: R
s^d: S

To calculate **y**:
None of element of array have any prime factor that **x** has OR it may have some of it missing

So, calculate the number of arrays such that either **(P or its multiple are not present) OR (Q or its multiple are not present) OR (R or its multiple are not present) OR (S or its multiple are not present)**.

y = |not(P) U not(Q) U not(R) U not(S)|

Applying Principle of Inclusion-Exclusion Principle

A = power(m - m/P,n) + power(m - m/Q,n) + power(m - m/R,n) + power(m - m/S,n);

B = power(m - m/P - m/Q + m/(P*Q), n) + power(m - m/Q - m/R + m/(Q*R), n) . . .

C = power(m- m/P -m/Q - m/R + m/(P*Q) + m/(Q*R) + m/(P*R) - m/(P*Q*R),n) . . .

D = powmod(m - m/P - m/Q - m/R - m/S + m/(P*Q) + m/(Q*R) + m/(P*R) + m/(R*S) + m/(P*S) + m/(Q*S) - m/(P*Q*R)- m/(Q *R*S) - m/(P*Q*S) - m/(P*R*S) + m/(P*Q*R*S),n);

Final Answer will be:
y = A - B + C - D
ans = m ^ n - y
= m ^ n - A + B - C + D

MATHEMATICAL EXPECTATION

Mathematically, for a discrete variable X with probability function P(X), the expected value E(X) is given by **Σ xiP(xi)** the summation runs over all the distinct values xi that the variable can take.
For example, for a dice-throw experiment, the set of discrete outcomes is { 1,2,3,4,5,6} and each of this outcome has the same probability 1/6. Hence, the expected value of this experiment will be 1/6*(1+2+3+4+5+6) = 21/6 = 3.5.

For a continuous variable X with probability density function P(x) , the expected value E(X) is given by **∫ xP(x)dx**

- Mathematical expectation is some sort of average value of your random variable.

- Expected value is not same as "**most probable value**" - rather, it need not even be one of the probable values. For example, in a dice-throw experiment, the expected value, viz 3.5 is not one of the possible outcomes at all.

- Rather the most probable value is the value with max probability.

- The rule of "**linearity of the expectation**" says that E[ax1 + bx2] = aE[x1] + bE[x2].

1. What is the expected number of coin flips for getting a head?

Ans:
Let the expected number of coin flips be x. Then we can write an equation for it -
a. If the first flip is the head, then we are done. The probability of this event is 1/2 and the number of coin flips for this event is 1.
b. If the first flip is the tails, then we have wasted one flip. Since consecutive flips are independent events, the solution in this case can be recursively framed in terms of x - The probability of this event is 1/2 and the expected number of coins flips now onwards is x. But we have already wasted one flip, so the total number of flips is x+1.

The expected value x is the sum of the expected values of these two cases. Using the rule of linearity of the expectation and the definition of Expected value, we get

$x = (1/2)(1) + (1/2)(1+x)$
Solving, we get $x = 2$.

Thus the expected number of coin flips for getting a head is 2.

Q2. What is the expected number of coin flips for getting two consecutive heads?

Let the expected number of coin flips be x. The case analysis goes as follows:
a. If the first flip is a tails, then we have wasted one flip. The probability of this event is 1/2 and the total number of flips required is x+1
b. If the first flip is a heads and second flip is a tails, then we have wasted two flips. The probability of this event is 1/4 and the total number of flips required is x+2
c. If the first flip is a heads and second flip is also heads, then we are done. The probability of this event is 1/4 and the total number of flips required is 2.

Adding, the equation that we get is -
$x = (1/2)(x+1) + (1/4)(x+2) + (1/4)2$

Solving, we get $x = 6$.

Thus, the expected number of coin flips for getting two consecutive heads is 6.

Q3. What is the expected number of coin flips for getting N consecutive heads, given N?

Let the exepected number of coin flips be x. Based on previous exercises, we can wind up the whole case analysis in two basic parts

a) If we get 1st, 2nd, 3rd,...,n'th tail as the first tail in the experiment, then we have to start all over again.
b) Else we are done.

For the 1st flip as tail, the part of the equation is $(1/2)(x+1)$
For the 2nd flip as tail, the part of the equation is $(1/4)(x+2)$
..
For the k'th flip as tail, the part of the equation is $(1/(2^k))(x+k)$
..
For the N'th flip as tail, the part of the equation is $(1/(2^N))(x+N)$
The part of equation corresponding to case (b) is $(1/(2^N))(N)$

Adding,

x = (1/2)(x+1) + (1/4)(x+2) + ... + (1/(2^k))(x+k) + .. + (1/(2^N))(x+N) + (1/(2^N))(N)

Solving this equation is left as an exercise to the reader. The entire equation can be very easily reduced to the following form:
$x = 2^{N+1}-2$
Thus, the expected number of coin flips for getting N consecutive heads is $(2^{N+1} - 2)$.

Q4. Candidates are appearing for interview one after other. Probability of each candidate getting selected is 0.16. What is the expected number of candidates that you will need to interview to make sure that you select somebody?

This is very similar to Q1, the only difference is that in this case the coin is biased. (The probability of heads is 0.16 and we are asked to find number of coin flips for getting a heads). Let x be the expected number of candidates to be interviewed for a selection. The probability of first candidate getting selected is 0.16 and the total number of interviews done in this case is 1. The other case is that the first candidate gets rejected and we start all over again. The probability for that is (1-0.16)*(x+1). The equation thus becomes -

x = 0.16 + (1-0.16)*(x+1)

Solving, x = 1/0.16, i.e. x = 6.25

Q5. (Generalized version of Q4) - The queen of a honey bee nest produces off-springs one-after-other till she produces a male offspring. The probability of producing a male offspring is p. What is the expected number of off-springs required to be produced to produce a male offspring?

This is same as the previous question, except that the number 0.16 has been replaced by p. Observe that the equation now becomes -

x = p + (1-p)*(x+1)
Solving, x = 1/p

Thus, observe that in the problems where there are two events, where one event is desirable and other is undesirable, and the probability of desirable event is p, then the expected number of trials done to get the desirable event is 1/p

Generalizing on the number of events - If there are K events, where one event is desirable and all others are undesirable, and the probability of desirable event is p, then also the expected number of trials done to get the desirable event is 1/p

The next question uses this generalization -

Q6. What is the expected number of dice throws required to get a "four"?

Let the expected number of throws be x. The desirable event (getting 'four') has probability 1/6 (as each face is equiprobable). There are 5 other undesirable events (K=5). Note that the value of the final answer does not depend on K. The answer is thus 1/(1/6) i.e. 6.

Q7. Candidates are appearing for interview one after other. Probability of k-th candidate getting selected is 1/(k+1). What is the expected number of candidates that you will need to interview to make sure that you select somebody?

The result will be the sum of infinite number of cases -

case-1: First candidate gets selected. The probability of this event is 1/2 and the number of interviews is 1.
case-2. Second candidate gets selected. The probability of this event is 1/6 (= 1/2 of first candidate not getting selected and 1/3 of second candidate getting selected, multiplied together gives 1/6) and the number of interviews is 2
case-3. Third candidate gets selected. The probability of this event is 1/2 * 2/3 * 1/4 = 1/12 (= first not getting selected and second not getting selected and third getting selected) and the number of interviews is 3.
..
case-k. k'th candidate gets selected. The probability of this event is 1/2 * 2/3 * 3/4 * .. * (k-1)/k * 1/(k+1). (The first k-1 candidates get rejected and the k'th candidate is selected). This evaluates to 1/(k*(k+1)) and the number of interviews is k
..

[Note that similar to problem 4, here we can't just say - if the first candidate is rejected, then we will start the whole process again. This is not correct, because the probabilty of each candidate depends on it's sequence number. Hence sub-experiments are not same as the parent experiment. This means that all the cases must be explicitly considered.]
The resultant expression will be

x = 1/(1*2) + 2/(2*3) + 3/(3*4) + 4/(4*5) + ... + k/(k*(k+1)) + ...
= 1/2 + 1/3 + 1/4 + ...

This is a well-known divergent series, which means that sum doesnot converge, and hence the expectation doesnot exist.

Q8: A random permutation P of [1...n] needs to be sorted in ascending order. To do this, at every step you will randomly choose a pair (i,j) where i < j but P[i] > P[j], and swap P[i] with P[j]. What is the expected number of swaps needed to sort permutation in ascending order. (Idea: Topcoder)

This is a programming question, and the idea is simple - since each swap has same probability of getting selected, the total number of expected swaps for a permutation P is
E[P] = (1/cnt) * Σ (E[P_s] + 1)
where cnt is the total number of swaps possible in permutation P, and P_s is the permutation generated by doing swap 's'. Since all swaps are equiprobable, we simply sum up the expected values of the resultant permutations (of course add 1 to each to account for the swap done already) and divide the result by the total number of permutations. The base case will be for the array that has been already sorted - and the expected number of permutations for a sorted array is 0.

Q9. A fair coin flip experiment is carried out N times. What is the expected number of heads?

Consider an experiment of flipping a fair coin N times and let the outcomes be represented by the array Z = {a_1, a_2,.. ,a_n} where each a_i is either 1 or 0 depending on whether the outcome was heads or tails respectively. In other words, for each i we have -
ai = if the i'th experiment gave head then 1 else 0.

Hence we have:
number of heads in z = a1+ a2 + ... + an
Hence E[number of heads in z] = E[a1+ a2 + ... + an]
= E[a1] + E[a2] + ... + E[an]

Since ai corresponds to a coin-toss experiment, the value of E[ai] is 0.5 for each i. Adding this n times, the expected number of heads in Z comes out to be n/2.

Q10: (Bernaulli Trials) n students are asked to choose a number from 1 to 100 inclusive. What is the expected number of students that would choose a single digit number?

This question is based on the concept of **bernaulli trials**. An experiment is called a bernaulli trial if it has exactly two outcomes, one of which is desired. For example - flipping a coin, selecting a number from 1 to 100 to get a prime, rolling a dice to get 4 etc. The result of a bernaulli trial can typically be represented as "yes/no" or "success/failure". We have seen in Q5 above that if the probability of success of a bernaulli trial is p then the expected number of trials to get a success is 1/p. is
This question is based on yet another result related to bernaulli trials - If the probability of a success in a bernaulli trial is p then the expected number of successes in n trials is n*p. The proof is simple -

The number of successes in n trials = (if 1st trial is success then 1 else 0) + ... + (if nth trial is success then 1 else 0)
The expected value of each bracket is 1*p + 0*(1-p) = p. Thus the expected number of successes in n trials is n*p.
In the current case, "success" is defined as the experiment that chooses a single digit number. Since all choices are equiprobable, the probability of success is 9/100. (There are 9 single digit numbers in 1 to 100). Since there are n students, the expected number of students that would contribute to success (i.e the expected number of successes) is n*9/100

Q11. What is the expected number of coin flips to ensure that there are atleast N heads?

The solution can easily be framed in a recursive manner -

N heads = if 1st flip is a head then N-1 more heads, else N more heads.
The probability of 1st head is 1/2. Thus

E[N] = (1/2)(E[N-1]+1)+ (1/2)(E[N] + 1)
Note that each term has 1 added to it to account for the first flip.

The base case is when N = 1 :
E[1] = 2 (As discussed in Q2)

Simplifying the recursive case,
E[N] = (1/2)(E[N-1] +1 + E[N] + 1)
= (1/2)(E[N-1] + E[N] + 2)
=> 2 * E[N] = (E[N-1] + E[N] + 2)
=> E[N] = E[N-1] + 2

Since E[1] = 2, E[2] = 4, E[3] = 6,.., in general E[N] = 2N. Thus, the expected number of coin flips to ensure that there are atleast N heads in 2N.
The next problem discusses a generalization :

Q12. What is the expected number of bernaulli trials to ensure that there are at least N successes, if the probability of each success is p?

The recursive equation in this case is -

E[N] = p(E[N-1]+1)+ (1- p)(E[N] + 1)

Solving, E[N]-E[N-1] = p. Writing a total of N-1 equations:

E[N]-E[N-1] = 1/p
E[N-1]-E[N-2] = 1/p
E[N-2]-E[N-3] = 1/p
..
E[2]-E[1] = 1/p

Adding them all, E[N] - E[1] = (n-1)/p. But E[1] is 1/p (lemma -1). Hence E[N] = n/p.

Moral: If probability of success in a Bernoulli trial is p, then the expected number of trials to guarantee N successes is N/p.
This completes the discussion on problems on Mathematical Expectation.

Reference : Codechef

TRY IT YOURSELF !

1. A game involves you choosing one number (between 1 to 6 inclusive) and then throwing three fair dice simultaneously. If none of the dice shows up the number that you have chosen, you lose $1. If exactly one, two or three dice show up the number that you have chosen, you win $1, $3 or $5 respectively. What is your expected gain?

2. There are 10 flowers in a garden, exactly one of which is poisonous. A dog starts eating all these flowers one by one at random. whenever he eats the poisonous flower he will die. What is the expected number of flowers he will eat before he will die?

3. A bag contains 64 balls of eight different colours, with eight of each colour. What is the expected number of balls you would have to pick (without looking) to select three balls of the same colour?

4. In a game of fair dice throw, what is the expected number of throws to make sure that all 6 outcomes appear atleast once?

5. What is the expected number of bernaulli trials for getting N consecutive successes, given N, if the probability of each success is p?

COUPON COLLECTOR PROBLEM

Problem Statement: A certain brand of cereal always distributes a coupon in every cereal box. The coupon chosen for each box is chosen randomly from a set of 'n' distinct coupons. A coupon collector wishes to collect all 'n' distinct coupons. What is the expected number of cereal boxes must the coupon collector buy so that the coupon collector collects all 'n' distinct coupons?

Solution:

Let random variable Xi be the **number of boxes it takes for the coupon collector to collect the i-th new coupon after the i-1th coupon has already been collected.** (Note: this does NOT mean assign numbers to coupons and then collect the i-th coupon. Instead, this means that after Xi boxes, the coupon collector would have collected i distinct coupons, but with only Xi-1 boxes, the coupon collector would have only collected i-1distinct coupons.)
Clearly E(X1)=1, because the coupon collector starts off with no coupons. Now consider the i-th coupon.
After the i-1-th coupon has been collected, then there are n-(i-1) possible coupons that could be the new i-th coupon. Each trial of buying another cereal box, "success" is getting any of the n-(i-1) uncollected coupons, and "failure" is getting any of the already collected i-1 coupons. From this point of view, we see that

p = (n-(i-1)) / n

This is a bernaulli trial with probability of success p and failure (1-p). In bernaulli trial, the expected number of trials for i-th success is 1/p i.e. 1/(success of the i-th outcome).

E(Xi) = 1/p = n/n-(i-1).

To compute the number of cereal boxes X, required by the coupon collector to collect all n distinct coupons:

E(X) = E(X1 + X2 + X3 + X4 + ... + Xn)

E(X) = E(X1) + E(X2) + ... + E(Xn)

E(X) = n(1 + 1/2 + 1/3 + 1/4 + ... + 1/n)

TRY IT YOURSELF !

1. **Favorite Dice (Spoj)**
   What is the expected number of throws of N sided dice so that each number is rolled at least once?
   http://www.spoj.com/problems/FAVDICE

ITS TIME FOR PRACTICE

1. **Fibonacci Sum (Spoj)**
   Given two non-negative integers N and M, you have to calculate the sum (F(N) + F(N + 1) + ... + F(M)) mod 1000000007 where F(N) denotes the nth Fibonacci Number.
   http://www.spoj.com/problems/FIBOSUM/ (Matrix Exponentiation)

2. **Modulo Sum (Codeforces)**
   You are given a sequence of numbers $a_1, a_2, ..., a_n$, and a number *m*.
   Check if it is possible to choose a non-empty subsequence a_{ij} such that the sum of numbers in this subsequence is divisible by *m*.
   http://codeforces.com/contest/577/problem/B

3. **Pigeonhole Tower (Spoj)**
   Pigeon SSNA want to build a tower with some wood walls. Let's describe the tower they want to make using N wood Walls. Refer Spoj for details.
   http://www.spoj.com/problems/PHT/ (Maths, Adhoc)

4. **Tavas and SaDDas (Codeforces)**
   You are given a lucky number *n*. Lucky numbers are the positive integers whose decimal representations contain only the lucky digits 4 and 7. For example, numbers 47, 744, 4 are lucky and 5, 17, 467 are not.
   If we sort all lucky numbers in increasing order, what's the 1-based index of *n*?
   http://codeforces.com/problemset/problem/535/B (Maths, Counting)

5. **Count the Binary Trees (HackerBlocks)**
   Given n, you have to find the number of possible binary trees that can be made using

N nodes.
http://codeforces.com/problemset/problem/535/B (Maths, Catalan Number)

6. **Summing Sums (Spoj)**
Refer Spoj for the problem statement.
http://www.spoj.com/problems/SUMSUMS/ (Mathematics)

7. **Marbles (Spoj)**
Refer Spoj for the problem statement.
http://www.spoj.com/problems/MARBLES/ (Maths)

8. **Player (Codechef)**
Refer Spoj for the problem statement.
http://www.spoj.com/problems/MARBLES/ (Maths, Probability)

9. **Calculating Function (HackerBlocks)**
Refer Hackerblocks for the problem statement.
http://www.spoj.com/problems/MARBLES/ (Maths, Probability)

http://www.spoj.com/problems/REC/ (Recurrence, Big Integers)

Other Relevant Problems
http://www.spoj.com/problems/SUMSUMS
http://www.spoj.com/problems/MARBLES/
https://www.codechef.com/problems/RRPLAYER
https://www.codechef.com/problems/FCTRL (Maths,Factorial)