HackerRank

Project Euler

Author	: Avinash Sorab
Date Created	: 24th Aug 2019
Language Used	: Python 3
Platform	: HackerRank

Contents

1. Multiples of 3 and 5
2. Even Fibonacci numbers
3. Largest prime factor
4. Largest palindrome product
5. Smallest multiple
6. Sum square difference
7. 10001st prime
8. Largest product in a series
9. Special Pythagorean triplet
10. Summation of primes
11. Largest product in a grid
12. Highly divisible triangular number
13. Large sum
14. Longest Collatz sequence
15. Lattice paths
16. Power digit sum
17. Number to words
18. Maximum path sum I
19. Counting Sundays
20. Factorial digit sum
21. Amicable numbers
22. Names scores
23. Non-abundant sums
24. Lexicographic permutations

25. N-digit Fibonacci number

26. Reciprocal cycles

27. Quadratic primes

28. Number spiral diagonals

29. Distinct powers

30. Digit Nth powers

1. Multiples of 3 and 5

The task is to find sum of all the multiples of 3 and 5 below N.

Multiples of 3 and Multiples of 5 form an AP.

Adding them you will get the sum of multiples, but 15 is repeated once. So, the final equation is

$$S_3 + S_5 - S_{15}$$

and

$$S_n = n * (firstterm + lastterm)/2$$

2. Even Fibonacci numbers

The task is to find the sum of all the Fibanocci number below N.

$$F_m = F_{m-1} - F_{m-2}$$

First step is to generate the Fibonacci number and store in an array since we know the Fibonacci number limit can be reached in fewer iterations.

Every 3rd element in the list is even.

Reason: First 2 numbers (0th and 1st) in the series is 0 and 1.

N	Even/Odd
0th Number	0 which is **Even**
1st Number	1 which is Odd
2nd Number	Even + Odd = Odd
3rd Number	Odd + Odd = **Even**
4th Number	Odd + Even = Odd
5th Number	Even + Odd = Odd
6th Number	Odd + Odd = **Even**

and so on

So, you don't have to check if every number is even by dividing it by 2 and checking the reminder.

Instead, use a loop that iterates over every 3rd element starting from 0th element and add it to a sum variable.

3. Largest prime factor

4. Largest palindrome product

5. Smallest multiple

The task is to find the smallest multiple or (LCM) of all the numbers till N where N ranges till 40.

The root concept behind this is iterative calculation of LCM done easy.

$$LCM(a,b,c,d) = LCM(LCM(LCM(a,b),c),d)$$

and

$$LCM(a,b) = (a*b)/GCD(a,b)$$

6. Sum square difference

The task is to find the absolute difference between the sum of the squares of the first natural numbers N and the square of the sum.

If N is 3 then

$$(1+2+3)^2 - (1^2+2^2+3^2) = 22$$

is the answer, absolute difference

The first sequence follows the order

$$(\Sigma n)^2$$

The second sequence follows the order

$$\Sigma n^2$$

Store the result of both and print the absolute difference, either that or Solve them to get a final equation

$$Answer = n*(n+1)*(n-1)*(3n-2)/12$$

and print the same

7. 10001st prime

The task is to print the Nth prime number where N is less than or equal to 10000.

Since we need to do this process for 1000 more times, storing all the first ten thousand times and printing the nth prime would be more efficient time wise.

To store 10000 prime numbers. Find out 10000th prime number, by fact it is 104730.

Sieve of Eratosthenes method of finding out all the prime numbers below N is very much efficient than any other method. It basically takes a number N below which it must find out primes, declares an array of True values of length N.

Then it initializes a counter, whose square should always be less than N.

Till then the counter keeps incrementing, and multiples of that counter is considered, this number is the index of the list whose value is made False since multiples are not primes.

Ex. If 3 is prime, then 6,9,12,15 are not. So those numbers are used as indices whose values are made False.

We get an array of Boolean values which are False at prime indices.

To find out Nth prime number we need to find Nth False value in the list and that's the solution.

8. Largest product in a series

9. Special Pythagorean triplet

10. Summation of primes

11. Largest product in a grid

12. Highly divisible triangular number

13. Large sum

14. Longest Collatz sequence

15. Lattice paths

The task is to find the number of ways there can be to travel from top left to the bottom right of a given lattice structure, only being able to travel right or down.

Since there are different combinations of routes that we can draw from top left to the bottom right of the lattice. We can identify that it is a problem dealing with the combination.

If it was all possible routes among MxN lattice, then the number of possible routes would be calculated as

$$(m+1)*(n+1)$$

Since it was only bottom or right traversal route, the number of possible routes follow the pattern

$$^{(m+n)}C_m$$

or

$$^{(m+n)}C_n$$

Combination Table

M and N	1	2	3	4
1	2	3	4	5
2	3	6	10	15
3	4	10	20	35
4	5	15	35	70

Final answer needs to be $^{(m+n)}C_n\%(10^9+7)$

Got to make the combination calculations efficient

16. Power digit sum

17. Number to words

18. Maximum path sum I

19. Counting Sundays

20. Factorial digit sum

21. Amicable numbers

22. Names scores

23. Non-abundant sums

24. Lexicographic permutations

The task is to find the Nth lexicographic permutation of the string 'abcdefghijkl'.

Example: if 'abcde' is the string and if we are required to calculate 14^{th} lexicographical permutation, i.e., 'adcbe'

This problem requires the knowledge of **factorial number system**.

If we observe closely,

abcde	1!	That is 1st lexicographic permutation
abced	2!	That is 2nd and abdce is 3rd
abedc	3!	That is 6th and acbde is 7th
aedcb	4!	That is 24th and bacde is 25th
edcba	5!	That is 120th and last lexicographic permutation

The pattern here is that

abcde	1!	The fifth letter changes after 0! = every 1 permutation
abced	2!	The fourth letter changes after 1! = every 1 permutation
abedc	3!	The third letter changes after 2! = every 2 permutations
aedcb	4!	The second letter changes after 3! = every 6 permutations
edcba	5!	The first letter changes after 4! = every 24 permutations

If we can find the factoradic representation of the number N, that helps us to find the order of Nth permutation of the string.

Factorial Number System uses factorials instead of powers of 10 as in decimal or powers of binary as in binary number systems

First few numbers represented as factoradics

Number	Factoradic	Expansion
0	0	0 * 0!
1	10	1 * 1! + 0 * 0!
2	100	1 * 2! + 0 * 1! + 0 * 0!
3	110	1 * 2! + 1 * 1! + 0 * 0!
4	200	2 * 2! + 0 * 1! + 0 * 0!

And so on… there is always a unique factoradic for each integer

To calculate factoradic of an integer, keep dividing the integer from an incrementing counter until the quotient is 0, keep reverse track of the reminder and that will be your factoradic of that integer.

Example to find the factoradic of 349,

- 349/1 -> 349 quotient with 0 remainder
- 349/2 -> 174 quotient with 1 remainder
- 174/3 -> 58 quotient with 0 remainder
- 58/4 -> 16 quotient with 2 remainder
- 14/5 -> 3 quotient with 4 remainder
- 2/6 -> 0 quotient with 2 remainder

Factoradic of 349(10) => 242010

Similarly factoradic of 14 is 2100

So to find 14th lexicographic permutation of 'abcde', since the string has 5 characters

Prepend '0' to make factoradic a 5 character number 02100

Now loop through the factoradic number, the ith character serves as an index to fetch characters from the string and delete it from the string.

Ex: 0th character is a in abcde, delete a

2nd character is d in bcde, delete d

1st character is c in bce, delete c

0th character is b in be, delete b

0th character is e in e, delete e

Final string will be adcbe, the 14th lexicographic permutation in 'abcde'.

Special care must be taken to calculate factoradic number, as the number gets bigger, say to calculate factoradic of 13! => 1211109876543210, the factoradic number actually should be treated as an array of 13 elements ['12', '11', '10', '9', '8', '7', '6', '5', '4', '3', '2', '1', '0'], so that reversal operations won't hurt the actual elements and also calculating number of elements is easier because

0123456789101112 when reversed actually becomes 2111019876543210 and also contains 16 characters which are undesirable for our calculations, use list in place of strings to avoid this.

25. N-digit Fibonacci number

Task is to find the first Fibonacci number whose length is N.

This technique requires the knowledge of memoization, if we can memoize efficiently, this can be solved.

We calculate all the Fibonacci number in an order, check its length, if the length of the current fibo number is greater than previous fibo number then store in a dictionary the length as key and the Fibonacci number's occurrence number as the value.

Ex: If 12th Fibonacci number is the first Fibonacci number to have 3 digits Then fibo_dict[3] = 12.

Python can handle really long integer addition so carry out this process till you have dict[5000] = 23922 or somewhere.

The next steps are to just access the value whose key is user input, can be written in a single line.

26. Reciprocal cycles

27. Quadratic primes

28. Number spiral diagonals

29. Distinct powers

30. Digit Nth powers