Class-3-Mathematical-Problems-And-Prefix-Sum

**[PROBLEM] Count digits in a given integer N.**

-1012 <= N <= 1012

**INPUT**

-1342

**OUTPUT**

4

**APPROACH 1:**

Time Complexity: 1 + floor(log10N) = O(logN)

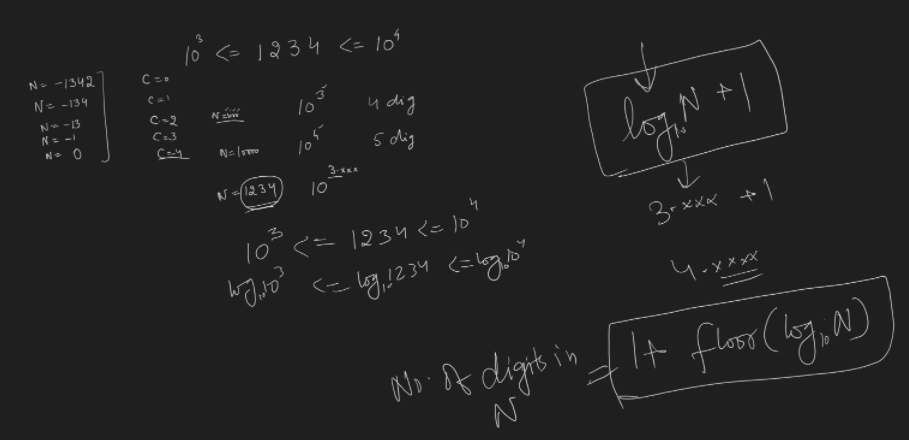
count = 0

Unless N becomes 0

Remove Last Digit and Update N

count++

Count is answer



N =-1342

N%10 *//* ***-2,* a%b gives remainder but keeps sign of numerator**

N = N/10

|  |  |
| --- | --- |
| **Python**  def countDigits(n : int) -> int  if n ==0:  return 1  c =0  while n != 0:  n = n // 10  c += 1  return c | C++  // int: 2 bytes of 4 bytes  // long int: 4 or 8 bytes  // long long int: 8 bytes  // long long int can be written as long long  // long int is same as long  int countDigits(long long int val) {  if (val == 0)  return 1;  int c = 0;  while (val != 0)  {  val = val / 10;  c++;  }  return c;  } |

TC: 1 + floor(log10N) = O(logN)

SC: 0 = O(1)

**APPROACH 2**:

Convert given input to string and then it’s length of the string.

If 0th char is ‘-‘ return length of string as 1 or return length of string.

int countDigits(long long int val)

{

// convert val to +ve

*string* s = *to\_string*(val);

return s.*length*(); // s.size()

}

TC: 1 + log10N = O(logN)

SC: 1 + log10N = O(logN)

**Space complexity**: if we use an array or string of size X in solving a problem. Then arrays or strings are considered to be using X variables (ints or chars) rather than counting them as 1.

Time is given more priority, once time is optimized. Then we think of optimizing space.

**APPROACH 3**:

int countDigits(long long int val)

{

if (val == 0) return 1;

if (val < 0) val = -val;

return 1 + (int)(*log10*(val));

}

**TC**: Time Complexity of built-in log method.

**SC**: Space Complexity of built-in log method.

**TASK: Figure out how to find floor and Iog10 in your own preferred language.**

Log10X = logeX / loge10

**Log of 0 is MATHEMATICALLY NOT DEFINED (JS: NaN)**

**[PROBLEM] Given integer N, reverse and return N.**

INPUT

-102

OUTPUT

-201

STRING

INT

APPROACH

long long reverselnt(int N)

{

long long rev = 0;

while (N != 0)

{

rev = rev \* 10 + N % 10;

N = N / 10;

}

return rev;

}

<https://leetcode.com/problems/reverse-integer/>

***Il -ve of a 32-bit number may not fit into the 32-bit number itself.***

***II Reverse of a 32-bit number may not fit into 32-bit number***

**[PROBLEM] Given N, tell if N is prime or not.**

0 <= N <= 1012

BIG PRIME NUMBER: 1000000007 (109+7), 1000000009 (109+9)

**APPROACH 1:**

count = 0

*i* = [1, *N*]

if (*N*%*i* == 0)

count++;

return count == 2

TC: N = O(N) ===> TLE

SC: 1 =0(1)

**APPROACH 2:**

count = 0

*i* = [1, *N*]

if (*N*%*i* == 0)

count++;

if (count > 2)

return false

return count == 2

TC: N = O(N) ===> TLE

SC: 1 = 0(1)

A Number is divided by its HALF after that except the number itself, we don’t find any other number dividing it.

**APPROACH 3:**

count = 0

*i* = [1, *N* / 2] // for(int i = 1 ; i <= n/2; i++)

if (*N*%*i* == 0)

count++;

if (count > 1)

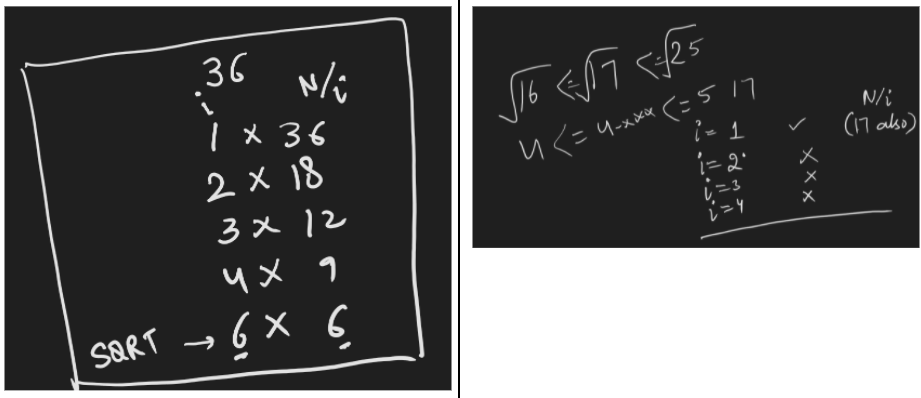
return false;

return count == 1

TC: N/2 = O(N) ===> TLE

**APPROACH 4:**

**Dividing up to sqrt of a Number is good enough to find two factors of divisors**



|  |  |
| --- | --- |
| bool isPrime(long long N)  {  if (N <= 1) return false;  for (int i = 2; i <= *sqrt*(N); i++)  {  if (N%i == 0) return false;  }  return true;  } | **TC: *sqrt*(N) - 1 = 0(*sqrt*(N))**  **SC: 1 = 0(1)**  bool isPrime(long long N)  {  if (N <= 1) return false;  for (int i = 2; i\*i <= N; i++)  {  if (N%i == 0) return false;  }  return true;  } |

**READ:** Primality Check Articles on GEEKS FOR GEEKS

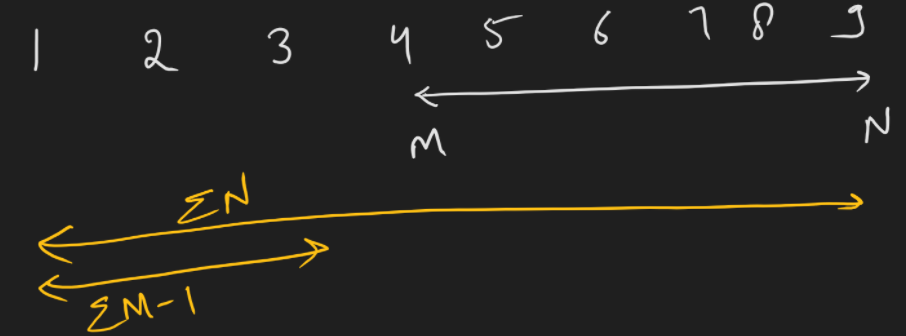
<https://www.geeksforgeeks.org/primality-test-set-1-introduction-and-school-method/>

**[PROBLEM]: Count Factors of a given Number N (<=1012)**

**[PROBLEM] Given M and N, find sum of all numbers in between M and N including Both.**

1 <= M <= N <= 109

Sum(M, N) = SIGMA(N) - SIGMA(M-I)



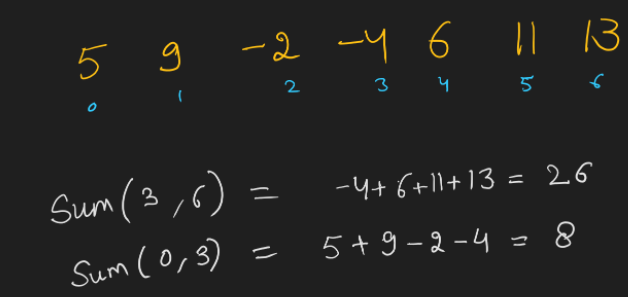
**RANGE SUM QUERIES**

**[PROBLEM]** Given an array A of size N. Answer (T Test Cases) Q queries [L, R] given sum of all elements of array in range A[L] to A[R]

1 <= N <= 106

1 <= Q <= 105

0 <= L <= R <= N-1



|  |  |
| --- | --- |
| SAMPLE INPUT  7  5 9 -2 -4 6 11 13  2  3 6  0 3 | SAMPLE OUTPUT  26  8 |

**BRUTE FORCE**

int N;

*Read* N;

Declare *Array* A; // Assume it takes constant time,

for *i*=[0, N-1] // N

*Read* A[*i*];

int Q;

*Read* Q;

for qi = [0, Q - 1] // Q

int L, R;

*Read* L, R;

int s = 0;

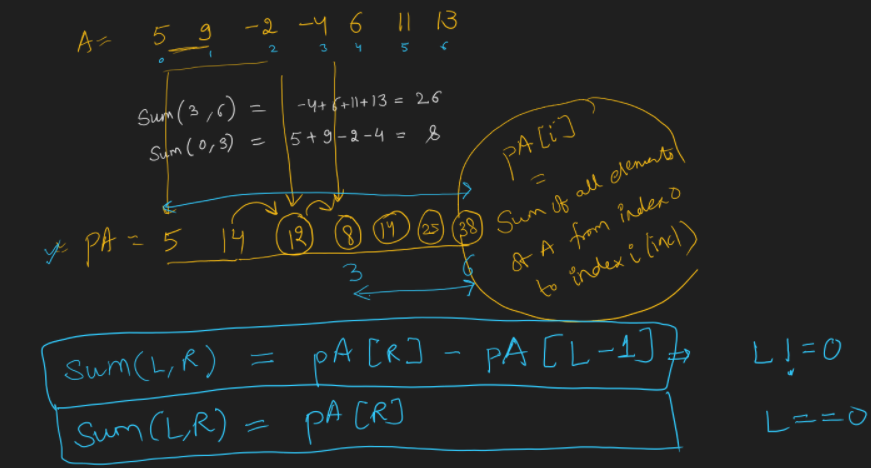
for *i* = [L, R] // R - L + 1

s+= A[*i*] ;

print(s, "\n");

**TC: N + Q \* (R - L + 1) = Worst Case N + Q \* N = 0(Q\*N) == = > TLE**

**SC : 2 = 0(1)**



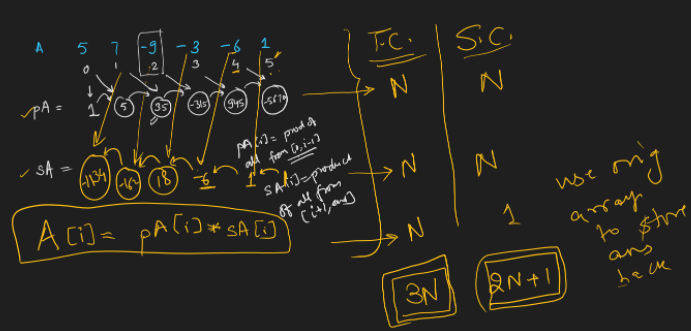
1. How do we fill a prefixSum array in O(N) time.
2. How do we use the prefixSum array to answer each query in O(1) time.

RANGE SUM PROBLEMS

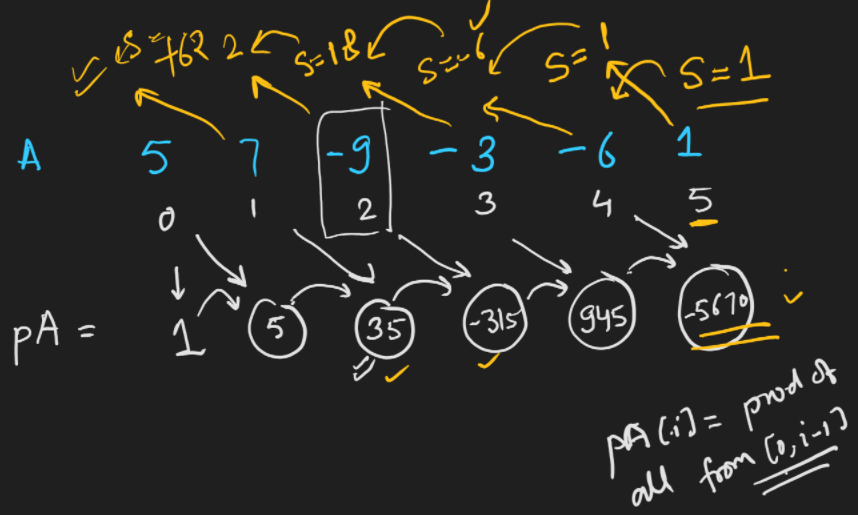
1. Prefix Sum
2. Each Element needs answers from all on LEFT and all on RIGHT.
3. Prefix MAX
4. Suffix MAX

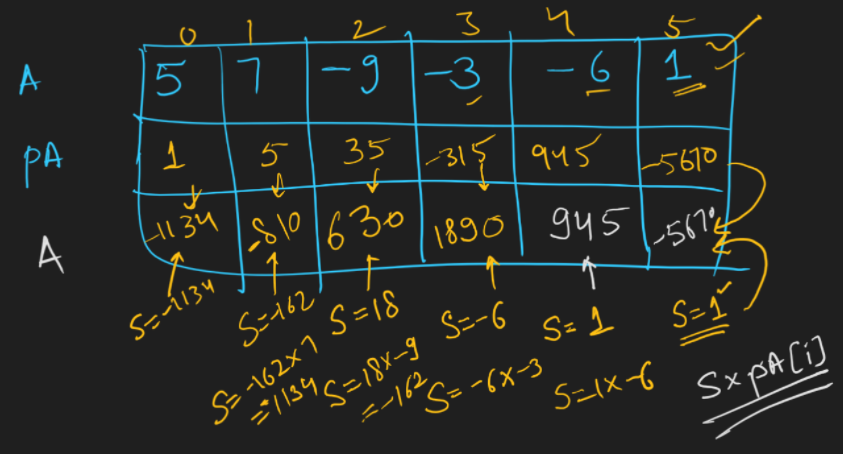
**[PROBLEM]** Fill A[i] with product of all numbers except A[l]

<https://leetcode.com/problems/product-of-array-except-self/>



Can you reduce Time Complexity: 2N and Space to N + constant variables (2 or 3 max)





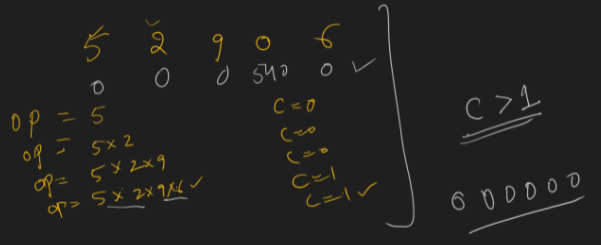
[APPROACH]

Take product of entire array (be aware of overflow and divide by 0)

A[i] = entireProd / A[i] (for non-zeroes)

If there is a single 0 in original array.

Answer array will have all 0s, except index that has 0.



**OPTIMIZED APPROACH**

int N;

Read N;

Declare prefixSumA; **// Assume it takes constant time.**

for i=[0, N-1] // N

**Fill prefixSumA[i]**

int Q;

Read Q;

for qi = [0, Q-1] // **Q**

int L, R;

Read L, R;

if(L==0)

print(prefixSumA[R])

else

print(prefixSumA[R] - prefixSumA[L-1])

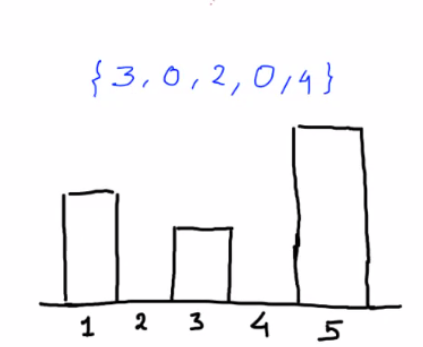
Time Complexity: Q\*1 + N = N + Q = O(N+Q) = ACCEPTED

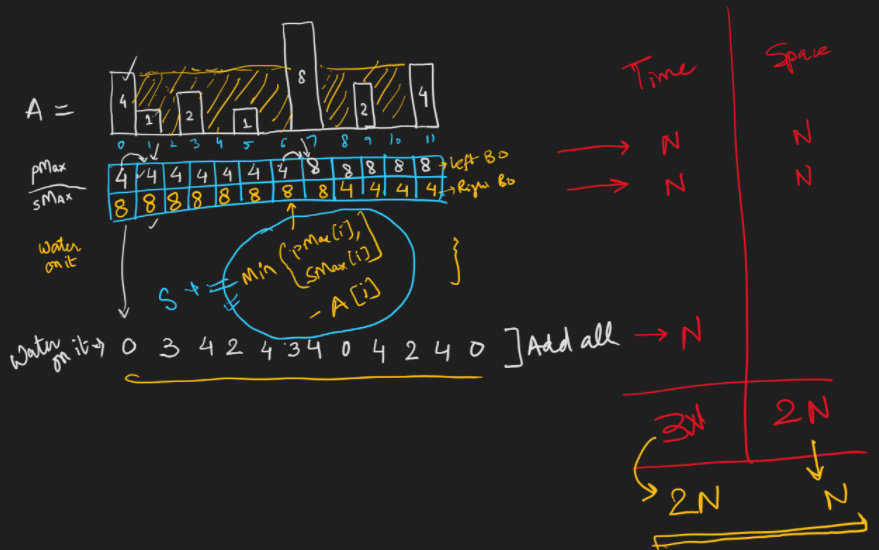
Space Complexity: 1 (We used original array itself to be used as prefixSum array)

<https://leetcode.com/problems/range-sum-query-immutable/>

**[PROBLEM] RAIN WATER TRAPPING**

<https://leetcode.com/problems/trapping-rain-water/>





EXERCISES

**Do at least 10 new problems in this week on PREFIX SUM from LEETCODE**

<https://leetcode.com/tag/prefix-sum/>

<https://leetcode.com/problems/maximum-fruits-harvested-after-at-most-k-steps/>