(1) The fibonacci numbers, commonly denoted $F(n)$ form a sequence, called the fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is

$F(0) = 0, \; F(1) = 1$

$F(n) = F(n-1) + F(n-2)$, for $n > 1$

Given n, calculate $F(n)$.

Example 1:

Input: $n = 2$

Output: 1

Explanation $F(2) = F(1) + F(0) = 1 + 0 = 1$.

Given that,

$F(0) = 0 \; , \; F(1) = 1$

$F(n) = F(n-1) + F(n-2)$ —— ①

$F(n-1) = F(n-2) + F(n-3)$ —— ②

$F(n-2) = F(n-3) + F(n-4)$ —— ③

Sub ③ in ②

$F(n-1) = F(n-3) + F(n-4) + F(n-3)$

$F(n-1) = 2F(n-3) + F(n-4)$ —— ④

Sub ④ in ①

$F(n) = 2F(n-3) + F(n-4) + F(n-2)$

After k^{th} iteration.

$F(n) = 2F(n-3) + F(n-4) + F(n-2)$

$F(n) = F(n-4) + 2F(n-3) + F(n-2)$

$F(n) = F(n-k) + 2F(n-(k-1)) + F(n-(k-2))$

Initializing the $F(1)$

$\Rightarrow n-k = 1$

$n = k+1, k = n-1.$

$F(n) = F(1) + 2F(n-(n-1-1)) + F(n-(n-1-2))$

$= 1 + 2F(n-n+2) + F(n-n+3)$

$= 1 + 2F(2) + F(3)$

$= 1 + 2F(2) + F(3)$ $\Rightarrow F(2) = F(1) + F(0)$

$= 1 + 0$

$= 1 + 2(1) + 2$ $F(2) = \cancel{0}\ 1$

$= 5$

$F(3) = F(2) + F(1) + F(0)$

$F(n) = 5 \rightarrow$ order of growth $= 1 + 1 + 0$

$= O(1)$ - Numerical/constant

$\Rightarrow n-k = 0 \Rightarrow n = k$ $F(3) = 2$

$F(n) = F(0) + 2F(n-(n-1)) + F(n-(n-2))$

$= 0 + 2F(1) + F(2)$

$= 0 + 2 + 1 = 3$

$F(n) = 3 \rightarrow$ order of growth $= O(3)$ - Numerical/constant.

(3) Given the head of a linked list, reverse the nodes of the list k at a time, and return the modified list. k is a positive integer and is less than or equal to the length of linked list. If the number of nodes is not a multiple of k then left-out nodes, in the end should remain as it is.

You may not alter the values in the list's nodes, only nodes themselves may be changed.

Given that,

'n' is the nodes in the linked list

'k' be the time consumed.

$n \leq k \Rightarrow$ the algorithm considered as best case

$T(n) = mk$

let is satisfies for $(m+1)k$

$T(n) = (m+1)k$

$T(n) = mk + k$ $\qquad n \leq k \Rightarrow n = k$.

$T(n) = mn + n$.

$T(n) = mn + n$

$\therefore$ order of growth

$= O(mn) \rightarrow$ linear.

$O(k) + O(n) = O(n)$.

$\therefore$ Satisfies that $O(mn) \rightarrow$ linear.

(5) Given a string expression of numbers and operators. return all possible results from computing all the different possible ways to group numbers and operators.
You may return the answer in any order.
The test cases are generated such that the output values fit in a 32-bit integer and the number of different

results does
not exceed 104.

Example 1:
Input: expression = "2-1-1"
output: [0,2]
Explanation:
((2-1)-1 = 0
((2-(1-1)) = 2.

Given, that

Expression = "2-1-1"

output = [0,2].

((2-1)-1) = 0 ⇒ (2-(1-1)) = 2

$$F(n) = \sum_{i=1\,(+,-,*)}^{n-1} a \text{ op } b$$

$$F(1) = 1$$

$$F(n-1) = \sum_{i=1}^{n-1} a \text{ op } b$$

$$F(n-1) = 1 \le i \le n.$$

are [2,2,3] then 6 and 12 are nice divisors, while 3 and 4 are not.

Return the number of nice divisors of n. Since that number can be too large, return it modulo 109 + 7.

Note that a prime number is a natural number greater than 1 that is not a product of two smaller natural numbers. The prime factors of a number n is a list of prime numbers such that their products equals n.

Given that

prime factor of a Number.

'n' is the ~~co~~ prime factors.

$T(n) = O(1)$ if $\overset{n}{q} = 0$

$T(n) = O(n\log n)$ if $n > 0$

$T(n) = n\log n$.

$T(n-1) = (n-1)\log(n-1)$.

$T(n) = 2^P \rightarrow$ prime factors representation.

for $n = 0$

$T(n) = 0\log 0$.

$T(n) = 0$.

for $n = 0$

$T(n) = 1\log 1$

$T(n) = 1$

for $n = k$

$$T(k) = k \log k$$

~~for~~ $n = k+1$

$$T(k+1) = (k+1)\log(k+1)$$

$$= k \log k$$

order of growth = $\log(n)$ → logarmthmic

ith n vertices. The