Friends Pairing Problem

Easy Accuracy: 43.16% Submissions: 57066 Points: 2

Given N friends, each one can remain single or can be paired up with some other friend. Each friend can be paired only once. Find out the total number of ways in which friends can remain single or can be paired up.
Note: Since answer can be very large, return your answer mod 10^9+7.

Example 1:

```
Input:N = 3
Output: 4
Explanation:
{1}, {2}, {3} : All single
{1}, {2,3} : 2 and 3 paired but 1 is single.
{1,2}, {3} : 1 and 2 are paired but 3 is single.
{1,3}, {2} : 1 and 3 are paired but 2 is single.
Note that {1,2} and {2,1} are considered same.
```

Example 2:

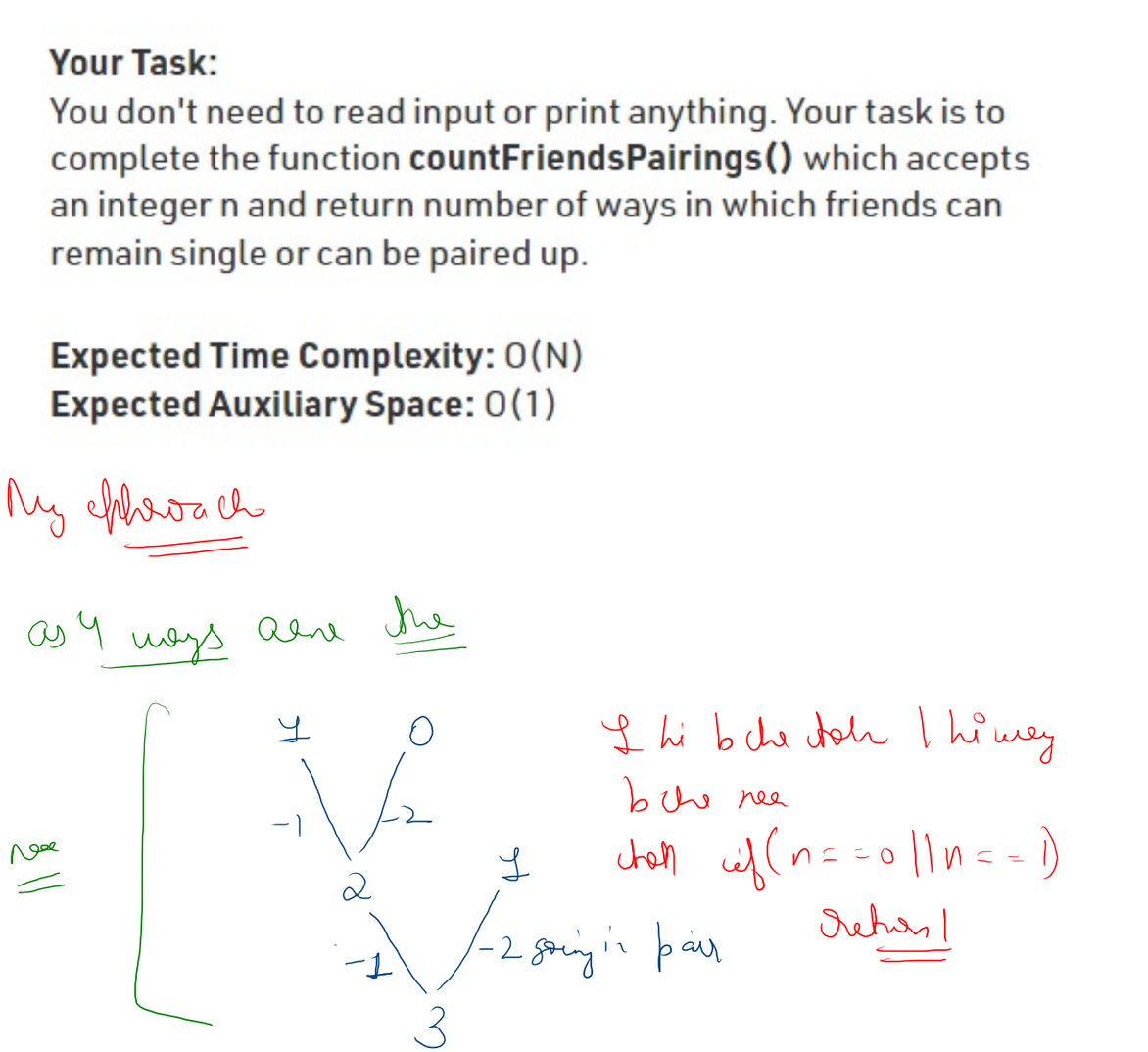
```
Input: N = 2
Output: 2
Explanation:
{1} , {2} : All single.
{1,2} : 1 and 2 are paired.
```

Your Task:
You don't need to read input or print anything. Your task is to complete the function **countFriendsPairings()** which accepts an integer n and return number of ways in which friends can remain single or can be paired up.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

My approach

as 4 ways are the



pair is ya toh nai akele jaay ge pair nai bakiapne khud dekh lo

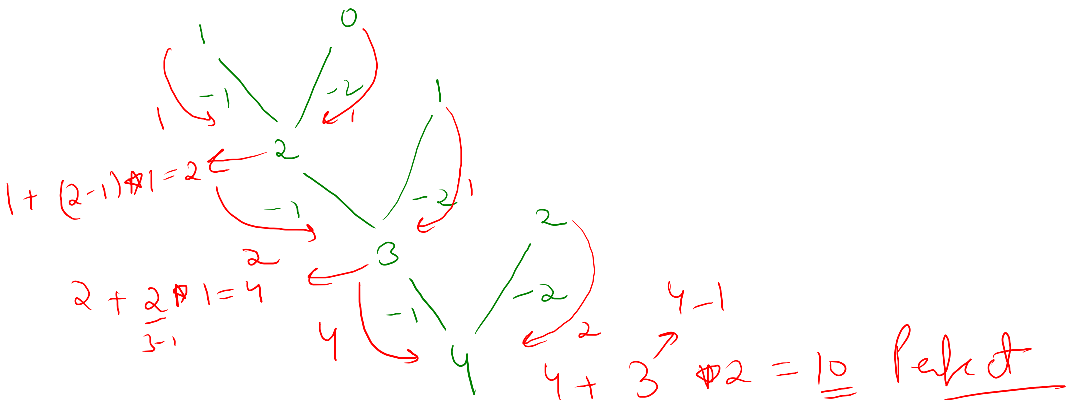
pair nai jane ke (n-1) logo ke sath pair do skta

Eg {1,2,3,4,5} → {1} single

{1,2} {1,4}
{1,3} {1,5}] 4 pair

$(n-1) * f(n-2) + f(n-1)$ → ek ko le lo {1,2} uske answer ko (n-1) se multiply kar do 4 pair ko answer aa jayega

Perfect



Memoization

```java
class Solution
{
    public long cfp(int n,long [] dp,long mod){
        if(n==0||n==1) return dp[n]=1;
        if(dp[n]!=0) return dp[n];
        return dp[n]=(cfp(n-1,dp,mod)+(n-1)*cfp(n-2,dp,mod))%mod;
    }


    public long countFriendsPairings(int n)
    {
        long mod=(long)(1e9+7);
        long dp[]=new long[n+1];
        return cfp(n,dp,mod);
    }
}
```

Observation

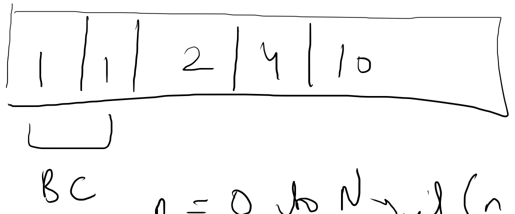
For Input:

4

Your Output:

1 1 2 4 10 10

(1 1 2 4 10 → dp; 10 → answer)



n = 0 to N → if(n==0||n==1) dp[n]=1

$dp[n] = dp[n-1] + (n-1) * dp[n-2]$

Now need to do observation do some tabulation

On next page tabulation + optimization

```java
public long countFriendsPairings(int N)
    {
        long mod=(long)(1e9+7);
        long dp[]=new long[N+1];
        for(int n=0;n<=N;n++){
            if(n==0||n==1) {
                dp[n]=1;
                continue;
            }

         dp[n]=(dp[n-1]+(n-1)*dp[n-2])%mod;
        }
        return dp[N];
    }
}
```

tabulation ↑

As we need only previous 2 values why to have whole array

1, 1, 2, 4, 10   n = 4

1, 1, 2, 4, 10
n1 n2 res

$n_1 = n_2$
$n_2 = res$

Till n, reaches to
so 0 to < N ke loop
chalega

$n_1 * (n-1) + n_2$

$2 = 1 * 1 + 1$
n=2 so (n-1) choice
so n-1 = 1

→ 0 se chalae toh
$(n+2-1) = (n+1)$ multiply hoga

Optimized

```java
public long countFriendsPairings(int N)
{
    long mod=(long)(1e9+7);
    long n1=1;
    long n2=1;
    for(int n=0;n<N;n++){
        long res=(n2+(n+1)*n1)%mod;
        n1=n2;
        n2=res;
    }
    return n1;
}
```