APPLICATION CLUB

What is recursion? And why is it important?

when a thing is defined in terms of itself.

Examples

A man have 100rs
He spent 10rs daily untill he completes his 100rs means the condition reaches its end after 10 days like each day 10rs

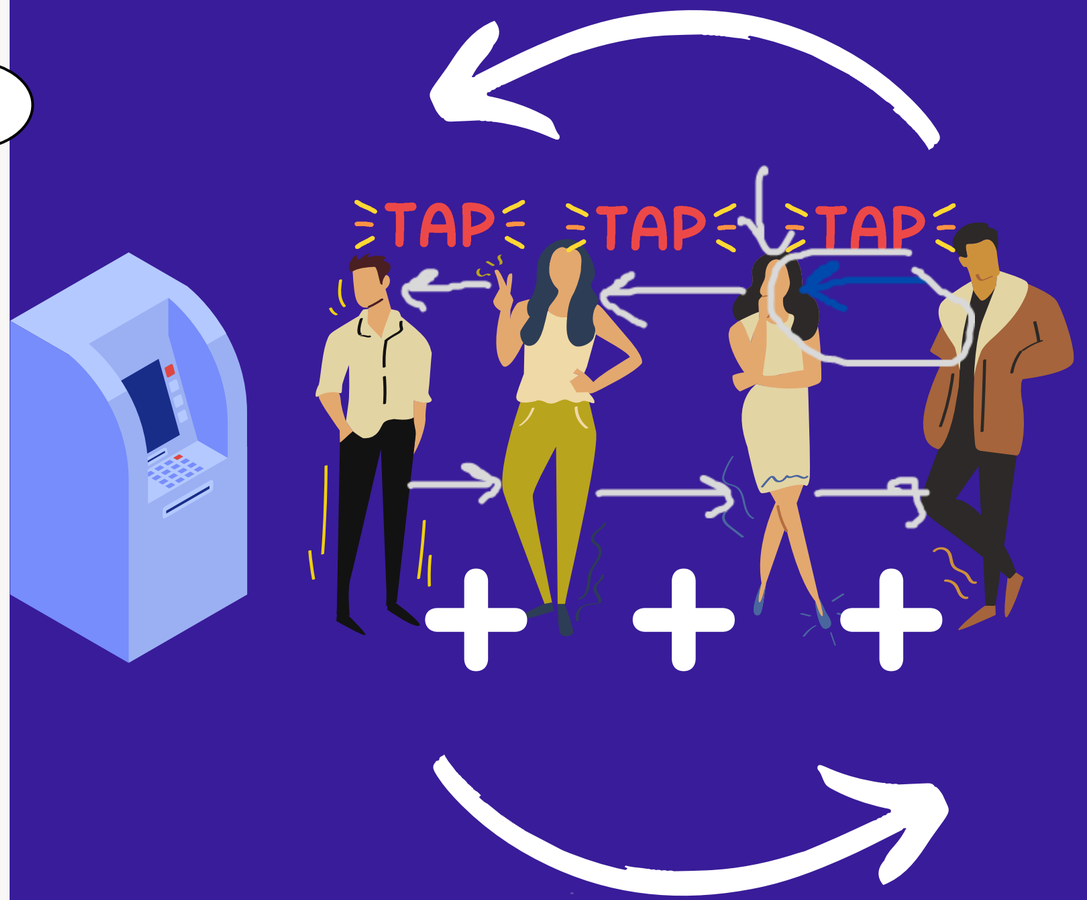
Teacher asks you 100 Rs money for school function,
You ask mummy 200,
Mummy asks papa 500.
Now papa is base case.

Coronavirus conference gets cancelled because of Coronavirus

Define what is recursion?

Recursion is a powerful technique that helps us bridge the gap between complex problems being solved with elegant code.

Demonetization



How to set up a recursion problem

Step 1

try to break the problem into subproblems

This step will be called finding optimal substructure for the problem

Step 2

Try to find edge cases/ base cases

Step 3

Test your code

Step 4

Find complexity of your program and see if it can be optimized

This step will be called finding overlapping subbproblems

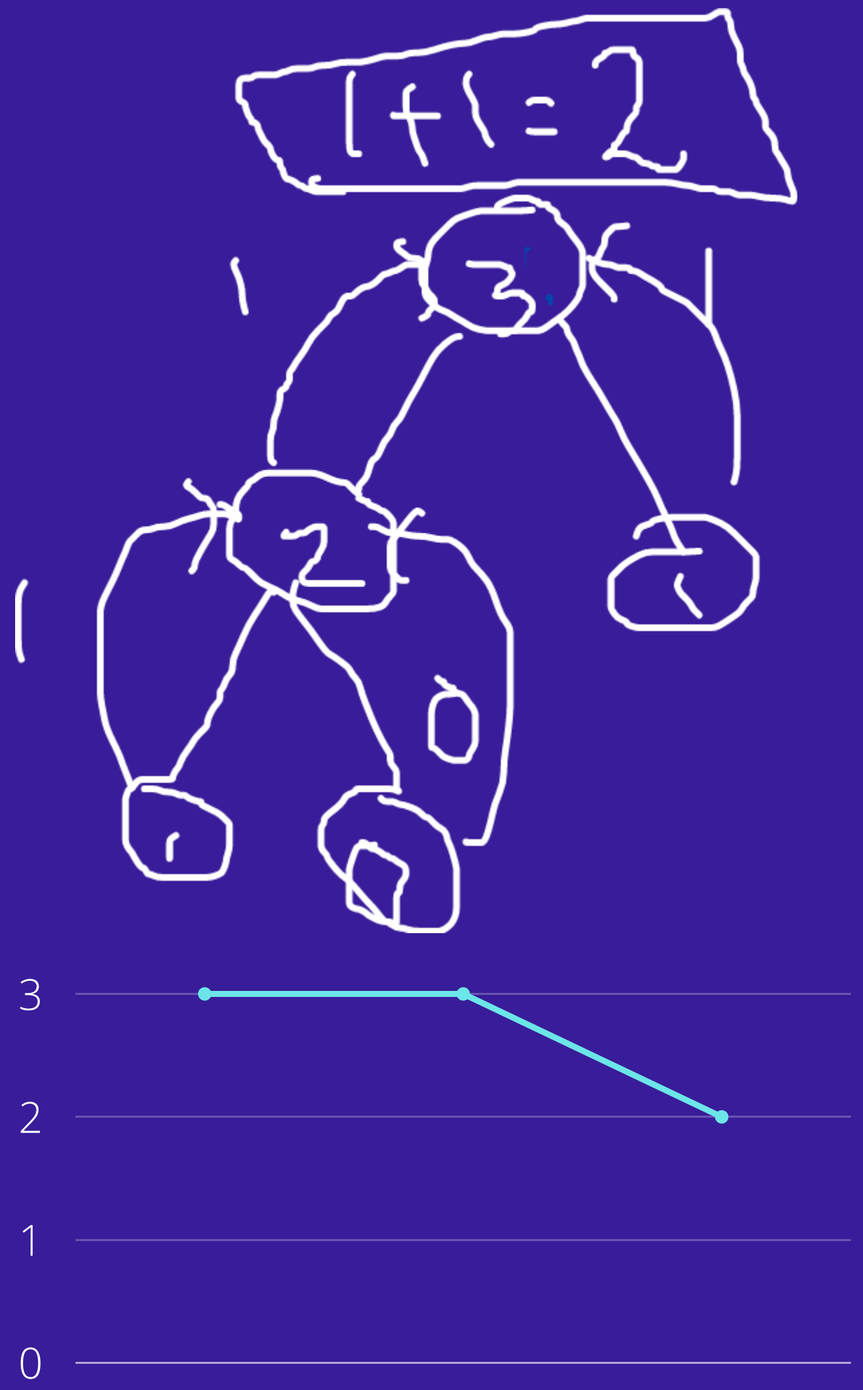
Step 5

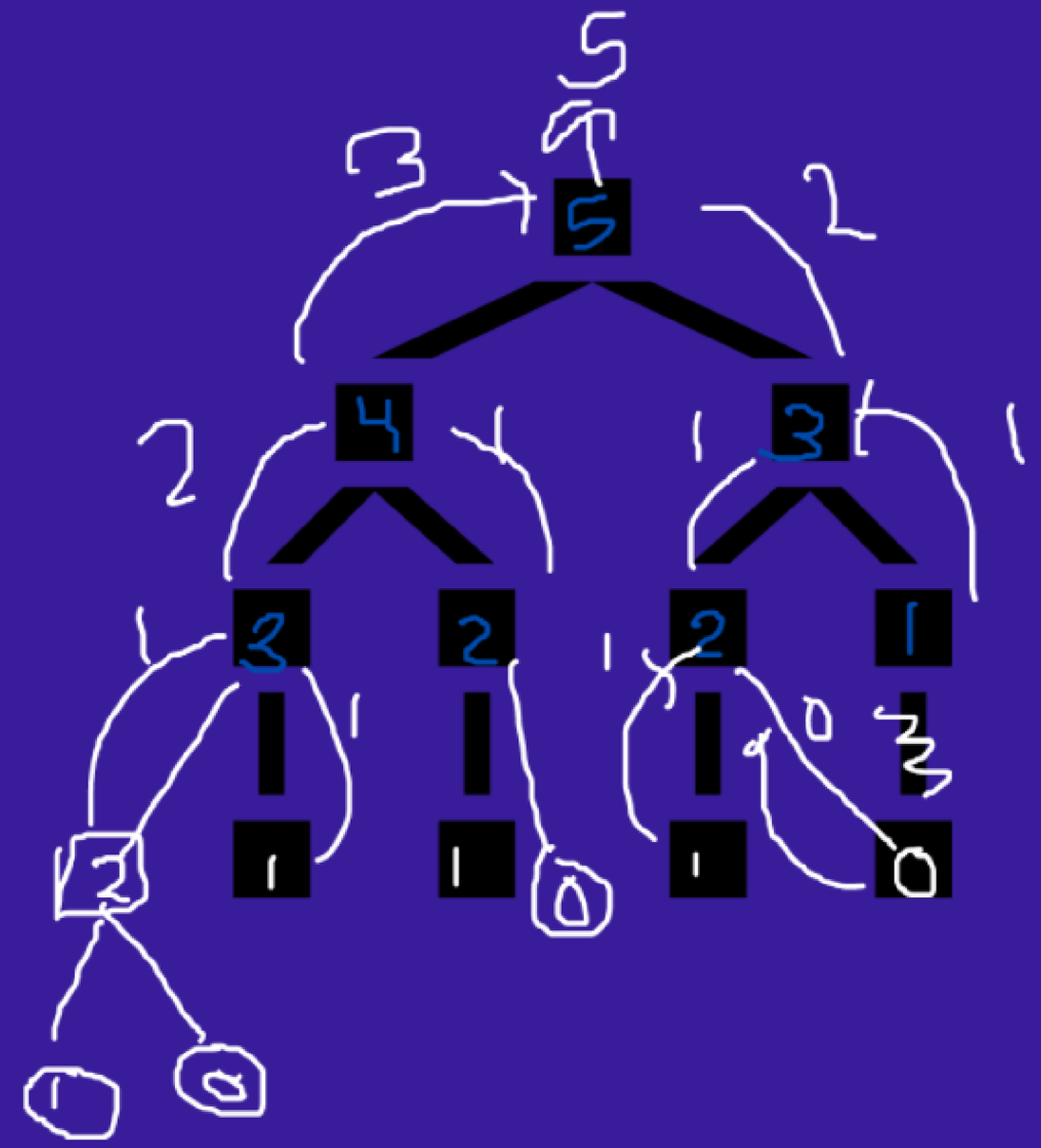
Problem Solved !!!
Next problem

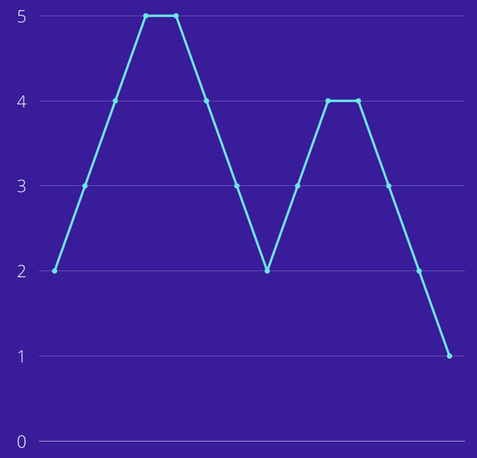
Fibonacci series

```
int fib(int n){
    //base case
    if(n==0 or n==1)
        return n;
    //recursive case
    return fib(n-1)+fib(n-2);

}
```







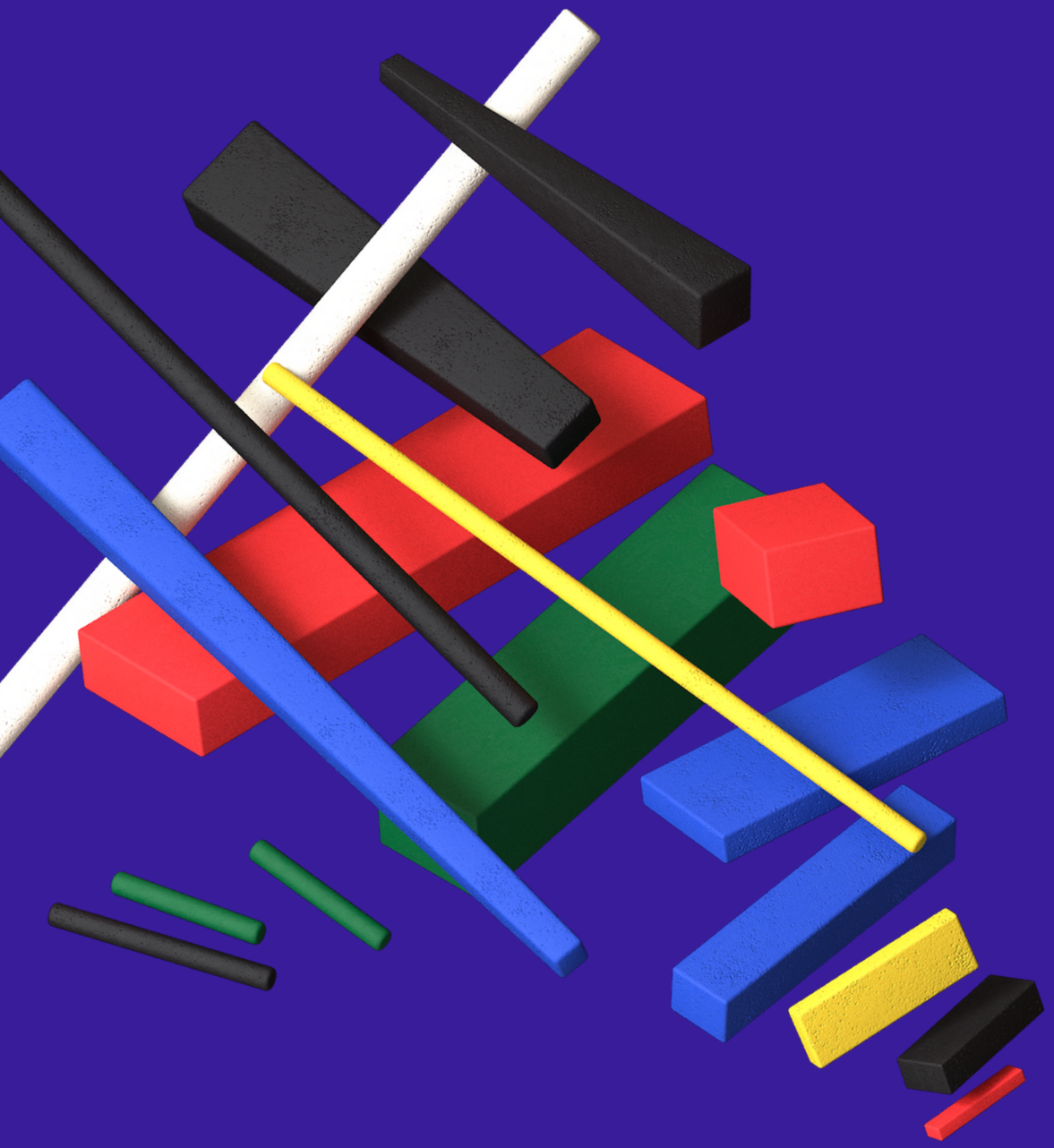
max elements in call stack = 5 ~ O(N)

Since the number of nodes in a single level form a gp with r =2

1+2+4+8+16 + ...

= 2^N -1 ~~ 2^N

Time Complexity = O(2^N)

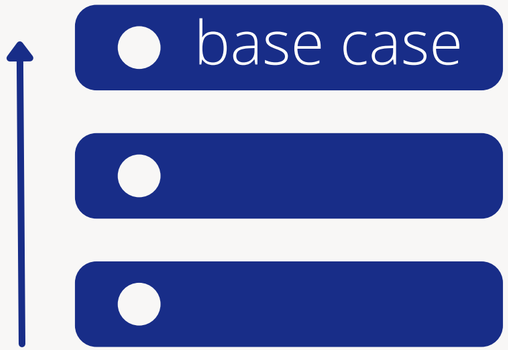


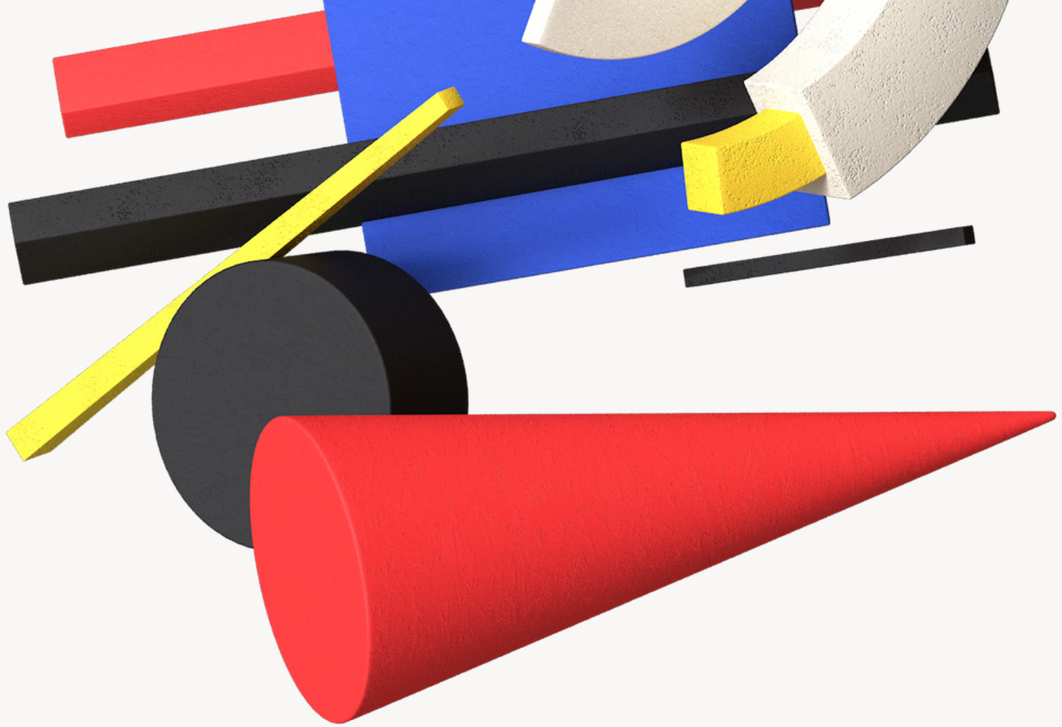
Recursion directions

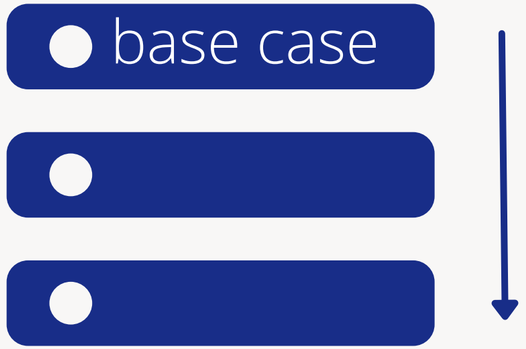
Top down approach

When we go towards the base case while problem being solved as sub problems

i.e computations are made before calling the function







Bottom up apprpach

When we start from base case to the main problem

i.e computations are made after the recursive cases

Top down

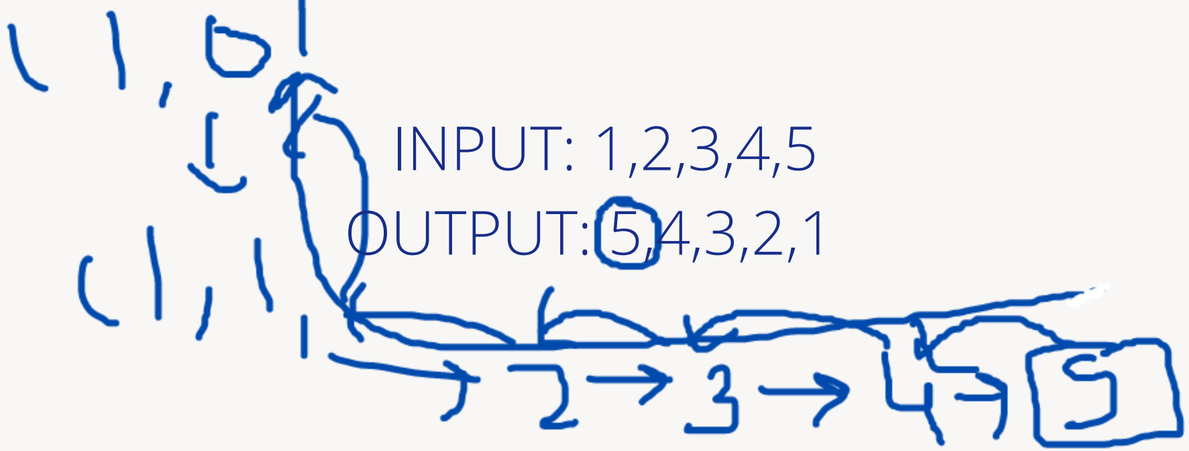
```
void printArray(vector<int> v,int i){
    if(i == v.size())
        return;
    cout<<v[i]<<",";
    printArray(v,i+1);
}
```

INPUT: 1,2,3,4,5

OUTPUT: 1,2,3,4,5

Bottom up

```
void printArray(vector<int> v,int i){
    if(i == v.size())
        return;
    // cout<<v[i]<<",";
    printArray(v,i+1);
    cout<<v[i]<<",";
}
```



INPUT: 1,2,3,4,5

OUTPUT: 5,4,3,2,1

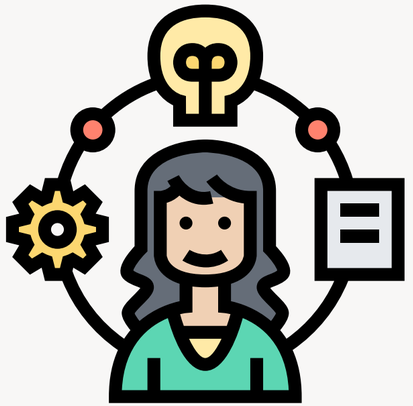
PROBLEM :

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.
```

THINGS TO KEEP IN MIND:



- THE RECURSION PROBLEM MUST HAVE A BASE CASE
- IF BASE CASE IS ENCOUNTERED EXIT RECURSION CALL CHAIN
- ELSE CALL THE RECURSIVE CASE WITH CHANGE IN PARAMETERS

PROS OF RECURSION

- **RECURSION CAN REDUCE TIME COMPLEXITY.**
- **THE CODE MAY BE EASIER TO WRITE.**
- **RECURSION IS BETTER AT TREE/GRAPH TRAVERSAL.**

CONS OF RECURSION

- **RECURSION USES MORE MEMORY.**
- **THE COMPUTER MAY RUN OUT OF MEMORY IF THE RECURSIVE CALLS ARE NOT PROPERLY CHECKED.**
- **VERY LOGICAL SO IT IS DIFFICULT TO TRACE**

Do you have any questions?

Send it to us! Or ask right now if any

Next Topic : BACKTRACKING

INTRESTING LINKS

RECURSION

RECURSION AND ITERATION

ITERATIVE TO RECURSIVE

PROBLEMS

Josephus problem

LUCKY NUMBER

Tower of Hanoi

String permutations

Design patterns