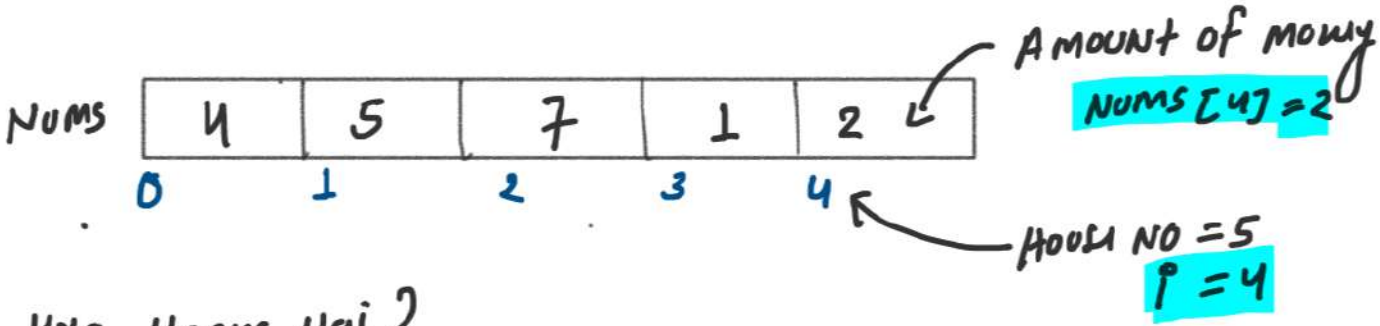
22/10/2023

RECURSION MARATHON
EXTRA CLASS - 24

1. Maximum sum of non - adjacent element (House Robber - I on Leetcode-198)

Ex:1



Find kya karna hai?

↳ Return the maximum amount of money

Case:1
Chori karni wala ghar

- ✓1th
- ✓2th
- ✓4th
- ~~3th~~
- ~~3th~~ ~~1th~~
- ~~3th~~

amount
4
7
2
Total = 13

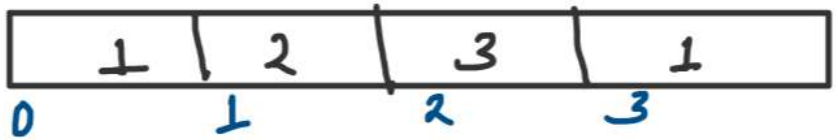
Case:2
Chori na karne wala ghar

- ✓1th ✗0th ✗2th
- ✓3th ✗2th ✗4th

Amount
5
1
Total = 6

maxAmt = 13 output

Example: 02



Case-1

Chori Kare wala Ghar — Amount

✓ 0th — ✗ 1th — 1

✓ 2th — ✗ 4th — ✗ 3th — 3

4

Case-2

Chori Na Karne wala Ghar — Amount

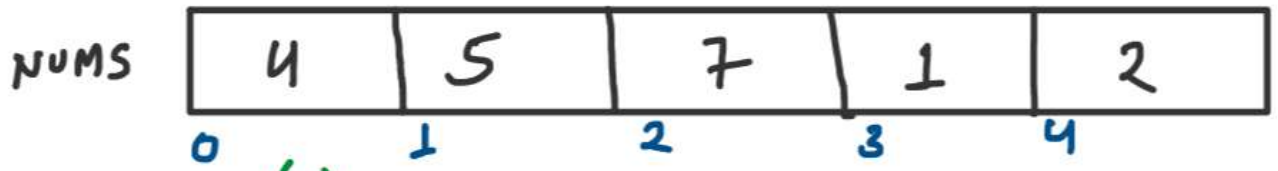
✓ 1th — ✗ 0th — ✗ 2th — 2

✓ 3th — ✗ 2th — 1

3

maxAmount = 4

output



Case:1

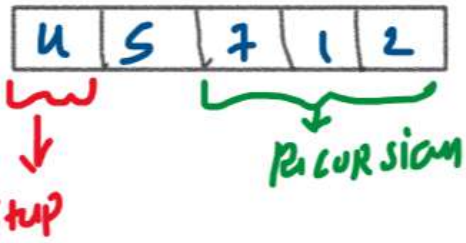
ith char me chori karunga

OR

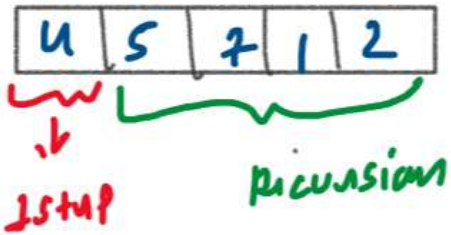
Case:2

0th char ni chori nahi karunga

$4 + f(i+2, N-1)$

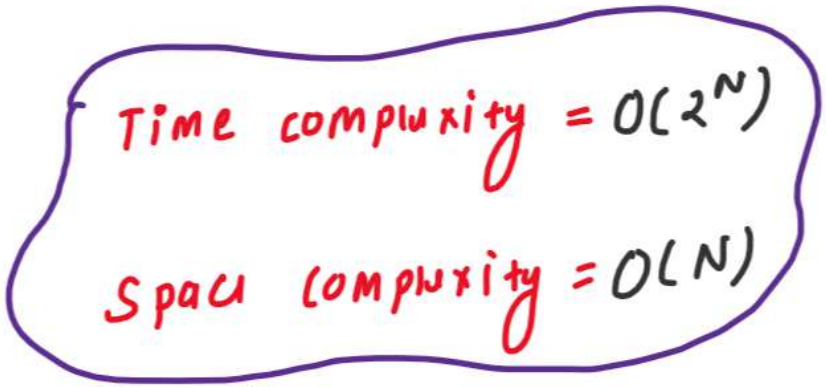


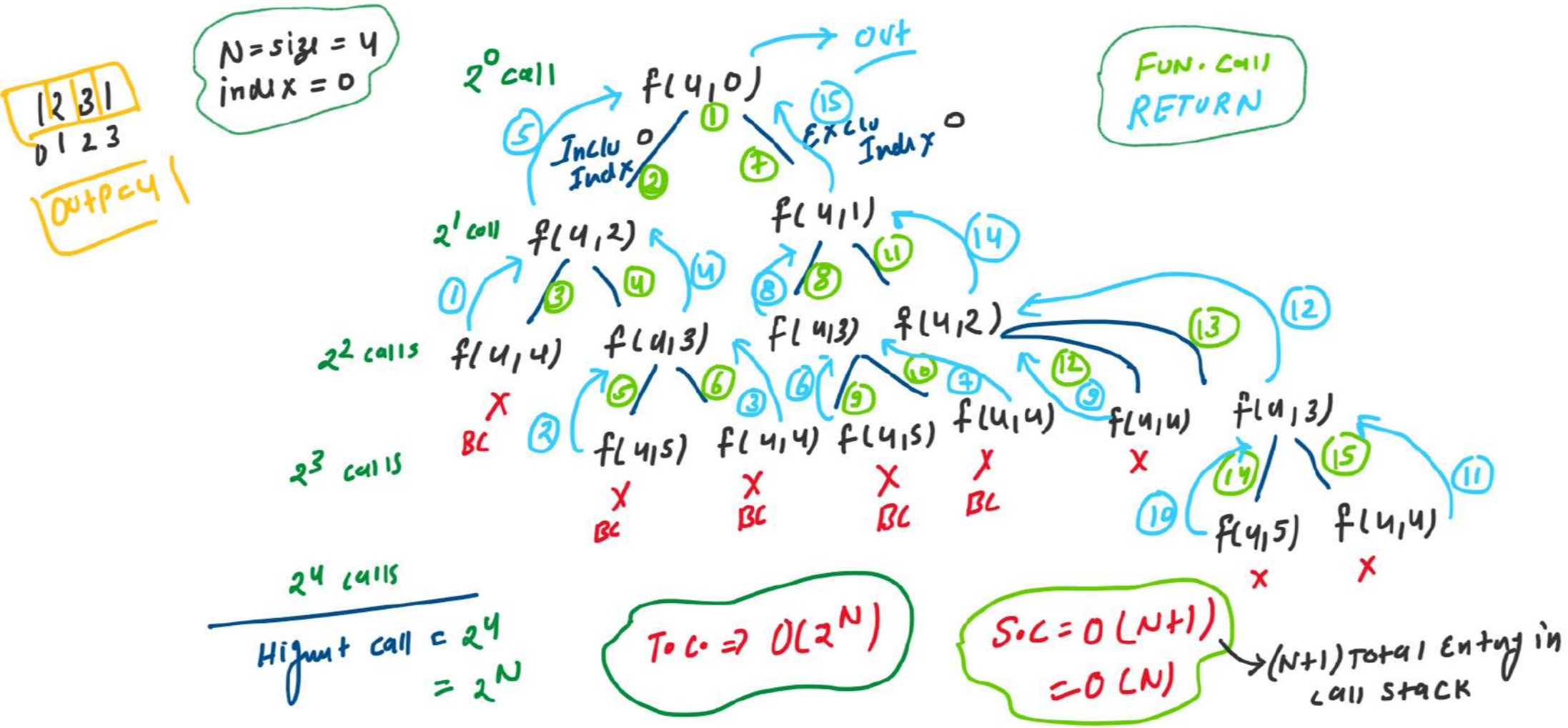
$0 + f(i+1, N-1)$



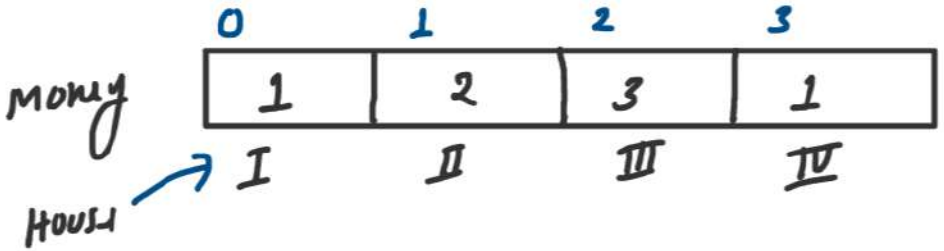
} maxAmount ⇒ ?

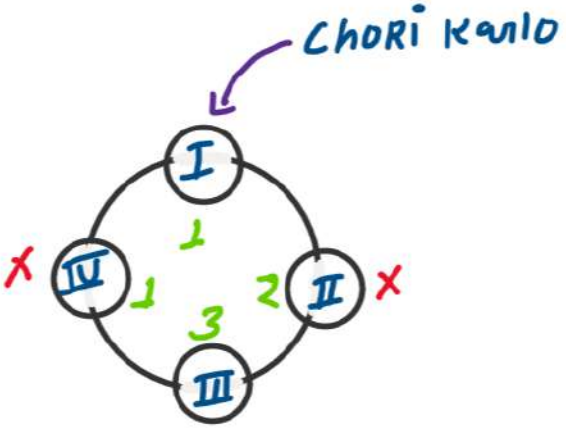
```cpp
// ✅Program 06: House Robber (Leetcode-198)
class Solution {
public:
    int solve(vector<int>& nums, int size, int index){
        // Base Case
        if(index >= size){
            return 0;
        }

        // Chori karlo --> ith index par
        int option1 = nums[index] + solve(nums, size, index + 2);

        // Chori mat karo --> ith index par
        int option2 = 0 + solve(nums, size, index + 1);

        // return the Maximum Amount
        int finalAns = max(option1, option2);
        return finalAns;
    }
    int rob(vector<int>& nums) {
        int size = nums.size();
        int index = 0;
        int ans = solve(nums, size, index);
        return ans;
    }
};
```




1 2 3 1
0 1 2 3
Output
N = size = 4
index = 0
2^0 call
f(4,0)
out
Inclu Index 0
Exclu Index 0
FUN. call RETURN
2^1 call
f(4,2)
f(4,1)
2^2 calls
f(4,4)
f(4,3)
f(4,3)
f(4,2)
BC
2^3 calls
f(4,5)
f(4,4)
f(4,5)
f(4,4)
f(4,4)
f(4,3)
f(4,5)
f(4,4)
2^4 calls
Highest call = 2^4 = 2^N
T.C. => O(2^N)
S.C = O(N+1) = O(N)
(N+1) Total Entry in call stack

2. House Robber - II on Leetcode-213

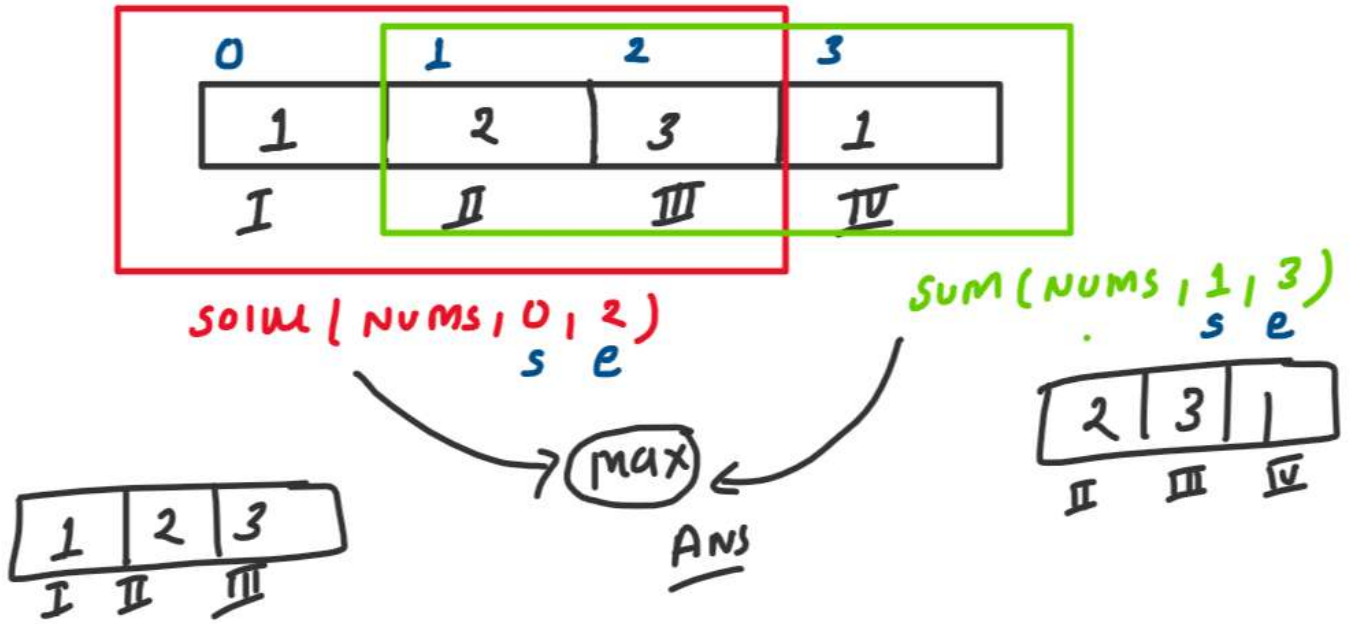




Note:
All houses at this place are arranged in a circle.
That means the first house is the neighbour of the last one.

Find kya karna hai: *Return maximum amount of money you can rob tonight*

Like as
House Robbing-I



```cpp
class Solution {
public:
    int solve(vector<int>& nums, int s, int e){
        // Base Case
        if(s > e){
            return 0;
        }

        // Robbering Karlo --> ith index par
        int opt1 = nums[s] + solve(nums, s+2, e);

        // Robbering mat Karo --> ith index par
        int opt2 = 0 + solve(nums, s+1, e);

        // Maximum amount of money
        int maxAmount = max(opt1, opt2);

        return maxAmount;

    }
    int rob(vector<int>& nums) {
        int n = nums.size();

        // Single element --> yanha par maine galti ki thi
        if(n == 1){
            return nums[0];
        }

        int opt1 = solve(nums, 0, n-2);
        int opt2 = solve(nums, 1, n-1);
        int ans = max(opt1, opt2);

        return ans;
    }
};
```

$T.C. = O(2^N)$

$S.C. = O(N)$

where N is size of Array

📂 3. Count Derangements on GFG

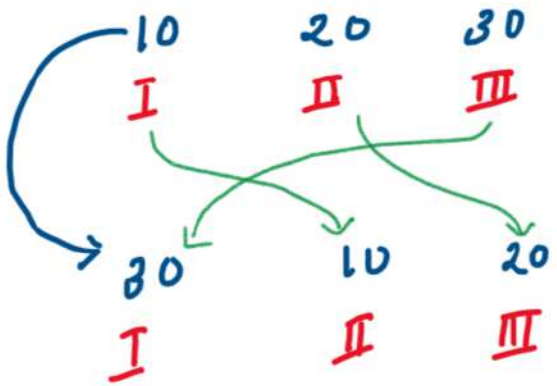
Permutation such that no element appears in its original position

Example 01:
Input: n = 2
Output: 1
For two elements say {0, 1}, there is only one possible derangement {1, 0}

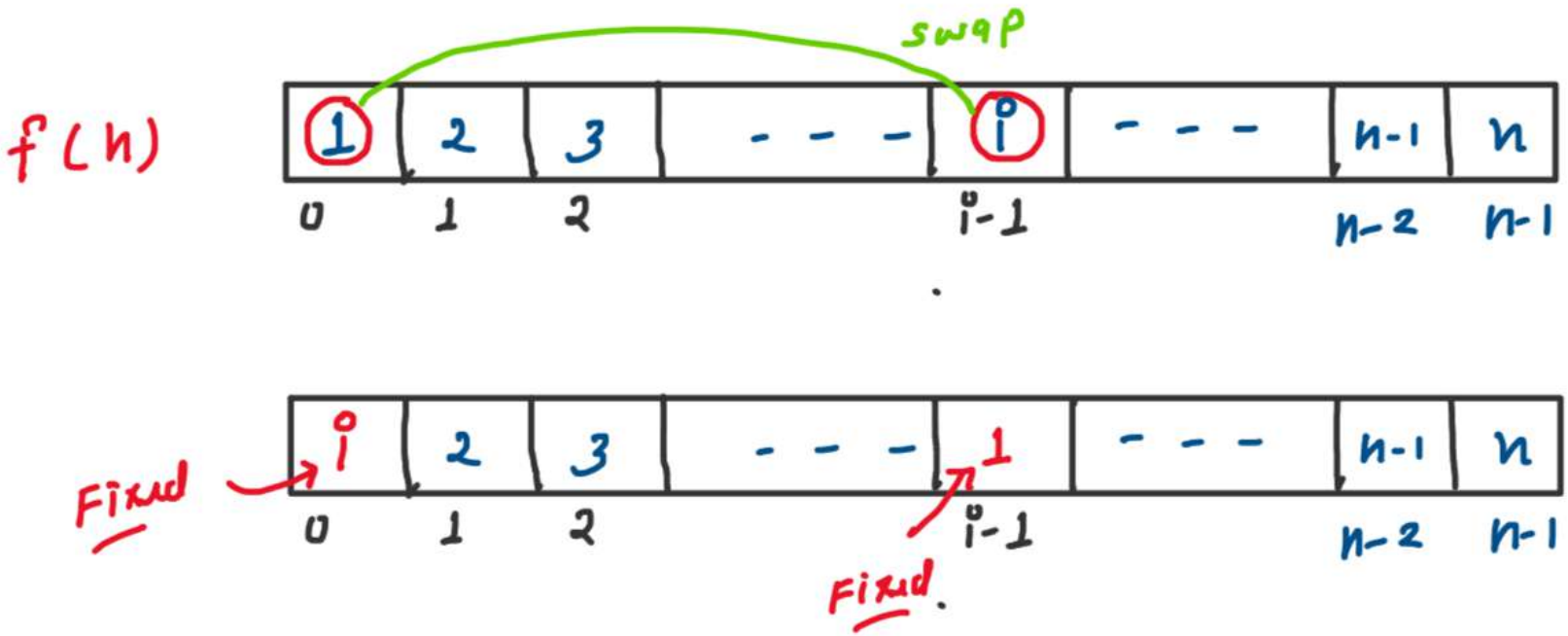
Example 02:
Input: n = 3
Output: 2
For three elements say {0, 1, 2}, there are two possible derangements {2, 0, 1} and {1, 2, 0}

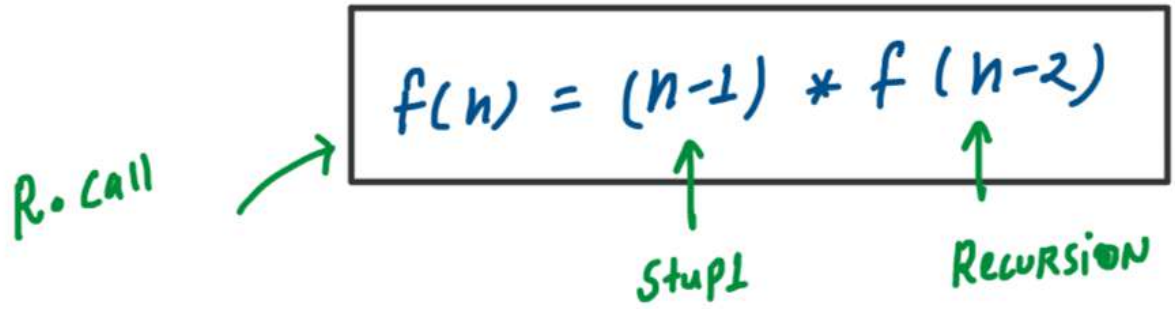
Example 03:
Input: n = 4
Output: 9

· what is
Derangements



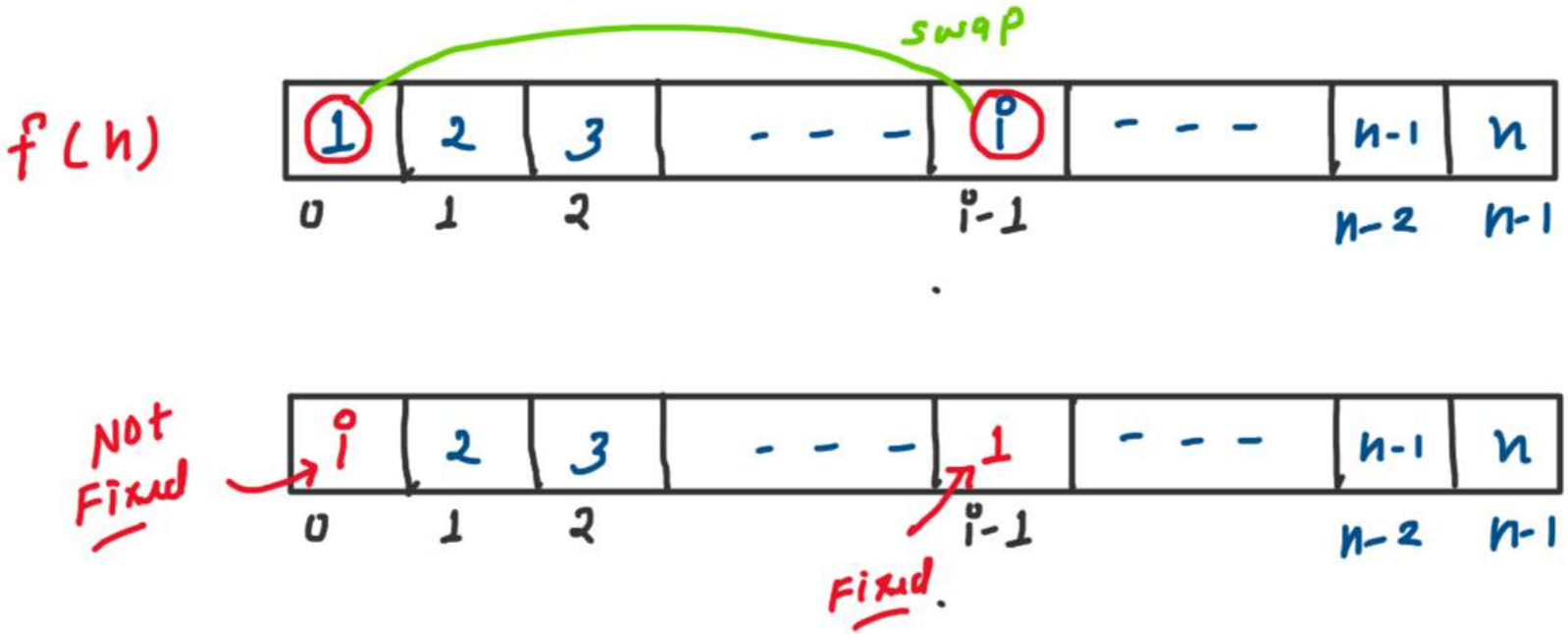
no element appears in its original position

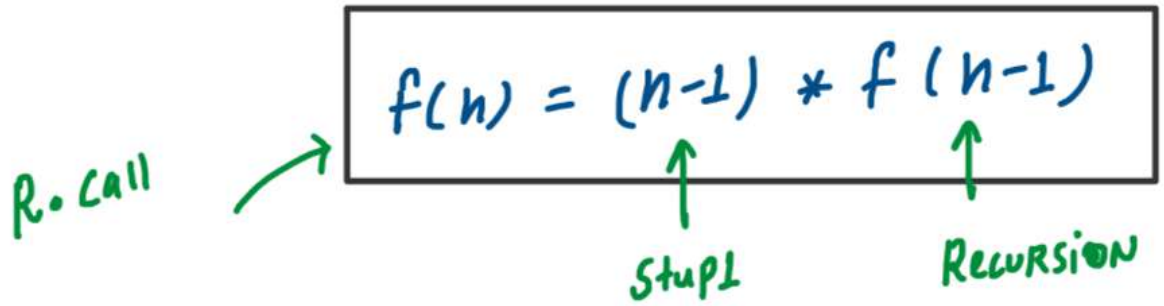




CASE-I

When we swap 1 & i and we consider 1 and i's position are fixed.





CASE-II

When we swap 1 & i and we consider 1's position is fixed and i's position is not fixed.

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

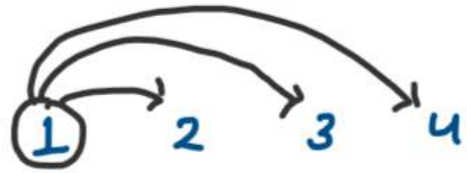
$$= (n-1) * [f(n-2) + f(n-1)]$$

R. Relation

Ex $N=4$ $\{1 \quad 2 \quad 3 \quad 4\}$

Fixed

Not Fixed



Now 1 ko swap karne ki 3 possibility hai
when n = 4

$f(4) = 3 * f(2)$

$f(2) = 1$ BS ✗

$\Rightarrow 3 \times 1$

$\Rightarrow 3$

$3+6$

9

$f(4) = 3 * f(3)$

$f(3) = 2 * f(2)$

$f(2) = 1 * f(1)$ ✗ BS

$\Rightarrow 3 \times 2 \times 1$

$\Rightarrow 6$

```cpp
#include<iostream>
using namespace std;

int solve(int n){
    // Base case
    if(n == 1){
        return 0;
    }
    if(n == 2){
        return 1;
    }

    // Recursive call
    int ans = (n-1) * (solve(n-1) + solve(n-2));
    return ans;
}

int main(){
    int n = 4;
    cout<< solve(n) << endl;

    return 0;
}
```