Agenda → Problems (Medium) on recursion

Solve Recursively → (code + intuition)

Pre-requisite → Basic recursion knowledge (factorial / fibonacci)

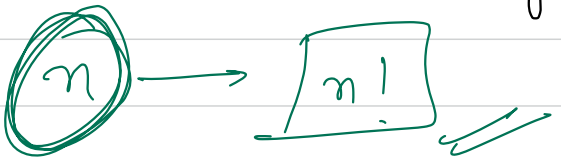
Q-1) Given an array of non-negative integers. The array is "special" if for every pair of adjacent elements, their sum is a perfect square.

$n \leq 10$

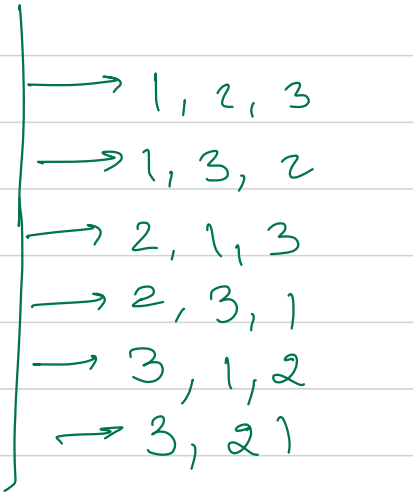
Return the no. of permutations of A that are special. Two permutation p_1 & p_2 differ if and only if for some i we have $p_1[i] != p_2[i]$

Ex → [1,17,8] ans = 2 [1,8,17] [17,8,1]

Brute force → to generate all permutations, & then check if they are special.



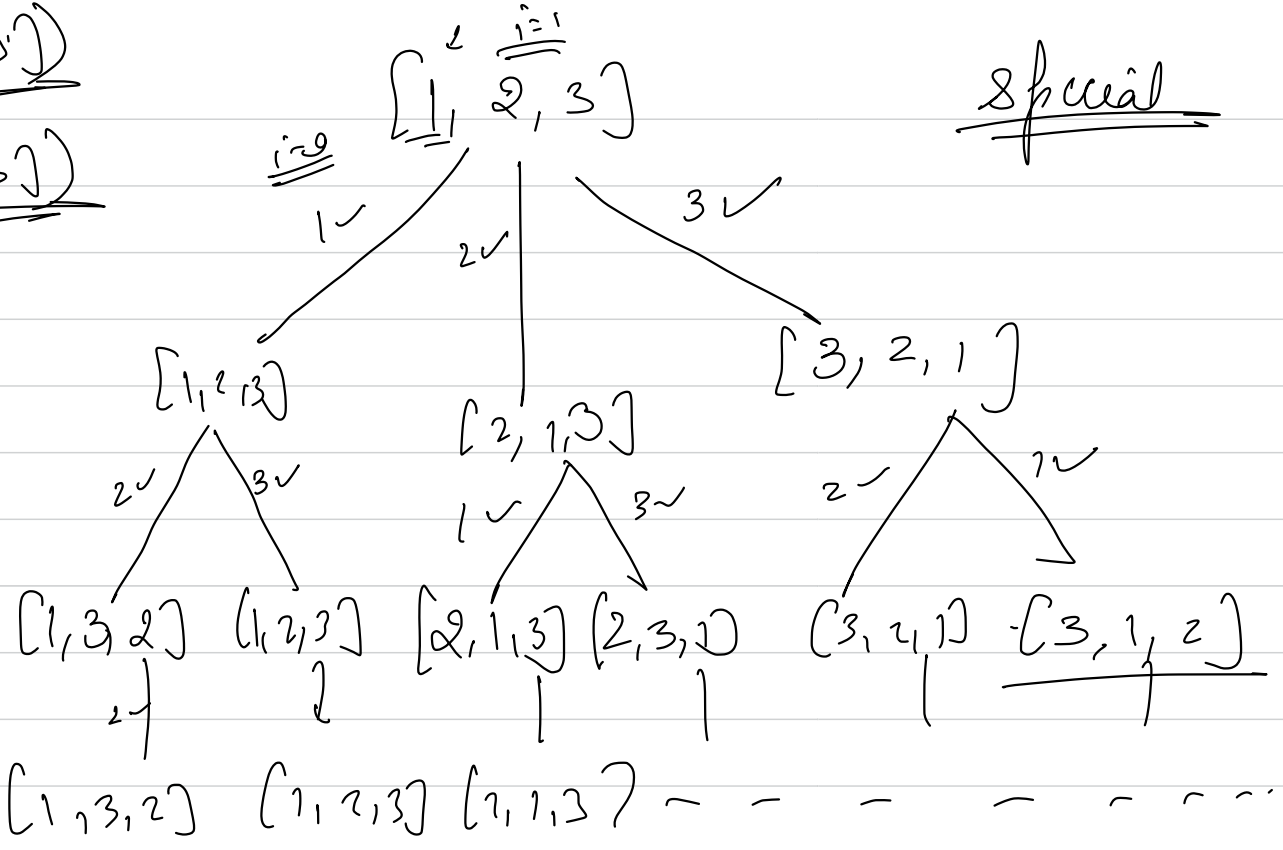
⇒ [1, 2, 3]



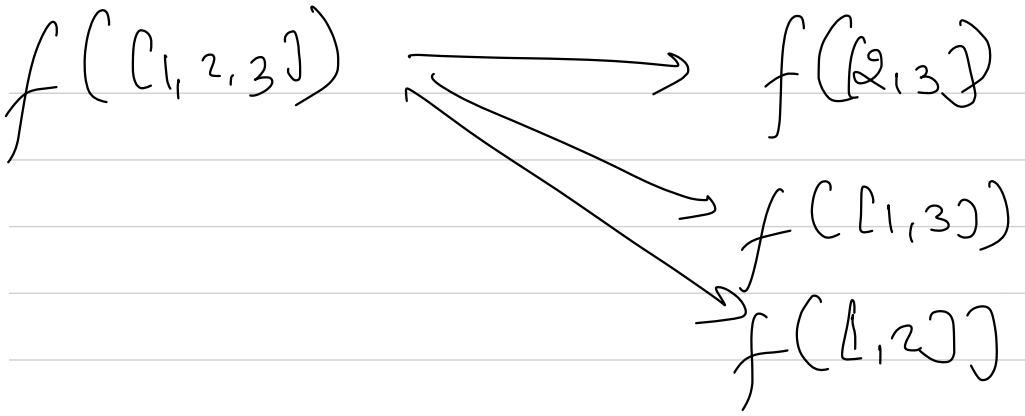
3 → Base Case
Self work
Recursive Intuition

f([1,2,3])

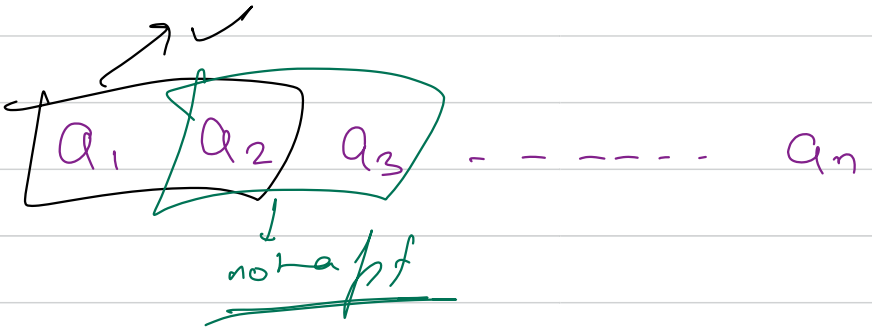
f([2,3])

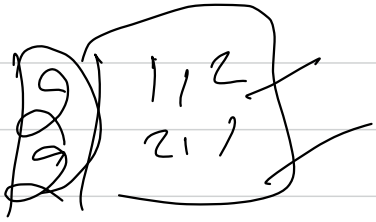


special

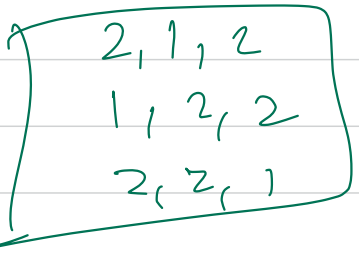

f([1,2,3])
f((2,3])
f([1,3))
f([1,2])

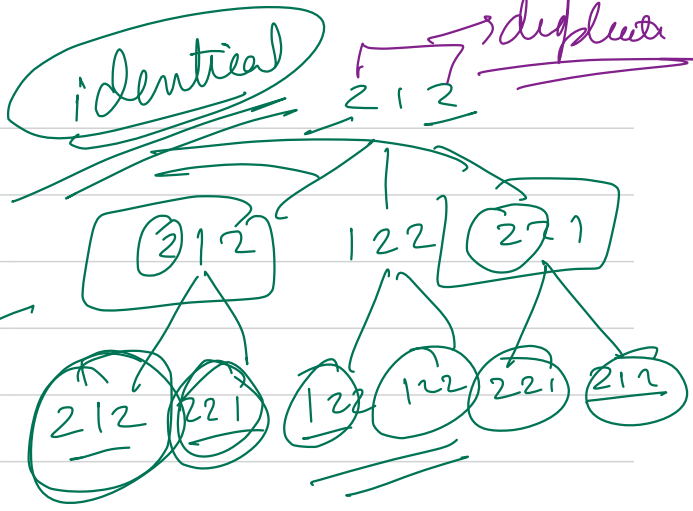
Let's say a[i], a[i+1] don't make a perfect square, the array's permutation can never be spiral.

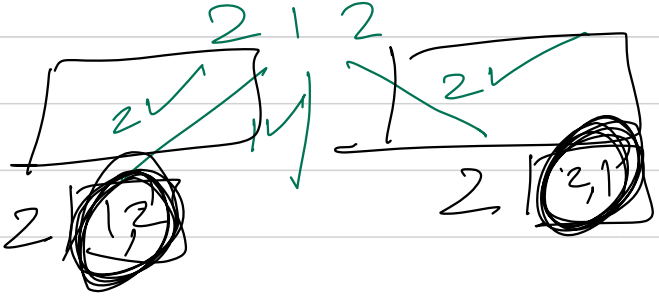



1, 2
2, 1

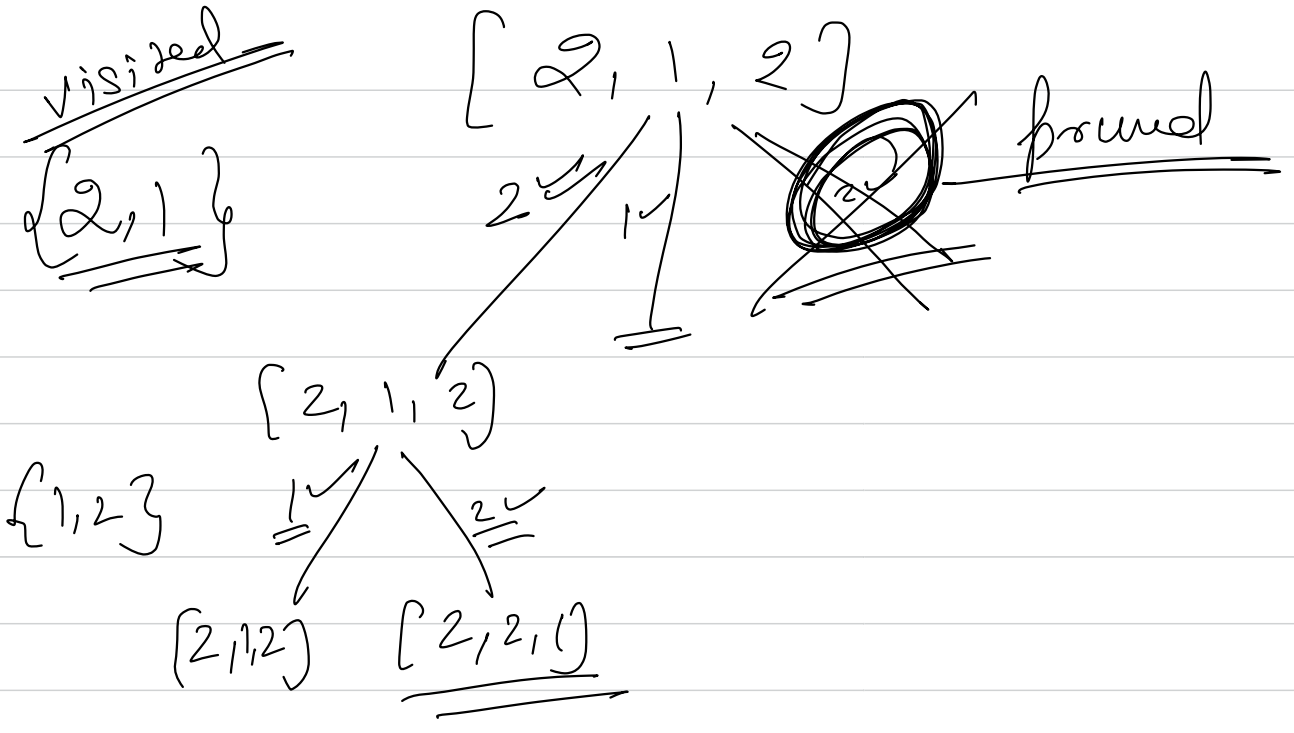
[2, 1, 2]

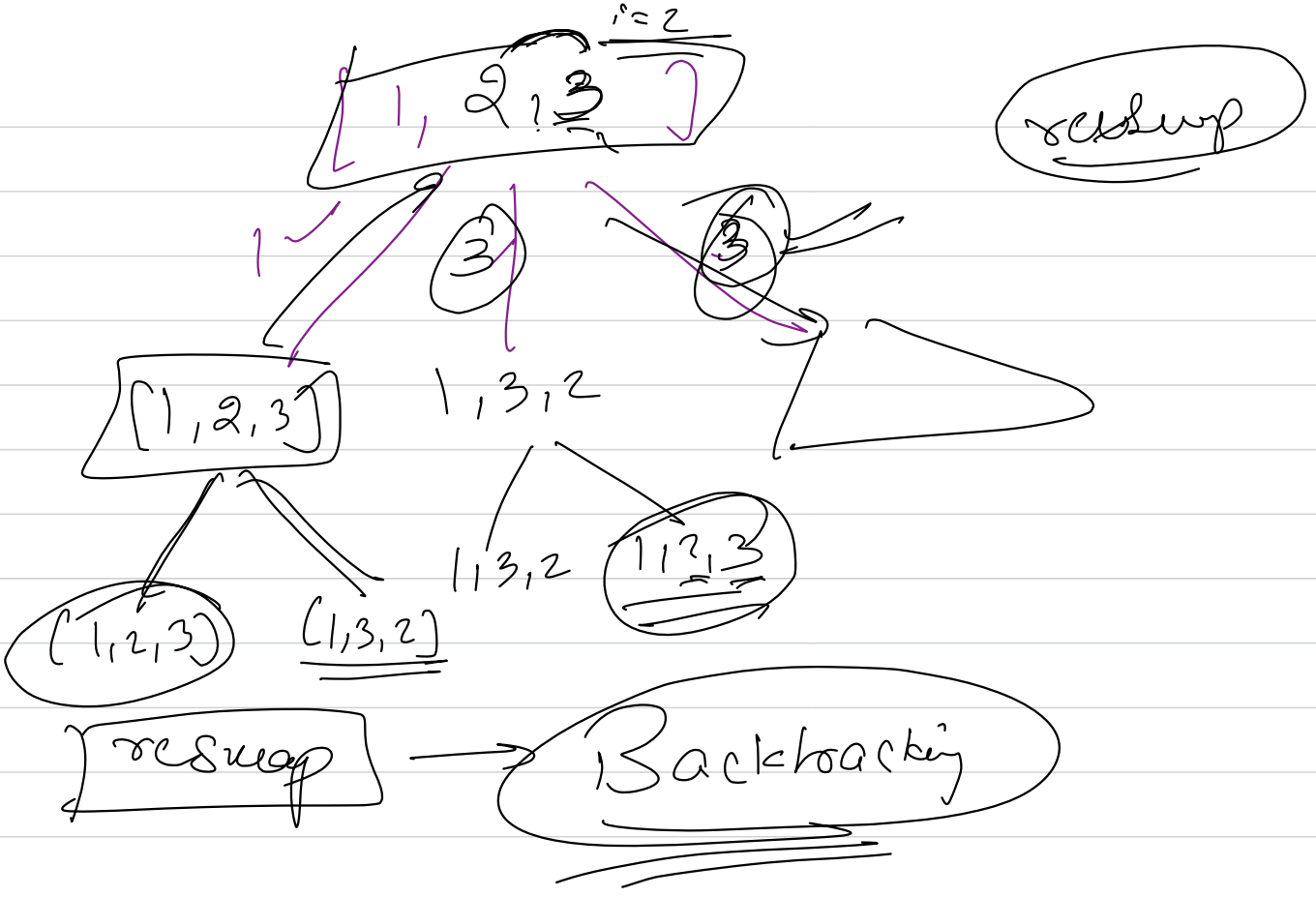

2, 1, 2
1, 2, 2
2, 2, 1

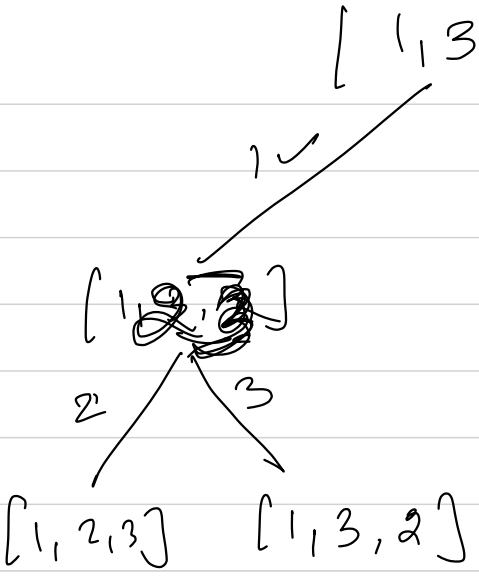

identical
duplicate
2 1 2
2 1 2
1 2 2
2 2 1
2 1 2
2 2 1
1 2 2
1 2 2
2 2 1
2 1 2


2 1 2
2
2
2 1 2
2
2 2, 1

X


visited
{2,1}
[2,1,2]
2
1
found
[2,1,2]
{1,2}
1
2
[2,1,2]
[2,2,1]


i = 2
[1, 2, 3]
reswap
1
3
3
[1,2,3]
1,3,2
1,3,2
1,2,3
(1,2,3)
[1,3,2]
reswap
Backtracking


[1,3,2]
1
[1,2,3]
2
3
[1,2,3]
[1,3,2]

m = 1 n = 1

ans = 9

0	0	0
0	0	0
0	0	0

3x3 grid

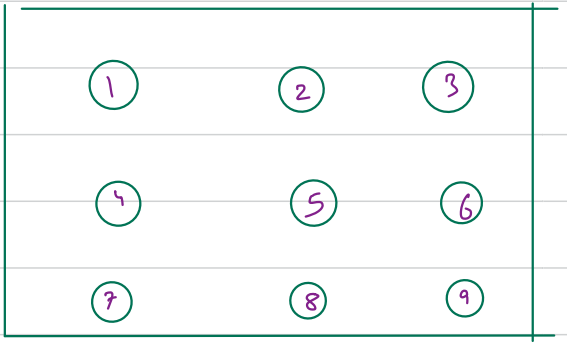
Given 2 integers m and n, return the unique & valid unlock patterns of android grid that consist of atleast m cells & atmost n cells.

→ valid pattern

(1) All the dots in the sequence are distinct.

(2) The line segment connecting 2 consecutive dots in the sequence passes through other dots. & the other dots must have previously appeared.

one chosen cell == 1



m = 4 n = 4

x = 4

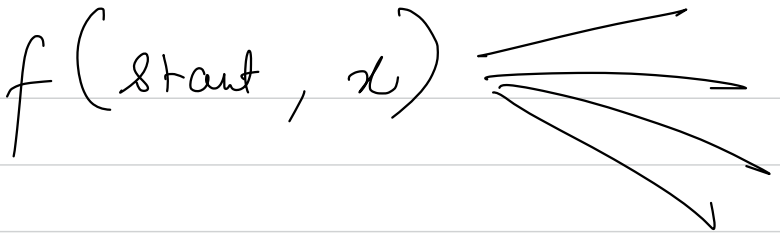
x = 5

1st observation

f(2, 5)
f(5, 4) → f(4, 3)
f(3, 2)

f → no. of passwords starting from 1 x 4
" " " " 2 x 4
" " " " 5

3) Base Case
Self work
Recursion



Skip table → skip [1] [3] = skip [3] [1] = 2

1	7	7	1
3	9	9	3
7	9	9	7
.		.	
.		.	
.		1	
		1	
		1	

Q. friends pairing problem.

n friends who want to go to party.

1 person can either go alone in the party or can go in a pair. Find the no. of way n friends can go.

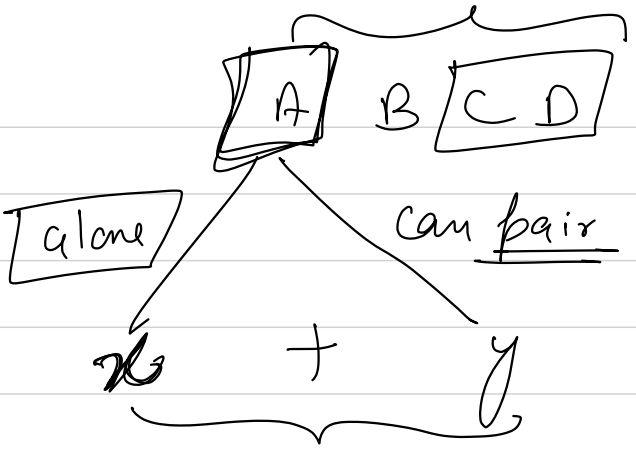
$n = 3$

(A) (B) (C) → 1
(A B) (C) → 1
(A) (B C) → 1
(A C) (B) → 1

} → 4

$f(n)$

f that counts no. of way for n friend



$x = f(n-1)$

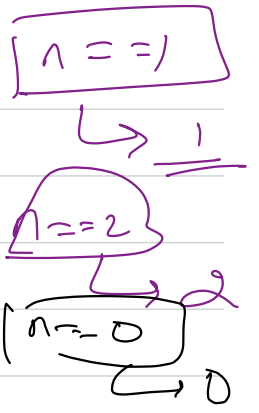
y = no. of way to x $f(n-2)$ make a pair

$= (n-1) \times f(n-2)$

(A) (B) (C) (D)
(A) (B) (D)
(A) (BD) (C)
(A) (B) (DC)

(AB) (C) (D)
(AB) (CD)
(AC) (B) (D)
(AC) (B) (D)
(AD) (B) (C)
(AD) (BC)

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




```
int f(n) {
    if (n==1 or n==2 or n==0)
        return n;
    return f(n-1) + (n-1)*f(n-2);
}
```