Q => Given an array of integers, try to sort the array in asc order using recursive bubble sort.

[4, 3, 2, 1]

↓

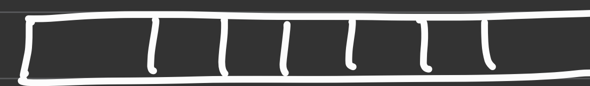
[1, 2, 3, 4]

void sort(std::vector<int> &v)

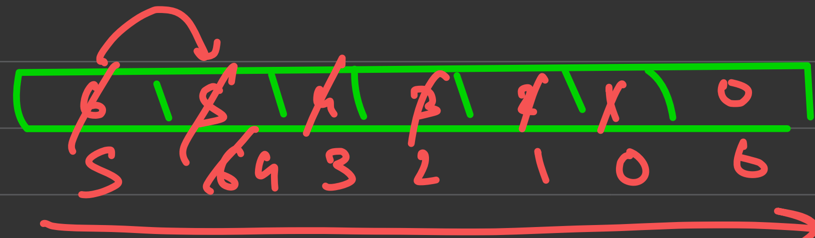
Base Case

$\boxed{x} \rightarrow$ single elem it is already sorted

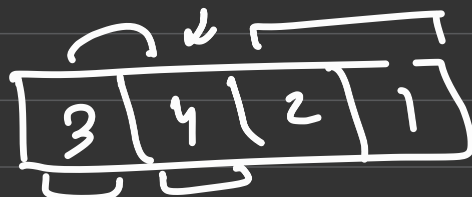
Recursive Assumption



Self work



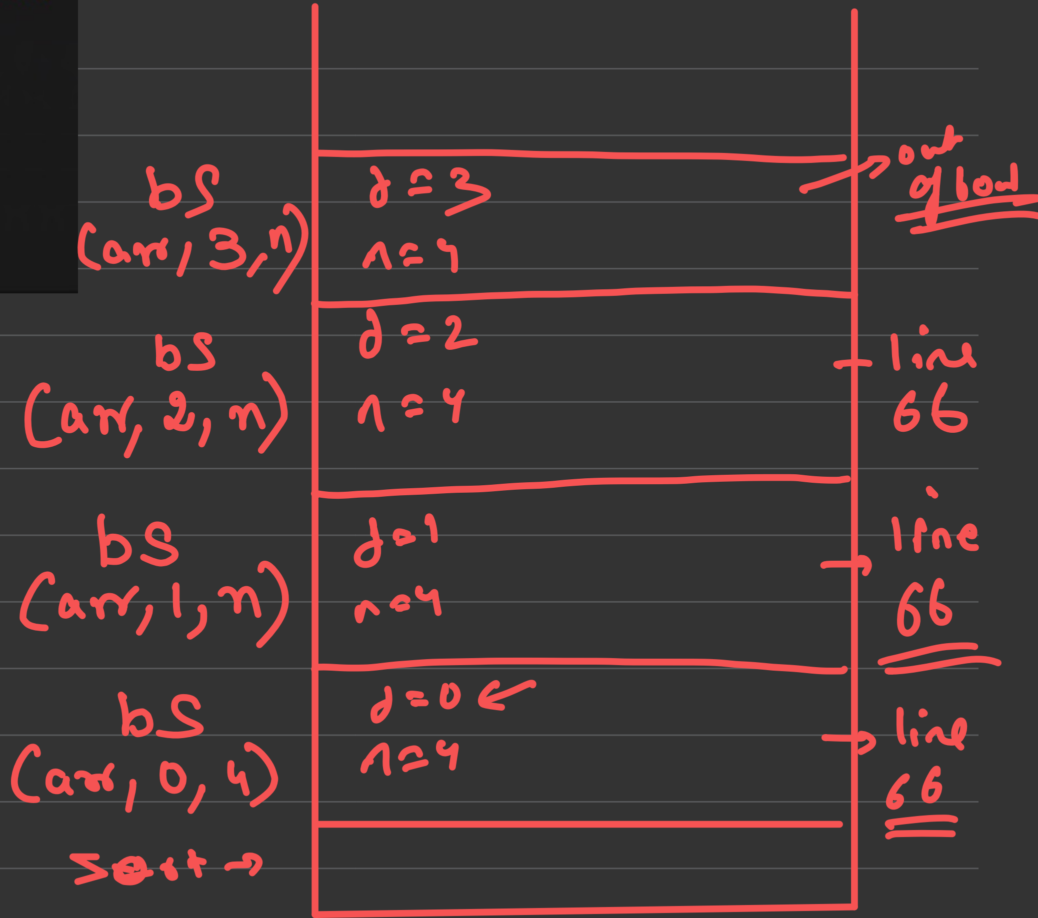
In one iteration



it will move the largest element to the end of the array

```cpp
57  void bubblesort(std::vector<int> &arr, int j, int n) {
58      // base case
59      if(n == 1) {
60          return;
61      }
62
63      if(arr[j] > arr[j+1]) { // error
64          std::swap(arr[j], arr[j+1]);
65      }
66      bubblesort(arr, j+1, n);
67  }
```

[3, 2, 1, 4]



```cpp
void bubblesort(std::vector<int> &arr, int j, int n) {
    // base case
    if(n == 1) {
        return;
    }
    if(j == n-1) {
        bubblesort(arr, 0, n-1);
        return;
    }
    if(arr[j] > arr[j+1]) { // error
        std::swap(arr[j], arr[j+1]);
    }
    bubblesort(arr, j+1, n);
}
```

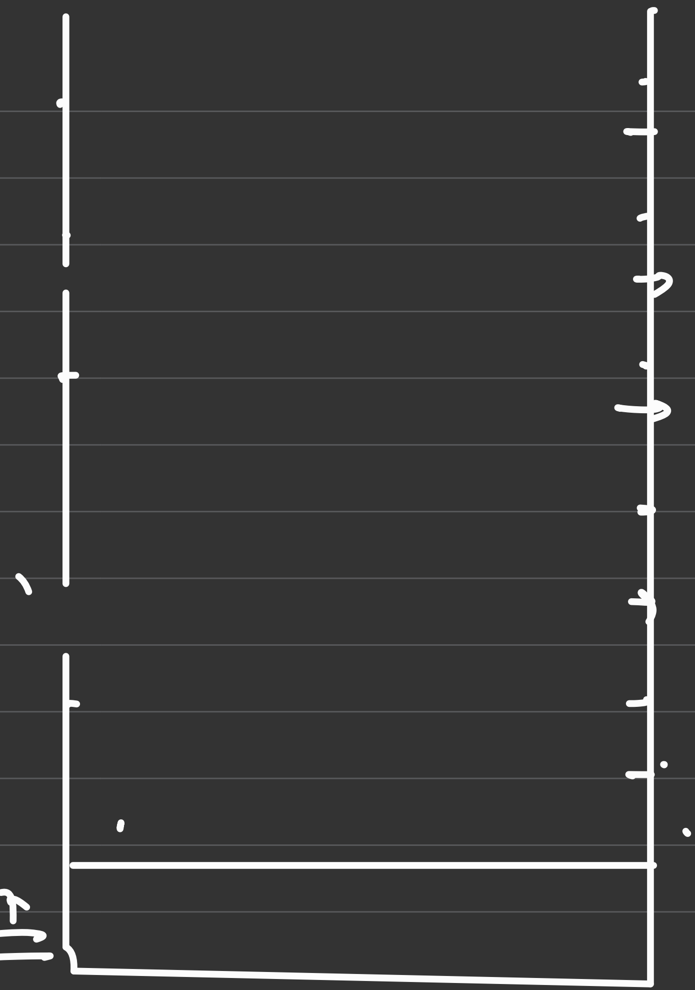
→ restart sorting with less elements

→ further swapping

[1, 2, 3]

3, 4

4, 3



sort

Q:- N friends want to go to a party.

There is one constraint, on each of them,

(1) Either the friend can go alone to the party

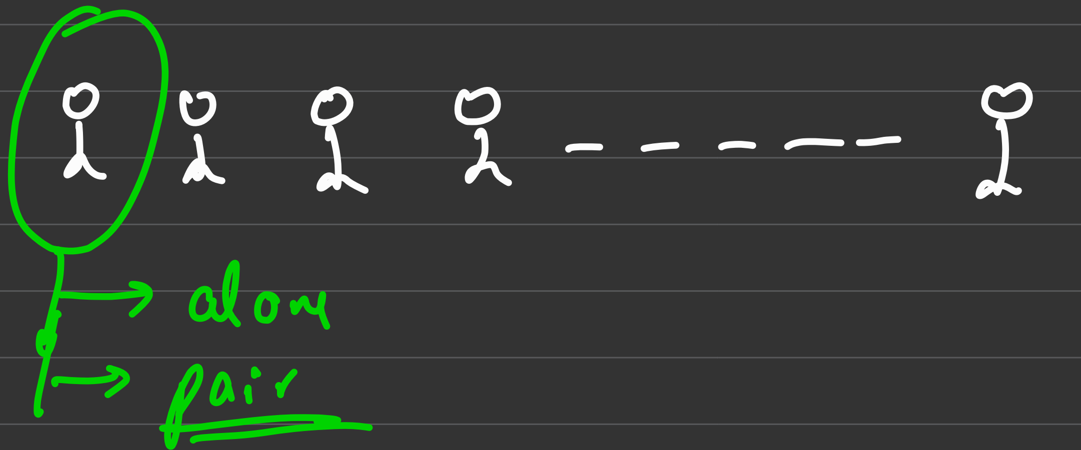
(2) Or they can go in a pair.

Find the total no. of combinations about how they can go??

$n = 3$ → 4 and

(A) (B) (C) (A, B) (C) (A C) (B) (A) (B C)

N → persons (identical)



Base Case → $n = 1$ → 1 → (A)
→ $n = 2$ → 2 → $(A)\ (B),\ (AB)$

Self work

<u>Recursive assumption</u>

A B C D

goalon

(A) (B) (C) (D)

(A) (B C) (D)

(A) (BD) (C)

(A) (CD) (B)

(A B) (C) (D)
(AB) (CD)

(A C) (B) (D)
(AC) (BD)

(A D) (B) (C)
(A D) (BC)

How many pairs possible for A, ??

$\hookrightarrow (N-1)$

$$f(N) = \underbrace{f(N-1)}_{\text{1st person goes alone}} + \underbrace{(N-1) \times f(N-2)}_{\text{1st person makes pair}}$$

$f(N)$ returns the no of ways n friends go to party.

16 + 10

26

A B C D E

(N-1)

N=1 → 1

N=2 → 2

glow

(A) (B) (C)(D)(E)

(A) (B) (CD)(E)

(A) (B) (CE)(D)

(A) (B) (ED)(C)

(A) (BC) (D)(E)

(A) (BC) (DE)

(A) (BD) (C)(E)

(A) (BD) (CE)

(A) (BE) (C)(D)

(A) (BE) (CD)

10

(AB) (C)(D)(E)

(AB) (CD) (E)

(AB) (DE) (C)

(AB) (CE) (D)

(AC)

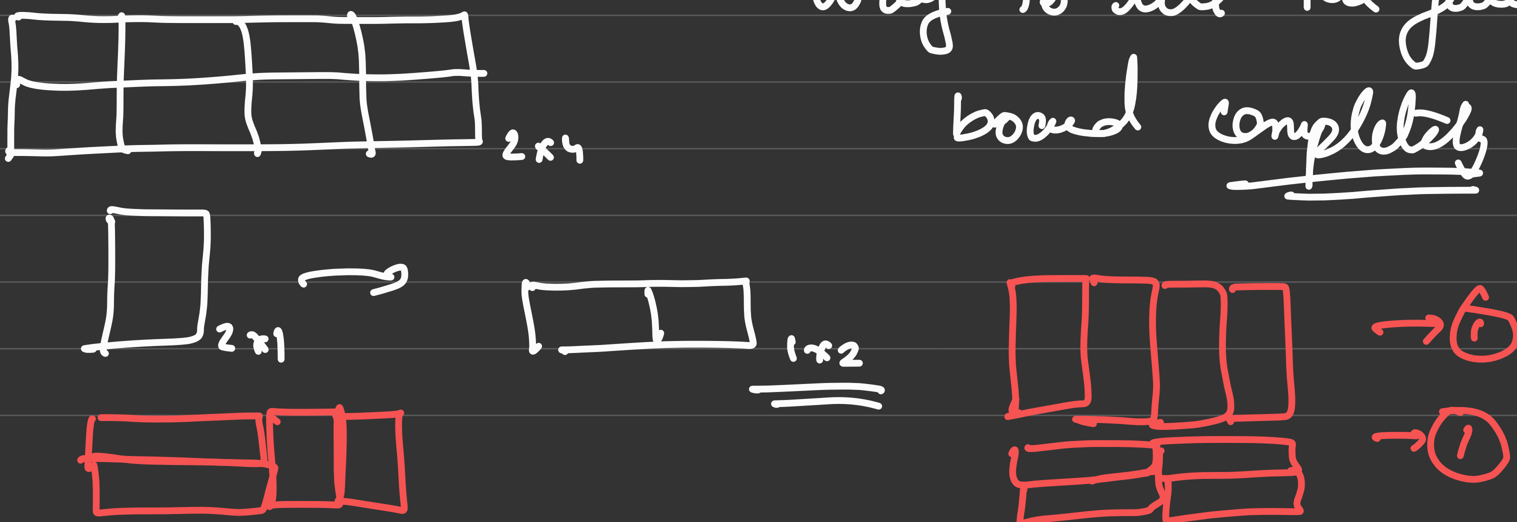
(AD)

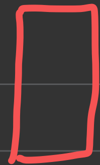
(AE)

16

$O(n)$

Q: You're given a 2×n board, and tiles of size 2×1. You can also rotate it horizontally to make it a 1×2 tile. Count the no. of way to tile the given board completely



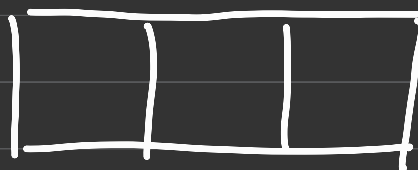
 2x1 Board

N=1 → 1

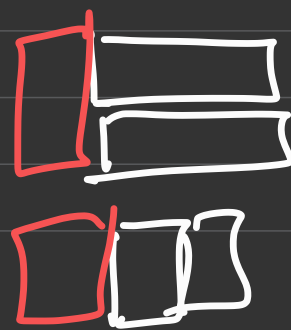
 2x2 Board

N=2 → 2

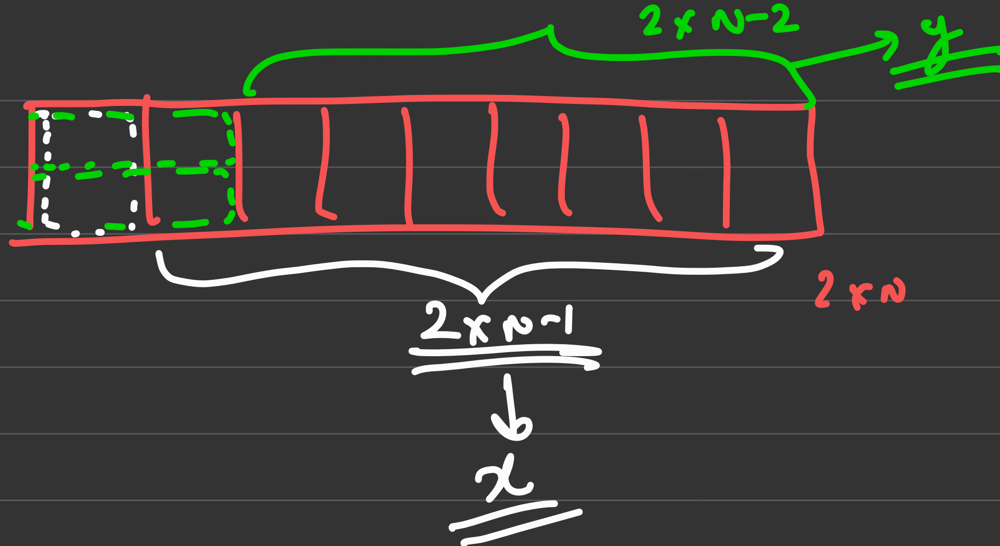
Base Case

 2x3

3

 → 2

1



$2 \times N = x + y$

$$\boxed{f(N) = f(N-1) + f(N-2)}$$

returns no of way to set tile on 2xN board

recurrence

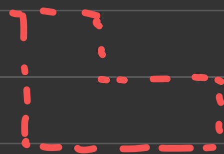
fibonacci

3 + 2 → 5

1
2
3
5
8
...

HW →

2x1



L-shaped tile case

2xn

Q:- Given a no. N, count the no. of Binary Strings (string of 0's & 1's) that donot have consecutive ones.

n = 3 $\longrightarrow$ 000
010
001
100
101

ans $\rightarrow$ 5

$n=1$ 0
1

$n=2 \rightarrow$ 00
01
10

$n=3 \rightarrow$ 000
001
010
100
101

$n=4 \rightarrow$ 0000 0101
0001 1010
0010 1001
0100
1000

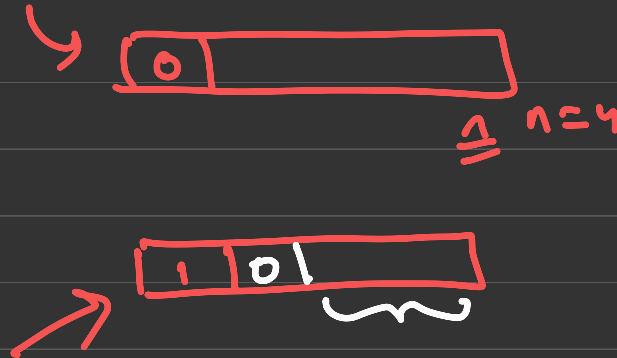
$n=1 \rightarrow 2$
$n=2 \rightarrow 3$

Base Case

Fib

$n=3 \rightarrow 5$

$\rightarrow n=4 \rightarrow 8$



$$f(n) = f(n-1) + f(n-2) \rightarrow \text{fibona}$$