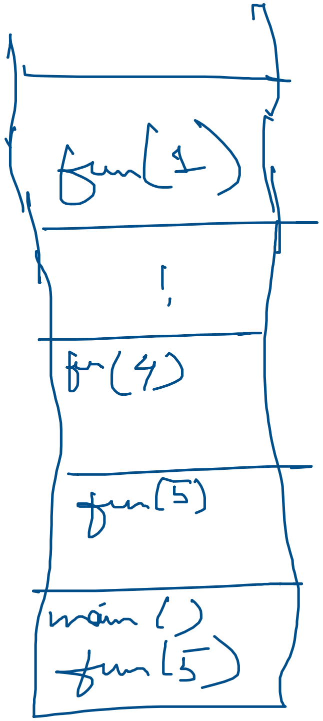
July - 15

1. **Ascending Print Recursion** ✓
2. **Fibonacci** ✓
3. **Binary Search Using recursion** ✓
4. **Sliding Window**
5. **Count Pair Sum**
6. **Reverse a String(If time Permits)**

1



fun(n-1)
S.o.P

1 2 3 4 5

2 Fibonacci

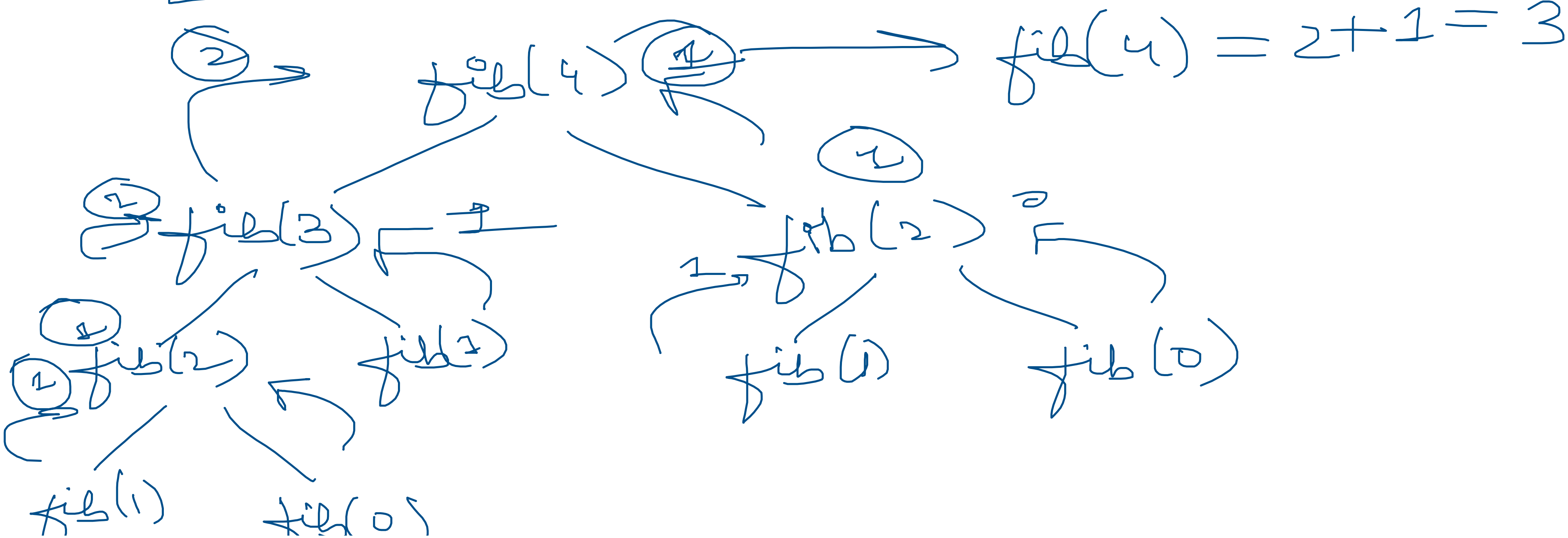
$F_0 = 0, \quad F_0 = 1$

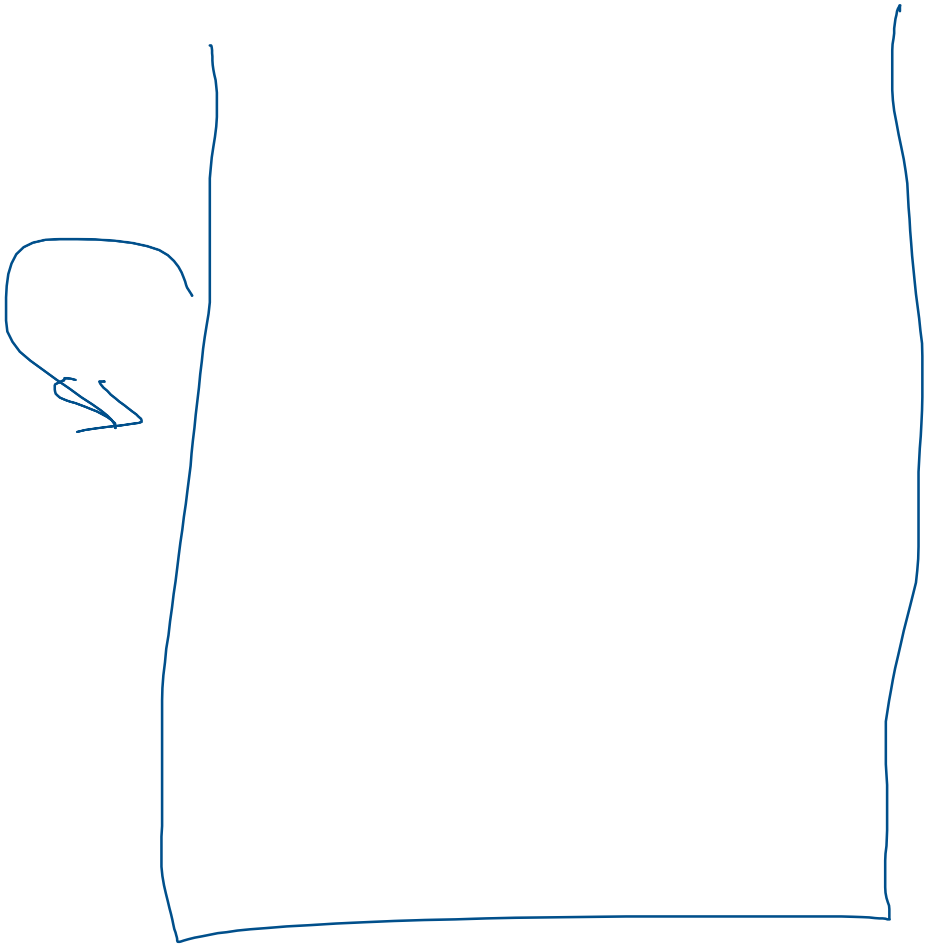
$$\begin{array}{cccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & & & & \\ 0, & 1, & 1, & 2, & 3, & 5, & 8, & 13 & 21, & 44, & 55, & 89--- \end{array}$$

$$\boxed{F_n = F_{n-1} + F_{n-2}}$$

function → fib(n) → returns us fibonacci of n

$$\boxed{fib(n) = fib(n-1) + fib(n-2)}$$





B → fib(n-1)

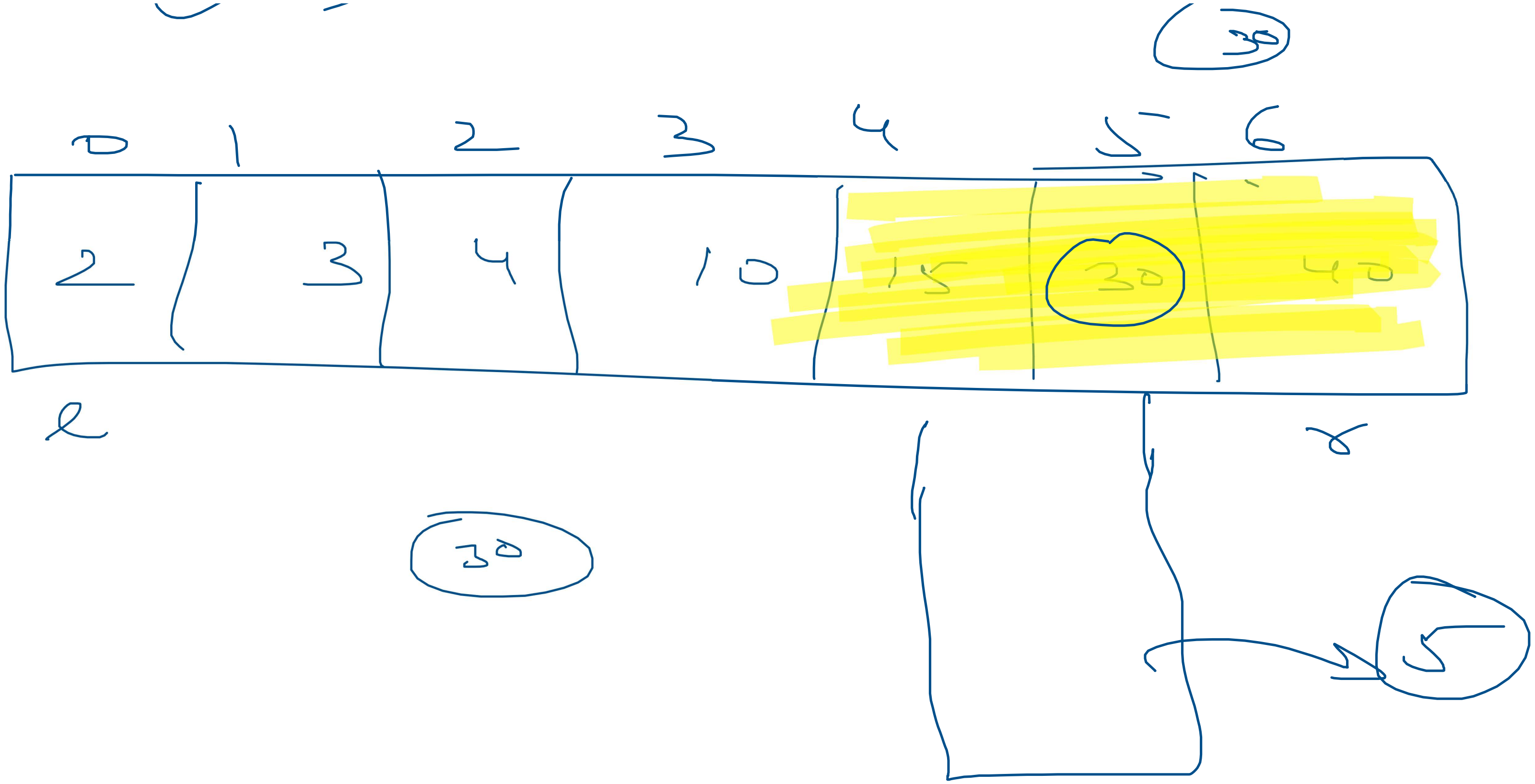
C → fib(n-2)

3

2 Cases to pop() from Stack

① → either the terminating condition runs.

② → function is completed



Sliding Window

Approach-1

1	4	2	10	23	3	1	0	20

Size = $\boxed{n = 9}$

k = 4

Max sum of k consecutive elements in the array.

31

$i = 0$

1	4	2	10	23	3	1	0	20

$i \quad i+1, \; i+2 \quad i+k$

3

1	4	2	10	23	3	1	0	20

39

1	4	2	10	23	3	1	0	20

$\boxed{39}$

1	4	2	10	23	3	1	0	20

1	4	2	10	23	3	1	0	20

$\boxed{9 - 4}$

0	1	2	3	4				
1	4	2	10	23	3	1	0	20

$\boxed{i = 0 \; ; \quad i <= n-k}$

1 → k

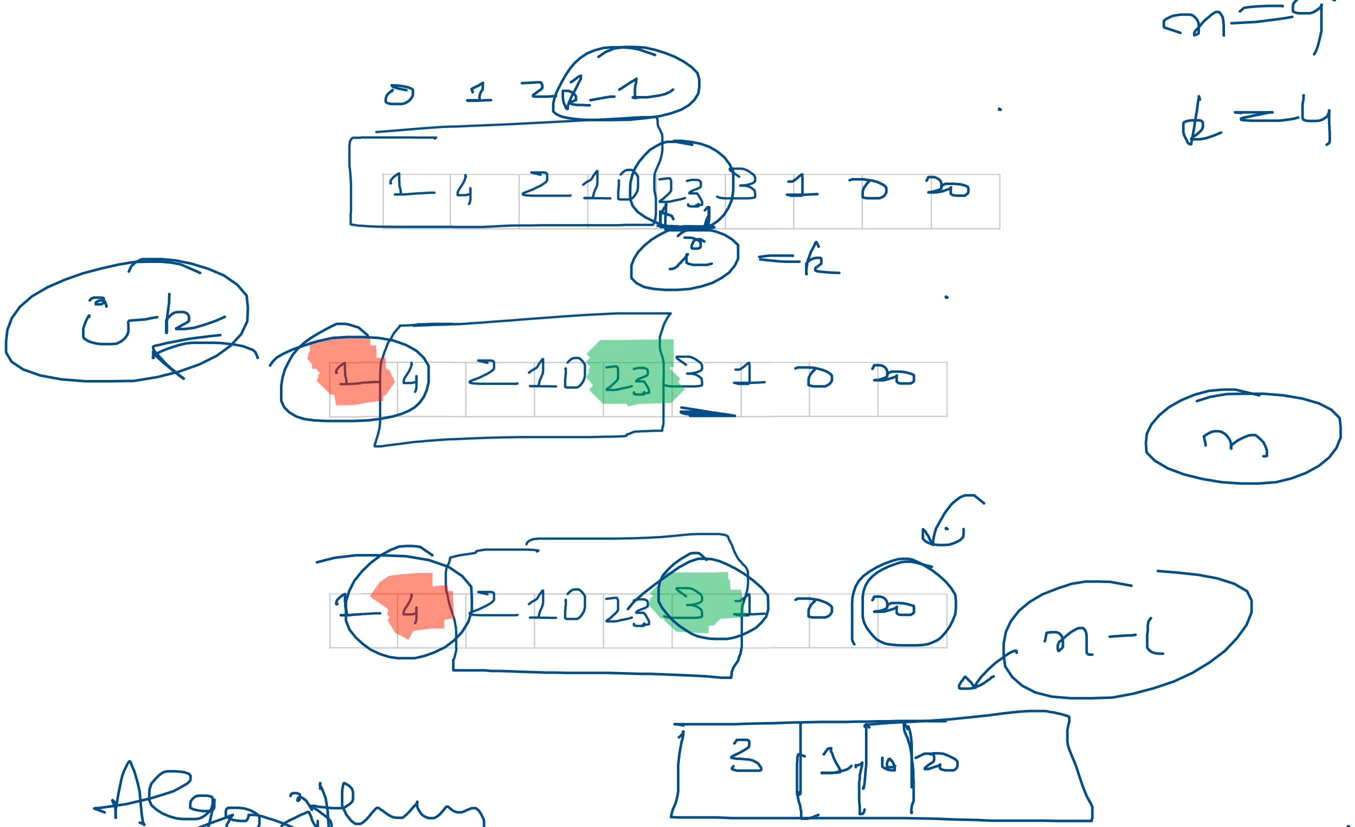
2 → k

$\boxed{(n-k) \times k}$

(n-k) → k

O(n*k)

Approach 2



Algorithm

1 -> Find sum of first window

add next element

2 => Subtract first element _ with next, [illegible]

3 => Save the max - sum .

Two Pointer Concept

l r

10	20	35	50	75	80

155

l <= r

Check if there exist 2 no. whose sum is 95

95

135

(90 < required)

(100 > required)

[illegible] => return true

Insertion at end of Array

0	1	2	3	4
3	4	5	10	

4th value = 10

arr[3] = 10

If array filled

0	1	2	3	4
3	4	5	10	20

0	1	2	3	4	5	6	7	8	9
3	4	5	10	20	30				