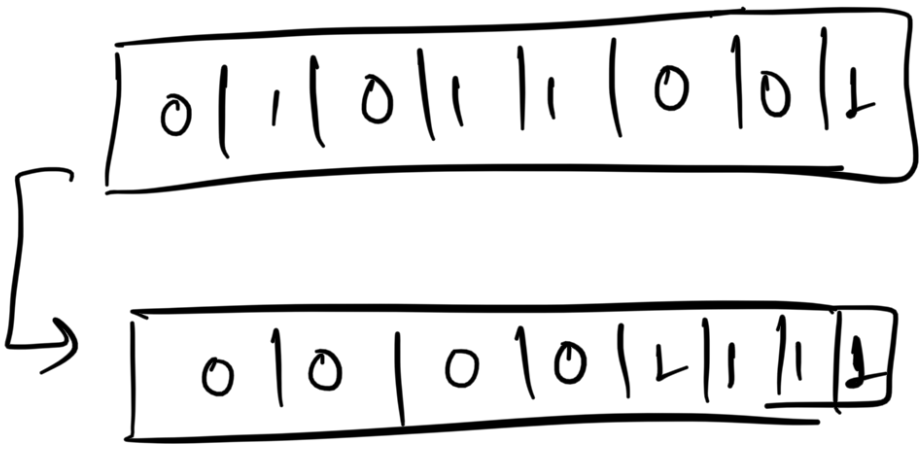


(1) Approach 1 :-

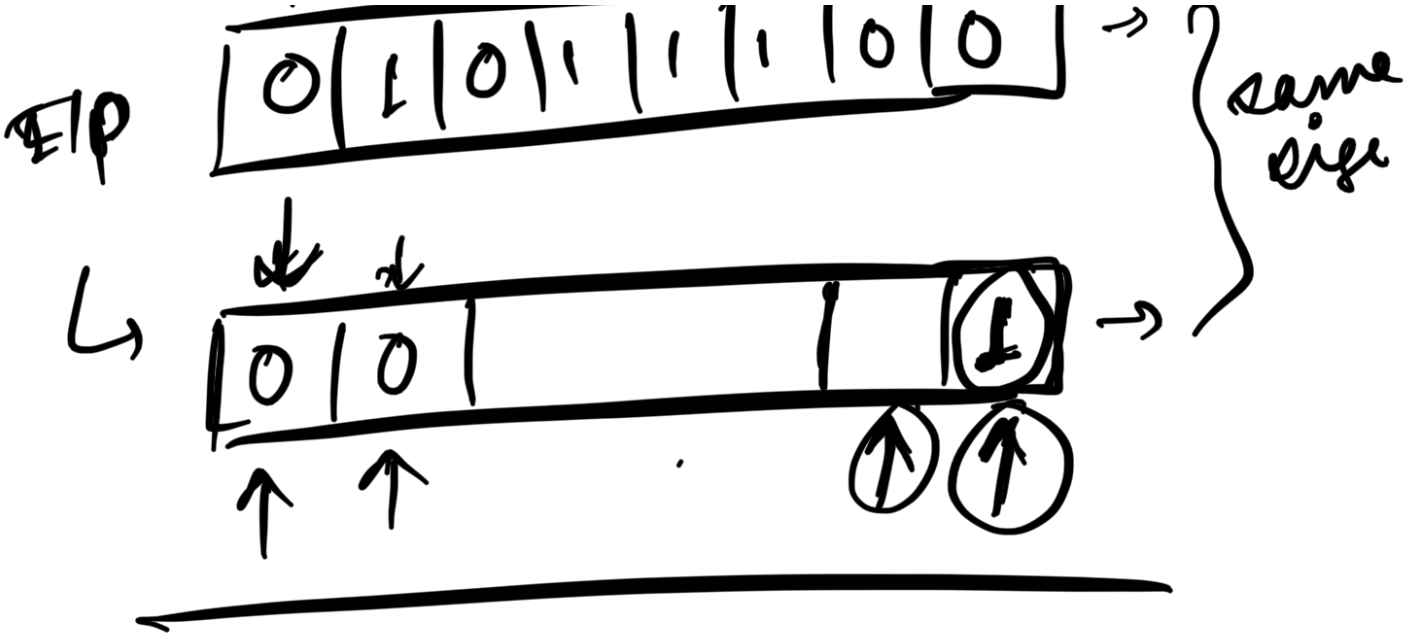
Count the total no. of 0's
→ so length of arr - total 0's will be 1's.

4 zeros

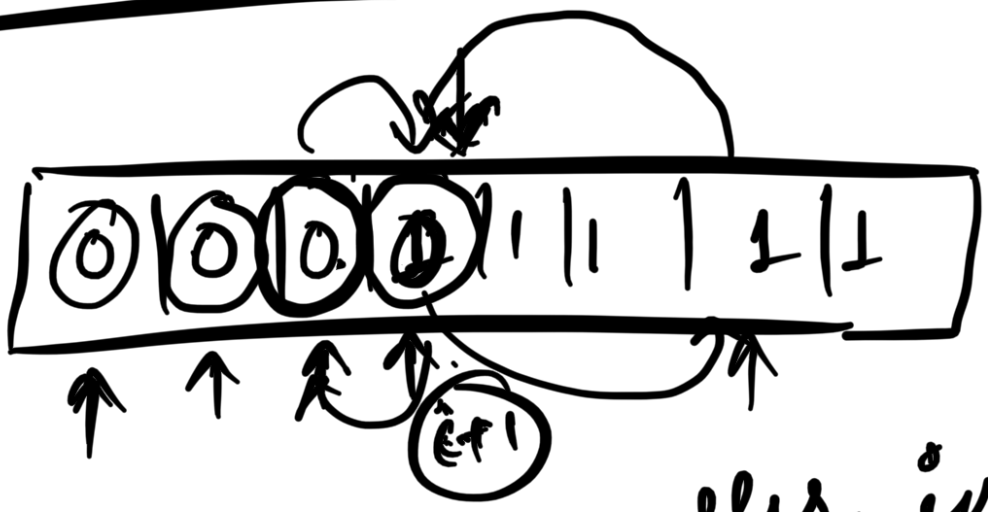
→ I/P update the same array

---

(2) Approach 2:



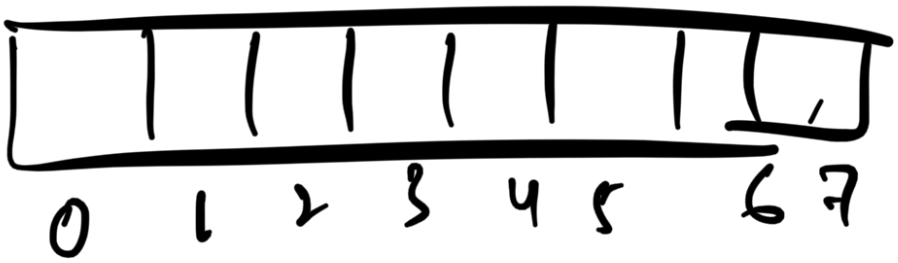
Approach 3:-



if (value at smaller index
> value at greater index)

1 → e (temp) = arr(i);

0 → i+1 arr(i) = arr(i+1);



- PUT VALUE OF arr(i+1) in arr(i) and value of arr(i) in arr(i+1);

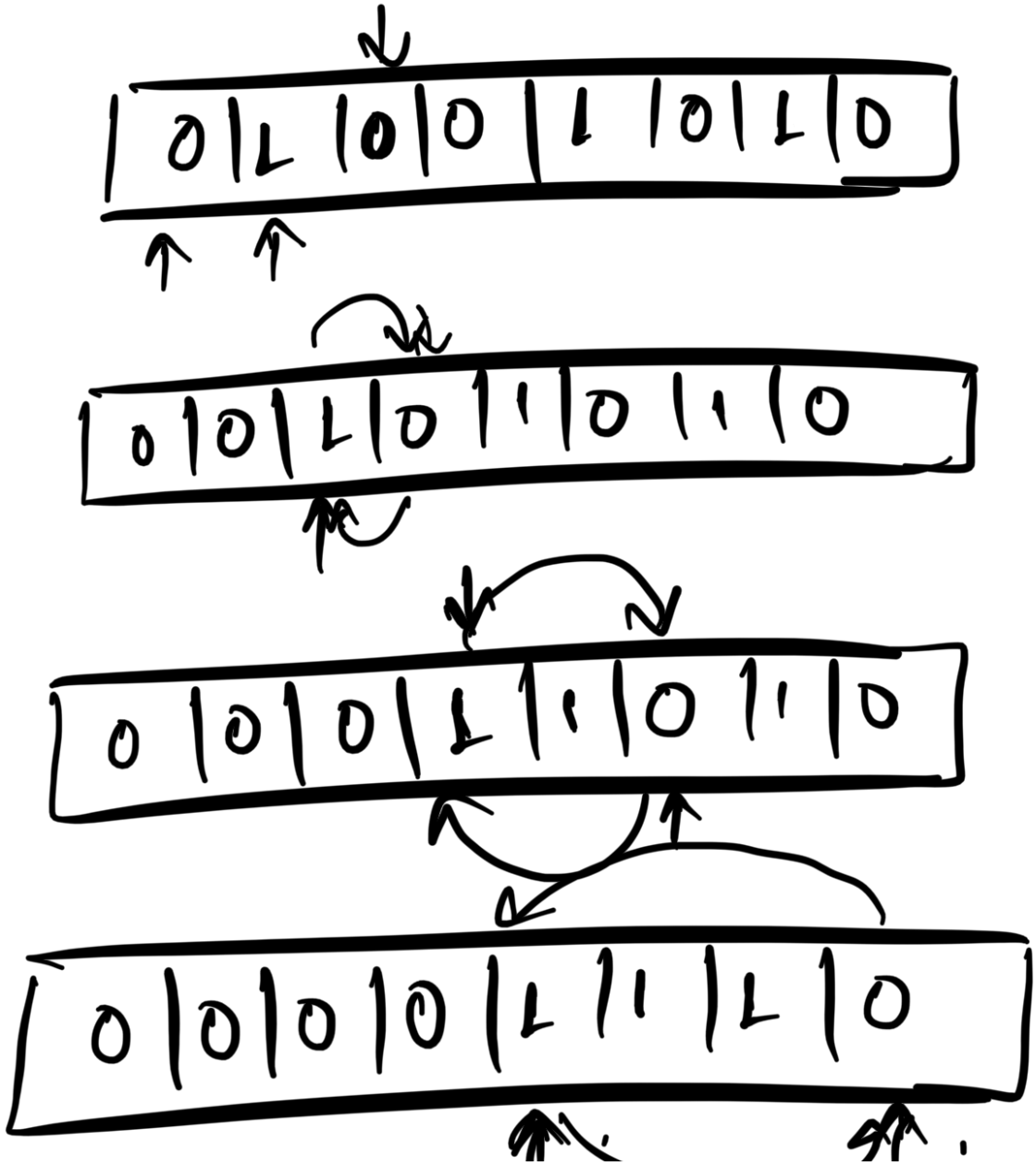
arr(i) = arr(i+1)

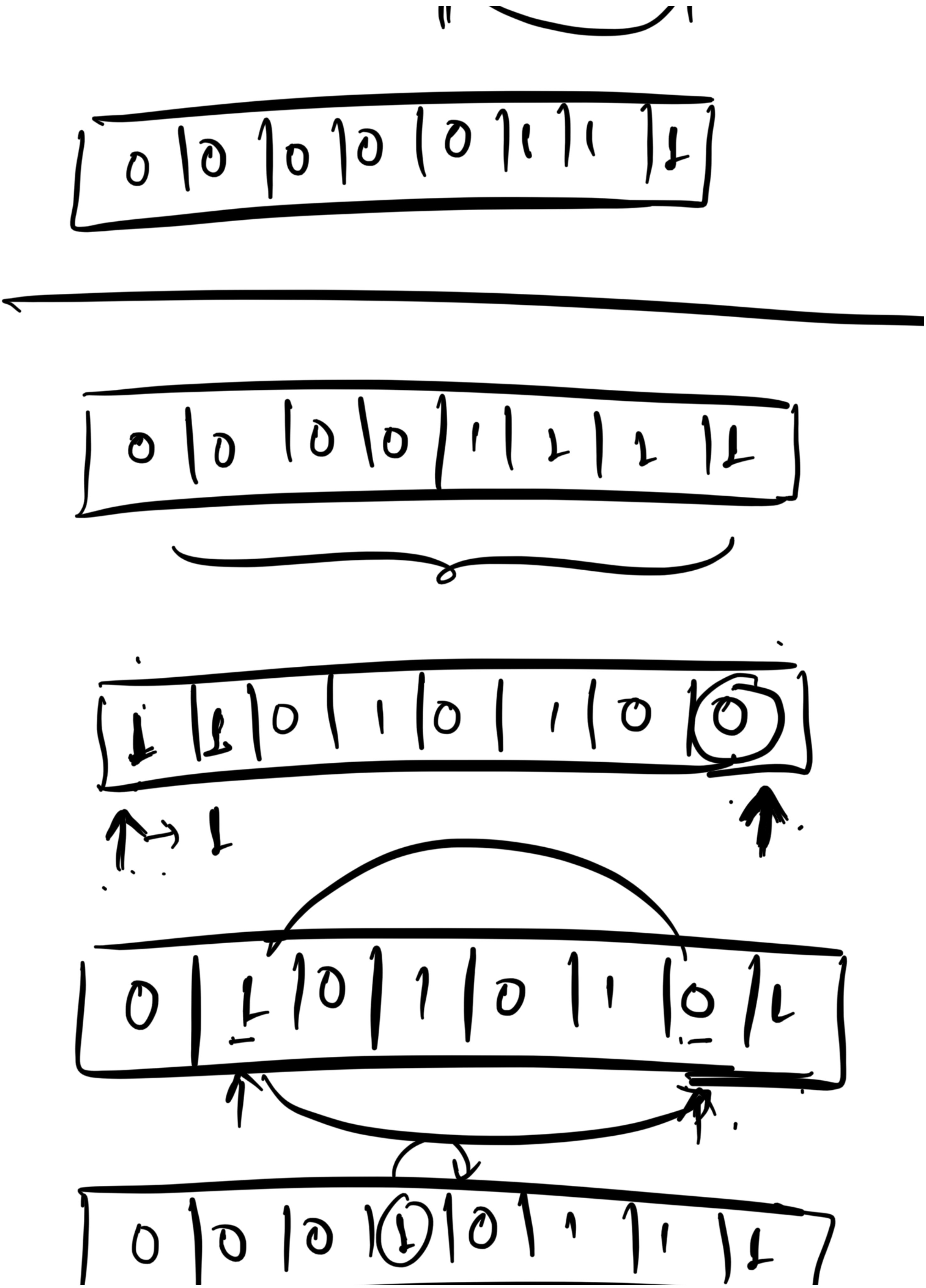
arr(i+1) = arr(i)

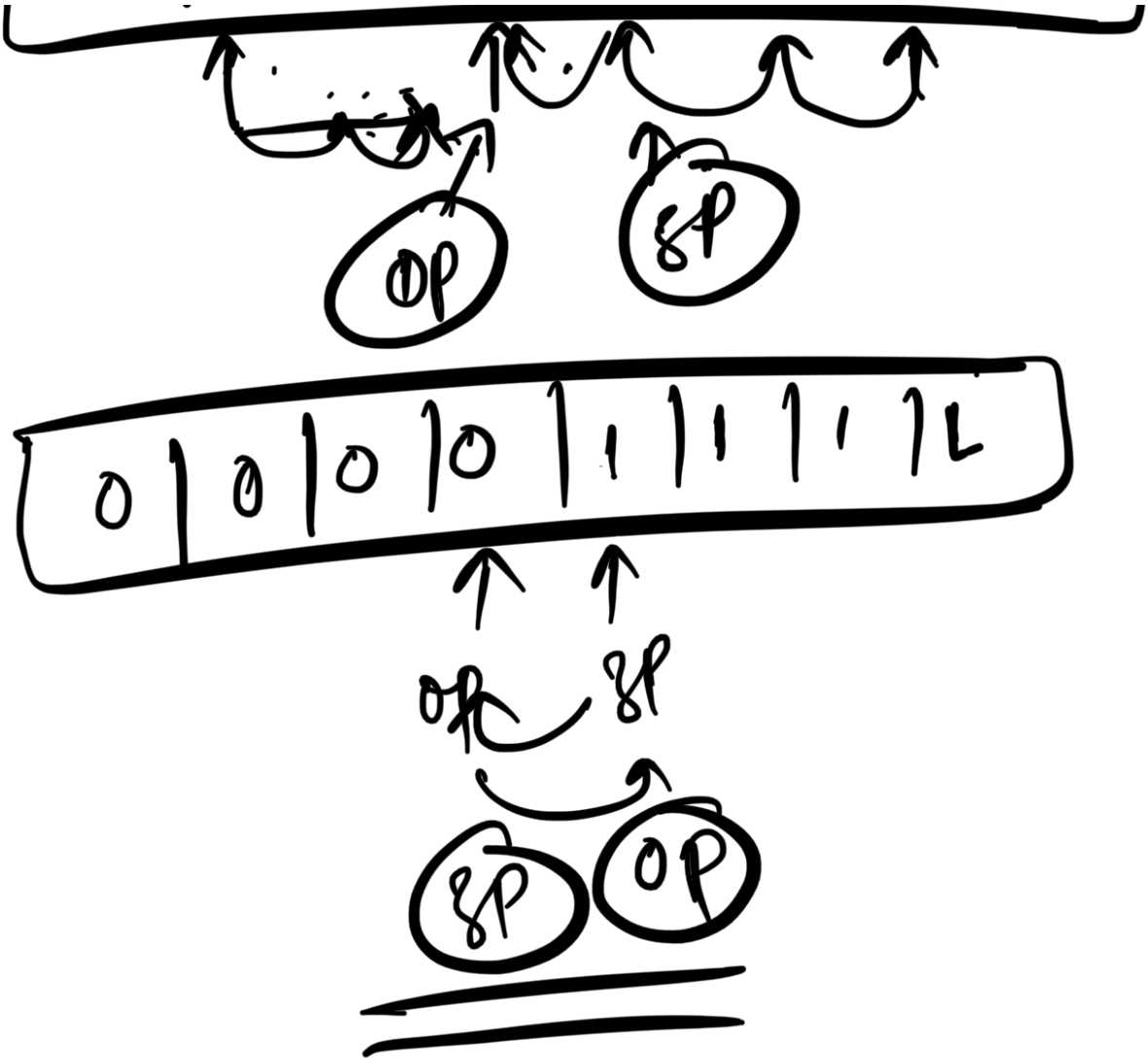
swap

01

```
temp = arr[i];
arr[i] = arr[i+1];
arr[i+1] = temp;
```



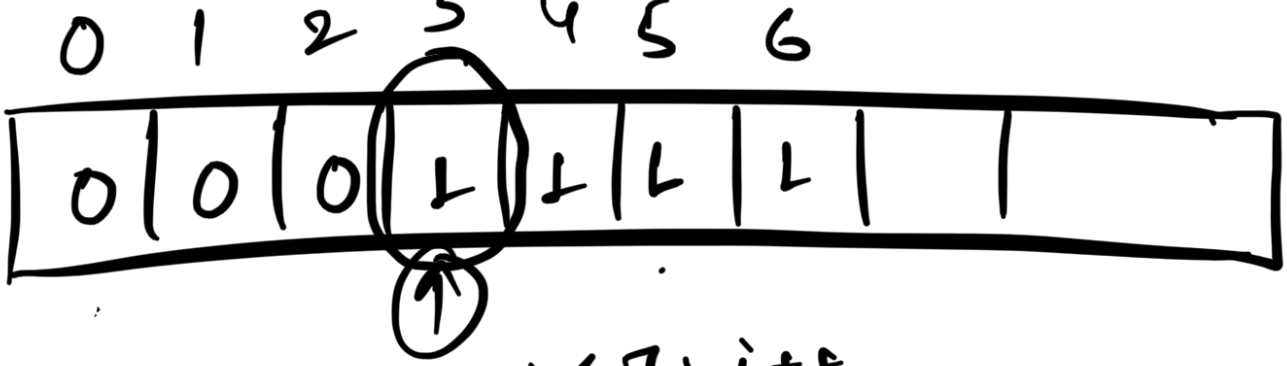

0 0 0 0 0 1 1 1
0 0 0 0 1 1 1 1
1 1 0 1 0 1 0 0
1
0 1 0 1 0 1 0 1
0 0 0 1 0 1 1 1



countOf0's = 3, countOf1's = 4

i = 3

i++



i = 3; i < 7; i++.

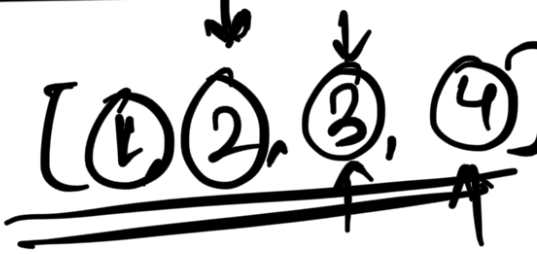
i < countOf4's; i++.

i = countof0's, ...

i = 3; i < 4; i++.

i = countof0's; i <
countof0's + countof1's

(i = countof1's; i < length; i++)



1*2

1*2*3

Two arrays.

1 array → left array
stores product of all values
to the left of index

[index] → left array.

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

[24, 12, 4, 1] → right array

[2*3*4, 3*4, 1*4, 1]

indx

left → product till indx - 1

right → product till indx + 1

i

left(i) * right(i).

(1, 2), 3 (4, 5, 6)

left(i) 1*2

right[i] 4*5*6

bit()

prod = 1 , bit(0) = 1

bit(1) = 1

prod *= 2 , prod = 2

bit(2) = prod (2)

prod *= 3 , prod = 6.

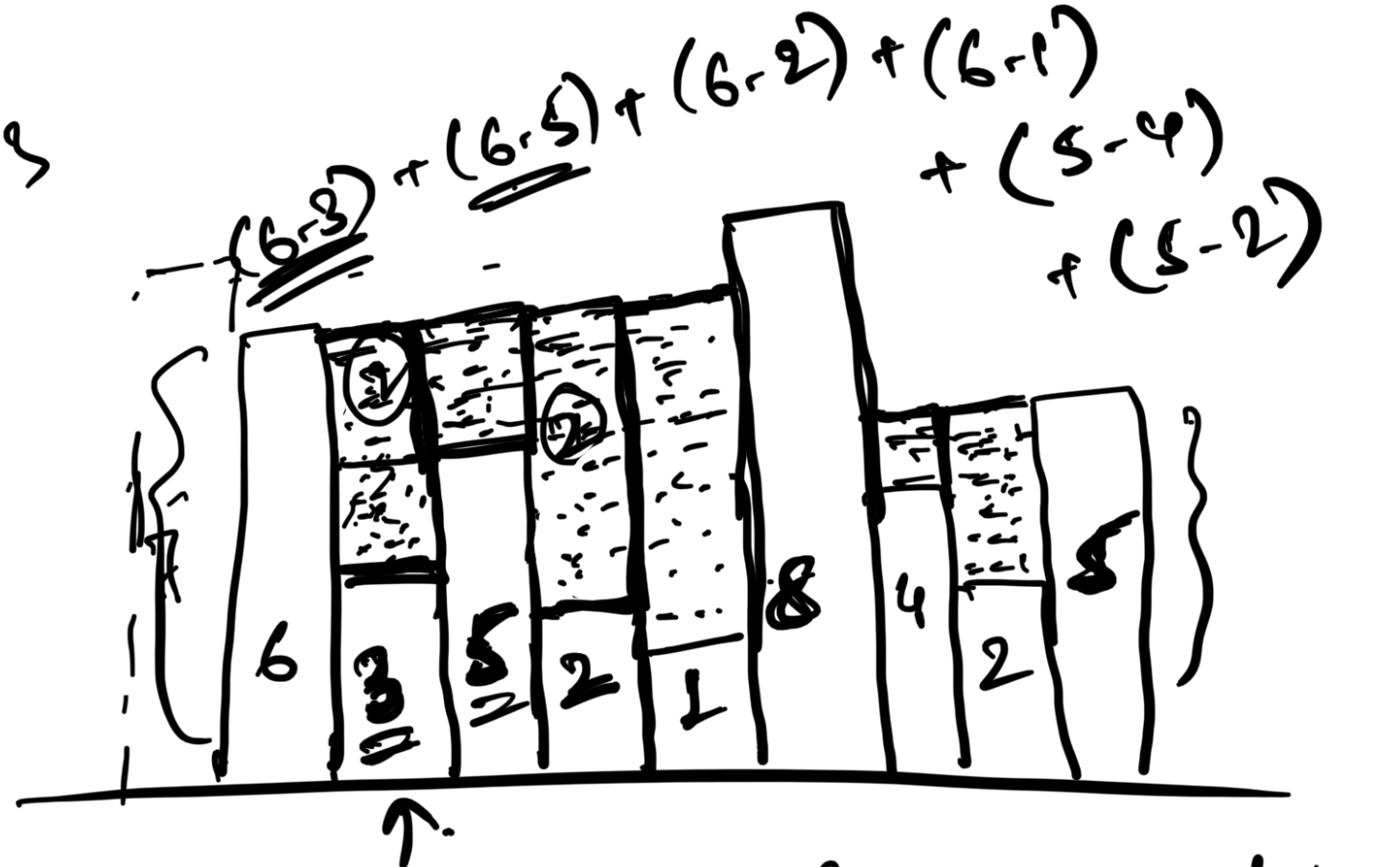
bit(3) = prod (6)

prod *= 4 ; = 24

bit(4) = prod (24)

prod * 2 num(i) (120)

cpe(5) = prod (120)



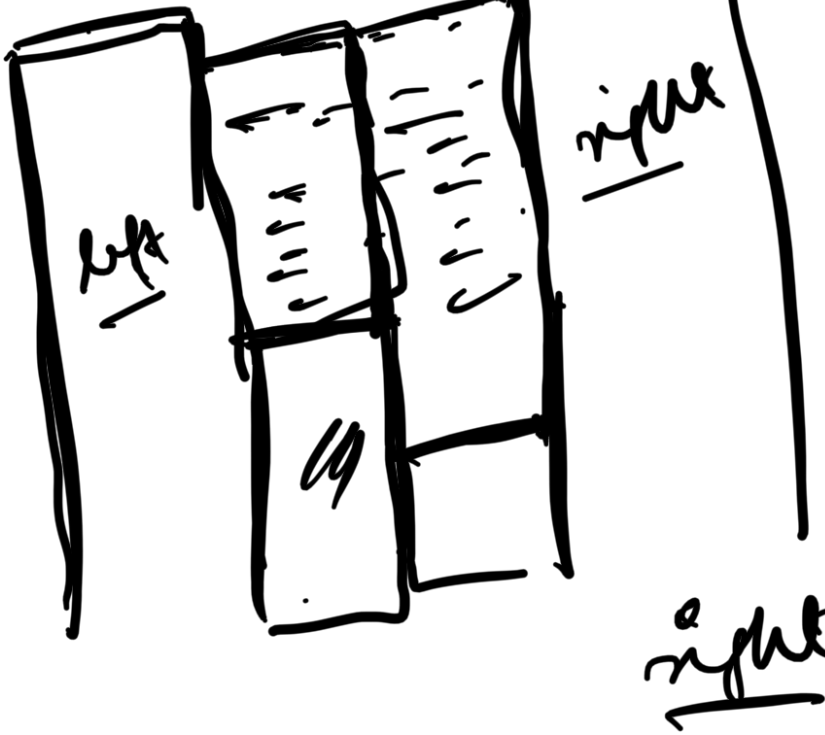
Given the height of each building. Calculate the total amount of water collected

I/P :- array of heights

[6, 3, 5, 2, 4, 8, 4, 2, 5]

O/P :- 3 + 1 + 4 + 5 + 1 + 3 = (17)

consider the width of each building as 1 meter



height of shorter

- height of curr building

left -

:lding at index

for every building

i;

we need to find tallest building to the left.

we need to find tallest building to the right

---

we need to find min of the left and right of current index

min(left[i], right[i]) - height[i];