Page No.:
Date: / /

Lecture - 39

(Queue problems)

problem : 1 → Circular Tour

→ A circle

→ N petrol pump on the circle

You have two array :

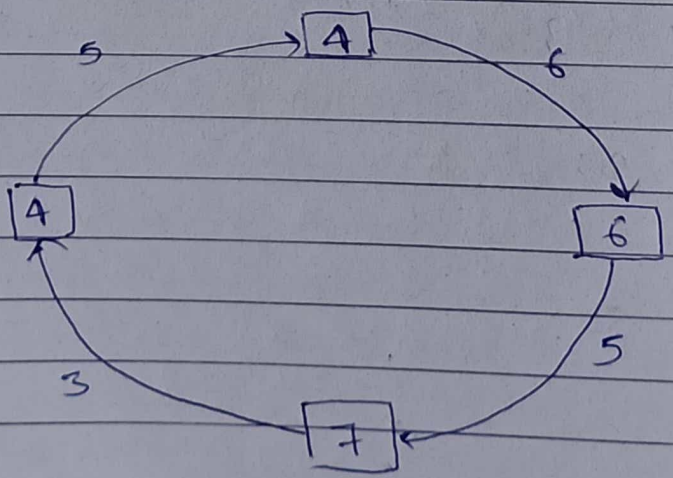
1) Amount of petrol that every petrol pump has.

2) Distance from that petrol pump to next petrol pump.

1 litre petrol ⟶ 1 unit of distance

→ N = 4

petrol :	4	6	7	4
Distance :	6	5	3	5



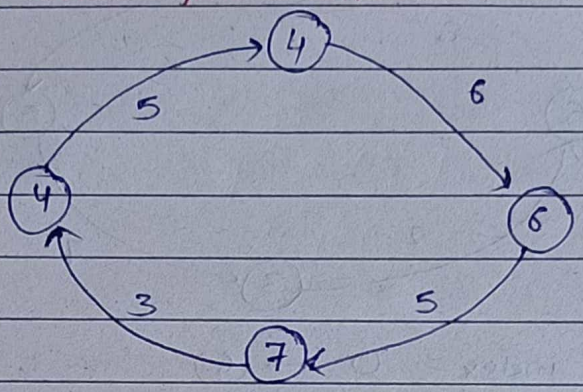
Teacher's Signature.....

Page No.
Date : / /

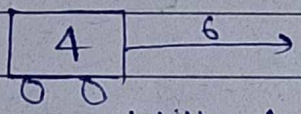
Return the point where it can start and complete all the petrol pump.

otherwise return -1.

Approach 1 (Brute force Approach)

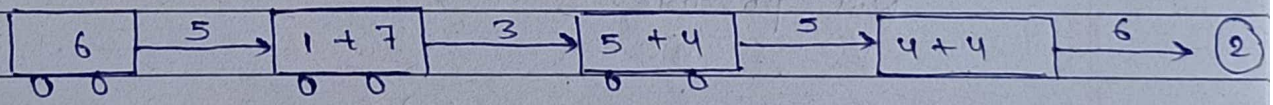


Start with index → 0 (4) ✗



With 4 litere of petrol we can't cover 6 unit distance
Not possible

Start with index → 1 (6)



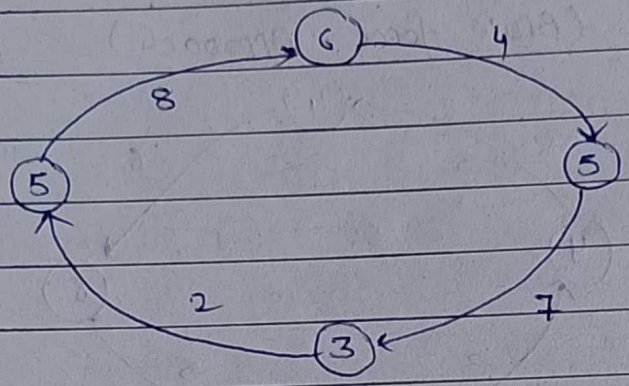
When we start from index 1 where 6 litre of petrol, then we can traverse full circle.

return (1) → index.

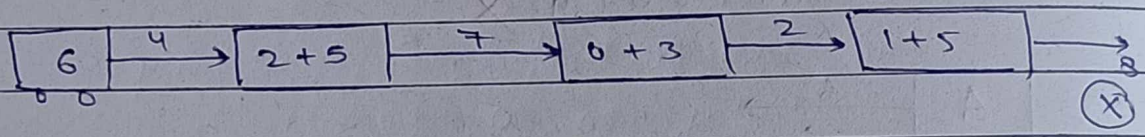
Teacher's Signature.....

Example-2 :

petrol	6	5	3	5
Distance	4	7	2	8

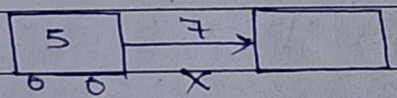


Start With index 0 → (6)



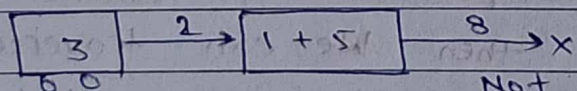
Not possible

Start With index 1 → (5)



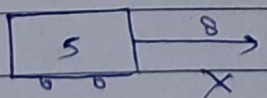
Not possible

Start With index 2 → (3)



Not possible.

Start With index 3 → (5)



Not possible

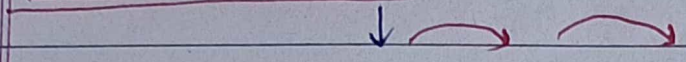
return -1

Teacher's Signature.....

In this approach we check from all petrol pump and go to all other petrol pump.

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

Optimal solution



Petrol	6	6	5	7	11
Distance	4	7	10	5	9

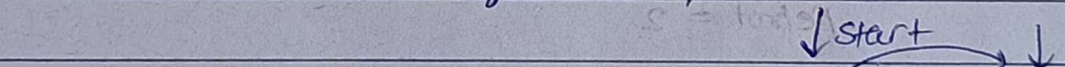
6 - 4 = 2

2 + 6 = 7, 8 - 7 = 1

1 + 5 = 6, 6 - 10 = -4 ✗

If we start from index 0 then not possible.

the next starting index is where we fail, we start one index after the failed index.



petrol	6	5	5	7	11
Distance	4	7	10	5	9

7 - 5 = 2

2 + 11 = 13, 13 - 9 = 4

(pointer = (pointer + 1) % N)

6	6	5	7	11
4	7	10	5	9

4 + 6 = 10, 10 - 4 = 6

6 + 6 = 12, 12 - 7 = 5

5 + 5 = 10, 10 - 10 = 0

Teacher's Signature.....

On this case we reached all petrol pump if we start from index 3.

T.C = O(n)

Implementation:

front ↓ rear ↓

Petrol	6	5	5	7	12
Distance	4	7	10	5	9

6-4 = 2

put two pointer front & rear on index 0 of petrol.

~~check~~ ~~petrol[i] + petrol[0]~~

→ check petrol[0] - Distance[0] > 0

petrol = 2

→ rear = rear + 1

front ↓ rear ↓

petrol	6	5	5	7	12
Distance	4	7	10	5	9

6-4 = 2 2+5 = 7

7-7 = 0

Petrol = petrol + petrol[rear] - Distance[rear];

Petrol = 0

(0 >= 0)

rear++

Teacher's Signature.....

front ↓ rear ↓

petrol	6	5	5	7	12
Distance	4	7	10	5	9
	2	0	0+5=5 5-10 =-5		

(-5 < 0) → front = rear + 1;
rear = front;
petrol = 0

rear ↓ front ↓

petrol	6	5	5	7	12
Distance	4	7	10	5	9
				7-5 =2	

petrol = (2 > 0) ✓
rear++

front ↓ rear ↓

6	5	5	7	12
4	7	10	5	9
			2	2+12=14 14-9=5

petrol = (5 > 0)
rear++;
rear = (rear+1)%n;

rear ↓ front ↓

6	5	5	7	12
4	7	10	5	9
5+6=11				5
11-4=7				

petrol = 7 > 0 rear++;
rear = (rear+1)%n;

Teacher's Signature.....

rear ↓ front ↓

6 5 5 7 12

4 7 10 5 9

7 7+5 =12 12-7=5 (5>0) rear++;

rear ↓ front ↓

6 5 5 7 12

4 7 10 5 9

5 5+5=10 10-10 =0

(0>=0) rear++;

rear ↓ front ↓

6 5 5 7 12

4 7 10 5 9

(front == rear)
return front;

Code:

```
int tour (petrolpump p[], int n) {
    int front = 0, rear = 0;
    int petrol = 0;

    While (front < n) {
petrol += p[rear].petrol - p[rear].distance;
  if (petrol < 0) {
     if (rear < front)
        return -1;
```

```
            petrol = 0;
            front = rear + 1;
            rear = front;
        }
        else {
            rear = (rear + 1) % n;
            if (front == rear)
                return front;
        }
    }
    return -1;
}
```

Approach 3:

petrol	6	5	5	7	12
Distance	4	7	10	5	9
	6-4 = 2	2+5-7 = 0	0+5-10 = -5	0+7-5 = 2	2+12-9 = 5
Defict :			-5		
Balance :	~~2~~	~~0~~	~~-5~~ 0	~~2~~	5

At last we check

balance + deficient >= 0

↳ true

↳ then return start

↳ false

↳ return -1

Teacher's Signature.....

start जहाँ पर balance 0 से कम हो जाए
उसके एक आगे start को point कर
दो।

Code:

```
int deficit = 0, balance = 0, start = 0;

for (int i = 0; i<n; i++) {
    balance += p[i].petrol - p[i].distance;
        if (balance < 0) {
            deficit += balance;
            balance = 0;
            start = i+1;
        }
    }

    if (balance + deficit >= 0)
        return start;

return -1;
}
```

Teacher's Signature.....

Dequeue

updated version of queue
push and pop from both side.

Dequeue operation:

```
dequeue <int> d;

d.push_back(9);
d.push_front(11);
d.pop_back();
d.pop_front();
d.front();
d.back();
```

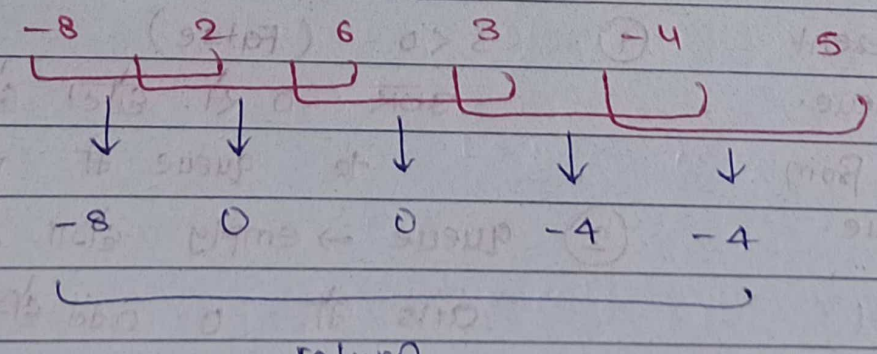
> for getting element.

problem 2: First Negative integer in every window of size K.

N = 5

A[] = {-8, 2, 6, 3, -4, 5}

(K=2) ↳ window size



O(n*k)

return

Teacher's Signature.....

Page No.:
Date: / /

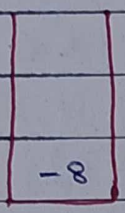
Better Approach

Size = 8

K=3

-8	2	6	3	-7	-11	13	14

from the 1st Window
put all the negative number
in ~~stack~~ queue.



so for 1st Window → 1st Negative is
q.front()
↳ -8

Ans: -8

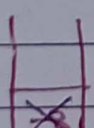
Size of Ans = size of A[] - K + 1
= 8 - 3 + 1
= 6
↳ 6 Window possible.

From 2nd Window:

→

-8	2	6	3	-7	-11	13	14

check -8 is present in queue.
Pop it from queue.
q.pop()

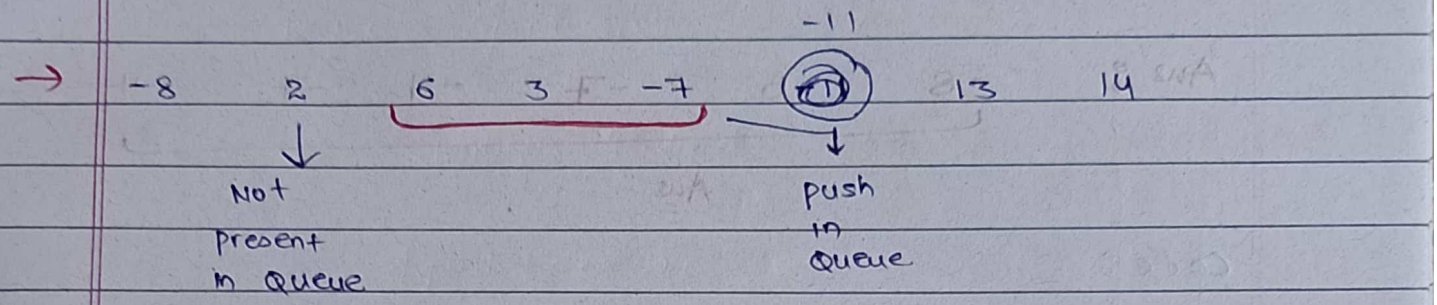


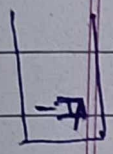
यहाँ 3 चीज देखना है।
① 3 < 0 (False)
अगर 0 से छोटा होगा
to queue में push कर देंगे।
② queue → empty होगा तो।
ans में 0 add होगा।

Teacher's Signature.....

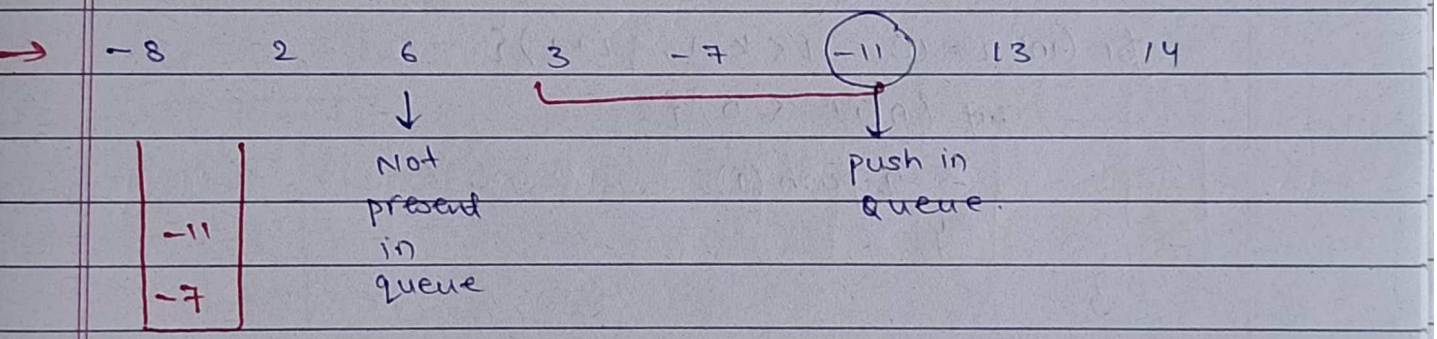
③ q अगर empty नहीं है तो
ans में ~~q.push_back~~
q.front() को push करेंगे।

Ans: -8 0

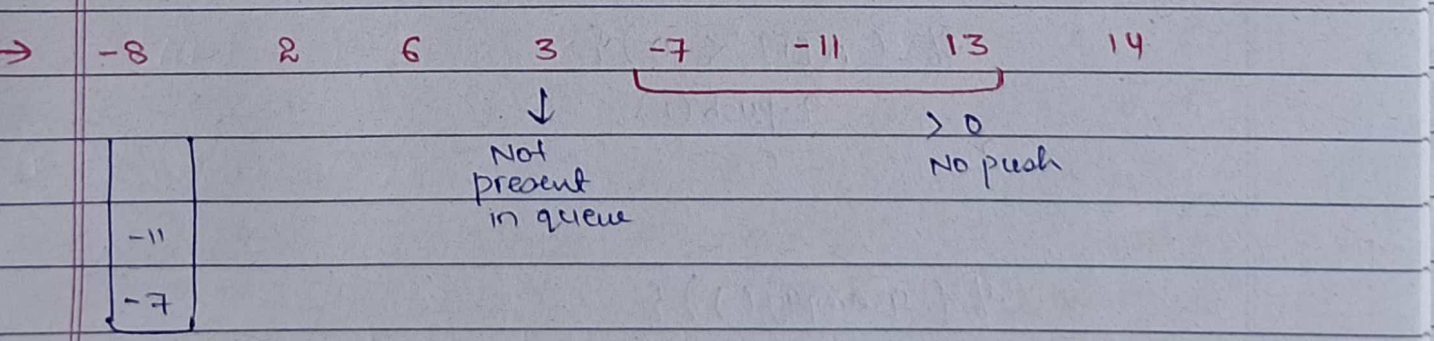




As: -8 0 -7
q.front()



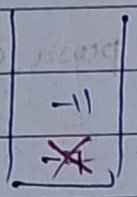
Ans: -8 0 -7 -7



Ans: -8 0 -7 -7 -7

Teacher's Signature.....

→ -8 2 6 3 -7 -11 13 14



pop from queue

>0 Not push

Ans : -8 6 -7 -7 -7 -11

Ans

Code :

```
vector < long long int > ans;
vector < long long int) q;

for (int i = 0;  i < k-1;  i++){
        if (A[i] < 0){
            q.push(i);
        }
    }

for (int i = k-1;  i < N;  i++ ){
        if ( A[i] < 0){
            q.push(i);
        }

    if ( q.empty() ){
        ans.push_back (0);
    }
    else {
        ans.push_back (A[q.front()]);
    }
```

Teacher's Signature.....

```
        if((!q.empty()) && q.front() <= i-k+1){
            q.pop();
        }
    }
    return ans;
}
```

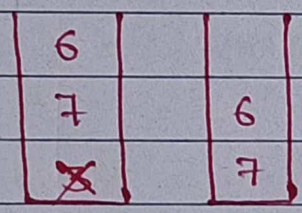
problem 3: Sliding Window Maximum (InterviewBit)

3 7 6 8 11 5 -2 4

Find Maximum from Window

k = 3

→ 3 7 6 8 11 5 -2 4



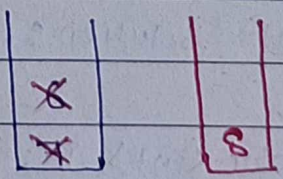
Max = 7

Ans:

→ 3 7 6 8 11 5 -2 4

3 → Not present in Queue

8 → greater than 6 & > 7

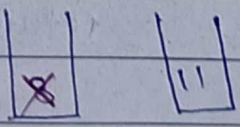


Ans: 7 8

→ 3 7 6 8 11 5 -2 4

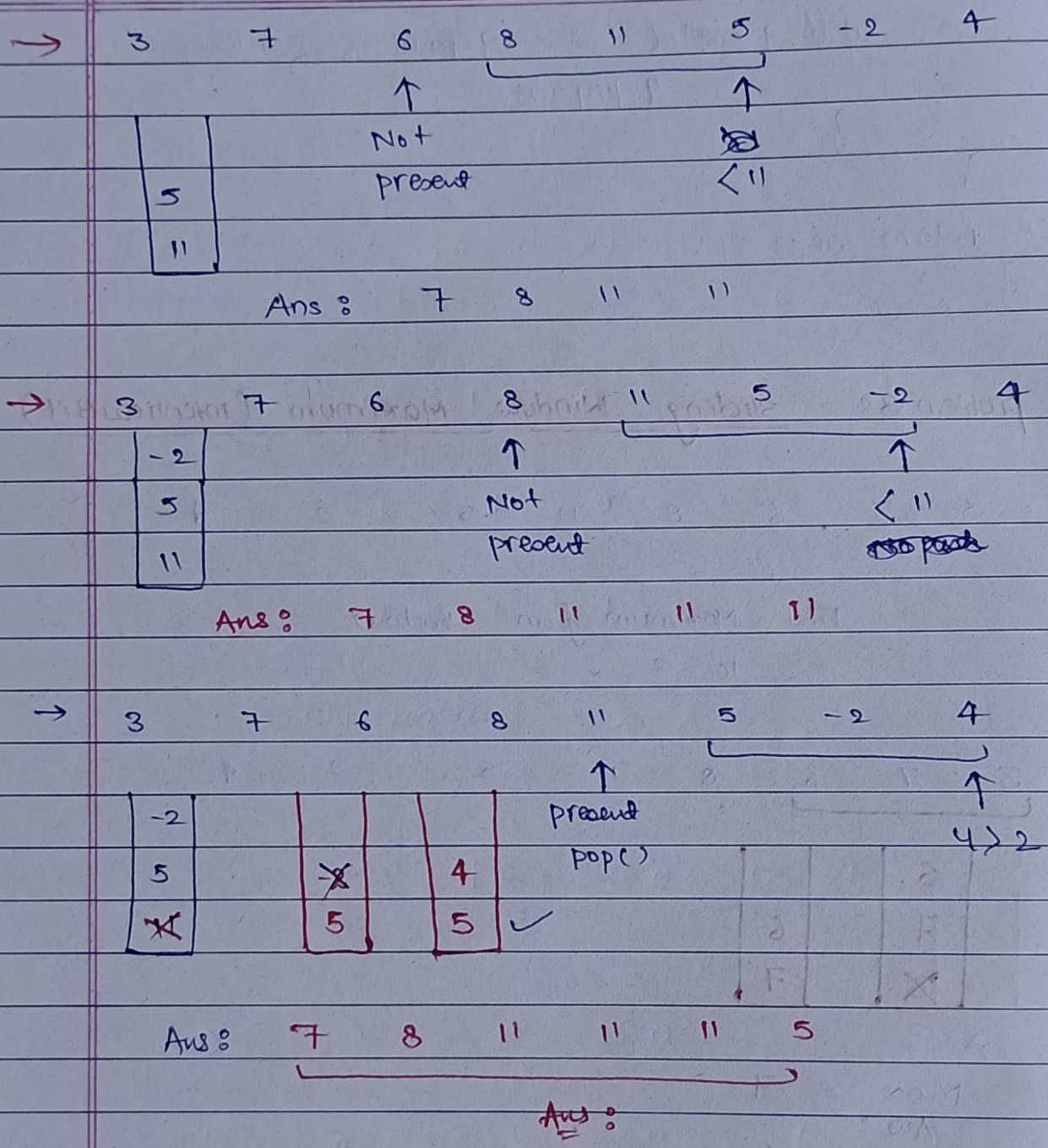
7 → Not present

11 → > 8



Ans: 7 8 11

Teacher's Signature.....



Ans:

Code:

```
vector<int> Solution::slidingMaximum(const vector<int> &A,
                                     int B){
    vector<int> v;
    if(A.size()==0)
        return v;

    deque<int> d;
```

Teacher's Signature.....

```
    int i=0, n = A.size();
    for(i=0; i<n && i<B ; i++){
        While (d.size() && A[d.back()] < A[i])
            d.pop_back();

        d.push_back(i);
    };

    V.push_back( A[d.front()]);
     if (B >= A.size())
        return V;

    for( ; i<n; i++) {
        if ((i-B) == d.front())
            d.pop_front();

        While ( d.size() && A[d.back()] < A[i])
            d.pop_back();
            d.push_back(i);
        V.push_back (A[d.front()]);
    }

    return V;
}
```

Teacher's Signature.....

Page No.:
Date: / /

Problem 4: Minimum Number of k Consecutive Bit Flips

Example: nums = [0, 1, 0]

k = 1

0 1 0

1 1 0

1 1 1

Flip required = 2

Example:

num = [0, 1, 0, 0, 1, 0, 1]

K = 3

0 1 0 0 1 0 1

1 0 1 0 1 0 1

1 1 0 1 1 0 1

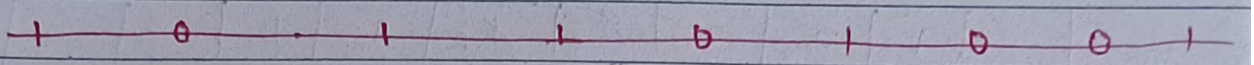
1 1 1 0 0 0 1

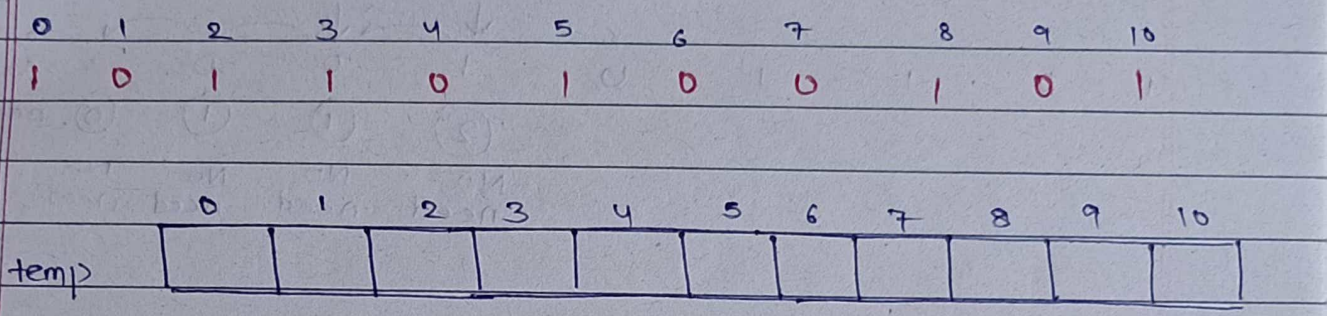
1 1 1 1 1 1 1

Flip required = 4

Teacher's Signature.....

Find zero → पहले zero find करना है फिर flip करना है।



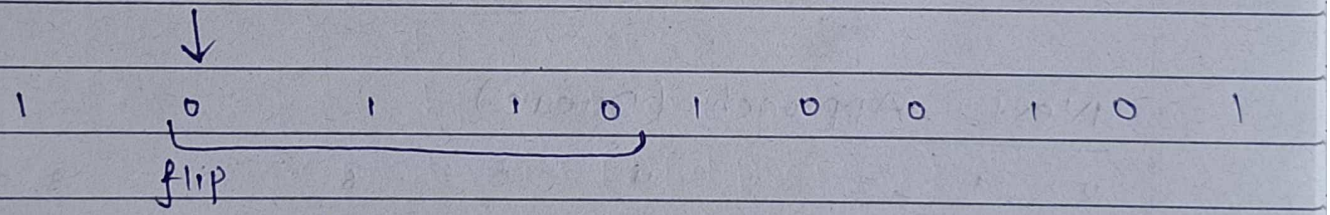


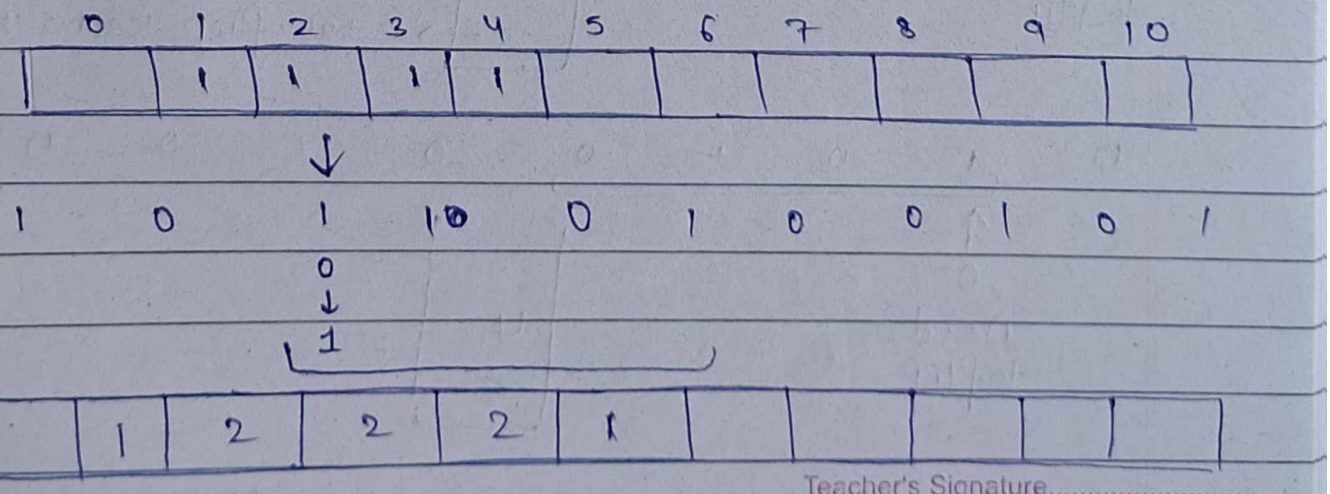
find the zero

Take subarray of size k.

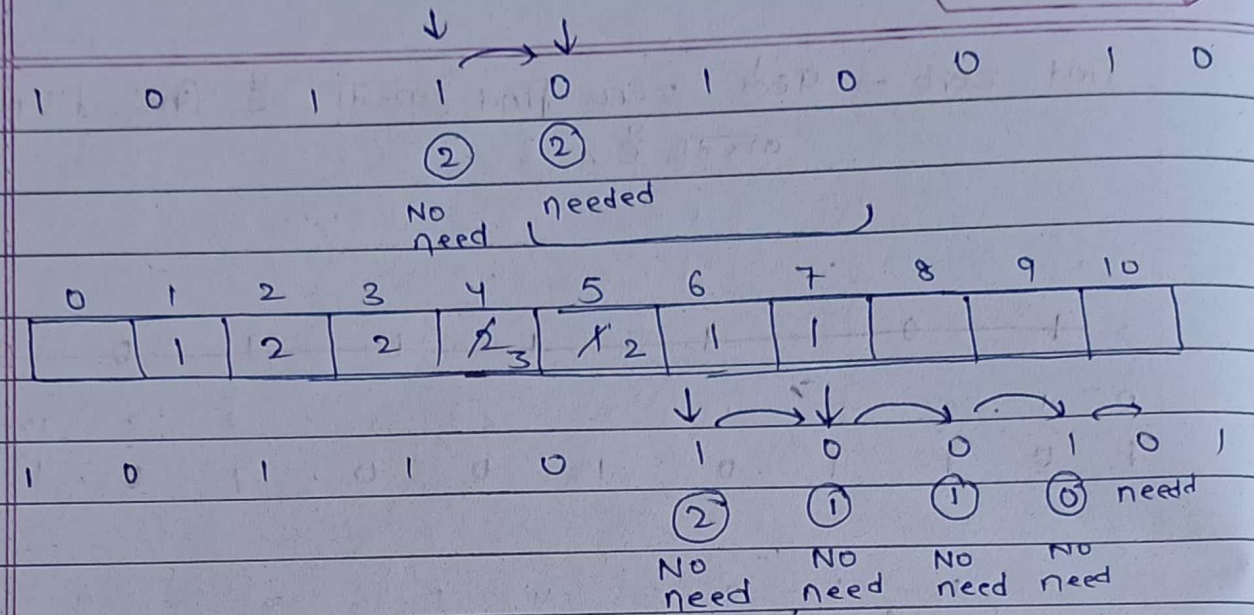
→ k-size subarray return - 1

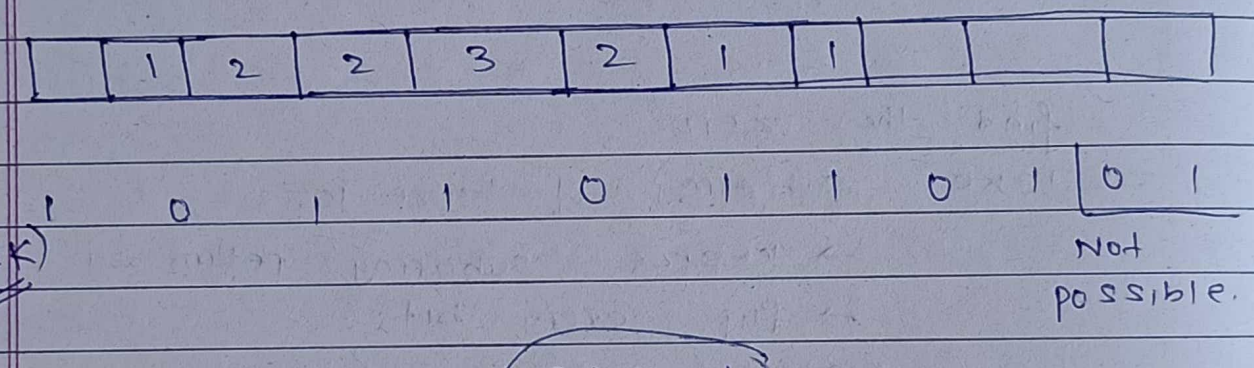
→ flip every bit.





Teacher's Signature.....

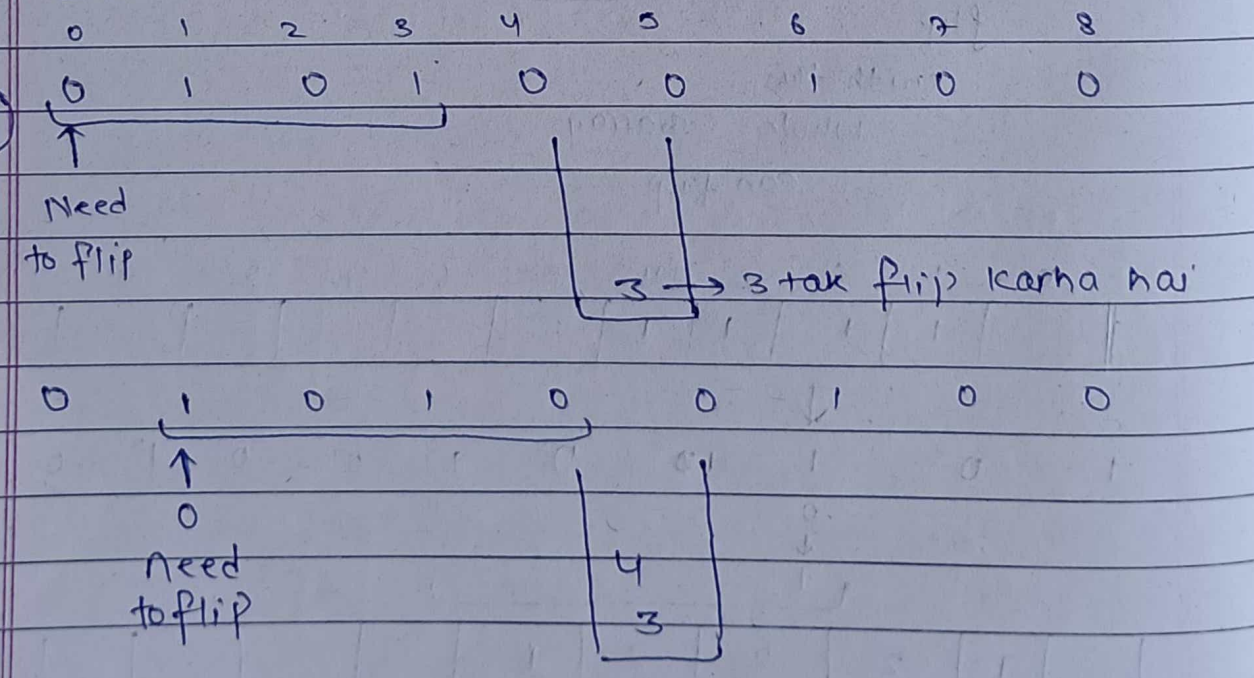




O(n*k)

return -1

Next Approach (Queue)



Teacher's Signature.....