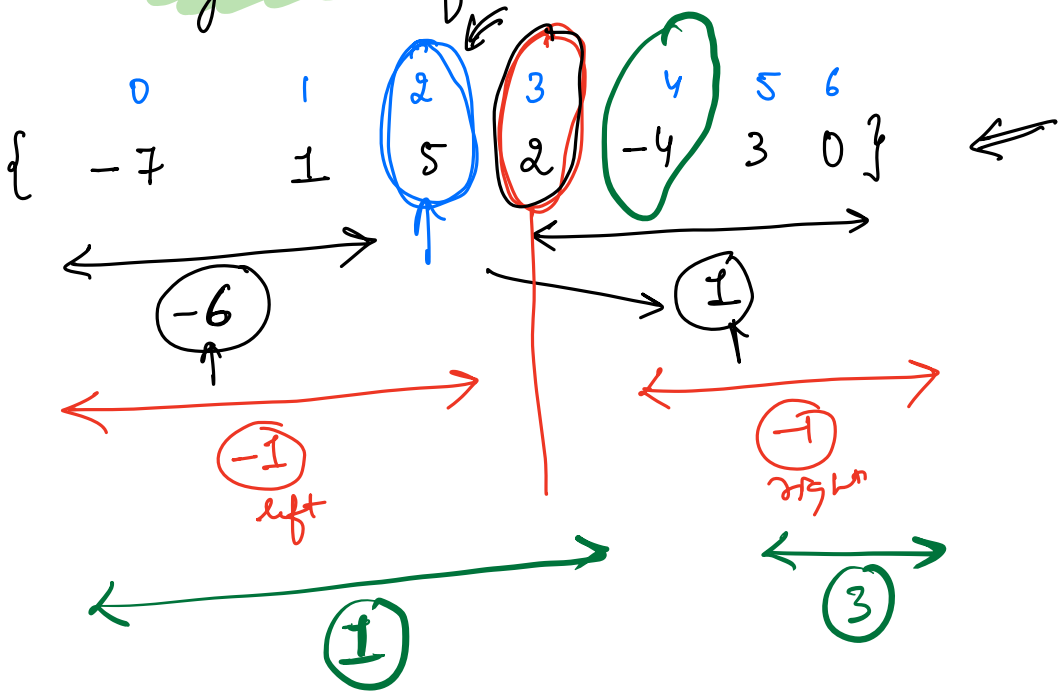
Q given array of size N. Find Equilibrium index of the array

Equilibrium index :- an index such that sum of elements on left side of index is equal to sum on right side of index.



Brute force

consider every single index → n

0 → n-1

i

left sum → 0 - i-1 subarray

right sum → i+1 → n-1

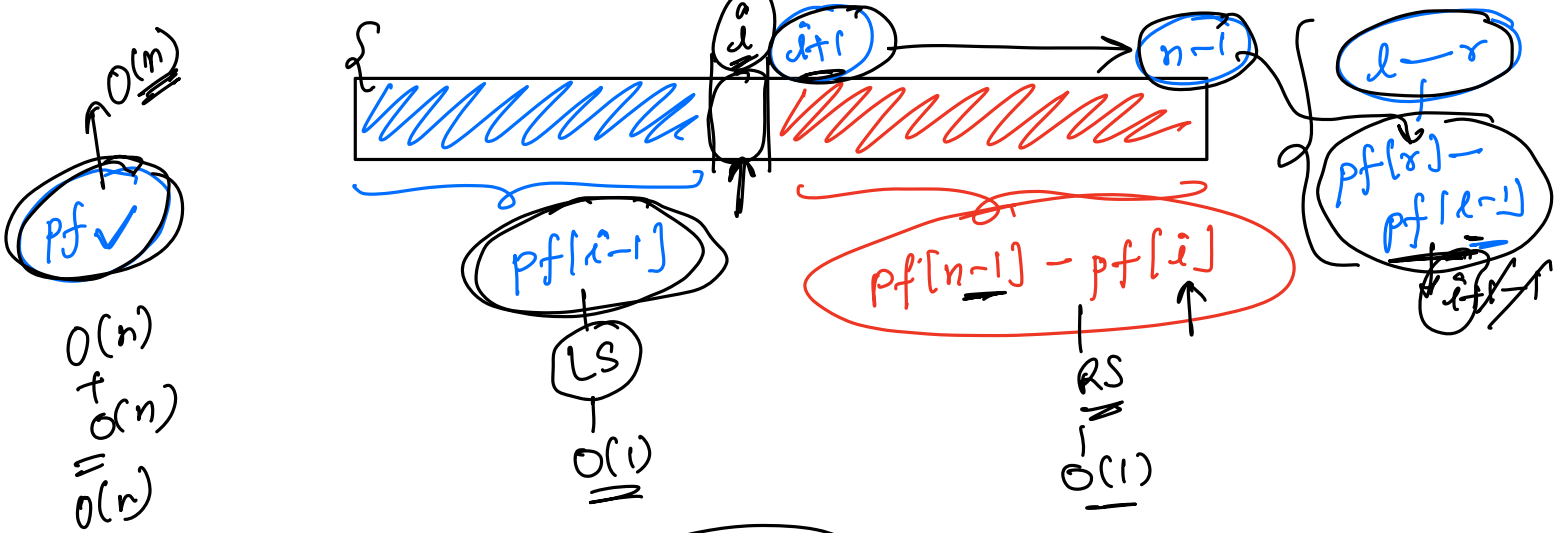
O(n)

O(1)

O(n²)

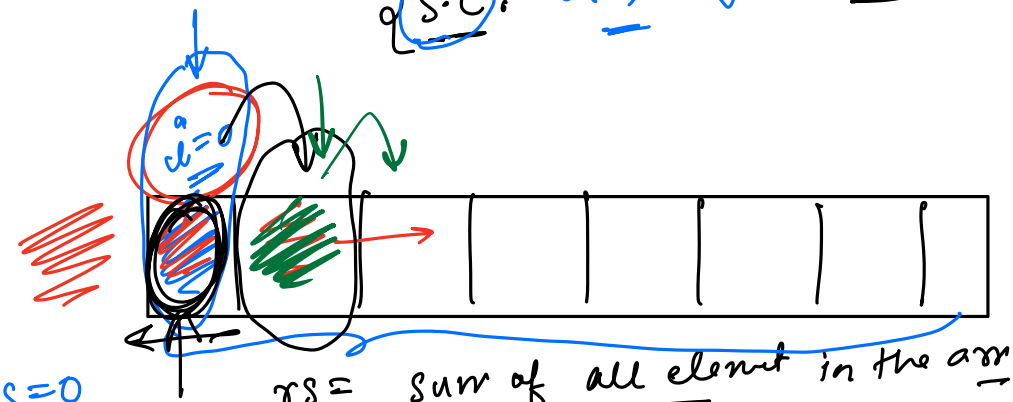
O(n)

→ to improve sum calculation → prefix sum



T.C: O(n) & O(1) & pf

S.C: O(n)

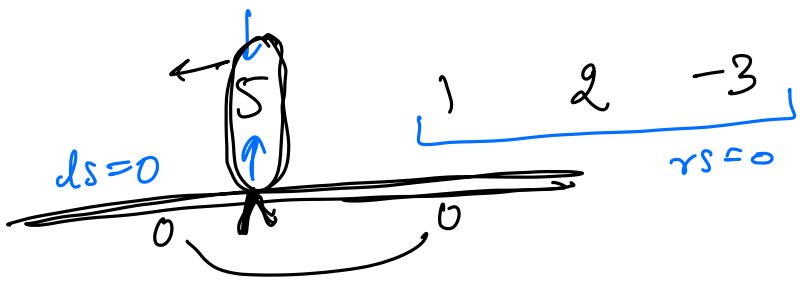


ls = 0

rs = sum of all elemt in the arr — O(n)

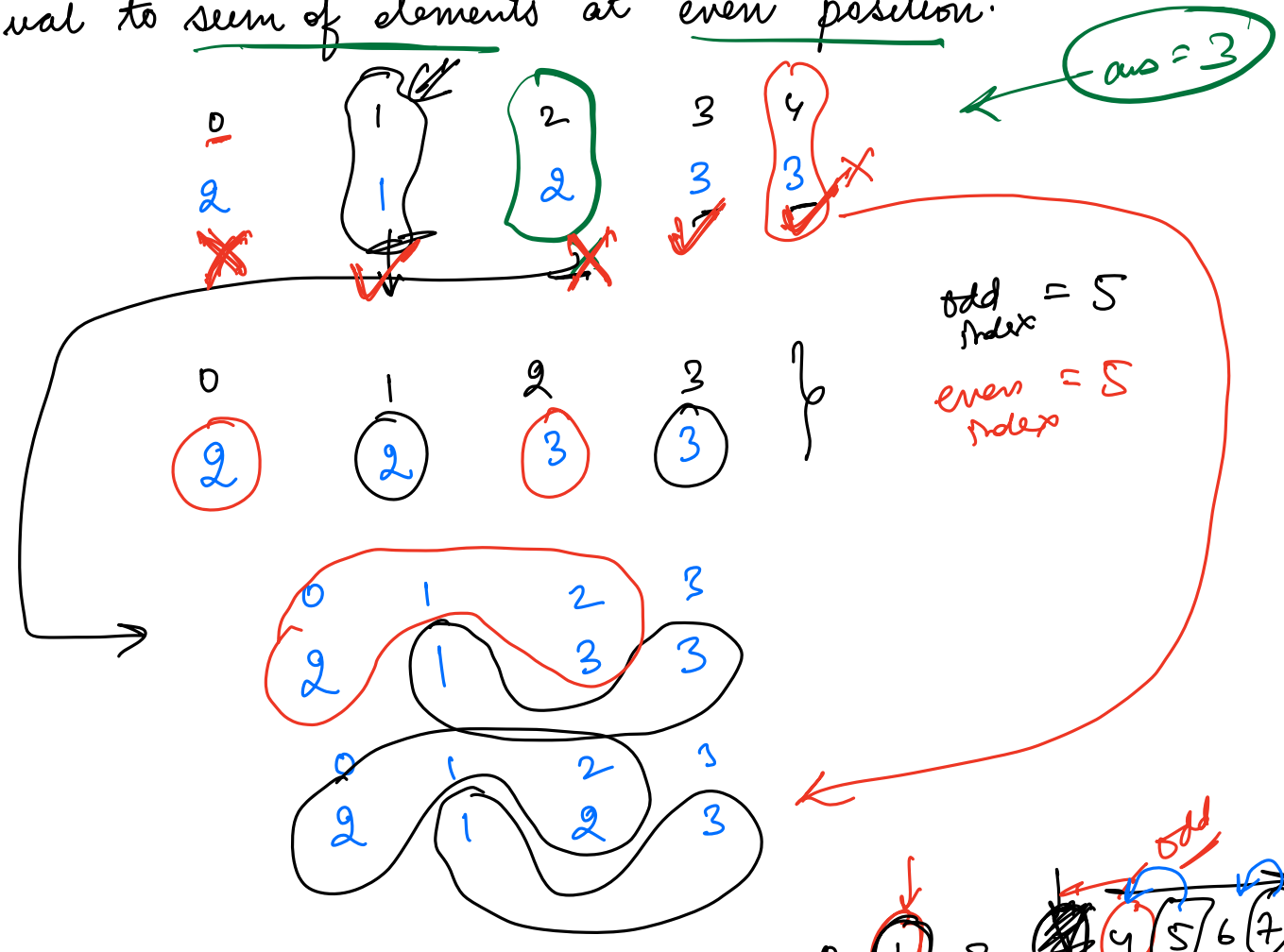
i = 0

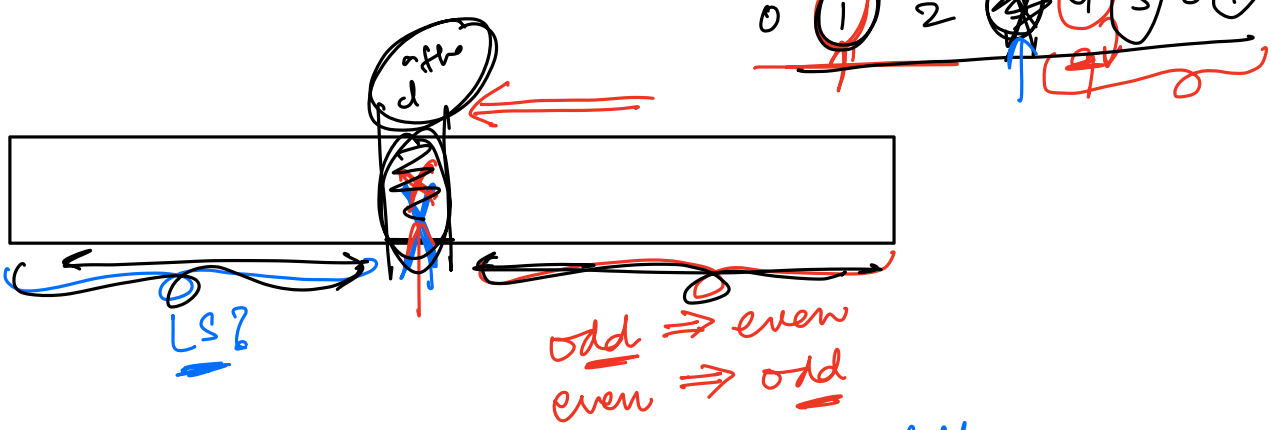
```
for i=0 → n
{   rs -= arr[i]
    if( ls == rs)
        return i;
    ls += arr[i];
}
```



Q Count Special Elements.

An element is special if after removing this element from the array sum of elements at odd position is equal to sum of elements at even position.





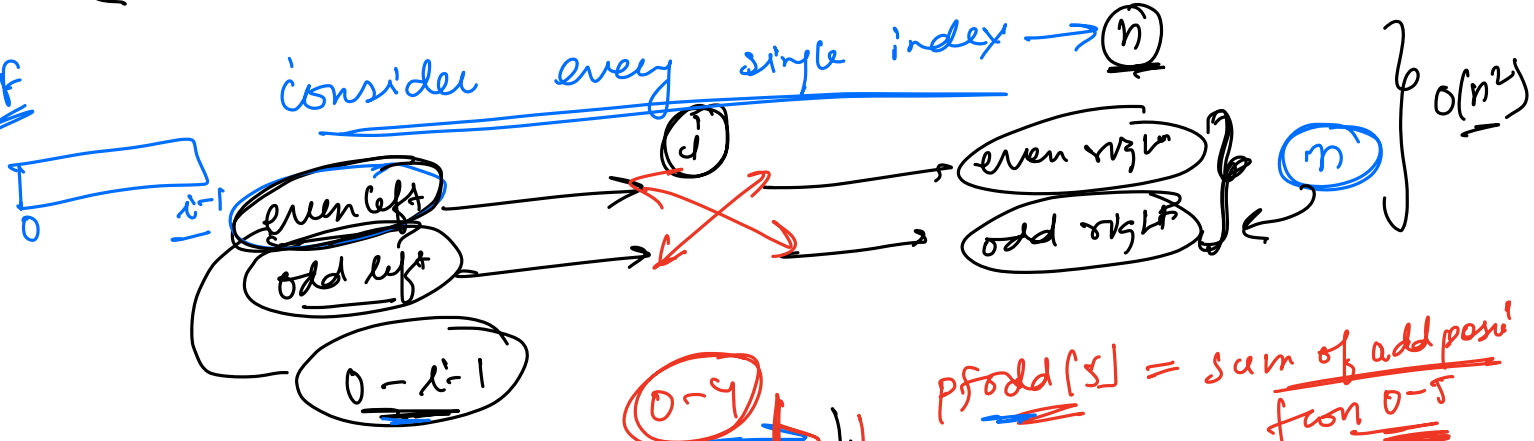
odd position sum = sum of odd on left + sum of even position on right

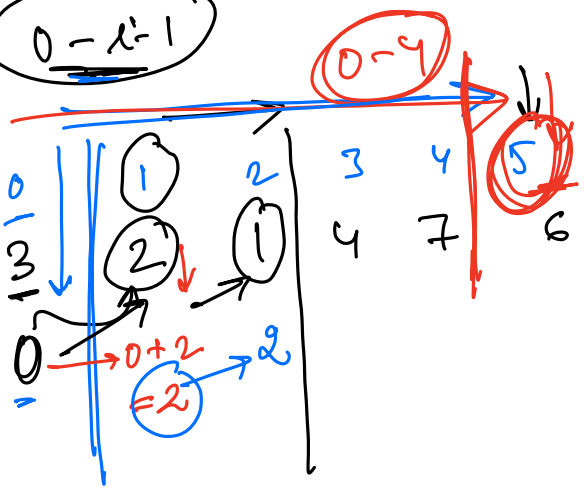
even position sum = sum of even on left + sum of odd position on right

sum of odd on right

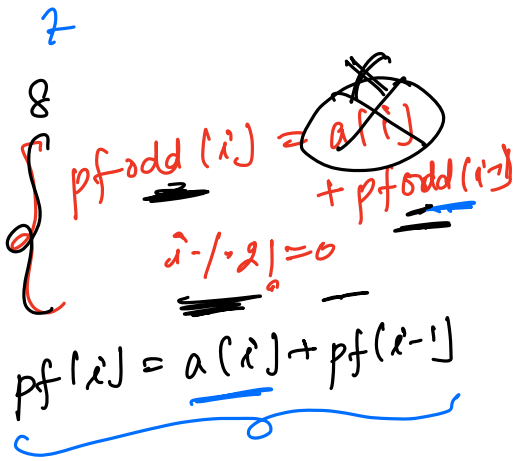
B.F

consider every single index → n

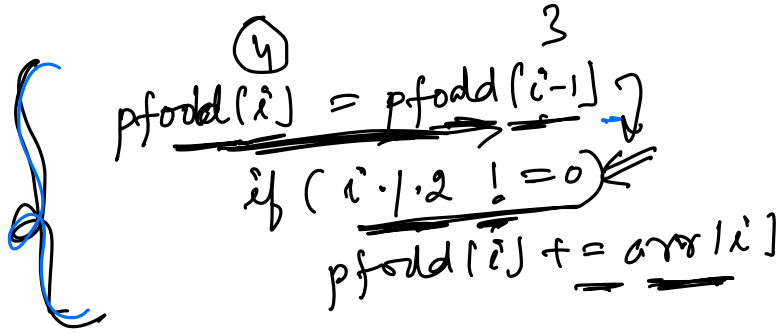




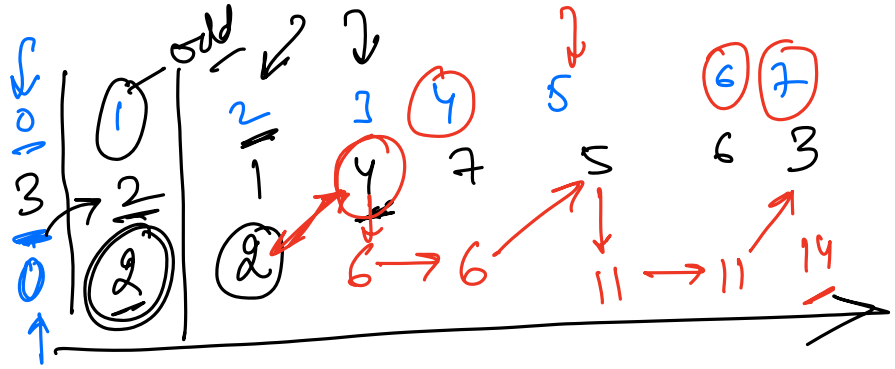
pfodd[5] = sum of odd posn from 0-5



pfodd(i)
odd positions 0-i



pfodd(3) = pfodd[4]



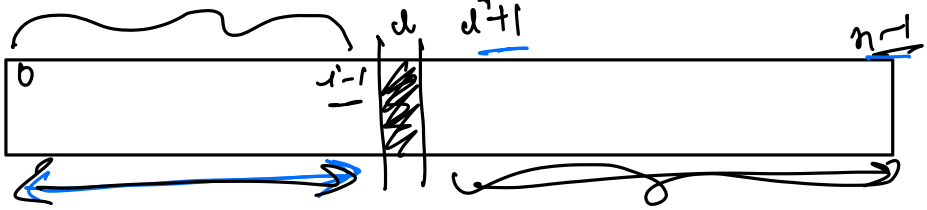
pfodd(i)

pfeven(i)

sumodd 0-i

pfodd

pfeven → sum even 0-i



odd left = pfodd[i-1] ↔ odd right = pfodd[n-1] − pfodd[i]

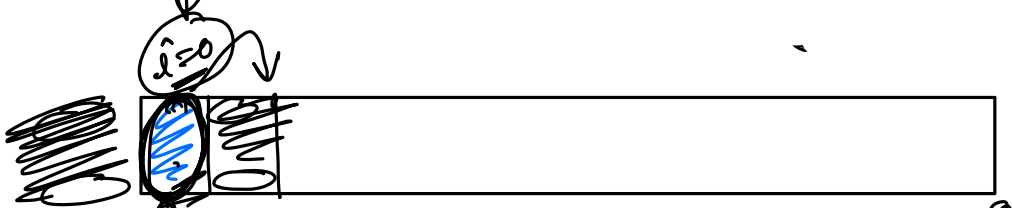
even left = pfeven[i-1] ↔ even right = pfeven[n-1] − pfeven[i]

odd left + even right

even left + odd right

T.C : O(n)

S.C : O(n)



lo = 0
le = 0

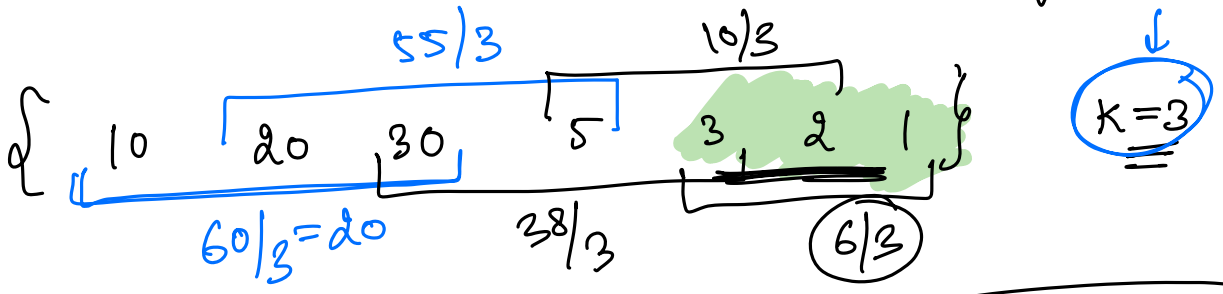
ro = sum of all odd position
re = sum of even position

```
// calculate ro & re
for (i=0 → n
{   if( i%2 != 0)
        ro += arr[i]
    else
        re += arr[i]
    if (lo + re == le + ro)
        ans++;

    if( i%2 != 0)
        lo += arr[i]
    else
        le += arr[i]
}
return ans;
```

Q Least Average value subarray

given an array of size N. Find a subarray of size K with least average.

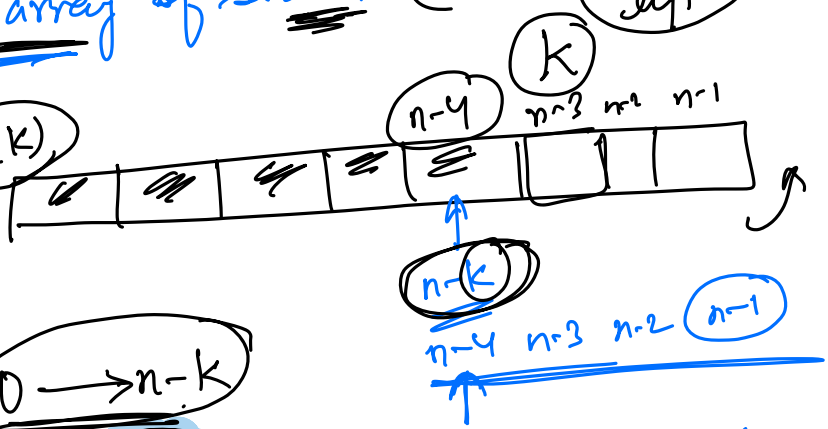


average = sum / K → minimum sum subarray

B-F → consider all subarray of size k — (n-k+1) → start idx

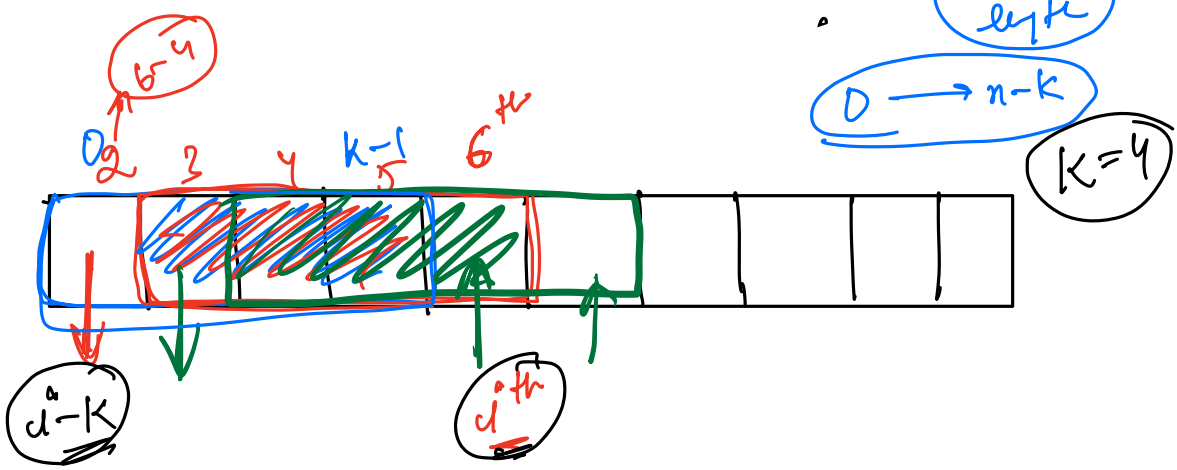
Calculate sum → O(k)

T.C: (n-k+1)(k)



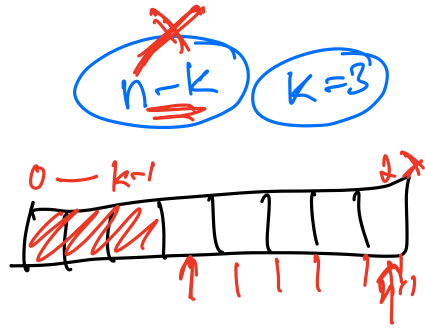
0 → n-k

n-k+1

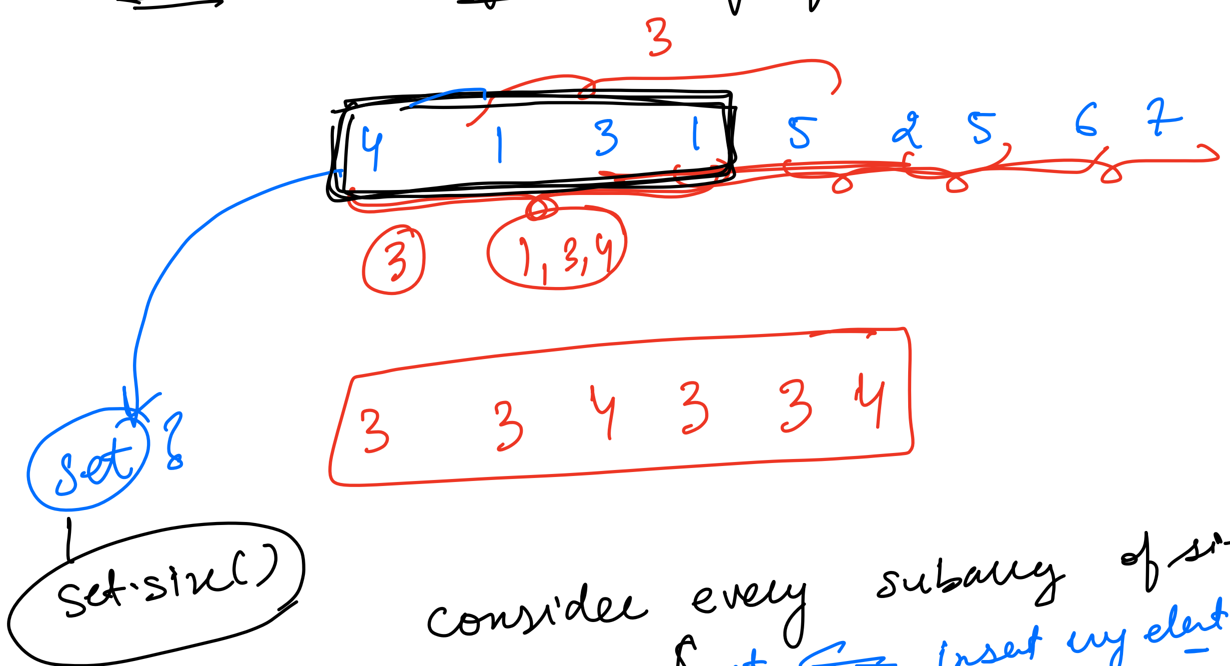


①

```
int sum = 0        ans = INT_MAX        0 → k-1
for i = 0 → i < k
    sum += arr[i]
ans = min(ans, sum);
for i = k → i < n
    sum += arr[i]
    sum -= arr[i-k]
    ans = min(ans, sum)
return ans/k;
```



Q. Given an array of size N. Find the number of distinct elements in every subarray of size k.



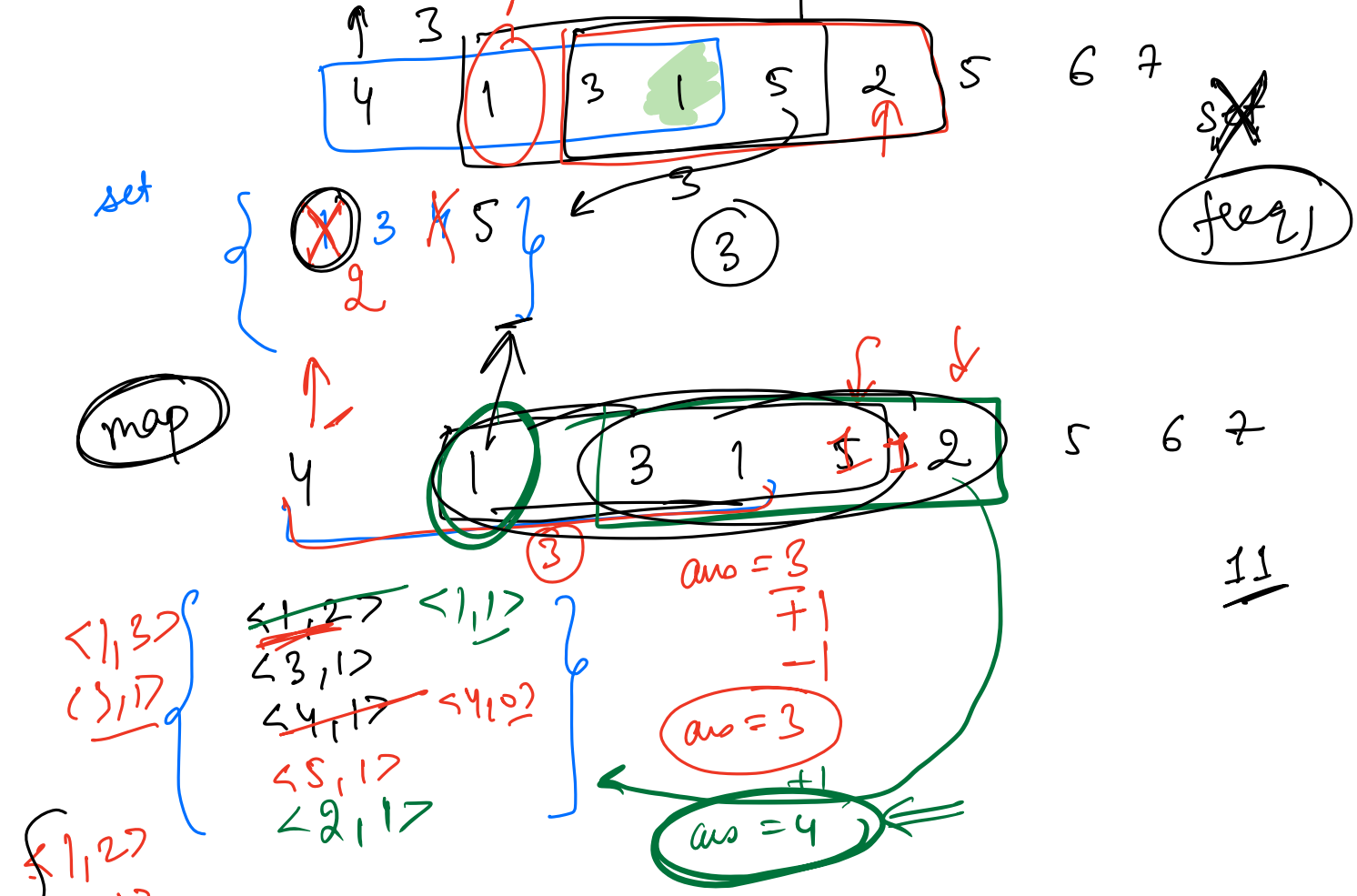
consider every subarray of size k

set ⇐ insert my elem

set.size()

O((n-k+1)·k) ⇒ ~~O(n²)~~

S.C: O(k)



map → m <int, int>
int ans = 0; arr[i]

```
for i=0 → k
{
    m[arr[i]]++;
    if( m[arr[i]] == 1)
        ans++;
}
cout << ans;
```

T.C: O(n)
S.C: O(n)
O(k)

```
for i=k → i<n
{
    m[arr[i]]++;
    m[arr[i-k]]--;
    if( m[arr[i]] == 1)
        ans++;
    if( m[arr[i-k]] == 0)
        ans--;
}
cout << ans;
```