Two Pointer

1 Binary subarrays with sum

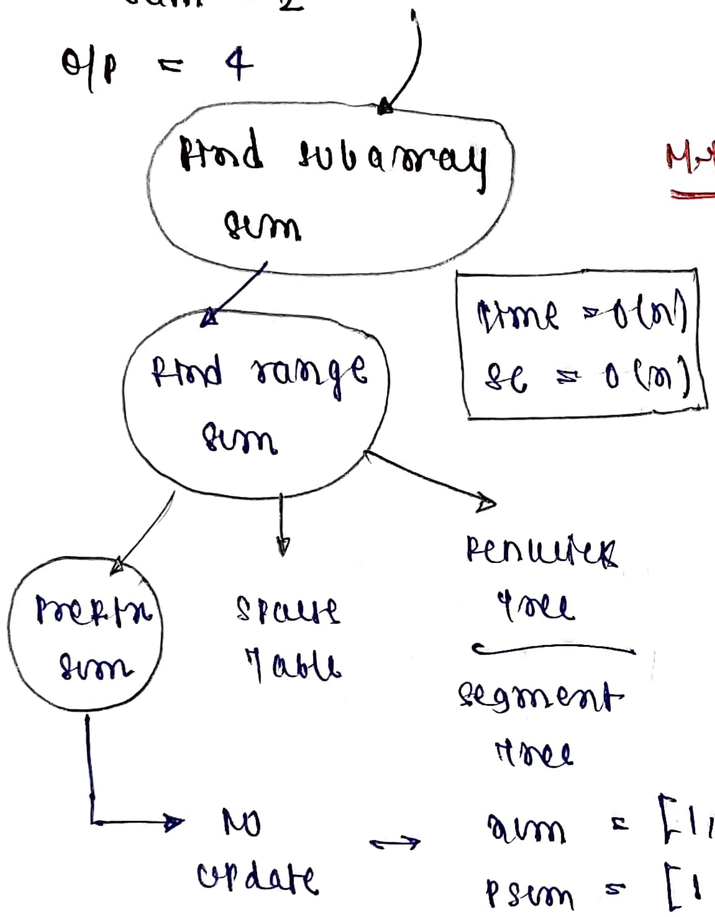
eg. nums = [1,0,1,0,1]
sum = 2
O/P = 4

M-I Find out all subarrays O(n^2)
Find sum of that subarray O(n)

O(n^3).

M-II
num = [1,2,3] sum += 1
num = [1,2,3] sum += 2
num = [1,2,3] sum += 3

Here subarray is found simultaneously so O(n^2).



num = [1,2,3] and goal = 3.
psum = [1,3,6].

eg. num[] = 0 1 0 1 0 1
psum[] = 0 0 1 1 2 2 3

• range sum = psum[j] − psum[i−1] = goal.
psum[j]: You at right now
goal: known

So, psum[i−1] = psum[j] − goal.

So, 0 0 1 1 2 2 3

• ek zero extra add kr lenge or sath ke sath jaha tak j phuchega waha se pehle sare element ko map me v dalte jayenge.

eg 1 0 1 0 1 goal = 2
psum 1 1 2 2 3
total = 4

ans = ~~0~~ ~~1~~ ~~2~~ 4
psum = ~~0~~ ~~1~~ ~~1~~ ~~2~~ ~~2~~ 3

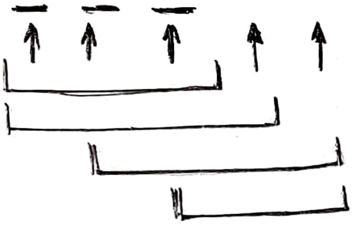
mpp
0 → 1
1 → ~~1~~ 2
2 → ~~1~~ 2
3 → 1

2 subarray sum equals k

same code for previous one will be done by it.

Now another way to solve 930 (Binary subarrays with sum)

eg 1 0 1 0 1   goal = 2



ans = ~~0~~ ~~1~~ ~~1~~ ~~2~~ ~~2~~ ~~3~~ ~~2~~ 2

count = ~~1~~ ~~2~~ ~~3~~ 4

so,
101
1010
0101
101

3 Number of zero filled subarrays [PB]

eg nums = [1,3,0,0,2,0,0]   eg. nums = [0,0,0,2,0,0]

So, total = 6   so total = 9.

so, firstly we have to find no. of zeroes continuous

[0,0,0]

$\frac{n(n+1)}{2}$

Total subarrays.

M-I Brute force

$O(N^2) \cdot O(N) = O(N^3)$

space = $O(1)$.

M-II find all zero subarrays i.e start finding no. of subarrays only when starts with 0.

Time = $O(N^2)$

space = $O(1)$.

M-III Isse pehle group nikal lenge 0 ke groups ka

eg. nums [0,0,0,2,0,0]

⇒ [3,2]

↳ $n(n+1)/2$.

M-IV Now, use previous logic but don't store zeroes count, direct compute.

4 Boats to save people

2 person wt ≤ limit

1 person ≤ limit

eg people = [3, 2, 2, 1]

limit = 3

sort this so, 1 2 2 3

now
↑ l ↑ r

1st $p[l] + p[r] <= limit$

$1 + 3 \le= 3$ ✗ cnt++

so, only place right person. (r--)

2nd $1+2 \le= limit$

yes cnt++;

r--

l++

3rd $2+2 \le= limit$

yes r++ cnt++;

TC = O(nlogn)

SC = O(1).

5 Maximum points you can obtain from cards

arr[] = [6, 2, 3, 4, 7, 2, 1, 7, 1]

k = 4

- Pickup either from front and also from back else nothing.

so,

6 2 3 1 = 12

6 2 3 4 = 15

6 2 1 7 = $\boxed{16}$ yes best

brute way

4	0
3	1
2	2
1	3
0	4

→ Good

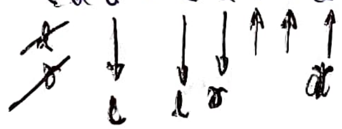
lsum	rsum	sum
= ~~0~~ ~~15~~	= ~~0~~	15.
~~11~~	1	12
9	8	$\boxed{16}$
6	9	15
0	11	11

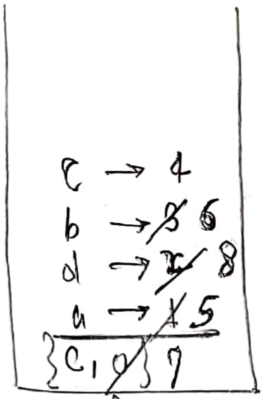
store this and give o/p.

6 Longest substring without repeating characters.

~~0 1~~ 2 ~~3 4 5 6 7 8~~

Eg. s = c a d b c a b c d





map {char, index}

maxlen = ~~0 1 2 3 4~~ 5

```
func (string s) {
    hash [256] = [-1]
```

array hai ye

```
    l=0 , r=0 , maxlen =0
    n = s.size ();
    while (r < n) {
        if ( hash [ s[r] ] != -1) {
            if ( hash [s [r] ] >= l) {
                l = hash [ s[r] ] +1;
            }
        }
        len = r-l+1;
        maxlen = max (len , maxlen) ;
        hash [ s[r] ] =r
        r++ ;
    }
    return maxlen ;
```

→ TC = O (N)

→ SC = O (256) .

9 Max consecutive ones III

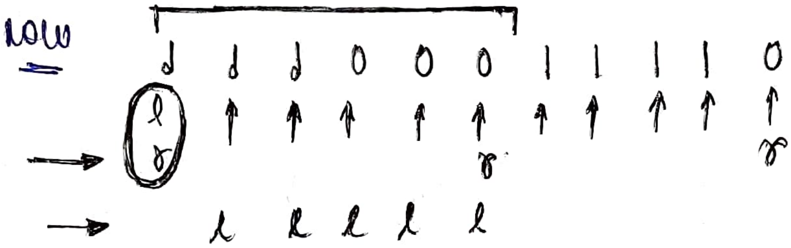
arr[] = [1 1 1 0 0 0 1 1 1 1 0]   k = 2

allow to flip at most k zeroes.

↳ o/p = 6

↓

• Find longest subarray with max zeroes as k.



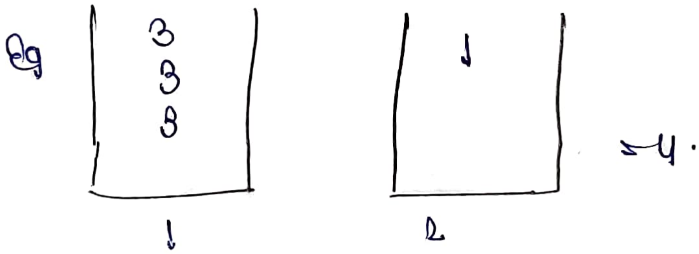
zeroes = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~2~~ ~~3~~ 2

maxlen = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6

Ans

8 Fruits into Baskets.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

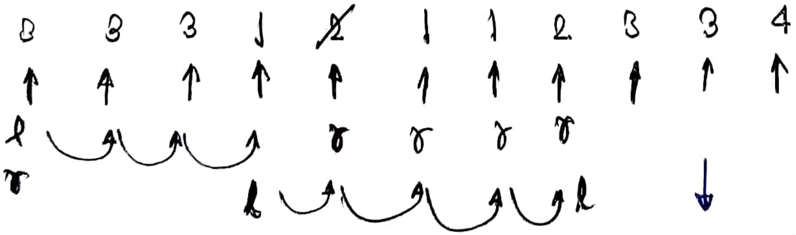


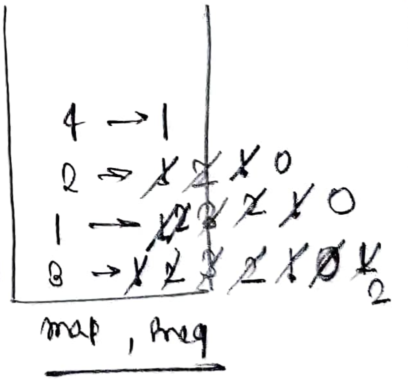
only 2 buckets, that only stores similar types of thing.

↓

• max length subarray with at most two types of numbers.

Eg 1 1 1   2 2 = 5





TC = O(N+N)

≅ O(N)

O(2N)

SC = O(3)

maxlen = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

• One another way to optimise this will be simply do the subarray and just store in map when map size == 2 so return ans.

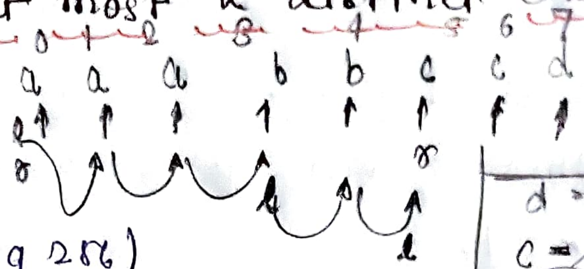
↳ max consecutive ones III.

9 Longest substring with at most k distinct characters

Eg. s = aaabbccd

k = 2.

Eg.



↓

TC = O(N) + O(log 256)

SC = O(256).

d - 1

c - ~~1~~ 2

b -> ~~2~~ ~~1~~ 0

a - ~~3~~ ~~2~~ ~~1~~ 0

maxlen = ~~0~~ ~~1~~ ~~2~~

~~3~~ ~~4~~ 5

10 Number of substrings containing all the 3 characters.

S = b b a c b a

len = ~~1~~ 2

Sliding window

01 Longest substring without repeating char.

Eg. p w w k e w

Smp ans = pw

~~ans~~

p w w k e w

map

~~p → 1~~

w → 1

ans = 3

```
for ( —— ) {
    while (mp (s[i]) > 0) (
        ans = i-j
        mp [s[j]] --;
        j--;
    }
    mp [s[i]] ++;
```

s/e

0	1	2	3	4	5
p	w	w	k	e	w
s	s	~~e~~ ~~s~~ s	~~e~~	~~e~~	e

map

p → ~~1~~ 0

w → ~~1~~ ~~0~~ ~~1~~ ~~0~~ 1

k → 1

e → 1

ans = ~~0~~ ~~2~~ 3

```
int fun (string s) {
    int start = 0, end = 0, ans = 0;
    unordered_map <char, int > mp;
    while (end < s.length ()) {
        while (mp [s [end] ] > 0) {
            ans = max (ans, (end - start ));
            mp [s [start ] ] --;
            start ++;
        }
        mp [s [end ] ] ++;
        end ++;
    }
    ans = max (ans, (end - start));
    return ans;
}
```

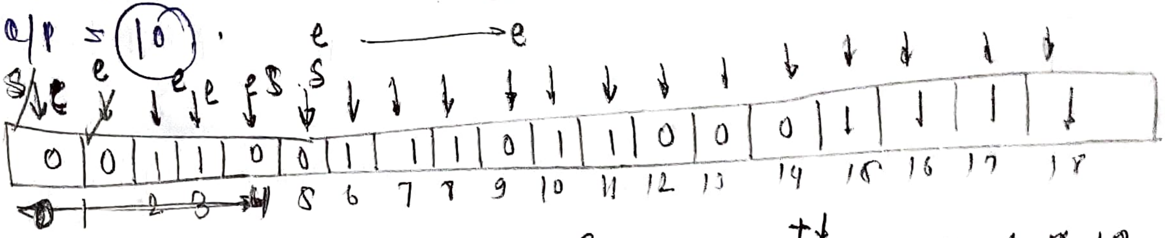
02 max consecutive ones.

eg. nums = [0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1]

ans.

k = 3

o/p = 10 .



k = ~~3~~ ~~2~~ ~~1~~ ~~0~~ ~~-1~~ 0

ans = (head - tail + 1) = ~~5~~ ~~7~~ ~~8~~ 10

```
int func ( vector <int > nums , int k) {
    int start = 0 , end = 0 , ans = 0 ;
    while ( end < nums . size () ) {
        if ( nums [i] = = 0 ) {
        }   k-- ;
        while ( k < 0 ) {
            if ( nums [start] = = 0 ) {
            }       k++ ;
        }   start ++ ;
        ans = max ( ans , ( end - start + 1 ) ) ;
        end ++ ;
    }
}   return ans ;
```

03 Get equal substrings within budgets.

s = 'abcd'
t = 'bcdf'
maxcost = 3

```
int equalSubstring ( string s, string t,
                           int maxcost) {
    int i = 0, j = 0 ;
    int cost = 0 ; int ans = 0;
    while (j < s.length ()) {
        cost = cost + abs ( s[j] - t[j]);
        if ( cost > maxcost) {
            cost = cost - abs (s[i] - t[i]);
            i++;
        }
        ans = max (ans, (j + 1 - i));
    } j++;
    return ans;
```

04 subarray product less than k.

yaha subarray ka wgte check krna hai bs.

10	5	2	6

k = 100.

10 ✓
10 5 ✓
5 ✓

5 2 ✓
5 2 6 ✓
2 ✓

2 6 ✓
6 ✓

Total 8 subarrays.

```
int count (vector <int> nums,
                    int k) {
    int i = 0, j = 0, ans = 0,
        prod = 1;
    if (k <= 1) return 0;
    while (j < nums.size ()) {
        prod = prod * nums[j];
        while (prod >= k) {
            prod = prod / nums[i];
            i++;
        }
        ans = ans + (j - i + 1);
        j++;
    }
    return ans;
```

yehi subarray de rha hai.

5 Maximum erasure value

eg [4 2 4 5 6]
17

hme value erase kro jo unique ho array me uska max sum return krna hai.

```
int i = 0, j = 0, sum = 0, ans = 0;
unordered_map < int, int > mp;
while ( j < nums.size ()) {
    while ( mp [nums [j]] > 0) {
        mp [nums [i]] --;
        sum = sum - nums [i];
    }   i++;
    sum = sum + nums [j];
    ans = max (ans, sum);
    mp [nums [j]]++;
}   j++;
return ans;
```

6 Longest repeating character replacement

eg s/e
A A B A B B A
k = ~~1~~ 0
ans = A AA / B 4

```
int i = 0, j = 0, ans = 0;
vector < int > count (26, 0);
while (j < s.length ()) {
    count [s [j] - 'A'] ++;
    while ( (j - i + 1) - (*max_element
        (count.begin (), count.end ())) > k) {
        count [s [i] - 'A'] --;
        i++;
    }
    ans = max (ans, j - i + 1);
    j++;
}
return ans;
```

good que
logic.

07 Minimum size subarray sum

Eg. nums = [2,3,1,2,4,3]

target = 7

```
int i = 0, j = 0, sum = 0;
int ans = INT_MAX;
while (j < nums.size()){
     sum = sum + nums[j];
     while (sum >= target){
          ans = min(ans, (j-i+1));
          sum = sum - nums[i];
          i++;
     }
     j++;
}
if (ans == INT_MAX)
return 0;
return ans;
```

qus Minimum operations reduce to zero.

nums = [1, 1, 4, 2, 3]

5 - 3 = 2

2 - 2 = 0

Eg [4 | 2 | 4 | 2 | 3]

5 - 3 = 2

2 - 2 = 0

Eg [6 | 2 | 1 | 1 | 5 | 5]

x = 10

- 6 - 2 - 1 - 1 = 0

4 steps

- 5 - 5 = 0.

- yaha ye kro sbse bada size ka subarray chuno jiska length max ho or sum total - k ho.

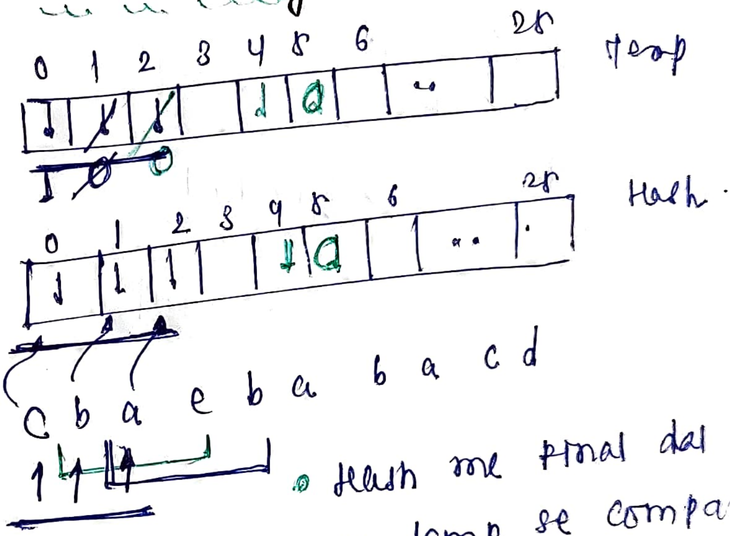
qus find all anagrams in a string

s = c b a e b a b a c d

0 1 2 3 4 5 6 7 8 9

[cba] [bae]

so, [0, 6]



- Hash me final dal do, or temp se compare karte sho temp se 0/1 pe switch kre sho.

qus longest subarray of 1's after deleting one element

↓

Eg [1, 1, 0, 1]

o/p = 3.

↓

same as max consecutive ones III

k = variable

yaha k = 1 or ans - 1.

qus count subarrays with score less than k

Eg.

0	1	2	3	4
2	1	4	3	5

```
sum = sum + nums[j]
while (sum * (j-i+1) >= k) {
    sum = sum - nums[i];
    i++;
}
ans = ans + (j-i+1);
} j++;
```

ques fruit into baskets.

eg [1,2,1] we have only 2 baskets

o/p : 3

isme se v maximum subarray chunna hai jiska length sble jada ho & tree hi aa jaye.

ques Minimum consecutive cards to pick up.

almost same.

ques frequency of most frequent element

eg [1 2 4] k=5.

So, $\cancel{1}\ \cancel{2}\ \cancel{3}\ 4 \rightarrow 3$ $\quad$ $\boxed{5}$

$\cancel{2}\ \cancel{3}\ 4 \rightarrow 2$

4 $\quad$ ↳ [4, 4, 4]

$\boxed{o/p = 3}$

ques count the no. of good subarrays

ques Number of zero filled subarrays.

ques No. of smooth descent periods of stock.

same ques.