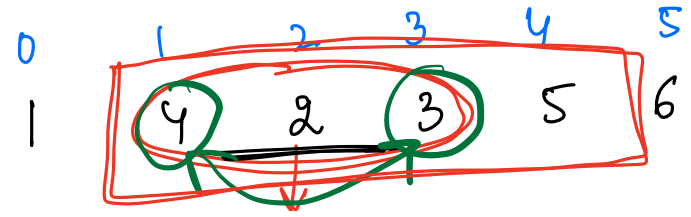
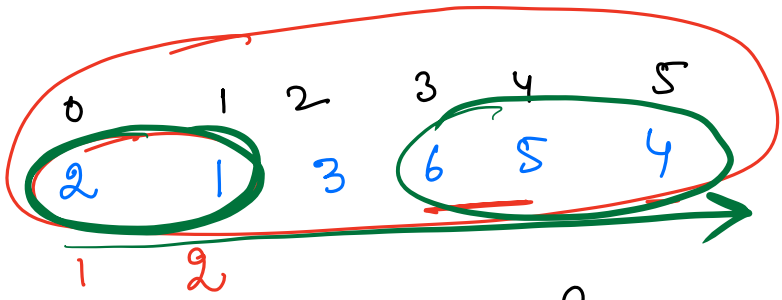
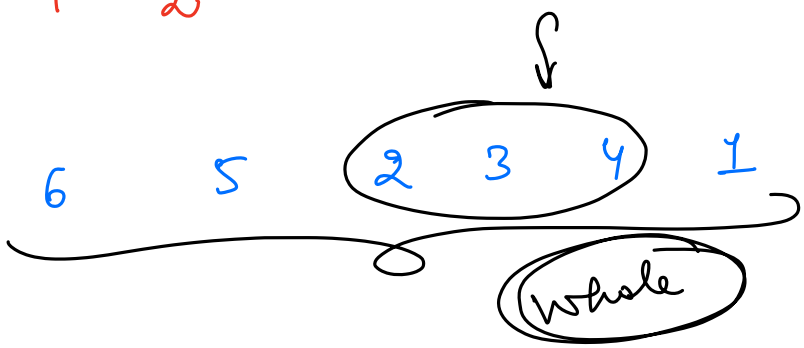
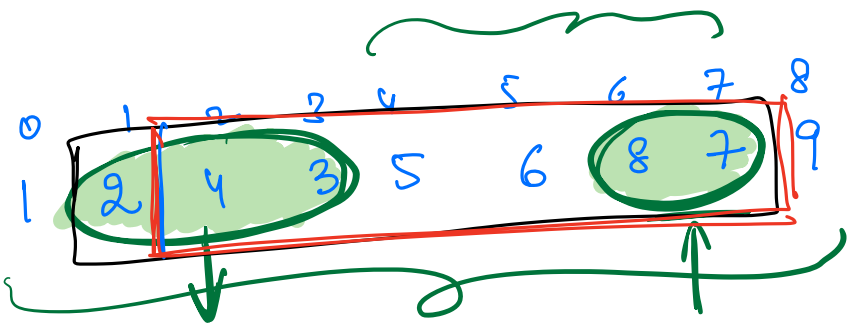
① You are given with an array of size N.
Find minimum length subarray such that if you sort this subarray whole array would be sorted. ex

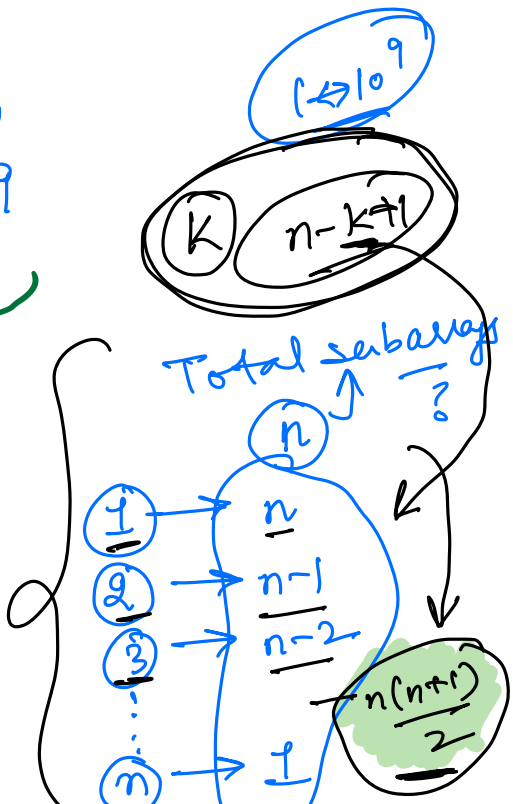


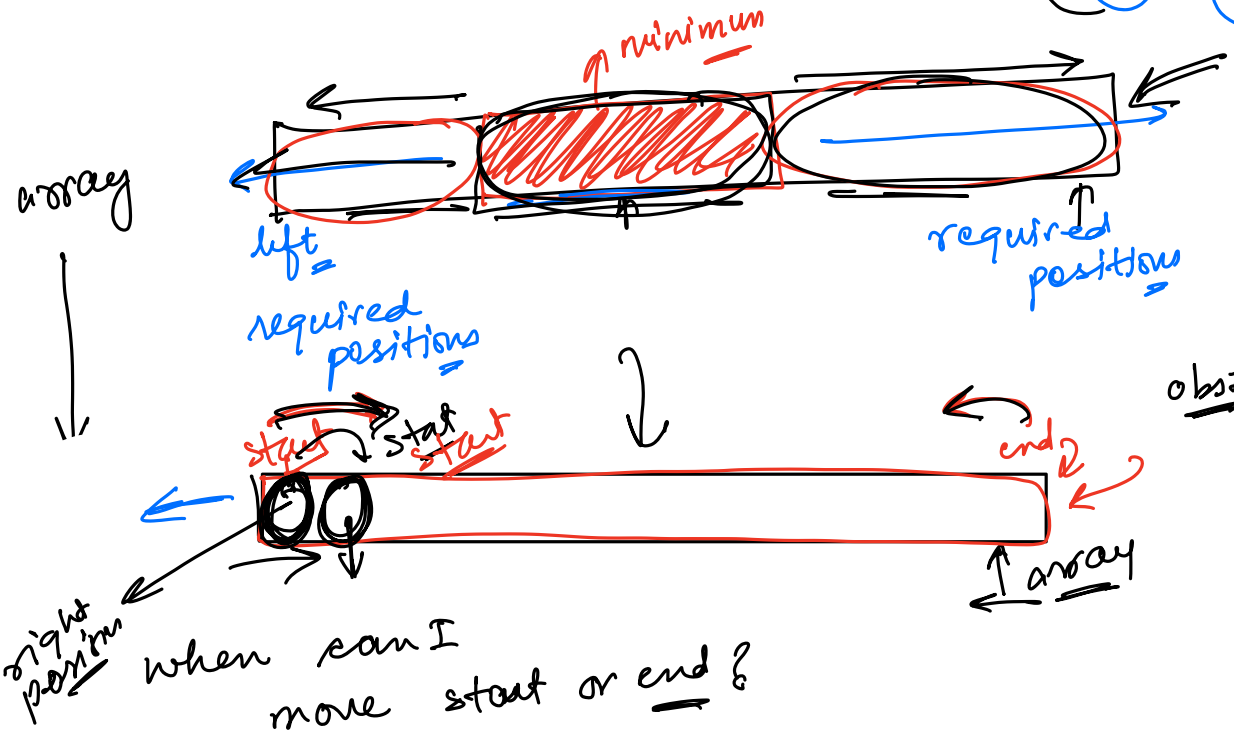
1 2 3 4 5 6 sorted ? Yes







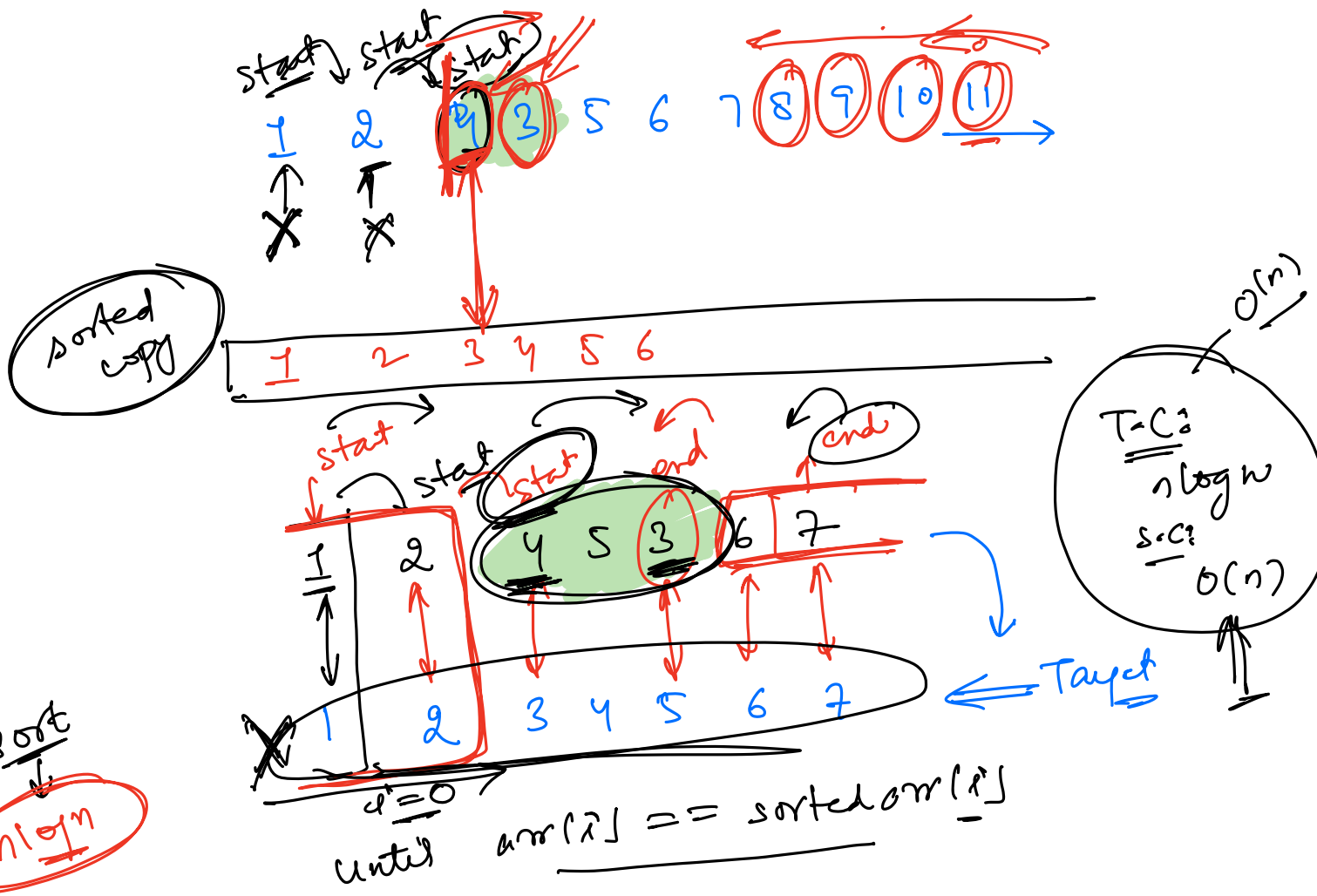




when can I move start or end ?

get a copy and sort it

arr[i] == sorted arr[i] — right position



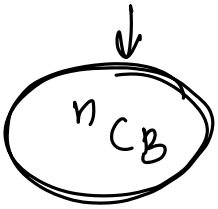
1) copy the given array →

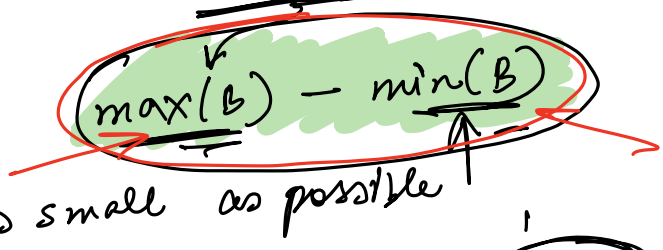
2) sort the copy

```
int start = 0;   end = n-1;

while( start < n )
{
    if( arr[start] == sorted arr[start] )
        start++;
    else
        break;
}
// write the loop for end also
```

Q2 N elements ⇒ choose B elements





B = 4

10 15 6 9 13 18 →

max = 15, min = 6 } (9)

1) 15 6 9 13

max = 18, min = 10 } (8)

2) 10 15 13 18

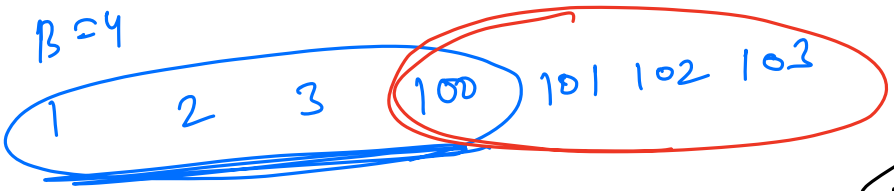
max = 13, min = 6 } (7)

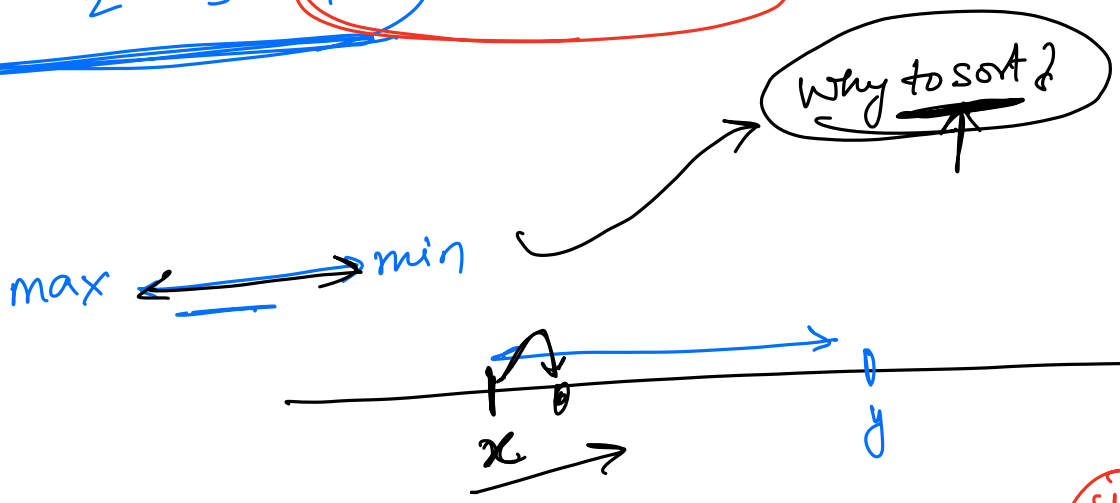
3) 10 6 9 13

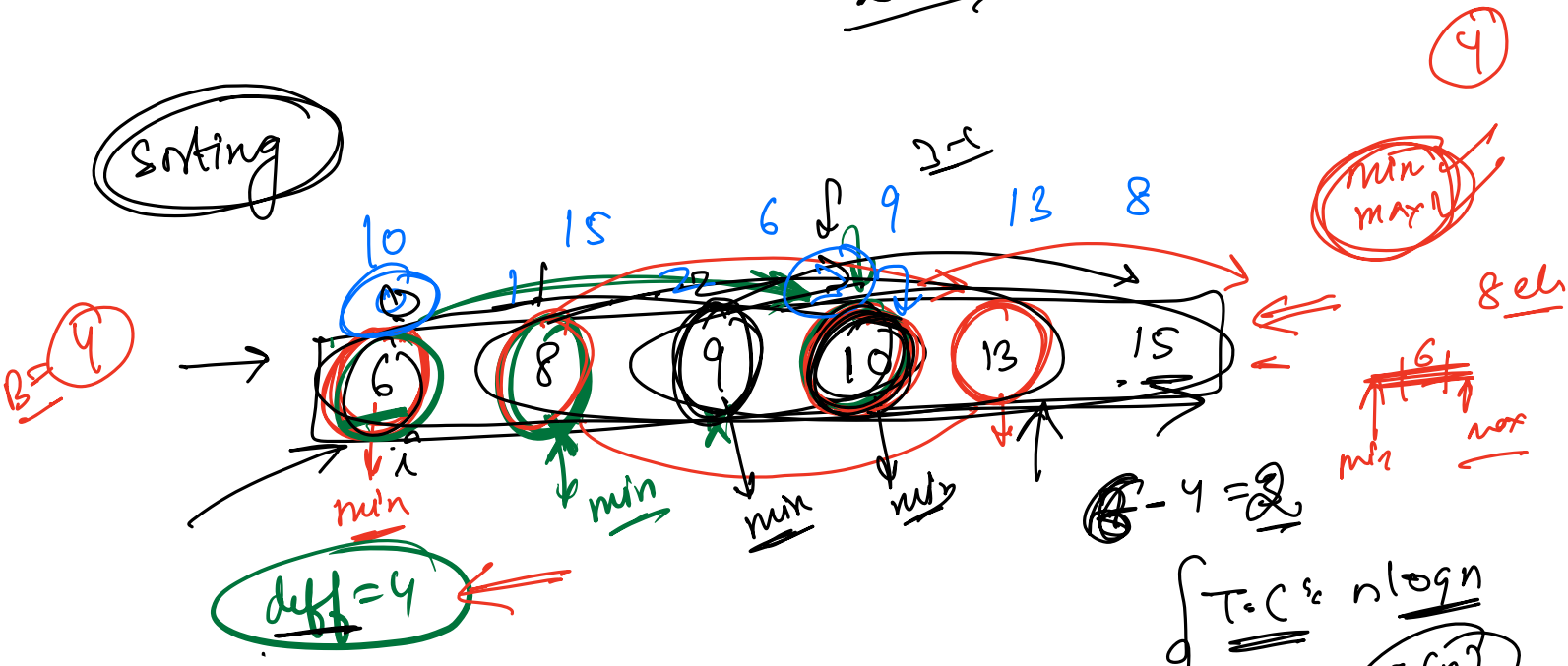
max = 15, min = 9 } (6)

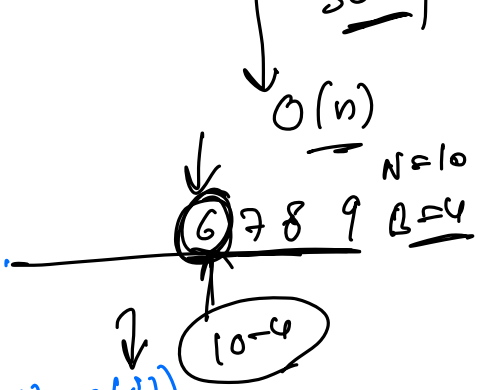
4) 10 9 13 15

sort & take first B elements









1) sort the array

diff = INTMAX

for $i = 0 \rightarrow n-b$ ($n-b+1$)

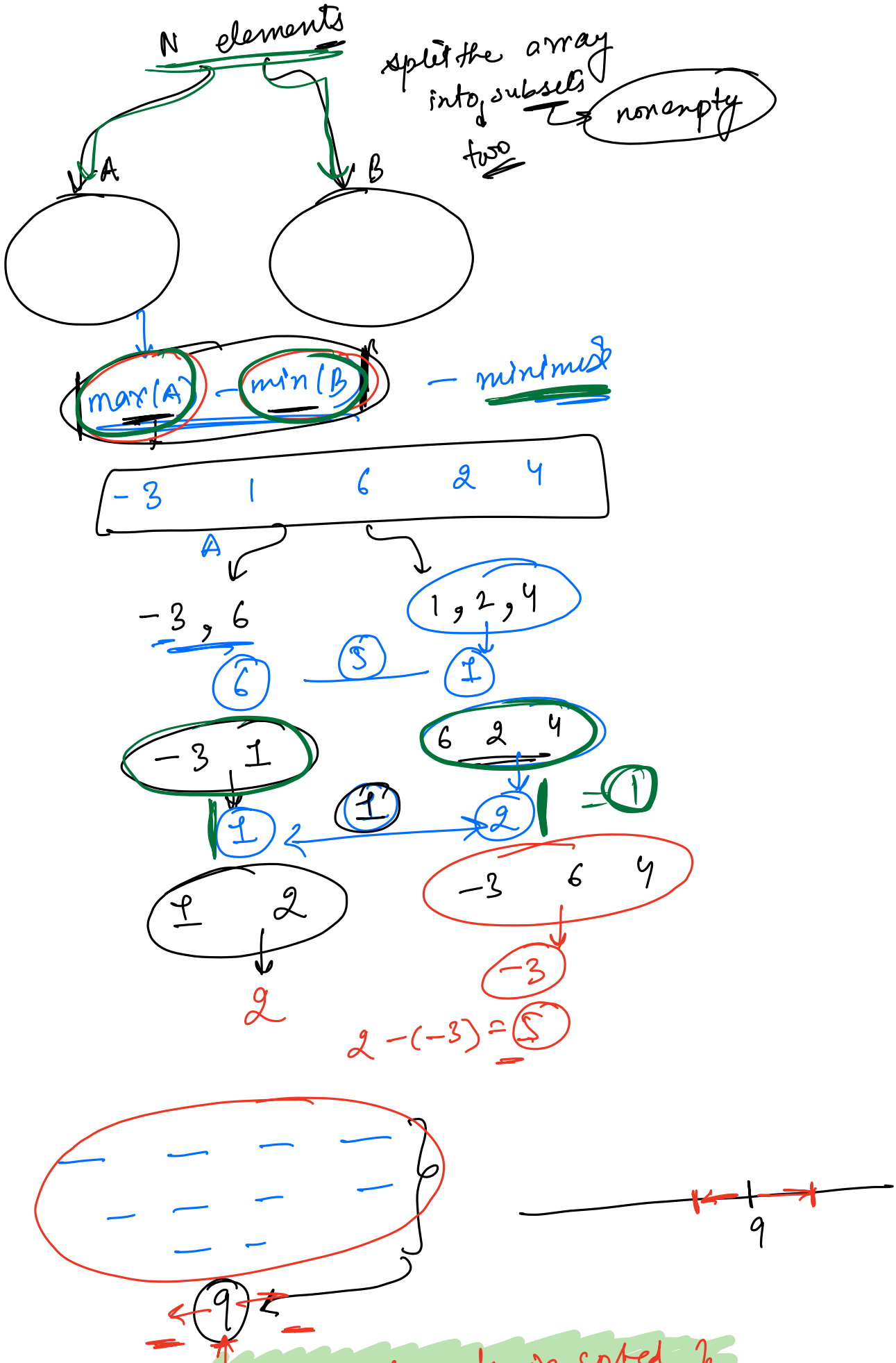
diff = min {diff, a[i+b-1] - a[i]}

return diff

S.C: O(1)

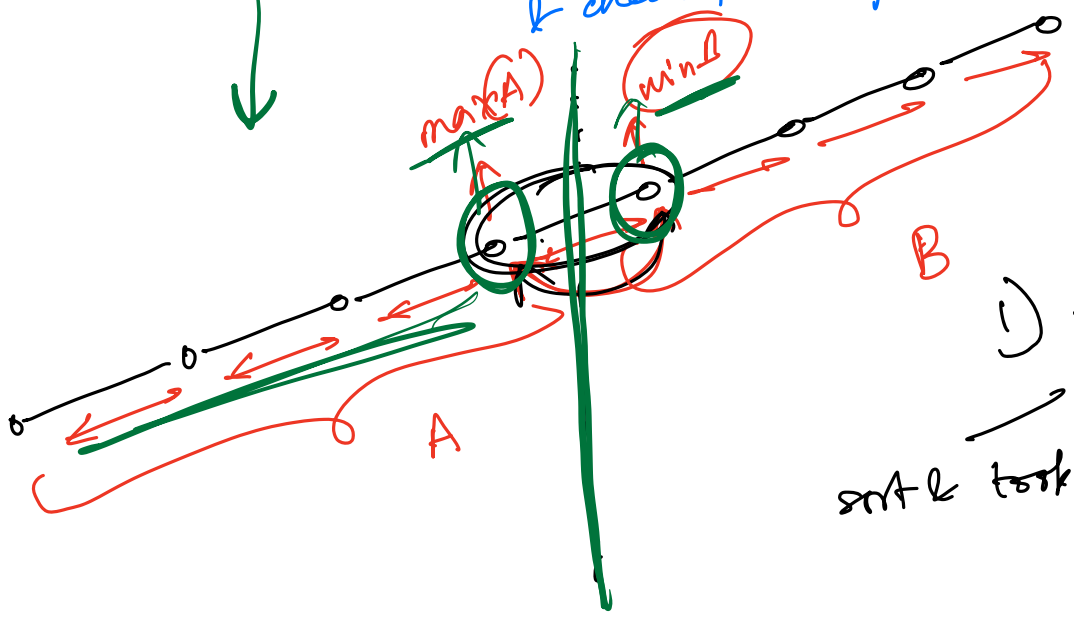
T.C: O(n)

Q3


N elements
split the array into subsets
two
non empty
A
B
max(A) - min(B)
minimum
-3 1 6 2 4
A
-3, 6
1, 2, 4
6
5
1
-3 1
6 2 4
1
1
2
1
-3 6 4
1 2
-3
2
2 - (-3) = 5
9
9
min diff = adjacent elements in sorted manner

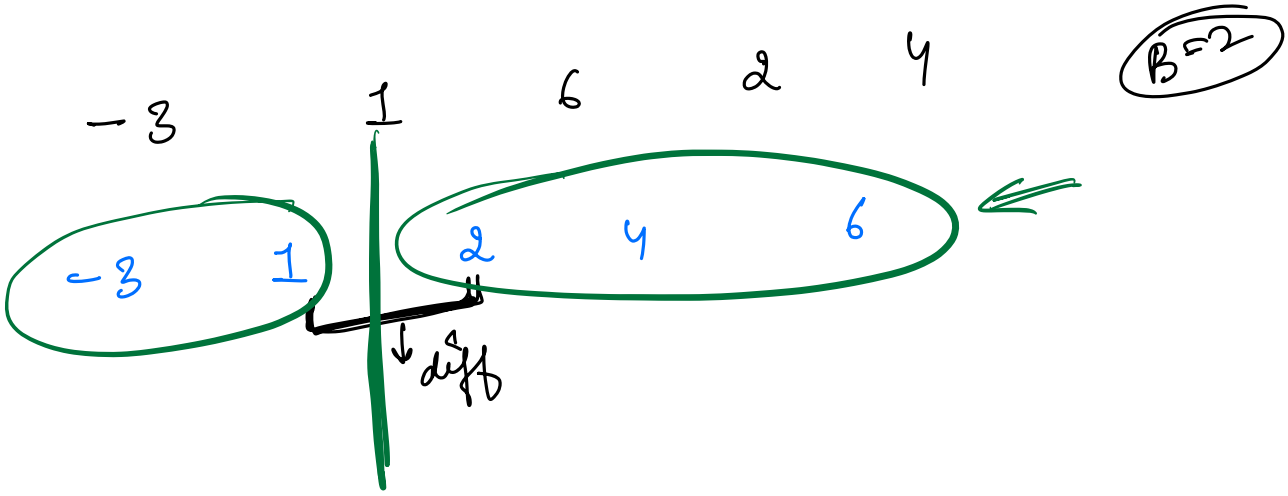
array → min diff b/w any pair

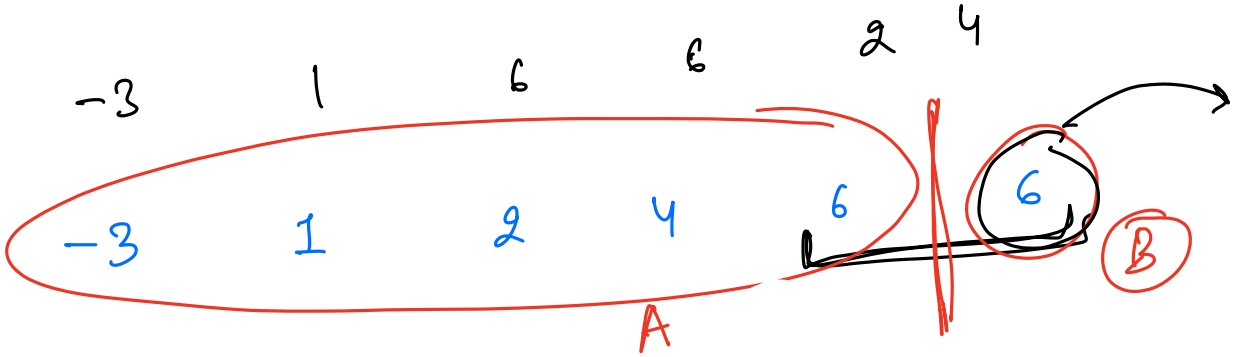
↳ sort the elements & check the adjacent ele



1) find what is min possible diff

sort & took adjacent ele





1) sort the array

diff = INT_MAX

ind = -1

i = 0 → i < n-1

if(a[i+1] - a[i] < diff)

{ T.C: n log n
S.C: N

diff = a[i+1] − a[i];

ind = i;

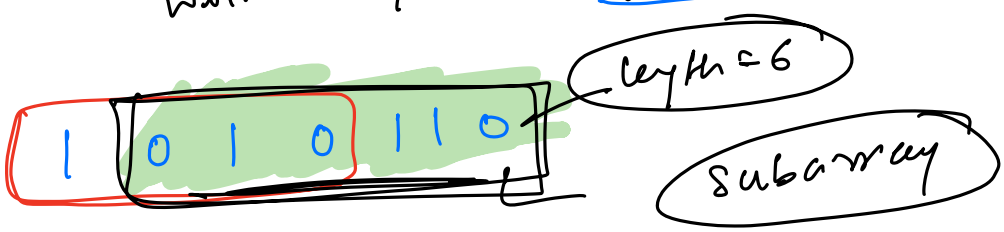
p

0 – ind — A

ind+1 → n−1 — B

(4) size N – array — 0's and 1's ← all the elements

Find the length of longest subarray with equal no of 0's and 1's



B-F →

consider all subarrays

for i = 0 → n

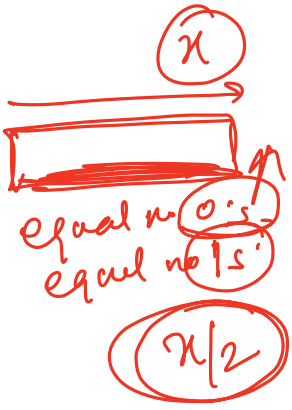
for j = i → n

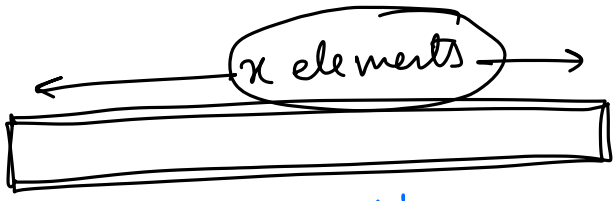
[i, j]

for k = i → j

n^3

p





no 0's = no 1's

a a

a + a = x

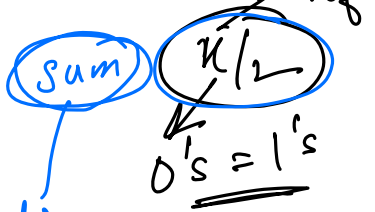
a = x/2

x/2

sum

prefix sum

O(1)



int ans = 0;

n^2 { for i = 0 → n
for j = i → n }

[i ... j]

len = j - i + 1 — O(1)

sum = len / 2 — O(1)

exact_sum = pf[j] - pf[i-1] — O(1)

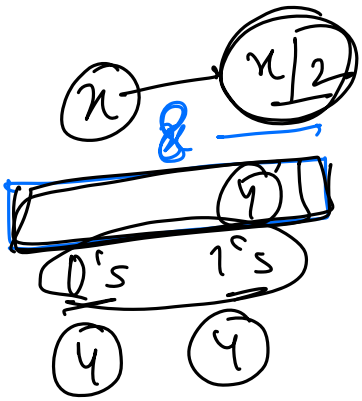
if(sum == exact sum) — O(1)

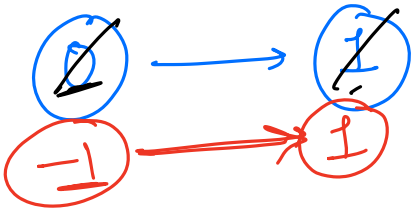
ans = max(ans, len) — O(1)

pf — O(1)

$n^3 \rightarrow n^2$

S.C: O(n)





subarray

equal no 0's = equal no' 1's

sum = ?

replace every 0 with -1 subarray

sum = 0 → equal no 0's & 1's

i, j

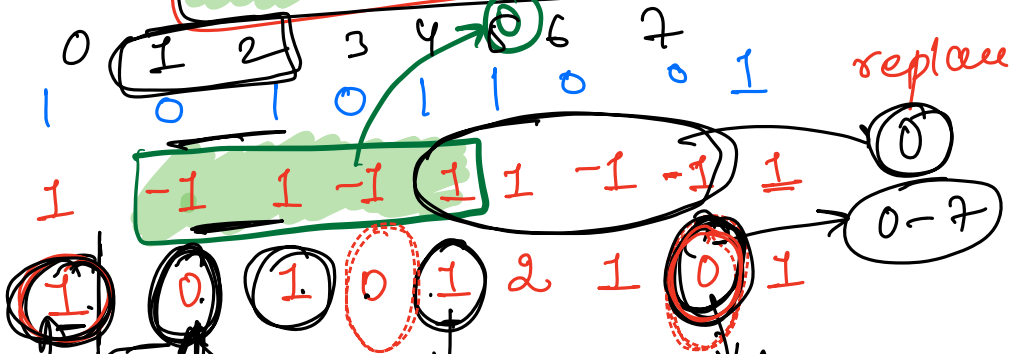
pf[j] - pf[i-1] = 0

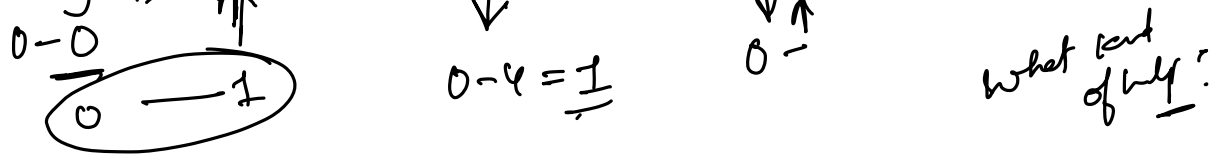
pf[j] = pf[i-1]

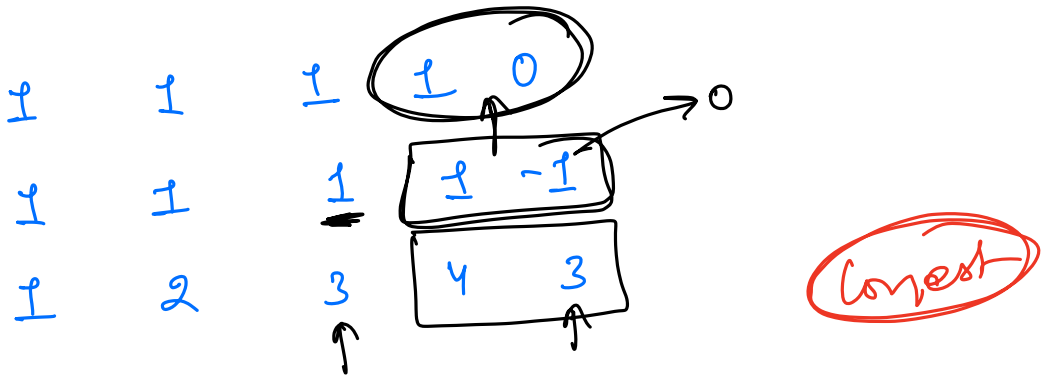
i - 4

pf[4] - pf[0] = 0

pf[4] = pf[0]



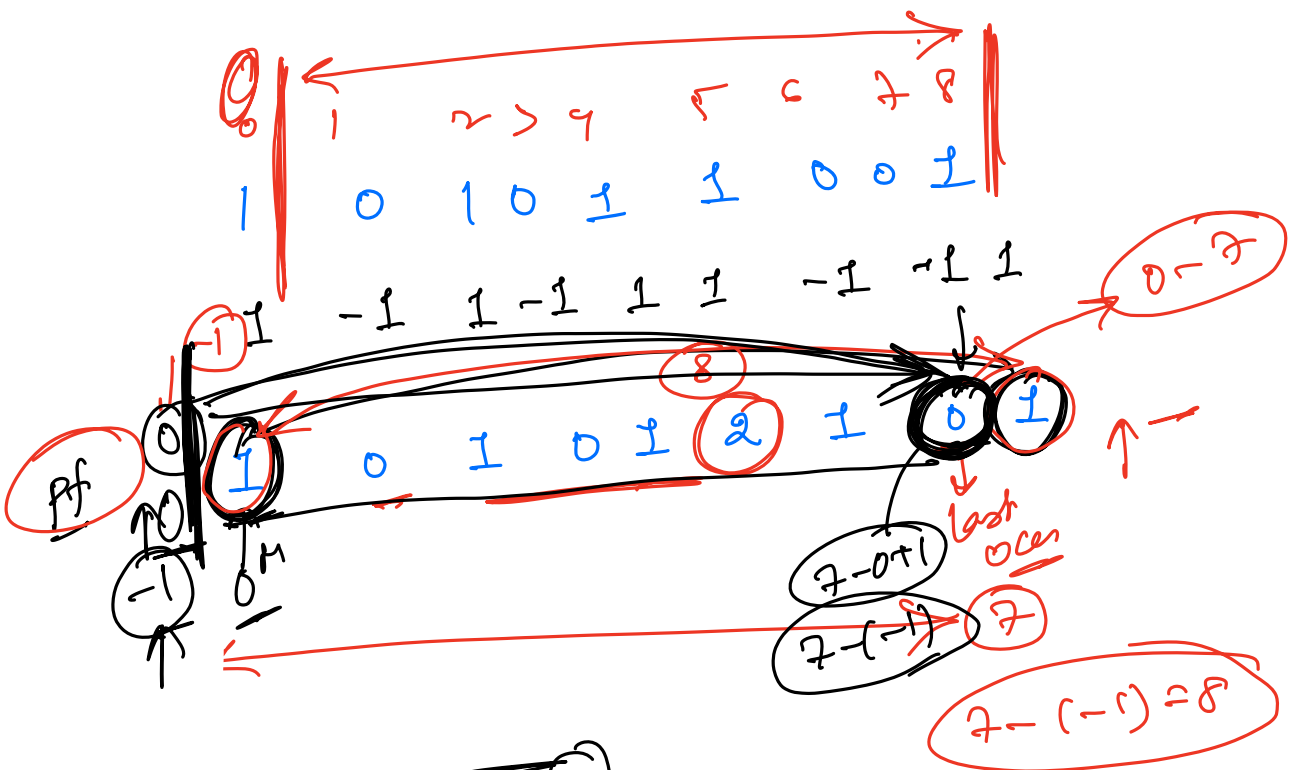


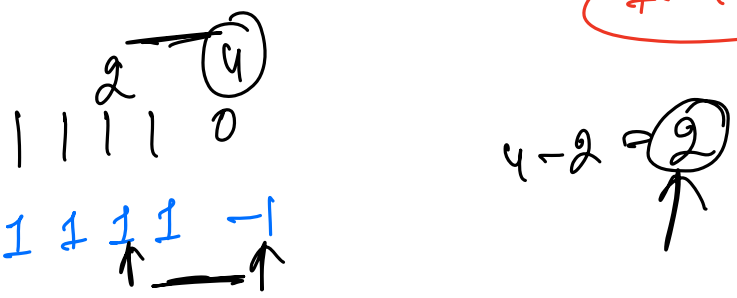


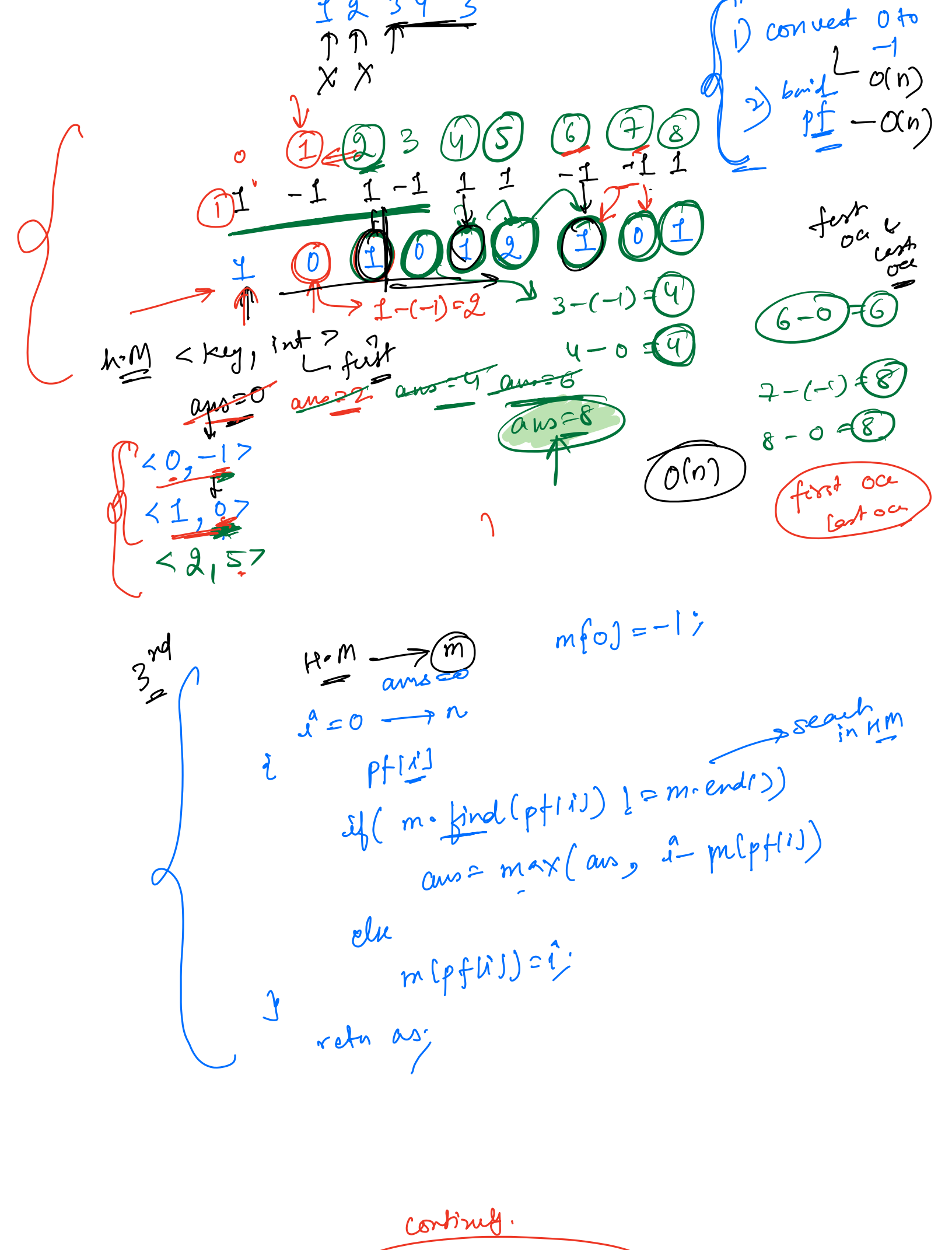
1) subarray sum = 0, b/w two duplicates in pf array

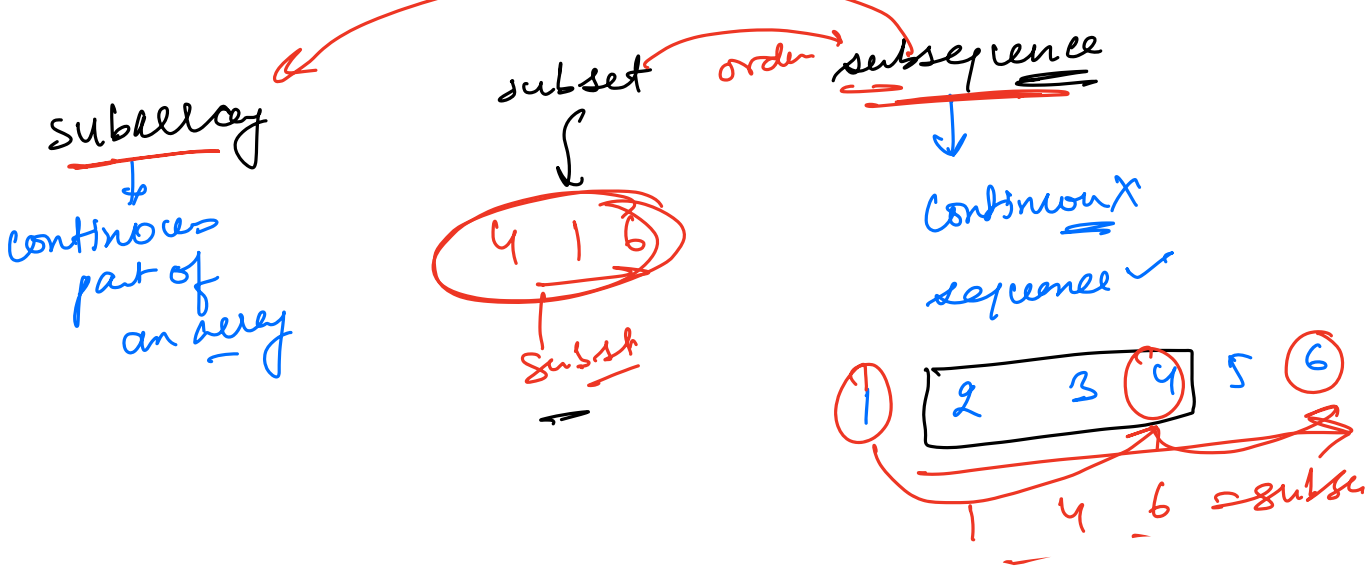
2) subarray sum = 0 (0 - i) if pf[i] = 0

equal no 0's & 1's






1) convert 0 to -1
O(n)
2) build pf — O(n)
first occ & last occ
6-0=6
7-(-1)=8
8-0=8
first occ
last occ
O(n)
h.M <key, int>
ans=0
ans=2
ans=4
ans=6
ans=8
<0,-1>
<1,0>
<2,5>
3-(-1)=4
4-0=4
1-(-1)=2
m[0]=-1;
H.M → m
ans=0
i=0 → n
pf[i]
search in HM
if(m.find(pf[i]) != m.end())
ans = max(ans, i - m[pf[i]])
else
m[pf[i]] = i;
return ans;
3rd
continue.


subarray
continuous part of an array
subset
order
subsequence
4 1 6
Subset
continuous sequence
1
2 3 4 5 6
4 6 = subs