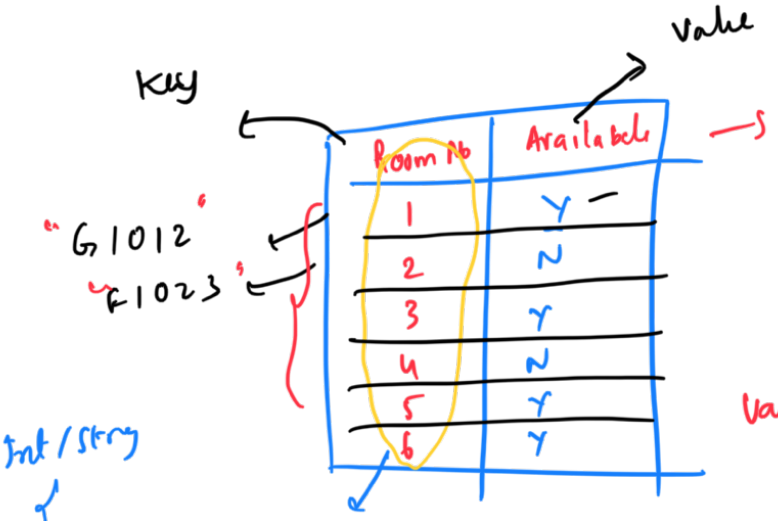
Hashing - 1

[3 5 1 6 4]



Hashmap
↓
< key, value > pair
↓
Have to be unique.

Key: Int / String
{

Value
{
AC / NON AC
Suite / Deluxe

key : Integer
Strings

Convert String → Integer (Polynomial Rolling Hash Function)

Hashmap : < key, Value >
Hashset : < key >

A = 3 7 9 6 5 3 → O(n)

hash_set = { 3, 7, 9, 6, 5, 3 }

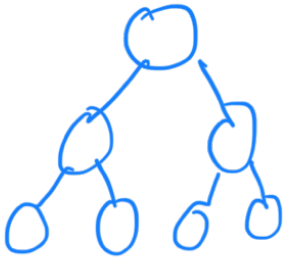
Types of Hashmaps / hashsets

Hashing Library	C++	Java	Python	T.C
1) Unsorted Map (Hash Table)	unordered_map	Hashmap	dict	O(1) T.C
2) Unsorted Set (Hash Table)	unordered_set	Hashset	set	O(1)
3) (Sorted Map)	map	Treemap	dict	O(log n)

Balanced B.S.T

Red-Black trees / AVL Trees

4) sorted set → set → Treeset → sorted list. O(log n)



Can we have a pair or a vector as a key in a map?

1) Unordered_maps ⇒ Not Possible

(Int, String)

Its implemented using hash table,

key → Integer

<2, 3> → {2, 3, 4, 5, 1, 2} → Integer

pair → Integer
vector → Integer

→ unordered_map<vector<int>, int> → ERROR

If we pass a hash function which converts pair/vector to integer ⇒ then it works

Ordered Maps!

map<vector<int>, int> hashmap;

Question: Shaggy is distance

Given a pair of integers, find a pair (i, j) such that a[i] == a[j] and |i − j| is minimum.

	0	1	2	3	4	5	6	7	8
A =	1	2	3	2	1	6	5	2	1

(1, 3)

A[i] == A[5] ? YES

d.H = 3 − 1 = 2

(1, 3)

Brute Force:

Consider all pairs

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

S.C: $O(1)$

Sorting:

	0	1	2	3	4	5	6	7	8
A =	1	2	3	2	1	6	5	2	1
sort(A) =	1	1	1	2	2	2	3	3	6
	0	4	8	1	3	7	2	6	5

S.C: $O(n)$

ans: 4/2

vector<pairs>

1) vector<pair<int, int>>

A = [3 5 6 4 1 6]

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

V = [(3, 0), (5, 1), ...]

val ind

sort(V)

= [(1, 4) (3, 0) (4, 3) (5, 1) (6, 2) (6, 5)]

O(nlogn)

C++

```
struct {
    int val;
    int ind;
    int x, y, z;
} Node;
```

V = vector<Node>

vector<int>

```
sort(v, comparator);

bool comparator(Node a, Node b){
    return a.val < b.val
}
```

T.C: O(nlogn + n) = O(nlogn)

S.C: O(n)

Approach 3: Hashing

A = [1 2 3 2 1 6 5 2 1]

0 1 2 3 4 5 6 7 8

(0, 4)
(0, 8)
(4, 8)

3 pairs

(4, 7)

(0, 8) is useless ?

→ Consider only adjacent pairs of equal value

A = 1 2 3 2 1 6 3 2 1
0 1 2 3 4 5 6 7 8

ans = (4)

(3-1) = 2

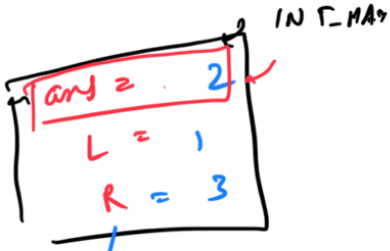
Latest occur(1) = 4

Hashing

y = y

4 - 0 = 4

6 - 2 = 4



Key	Value
A[i]	index of Latest occurence
1	4
2	3
3	2
6	5

T.C:

O(1)

→ Fetch the index of past occurence O(1)

→ updating ans O(1)

→ updating hashmap O(1)

T.C: n × O(1)

O(n)

S.C: O(n)

Que: Count no of pairs (i, j) such that a[i] == a[j] and i != j

(5C2)

A = [1, 1, 2, 2, 1, 2, 1] 3 3 3 3 3
0 1 2 3 4 5 6

= 5C2

$^4C_2 = \frac{4 \times 3}{2} = 6$

[1 1 1]

[2 2 2] 2 3 5 ⇒ $^3C_2$ = 3

6 + 3 = 9

if ans += (freq[i] * (freq[i] - 1)) / 2.

Freq. Hash : { '1' : 4, '2' : 3, '3' : 5 }

$^nC_2$ = n(n-1)/2

4(4-1)/2 + 3(3-1)/2 + 5(5-1)/2

TC : O(n) + O(n) = O(n)

SC : O(n)

2 Iterations

1) Traversing Array (To generate) hash map

2) Traversing

Approach 2 :

A = 1 2 3 2 1 6 3 2 1 2

0 1 2 3 4 5 6 7 8 9

ans += 3

A = 1 2 3 2 1 6 3 2 1 1

Hash Map

Key A[i]	Value Current A[i] Freq
1	4
2	3
3	2
6	1

ans = 3 + 2 + 1 + 0 + 0 + 0 = 10

```
ans += mp[a[i]];
mp[a[i]]++;
```

T.C : O(N)

S.C : O(N)

Question: Replicating a substring

B = [" a a b b "]

A = 2

Similar string ⇒ Equal strings

aa bb

ab ab

ab + ab = abab

B = a b b b b a b b a

A = 3

(Freq(b) % 3 == 0)

abb abb abb

S1 S2 S3

B = a b b b b a b b a a

A = 3

→ find Frequency of all characters
if # char (freq[char] % A == 0)
TRUE ✓

else
FALSE

T.C: O(n)
S.C: O(256) => O(1)

Max characters: 256 chars

Question: Sort based on a given dictionary.
→ Lexicographical

S = "aaabcade" ⇒ aaabcde

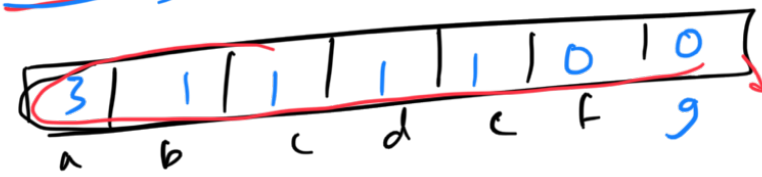
D = "cbfade"

Output: cbaaade

cbaaade

Approach 1:
Count sort →

T.C:
S.C:



Sum of Frequency = Length of String

aaabcde

```
for(i=0; i< s.length; i++){
    count[s[i]]++;
}
```

O(n)

freq_hashmap

}

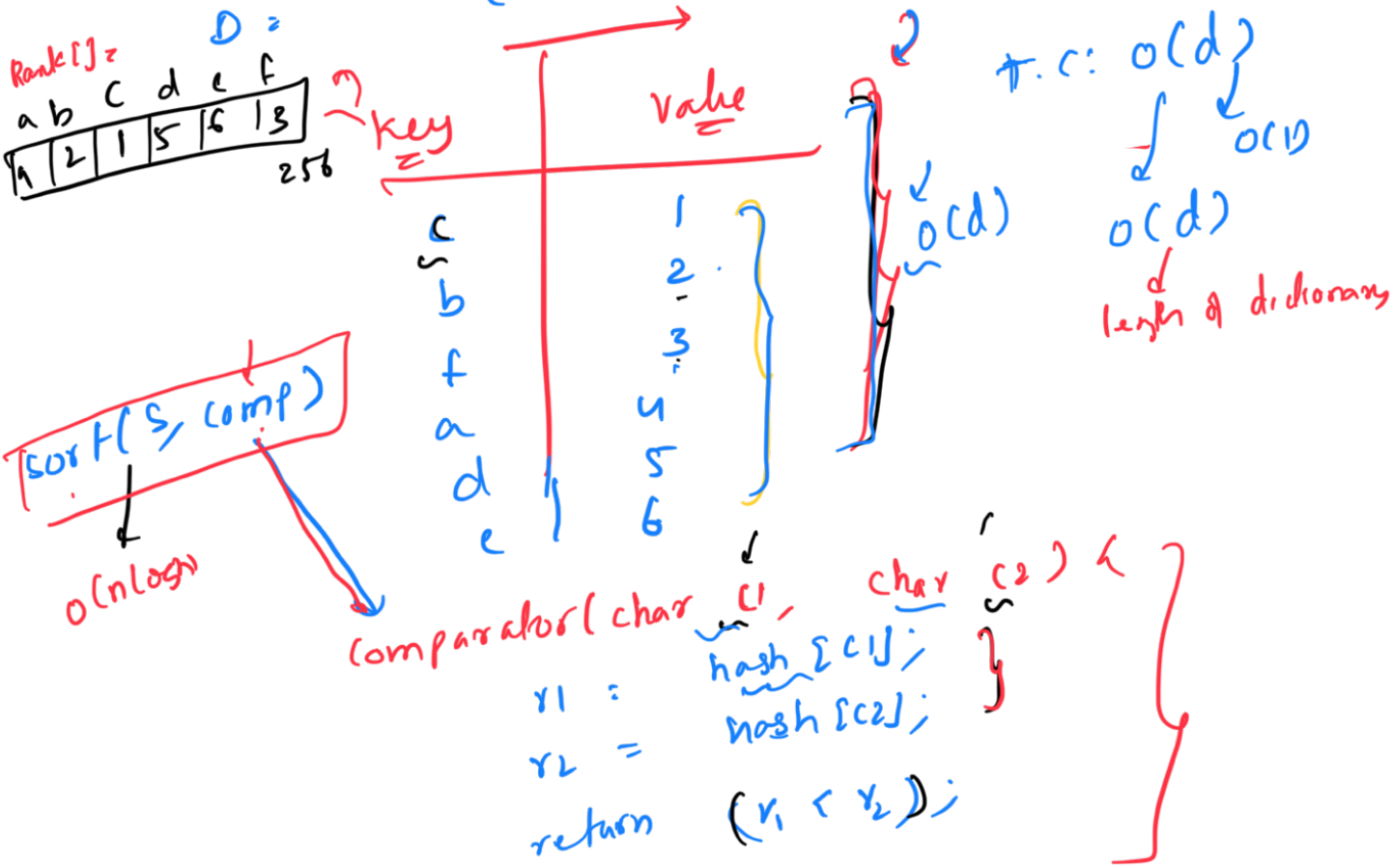
for (i=0; i< d.length; i++) {
    freq = count[d[i]];
    // Print freq or do sum
}

O(n)+O(n) = O(n)
S.C: O(1)

Approach2:

Comparator 1 2 3 4 5 6
"c b f a d e"

Max length of D = 256



}

T.C: O(nlogn)
S.C: O(d)

@ , ! _ , ,

Question: check if there is a subarray with sum = 0

2 Pointers / sliding window

Only works when array has positive elements

T.C: O(n)
S.C: O(1)

[-3, 1, 2] = 0

A:	1	-3	1	2	-4	6	2	-4	1	-5	7
	0	1	2	3	4	5	6	7	8	9	10

Brute force

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

T.C for sum of 1 subarray = O(n)

T.T.C: $O(n^3)$
S.C: O(1)

T.C: $O(n^2)$
S.C: O(n)

$O(n^2)$ T.C, O(1) S.C

pre[0] - pre[i-1]

O(1)

Prefix Sum:

	0	1	2	3	4	5	6	7	8	9	10
A =	1	-3	1	2	-4	6	2	-4	1	-5	7
Pre =	1	-2	-1	1	-3	3	5	1	2	-3	4

sum = 1

sum = 1

pre[7] - pre[3] = 0

Hashset = { 1, -2, -1, }

```
bool checkZeroSumSubarray ( int a[], int n ){
    // Generate pre[] from a[];
    hashset<int> st;
    for (i=0; i<n; i++) {
        if (pre[i] == 0 || pre[i] == k || pre[i] is in st)
            return true;
        else
            st.add(pre[i]);
    }
    return false;
}
```

(pre[i] - k)

(pre[i] == k

|| pre[i] == 0

A :

0	1	2	3	4
2	-1	3	-4	6

Pre :

0	1	2	3	4
2	1	4	0	6

sum[0 .... i] = 0

T.C : O(n)

S.C : O(n) → hashset

pre :

0	1	2	3	4	5
2	1	3	2	4	2	3

count = 1 + 1 + 1 (2)

Hashset : {2, 1, 3, 4}

Hashmap : <a[i], freq>

pre[i] = 0

Question : check if there is a subarray with sum = k (i, j)

pre[i] - pre[j] = k

pre[i] - k = pre[j]

pre [ ]

Question: Longest Subarray with Sum 0

	0	1	2	3	4	5	6	7	8	9
A =	4	3	2	-2	-3	5	-3	1	2	-3
pre =	4	7	9	7	4	9	6	7	9	6

F(4) = 0 L(4) = 4 4-0 = 4

F(7) = 1 L(7) = 7 7-1 = 6

Approach:

Sol:

2 Hashmaps

- First occurence of a num → Hash1
- Last occurence of a num → Hash2

firstOcc = { 4 : 0, 7 : 1, 9 : 2, 6 : 6 }

lastOcc = { 6 : 9, 9 : 8, 7 : 7, 4 : 4 }

ans = 6

F(4) = 0, L(4) = 4

F(7) = 1, L(7) = 7

F(9) = 2, L(9) = 8

F(6) = 6, L(6) = 9

6 − 3

T.C:

Step 1) Generate Prefix sum : O(n)

i) Generate First Occurence Hashmap : O(n)

2) " Last " " : O(n)

3) Iterate any hashmap : O(n)

T.C: O(n)

S.C: O(2n) ≈ O(n)

hashmaps

2 Hashmap

Soln 2:

Key : A[i]
value : < 1st Occur, 2nd Occur >

	0	1	2	3	4	5	6	7	8	9
A :	4	3	2	-2	-3	5	-3	1	2	-3
Pre :	4	7	9	7	4	9	6	7	9	6

(4) (4-0) = 4

first occurrence of a number

1 Hashmap : Store

Hashmap : { 4 : 0, 7 : 1, 9 : 2, -6 : 6 }

3 - 1 = 2

maxLen = 2 4 6

5 - 2 = 3

7 - 1 = 6

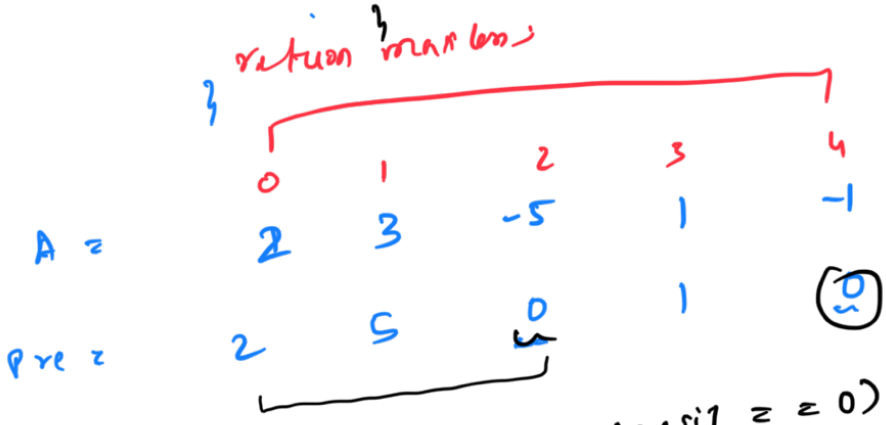
T.C : O(n)
S.C : O(n) 1 Hashmap

```
longestSubarray ( int arr[] int N) {
// Generate prefix Sum Array
hashmap <int,int> map;
maxLen = 0;
map.add{(0,-1};
for(i=0; i< n; i++) {
if(pre[i] == 0) maxLen = max(i+1,maxLen)
if ( pre[i] is not in map) {
    map[pre[i]] = i
}
else {
    len = i - map[pre[i]];
```

maxLen = max(len, ...

}

return maxLen;



if (pre[i] == 0) {
    len = i+1;
}

{ 0, -1 }

A[i] → index of 1st occurrence

if (pre[i] == 0) {
    i+1
    i - (-1)
}

if (pre[i] == 0) {
    maxLen = max(maxLen, i - map[pre[i]])
    i - (-1)
    map(maxLen, i+1);
}

Question: Find longest consecutive number subset — Kadane's Algo

A = 4 11 2 -1 3 5 12 -2 6 10 15 0

{ 10, 11, 12, 13 } : 4

{ 2, 3, 4, 5, 6 } : 5

Ans = 5

{ -2, -1, 0 y : 3

A2 = 7 4 2 6 3 5 8

{ 2, 3, 4, 5, 6, 7, 8 } = Ans = 7

Approach 1:

Sort the Array and check adjacent elements.

sort(A) = -2 -1 0 2 3 4 5 6 10 11 12 13

O(nlogn + n)

T.C. = O(nlogn)

S.C: O(1)

Approach 2:

A = 4 11 2 -1 3 5 12 -2 6 10 13 0

MAP

Hashset

'4': {4, 5, 6}

11: {11, 12, 13}

'2': {2, 3, 4, 5, 6}

=> 3

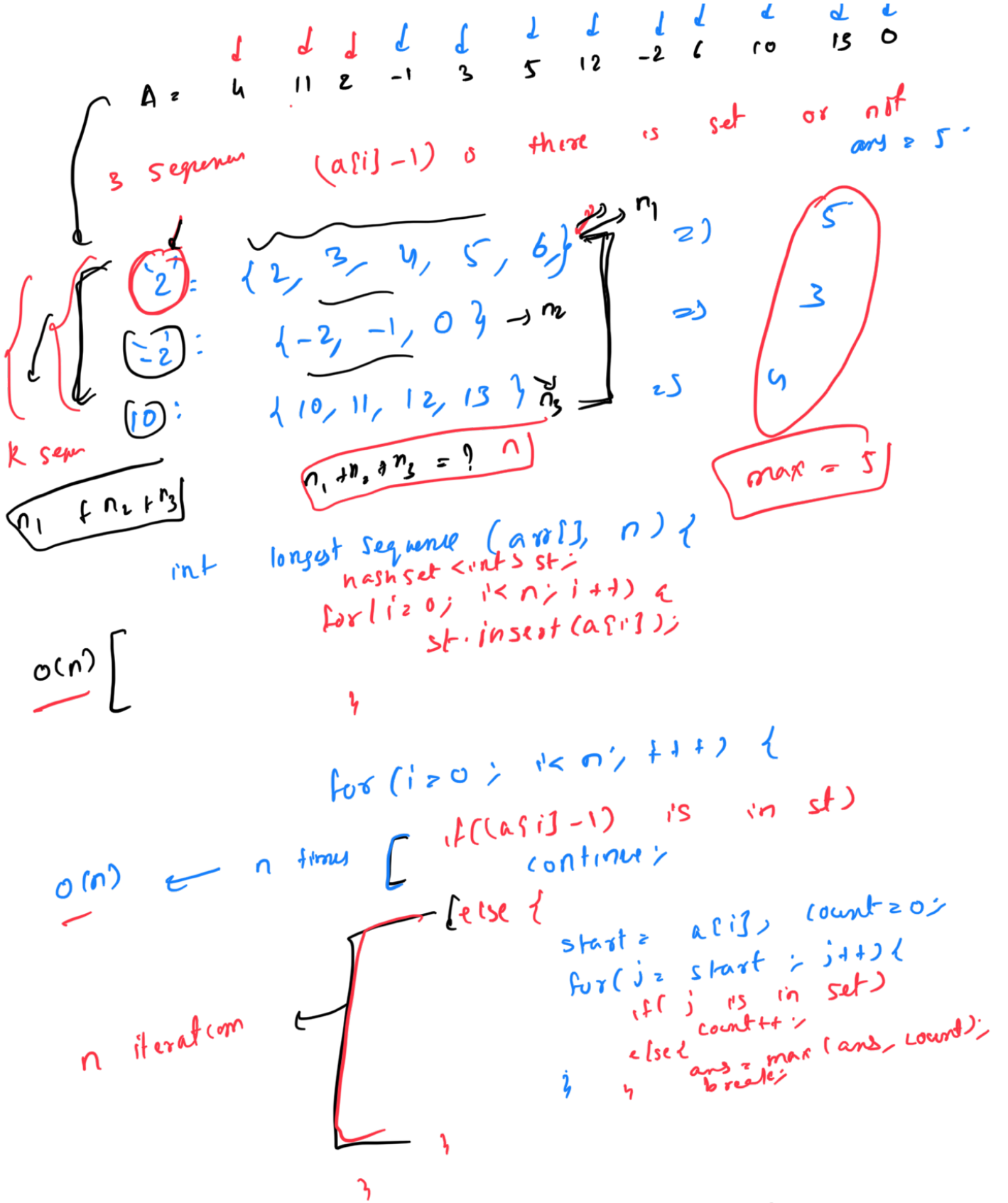
=> 3

=> 5

(a[i]-1)

T.C: O(n^2)

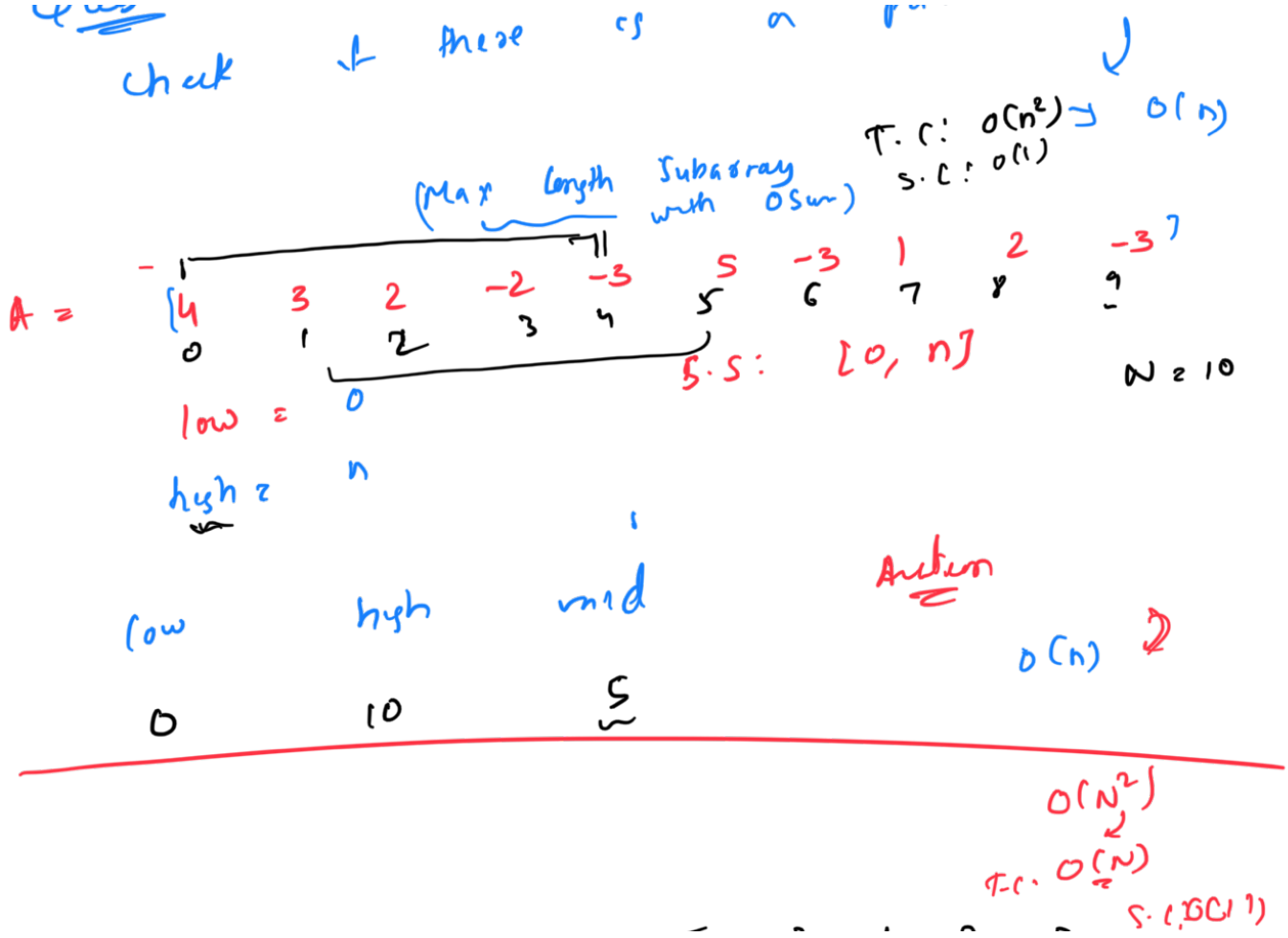
S.C: O(n)



T.C: $O(n + n) = O(n)$

S.C: $O(n)$

Problem: pair with sum = k



Question: Find the closest pair (i, j)such that a[i] == a[j]

```
minDist = INT_MAX;
for(int i = 0; i < n; i ++){
      if(a[i] not in hashmap)
            hashmap[a[i]] = i;
      else{
            if(i - hashmap[a[i]] < minDist){
                  minDist = i - hashmap[a[i]];
                  L = hashmap[a[i]];
                  R = i;
                  hashmap[a[i]] = i;

            }
      }
}
```

Question: Longest subset with consecutive elements

```
unordered_set<int> st;
Insert elements in hashset;
int ans = 0;
for(int i= 0 ; i<n; i++){
     if((a[i] - 1) is there in set)
          continue:
     else{
          int c = 1;
          for(j = a[i] + 1; ;j++){
               if((j) is there in set) c++
               else{
                    ans = max(ans, c);
                    break;
               }
          }
     }
}
return ans;
```

Question: Check if there is a subarray with 0 sum

```
bool checkZeroSubarray(int pre[], int n){
     unordered_set<int> st;
     for(int i=0;i<n;i++){
          if(pre[i] == 0)
               return true;
          if((pre[i]) is there in set)
               return true;
          else
               st.insert(pre[i]);
     }
return false;
}
```

Question: longest subarray with 0 sum

```
unordered_map<int, int> mp;
mp.insert(0, -1);
for(int i=0;i<n;i++){
     if((pre[i] is there in set){
          len = i - mp[pre[i]];
          ans = max(ans, len);
     }
     else
          mp[pre[i]] = i;
}
```