class starts at :- 7:37 a.m

Agenda: [Hashing]

- Pairs with given xor
- subarray with given sum
- valid sudoko
- Is dictionary ← [h(w)]
- subarray sum equals K [Time permits]

Pairs with given XOR

Given: Given an array of N distinct integers

To find: No of pairs with given XOR

Example: A = [3, 6, 8, 10, 15, 50]

B = 5

3 ^ 6

10 ^ 15

ans: 2

3 ^ 6 = 0 1 1 / 1 1 0 / 1 0 1 → 5.

XOR: 1 → diff values

0 → same values

Brute force:

- Go through all the pairs — for(i=0; i<n; i++){ for(j=i; j<n; j++){
  - find xor of that pair }
    - if xor == B
      - increase my ans

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

optimised approach

$A[i] \wedge A[j] = B$

observations:

2 indexes : i

j

(n^2)

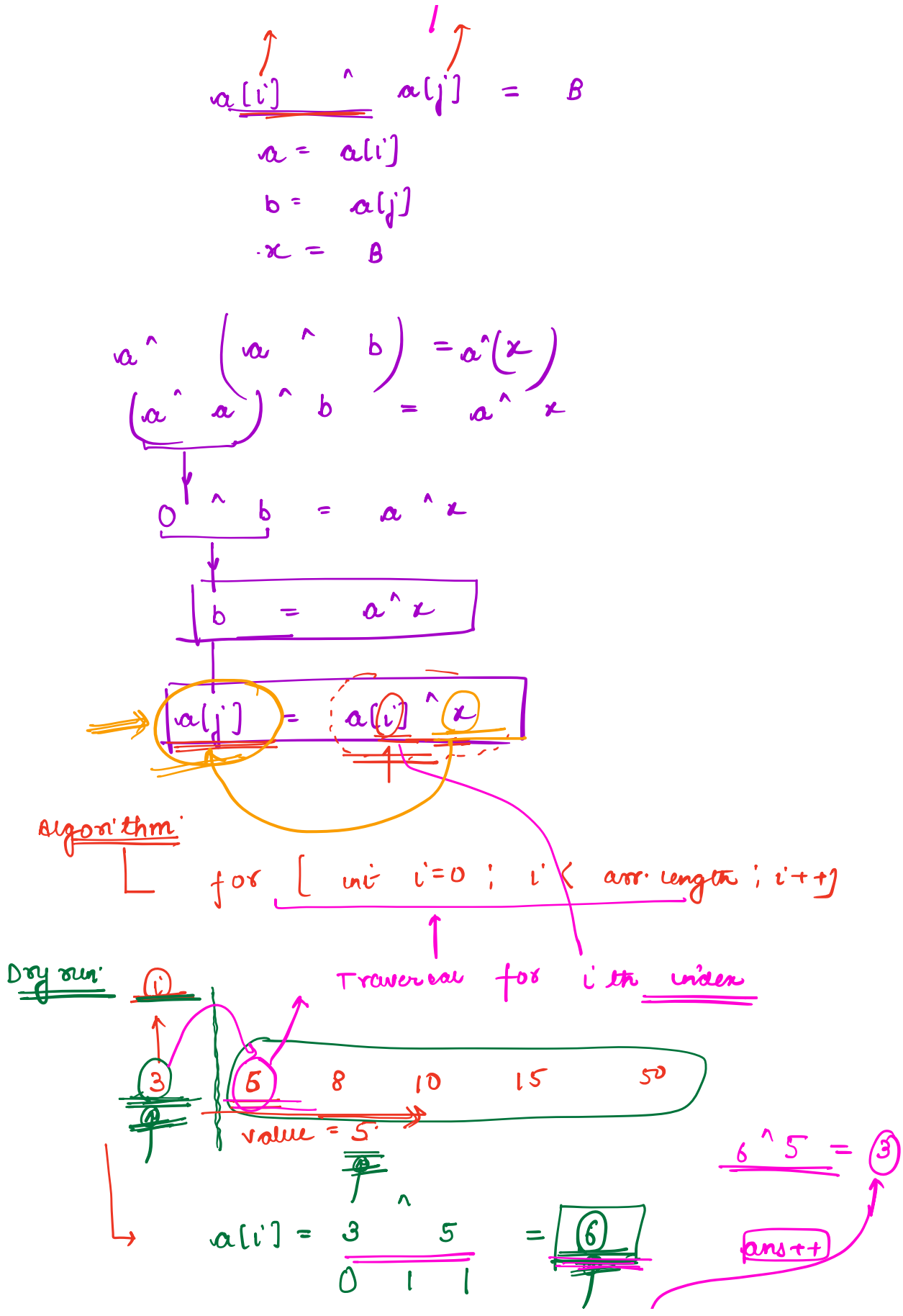
1 index — $O(n)$

mathematical concept

$a[i] \wedge a[j] = B$

if and only if

$a[i] \wedge B = a[j]$


a[i] ^ a[j] = B
a = a[i]
b = a[j]
x = B
a ^ (a ^ b) = a^(x)
(a ^ a) ^ b = a ^ x
0 ^ b = a ^ x
b = a ^ x
a[j] = a[i] ^ x
Algorithm
for [int i=0 ; i < arr. length ; i++]
Traversal for ith index
Dry run
i
3
5
8
10
15
50
value = 5
a[i] = 3 ^ 5 = 6
0 1 1
6 ^ 5 = 3
ans++

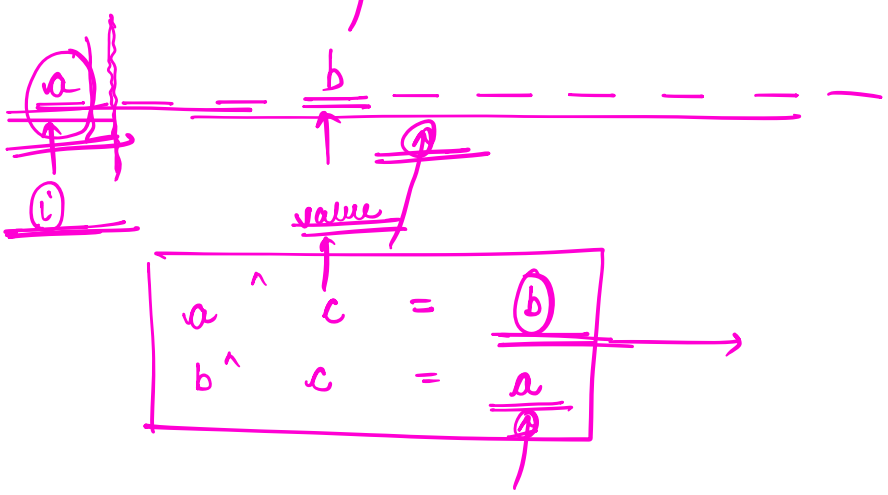
1 0 1
1 1 0

hash set: [4, 3]

hashmap
(or)
hashset

count ?

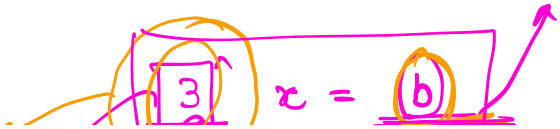
Distinct integers = hashset
Duplicates :- hash map
↳ frequency

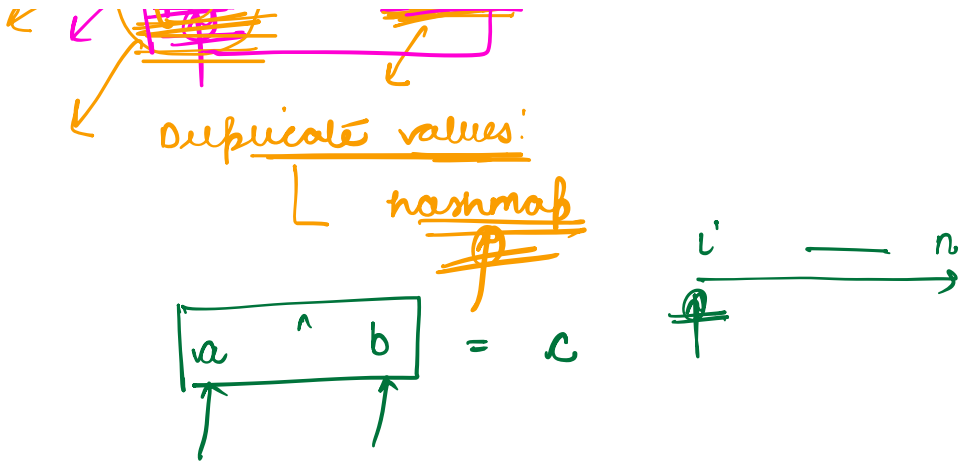


If insert every entry in map
↳ duplicate pairs

[3, 6, 8, 10, 15, 30]

hashset: [3, 6, 8, 10, 15, 30]





Try to store that element which you are 100 % sure, that is part of array

Pseudo code:

```
solve ( int[] A , int x) {
    Set < Integer > set = new HashSet<>();
    for (i = 0 ; i < n; i++) {
        if ( set.contains(A[i] ^ x))
            count ++;
        set.put(A[i]);

    return count;
}
```

Subarray with given sum

Given: A[] → positive integers

B

TO find: find and return the first continuous subarray with sum B.

Example: A = [1 , 2 , 3 , 4 , 5]

B = 5

subarray 1 = [2 , 3] → 5

subarray 2: [5] = 5

Brute force:

- Go through all subarrays
  - find sum
    - if sum == B
      - return the subarray

T.C:- $O(n^3)$

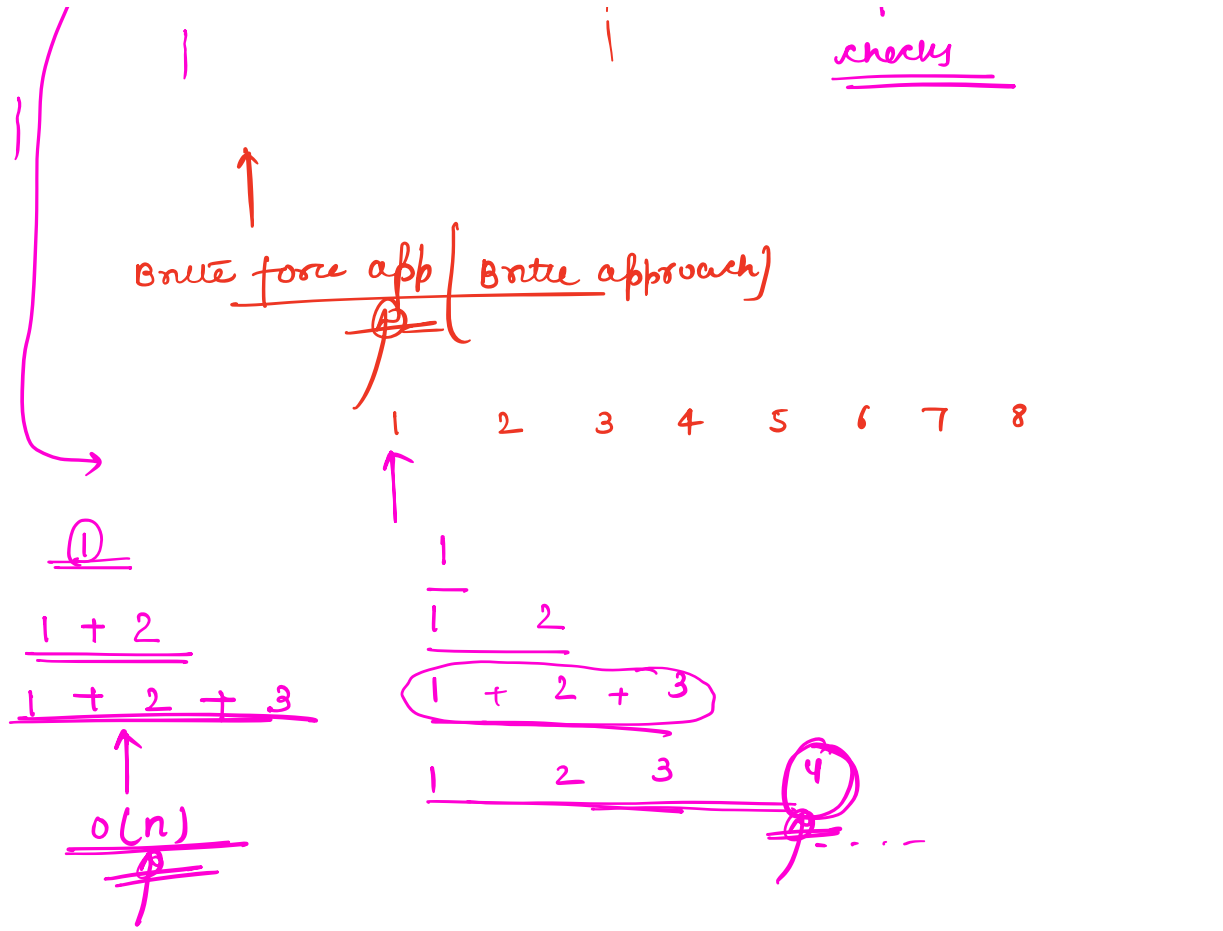
$O(n^2)$

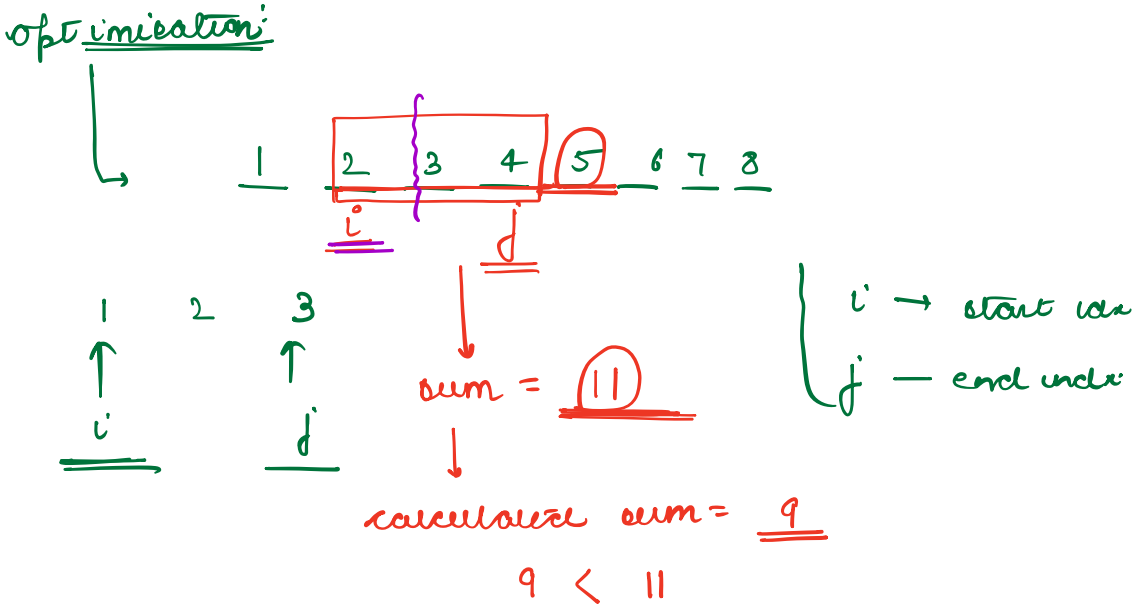
n^3

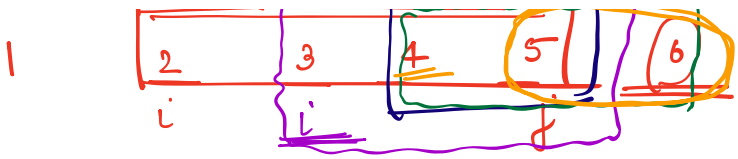
```
for(i=0; i<n; i++)
    for(int j=i; j<n; j++)
        int sum=0;
        for(int k=i; k<j; k++)
            sum += arr[k];
```

n^2

```
for(i=0; i<n; i++){
    int sum=0;
    for(int j=i; j<n; j++)
        sum = sum + A[j]
}
```


checks
Brute force app (Brute approach)
1 2 3 4 5 6 7 8
1
1 + 2
1 + 2 + 3
O(n)
1
1 2
1 + 2 + 3
1 2 3
4


optimisation
1 2 3 4 5 6 7 8
i
j
1 2 3
i
j
sum = 11
i → start index
j — end index
calculate sum = 9
9 < 11



sum = 11

calculated sum = 14

14 > 11 [j++ ✗

cal-sum [3, 4, 5] = 12

12 > 11

calculated_sum [4, 5] = 9.

9 < 11

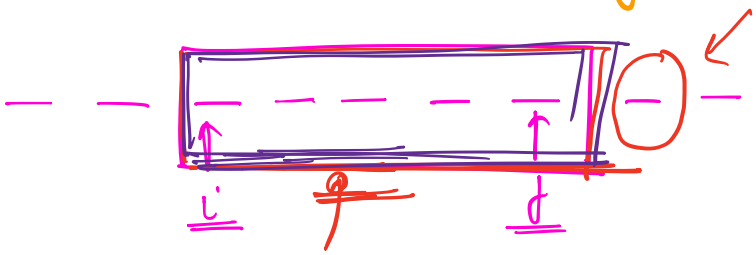
cal-sum [4, 5, 6] = 15

15 > 11

cal-sum [5, 6] = 11

11 = = 11

└ return my ans



sum[i, j] = = sum

(return my ans)

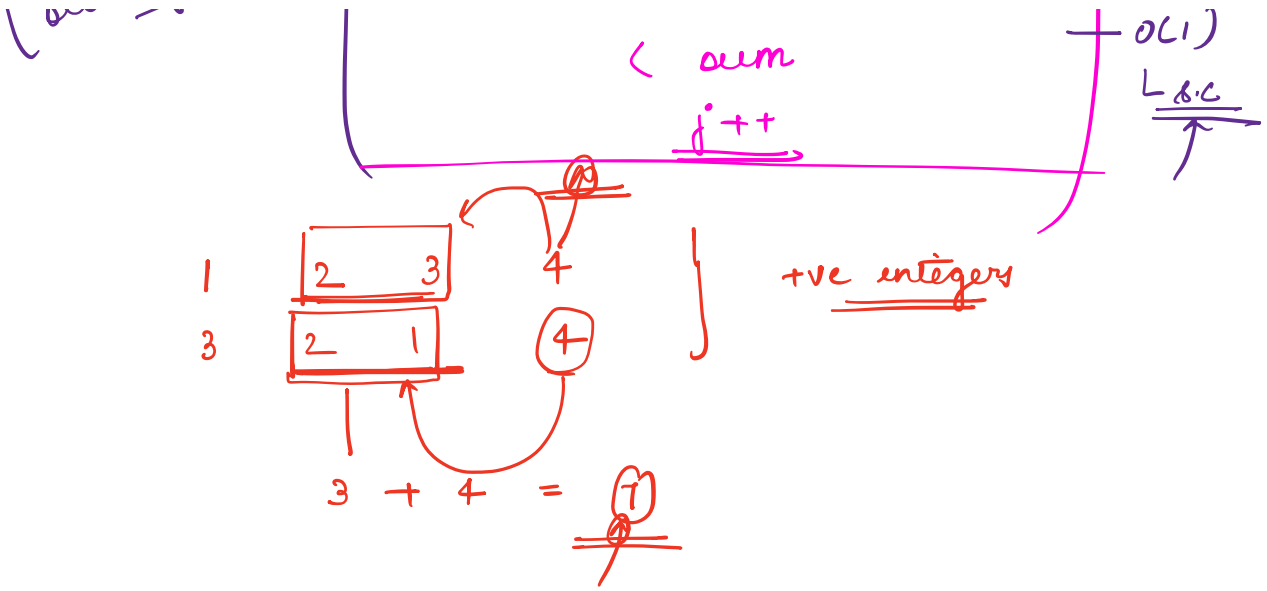
> sum

i++

Algo

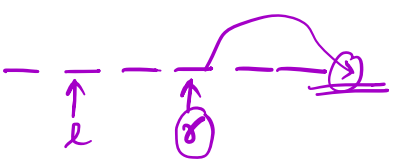
sliding window

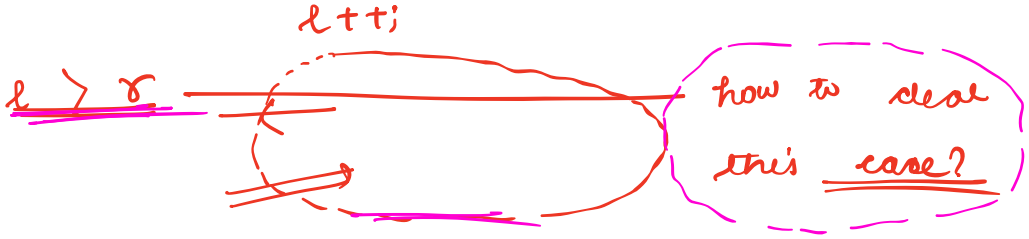
→ O(n)



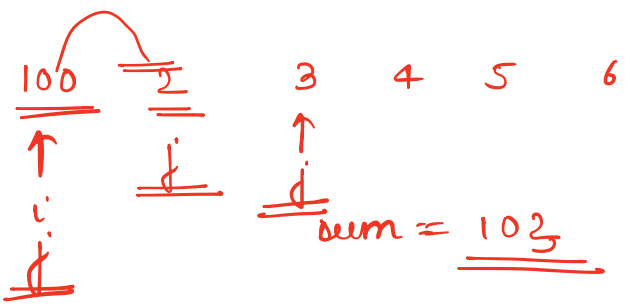
Pseudo code:

```
fun (int[] A,  int x )
    int l = 0;
    int r = 0;
    int sum = A[l];
    while ( r < n ) {
        if ( sum == x) {
            return the subarray from
            l to r;
        }
        else if ( sum > x)
            sum - = A[l]
            l++;
```



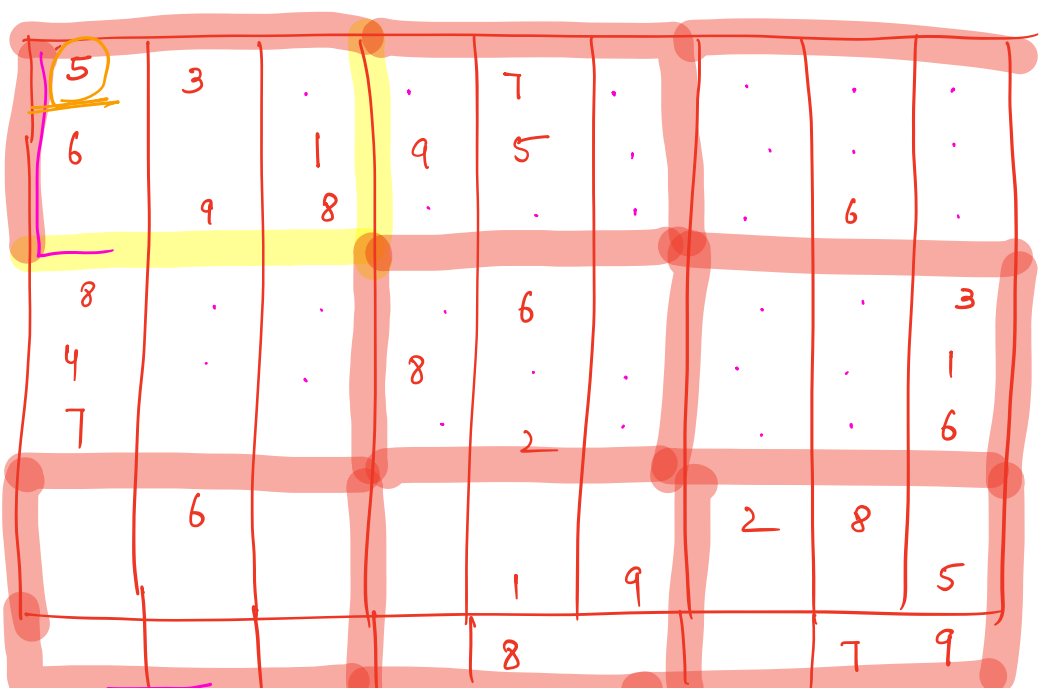


```
else if (sum < x)
    sum += A[r]
    r++;
}
}
```



Break till: 9:00

Valid sudoku



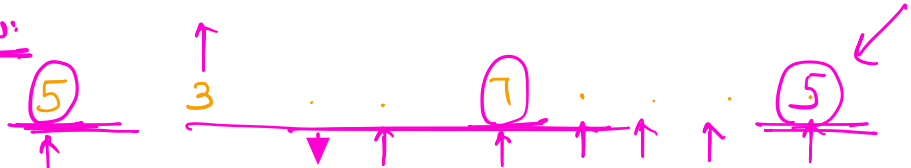
Rules:

- Every row should have distinct values
  (invalid) 1 2 3 1 4 5 8 9 7
- Every col should have distinct values
- Every box should have a distinct value

Please note, we need to check whether the number is valid for existing value only

DS :- Set [Hashset]

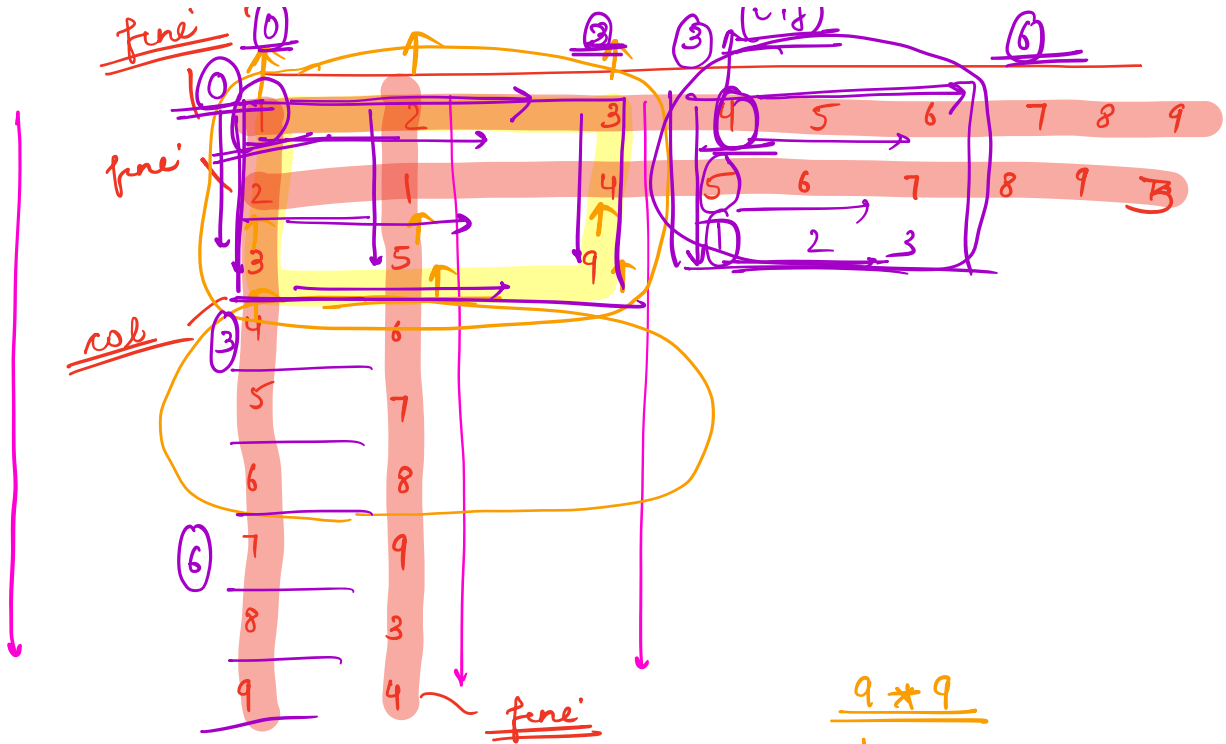
Row:



Set: 5, 3, 7,

This set stores the already visited values

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



9 * 9

└ 3 * 3

Algo: for every element -

check whether it exists in my row, col and box

TC: $O(n^2)$

Pseudo code: mat[][] HashSet [set]

```
// check for row
for ( i = 0; i < n; i++) {
    set.clear();
    for( j = 0; j < 9; j++) {
        if ( mat[i][j] == '.') {
            continue;
        }
    }
```

```
if ( set.contains (mat[i][j]) )
    return false;
set.add (mat[i][j]);
}
```

// check for cols [Assignment]

// check for box

mat 16 * 16 [Box = 4*4]

```
for ( int i = x ; i < x + n ; i++ ) {
    for ( int j = y ; j < y + n ; j++ ) {
        if ( c == '.' ) {
```

(x: 0 → 3; y: 0 → 3; c = mat[i][j])

for one box

```
            continue;
        }
        if ( set.contains (c) )
            return false;
        set.add ( mat[i][j] )
    ;
```

9 * 9 → n = 9

0 3 6

y y+3 y+6

(y+3)+3

0 3 6

x+3

x+6

(x+3)+3

k=0 → k=0 x = 0/3 = 0

y = 0%3 = 0

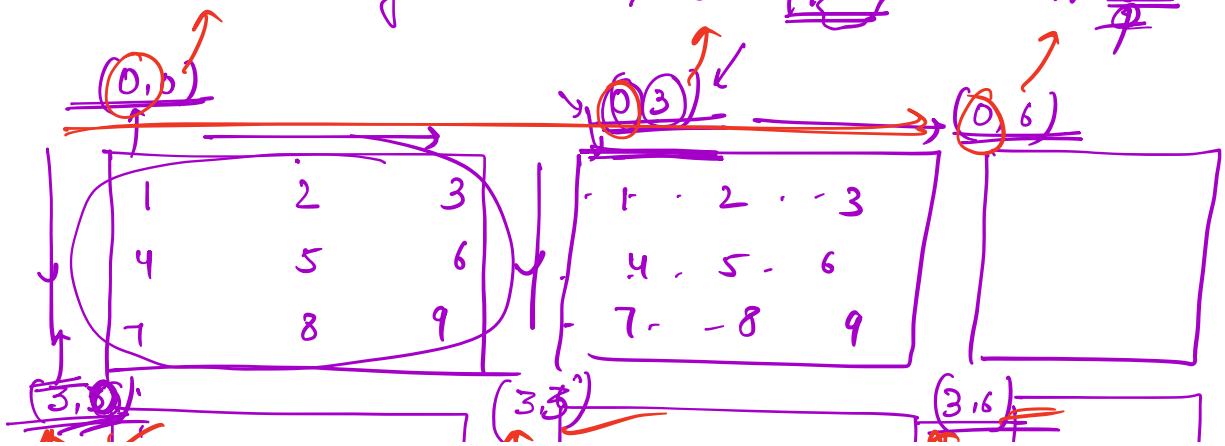
i = x i < x + 3

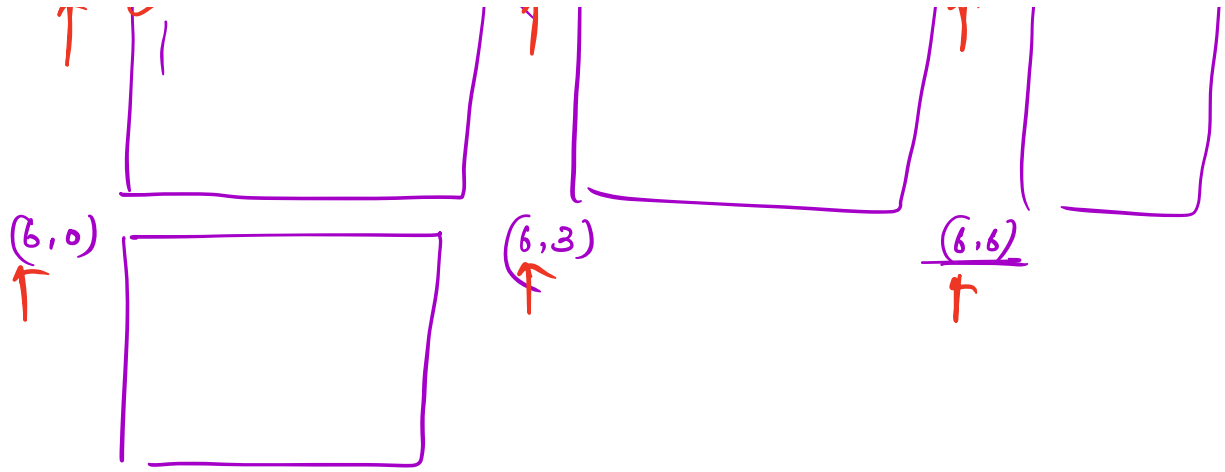
j = y; j < y + 3

9 * 9

└ no of boxes = 9

```
for( int box = 0; box<9; box++){
    x = box/n;   // 0   // 0
    y = box%n;   // 0   // 1 2
```





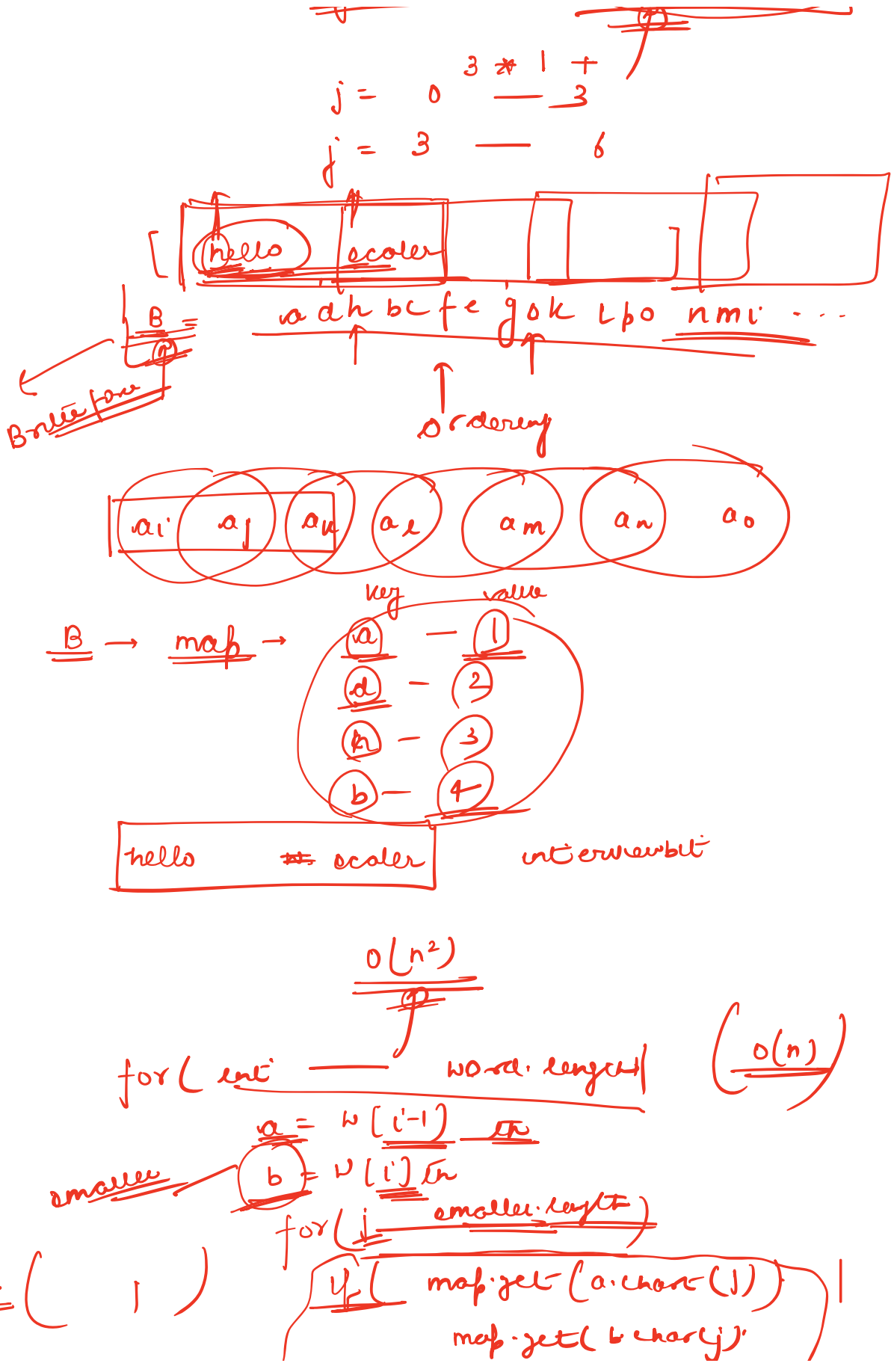
Request: Go through the box implementation

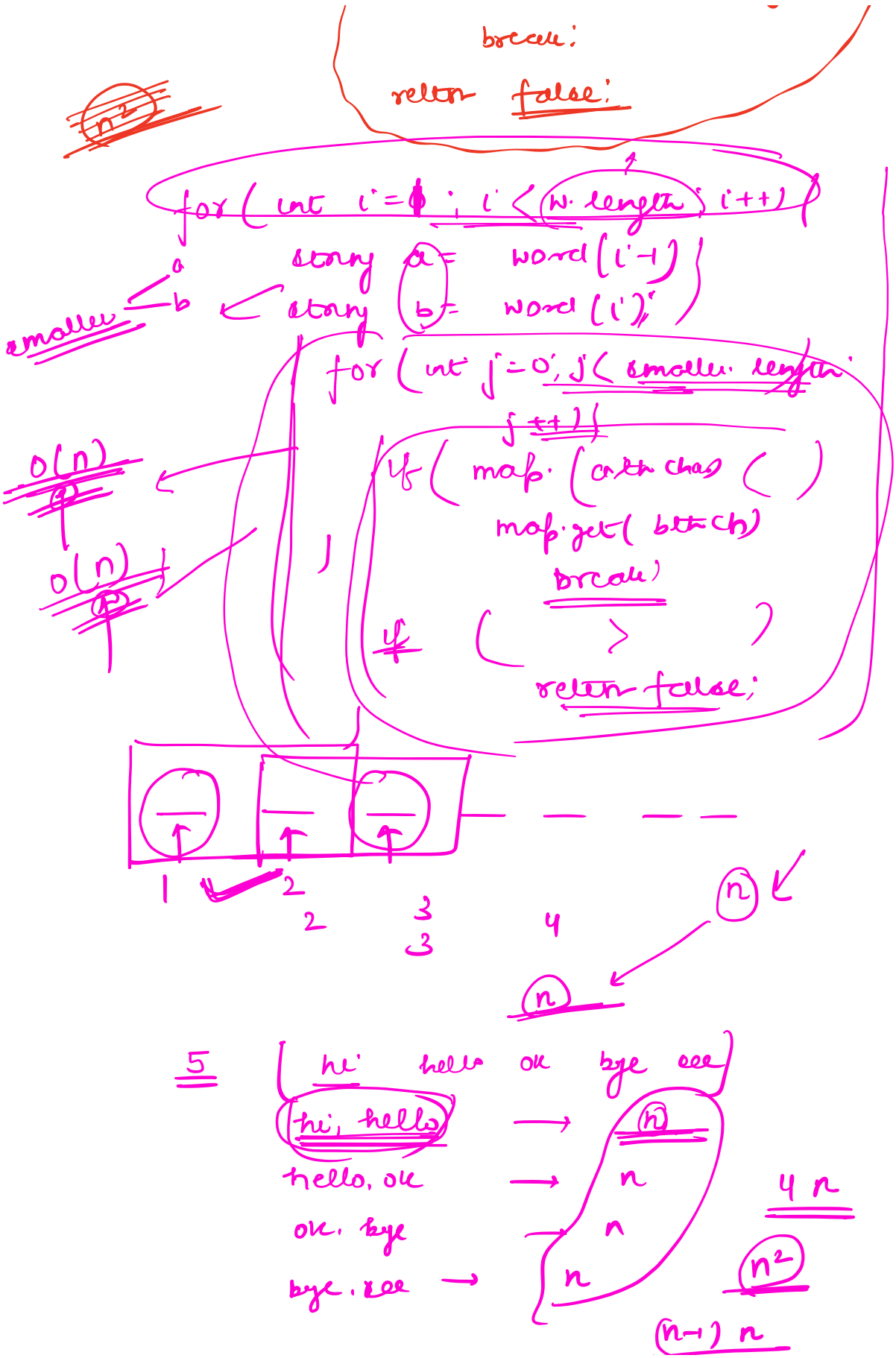
```
for( int box = 0; box( n*n; box++)
    int x = box( % n ;  // (0)
    int y = box % n;    3*(x)+(3) //(1)

    for( int i = x*n i < n*x + n ; i++)
    for(int j = y*n j ( n*y +n ; j++)
    {
                0*(6)
                j< 3*1+3
        // all my checks
    }
)
```

box=0 ; x=0

y=0


3 * 1 +
j = 0 — 3
j = 3 — 6
hello
ecoler
a d h b c f e g o k l p o n m i . . .
B
Broke for
ordering
a1 a3 a4 a2 am an a0
key
value
B → map →
a — 1
d — 2
h — 3
b — 4
hello ecoler
interview bit
O(n²)
for (int — word.length
O(n)
a = w[i-1] th
smaller
b = w[i] th
for (j — smaller.length
N. (1)
if (map.get(a.charAt(j))
map.get(b.charAt(j)


break;
return false;
n^2
for (int i = 0 ; i < w.length ; i++)
string a = word(i-1)
string b = word(i)
smaller
a
b
for (int j = 0 ; j < smaller.length ; j++)
O(n)
O(n)
if (map.(a.charAt(j)) map.get(b.charAt(j))
break;
if (>)
return false;
1
2
2
3
3
4
n
n
5
hi hello ok bye eee
hi, hello
hello, ok
ok, bye
bye, eee
n
n
n
n
4 n
n^2
(n-1) n