Hashing 2 - problems

- subarray sum eq. k
- longest consecutive sequence
- Shaggy & distances

P1 Given an array of integers A and a number k find total number of subarrays having sum = k

ex $\{\overset{0}{10}, \overset{1}{2}, \overset{2}{-2}, \overset{3}{-20}, \overset{4}{10}\}$ k=-10

ans 3

ex $\{\overset{0}{9}, \overset{1}{4}, \overset{2}{20}, \overset{3}{3}, \overset{4}{10}, \overset{5}{5}\}$ k=33 ans=2

idea1 (brute force)

```
for i=0 → n-1                [i, j]
   for j=i → n-1
      sum=0
      for k=i → j   sum+=a[k]
```

TC: $O(n^3)$

SC: O(n) ← PS ——→ O(n)

idea2 PS

```
for i=0 → n-1
   for j=i → n-1
      [i,j] → only this query can be answered with O(1)
```

TC: $O(n+n^2) = O(n^2)$

idea3 CF

```
for i=0 → n-1
   sumSofar=0
   for j=i → n-1
      sumSofar+=a[j]
      ... check[i,j] → k
```

TC: $O(n^2)$

SC: O(1)

idea3 3 min

Calculate PS. → TC: O(n)   k

inject PS into a hashSet

meanwhile if PS element ($PS_i - k$) exists then ans++

$$PS_i - k = +PS_j \rightarrow PS_i = PS_j + k$$

HS

0 9
13 33
46

{9, 4, 20, 3, 10, 5} k=33 (indices 0 1 2 3 4 5)

PS→{9, 13, 33, 36, 46, 51}

i

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

$9-\overset{k}{33}=-24$
$13-33=-20$
$33-33=0$

$46-33=13$

Pro code:

- Calculate P.S as we go
- Count up in hash map

avoid common bug in interviews → for true/false HS is enough

total Count → note that for Count HM is needed

≠0

HM

(0,1)

SC: O(n)
TC: O(n)

```
int countSumk(int a[], int k){
    n = a.len
    hm = new hashMap();
    hm.add(0,1) // why? → explained
    ans = 0 ;   curSum = 0
    for(i=0; i<n; i++){
        curSum += a[i]
        if(hm.containskey(CurSum-k)){
            ans += hm[CurSum-k]
        }
        hm.add(curSum, hm[curSum]+1)
    }
    ret ans;
```

Care. overflow?!

default → if not exist ret 0

HM

(0,2) (9,1)
(13,2)

{9, 4, 20, 3, 10, 5, -5} k=33 (indices 0 1 2 3 4 5)

ans=1

curSum= ~~9~~, ~~13~~, 33, 36, 46, 51, 46

i

$46-33=13$

good example when 13 count 2

ex {10, 2, -2, -20, 10} k=-10 (indices 0 1 2 3 4)

PS {10 12 10 -10 0}

Alternative problem
rectangle Counting

distict elements

P2 Given an integer array a, find the length of longest chain of consecutive elements.

wait for the example

ans= 4

ex $a : \{\overset{0}{100}, \overset{1}{4}, \overset{2}{3}, \overset{3}{6}, \overset{4}{10}, \overset{5}{20}, \overset{6}{11}, \overset{7}{5}, \overset{8}{101}\}$

{100, 101}
{3,4,5,6}
{10,11}
{20}

idea1

sort O(n log n)

{3,4,5,6,10,11,20, 100,101}

idea2

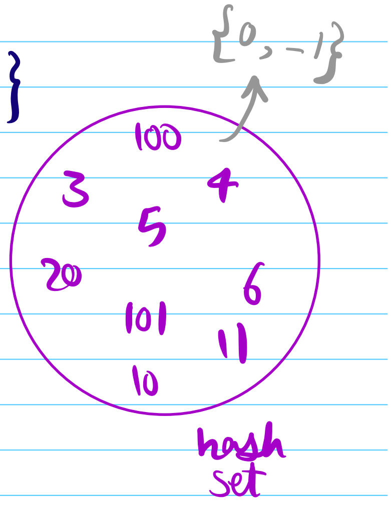
O(n)
hash set Filled

$a : \{\overset{0}{100}, \overset{1}{4}, \overset{2}{3}, \overset{3}{6}, \overset{4}{10}, \overset{5}{20}, \overset{6}{11}, \overset{7}{5}, \overset{8}{101}\}$

i

left right

3 4 5 6



```
- inset all elements in hashset : O(n)
- for i = 0 → n-1 {
      if(a[i]-1 is not Present in set) continue //skip
      else // first number in consec. series
          | while(a[i]+next is in set) next+=1
          | maxLen
```

TC : O(n)
SC : O(n)

Already dicussed last session!

P3 Shaggy & distances!

Given an integer array a, find min $|\overset{\text{index}}{i - j}|$ st.

$i != j$, $a[i] = a[j]$ & $|i-j|$ is minimum.

Absolute

wait for this!

$a: \{\overset{0}{2}, \overset{1}{3}, \overset{2}{5}, \underbrace{\overset{3}{7}, \overset{4}{2}, \overset{5}{6}, \overset{6}{8}, \overset{7}{7}}_{4}, \overset{8}{3}, \overset{9}{5}, \overset{10}{2}, \overset{11}{3}\}$

Brute Force:

```
For i = 0 -> n-1
    j = i+1 -> n-1
        a[i] = a[j] ...
```

hash:

key	value
a[i]	

```
int getMinDistance(int a[]){
    n=a.len
```

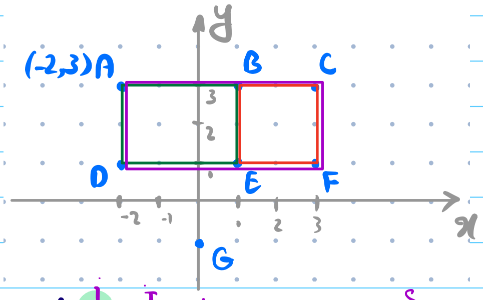
TC:

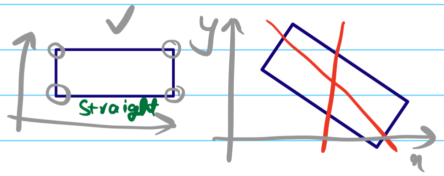
SC:

Optionals

Amir microsoft screening

P1 Given n points in a 2D plane, count the number of rectangles with sides prallel to xy axis. All four corners must be part of input & two rectangles are diffirent if one point is different.

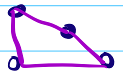


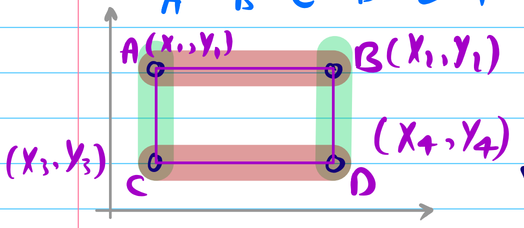


ans = 3

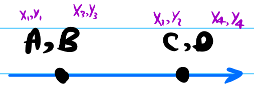
input

	i	j	k			s	
X =	-2	1	3	-2	1	3	0
Y =	3	3	3	1	1	1	-1
	0	1	2	3	4	5	6
	A	B	C	D	E	F	G



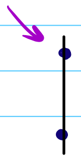


$$\begin{cases} x_1 = x_3 \;\&\& \\ x_2 = x_4 \;\&\& \\ y_1 = y_2 \;\&\& \\ y_3 = y_4 \end{cases}$$



$\&\& \; y_1 \neq y_3$

$\&\& \; x_1 \neq x_2$

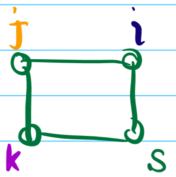


idea1: ∀ four point in input test
    Count++;

→ ret Count ?

TC: $O(N^4)$

SC: $O(1)$



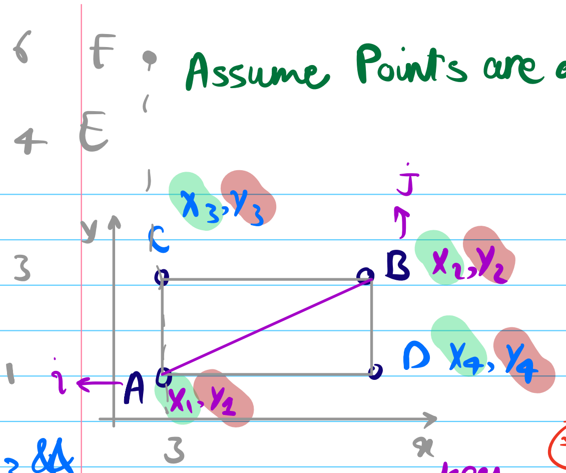
```
for i = 0 → n-1
    j = 0 → n-1
        k = 0 → n-1
            s = 0 → n-1
                if( check( x[i], x[j], x[k], x[s]
                           y[i]  y[j]  y[k]  y[s] ) )
                    Count ++
```

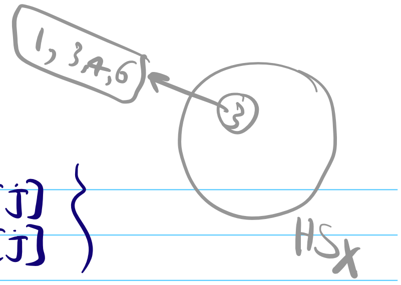
ret Count / 4 // because each □ will be counted 4 times

Q1

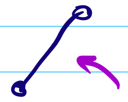
idea2

Assume Points are distinct





$\left\{ \begin{matrix} x[i] , & x[j] \\ y[i] , & y[j] \end{matrix} \right\}$



Condition of being diag. of a □

⊛ $x[i] \neq x[j]$ && $y[i] \neq y[j]$

< <

$\left\{ \begin{matrix} x_1 = x_3 \text{ \&\&} \\ y_2 = y_3 \end{matrix} \right.$

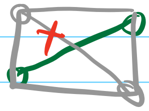
$x_2 = x_4$

$y_1 = y_4$

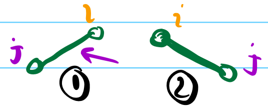
hashSet< $\overbrace{\text{Pair}\langle \overset{\text{int}}{X}, \overset{\text{int}}{Y} \rangle}^{\text{key}}$ > HS

// whole Pair<X,Y> is object for key.

inset all points to hash Set → O(n)



// form



$C(\underset{\sim}{X_1}, \underset{\sim}{Y_2})$

$D(\underset{\sim}{X_2}, \underset{\sim}{Y_1})$

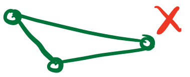
```
i = 0 ⟶ n-1
  j = i+1 ⟶ n-1
    if( check diag( x[i] , x[j] ) ){
                    y[i] ; y[j]
       HS.Contain( new Pair (X[i], Y[j]) ) &&
       HS.Contain( new Pair (X[j], Y[i]) )
           ans++
    }

ret ans/2 ;
```

A B

St all A,B,C,D form □

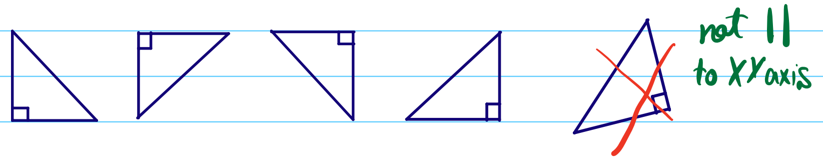
$O(n^2)$: TC

$O(n)$: SC

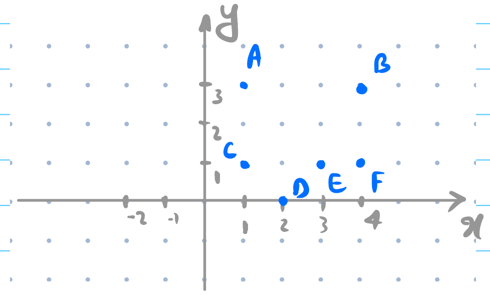


right angle

P2 Given n points in a 2D plane, count the number of triangles with sides prallel to xy axis. All three corners must be part of input & two triangles are diffirent if one point is different. points are ditinct

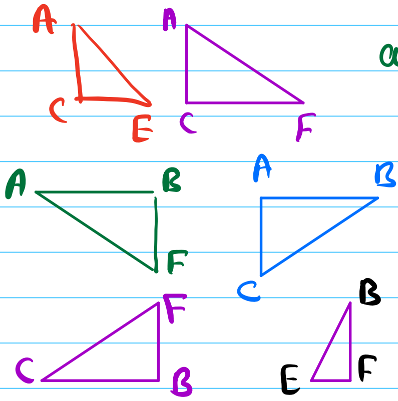


Q2

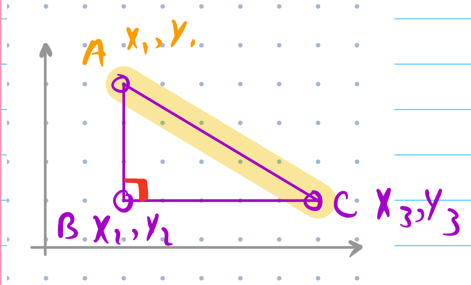


	A	B	C	D	E	F
X=	1	4	1	2	3	4
Y=	3	3	1	0	1	1

ans=6



$A(x_2, ?)$
$C(?, y_2)$
$\frac{x_2 \quad y_2}{B}$



$$\begin{cases} X_1 = X_2 \\ Y_2 = Y_3 \end{cases} \&\& \; X_1 != X_3$$

idea1 similar to prev solution with modification for ◺

TC: $O(n^2)$
SC: $O(n)$

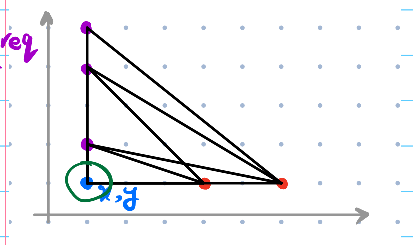
idea 2

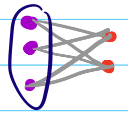
hash map for Freq x of all point

x HM ←

y HM ←

hash map for Freq y of all point





3x2 = 6 different ◺

x,y ←

```
for i = 0 → n-1
    x[i]
    y[i]
    { xHM[X[i]]++
    { yHM[Y[i]]++
```

```
xHM → Hash map for all X coordinations   O(n)
yHM → Hash map for all Y coordinations   O(n)
for i = 0 → n-1 // focus on each Corner
    ans += (xHM[X[i]]-1) * (yHM[Y[i]]-1)
ret ans;
```

exclude the current Point i