HASHMAP 2

Content for Today

01. Count no. of distinct points
02. Count no. of rectangles
03. Count no. of triangles
04. Count permutation of string A in string B

01. Given N 2D points, calculate no. of distinct points.

$\{x, y\}$

Ex:-

	0	1	2	3	4
$x[s] = \{$	2	3	2	5	3 $\}$
$y[s] = \{$	3	6	3	7	6 $\}$

// i^{th} point : $x[i]$. $y[i]$

$\boxed{ans = 3}$

Idea (a) → Insert all points in hashmap <key, val>

HM < x , y >

Eg:- (2, 4) (2, 6)

HM = { 2 ~~→~~ 4 , 2 → 6 }

Data might get overridden

(b) Store complete point as a key

HS <String> set = new HS<>();

(x, y) = "x" + "y"

(12, 3) = "12" + "3" = 123

(1, 23) = "1" + "23" = 123

→ you just need a separator

$(x,y) = x + "@" + y$:

$(12,3) = "12@3"$

$(1,23) = "1@23"$

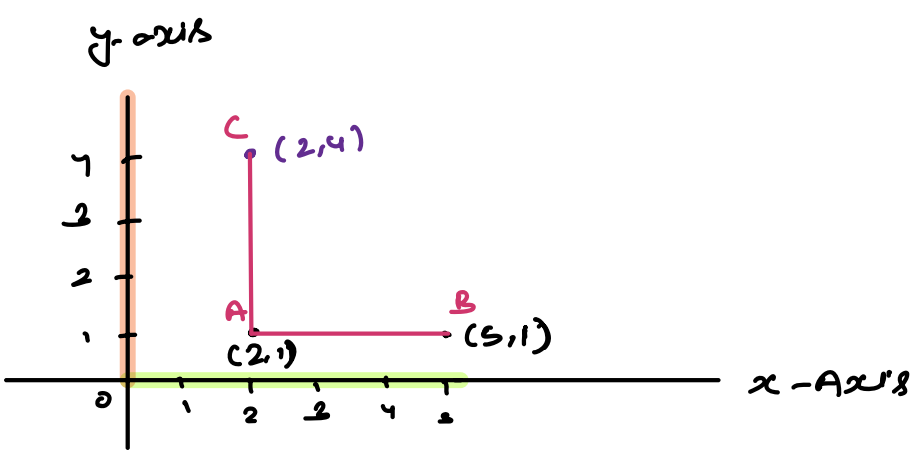
3

```
int 2Dpoints ( int x[n] , int y[n])
    HashSet<String> set = new HashSet<>();
    for (int i=0; i<n ; i++)
        // pts = x[i], y[i]
        String p = x[i] + "@" + y[i];
        set.add(p);
    }
    return hs.size();
}
```

TC = O(n)

SC = O(n)

// Geometry basics : 2D points & lines along x & y axis



point → 2 co-ordinates (x, y)

Line → 2 points

$(2, 1) \rightarrow (x, y)$

A → (2,1)

B → (5,1)

Obs 1 → For a line parallel to x-axis, y co-ordinate must be same

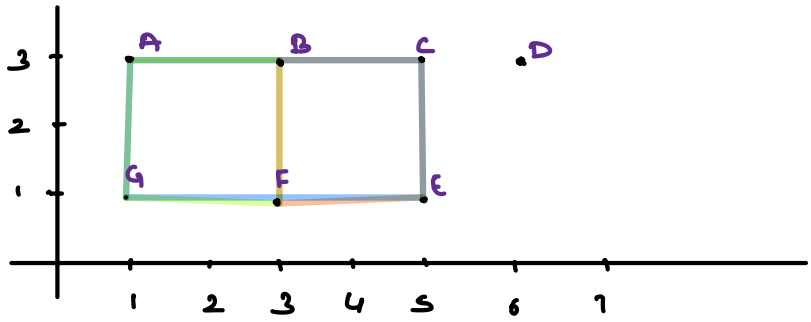
Obs 2 → For a line parallel to y-axis, x co-ordinate must be same

Google

Q1. Given N distinct points, calculate no. of rectangles formed such that sides are parallel to x-axis & y-axis.

Ex: Given 7 points $\{(1,3)\ (3,3)\ (5,3)\ (6,3)\ (5,1)\ (3,1)\ (1,1)\}$

0 1 2 3 4 5 6



Rectangles

ABGF

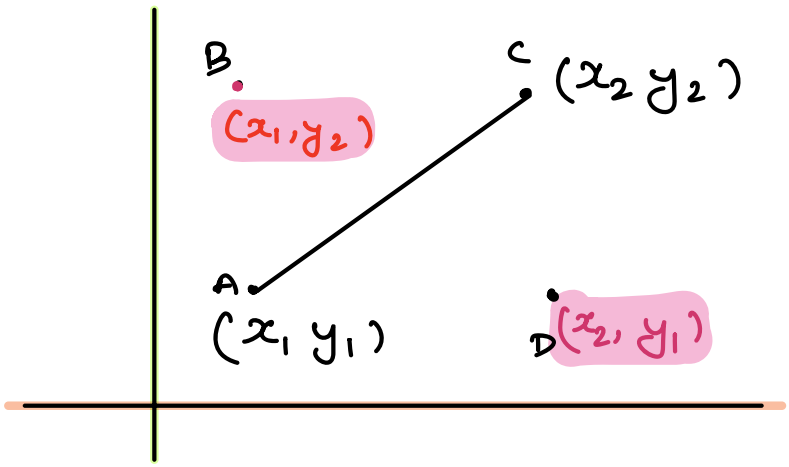
ACEG

BCEF

$\boxed{\text{Ans} = 3}$

Idea → Try to fix some points first

Idea → By fixing 2 points, we can find the complete rectangle



Obs 2 → All the points are present in your data set or not. → Can be done by Hash set

Final Idea → We need to check for each & every pair

→ Check if those two points are diagonal point

→ Figure out the other two co-ordinates & then check if they are present in the data.

$\{ (1,3) \quad (3,3) \quad (5,3) \quad (6,3) \quad (5,1) \quad (3,1) \quad (1,1) \}$

$\quad\;\; 0 \qquad\;\; 1 \qquad\;\; 2 \qquad\;\; 3 \qquad\;\; 4 \qquad\;\; 5 \qquad\;\; 6$

(x_1, y_1)	(x_2, y_2)	$(x_2\ y_1)(x_1 y_2)$
$(1,3)$	$(3,3)$	parallel to x-axis
$(1,3)$	$(5,3)$	parallel to x-axis
$(1,3)$	$(6,3)$	parallel to x-axis
$(1,3)$	$(5,1)$	$(5,3)(1,1) \rightarrow$ Rectangle
$(1,3)$	$(3,1)$	$(3,3)(1,1) \rightarrow$ Rectangle
$(1,3)$	$(1,1)$	parallel to y-axis
$(3,3)$	$(5,3)$	parallel to x-axis
$(3,3)$	$(6,3)$	parallel to x-axis
$(3,3)$	$(5,1)$	$(5,3)(3,1) \rightarrow$ Rectangle
$(3,3)$	$(3,1)$	parallel to y-axis
$(3,3)$	$(1,1)$	$(1,3)(3,1) \rightarrow$ rectangle

⋮

Final ans = count/2

Pseudocode →

```
int Rectangles (int x[n], int y[n])
    HashSet<String> hs = new HashSet<>();
    // Insert all the points in form of string

    for (i=0; i<n; i++)
        for (j=i+1; j<n; j++)
            // two points (x1, y1) = x[i], y[i]
                          (x2, y2) = x[j], y[j]
            if (x1 ≠ x2 & y1 ≠ y2)
                // search for (x2, y1) & (x1, y2)
                String P1 = x2 + "@" + y1
                String P2 = x1 + "@" + y2
                if (hs.contains(P1) && hs.contains(P2)){
                    count++;
                }
            }
        }
    }
    return count/2;
}
```

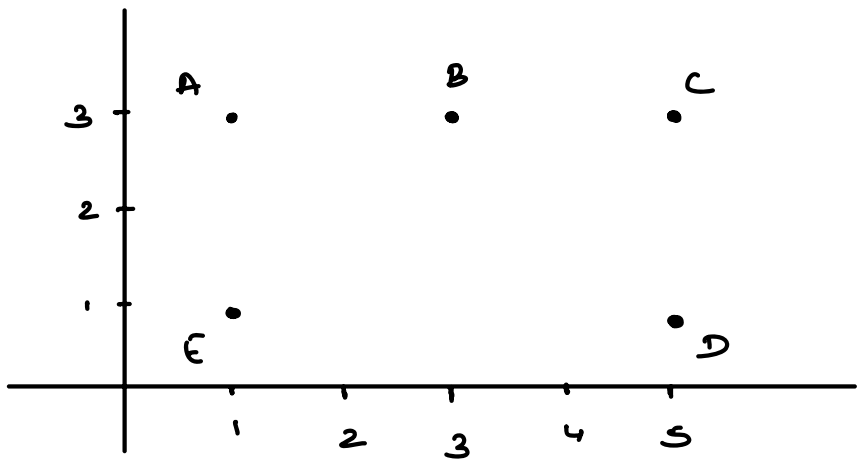
$TC = O(n^2)$

$SC = O(n)$

02. Given co-ordinates of N distinct points on a 2D plane

Count the no. of right angled triangle using the given set of points such that two small sides of a ◺ should be parallel to x-axis & y-axis

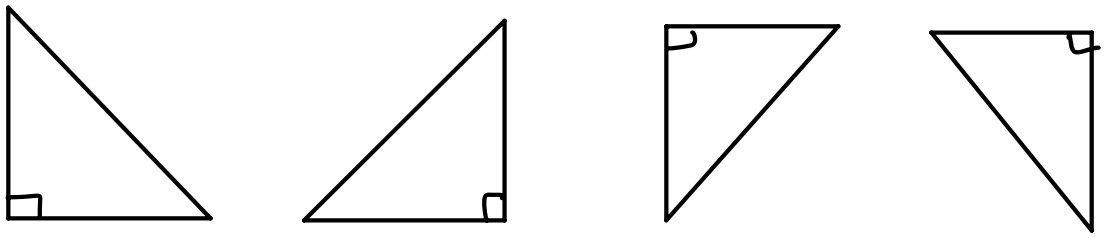
arr = { (1,3) , (3,3) , (5,3) , (5,1) , (1,1) }



◺

ABE
ACE
BCD
ACD
ADE
CDE

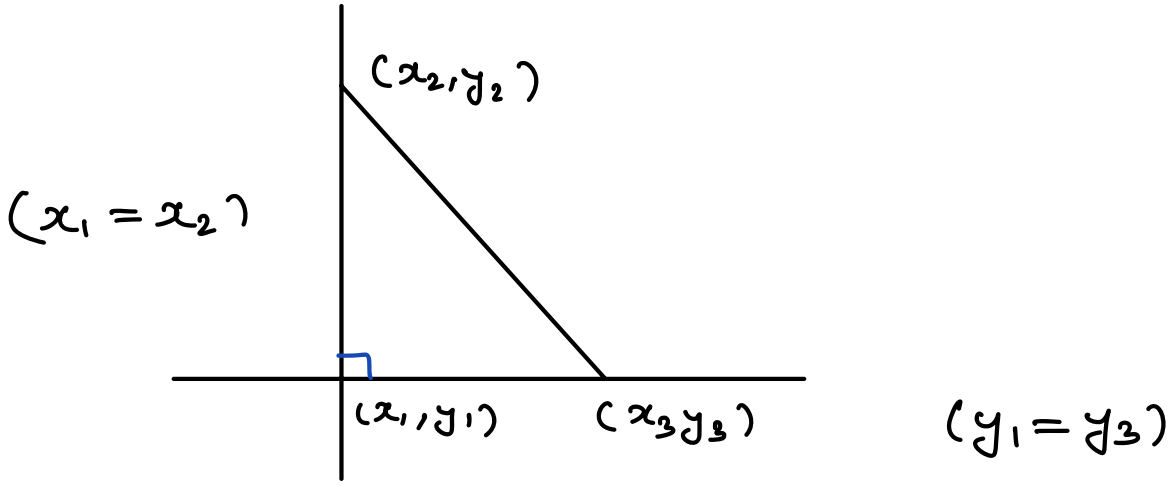
6



Brute force → 3 point are required

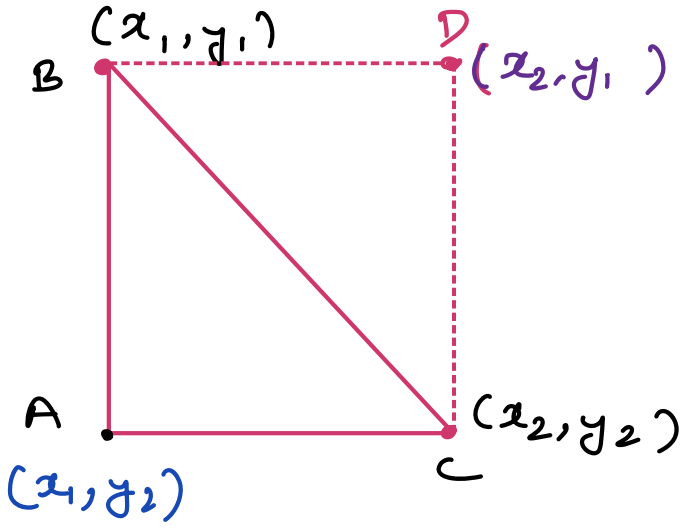
→ small sides are parallel to x-axis & y axis or not ?

TC = O(n³) SC = O(1)



Condition → $(x_1 == x_2)$ && $(y_1 == y_3)$ → count++

Idea 2



Find Idea → Consider all the pairs

→ Check for the other point if present in hashset or not.

Pseudocode

```
HashSet<String> hs = new HashSet<>();
// → Insert all the points (as string) in hs

for (i=0; i<n; i++){
    for (j=i+1; j<n; j++){
        if (x1==x2 || y1==y2) continue;
        String P1 = x[i]+"@"+y[j]   {x1, y2}
        String P2 = x[j]+"@"+y[i]   {x2, y1}
        if (hs.contains(P1)) count++;
        if (hs.contains(P2)) count++;
    }
}
return count;
```

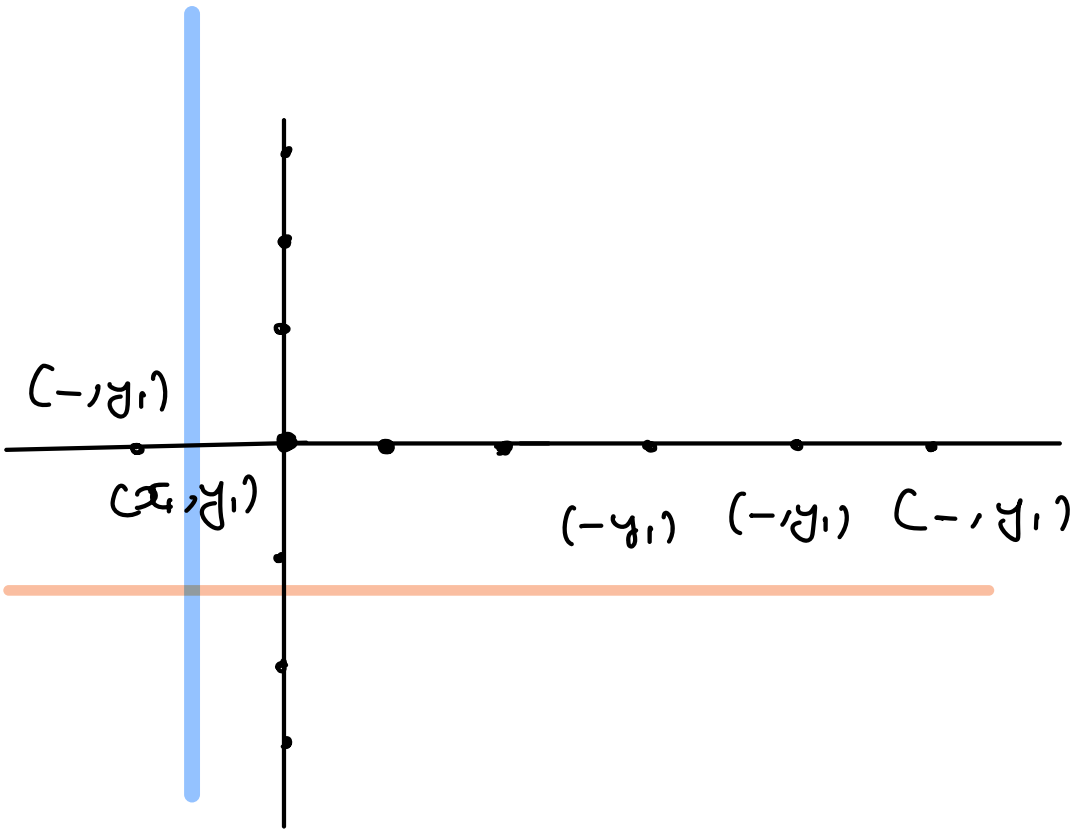
i & j loops are to consider every pair

$TC = O(n^2)$

$SC = O(n)$

Optimise again

$TC = O(n)$

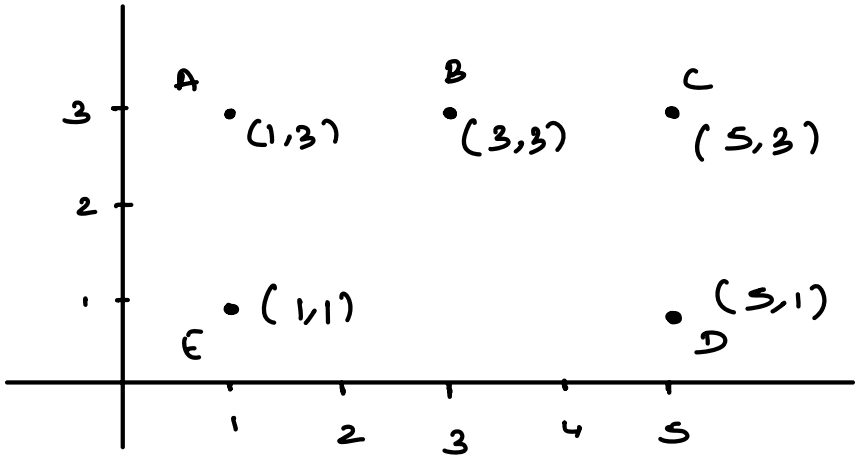


$\Delta \rightarrow$ right angle at $(x_1, y_1) \Rightarrow 6 * 6$

$= 36$

Idea $\rightarrow$ For a particular point (x_1, y_1)

$\left\{ \text{count the no. of points with same co-ordinate as } x_1 - 1 \right\} * \left\{ \text{count the no. of pts of } y \text{ co-ordinates same as } y_1 - 1 \right\}$



$A(1,3) = (2-1) * (3-1) = 2$

$B(3,3) = (1-1) * (3-1) = 0$

$C(5,3) = (2-1) * (3-1) = 2$

$D(5,1) = (2-1) * (2-1) = 1$

$E(1,1) = (2-1) * (2-1) = 1$

$\underline{\underline{6}}$

Find Idea → Create two hashmap HM1 & HM2

HM1 ↓ frequency of x-coordinates

HM2 ↓ frequency of y-co-ord

02 Iterate on array

```
for (i=0; i<n; i++) {
    int c1 = hm1.get(x[i]) - 1 ;
    int c2 = hm2.get(y[i]) - 1 ;

    ans += c1 * c2
}
return ans;
```

HM1

$\{1 \to 2, 3 \to 1, 5 \to 2\}$

HM2

$\{1 \to 2, 3 \to 3\}$

$x[\ ] = \{2, 3, 2, 4, 5\}$

creation of freqmap

```
for (i=0; i<n; i++) {
    int val = x[i];
    if (map1.contains(val) == false)
            map1.put(val, 1);
    else {
        map.put(val, map.get(val) + 1);
    }
}
```

or

```
map.put(val, map.getOrDefault(val, 0) + 1);
```

⇒

Q → Given two strings A & B, A with length N & B with length M. Count all permutations of A present in B as substring.

Note :- A & B contains lowercase characters

Eg :- A = "xyz"

B = "xyxzxy"

Ans = 2

Eg :- A = "abac"

B = "abcaacbbccbaca"

count = 4

A = xyz

Permutations →

xyz
xzy
yxz
yzx
zxy
zyx

consider as 1

Obs 1 → $\underbrace{\text{Substring of B = length of A}}_{\text{sliding window + checking of character freq}}$

Find obs → sliding window + Freq arr

A = "abac"
0 1 2 3

B = "abcaacbbccbaca"
0 1 2 3

FreqA = [2 1 1 0 0 . . . 0]
0 1 2 3 4 . . . 25
a b c d e . . .

FreqB = [~~2~~ 3 1 1 0 0 . . . 0]
0 1 2 3 4 5 . . . 25
a b c d e . . . .

TC = O(26)
= O(1)

Slide your window → add the next character & drop the last character

Pseudocode :-

```
for(i=0; i<n; i++)
    freqA[A.charAt(i) - 'a']++;
}
```

freq arr of A

TC = O(n)

```
for (i=0; i<n; i++)
    freqB [B.charAt(i) - 'a']++;
}
```

freq arr of B (n)

TC = O(n)

```
if (compare (freqA, freqB)) → count++;
```

→ TC = O(1)

```
for ( i = n; i < m; i++){
    freqB [B.charAt(i) - 'a']++;
    freqB [B.charAt(i-n) - 'a']--;
    if (compare (freqA, freqB)) → count++;
}
```

for all sliding windows

TC = O(m-n)

$$TC = O(n) + O(n) + O(1) + O(m-n)$$

$$TC = O(m)$$

$$SC = O(26) = O(1)$$

```
Public boolean compare ( int [] freqA , int [] freqB)
    for ( i=0 ; i<25 ; i++ ) {
        if ( freqA [i] != freqB [i]) return false
    }
    return True ;
}
```