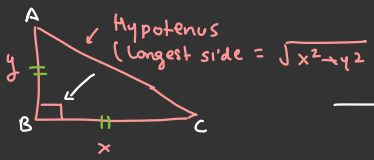
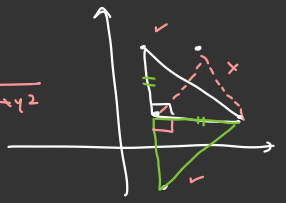
Hashing - 3

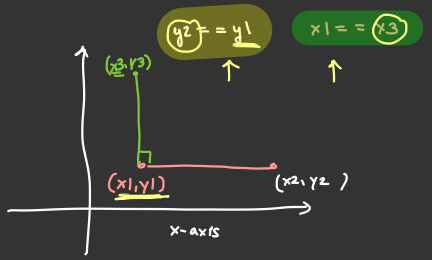
Problems
- ↳ Geometry / hashing
- ↳ String + hashing

Right Angled △

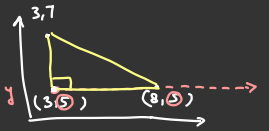




Short sides Parallel to x-axis & y-axis



(x1, y1) ←
(x2, y2)
(x3, y3)

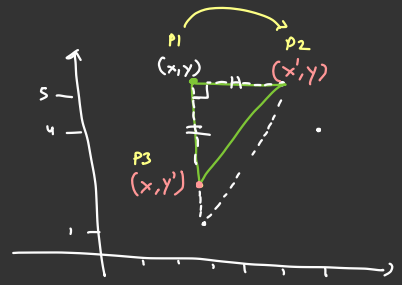


(3,5)

(4,12)
(8,5) ✓
(3,7) ✓

Examples (3) Given N points in 2D plane

(3, 8)
(3, 1)
(5, 8)
(6, 4)
(3, 2)
⋮



[P1.y == P2.y ← to form a right at any x,y
  P1.x == P3.x ←]

P1 ----> P2
P3

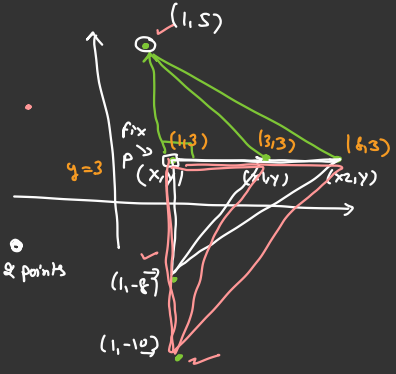
Hashmap Y[3] = 3
Hashmap X[1] = 4

Counting

P (x,y)

same y → 2
same x → 3

Triangles = 2 × 3 = (6)



(x,y) ----- 2 points
⋮
3 points

Algorithm → ① Hashmap to count the occ of given x,y

⇒ Hashmap<int, int> hmx, hmy;

Space → O(N)
Time → O(N)

```
for (i=0; i< N; i++) {
    x^i, y^i = points[i]
    hmx.put(x', hmx.getOrDefault(x^i, 0) + 1),
    hmy.put(y', hmy.getOrDefault(y', 0) + 1),
}
```

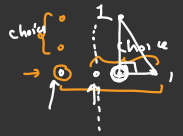
Freq.

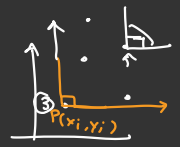
(2,3)
(2,5)
(6,1)
(3,5)

hm x	hm y
2-2	3-1
6-1	1-1
3-1	5-2

② Counting

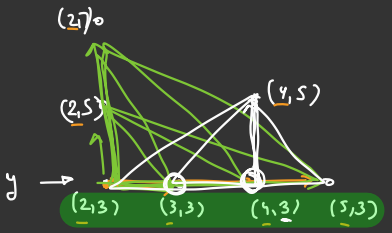
```
for(i=0, i< N; i++) {
    x_i, y_i = points(i)
    → C_x = hmx[x_i],
    → C_y = hmy[y_i]
    ans = ans + (C_x - 1) * (C_y - 1)
}
```





3

print(ans)



hmx

2 − 3

3 − 1

4 − 2

5 − 1

hmy

7 − 1

5 − 2

3 − 4

$(C_x - 1)\ (C_y - 1)$

$(3-1)\ (4-1)$

$= 2 \times 3$

$= 6$

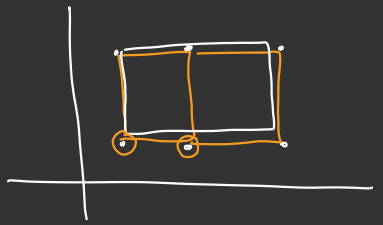
$(4,3)\ (C_x - 1)\ (C_y - 1)$

$(2-1)\ (4-1)$

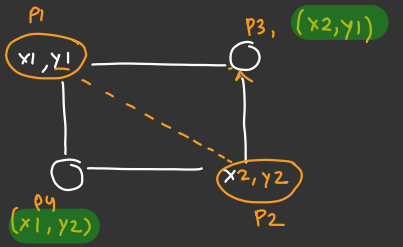
$= 1 \times 3 = 3$

Counting Rectangles

⇒ N distinct points in a 2D plane, find no of rectangles such that their sides are parallel to x-axis & y-axis



N points



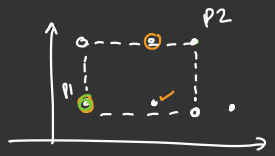
Pseudocode –

for every pair of points P1, P2

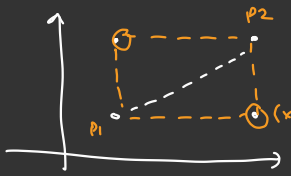
look if P3, P4 exist

cnt = cnt + 1

p(i) → P1
p(j) → P2
s.t they dont a matching x or y.

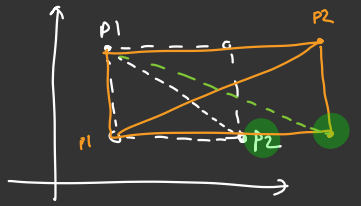


print (cnt/2);



Present

(Hashmap)



[Double Counting]

encode (x, y) {

return Str(x) + "$" + str(y)

}

CODE

points[ ], N

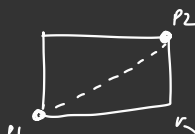
or

Pair x[ ] y[ ]

; (1,2) (3,4) (5,6) (7,8)

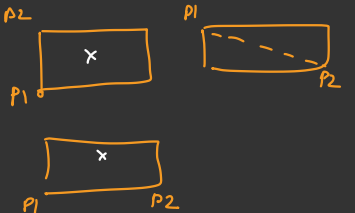
x = 1 3 5 7

y = 2 4 6 8



Twice

1,2 = "1 & 2"



Hashset <String> hs,

for (i=0 —— N-1) {

hs insert (encode (x,y));

}

// 2 loops

for (i=0 —— N-1)

for (j=i+1 —— N-1) {

if (y1 == y2 or x1 == x2) {

continue;

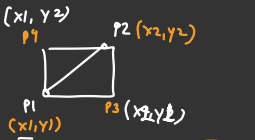
}

p4 = encode(x1,y2)

p3 = encode(x2,y1)

if (p3 & p4 are in hs) {

O(N²)



cnt = cnt + 1

}

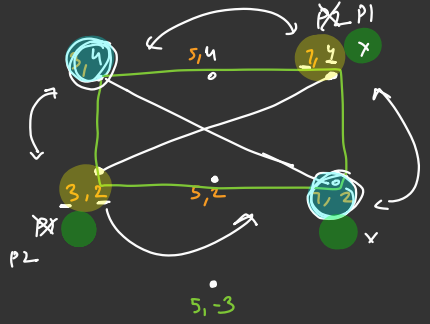
}

}

return cnt/2

$^4C_2$ = 6 ways - 4 ways

= 2 ways.



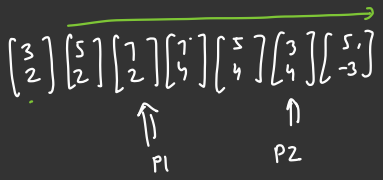
HashSet =

P2 = (7,4)
P1 = (3,2)
P3 = (7,2)
P4 = (3,4)

+1

P1 = (7,2)
P2 = (3,4)
P3 = — (3,2)
P4 = — (7,4)

+1



Hashing with Strings

2-questions

Q1. Cakewalk: Two Strings S1 and S2, Check whether S2 is a permutation of S1 or not

→ S1 = a b c d e → len1 == len2

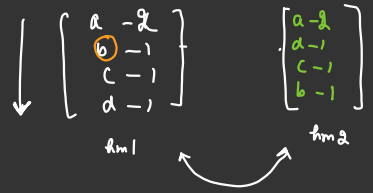
→ S2 = a a d c b

(I) Sorting S1 [a a b c d] O(NLogN)

S2 a a b c d

a b c d x

(II) hm ⇒ Iterate on S1 and update hm <char, int>



(III)

```
for(every key in hm1){
    if ( hm1[key] != hm2[key])
        return false
}
```

Q2. Largest substring with all unique characters

S = a b c a b c d d e len = 4

i = 1, j = 5

Brute force

N^2 substrings → $O(N^3)$ with $O(N)$ space

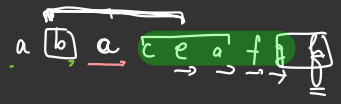
↳ check in O(N) time using hashmap

$O(N^3)$

```
for (i=0 ——— N-1) {
    for (j=i ——— N-1) {
        substring = S(i---j)
        = use hashmap to check repetition O(N)
    }
}
```

j >= i

Approach-2 Sliding Window



hash map
a - 0
b - 0

i ——— j

⇒ Sliding Window

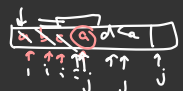
↳ Expand

↳ Contracting till we remove the repeating letter (f-1, g-1)

len = ~~4~~ ~~3~~ 3 4 5 ~~3~~ ~~4~~

Code

```
i = 0
j = 0
len = 0 ,
hashset<char> h;
while (j <= N-1) {
    if(s[j] Not in hs) {
        ⇒ hs insert (s[j]) j++}   expansion
        len = max(len, hs.size())
    }
    else {
        hs.remove(s[i]),          } Contraction
        i++
    }
}
print( ans);
```



Time & Space
→ O(N)

"GOOD Night"

25 Nov

DRY RUN

a b c d e a e f a

i j

hs

len = ~~4~~ 5