Hash Maps

Hash Sets

Let's start by 9:05

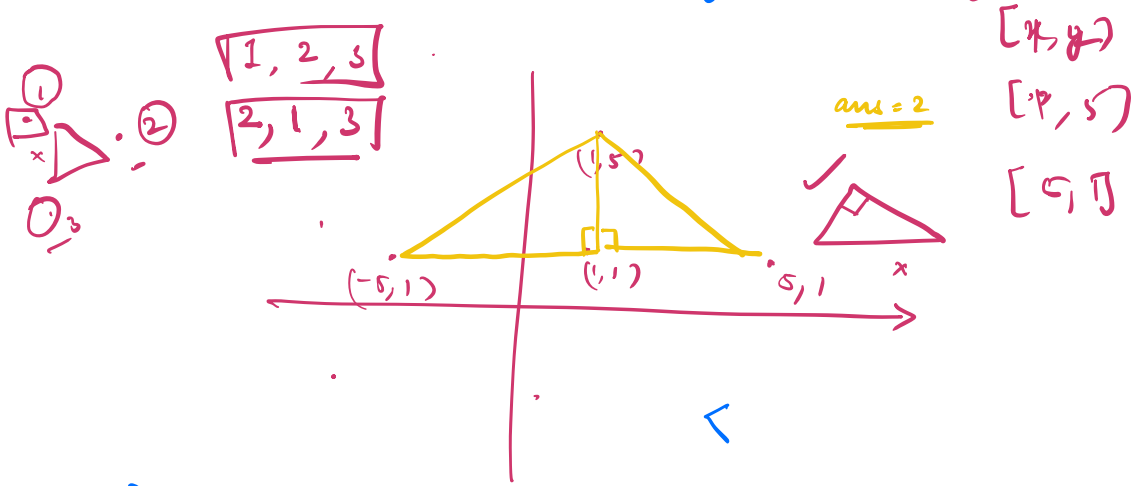
Agenda

1. Geometry + Hash Maps. ← GS → 2 Questions
2. Hashing Functions ← Bloomberg.

Rabin Karp

Q1 You are given N points on a 2D plane (x, y). Find the no. of right angled triangles in which the base and the height is parallel to x and y axis respectively.

[[x, y] [p, s] [q, r]



Brute Force

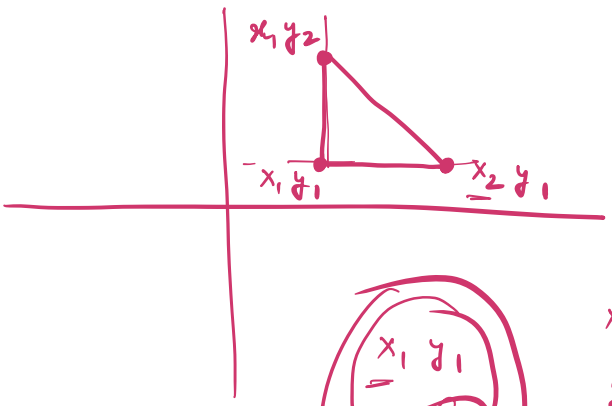
$O(N^3)$

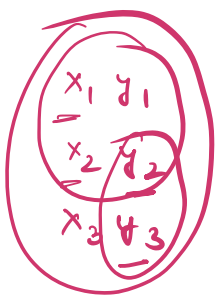
```
for ( i = 0 ; i < N ; i++ )   ← 1st point
    for ( j = i+1 ; j < N ; j++ )   ← 2nd point
        for ( k = j+1 ... k<N ) k++ o
```

cnt++

Given 3 pts.

⇒ How to check if triangle is right angled triangle. and 2 sides are parallel to x and y axis respectively





$x_1 == x_2$ ⇒ parallel to y axis

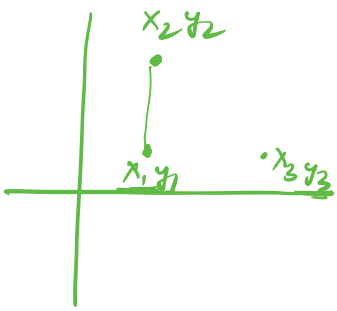
$y_2 == y_3$ ⇒ parallel to x axis.

check ($x_1, y_1, x_2, y_2, x_3, y_3$)

$x_1 == x_2$

$y_1 == y_3$

if (x_1, y_1) is assumed to be vertex opposite to hypotenuse.



$x_1 != x_3$

[

[1,2] i=0

[1,5] i=1

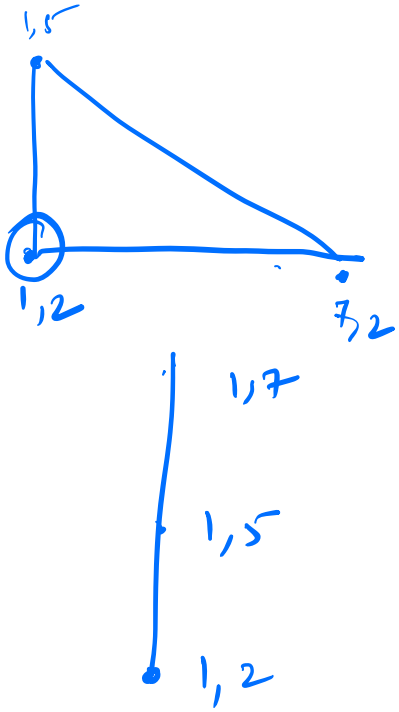
[7,2] K=2

[5,7]

[2,5]

]

[bool check (x_1 y_1 x_2 y_2 x_3 y_3)



x

x_1 y_1 as the base

$(x_1 == x_2$ and $y_1 == y_3)$ and $(x_1 != x_3$ and $y_1 != y_2)$

or

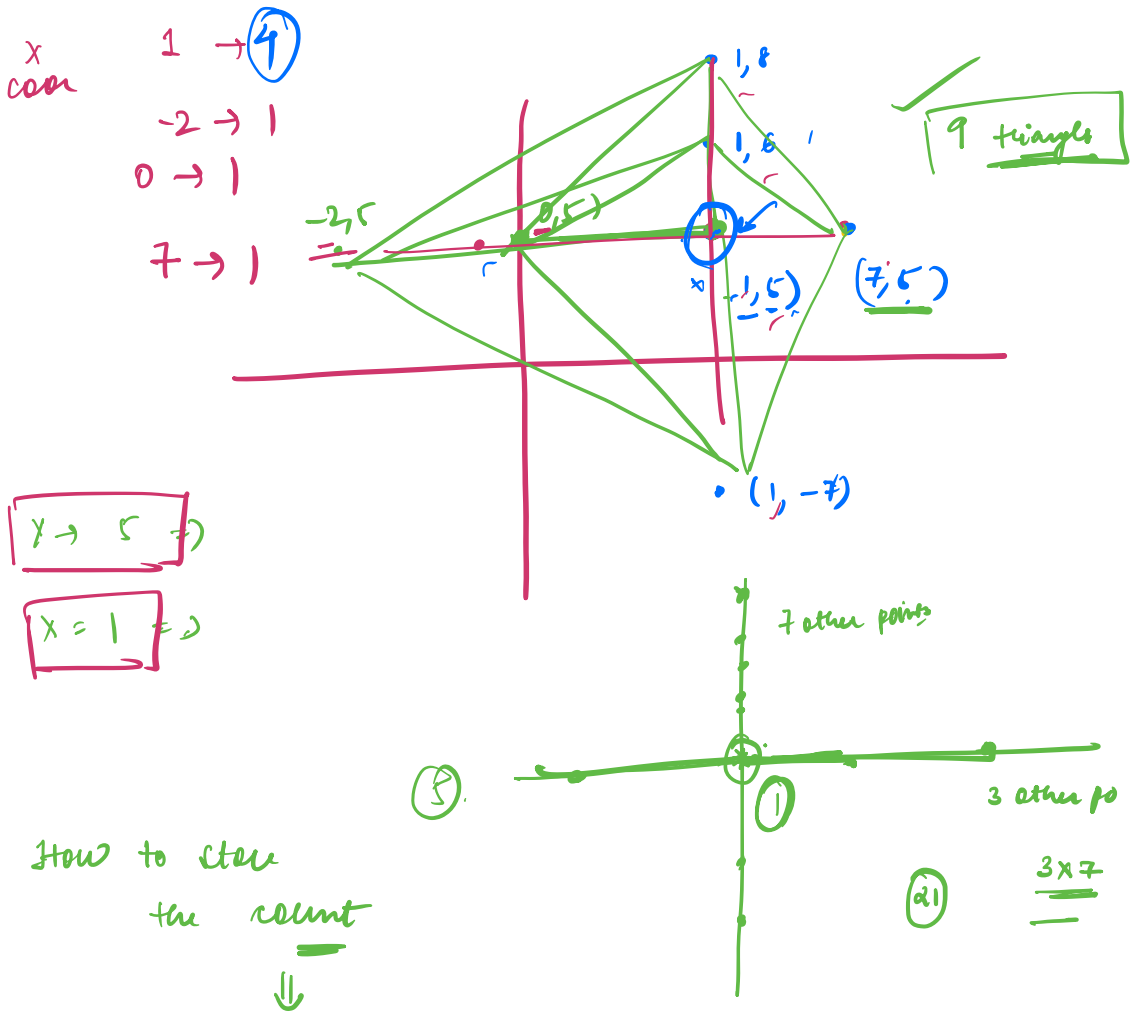
$(x_1 == x_3$ and $y_1 == y_2)$ $x_1 != x_2$ $y_1 != y_3$

x_2 y_2

ArrayList < ArrayList > points x_3 y_3

points.get(1).get(0)

get(1)



How to store the count

⇓

Hash Map

Algo :

① Iterate over all N points and store in Hashmap.

x, y

HashMap (int, int> freq-x [x] ++

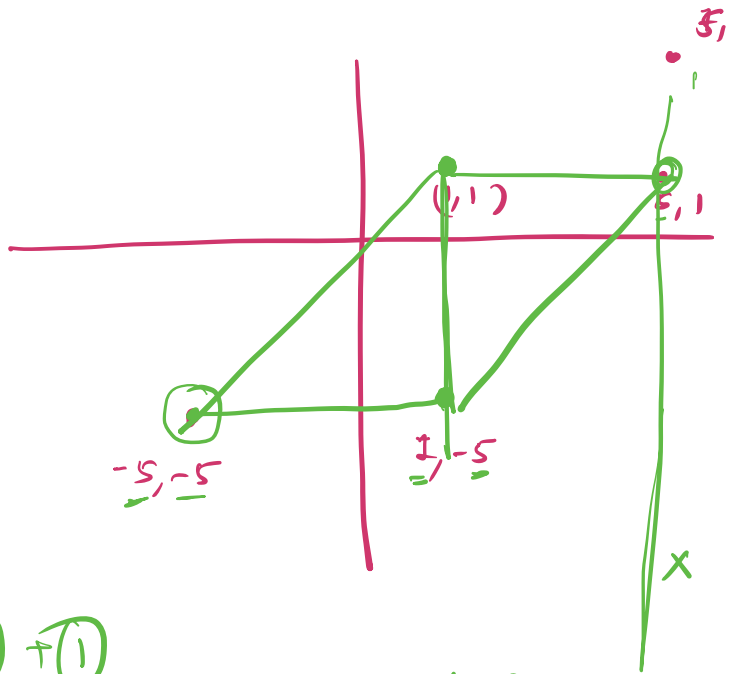
freq-y [y] ++

② Iterate over every point → considering it as the base. (1, 5)

count_x = freq[x] ← 4

count_y = freq[y] ← 4

sum += (count x - 1) * (count y - 1) ⑨



X
-5 → 1
1 → 2
5 → 2

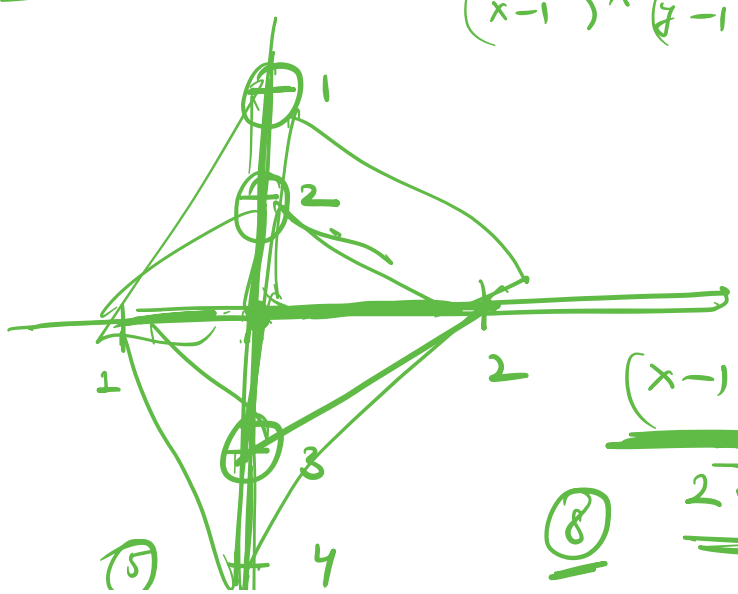
Y
-5 → 2
1 → 2
7 → 1

sum = 0 + 1 + 1

1 = 2
-5 = 2

sum = 2

(x-1) * (y-1)



-8 (3-1)

③ (x-1) * (y-1)

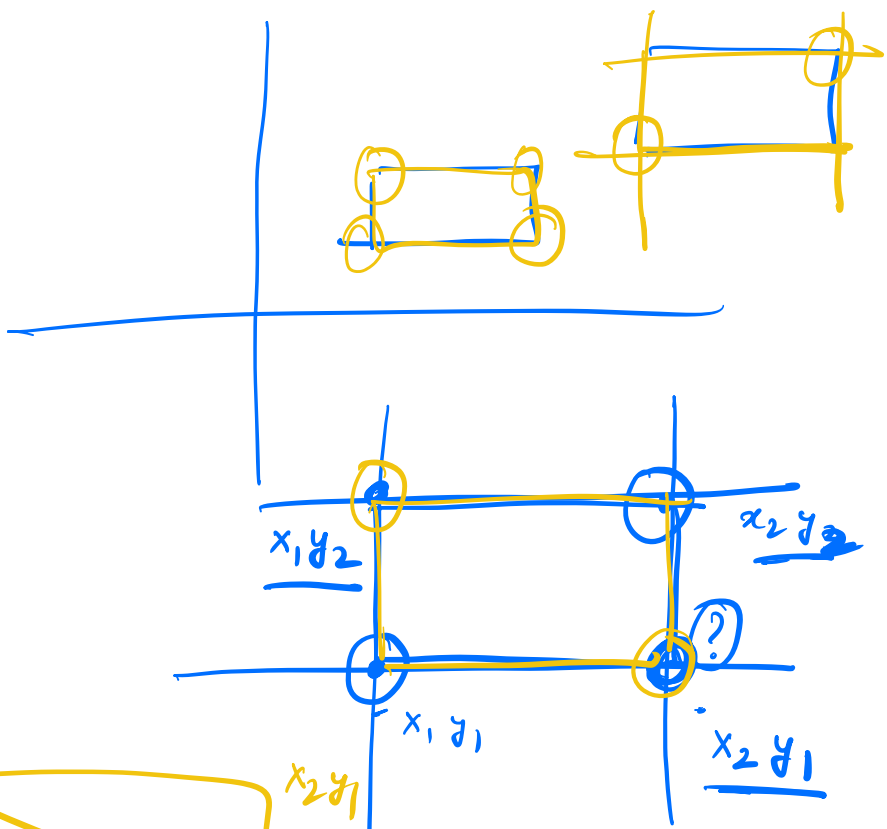
2 * 4

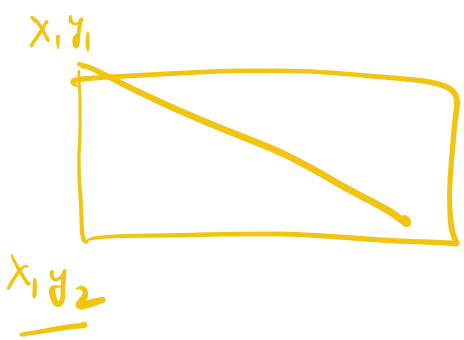
⑧

$4 = 5 - 1$

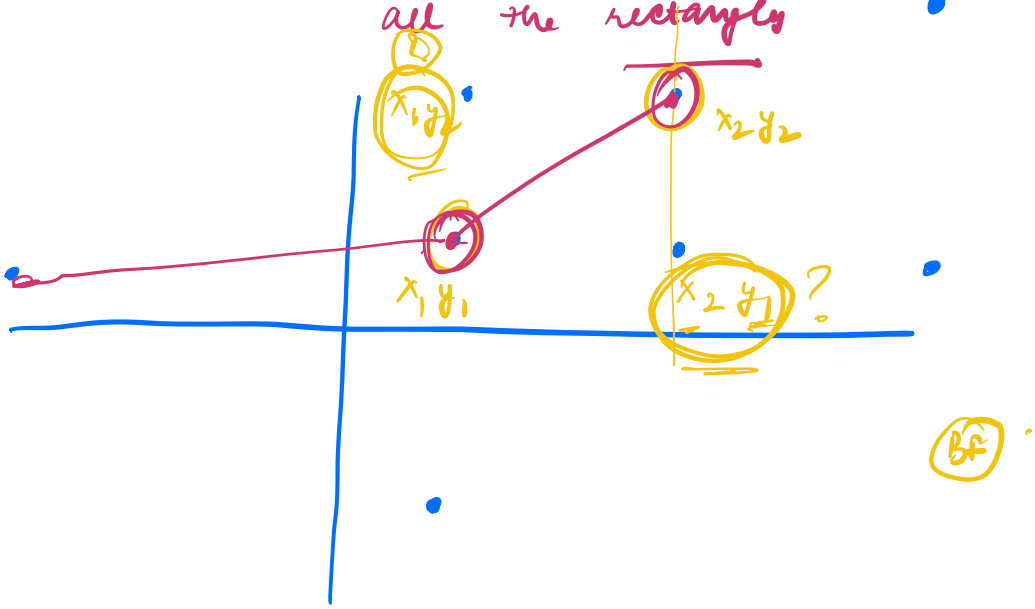
Rectangles

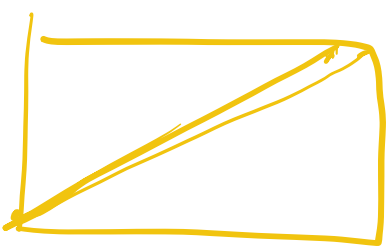
Consider Rectangles parallel to x and y axis only!





Q N points are given on a 2D plane. Return the count of all the rectangles





① Get 2 valid Diagonal ptr.

Find if other 2 ptr are needed exist or not!

```
for (i = 0 ; i < n : i++)
    x1 = points.get(i).get(0)
    y1 = points.get(i).get(1)
    for (j = i+1 , j < n ; j++)
        x2 =
        y2 =
```

N^2

N^2

```
if (x1 != x2 and y1 != y2)
```

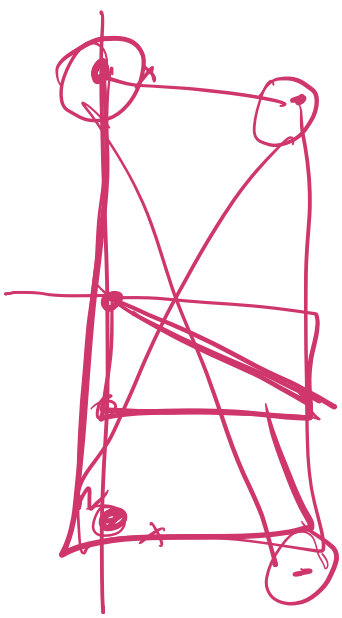
→ Now it can ACT as a diagnol

1,2
2,5
7,2
...

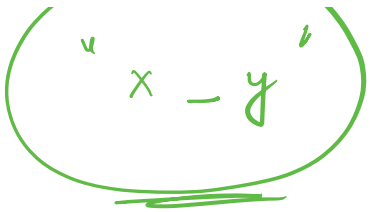
y (x_1, y_2) and (x_2, y_1) are points that are available?

```
ans += 1
```

```
class Point {
    int x
    int y
}
```



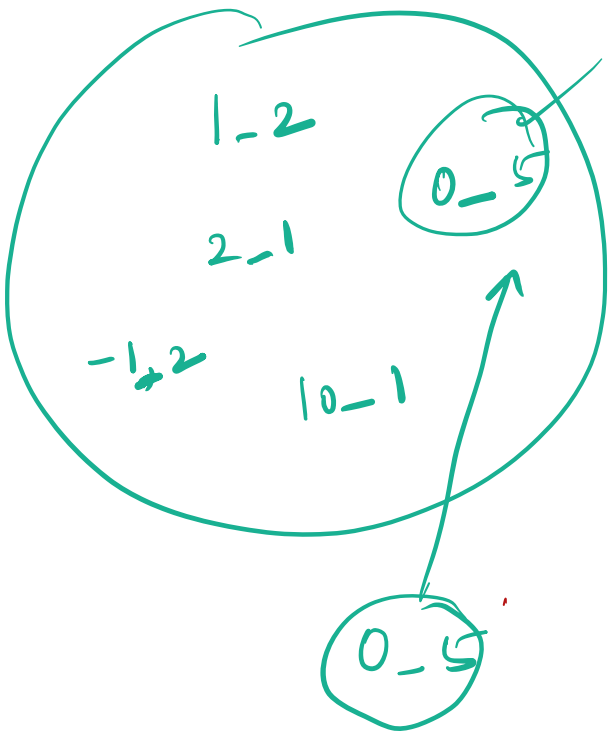
```
HashSet<Point> st;
```



Pairs

Break - 5 min.

10:35



10

All n points in Hashset

0,5

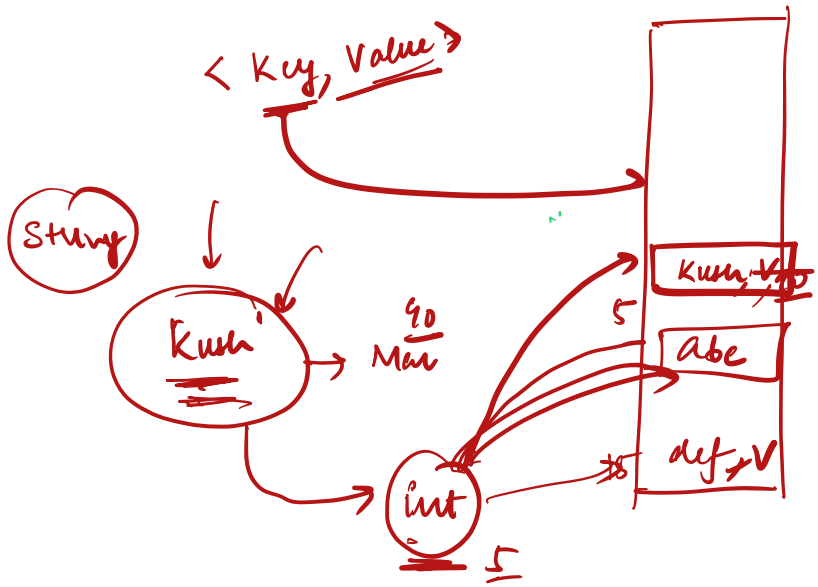
0_5

(0, 0), (0, 0) (1, 0) (0, 0)

"0_0" "0_0" "1_0" "0_0"

0_0

1_0



ASCII

! 0 !

Kush ⇒

Hashing Fn

0-25

abc → 1 + 2 + 3 = 6

def = 4 + 5 + 6 = 15

ceg

ae

cab / bad / acb

6 8

Q Consider sum of face values of characters !

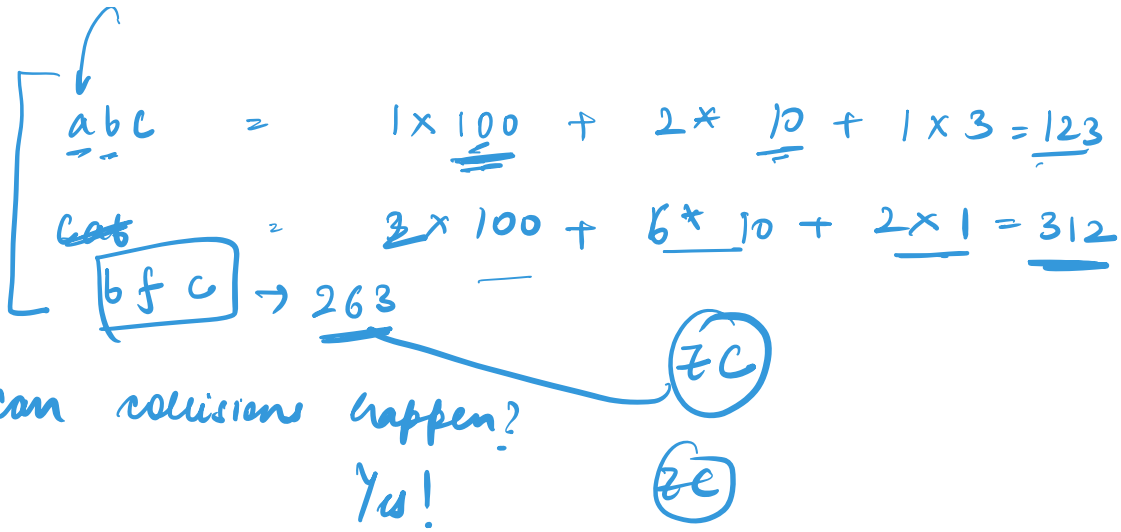
123 → Base system

321

100 * 2 + 3

3 * 100 + 2 * 10 + 1

Place Value also Matters!



can collisions happen?

Yes!

x10

999 → 900 + 90 + 9

1000 → 1000 + 0 + ...

123

345

979

a a a b c

1 2 3

. Place values as power of 20/27 and get distinct values of all strings

Large Integers

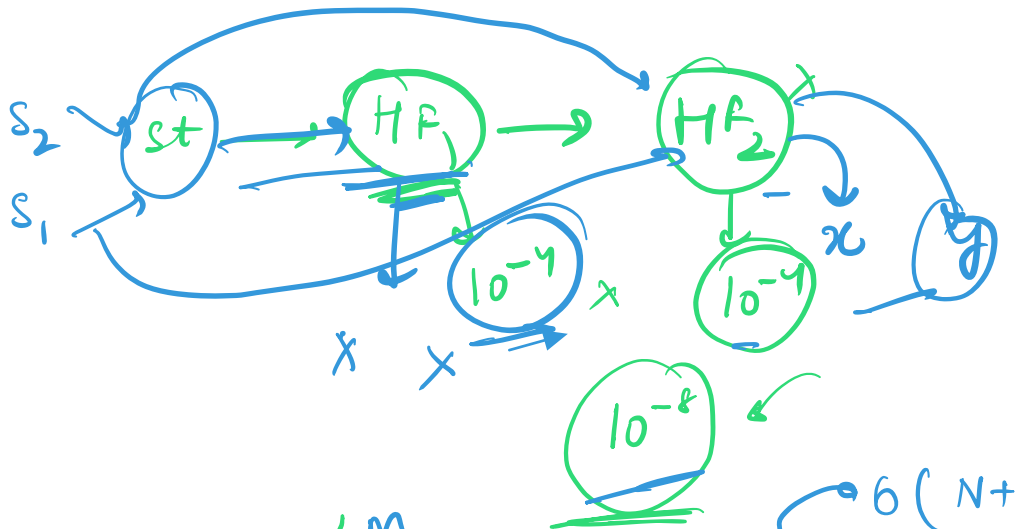
27^3

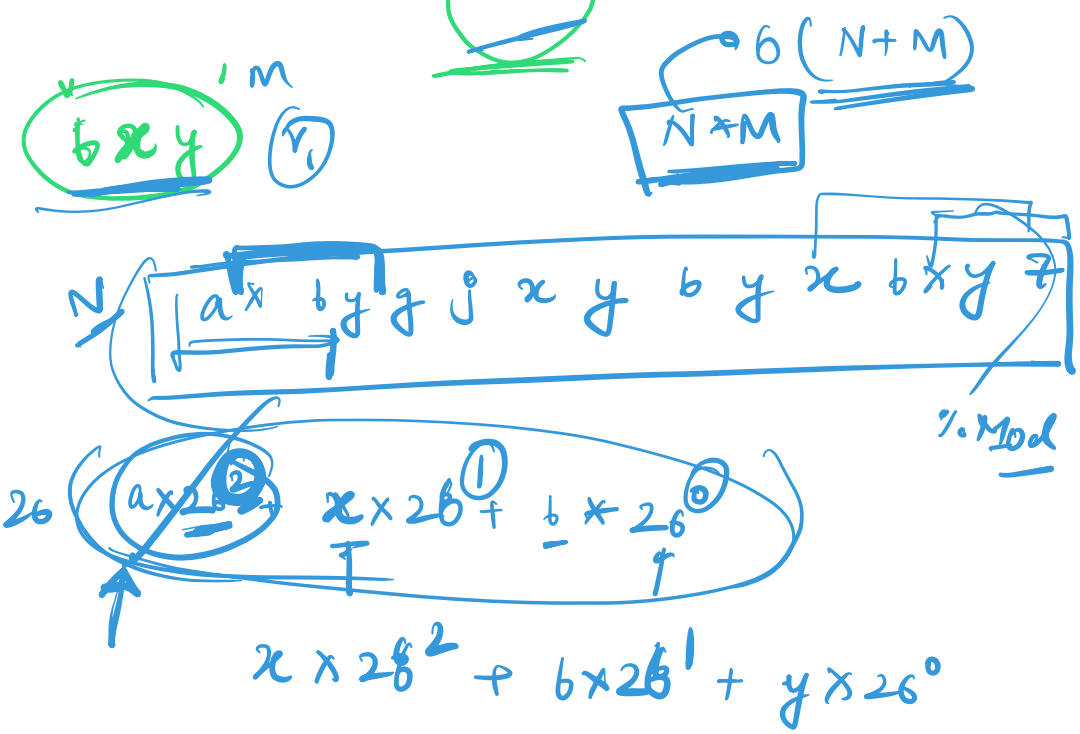
27^5

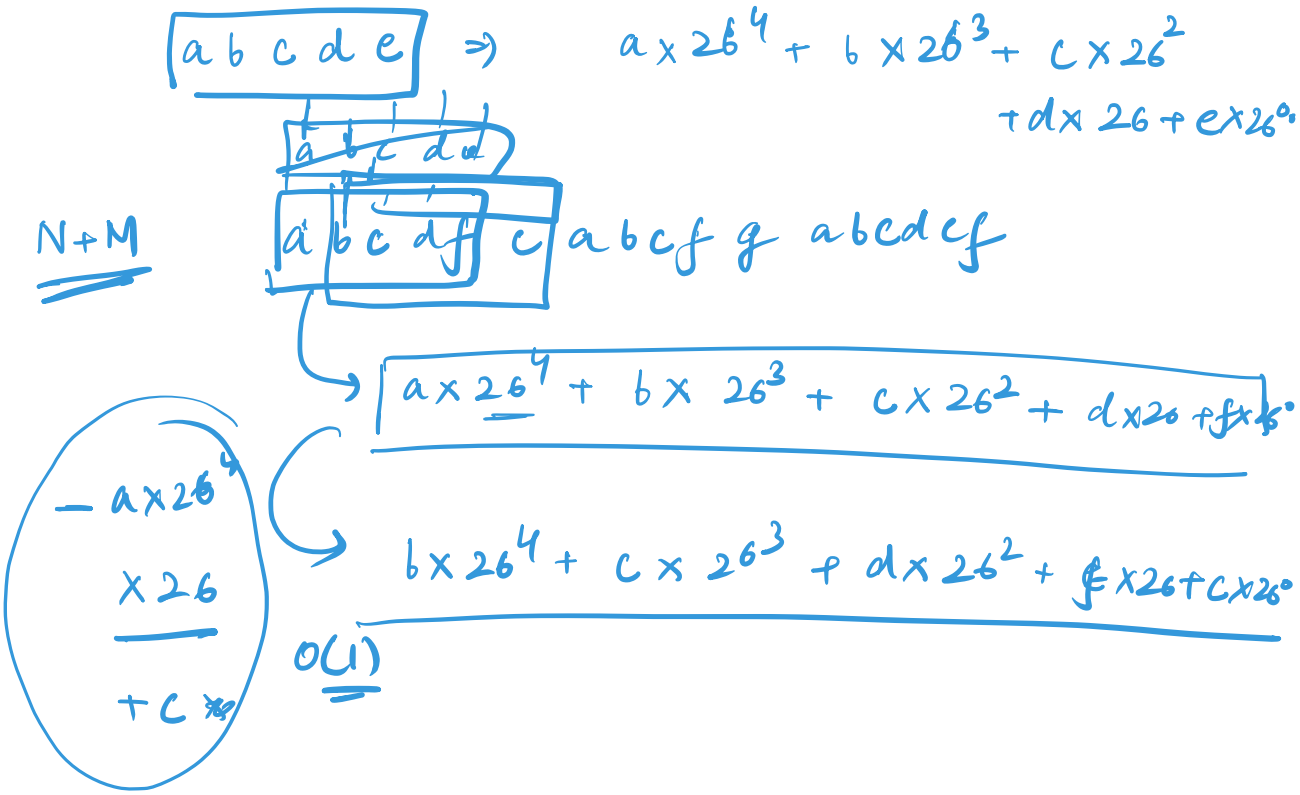
Modulo!

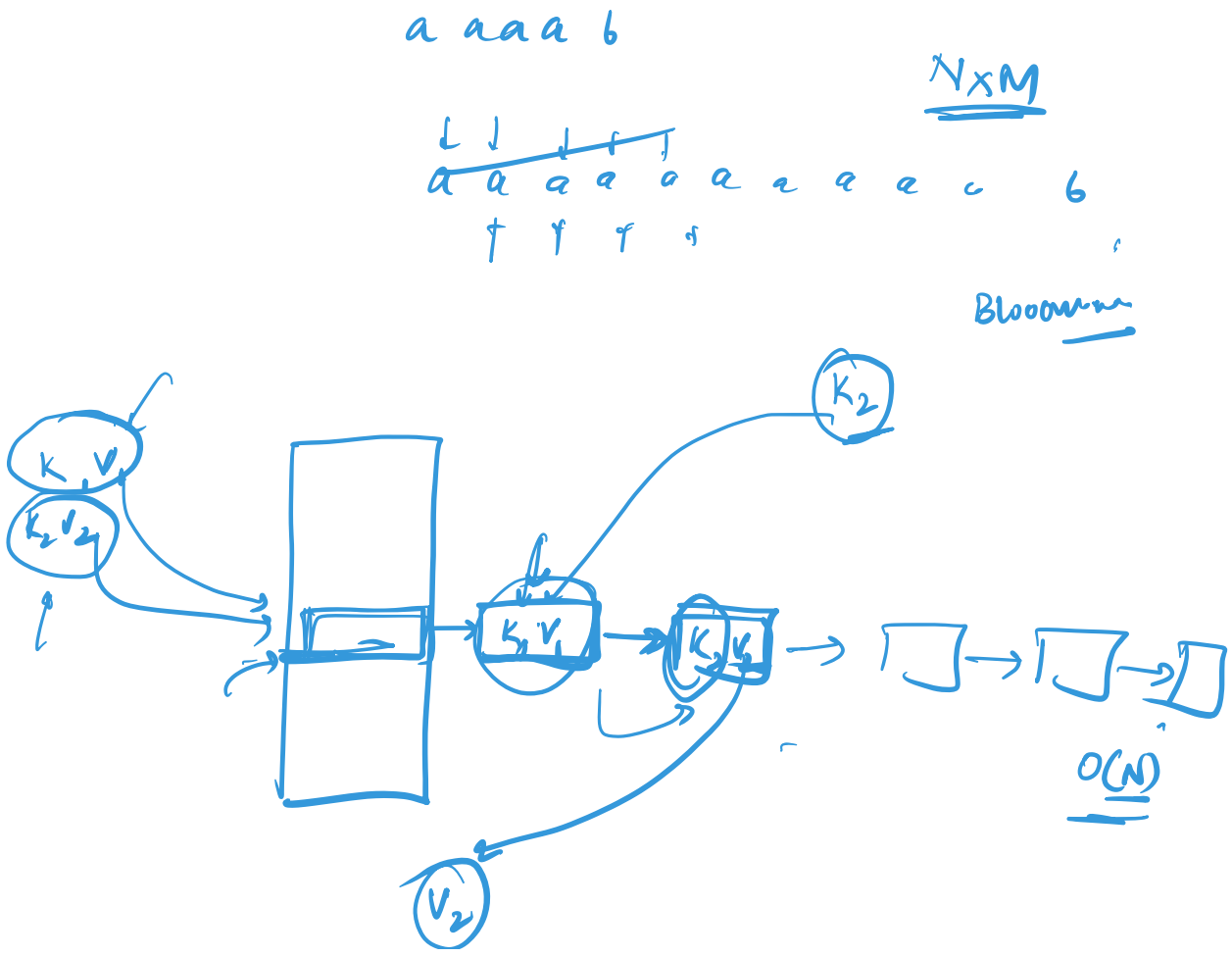
$HF_1 : \left(s[0] * 27^0 + s[1] * 27^1 + s[2] * 27^2 \right) \% 10^9+7$

Collision probability decreased to a large Extent






a b c d e ⇒ a x 26^4 + b x 26^3 + c x 26^2 + d x 26 + e x 26^0
N+M
a b c d f c a b c f g a b c d c f
a x 26^4 + b x 26^3 + c x 26^2 + d x 26 + f x 26^0
- a x 26^4
x 26
+ c x 26^0
b x 26^4 + c x 26^3 + d x 26^2 + f x 26 + c x 26^0
O(1)


a aaa b
N x M
a a a a a a a a a c b
Bloom
K1 V1
K2 V2
K2
K1 V1
K1 V1
V2
O(N)

(26)

123 = 100 + 20 + 3 ✓

positions

210

abc = 100×a + b×10 + c

0

cab

27^2 27^1 27^0

(26) 10^2

(0-15) ⇒ Base 16

2

27

16 (0-15) ✗

(16)

1 F

16 15

16 ✗

1F = 25

25

2×16+5