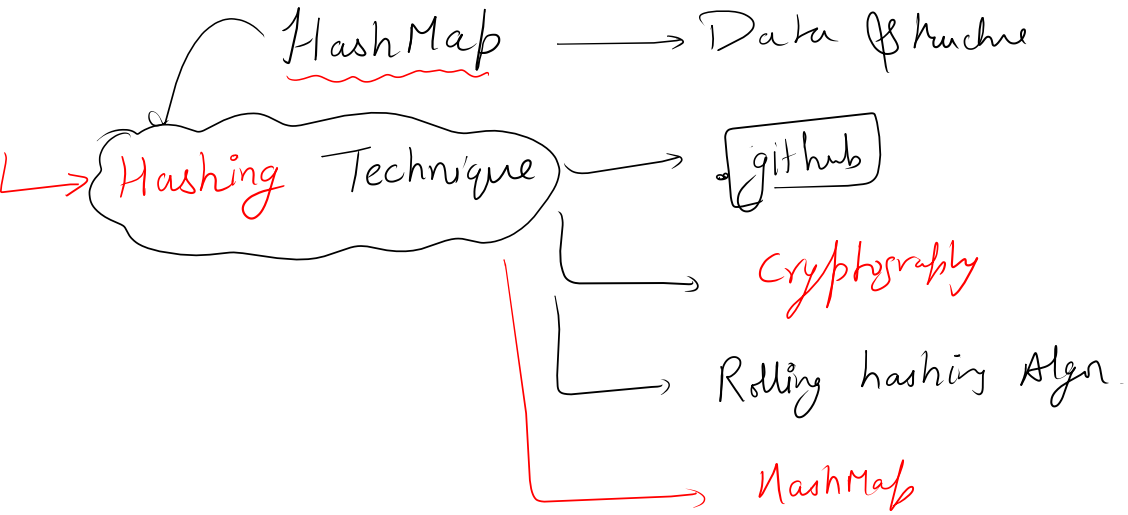
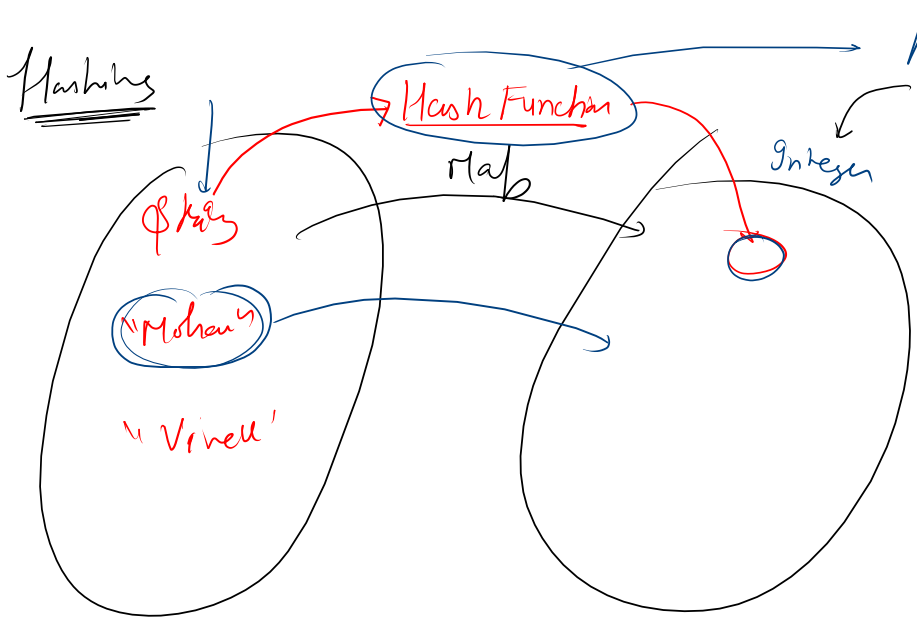
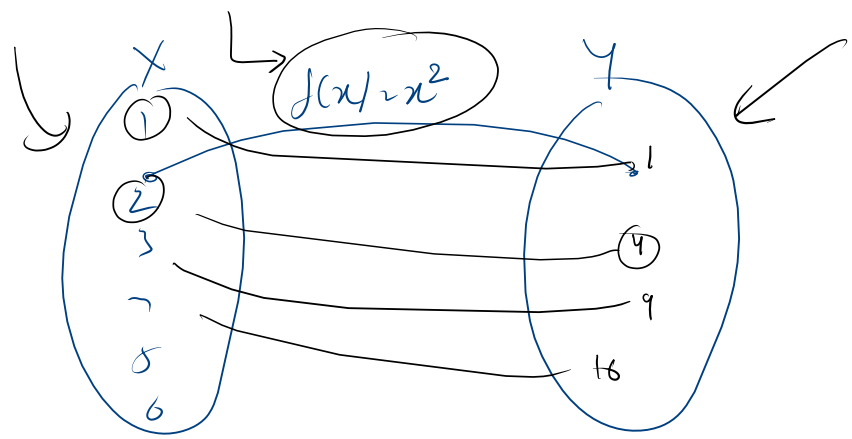
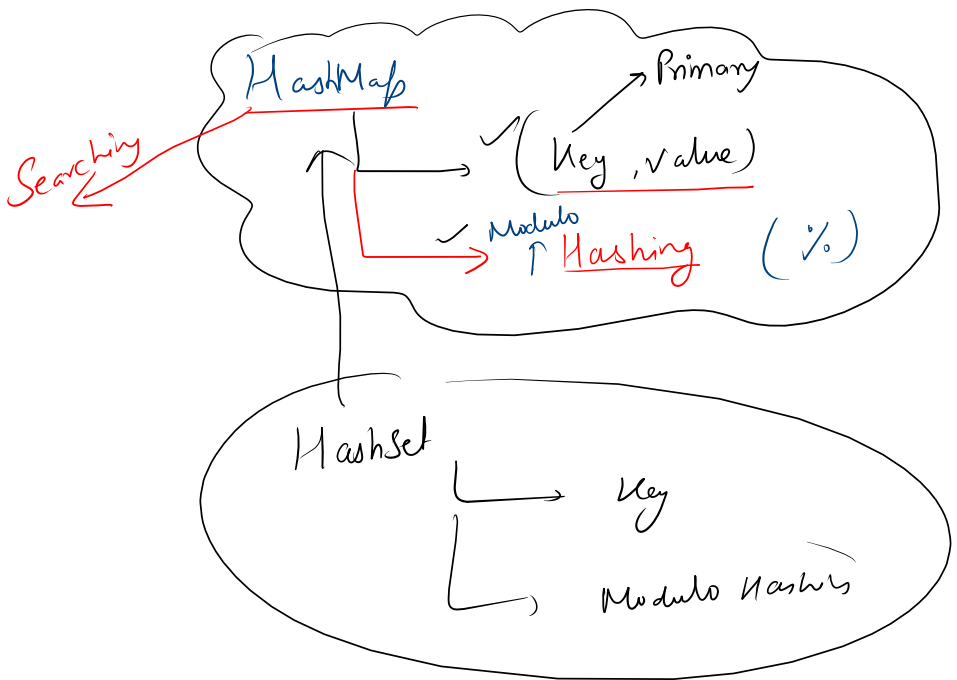
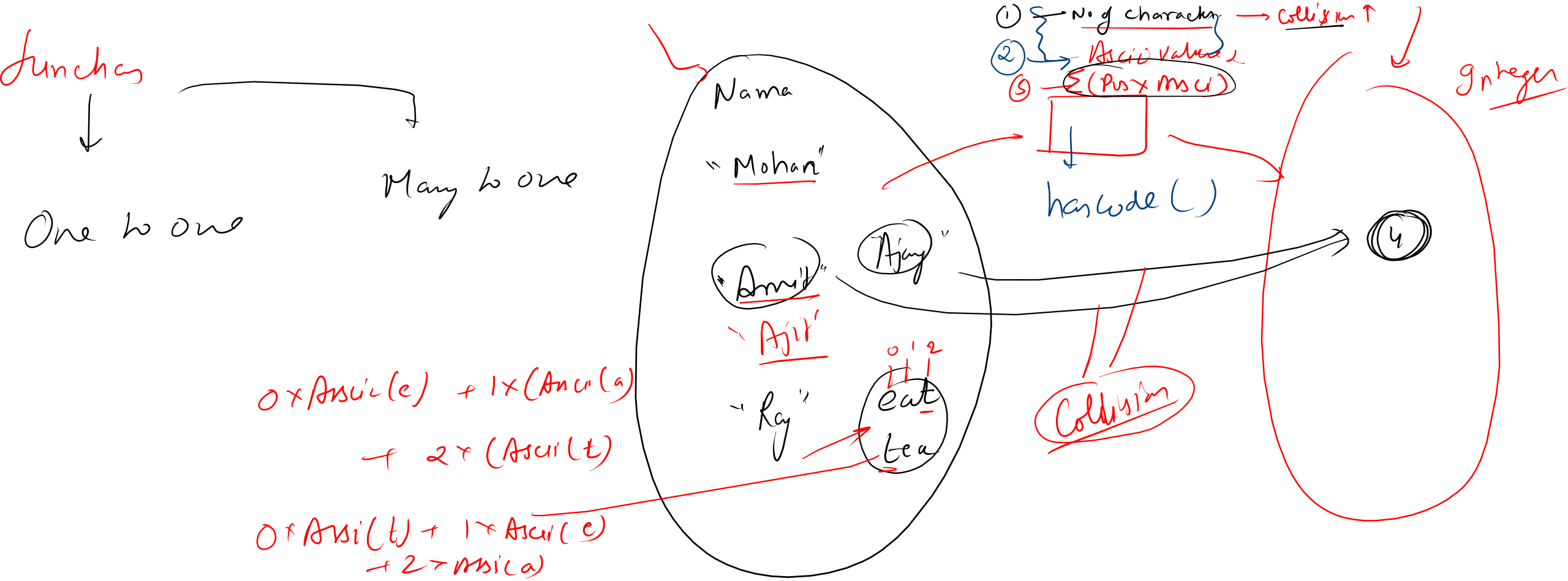

HashMap
Data Structure
Hashing Technique
github
Cryptography
Rolling hashing Algo
HashMap


Hashing
hashcode(" ");
Hash Function
Map
Integer
Keys
"Mohan"
"Vinell"


X
f(x) = x²
Y
1
2
3
4
5
6
1
4
9
16


HashMap
Primary
(key, value)
Searching
Modulo Hashing (%)
Hashset
key
Modulo Hashing


Functions
One to one
Many to one
Name
"Mohan"
"Amit"
"Ajit"
"Raj"
"Ajay"
0 1 2
eat
tea
0×Ascii(e) + 1×(Ascii(a)
+ 2×(Ascii(t)
0×Ascii(t) + 1×Ascii(e)
+ 2×Ascii(a)
① No of characters → Collision ↑
② Ascii values
③ Σ(Pos × Ascii)
hashcode()
Collision
Integer
4

Array of LinkedList

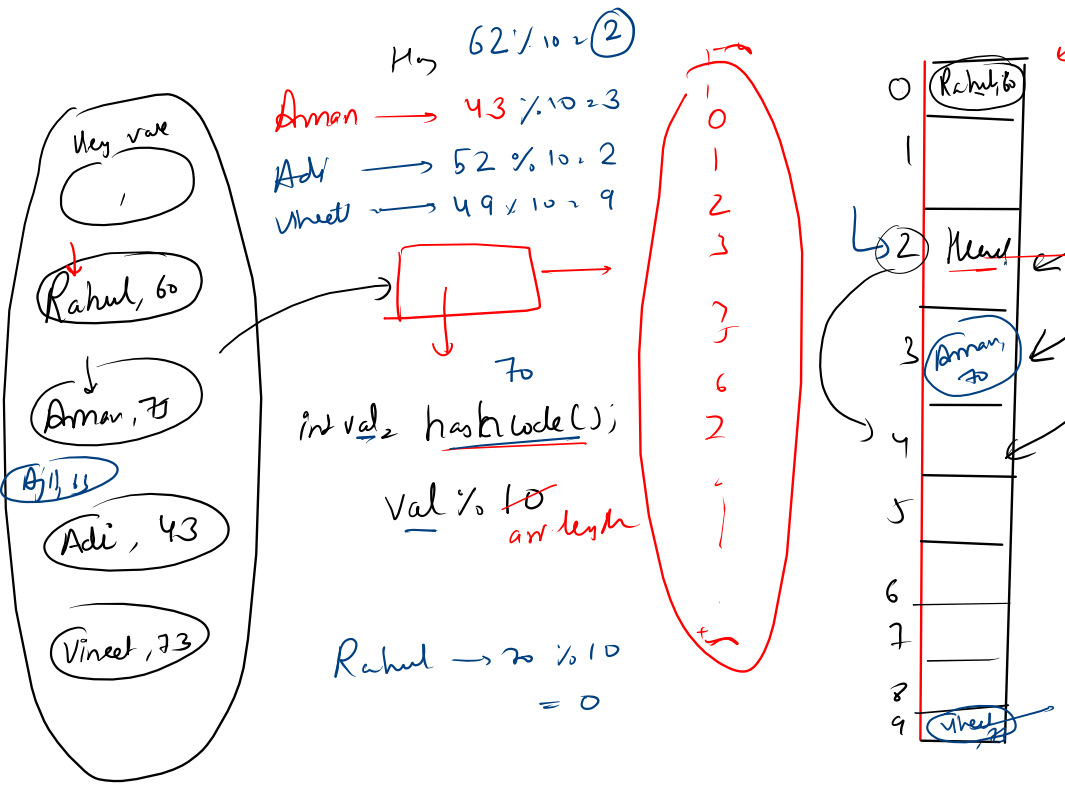
add → O(1)

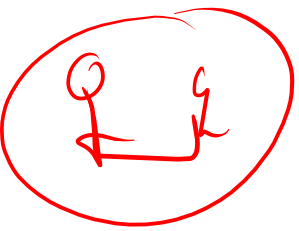
delete → O(1)

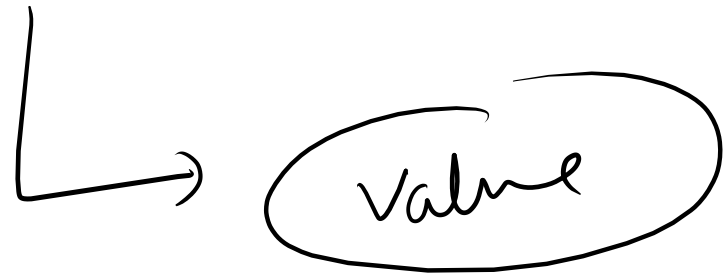
Search → O(1)

Collision Handling techniques

- Chaining
- Probing
  - Linear Probing
  - Quadratic
  - Double hashing

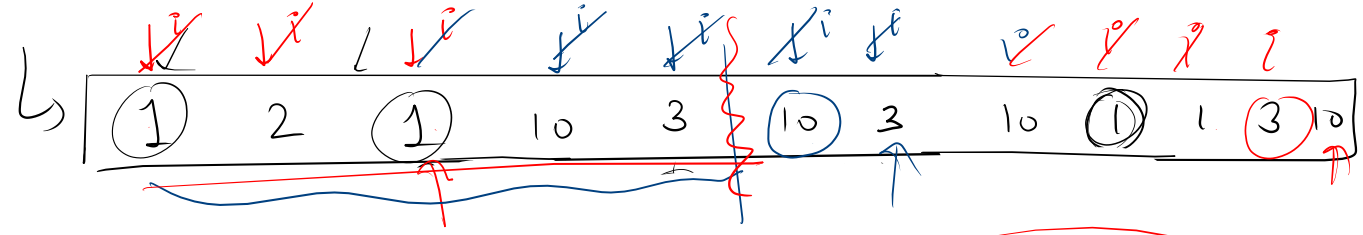


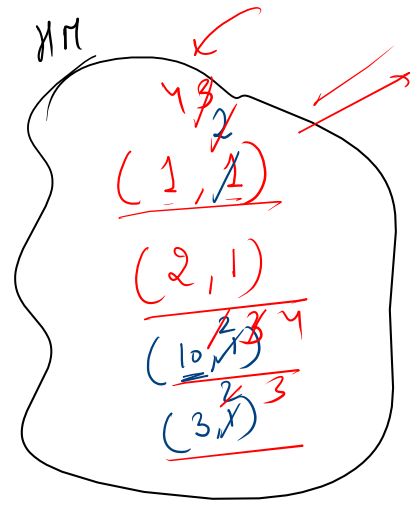


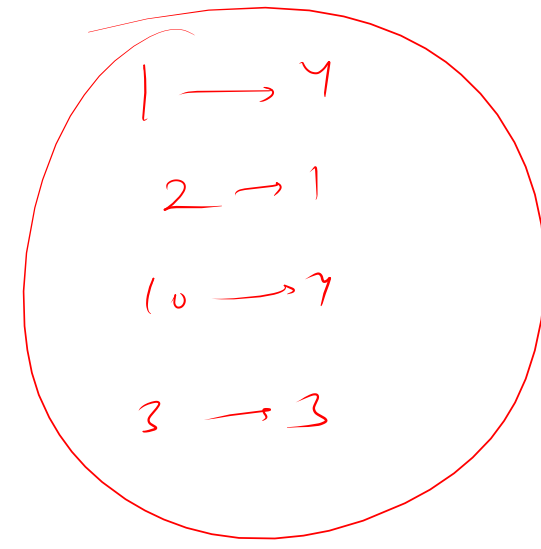


Entry


Searching
HashMap


1 2 1 10 3 10 3 10 1 1 3 10


HM
O(1)
(1,1)
(2,1)
(10,1)
(3,1)


1 → 4
2 → 1
10 → 4
3 → 3

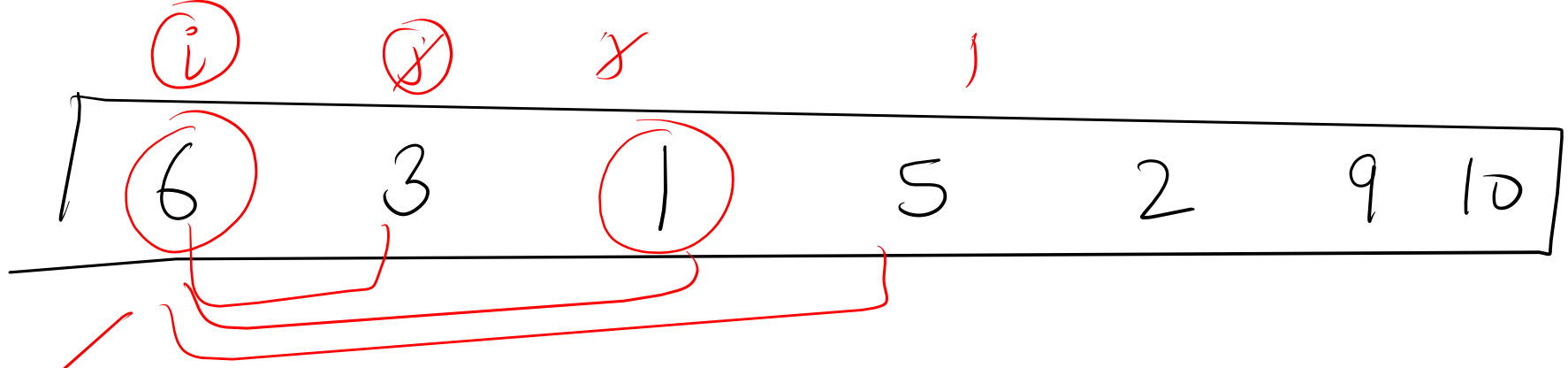
```
hm.put(arr[i], hm.getOrDefault(arr[i], 0)+1);
```

```
if(hm.containsKey(arr[i])) {
    int oldFre= hm.get(arr[i]);
    hm.put(arr[i], oldFre+1);
}else {
    hm.put(arr[i], 1);
}
```

#

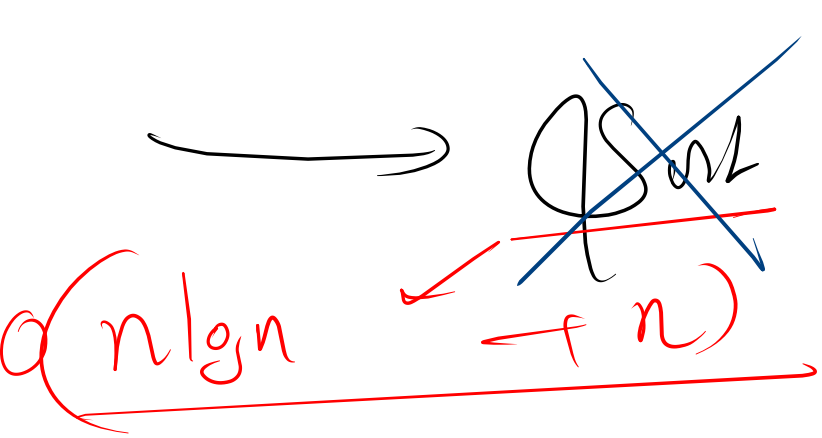
"abaabgded"

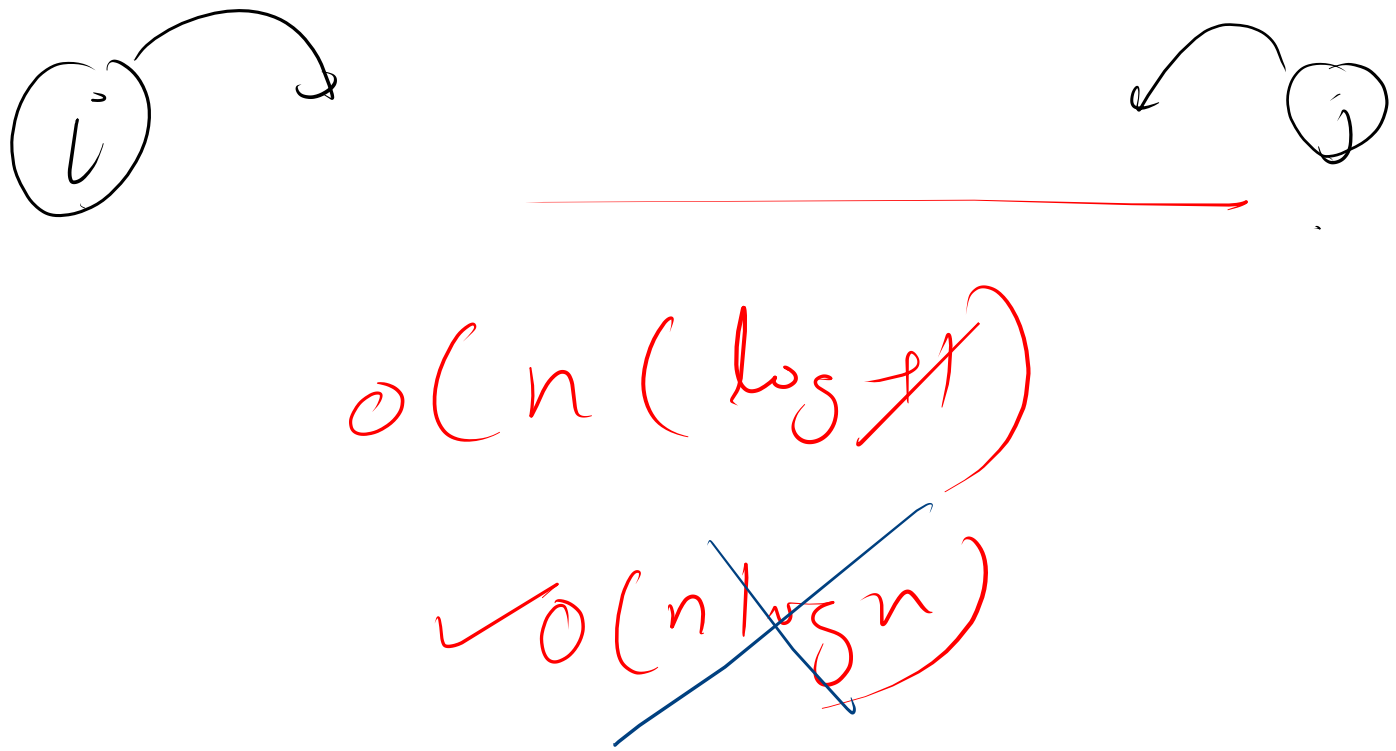
- a → 3
- b → 2
- g → 1
- d → 1
- e → 1
- f → 1



target = 7

$O(n^2)$ → (O , O)



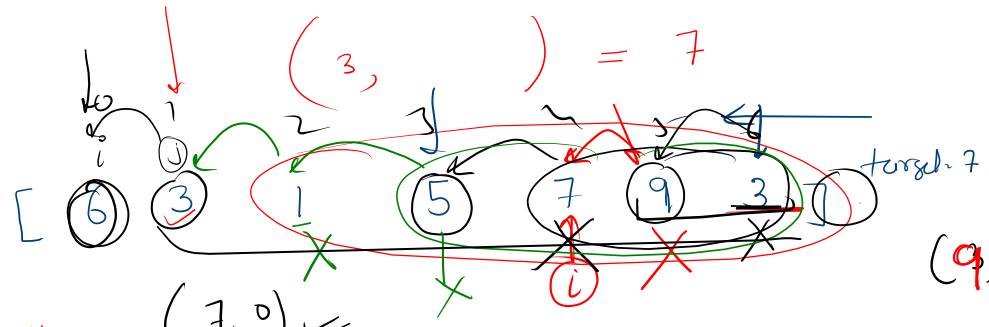


for(int i = n-1; i ≥ 0; i--){

3

3 + x = 7

x = 7 - 3 - (4)



(7, 0)

(5, 2)

(1, 6)

(3, 4)

(6, 1)

3

O(n²)

O(n) + n

O(n)

for (int i = n-1; i ≥ 0; i--) {

int secondVal = target - arr[i];

if (hm.containsKey(secondVal))
{ return {i, hm.get(secondVal)};
}

→ hm.put(arr[i], i);

(9, -2) = 7

hm

(3, 1)
(9, 5)
(7, 4)
(5, 3)
(1, 2)