Introduction of Hashmap :

get → retrieval
put → set
delete → remove

Expectation from any D.S.

val → val

get, put → optimise way

amortised O(1)

Hashing → frequency map using array

↳ for characters. 126

google

Experience → 4 years

Hashmap creation ?

Hashmap →

map.get(key) → return value

map.put(key, value) → Insert / Update

map.containskey(key) → True or false

map.getOrDefault(key, default value)

map.remove(key)

map.keySet() → set of keys

↳ we can iterate on Hashmap.

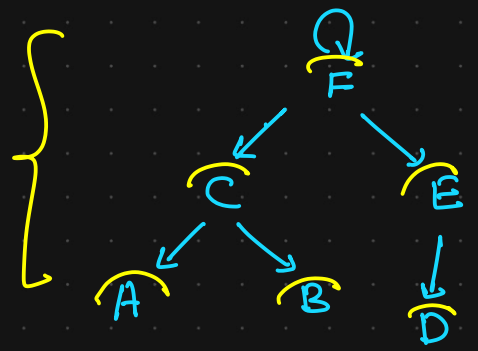
Number of employee under every manager :→

6
A C
B C
C F
D E
E F
F F → CEO

Root

↳ self Reporting Employee is CEO

TREE



A → 0
B → 0
C → 2
D → 0
E → 1
F → 5

Size of subtree from every node

Root

C → A, B
F → C, E
E → D

Generic Tree traversal

For print → post order printing.

F C E

Hashmap <String, ArrayList <String>>

Find itinerary from tickets →

4
Chennai Banglore
Bombay Delhi
Goa Chennai
Delhi Goa

Already available no chg.

[Bombay → Delhi → Goa → Chennai → Banglore]

Connectivity ≔

Chennai	Banglore
Bombay	Delhi
Goa	Chennai
Delhi	Goa

to know the starting point ≔

Banglore	False
Chennai	False
Delhi	False
Bombay	True
Goa	False

O(n)

source of journey

Step 1 → find starting point of journey

Step 2 → Iterate from starting point & solve itinerary.

Bombay → Delhi → Goa → Chennai → Banglore

O(n²)

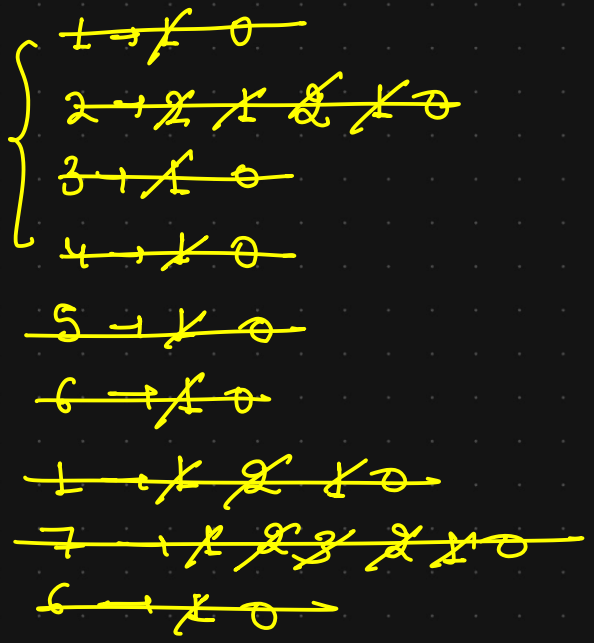
Count distinct Element in Every window of size K:

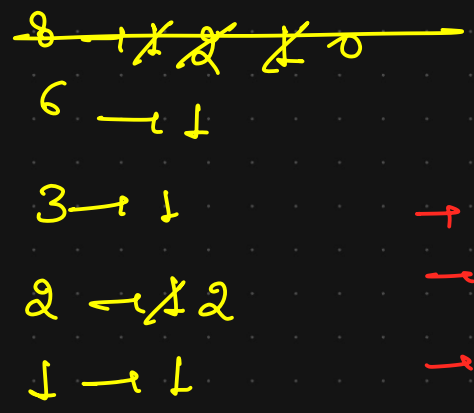
K=5

[4	4	5	4	5	5	4	4	3	3	4	4	4	4	4	5	4]			
1	2	3	2	4	5	6	2	1	7	7	6	1	7	8	8	6	3	2	1	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

i - k

4 - 5 + 1





Add in freq. map

i = 0 to i < k-1

→ add current index

→ print size of map

→ Reduce last index,

↳ if freq. 0 Remove key.

4

Chennai Banglore
Bombay Delhi
Goa Chennai
Delhi Goa

→ Banglore → False
→ Chennai → True False
→ Delhi → False
→ Bombay → True
Goa → True False

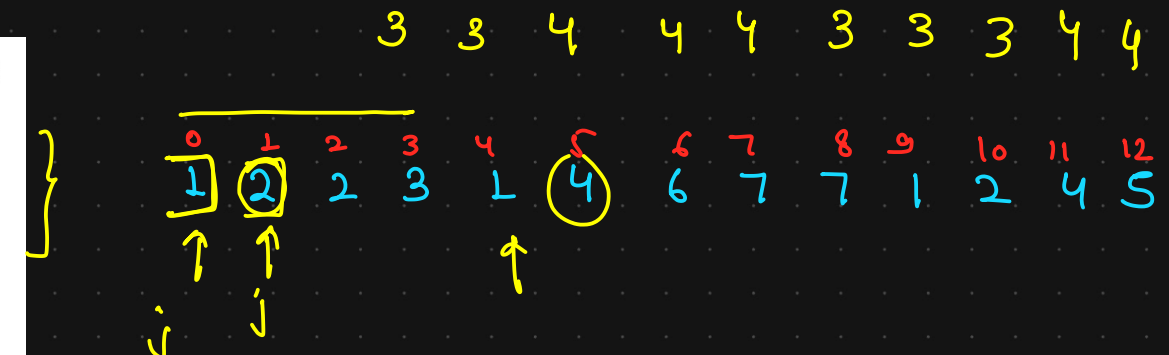
Starting point → Bombay

Bombay → Delhi → Goa → Chennai → Banglore.

```java
HashMap<Integer, Integer> fmap = new HashMap<>();
// add element from 0 to k-1
for(int i = 0; i < k - 1; i++) {
    int ofq = fmap.getOrDefault(arr[i], 0);
    fmap.put(arr[i], ofq + 1);
}
ArrayList<Integer> res = new ArrayList<>();
for(int i = k - 1; i < arr.length; i++) {
    // add current element
    int ofq = fmap.getOrDefault(arr[i], 0);
    fmap.put(arr[i], ofq + 1);
    // add size of fmap in result
    res.add(fmap.size());
    // reduce freq of first element of window, if
    int j = i - k + 1;
    ofq = fmap.get(arr[j]);
    if(ofq == 1) {
        fmap.remove(arr[j]);
    } else {
        fmap.put(arr[j], ofq - 1);
    }
}
return res;
```

Remove from freq map

Reduce freq. from freq map.



~~1 → 1~~

~~2 → 2 → 1~~

3 → 1

1 → 1

4 → 1

k=4

Array can be divided into pairs whose sum is divisible by k:

k = 10

array → 11 12 7 13 21 35 40 20 60 29 30 55 19 28

11 + 29 → 40
12 + 28 → 40
7 + 13 → 20
21 + 19 → 40
35 + 55 → 90
40 + 20 → 60
60 + 30 → 90

True.

21 k = 10

num1 = 2 × 10 + 1

$k x_1 + r_1$

num2

$k x_2 + r_2$

r_1 → remainder when num1 divide k

r_2 → remainder when num2 divide k

$r_1 = r_2$

$R_2 = k - r_1$

(num1 + num2) % k = 0

$(k x_1 + r_1 + k x_2 + r_2) \% k = 0$

$(r_1 + r_2) \% k = 0$

$r_1 + r_2 = k$

① No. in array element is Even.

num1 = k x1 + R1

num1 = divisor × quotient + remainder

array, 11 12 7 13 21 35 40 20 60 29 30 55 19 28 (5)

k <= 27

k = 10

k/2 = 5

Freq of remainder.

Remainder vs Freq.

$R_2 = k - r_1 \Rightarrow r_2 = 10 - 1 = 9$

1 → 2 → num2

2 → 1 → num2 → $R_2 = k - r_1 \Rightarrow r_2 = 10 - 2 = 8$

7 → 1 → $r_2 = 10 - 7 = 3$

3 → 1

5 → 2

0 → 4

9 → 2

8 → 1

k/2 → remainder = 0

2 * remainder ≤ k

k/2

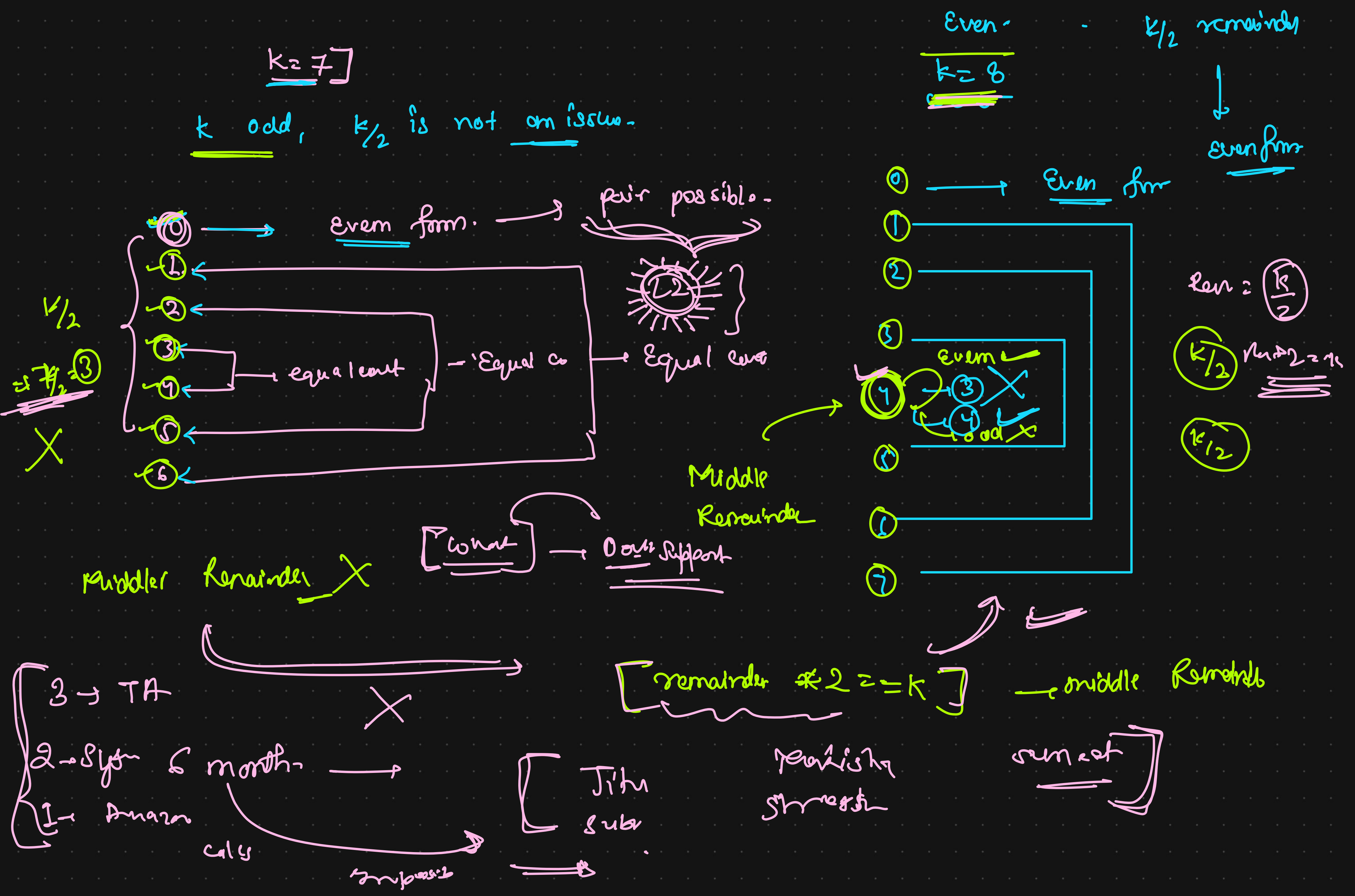
if remainder '0' already in Even fn:

0 - Remainder if Exist in Even form than no problem

4 — 2 pairs

2 — 1 pair

6 — 3 pairs


k=7
k odd, k/2 is not an issue.
even form.
pair possible.
equalent
Equal co
Equal case
k/2
Middle Remainder
Middle Remainder
Our Support
3 → TA
2 → Sys 6 month
1 → Amazon
Jitu
Subu
Even
k=8
k/2 remainder
even form
Even form
even
odd
Len : k/2
remainder * 2 == k
middle Remainder