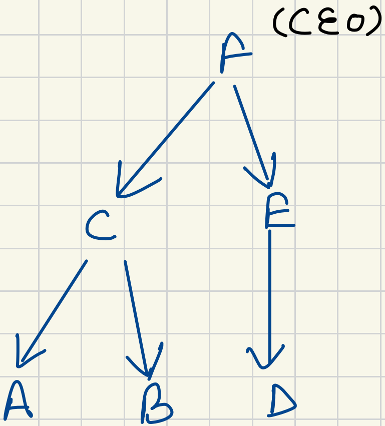
Agenda

- Employees and Manager
- Problem with a given diff.
- Array pair divisible by K
- Pair sum divisible by K
- Largest Subarray with zero sum
- Equilibrium index

Employee and Manager.

Map

Key (emp)	Value (Manager)
A	C ✓
B	C
C	F
D	E
E	F
F	F



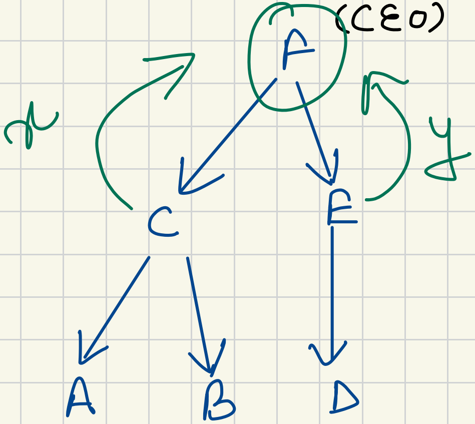
Direct reportees

Mgr	
f →	{C, E}
C →	{A, B}
E →	{D}

O/P

A →	0
B →	0
C →	2
D →	0
E →	1
F →	5

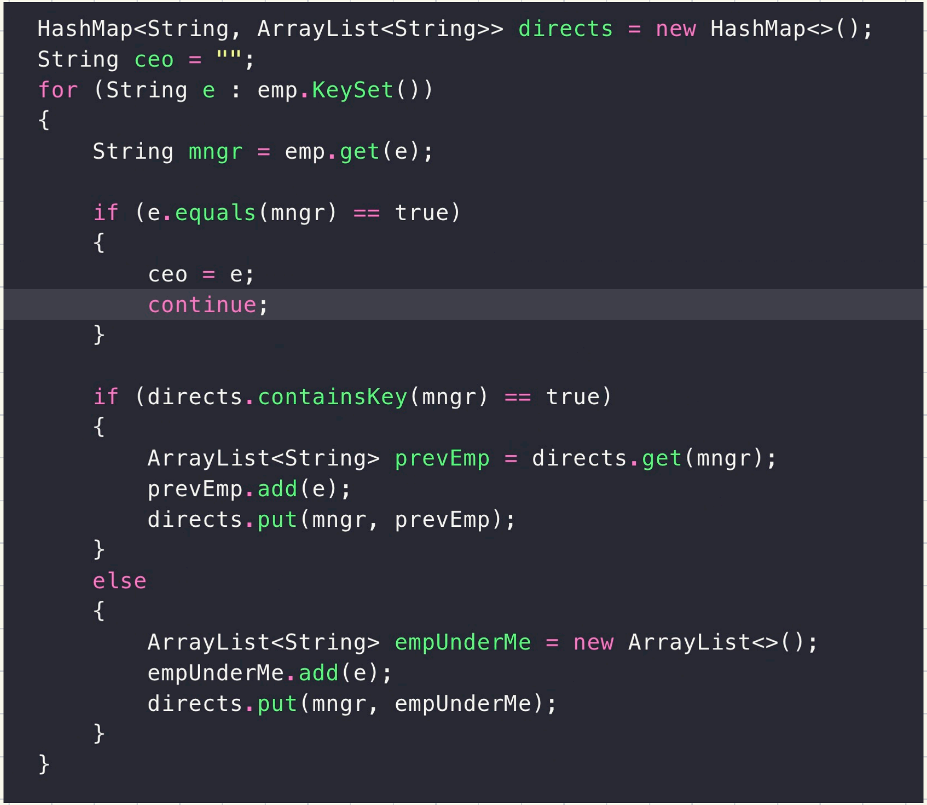
total F Manages = $(x+1)+(y+1)$ no. of Emp under a person .



No. of Emp under you = $\sum_i$ (no. of Emp Managed by your direct reportee)
+1

$$f(x) = \sum_{i}' f(x_i) + 1$$

Recursive Relation .



```java
HashMap<String, ArrayList<String>> directs = new HashMap<>();
String ceo = "";
for (String e : emp.KeySet())
{
    String mngr = emp.get(e);

    if (e.equals(mngr) == true)
    {
        ceo = e;
        continue;
    }

    if (directs.containsKey(mngr) == true)
    {
        ArrayList<String> prevEmp = directs.get(mngr);
        prevEmp.add(e);
        directs.put(mngr, prevEmp);
    }
    else
    {
        ArrayList<String> empUnderMe = new ArrayList<>();
        empUnderMe.add(e);
        directs.put(mngr, empUnderMe);
    }
}
```

Key (Emp)	Value (Manager)
A	C
B	C
C	F
D	E
E	F
F	F

Mngr	Emp
C	{A, B}
F	{C, E}
E	{D}

$\boxed{CEO = F}$

Mngr	Emp
C	{A, B}
F	{C, E}
E	{D}

CEO = F

Faith: returns no. of Emp under someone

```
fun(F)
{
    int total = 0;
    for (DirectReportees)
    {
        int x = fun(DR)
        total += (x+1)
    }
}
```

Problem with a given Difference.

int[] arr = { 5 , 10 , 3 , 2 , 50 , 80 }   B = 78

Bruteforce

```
for (int i=0 ———> n)
{
    for (int j = i+1 ———> n)
    {
        if (arr[i] - arr[j] = = B || arr[j] - arr[i] = = B)
            return true;
    }
}
return false;
```

TC : O(N²)

SC : O(1)

$$\text{int[] arr} = \left\{ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 5, & 10, & 3, & 2, & 50, & 80 \end{array} \right\}$$

$B = 78$ ✓

y

y ✓

(x, y)

$$\left.\begin{array}{l} x - y = B \\ y - x = B \end{array}\right\} \begin{array}{l} x = y + B \\ x = y - B \end{array}$$

✓ $y = 80$

$$\left.\begin{array}{l} x = 80 + 78 = 158 \\ \cancel{x} = 80 - 78 = 2 \end{array}\right\}$$

✓

find()

$\downarrow$

O(1)

HashSet

5, 10, 3, 2

50

$y = 2$

$$\left.\begin{array}{l} x = 2 + 78 = 80 \\ x = 2 - 78 = -76 \end{array}\right\}$$

Array Pair divisible by K.

int[] arr = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }   K=5

n/2 → pairs such sum of each pair is divisible by K

(9, 1) (8, 2) (7, 3) (6, 4) (5, 10)

↓ ↓ ↓ ↓ ↓

10 10 10 10 15

$arr = \{1, \cancel{6}, \cancel{14}, 24, \cancel{19}, \cancel{21}\}_6$ $K = 5$

✓ $\boxed{3}$ $(n)^2$

return true/false

$(14, 6)$ $(21, 19)$ $(1, 24)$

$(x + y)$ divisible by K

$$x + y = \alpha * K$$

$$(d_1 * K + r_1) + (d_2 * K + r_2) = \alpha * K$$

$$\underbrace{K * (d_1 + d_2)}_{\text{is divisible by } K} + \underbrace{(r_1 + r_2)}_{\text{has to be divisible by } K} = \alpha * K$$

$\{$equal to $K\}$

or

equal to 0

$$\underline{\underline{0 \le r_1, r_2 < K}}$$

$$K\overline{)\,x\,}(d$$

$$\underline{\qquad\qquad}$$

$$r$$

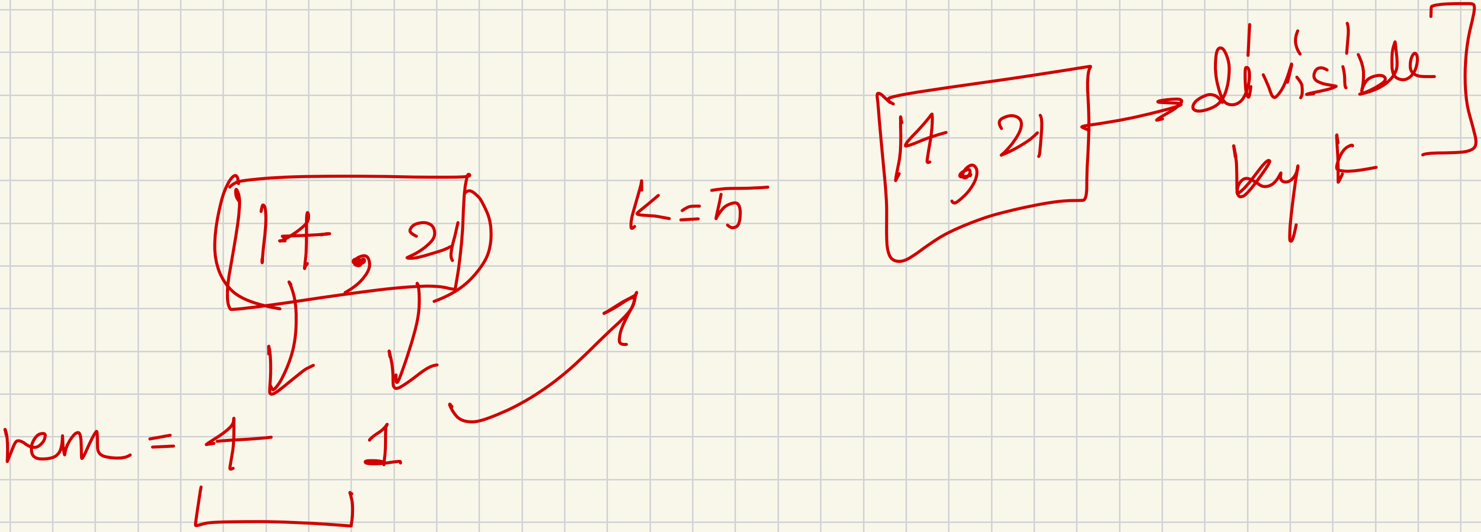
$$x = K * d + r$$

to make two numbers (x, y) divisible by k

$$(r_1 + r_2) = K$$

or

$$(r_1 + r_2) = 0$$



$\boxed{14, 21}$

rem = 4   1

K=5

$\boxed{14, 21}$ → divisible by k

$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ $K = 5$

	1	2	3	4	5	6	7	8	9	10
rem	1	2	3	4	0	1	2	3	4	0

$r_1 + r_2 = K$

$1 + r_2 = 5$

$r_2 = 4$

$2 + r_2 = 5$

$r_2 = 3$

{ you need equal freq of complementry rem.

fmap of rem

1 →	2
2 →	2
3 →	2
4 →	2
0 →	2

Largest Subarray with sum equal to zero

arr[] = {15 , -2 , 2 , -8 , 1 , 7 , -11 , -4}

Brute Force

→ Calc. all subarray sum

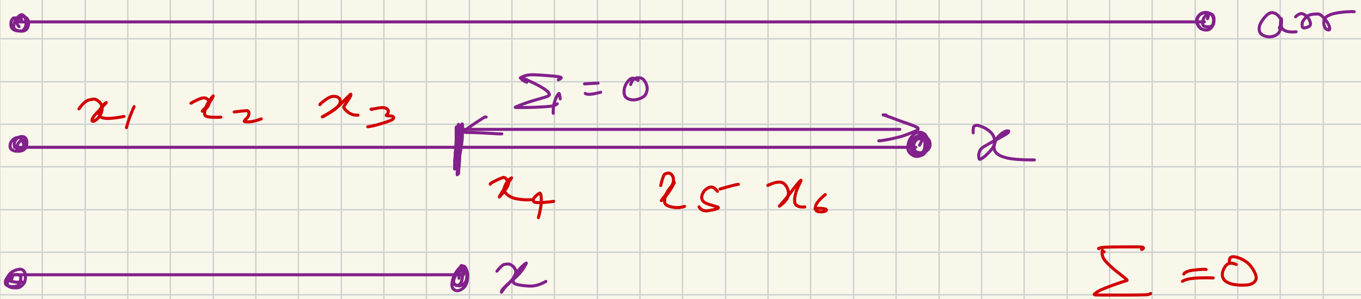
if (sum == 0)

Store maxLen

TC: $O(N^2)$

SC: $O(1)$

$$\begin{array}{lcccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ arr[\,] = \{ & 15, & -2, & 2, & -8, & 1, & 7, & -11, & -4 \} \end{array}$$

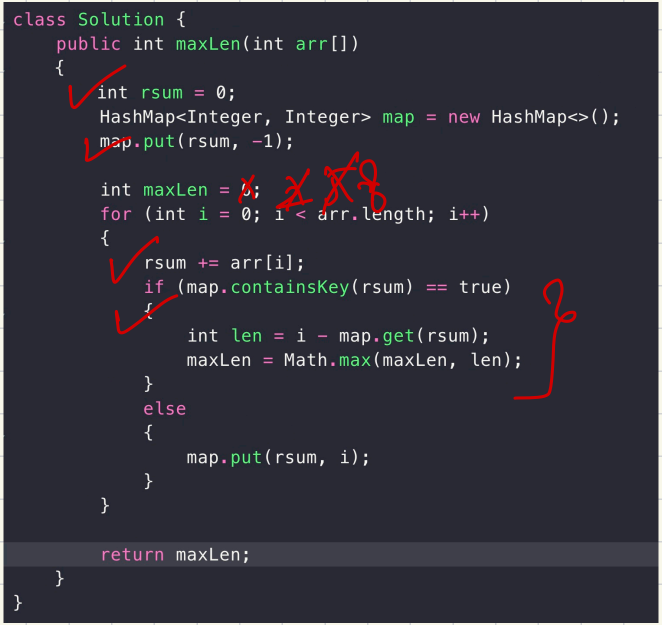


$$\underbrace{\boxed{x_1 + x_2 + x_3}}_{x_1 + x_2 + x_3 = x} + \overbrace{\boxed{x_4 + x_5 + x_6}}^{\Sigma = 0} = x$$

	-1	0	1	2	3	4	5	6	7
arr[] =		{15,	-2,	2,	-8,	1,	7,	-11,	-4}
sum of arr	0	15	13	15	7	8	15	4	0

len = ~~0~~ ~~2~~ ~~5~~ 8

len of Subarray = i − {index sum was first seen}



```
class Solution {
    public int maxLen(int arr[])
    {
        int rsum = 0;
        HashMap<Integer, Integer> map = new HashMap<>();
        map.put(rsum, -1);

        int maxLen = 0;
        for (int i = 0; i < arr.length; i++)
        {
            rsum += arr[i];
            if (map.containsKey(rsum) == true)
            {
                int len = i - map.get(rsum);
                maxLen = Math.max(maxLen, len);
            }
            else
            {
                map.put(rsum, i);
            }
        }

        return maxLen;
    }
}
```

~~0~~ ~~2~~ ~~5~~ 8

0	1	2	3	4	5	6	7
15	-2	2	7	1	-8	-11	-4

rsum	indexSeenFirst
0	-1
15	0
13	1
22	3
23	4
4	6

rsum = ~~0~~ ~~15~~ ~~13~~ ~~15~~ ~~22~~ ~~23~~ ~~15~~ ~~4~~ 0

2 - 0 = 2

5 - 0 = 5

7 - (-1) 8

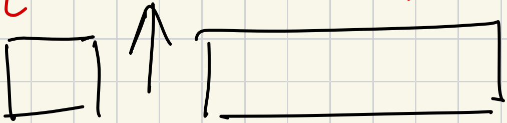
Equilibrium Index

```
             0   1   2   3   4   5
int[] arr = {9 , 3 , 7 , 6 , 9 , 10}
```

equilibrium index, is an index where sum of all the ele. on left- left of that index is equal sum of all the ele on right of that index.

$$\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array}$$

int[] arr = {9, 3, 7, 6, 9, 10}



stand at each index (i)

- get sum of left array $[0, i-1]$
- get sum of right array $[i+1, n)$

TC: $O(N * N) = O(N^2)$

SC: $O(1)$

```
               0   1   2   3   4   5
int[] arr  = { 9,  3,  7,  6,  9,  10 }

int[] lsum = { 0,  9,  12, 19, 25, 34 }

int[] rsum = { 35, 32, 25, 19, 10, 0 }
```

TC : O(N)
SC : O(N)

✓ Yes Same

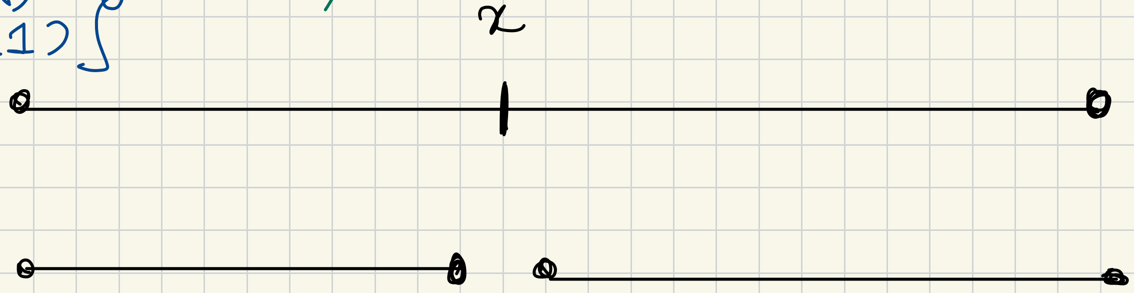
✓ Equilibrium Point

int[] arr = $\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 9, & 3, & 7, & \boxed{6}, & 9, & 10 \end{array}$

✓ totalSum = 44

✓ lsum = ~~0~~ ~~9~~ ~~12~~ 19

TC: O(N)
SC: O(1)



rsum = 44 − 9 − 0
= 35

rsum = 44 − 3 − 9
= 32

rsum = 44 − 7 − 12
= 25

rsum = 44 − 6 − 19
= 19

lsum + x + rsum = totalSumArray

rsum = totalSumArray − x − lSum

H.W.

- Count No. of Pairs with Absolute Diff. K
- Find the repeating and Missing Number.

Valid Anagram.

str1 str2

accio ociac

o → 1
c → 2
i → 1
a → 1

Brute force

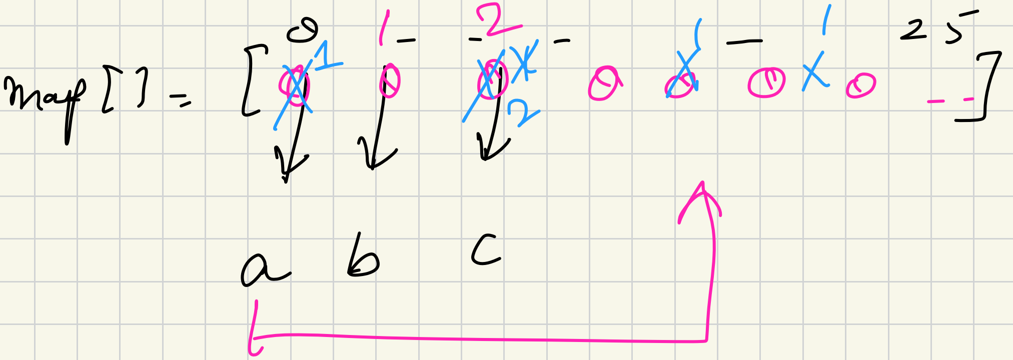
sort string: accio accio

TC: O(Nlogn)
SC: O(1)

fmap

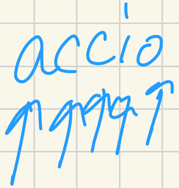
a → 1
c → 2
i → 1
o → 1

TC: O(N) + O(N) + O(26) = O(N)
SC: O(52) = O(1)



'm' − 'a' =

'c' − 'a' = 2



'a' − 'a' = 0

'c' − 'a' = 2