a b c d e f g h

0 1 2 3 4 5 6 7

a b c d e f g h

0 1 2 3 4 5 6 7

a b c d e f g h

mes < 6

Anagrams

cat , act → ✓

catt , acct → X

[act, god, cat, dog, tac]

[[act, cat, tac]
[god, dog]]

M1

key → [strings]

something common for all anagrams

anagrams.

hash map

M1 → [act, cat, tac]

M2 → [god, dog]

ac , bb
0 2 , 1 1

act ACT

if keys are sorted versions of the strings

TC = $n^2(m + m\log m) = O(nm\log m)$.

→ act

0 1 2 . . . . . 20 . . . 25
a c t a
1 1

→ TC: O(nm).

101000 . . . 100 . . .

a b
1 # 11 #

a b
11 # 1 #

TC = O(n m)

AS = ____

[abc, def, ghi]

M1 → [abc]
M2 → [def]
M3 → [ghi]

[abc, bca, cab]

M1 → [abc, bca, cab]

M1 → [0, 1, 2]

M1 → [0]
M2 → [1]
M3 → [2]

---

0 1 2 3 4 5 6 7

[1, 2, 7, -1, -8, 6, 3, 5]

0 1 3 10 9 1 7 10 15

(1, 4)

(i, j)

0 → -1
1 → 0
3 → 1
10 → 2
9 → 3

sum → find Or.

Recursion

fun(1) → Stop → stopping conditions are called Base Cases

fun(2)
fun(3)
fun(4)
fun(5)

→ call stack of function.

int fun (int n) {
if (n ≤ 1) return; } → base case.
cout << n << endl;
return fun (n-1);
}

int fun (int n) {
if (n ≤ 1) return; } → base case.
cout << n << endl;
fun (n+1);
}

Infinite Recursion.
Beware!!

Print (n-1)

```
void printNumbersNTo1(int n) {
    if (n <= 0) {
        return;
    }
    cout << n << endl;
    printNumbersNTo1(n - 1);
}
```

n=5

n=0
n=1
n=2
n=3
n=4
n=5

n=1
5
4
3
2
1

```
void printNumbers1ToN(int n) {
    if (n <= 0) {
        return;
    }
    printNumbers1ToN(n - 1);
    cout << n << endl;
}
```

n=0
n=1
n=2
n=3
n=4
n=5

n=1
1
2
3
4
5

Tower of Hanoi

from aux to

from aux to

1
2
to

||

1
2
3
from

1
2
3
Aux

1
2
3
To

A B C

(n, A, C, B) ⇒ (n-1, A, B, C)
move the nth disk from A to C
(n-1, B, C, A)

n-1
n
A B C

n-1
n
A B C

n-1
n
A B C