# Search in Matrix

| | | | | |
|---|---|---|---|---|
| 5 | 3 | 4 | 8 | 7 |
| 1 | 9 | 2 | 6 | 10 |
| 12 | 11 | 14 | 15 | 13 |
| 20 | 19 | 17 | 18 | 16 |
| 22 | 23 | 21 | 25 | 24 |

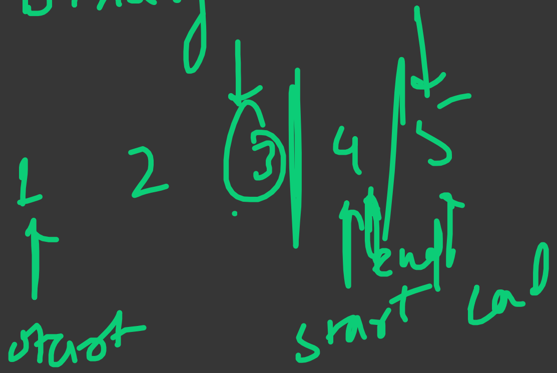
target = 14

Linear search

1 D → array

Binary Search

O(log N)



target = 4

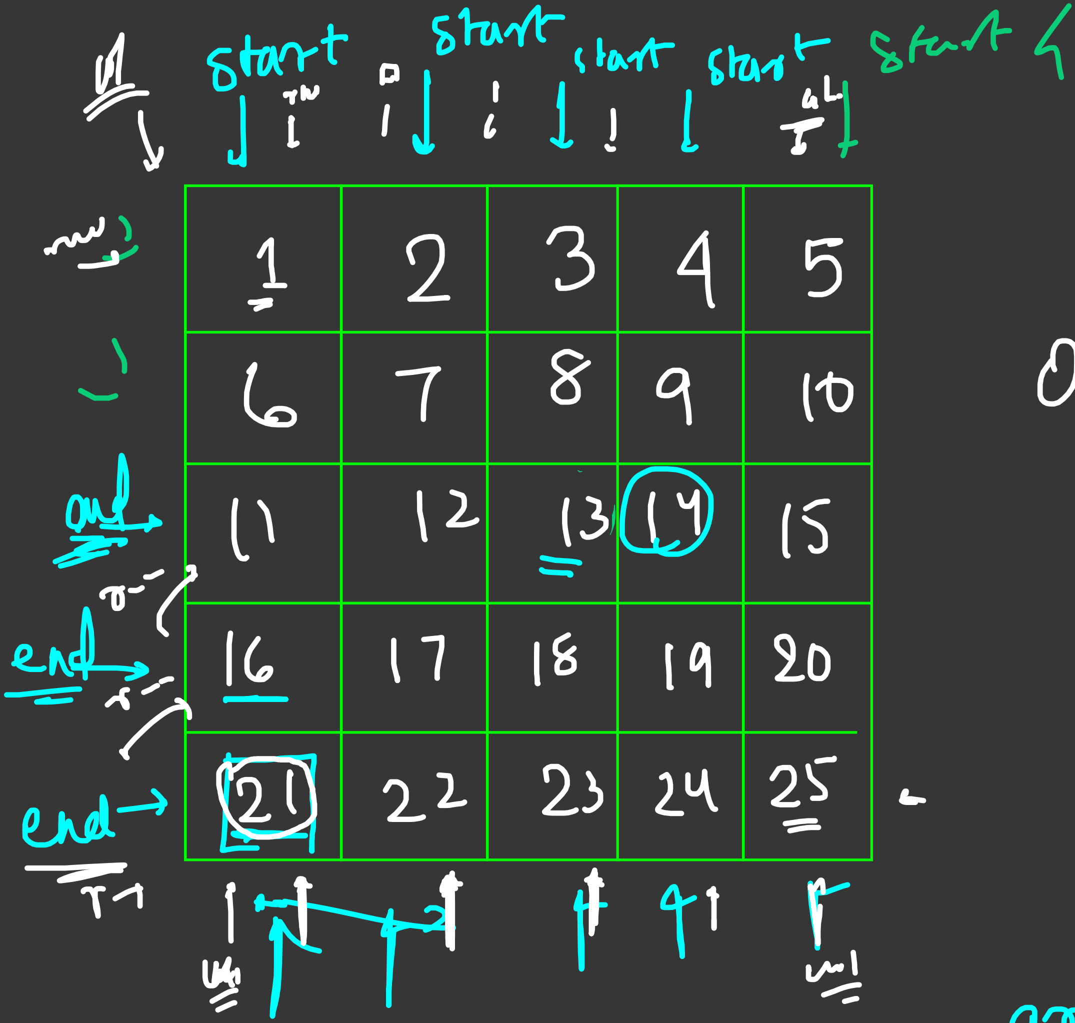
arr[mid] = = target , target

return true

|  |  |  |  |  |
|---|---|---|---|---|
| 5 | 3 | 4 | 8 | 7 |
| 1 | 9 | 2 | 6 | 10 |
| 12 | 11 | 14 | 15 | 13 |
| 20 | 19 | 17 | 18 | 16 |
| 22 | 23 | 21 | 25 | 24 |

→

|  |  |  |  |  |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 |



| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 |

next in Col > 21

row → sorted

col → sorted

O(N)

O(~~Nlog~~(M×N)) lg(m×n)) + O(N)

start = 0 (col)

end = n-1 (row-1)

target = 14

arr[start][end] > K {

end--;

arr[start][end] < K

## Pseudo code:

```
Sort the matrix    ( O(N log N)

    int r = n-1,    c=0,
while( start <n && end>=0)
        if (arr[r][c] == k)
                return true,
        else if ( arr[r][c] > k)
                r--,
        else
                c--,
```

# Optimised Binary Search :

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 1 | 6 | 7 | 8 | 9 | 10 |
| 2 | 11 | 12 | 13 | 14 | 15 |
| 3 | 16 | 17 | 18 | 19 | 20 |
| 4 | 21 | 22 | 23 | 24 | 25 |

(2,1)

→ Flatten the array in 1D- array hypothetically

Size of 1D array = $n*m$

0 ⟶

s 25 size

mid

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |

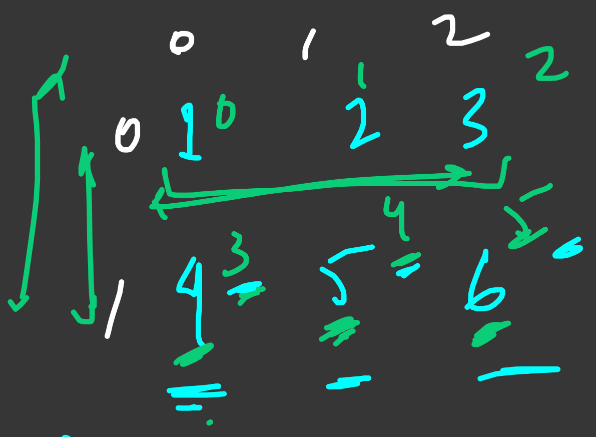
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

e

start = 0

end = $(n \times m) - 1$

mid = $(25 + 0)/2 = 12$



n = 2

m = 3

n × m = ~~8~~ 6

0

3

6

9

```
col = ind % m ;

row = ind / m;
```

4 → array

matrix

col

~~row~~

first index complete divisible by colsize

Binary

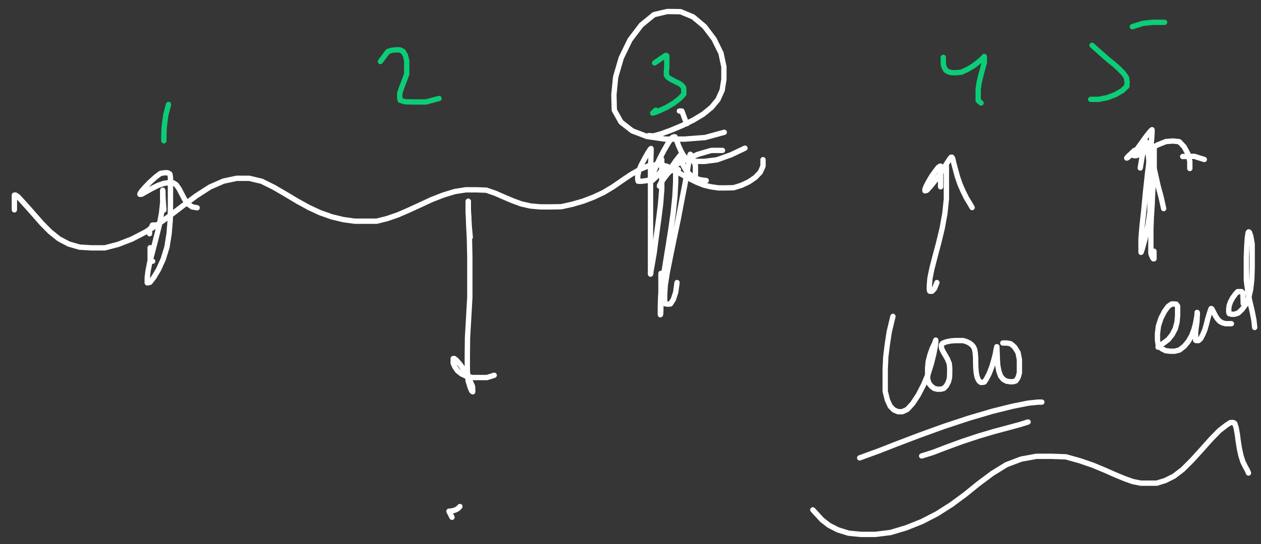
low mid high

$\frac{5+0}{2}$ 2 2

0 1 2 3 4

4

1 2 3 4 5



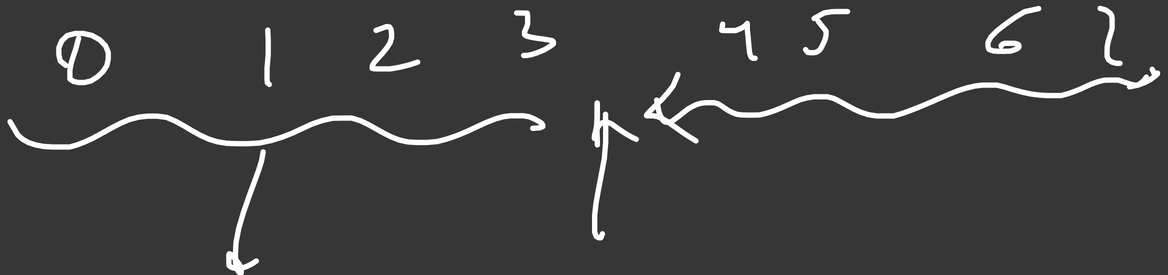
(log n)

matrix

T.C $= \log_2 (n \times m)$

S.C $= O(1)$

# Binary Search Time Complexity analysis



$O(\log(n))$