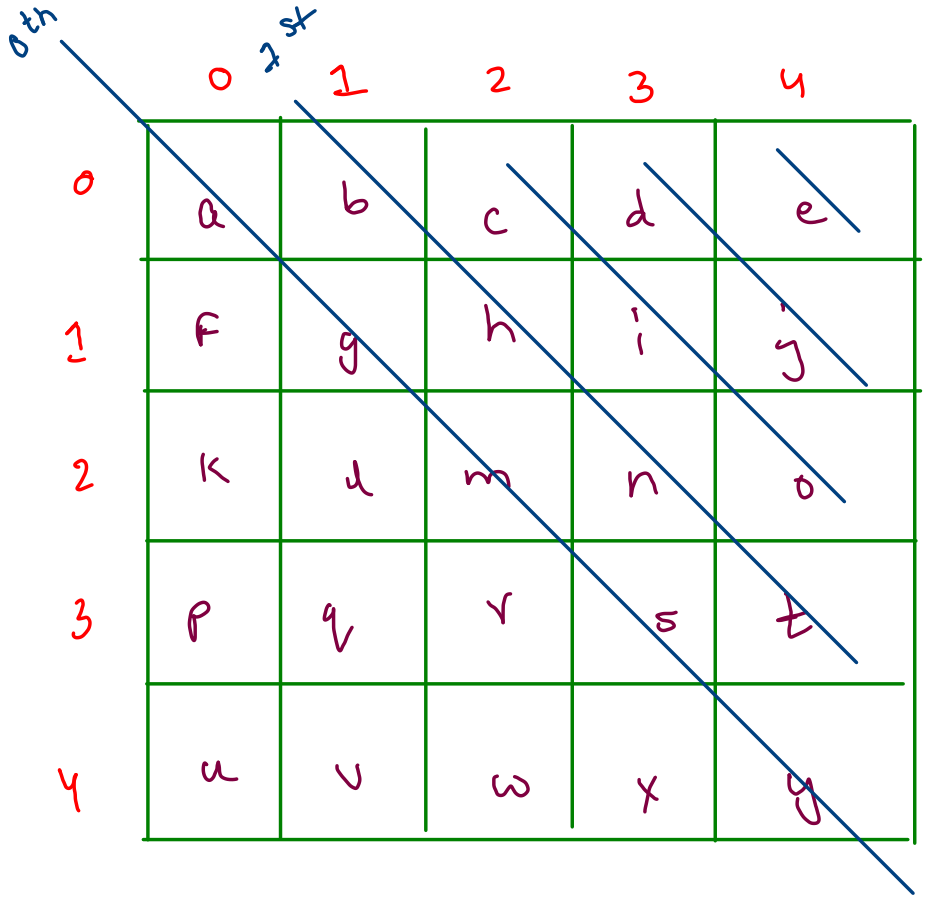
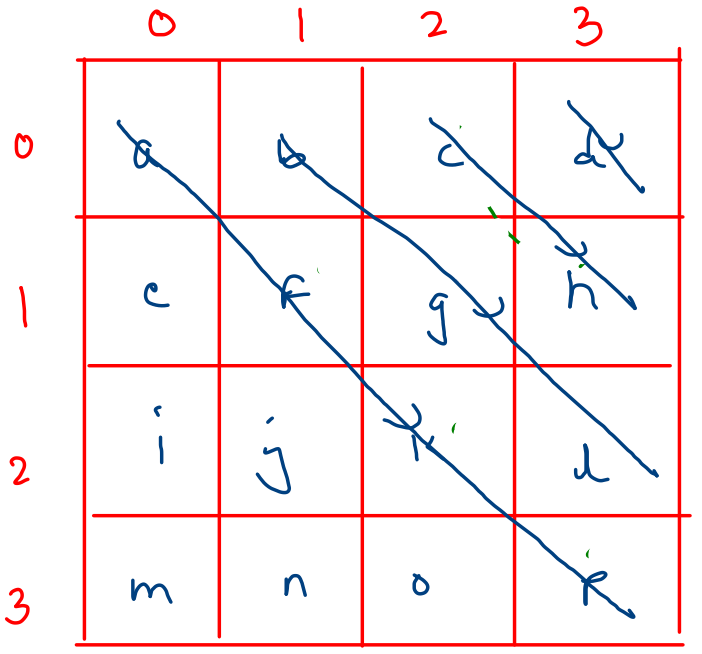


dth → (0, d)

[diagonal select
[r = 0, c = d
[

diagonal	row	cols
0	0	0
1	0	1
2	0	2
3	0	3

```java
public static void printDiagonally(int[][]mat) {
    int n = mat.length;

    //to select diagonal
    for(int d=0 ; d < n; d++) {
        //print dth diagonal
        for(int r=0,c=d; c < n;r++,c++) {
            System.out.println(mat[r][c]);
        }
    }
}
```

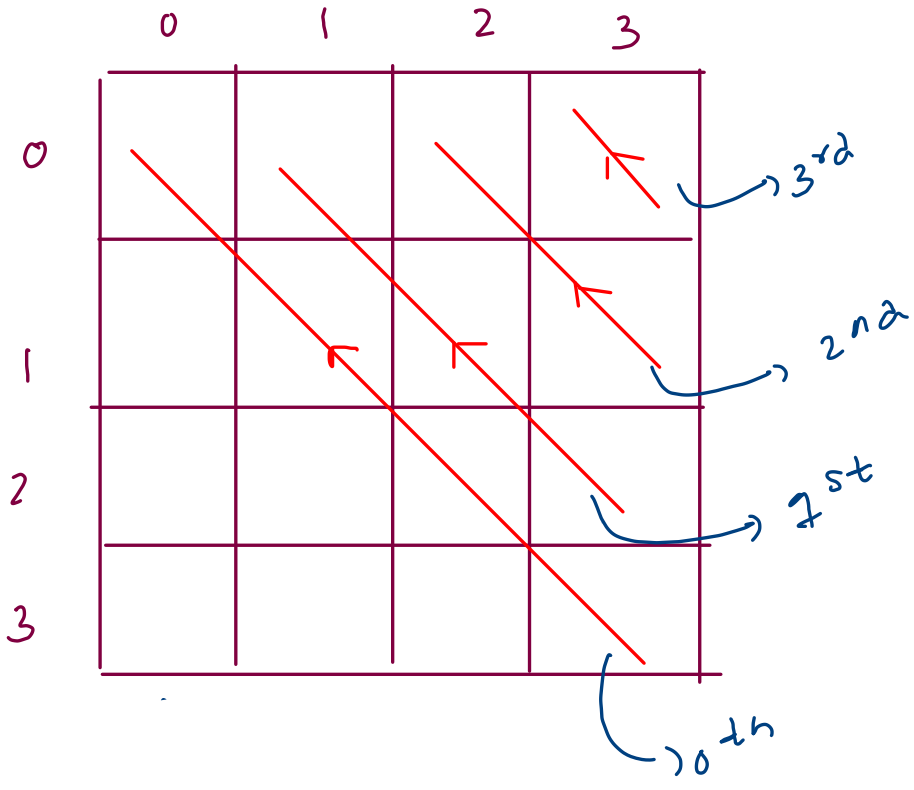


d -> 3

r = ~~0~~ 1

c = ~~3~~ 4

a f k p b g l c h d



d → (n-1-d, n-1)

= 1, 3

d	r	c
0	3	3
1	2	3
2	1	3
3	0	3

n-1 = 3

```
for( int d = 0; d < n ; d++) {
    if( d -> even) {
        [ r = 0, c = d , r++, c++
        [ ↓
    }
    else {
        [ r = n-d-1, c = n-1, r--, c--
        [ ↑
    }
}
```

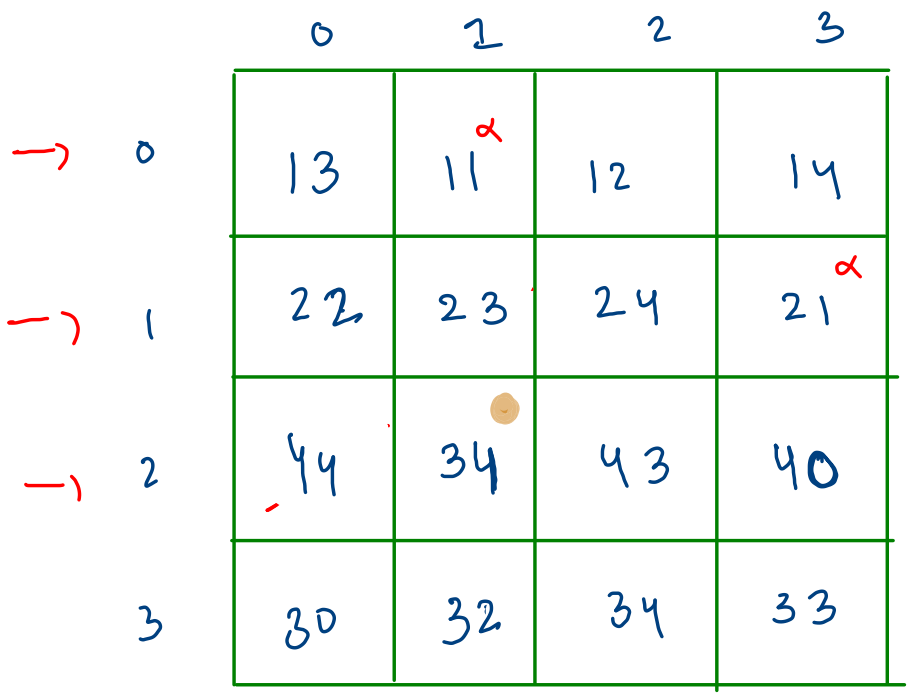
Saddle price

	0	1	2	3	
→ 0	13	11 α	12	14	min = 11 minc = 1
→ 1	22	21 α	24	23	min = 21 minc = 1
→ 2	44	42	43	31 α	min = 31 minc = 3
→ 3	41	32 α	34	33	min = 32 minc = 1

row → min

col → max

no saddle point



min = 11

minc = 1

min = 21

minc = 3

min = 34

minc = 1

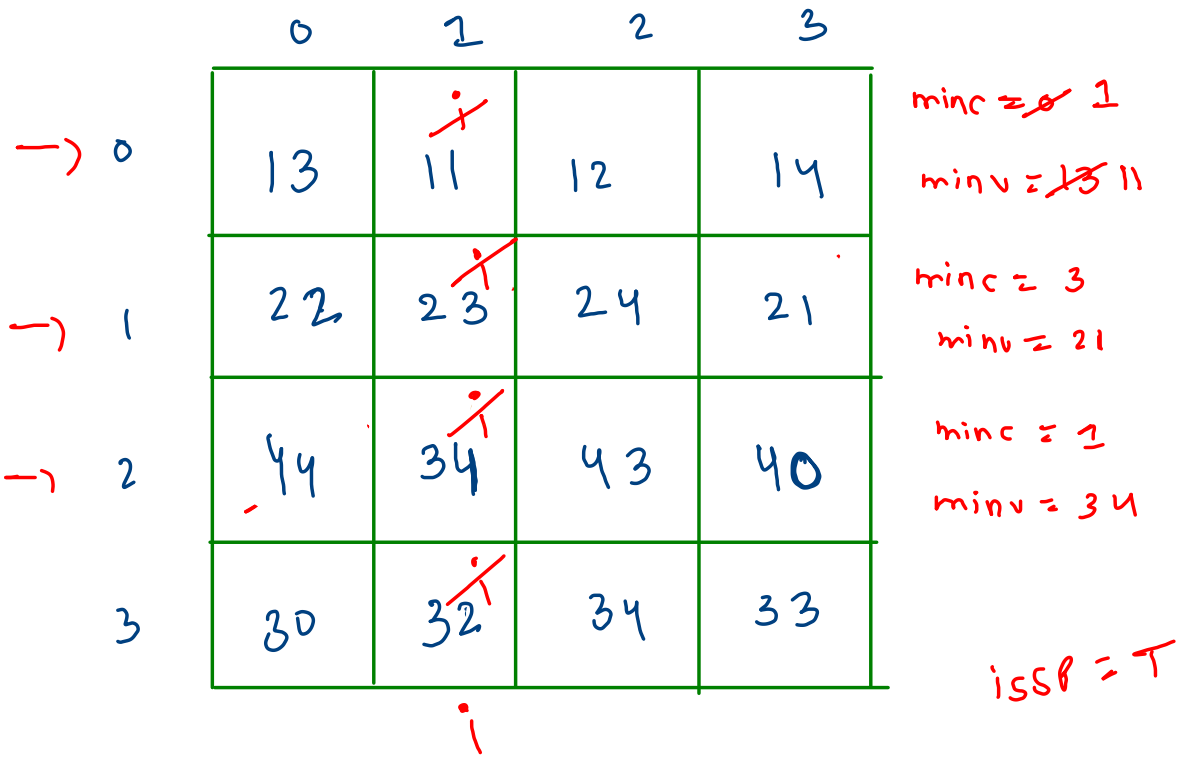
34

max = -∞

= Integer.MIN_VALUE

min = ∞

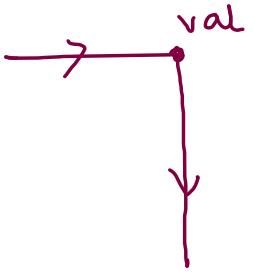
= Integer.MAX_VALUE




```java
for(int r = 0; r < n;r++) {

    //find min element and its col in rth row
    int minc = 0;
    int minv = mat[r][0];

    for(int c = 1; c < n;c++) {
        if(mat[r][c] < minv) {
            minv = mat[r][c];
            minc = c;
        }
    }


    //minv is the maximum value in minc
    boolean isSP = true;
    for(int i=0; i < n;i++) {
        if(mat[i][minc] > minv) {
            isSP = false;
            break;
        }
    }

    if(isSP == true) {
        return minv;
    }

}
```

target = 13

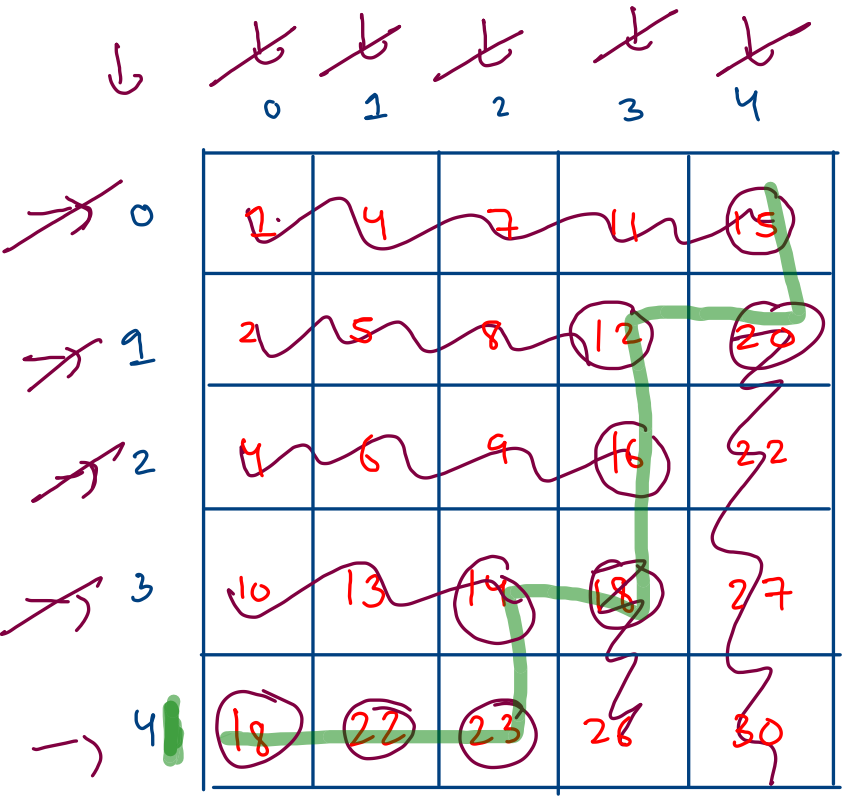
	0	1	2	3	4
0	1	4	7	11	15
1	2	5	8	12	20
2	4	6	9	16	22
3	10	13	14	17	27
4	18	22	23	26	30



val < target → r++

val > target → c--

val == target



val < target → r++

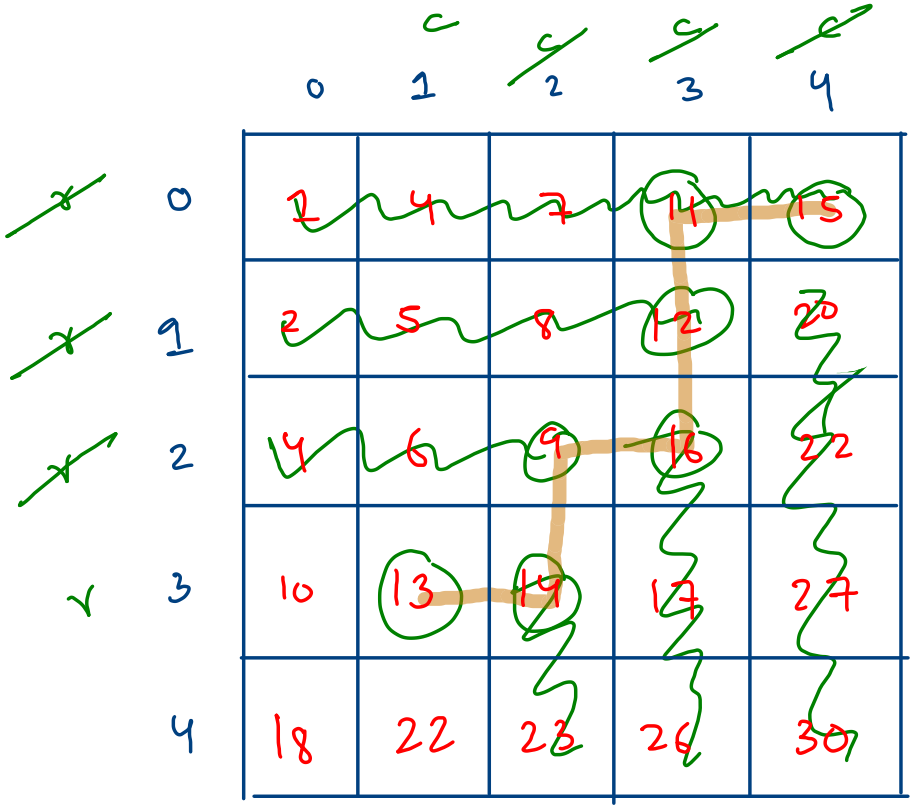
val > target → c--

val == target

r = 0 1 2 3 4

target = 17

c = 4 3 2 1 0 -1



target = 13

```java
//top right corner
int r = 0;
int c = mat[0].length - 1;

while(r < mat.length && c >= 0) {
    if(mat[r][c] == target) {
        System.out.println(r + "\n" + c);
        return;
    }
    else if(mat[r][c] < target) {
        r++;
    }
    else {
        c--;
    }
}

System.out.println("Not Found");
```

3 1

r = 0 1 2 3

c = 4 3 2 1

target = 63

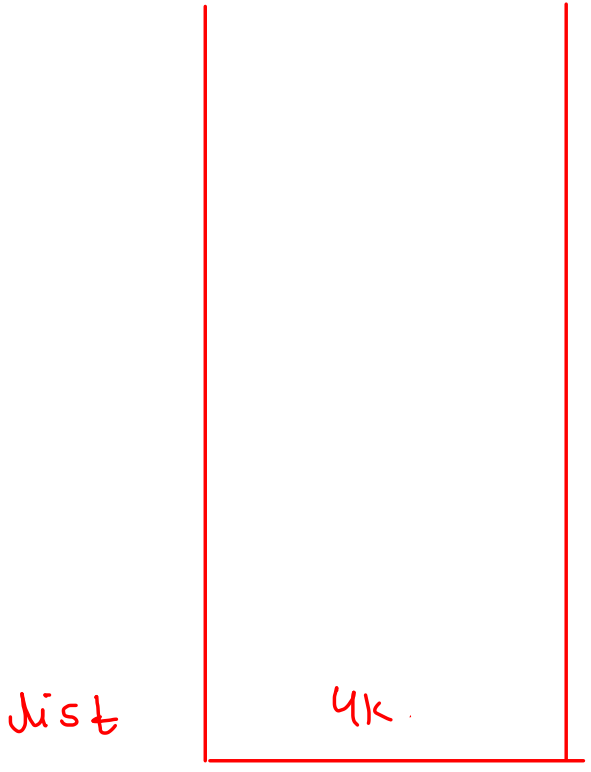
1. to find potential row
2. to find target in potential row.

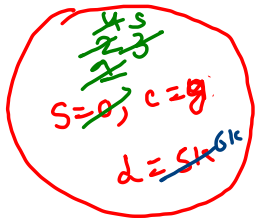
$\log(m) + \log(n)$

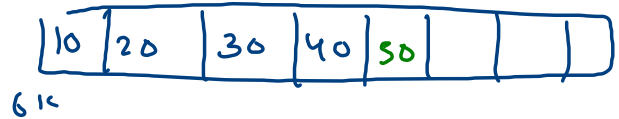
$\log(m \times n)$

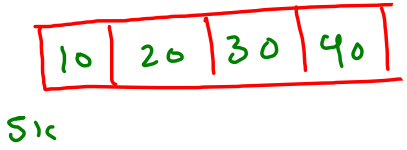
	0	1	2	3
0	1	3	5	7
1	10	11	16	20
2	23	30	34	60
3	62 (lo)	63	65	68 (hi)
4	71	72	74	80

ArrayList <Integer> list = new ArrayList<>();










```
list.add(10);
list.add(20);
list.add(30);
list.add(40);
list.add(50);
```