Search 1

1. Write an efficient algorithm that searches for a value in a m x n matrix. This matrix has the following properties:
a). Integers in each row are sorted from left to right.
b). The first integer of each row is greater than the last integer of the previous row.

	0	1	2	3	4
0	10	19	25	38	40
1	42	49	53	56	58
2	62	64	68	72	76
3	79	82	88	92	100
4	102	110	150	190	200

pot. row

key = 70

(i) to find potential row.

(ii) to find element in this potential row.

(i) row wise sorted

(ii) col wise sorted

(iii) prev row's last < next row's start

linear search → n

binary search → $\log_2 n$

$n, \frac{n}{2}, \frac{n}{4}, \ldots, 1$

$a = n$

$cr = \frac{1}{2}$

$ar^{k-1} = 1$

$n * \left(\frac{1}{2}\right)^{k-1} = 1$

$n = (2)^{k-1}$

$\log_2 n = (k-1)$

$k = \log_2 n + 1$

lci = 4

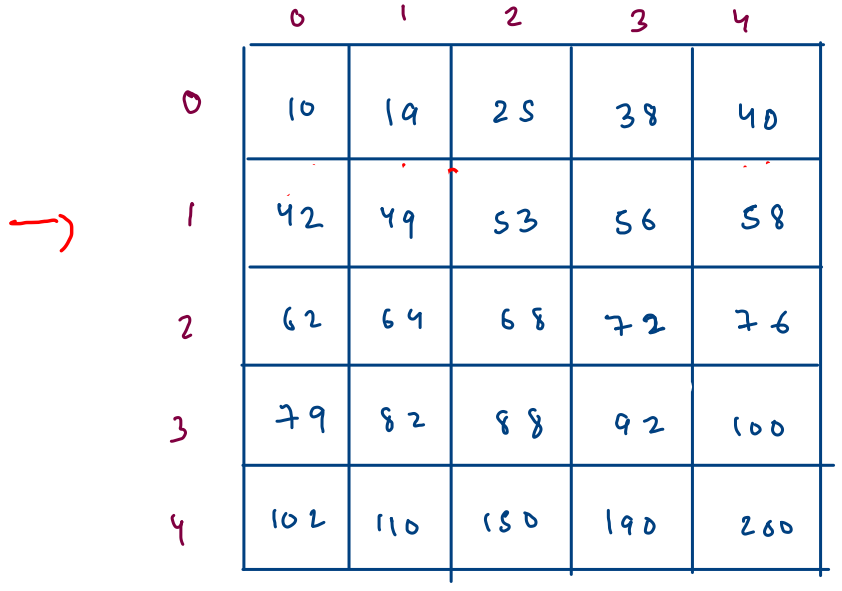
	0	1	2	3	4
0	10	19	25	38	40
1	42	49	53	56	58
2	62	64	68	72	76
3	79	82	88	92	100
4	102	110	150	190	200

m hi

lo 5

```java
public static int potential_row(int[][]mat,int target) {
    int lo = 0;
    int hi = mat.length-1;
    int lci = mat[0].length-1; //last column index

    while(lo <= hi) {
        int mid = (lo + hi)/2;

        if(target >= mat[mid][0] && target <= mat[mid][lci])
            return mid;
        }
        else if(target < mat[mid][0]) {
            hi = mid-1;
        }
        else {
            lo = mid+1;
        }
    }
```

val = 210



43 , pr = 1

```java
public static boolean search(int[][]matrix,int target) {
    //write your code here

    //to find potential row for this target element
    int pr = potential_row(matrix,target);

    if(pr == -1) {
        return false;
    }

    //to find whether target is actually there in potential row
    boolean ans = isElementPresent(matrix,target,pr);

    return ans;
}
```

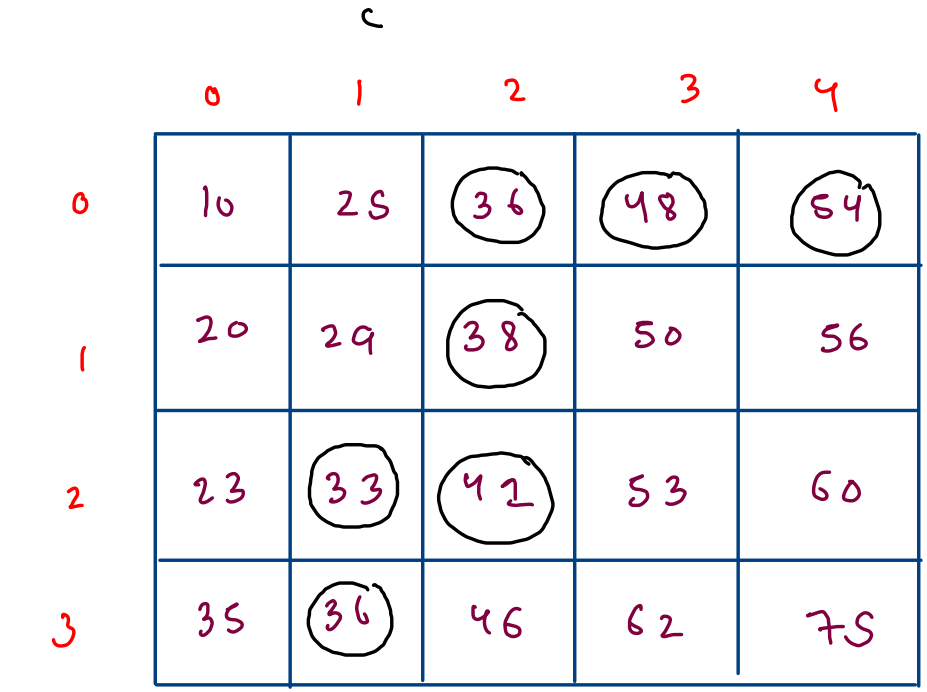
```java
public static int potential_row(int[][]mat,int target) {
    int lo = 0;
    int hi = mat.length-1;
    int lci = mat[0].length-1; //last column index

    while(lo <= hi) {
        int mid = (lo + hi)/2;

        if(target >= mat[mid][0] && target <= mat[mid][lci]) {
            return mid;
        }
        else if(target < mat[mid][0]) {
            hi = mid-1;
        }
        else {
            lo = mid+1;
        }
    }

    return -1;
}
public static boolean isElementPresent(int[][]mat,int target,int r) {
    int lo = 0;
    int hi = mat[0].length-1;

    while(lo <= hi) {
        int mid = (lo + hi)/2;

        if(mat[r][mid] == target) {
            return true;
        }
        else if(mat[r][mid] > target){
            hi = mid-1;
        }
        else {
            lo = mid+1;
        }
    }

    return false;
}
```

1. You are given a number n, representing the number of rows and columns of a square matrix.
2. You are given n * n numbers, representing elements of 2d array a.
Note - Each row and column is sorted in increasing order.
3. You are given a number x.
4. You are required to find x in the matrix and print it's location int (row, col) format as discussed in output format below.
5. In case element is not found, print "Not Found".

	0	1	2	3	4
0	10	25	36	48	64
1	20	29	38	50	56
2	23	33	41	53	60
3	35	36	46	62	75

(i) row wise sorted

(ii) col wise sorted



n -> rows
m -> cols

```
r = 0
c = mat[0].length-1;

while (r<n && c>=0){

    if(mat[r][c]==key){
        syso(r,c);
        return;
    }
    else if(mat[r][c]>key){
        c--;
    }
    else{
        r++;
    }
}
```

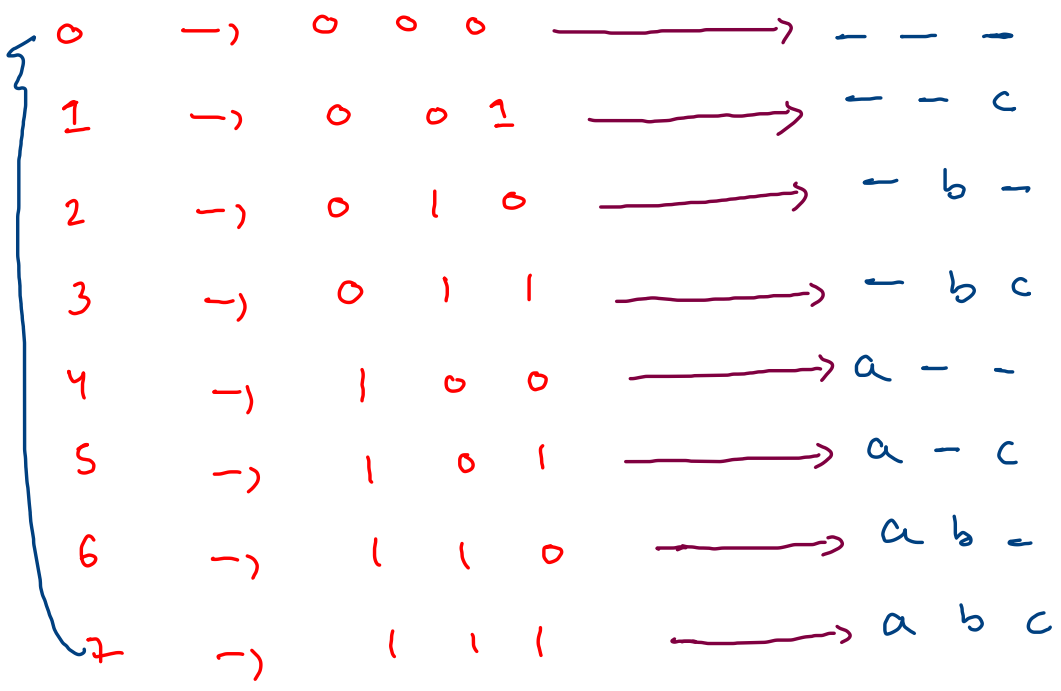
n+m

find = 40

arr : [a b c]

Subarray	subset
(i) a b c ab bc abc	- - - - - c - b - - b c a - - a - c a b - a b c
(ii) $\frac{n(n+1)}{2}$	$2 \times 2 \times 2 = 2^n$
(iii) continous	(iii) not continous

arr: a, b, c



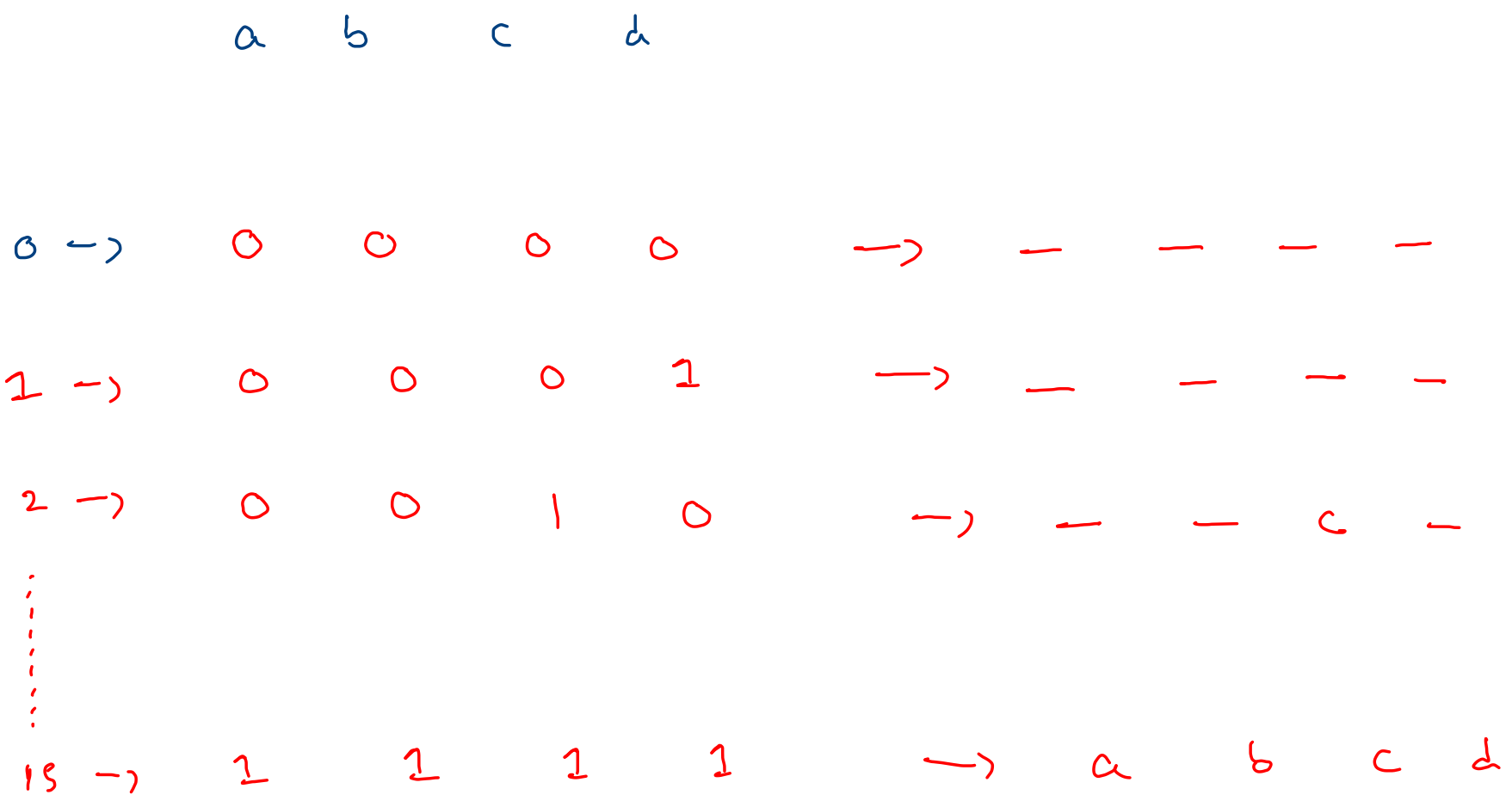
0 → no choice

1 → yes choice

n length binary no.

2^n

0 to $2^n - 1$


a b c d
0 -> 0 0 0 0 -> - - - -
1 -> 0 0 0 1 -> - - - -
2 -> 0 0 1 0 -> - - c -
15 -> 1 1 1 1 -> a b c d

```java
public static void print_subsets(int[]arr) {
    int n = arr.length;
    int ts = (int)Math.pow(2,n); //total subsets

    for(int k = 0; k < ts;k++) {
        //we want k's binary represation of length 'n'
        int[]bin = decToBin(k,n);

        //print subset
        for(int i=0; i < n;i++) {
            if(bin[i] == 0) {
                //ith element is excluded from this subset
                System.out.print("-\t");
            }
            else {
                //ith element is included in this subset
                System.out.print(arr[i]+ "\t");
            }
        }

        System.out.println();

    }
}

public static int[] decToBin(int dec,int len) {
    int[]arr = new int[len];
    int idx = len-1;

    while(dec > 0) {
        int d = dec % 2;
        dec = dec/2;

        arr[idx] = d;
        idx--;
    }

    return arr;
}
```

arr : 10 20 30
0 1 2

ts = 8 n = 3

0 → 0|0|0 — — —

1 → 0|0|1 — — 30

2 → 0|1|0 — 20 —

3 → 0|1|1 — 20 30

4 → 1|0|0 10 — —

5 → 1|0|1 10 — 30

6 → 1|1|0 10 20 —

7 → 1|1|1 10 20 30