

int h = 11

binary \rightarrow ³¹ 0000000: _ _ 000 ⁰ 1011

print \rightarrow 1011
13

rev = $\left(1101 \right)_2$
 \downarrow
13

1123

1

1 1 1 - - - -

1 0
•
T T T T
0

31

0000000: _ _ 000 10!!

0

rev →

3 2 1 0
1 1 0 1
0 1 0 0

bi

```
System.out.println();  
System.out.println(rev);
```

$y_{EV} = 80000 - \sim 0000$

1011
13

105

ar $[a, b, c, d]$

$$a \wedge b$$

$$a \wedge c$$

$$a \wedge d$$

$$b \wedge c$$

$$b \wedge d$$

$$c \wedge d$$

$$2 \quad 0, 5, 7$$

$$2 \wedge 0 = 2$$

$$2 \wedge 5 = 7$$

$$2 \wedge 7 = 5$$

$$0 \wedge 5 = 5$$

$$0 \wedge 7 = 7$$

$$5 \wedge 7 = 2$$

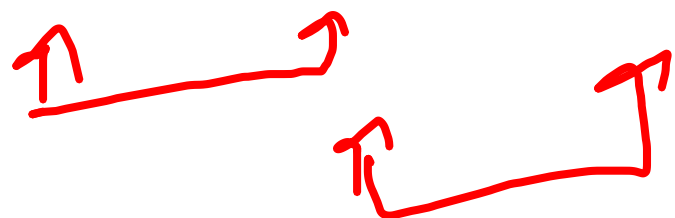
$$h^2 \downarrow O(h)$$

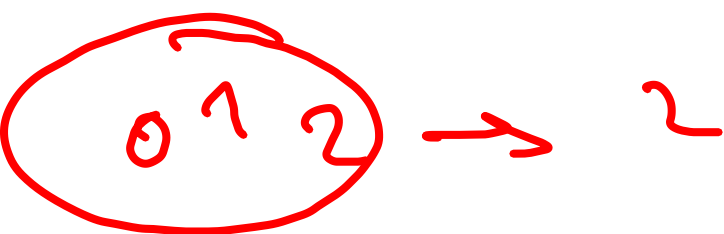
$$\begin{array}{r} 111 \\ 10 \\ \hline 5 \end{array}$$

$$2, 0$$

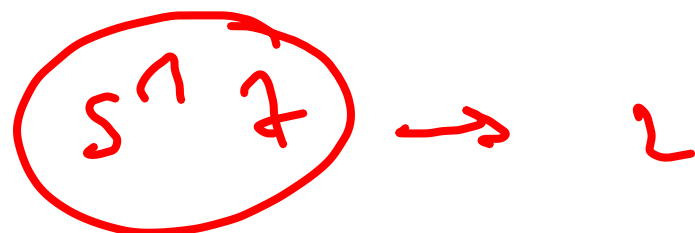
$$5, 7$$


 0 2 5 7

$a \leq b \leq c \leq d \leq e$



 $0 \wedge 2 \rightarrow 2$

$2 \wedge 5 \rightarrow 7$


 $5 \wedge 7 \rightarrow 2$

★

$a \leq b \leq c$

$a \wedge b$	$<$	$a \wedge c$
$b \wedge c$	$<$	$a \wedge c$

$$a < b < c$$

a 1 0 1 0 0 1 0 0 0 1 0 1 0
 b 1 0 1 0 0 1 1 - - - - -
 c 1 0 1 0 0 1 1 0 0 0 1 0 1

A red line connects the 1 in row b, column 7 to the 1 in row c, column 7. A red circle is drawn around the 1 in row b, column 7. A green circle is drawn around the 0 in row a, column 7.

```

Arrays.sort(arr);
int n = arr.length;

int min = Integer.MAX_VALUE;

for(int i=0;i<arr.length-1;i++){
    min = Math.min(min, arr[i]^arr[i+1]);
}

for(int i=0;i<arr.length-1;i++){
    if((arr[i]^arr[i+1]) == min){
        System.out.println(arr[i]+" "+arr[i+1]);
    }
}
  
```

b ^ c 0 0 0 0 0 0 0 - - - - -
 a ^ c 0 0 0 0 0 0 1 - - - - -

a ^ b 0 0 0 0 0 0 0 - - - - -
 a ^ c 0 0 0 0 0 0 1 - - - - -

The 1 in the second row is circled in red. The last four digits of both rows are circled in green.

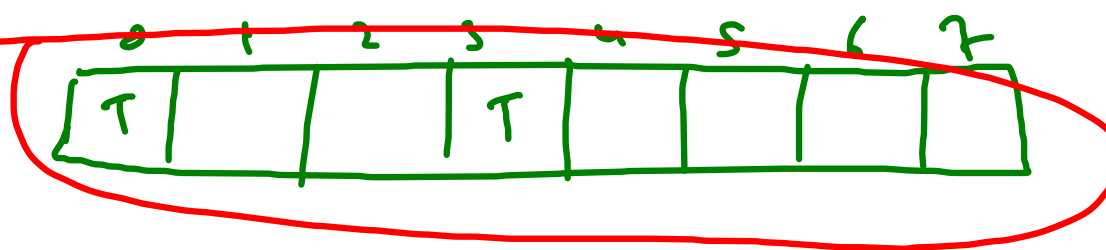
$n \times n$

n queries

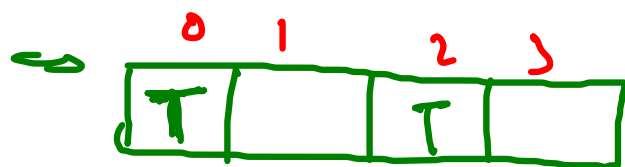
	0	1	2	3
0		Q		
1				Q
2	Q			
3			Q	

		Q	
Q			
			Q
	Q		

cols \rightarrow $000\dots 00000$
 \rightarrow 4 by 4



cols



Q			
		Q	
		•	

row
col

col

row-col + h

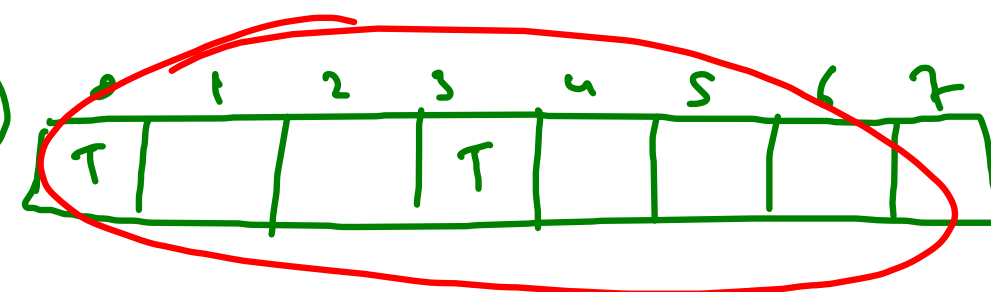
	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6

row + col

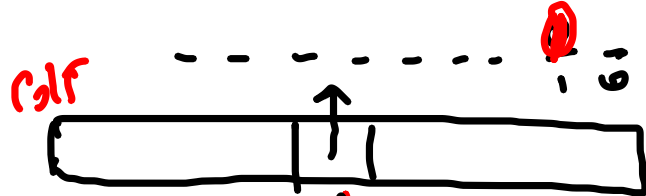
	0	1	2	3
0	34	45	6	26
1	23	34	5	8
2	21	23	34	45
3	10	1	23	34

col - row + h - 1

$h + h - 1$



1...9



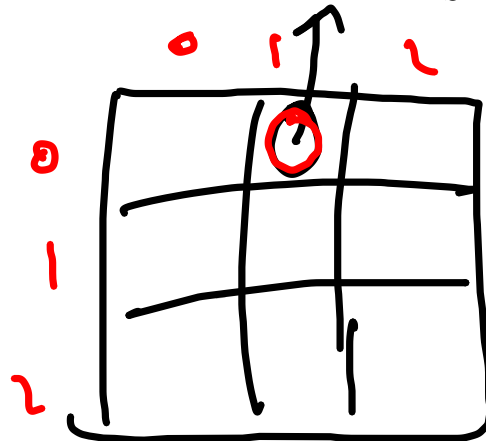
9 8 7 6 5 4 3 2 1 0

---1---10

rows

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

9 8 7 6 5 4 3 2 1



boxes

3	1	6	5	7	8	4	9	2
5	2	9	1	3	4	7	6	8
4	8	7	6	2	9	5	3	1
2	6	3	4	1	5	9	8	7
9	7	4	8	6	3	1	2	5
8	5	1	7	9	2	6	4	3
1	3	8	9	4	7	2	5	6
6	9	2	3	5	1	8	7	4
7	4	5	2	8	6	3	1	9

row/3
col/3

check exist

1	→	1	-	1
2	→	11	-	2
3	→	101		
4	→	111		
5	→	1001		
6	→	1111		
7	→	10001		
8	→	10101		
9	→	11011		
10	→	11111		

[3 1]
 [4]
 [5]
 [2 2]

11	→	1	0	0	0	1
12	→	1	0	1	1	0
13	→	1	1	0	0	1
14	→	1	1	1	1	1
15	→	1	0	0	0	0
16	→	1	0	0	1	0
17	→	1	0	1	0	1
18	→	1	0	1	1	0
19	→	1	1	0	0	1
20	→	1	1	0	1	1
21	→	1	1	1	1	1
22	→	1	1	1	1	1

[6 2]
 [7]

100110011001

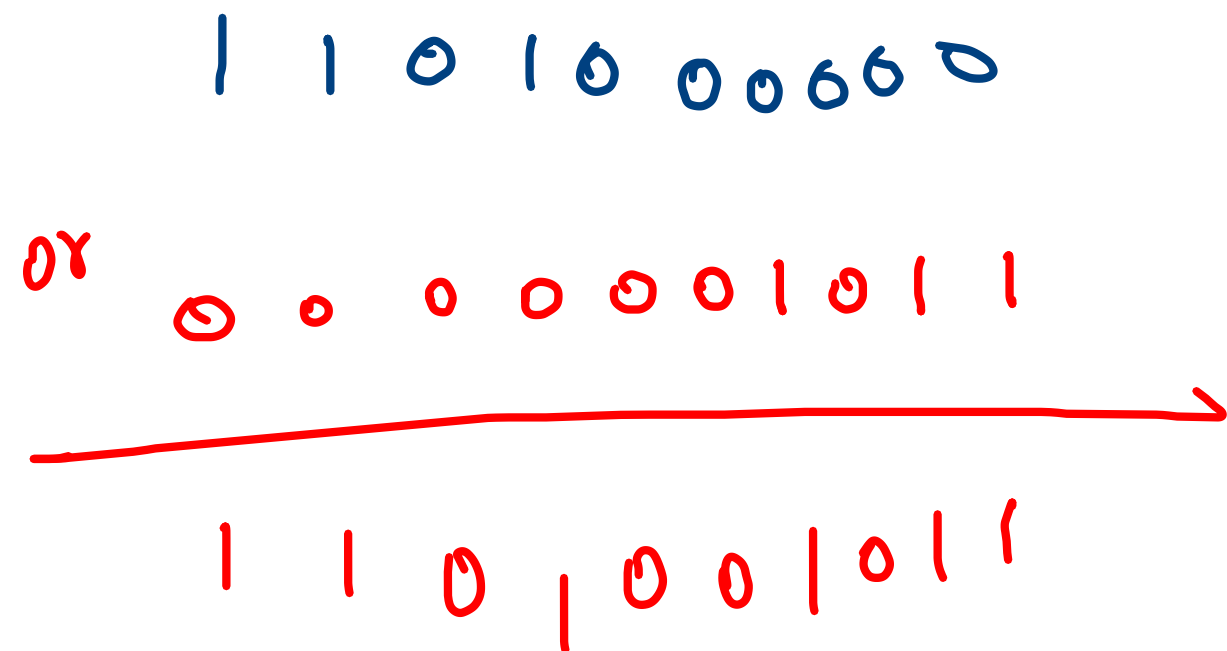
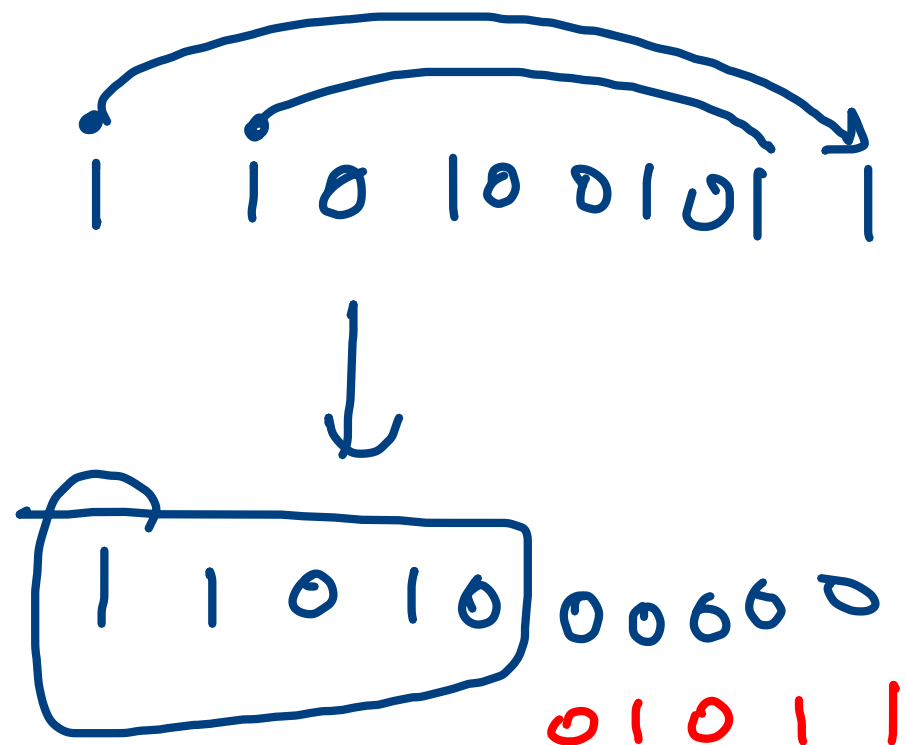
10011011001

4222

$$17-14 = 3-1 = 2$$

010223

010000



8 4 2 2 1 1 1
0 0 0 0 0 0 0

1 0 0 1 0 0 0 0

0 0 0 0 1 0 0 1

\rightarrow | 1001 00000
 ans = 1 0000 00000

also (1001)

```

len++; // 7
int offset = n-total-1; // 1 offset yyyy 1
int ans = (1<<(len-1)); // 1000000
ans = ans | (offset<<(len/2)); // 1 offset 000000
int rev = reverse(ans);
return ans | rev;
  
```

1001 << 5
 100100000

or
 110010000
 10011

 110010011

110010000
 000001001

 110010011