Lecture 7

Sunday, 19 January 2020 2:19 PM

Bit Masking, Manipulation.

1 -> 001
2 -> 010
3 -> 011
4 -> 100
5 -> 101
6 -> 110
7 -> 111
8 -> 1000

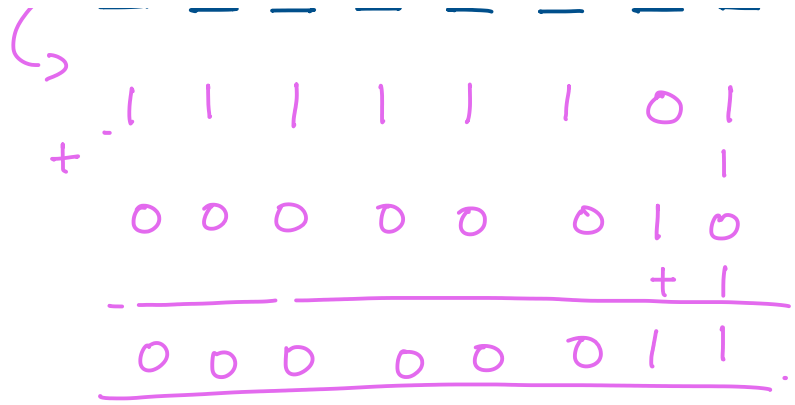
```
Syso(2^1) -> 3
Syso(2&1) -> 0
Syso(2|1) -> 3
Syso(~2)  -> -3
```

2 -> 010
1 -> 001
3 <- 011

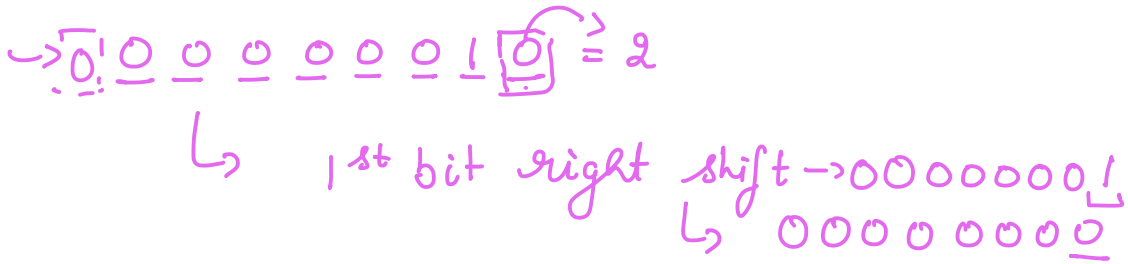
010
-> 101 5.

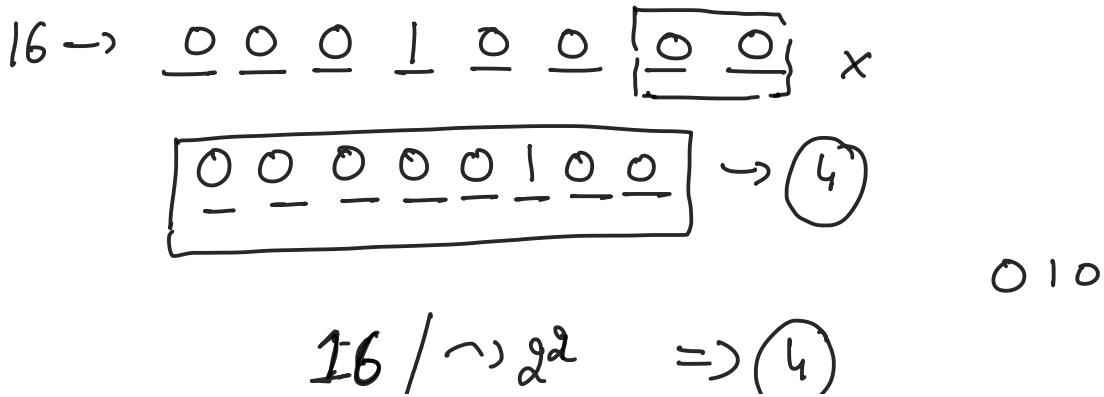
```
  010
  001
  ---
  000

  010
  001
  ---
  011
```

0 0 0 0 0 0 1 0



int num = 16;
num = num >> 2(bits).





~~17~~ 15

16 << 2 ⇒ 64

6 5 4 3 2 1 0
0 1 0 0 0 0 0 ↪ 64.

left shift -> multiply
right shift -> divide.

Syso(~(2^1)) ⟶ 011 / 100

```
public static int get_bit(int num, int i){   // num = 10, i = 4
    int mask = 1;                            // 0001
    mask = mask << (i-1);                    // 0010 -> 0100 -> 1000
    // int ans = mask & num;
    return (mask & num) == 0 ? 0 : 1;
}
```

1010 num
1000 mask

```
public static void set_bit(int num, int i){
    mask = 1
    mask = mask << i-1;

    num = num | mask;
    Syso(num);
}
```

4 3 2 1
1 0 1 0 num
0 1 0 0 mask
3 2 1 0
1 1 1 0
8 + 4 + 2 = 14

```
public static int right_most_set_bit(int num){
    int pos = 1;
    int mask = 1;
    while((num & mask)==0){
        mask = mask << 1;
        pos++;
    }
    return pos;
```

(2)

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

}

0 1 0 mask.
1 1 0
0 1 0

```
public static boolean powerOfTwo(int n){

    if((n & n-1)==0){
        return true
    }
    return false;
}
```

16 →
1 0 0 0 0
0 1 1 1 1
0 0 0 0 0

0 1 1 1 1
0 1 1 1 0
0 1 1 1 0

1 0 0 0 0

(i) Prob 1

int N = 9

arr = { 3, 4, 1, 5, 8, 9, 6 }

↳ N-2 length => 7.

↳ 2 and 7

↳ missing.

1 to 9 or N

a + b = 9.

[ab] = 14

$(a+b)^2 - 4ab = (a-b)^2$

Without Extra space → ✓

0 1 2 3 4 5 6

{ 3, 4, 1, 5, 8, 9, 6 }

```
int x = 0;
for( int i = 1 ; i<=n ; i++){
```

step 2

step 1

```
z = z ^ i;
```

```
for(int i=0; i<arr.length; i++){
    z = z ^ arr[i];
}
```

z = 1 ^ 2 ^ 3 ^ 4 ^ 5 ^ 6 ^ 7 ^ 8 ^ 9

z = 3 ^ 4 ^ 1 ^ 5 ^ 8 ^ 9

=> z = 2 ^ 7 =>

```
  0 0 1 0
^ 0 1 1 1
-------
  0 1 0 1
```

=> (5) in dec

z -> 0 1 0 1

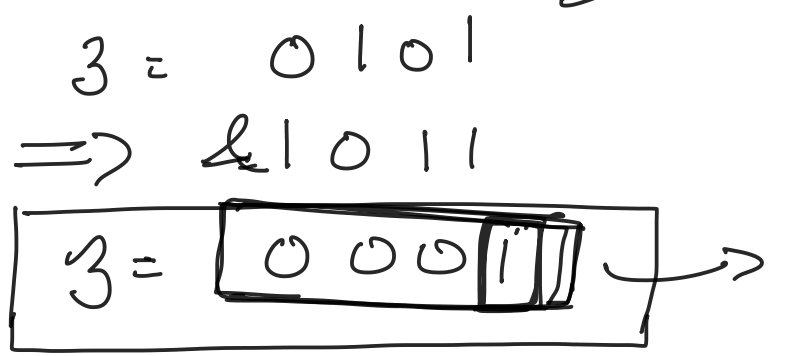
1 -> 0 0 0 1
2 -> 0 0 1 0
3 -> 0 0 1 1
4 -> 0 1 0 0
5 -> 0 1 0 1
6 -> 0 1 1 0
7 -> 0 1 1 1
8 -> 1 0 0 0
9 -> 1 0 0 1

(3)rd step -> find right most set b
but we can find any.

z = z & ~(z-1)

z -> 0 1 0 1

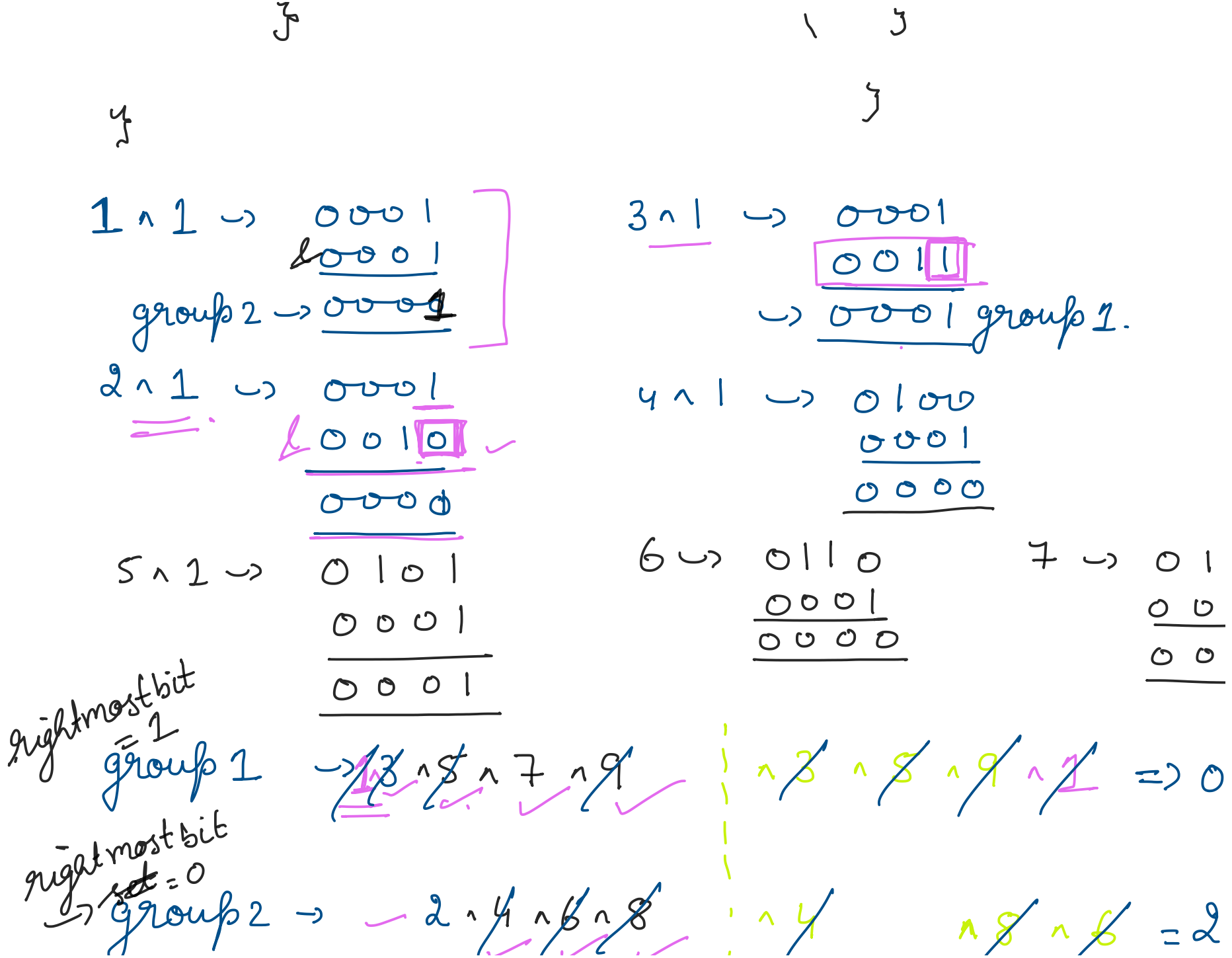
-> 0 1 0
iska nega
-> 1 0 1



4th step.

```
group1 = 0
for (int i = 1; i <= n; i++) {
    if ((i & 3) == 3) {
        group1 = group1 ^ i;
    } else {
        group2 = group2 ^ i;
    }
```

```
group2 = 0;
for (int i = 0; i < arr.length
    if ((arr[i] & 3) == 3)
        group1 = group1
    } else {
        group2 = group2 ^
```


}
}
1 ∧ 1 → 0001
0001
group 2 → 0000
2 ∧ 1 → 0001
0010
0000
5 ∧ 1 → 0101
0001
0001
rightmostbit = 1
group 1 → 1 ∧ 3 ∧ 5 ∧ 7 ∧ 9
rightmostbit = 0
group 2 → 2 ∧ 4 ∧ 6 ∧ 8
}
3 ∧ 1 → 0001
0011
→ 0001 group 1.
4 ∧ 1 → 0100
0001
0000
6 → 0110
0001
0000
7 → 0111
0001
0001
∧ 3 ∧ 5 ∧ 9 ∧ 1 => 0
∧ 4
∧ 8 ∧ 6 = 2

Sorted array.

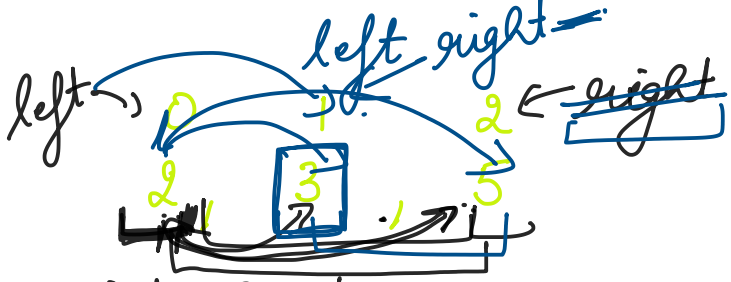
Q => arr = { 2, 3, 5 }

Count of distinct rectangles ?

area = 15.

2x2 , 2x3 , ~~3x5~~ , 3x3 , 2x5

5x2 , 3x2 => 6 distinct rectangles.



2 * 5 < 15

2x5	2x3
5x2	3x2

+ 2x2

3x3

```
int Count = 0
int area = 15;
while (left <= right) {
    if (arr[left] * arr[right] < area) {
        Count = Count + 2 * (right - l
        left++
    } else {
        right--;
    }
}
```

+1

two pointer approach.