Bit-Manipulation

video-28

Leetcode
- 3314
3315
Easy/Medium

codestorywithmik

CSwithMIK

codestorywithMIK

Try this channel to

see "Life behind the Scenes"
+ "Tech news"

Motivation

Always choose the harder right over the easier wrong.

The path to growth/success/reward is often harder. No shortcut.
If you have that will, you'll get it.

MIK...

3314. Construct the Minimum Bitwise Array I & II

Easy Medium | Topics | Companies | Hint

You are given an array `nums` consisting of `n` prime integers.

You need to construct an array `ans` of length `n`, such that, for each index `i`, the bitwise `OR` of `ans[i]` and `ans[i] + 1` is equal to `nums[i]`, i.e. `ans[i] OR (ans[i] + 1) == nums[i]`.

Additionally, you must **minimize** each value of `ans[i]` in the resulting array.

If it is *not possible* to find such a value for `ans[i]` that satisfies the **condition**, then set `ans[i] = -1`.

Example:- nums = [2, 3, 5, 7]

[1, 1, 4, 3]

Output = [-1, 1, 4, 9]

Part I

Constraints:

- `1 <= nums.length <= 100`
- `2 <= nums[i] <= 1000`
- `nums[i]` is a prime number.

$O(100 * 1000)$

T.C. = $O(10^5)$

Part-II

Constraints:

- `1 <= nums.length <= 100`
- `2 <= nums[i] <= 10^9`
- `nums[i]` is a prime number.

$O(100 * 10^9)$

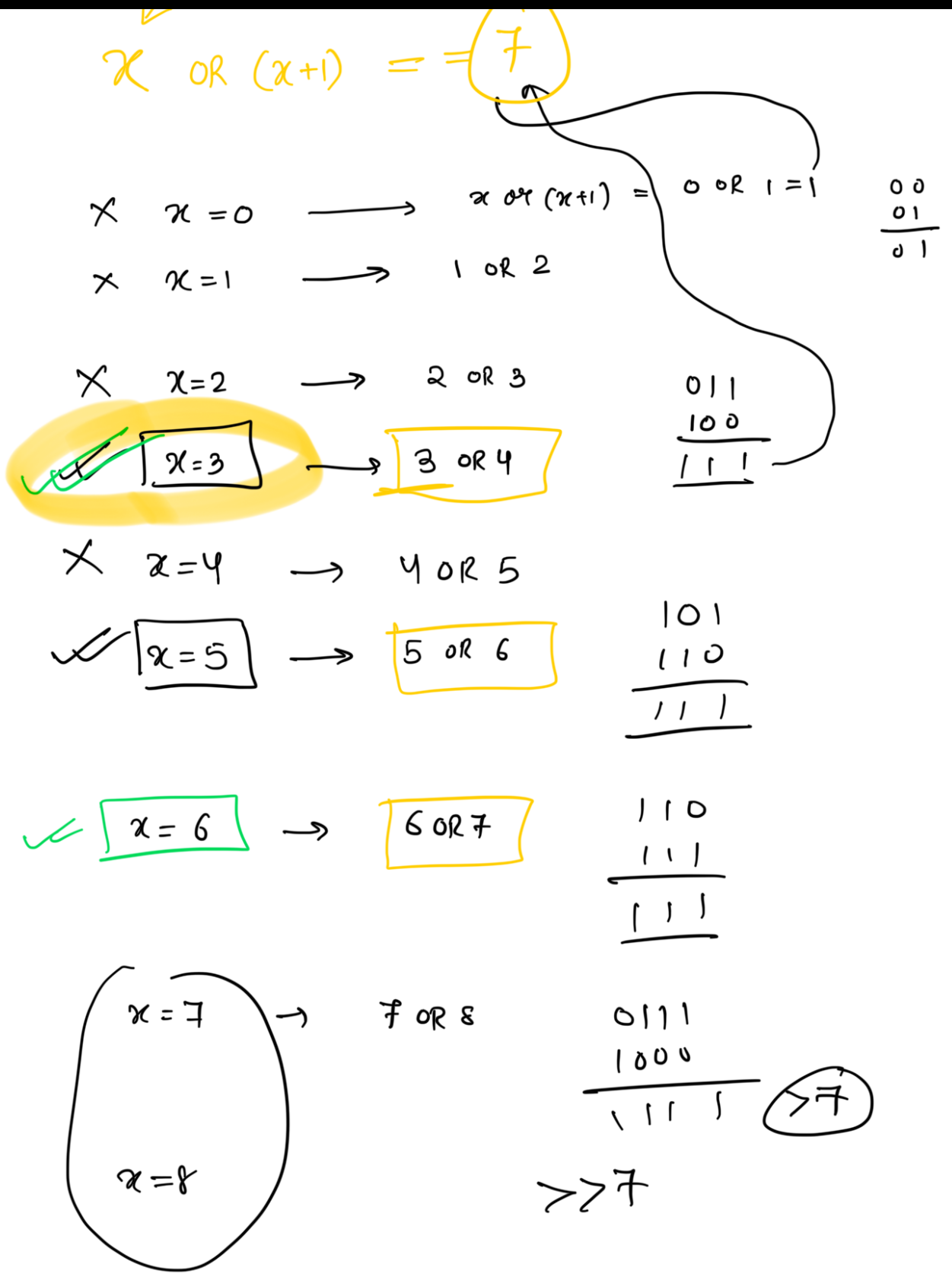
$O(10^{11})$ T.L.E.

Thought Process

nums = [2, 3, 5, 7]

{ x OR (x+1) x' OR (x'+1)

7



nums(i)

x = 0 to x < nums[i]

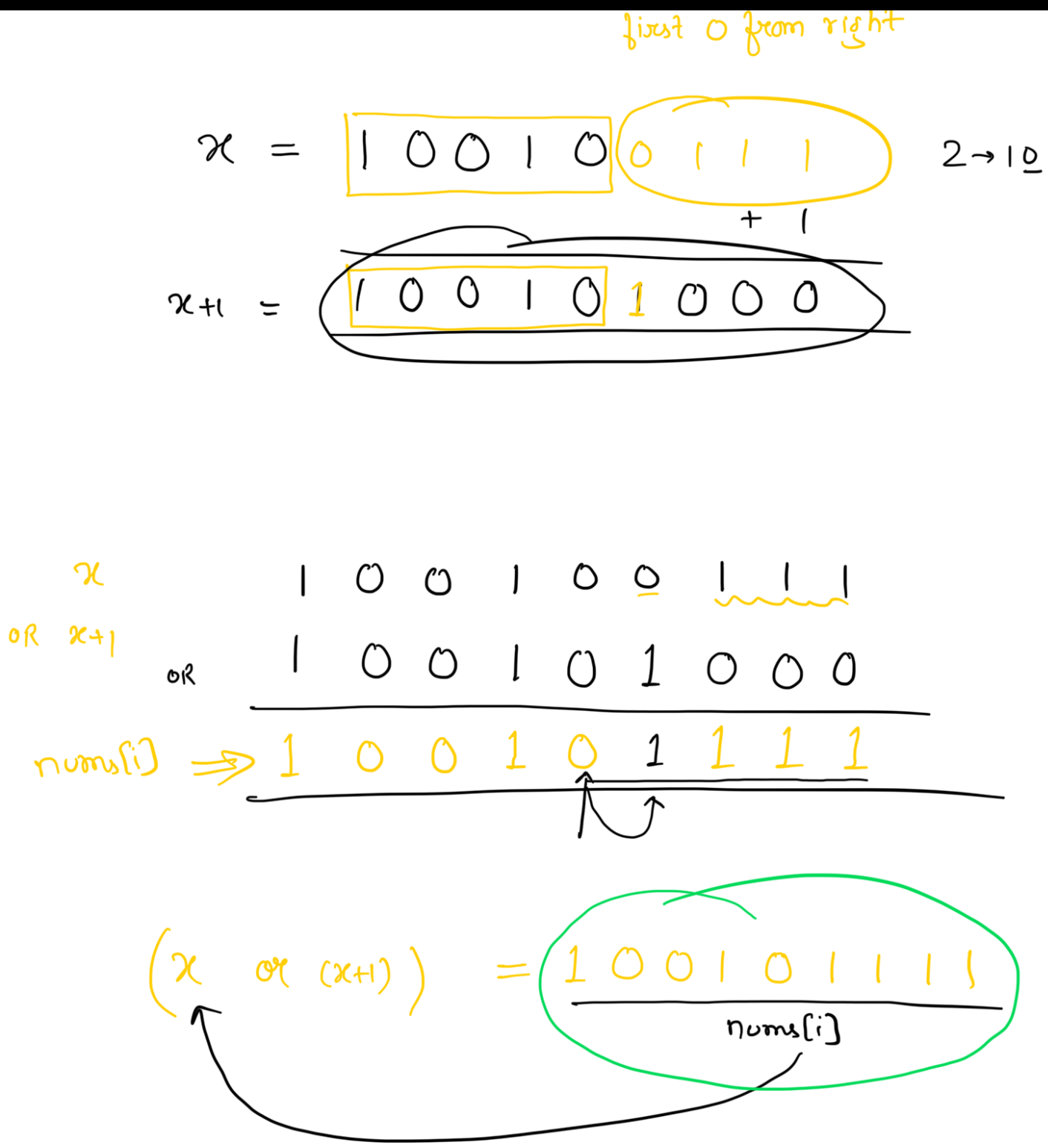
```
for (int i=0 ; i<n ; i++) {
    bool found = False;
    for (int x=0 ; x<nums[i] ; x++) {
        if ((x | (x+1)) == nums[i]) {
            result.push_back(x);
            Found = True;
            break;
        }
    }
    if (Found == False) {
        result.push_back(-1);
    }
}
```

T.C = O(n * max(nums[i]))

S.C = O(1)

Optimal Approach (Part - II)

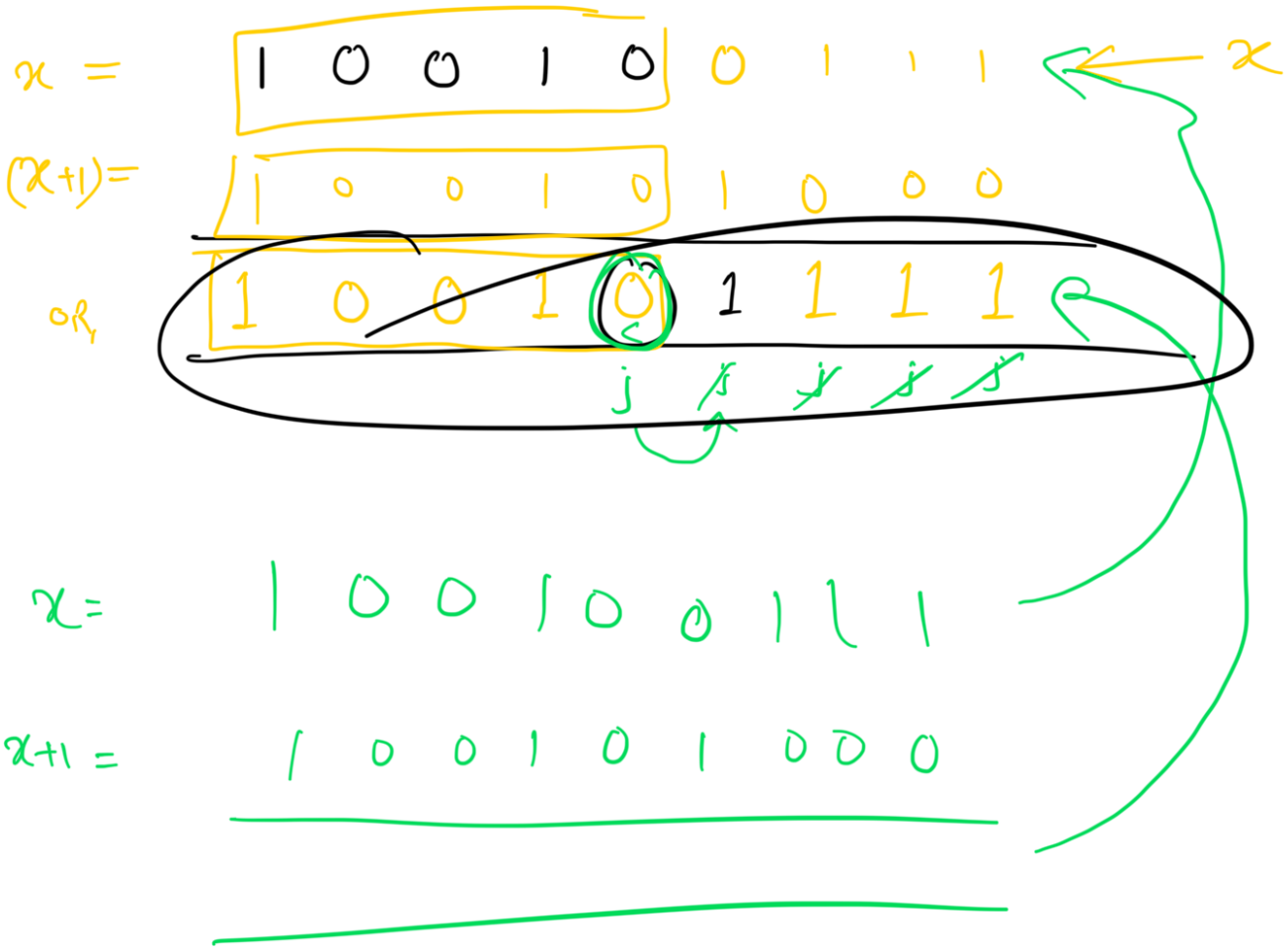
x OR (x+1) = nums[i]



nums[i] → find first 0-bit (j^{th} bit)

↓

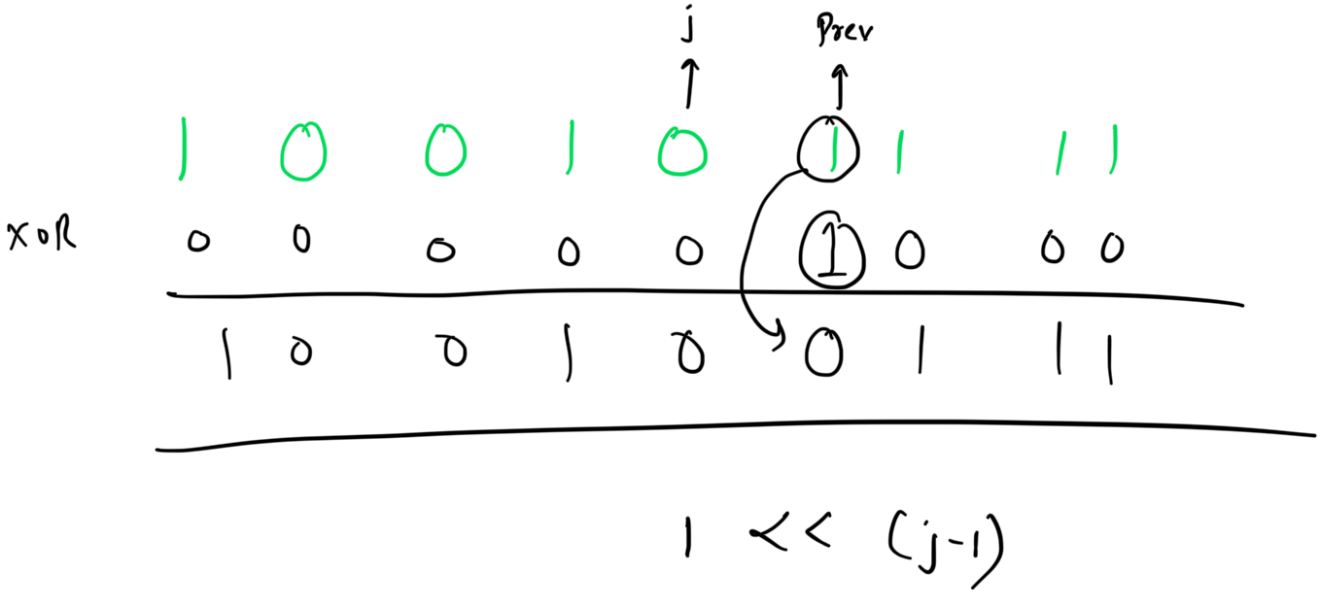
(j-1) → 1-bit → make it 0.




```
for (int i=0; i<n; i++) {
    if (nums[i]==2) {
        res.pb(-1); cont++;
    }
    for (j=0; j<32; j++) {
        if ((nums[i] & (1<<j)) >0) {
            continue;
        }
        Prev = (j-1);
```

T.C = O(n)
S.C = O(1)

```
//make Prev-bit 0;
x = (nums[i] ^ (1<<(j-1)));
result.push_back(x);
found = True;
break;
}
```



1 << (j-1)

nums[i] → Prime.

(2), 3, 5, 7, 11, 13 → odd.

Even.

nums[i]

2 → 1 0
j

Prev = (j-1)