Maximizing XOR

All topics

Bitwise XOR

XOR(^) is a binary operation called exclusive OR and works as

```
1^0 = 1
0^1 = 1
0^0 = 0
1^1 = 0
```

XOR by 1 can work like a toggle switch that turns `1` to `0` or `0` to `1`.

Another interesting thing to note is

```
x^0 = x
x^x = 0
```

Usage:

Problem 1: Given a number N. Flip all bits in its binary representation.

Solution 1: N^$((1 << 32) - 1)$ considering N is 32 bit integer.

Problem 2: Given two numbers A and B. Swap A and B without using airthmetic operator and without using third variable.

Solution 2:

$A = A \wedge B$

$B = A \wedge B$

$A = A \wedge B$

Related challenge for **Bitwise XOR**

Lonely Integer

Success Rate: **96.42%** Max Score: **20** Difficulty:

Solve Challenge

Finding Max Min

This is a simple operation. Suppose we need the index of the maximum element.

```
max_val = -1 \\initialise max
index = -1 \\take some initial impossible index to consider possibility of element not fou
nd.
for (int i = 0;i<length;i++) {
    if (arr[i]>max_val) {
        max_val = arr[i];
        index = i;
    }
}
```

Similarly, we can get minimum value and index.

We can also collect multiple maximum values by making a small modification as

```
\\let container be a vector or a list
if (arr[i]>max_val) {
    max_val = arr[i];
    index = i;
    container.clear();
}
else if (arr[i] == max_val) {
    container.push_back(i);
}
```

Related challenge for **Finding Max Min**

ACM ICPC Team

Success Rate: **87.86%** Max Score: **25** Difficulty:

Solve Challenge