Bit Manipulation

- Srivaths P

Goal:

- Understand bit manipulation and its use cases.
- Learn tips for bit manipulation.
- Learn brute-forcing using bitmasks.
- Find common patterns in bit manipulation problems.

Where is Bit Manipulation used?

Problems may or may not directly involve bit manipulation.

Some problems use bitwise operators in the problem statement itself, whereas some problems might use bit manipulation indirectly.

For example, if the problem involves powers of 2, it might be related to binary. Similarly, powers of other numbers might be in their own base.

How to approach Bit Manipulation problems?

“When solving a bitwise problem, think in bitwise”

- My mentor, 2020

When solving a problem that uses addition, multiplication, etc. we use decimal, which is the appropriate base for us.

When solving a problem that uses a bitwise operator, we need to use binary, which is the appropriate base.

Bitwise Operators - NOT

The NOT operator (“~”) flips every bit of the given number.
For example: $\sim 12 \; = \; -13$

A	Result
1	0
0	1

Bitwise Operators - OR

The OR operator (“|”) takes every corresponding bit of the two numbers and checks whether ***at least one*** of them is set.

A	B	Result
1	1	1
1	0	1
0	1	1
0	0	0

Bitwise Operators - AND

The AND operator (“&”) takes every corresponding bit of the two numbers and checks whether ***both*** of them is set.

A	B	Result
1	1	1
1	0	0
0	1	0
0	0	0

Bitwise Operators - XOR

The XOR operator (“^”) takes every corresponding bit of the two numbers and checks whether ***exactly one*** of them is set.

A	B	Result
1	1	0
1	0	1
0	1	1
0	0	0

Bitwise Operators - Misc

- Bitwise Left Shift ("<<"):

  $A << B$ shifts the A to the left by B bits, adding B zeros at the end.
  This value is the same as $A \times 2^B$.

$$18 << 3 = 144$$

- Bitwise Right Shift (">>"):

  $A >> B$ shifts the A to the right by B bits, deleting B zeros from the end.
  This value is the same as floor of $A \div 2^B$.

$$18 >> 3 = 2$$

Bitmasking

A bitmask is a sequence of N bits that encodes a subset, where the element is taken if a bit is set, and not taken if a bit is unset.

Ex: 10110 would mean indices $1, 2, 4$ are taken, while $0, 3$ are not.

By generating all bitmasks of some size, we can easily generate all subsets of an array.

This way, we can use iteration instead of recursion to generate subsets.

Bitmasking – Brute Force Code

```
for (int mask = 0; mask < (1 << n); mask++) {
    for (int i = 0; i < n; i++)
        if ((mask >> i) & 1)
            cout << a[i] << " ";

    cout << endl;
}
```

Types of problems:

- Bit-independent problems:
  Can solve for each bit separately. The solution for a bit ‘x’ will not depend on the solution for a bit ‘y’.

- Bit-dependent problems:
  Cannot solve for each bit separately. All the bits are interconnected and cannot be treated as independent.

Problems:

- https://www.codechef.com/problems/BITEQU
- https://www.atcoder.jp/contests/abc289/tasks/abc289_c
- https://www.codechef.com/problems/LOSTARRAY_
- https://www.codechef.com/problems/MAXAND18

Some Properties

- OR/AND/XOR are associative and commutative.
- $A \text{ ^ } 0 = A$
- $A \text{ ^ } A = 0$
- If $A \text{ ^ } B = C$, then $A \text{ ^ } C = B$
- $A \text{ ^ } B \text{ ^ } B = A$
- $A \,\&\, B \leq \min(A, B)$
- $A \mid B \geq \max(A, B)$
- $(A \mid B) + (A \,\&\, B) = A + B$
- $(A \,\&\, 1)$ is 1 if A is odd, else 0
- $A \,\&\, (A - 1)$ is 0 if A is a power of 2 (except when $A = 0$)

Resources

Number base:

https://brilliant.org/wiki/number-base/

Bit manipulation:

- https://codeforces.com/blog/entry/73490 (highly recommended)
- https://www.hackerearth.com/practice/basic-programming/bit-manipulation/basics-of-bit-manipulation/tutorial/

Bitsets:

- https://www.youtube.com/watch?v=jqJ5s077OKo

Resources

More properties:

https://stackoverflow.com/questions/12764670/are-there-any-bitwise-operator-laws

Useful tricks and formulas:

- https://www.geeksforgeeks.org/bitwise-hacks-for-competitive-programming/
- https://www.geeksforgeeks.org/bit-tricks-competitive-programming/
- https://www.geeksforgeeks.org/bits-manipulation-important-tactics/
- https://www.geeksforgeeks.org/builtin-functions-gcc-compiler/

Resources - Extra

On bitwise operators:

https://medium.com/biffures/bits-101-120f75aeb75a

https://medium.com/biffures/part-2-the-beauty-of-bitwise-and-or-cdf1d8d87891

https://medium.com/biffures/part-3-or-and-20ccc9938f05

https://medium.com/biffures/part-4-bitwise-patterns-7b17dae3eee0