A summary: how to use bit manipulation to solve problems easily and efficiently

LHearen ★ 2562 Last Edit: October 26, 2018 6:51 AM

1.7K

Wiki

Bit manipulation is the act of algorithmically manipulating bits or other pieces of data shorter than a word. Computer prog tasks that require bit manipulation include low-level device control, error detection and correction algorithms, data comp encryption algorithms, and optimization. For most other tasks, modern programming languages allow the programmer to directly with abstractions instead of bits that represent those abstractions. Source code that does bit manipulation make bitwise operations: AND, OR, XOR, NOT, and bit shifts.

Bit manipulation, in some cases, can obviate or reduce the need to loop over a data structure and can give many-fold sp bit manipulations are processed in parallel, but the code can become more difficult to write and maintain.

Details

Basics

At the heart of bit manipulation are the bit-wise operators & (and), | (or), ~ (not) and ^ (exclusive-or, xor) and shift operat and a >> b.

> There is no boolean operator counterpart to bitwise exclusive-or, but there is a simple explanation. The exclusive-or takes two inputs and returns a 1 if either one or the other of the inputs is a 1, but not if both are. That is, if both inputs both inputs are 0, it returns 0. Bitwise exclusive-or, with the operator of a caret, ^, performs the exclusive-or operation pair of bits. Exclusive-or is commonly abbreviated XOR.

- Set union A | B
- Set intersection A & B
- Set subtraction A & ~B
- Set negation ALL_BITS ^ A or ~A
- Set bit A |= 1 << bit
- Clear bit A &= ~(1 << bit)
- Test bit (A & 1 << bit) != 0
- Extract last bit A&-A or A&~(A-1) or x^(x&(x-1))
- Remove last bit A&(A-1)
- Get all 1-bits ~0

Examples

Count the number of ones in the binary representation of the given number

```
int count_one(int n) {
    while(n) {
        n = n&(n-1);
        count++;
    }
```