- 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++;
    }
    return count;
}
```

Is power of four (actually map-checking, iterative and recursive methods can do the same)

```
bool isPowerOfFour(int n) {
    return !(n&(n-1)) && (n&0x55555555);
    //check the 1-bit location;
}
```

^ tricks

Use ^ to remove even exactly same numbers and save the odd, or save the distinct bits and remove th

Sum of Two Integers

Use ^ and & to add two integers

```
int getSum(int a, int b) {
    return b==0? a:getSum(a^b, (a&b)<<1); //be careful about the terminating condit
}
```

Missing Number

Given an array containing n distinct numbers taken from 0, 1, 2, ..., n, find the one that is missing from example, Given nums = [0, 1, 3] return 2. (Of course, you can do this by math.)