Bit Manipulation

Bit Manipulation

- Basic operation
- Basic use case
- Interview Questions

Basic Operation

- Bitwise NOT
  - ~0111 = 1000
- Bitwise OR
  - 0101 | 1010 = 1111
- Bitwise AND
  - 0101 & 1100 = 0100
- Bitwise XOR
  - 0101 ^ 0011 = 0110

Basic Operation

- Bitwise SHIFT
  - 0001 << 3 = 1000
  - 1100 >> 1 = 1110 -- arithmetic
  - 1100 >>> 1 = 0110 -- logical

Basic use case (Tricks)

- Even & Odd
  - n & 1 == 1 ? Odd : Even
- Double
  - a / 2 = a >> 1;
  - a * (2^m) = a << m;
- Negative number
  - a = 101101100
  - -a = 010010011 + 1 = 010010100
  - Bit NOT on every bit and then plus 1.

Basic use case (Tricks)

- Specific bit operation
  - Get kth bit: (a >> k) & 1
  - Set kth bit 0: a = a & ~(1 << k)
  - Set kth bit 1: a = a | (1 << k)
  - Rightmost 1-bit: a & -a
- Absolute value ***
  - |a| = a ^ (a >> 31) - (a >> 31)

Basic use case (Tricks)

- XOR
  - a ^ a = 0
  - a ^ 0 = a
- Swap (int a, int b)
  - a ^= b;
  - b ^= a; // b = b ^ (a' ^ b) = a'
  - a ^= b; // a = (a' ^ b') ^ a' = b'

Interview Questions

- Bit operation
  - Single Number
  - Number of 1 Bits
  - Reverse Bits
- Reduce storage & improve efficiency
  - Combination

Single Number

Given an array of integers, every element appears *twice* except for one. Find that single one.

```
public int singleNumber(int[] A) {
    for (int i = 1; i < A.length; i++) {
        A[i] ^= A[i-1];
    }
    return A[A.length-1];
}
```

Number of 1 Bits

Write a function that takes an unsigned integer and returns the number of '1' bits it has. (also known as the Hamming weight)

```
public int hammingWeight(int n) {
    int result = 0;
    for (int i = 0; i < 32; i++) {
        result += n & 1;
        n >>= 1;
    }
    return result;
}
```

Reverse Bits

Reverse bits of a given 32 bits unsigned integer.

```
public int reverseBits(int n) {
    int reversed = 0;
    for (int i = 0; i < 32; i++) {
        reversed <<= 1;
        reversed |= n & 1;
        n >>= 1;
    }
    return reversed;
}
```

Single Number II

iven an array of integers, every element appears three times except for one, which appears exactly once. Find that single one.

Instead of using XOR directly, we need to count how many times 1 shows on each bit, it could be 1 or 3.

Single Number II

```
public int singleNumber(int[] nums) {
    int[] bitnum = new int[32];
    int res = 0;
    int n = nums.length;
    for(int i = 0; i < 32; i ++){
        for(int j = 0; j < n; j ++){
            bitnum[i] += (A[j] >> i) & 1;
        }
        res |= (bitnum[i] % 3) << i;
    }
    return res;
}
```

Single Number II

```
public int singleNumber(int[] nums) {
    int ans = 0;
    for(int i = 0; i < 32; i++) {
        int sum = 0;
        for(int j = 0; j < nums.length; j++) {
            if(((nums[j] >> i) & 1) == 1) {
                sum++;
                sum %= 3;
            }
        }
        if(sum != 0) {
            ans |= sum << i;
        }
    }
    return ans;
}
```

Single Number III

Given an array of numbers nums, in which exactly two elements appear only once and all the other elements appear exactly twice. Find the two elements that appear only once.

For example:

Given nums = [1, 2, 1, 3, 2, 5], return [3, 5].

Single Number III

We can use XOR to get the result, which is the XOR of the two numbers.

Then how we split them into two numbers?

Find the first bit that is different. Then the XOR value will be 1.

How to find the first bit 1?

Single Number III

```
public int[] singleNumber(int[] nums) {
    // Get the XOR of the two numbers we need to find
    int diff = 0;
    for (int num : nums) {
        diff ^= num;
    }
    // Get its last set bit
    diff &= -diff;

    int[] rets = new int[2];
    for (int num : nums)
    {
        if ((num & diff) == 0) // the bit is not set
        {
            rets[0] ^= num;
        }
        else // the bit is set
        {
            rets[1] ^= num;
        }
    }
    return rets;
}
```

Combination

Given n different numbers,return all combinations.

- n numbers, 2^n combinations
- 2^n status can be represented by n-bits integer
  - {1, 2, 3, 4}, 1100 -> {1, 2}
  - {1, 2, 3, 4}, 0000 -> {}

Combination

Given n different numbers,return all combinations.

```
public static ArrayList<ArrayList<Integer>> combination(int[] nums) {
    ArrayList<ArrayList<Integer>> results = new ArrayList<>();
    for (int i = 0; i < (1 << nums.length); i++) {
        results.add(generate(nums, i));
    }
    return results;
}

public static ArrayList<Integer> generate(int[] nums, int index) {
    ArrayList<Integer> result = new ArrayList<>();
    for (int i = 0; i < nums.length; i++) {
        if ((index & 1) == 1) {
            result.add(nums[i]);
        }
        index >>= 1;
    }
    return result;
}
```

Homework

Number Complement

Total Hamming Distance

Power Of Four

Maximum XOR of Two Numbers in an Array

Bitwise AND of Numbers Range