BINARY

Bit Manipulation in C

- **Operators**

  & AND **|** OR **^** XOR **~** NOT **<<** LEFT SHIFT **>>** RIGHT SHIFT

- **Common Operations**

  set bit: `num |= (1 << pos)`

  clear bit: `num &= ~(1 << pos)`

  toggle bit: `num ^= (1 << pos)`

  check bit: `(num & (1 << pos)) != 0`

  extract bit: `(num >> pos) & 1`

  extract a range of bits: `(num >> pos) & ((1 << length) - 1)`

- **Example**

```
void copyBit(int *dst, int src, int srcPos, int dstPos) {
      int bit = (src >> srcPos) & 1;  // extract bit
      *dst &= ~(1 << dstPos);  // clear destination bit
      *dst |= (bit << dstPos);  // set destination bit
}
```

Binary

- In C++, **`std::bitset`** represents a fixed-size sequence of N bits
- Example:

```
std::bitset<8> bitmap;
bitmap.reset(1)
bitmap.set(1)
if (bitmap.test(1)) { // true
…
```

- **reset** : set bit to false
- **set** : set a specific bit
- **test** : check a specific bit
- **count** : return the number of bits set to true
- **flip** : toggle the value of the bits (if true, set to false and vice-versa)

Problem – 371. Sum of Two Integers

Medium

LeetCode leetcode.com/problems/sum-of-two-integers

Problem

- Sum two integer numbers **a** and **b**
- You can't use **+** or **-**

Solution – 371. Sum of Two Integers

Medium

LeetCode leetcode.com/problems/sum-of-two-integers

Solution

- **Example:**

  a = 101 and b = 110

  expected result = 101 + 110 = 1011

- **Find the position where carry occurs using and operation:**

  101 & 110 = 100

- **Sum without carry using XOR:**

  101 ^ 110 = 011

- **Shift left the carry:**

  100 << 1 = 1000

- **Add the previous sum to the carry**

  011 ^ 1000 = 1011

- **Check the carry from the previous operation:**

  011 ^ 1000 = 0000

- **Now if it is zero, the result is 1011. Otherwise, repeat the process**

Code – 371. Sum of Two Integers

Medium

LeetCode leetcode.com/problems/sum-of-two-integers

Code

Time: **O(1)** Space: **O(1)**

```
int getSum(int a, int b) {
    while (b != 0) {
        // find which bits produce a carry
        // e.g. 101 & 111 = 101
        int carry = static_cast<unsigned>(a & b);
        // adds without carry (XOR)
        // 101 ^ 111 = 010
        a = a ^ b;
        // move carry to the left
        // b = 110
        b = carry << 1;
    }
    return a;
}
```

Problem – 191. Number of 1 Bits

Easy

LeetCode leetcode.com/problems/number-of-1-bits

Problem

- You are given a positive integer n
- Return the number of set bits
- **Example:**

  `n = 11`

  output: 3

  Binary is 1011, hence three set bits “1”

Solution – 191. Number of 1 Bits

Easy

LeetCode leetcode.com/problems/number-of-1-bits

Solution

- Use Brian Kernighan's Bit Counting Algorithm
- `removes the rightmost 1-bit from n in each iteration`
- `Example:`

  `n = 12 (1100)`

  First iteration: n=1100, n-1=1011, so 1100 & 1011 = 1000 (removed rightmost 1)

  Second iteration: n=1000, n-1=0111, so 1000 & 0111 = 0000 (removed last 1)

  Result: 2 iterations = 2 ones counted

Code – 191. Number of 1 Bits

Easy

LeetCode leetcode.com/problems/number-of-1-bits

Code

```cpp
// Brian Kernighan's algorithm
int hammingWeight(int n) {
    // return std::popcount(n);
    int count = 0;
    while (n) {
        n &= (n - 1);
        ++count;
    }
    return count;
}
```

Problem – 338. Counting Bits

Easy

LeetCode leetcode.com/problems/counting-bits

Problem

- You are given an integer n representing n different numbers starting from 0
- Return an array with the number of 1 for each integer
- **Example:**

  `n = 3`

  `0 → 0 (zero ‘1’s)`

  `1 → 1 (one ‘1’)`

  `2 → 10 (one ‘1’)`

  `3 → 11 (two ‘1’s)`

  Output: `[0,1,1,2]`

Solution – 338. Counting Bits

Easy

LeetCode leetcode.com/problems/counting-bits

Solution

- One solution is to code **Brian Kernighan's algorithm** and then call it **multiple times**; **or**
- **Use previously calculated results**: the bit count of any number i equals the bit count of i/2 plus 1 if the last bit is 1
- `res[i] = res[i >> 1] + (i & 1)`

  `i >> 1` removes the rightmost bit (divides by 2)

  `i & 1` checks if the rightmost bit is 1

Code – 338. Counting Bits

Easy

LeetCode leetcode.com/problems/counting-bits

Code

Time: **O(n)** Space: **O(n)**

```
vector<int> countBits(int n) {
    vector<int> res(n + 1, 0);
    for (int i = 1; i <= n; ++i) {
        // i >> 1 = i is divided by 2 (i with the last bit removed)
        // i & 1 is 1 if the last bit is set
        // number of set bits in i = number of set bits in i / 2 + 1 if last bit is 1.
        res[i] = res[i >> 1] + (i & 1);
    }
    return res;
}
```

Problem – 190. Reverse Bits

Easy

LeetCode leetcode.com/problems/reverse-bits

Problem

- You are given a **32 bits unsigned integer**
- **Reverse** its bits
- Example:

  00110 → 01100
- **Do not confuse this with "bit flipping"!**

Solution – 190. Reverse Bits

Easy

LeetCode leetcode.com/problems/reverse-bits

Solution

- Initialize a 32 bits integer **result** with 0
- As you know it is 32 bits, loop 32 times, and inside the loop:
- **Shift** all bits of **result** to the left (makes room for the new bit)
- Extract the right-most bit from the input
- Set the extracted bit as the new right-most bit of **result**
- Shift the input bits to the left

Code – 190. Reverse Bits

Easy

LeetCode leetcode.com/problems/reverse-bits

Code

Time: **O(-)** Space: **O(-)**

```
uint32_t reverseBits(uint32_t n) {
    int result = 0;
    for (int i = 0; i < 32; ++i) {
        // shift to left
        result = result << 1;
        // extract the right-most bit from n
        int bit = n & 1;
        // set bit
        result = result | bit;
        // shift to right to check the next bit
        n = n >> 1;
    }
    return result;
}
```

Negabinary

- Non-standard positional numeral system that uses base of -2
- Allow representing negative numbers in binary
- Example:

  1101_{-2}

  $(-2)^3 + (-2)^2 + 0 + (-2)^0 = -8 + 4 + 0 + 1 = -3$

Summing Negabinary

- Add as a regular binary number, but with **negative carry**

  $0 + 0 = 0$

  $1 + 0 = 1$

  $1 + 1 = 0$ with a negative carry 1

  1 $+ 1 = 0$ (subtract)

  1 $+ 0 = 1$ with a positive carry 1

Negabinary

Example 1

```
  11 1
   1011
 + 1110
------
 = 110001
```

red 1 = negative carry

green 1 = regular carry

1 + 0 = 1

1 + 1 = 0 with negative carry 1

1 + 1 = 0

1 + 1 = 0 with negative carry 1

1 + 0 = 1 with positive carry 1

1 + 0 = 1

Example 2

```
  1111
  101010
+ 101100
--------
= 11110110
```

Reference

https://math.stackexchange.com/questions/3251605/how-to-add-negabinary-numbers

GRAPH (BFS/DFS)