

极客大学算法训练营

第十六课

位运算

覃超

Sophon Tech 创始人，前 Facebook 工程师

目录

- 位运算符
- 算数移位与逻辑移位
- 位运算的应用

为什么需要位运算

- 机器里的数字表示方式和存储格式就是 二进制

- 十进制 \longleftrightarrow 二进制：如何转换？

<https://zh.wikihow.com/%E4%BB%8E%E5%8D%81%E8%BF%9B%E5%88%B6%E8%BD%AC%E6%8D%A2%E4%B8%BA%E4%BA%8C%E8%BF%9B%E5%88%B6>

4(d): 0100

8(d): 01000

5(d): 0101

6(d): 0110

位运算符

含义	运算符	示例
左移	<<	0011 => 0110
右移	>>	0110 => 0011

位运算符

含义	运算符	示例
按位或		<div>0011 -----=> 1011 1011</div>
按位与	&	<div>0011 -----=> 0011 1011</div>
按位取反	~	<div>0011 => 1100</div>
按位异或（相同为零不同为一）	^	<div>0011 -----=> 1000 1011</div>

XOR - 异或

异或：相同为 0，不同为 1。也可用 “不进位加法” 来理解。

异或操作的一些特点：

$$x \wedge 0 = x$$

$$x \wedge 1s = \sim x \quad // \text{注意 } 1s = \sim 0$$

$$x \wedge (\sim x) = 1s$$

$$x \wedge x = 0$$

$$c = a \wedge b \Rightarrow a \wedge c = b, b \wedge c = a \quad // \text{交换两个数}$$

$$a \wedge b \wedge c = a \wedge (b \wedge c) = (a \wedge b) \wedge c \quad // \text{associative}$$

指定位置的位运算

1. 将 x 最右边的 n 位清零: $x \& (\sim 0 \ll n)$
2. 获取 x 的第 n 位值 (0 或者 1) : $(x \gg n) \& 1$
3. 获取 x 的第 n 位的幂值: $x \& (1 \ll n)$
4. 仅将第 n 位置为 1: $x \mid (1 \ll n)$
5. 仅将第 n 位置为 0: $x \& (\sim (1 \ll n))$
6. 将 x 最高位至第 n 位 (含) 清零: $x \& ((1 \ll n) - 1)$
7. 将第 n 位至第 0 位 (含) 清零: $x \& (\sim ((1 \ll (n + 1)) - 1))$

实战位运算要点

- 判断奇偶：
 $x \% 2 == 1 \rightarrow (x \& 1) == 1$
 $x \% 2 == 0 \rightarrow (x \& 1) == 0$
- $x \gg 1 \rightarrow x / 2$.
即： $x = x / 2; \rightarrow x = x \gg 1;$
 $mid = (left + right) / 2; \rightarrow mid = (left + right) \gg 1;$
- $X = X \& (X - 1)$ 清零最低位的 1
- $X \& -X \Rightarrow$ 得到最低位的 1
- $X \& \sim X \Rightarrow 0$

实战题目

- <https://leetcode-cn.com/problems/number-of-1-bits/>
- <https://leetcode-cn.com/problems/power-of-two/>
- <https://leetcode-cn.com/problems/reverse-bits/>
- <https://leetcode-cn.com/problems/n-queens/description/>
- <https://leetcode-cn.com/problems/n-queens-ii/description/>

N皇后的位运算解法 - Python

```
def totalNQueens(self, n):
    if n < 1: return []
    self.count = 0
    self.DFS(n, 0, 0, 0, 0)
    return self.count

def DFS(self, n, row, cols, pie, na):
    # recursion terminator
    if row >= n:
        self.count += 1
        return

    bits = (~(cols | pie | na)) & ((1 << n) - 1) # 得到当前所有的空位

    while bits:
        p = bits & -bits # 取到最低位的1
        bits = bits & (bits - 1) # 表示在p位置上放入皇后
        self.DFS(n, row + 1, cols | p, (pie | p) << 1, (na | p) >> 1)
        # 不需要revert cols, pie, na 的状态
```

Java

```
class Solution {
    private int size;
    private int count;

    private void solve(int row, int ld, int rd) {
        if (row == size) {
            count++;
            return;
        }
        int pos = size & ~(row | ld | rd);
        while (pos != 0) {
            int p = pos & (-pos);
            pos -= p; // pos &= pos - 1;
            solve(row | p, (ld | p) << 1, (rd | p) >> 1);
        }
    }

    public int totalNQueens(int n) {
        count = 0;
        size = (1 << n) - 1;
        solve(0, 0, 0);
        return count;
    }
}
```

DP + 位运算

LeetCode 338:

<https://leetcode-cn.com/problems/counting-bits/description/>

代码

```
vector<int> countBits(int num) {  
    vector<int> bits(num+1, 0);  
    for (int i = 1; i <= num; i++) {  
        bits[i] += bits[i & (i - 1)] + 1;  
    }  
    return bits;  
}
```

THANKS! |  极客大学