190 Reverse Bits

2018年4月3日 21:34

Question:

Reverse bits of a given 32 bits unsigned integer.
For example, given input 43261596 (represented in binary as **00000010100101000001111010011100**), return 964176192 (represented in binary as **00111001011110000010100101000000**).
Follow up:
If this function is called many times, how would you optimize it?
Related problem: Reverse Integer

来自 <https://leetcode.com/problems/reverse-bits/description/>

颠倒给定的32位无符号整数的二进制位。
例如，给定输入 43261596（二进制表示为 00000010100101000001111010011100 ），返回 964176192（二进制表示为 00111001011110000010100101000000）。
问题进阶:
如果多次调用这个函数，你将如何优化它？

Solution for Python3:

```python
class Solution1:
    # @param n, an integer
    # @return an integer
    def reverseBits(self, n):
        t = list(bin(n)[2:])
        r = ['0']*32
        r[-len(t):] = t
        r.reverse()
        return int(''.join(r), 2)

class Solution2:
    # @param n, an integer
    # @return an integer
    def reverseBits(self, n):
        oribin = '{0:032b}'.format(n)
        reversebin = oribin[::-1]
        return int(reversebin, 2)

class Solution3:
    # @param n, an integer
    # @return an integer
    def reverseBits(self, n):
        return int(bin(n)[2:].zfill(32)[::-1], 2)

class Solution4:
    # @param n, an integer
    # @return an integer
    def reverseBits(self, n):
        res = 0
        for _ in range(32):
            res = (res << 1) + (n & 1)
            print(res)
```

```
            n >>= 1
        return res

class Solution5:
    # @param n, an integer
    # @return an integer
    def reverseBits(self, n):
        n = (n >> 16) | (n << 16);
        n = ((n & 0xff00ff00) >> 8) | ((n & 0x00ff00ff) << 8)
        n = ((n & 0xf0f0f0f0) >> 4) | ((n & 0x0f0f0f0f) << 4)
        n = ((n & 0xcccccccc) >> 2) | ((n & 0x33333333) << 2)
        n = ((n & 0xaaaaaaaa) >> 1) | ((n & 0x55555555) << 1)
        return n
```

Solution for C++:

```
//例子：abcdefgh
class Solution1 {
public:
    uint32_t reverseBits(uint32_t n) {
        //对半翻转：efgh abcd
        n = (n >> 16) | (n << 16);
        //两部分再分别各自对半翻转：gh ef cd ab
        n = ((n & 0xff00ff00) >> 8) | ((n & 0x00ff00ff) << 8);
        //继续各部分折半翻转：h g f e d c b a
        n = ((n & 0xf0f0f0f0) >> 4) | ((n & 0x0f0f0f0f) << 4);
        //间隔2位:1100 1100 1100 1100 ...(8块)
        //间隔2位:0011 0011 0011 0011 ...(8块)
        n = ((n & 0xcccccccc) >> 2) | ((n & 0x33333333) << 2);
        //间隔1位:1010 1010 1010 1010 ...(8块)
        //间隔1位:0101 0101 0101 0101 ...(8块)
        n = ((n & 0xaaaaaaaa) >> 1) | ((n & 0x55555555) << 1);
        return n;
    }
};

class Solution2 {
public:
    uint32_t reverseBits(uint32_t n) {
        int res = 0, k = 32;
        while (k--) {
            res = (res << 1) + (n & 1);
            n >>= 1;
        }
        return res;
    }
};
```

Appendix:

翻转字符串：abcdefgh:

翻转1/2：efghabcd

翻转1/4：ghefcdab

翻转1/8：hgfedcba

...

直到(1/n) * len(sgr) == 1