Next higher number with same 1-bits

CH Gowri Kumar
gkumar007@gmail.com

Problem

- Given a number *m* find the next higher number *r*, that has same number of 1-bits.
- Ex : 3 (0000011) => 5(0000101)
  6(0000110) => 9(0001001)
  11(0001011) => 13(0001101)
  23(0010111) => 27(0011011)
  24(0011000) => 33(0100001)
  44(0101100) => 49(0110001)
  46(0101110) => 51(00110011)

Observations I

- Look at the input and the outputs again and see if you can make some algorithm out of it
- 3 (0000011) => 5(0000101)
  6(0000110) => 9(0001001)
  11(0001011) => 13(0001101)
  23(0010111) => 27(0011011)
  24(0011000) => 33(0100001)
  44(0101100) => 49(0110001)
  46(0101110) => 51(00110011)

Observations II

- Hint : Now concentrate on the highlighted parts of input
- 3 (0000011) => 5(0000101)

  6(0000110) => 9(0001001)

  11(0001011) => 13(0001101)

  23(0010111) => 27(0011011)

  24(0011000) => 33(0100001)

  44(0101100) => 49(0110001)

  46(0101110) => 51(0110011)

Observations III

- As you can see,
  - the non-highlighted part is same in i/p and o/p as well
  - And the highlighted part is consecutive 1’s from the least-significant side (right hand side)
- 3 (0000011) => 5(0000101)

  6(0000110) => 9(0001001)

  11(0001011) => 13(0001101)

  23(0010111) => 27(0011011)

  24(0011000) => 33(0100001)

  44(0101100) => 49(0110001)

  46(0101110) => 51(0110011)

Observations IV

- As you can see, the non-highlighted part is same in i/p and o/p as well
- 3 (0000011) => 5(0000101)
  6(0000110) => 9(0001001)
  11(0001011) => 13(0001101)
  23(0010111) => 27(0011011)
  24(0011000) => 33(0100001)
  44(0101100) => 49(0110001)
  46(0101110) => 51(0110011)

Observations V

- Now lets just look at what changed
- 011 => 101

  0110 => 1001

  011 => 101

  0111 => 1011

  011000 => 100001

  01100 => 10001

  01110 => 10011
- Do you see a pattern?

Observations VI

- Yes, as you have rightly observed, left hand side is :
  - A 0 followed by
  - One or more 1's (say x) followed by
  - Zero or more 0's (say y)
- Is changed to
  - A 1 followed by
  - (y+1) zeroes followed by
  - (x-1) 1's
- 011 => 101

  011000 => 100001

Now let's frame the algorithm

- Given a bit-pattern, start from right, find successive zeroes (xxxx011110000)
- Followed by zeroes find successive 1's (xxxx011110000)
- Stop on hitting a zero (xxxx011110000)
- Interchange that zero with a 1 from successive 1's (xxxx101110000)
- Now move the remaining 1's to extreme right, filling the gap with zeroes (xxxx100000111)

Doing it programmatically in C

```
unsigned snoob(unsigned x) {
    unsigned smallest, ripple, ones;
    // x = xxx0 1111 0000
    smallest = x & -x; // 0000 0001 0000
    ripple = x + smallest; // xxx1 0000 0000
    ones = x ^ ripple; // 0001 1111 0000
    ones = (ones >> 2)/smallest; // 0000 0000 0111
    return ripple | ones; // xxx1 0000 0111
}
```

Reference

- Hackers Delight (chapter 2 – Basics)