explanation of bit manipulation method, might be easier to understand

Consider the following fact:

Write all numbers in binary form, then for any bit 1 that appeared 3\*n times (n is an integer), the bit can only present in numbers that appeared 3 times

e.g. 0010 0010 0010 1011 1011 1011 1000 (assuming 4-bit integers)  
2(0010) and 11(1011) appeared 3 times, and digit counts are:

**Digits** 3 2 1 0

**Counts** 4 0 6 3

**Counts%3** 1 0 0 0

Counts on 2,1,0 are all times of 3, the only digit index that has Counts % 3 != 0 is 3

Therefore, to find the number that appeared only 1 or 2 times, we only need to extract all bits that has Counts %3 != 0

Now consider how we could do this by bit manipulation

since counts % 3 has only 3 states: 0(00),1(01),2(10)  
we could use a TWO BIT COUNTER (Two, One) to represent Counts % 3, now we could do a little research on state transitions, for each bit, let B be the input bit, we can enumerate the all possible state transitions, Two+, One+ is the new state of Two, One. (here we need to use some knowledge in Digital Logic Design)

Two One B Two+ One+

0 0 0 0 0

0 0 1 0 1

0 1 0 0 1

0 1 1 1 0

1 0 0 1 0

1 0 1 0 0

1 1 0 X X (X represents we don't care)

1 1 1 X X

We could then draw the **Karnaugh map** to analyze the logic (<https://en.wikipedia.org/wiki/Karnaugh_map>), and then we get:

One+ = (One ^ B) & (~Two)

Two+ = (~One+) & (Two ^ B)

Now for int\_32, we need only 2 int\_32 two represent Two and One for each bit and update Two and One using the rules derived above

Code is here (C++):

**class** **Solution** {

**public**:

**int** **singleNumber**(vector<**int**>& nums) {

**int** counterOne = 0;

**int** counterTwo = 0;

**for** (**int** i = 0; i < nums.size(); i++){

counterOne = (~counterTwo) & (counterOne ^ nums[i]);

counterTwo = (~counterOne) & (counterTwo ^ nums[i]);

}

**return** counterOne;

}

};

The bitwise complement operator (~), which perform a bitwise negation of an integer value. Bitwise negation means that each bit in the number is toggled. In other words, all the binary 0s become 1s and all the binary 1s become 0s.