Sum of Two Integers

Calculate the sum of two integers *a* and *b*, but you are **not allowed** to use the operator `+` and `-`.

Example:
Given *a* = 1 and *b* = 2, return 3.

Credits:
Special thanks to @fujiaozhu for adding this problem and creating all test cases.

Solution 1

I have been confused about bit manipulation for a very long time. So I decide to do a summary about it here.

"&" AND operation, for example, 2 (0010) & 7 (0111) => 2 (0010)

"^" XOR operation, for example, 2 (0010) ^ 7 (0111) => 5 (0101)

"~" NOT operation, for example, ~2(0010) => -3 (1101) what??? Don't get frustrated here. It's called two's complement.

1111 is -1, in two's complement

1110 is -2, which is ~2 + 1, ~0010 => 1101, 1101 + 1 = 1110 => 2

1101 is -3, which is ~3 + 1

so if you want to get a negative number, you can simply do ~x + 1

Reference:

https://en.wikipedia.org/wiki/Two%27s_complement

https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html

For this, problem, for example, we have a = 1, b = 3,

In bit representation, a = 0001, b = 0011,

First, we can use "and"("&") operation between a and b to find a carry.

carry = a & b, then carry = 0001

Second, we can use "xor" ("^") operation between a and b to find the different bit, and assign it to a,

Then, we shift carry one position left and assign it to b, b = 0010.

Iterate until there is no carry (or b == 0)

```java
// Iterative
public int getSum(int a, int b) {
 if (a == 0) return b;
 if (b == 0) return a;

 while (b != 0) {
  int carry = a & b;
  a = a ^ b;
  b = carry << 1;
 }

 return a;
}

// Iterative
public int getSubtract(int a, int b) {
 while (b != 0) {
  int borrow = (~a) & b;
  a = a ^ b;
  b = borrow << 1;
 }

 return a;
}

// Recursive
public int getSum(int a, int b) {
 return (b == 0) ? a : getSum(a ^ b, (a & b) << 1);
}

// Recursive
public int getSubtract(int a, int b) {
 return (b == 0) ? a : getSubtract(a ^ b, (~a & b) << 1);
}

// Get negative number
public int negate(int x) {
 return ~x + 1;
}
```

written by zhaolz original link here

Solution 2

test cases all pass

0 ms

```
public int getSum(int a, int b) {
  if(b == 0){//没有进为的时候完成运算
      return a;
 }
 int sum,carry;
 sum = a^b;//完成第一步加发的运算
 carry = (a&b)<<1;//完成第二步进位并且左移运算
 return getSum(sum,carry);//
 }
```

written by lid004 original link here

Solution 3

```
class Solution {
public:
    int getSum(int a, int b) {
        int sum = a;

        while (b != 0)
        {
            sum = a ^ b;//calculate sum of a and b without thinking the carry
            b = (a & b) << 1;//calculate the carry
            a = sum;//add sum(without carry) and carry
        }

        return sum;
    }
};
```

written by vdvvdd original link here