Adding 2 BitSets in Java

uhmm well i need to add two BitSets in java .. i have tried out adding using the basic operations of XOR(for sum) and AND(for carry) .. **considering the carry as well** ..

but the answer isnt coming fully right ... this is what i have tried out..

```
public static BitStorage Add(int n, BitStorage ...manyBitSets)
{
    BitStorage sum = new BitStorage(0, n);      //discarding carry out of MSB
    System.out.print("Addition of: ");
    for(BitStorage bitStorage:manyBitSets)
    {
        //System.out.print(sum+"\t");
        //System.out.print(bitStorage+"\t");
        System.out.println("~~~~~");
        for(int i=n-1;i>=0;i--)
        {
            if(i==n-1)
            {
                System.out.println(sum + " + " +bitStorage);
                sum.set(i, sum.get(i)^bitStorage.get(i));
                //System.out.println(sum.get(i)+" XOR "+bitStorage.get(i));
            }
            else
            {
                System.out.println(sum + " + " +bitStorage+"\t"+(sum.get(i)?"1":"0"+"^"+
(bitStorage.get(i)?"1":"0")+"^"+(sum.get(i+1)?"1":"0"+"&"+
(bitStorage.get(i+1)?"1":"0"))));
                sum.set(i,
sum.get(i)^bitStorage.get(i)^(sum.get(i+1)&bitStorage.get(i+1)));      //carry taken here
                //System.out.println(sum.get(i)+" XOR "+bitStorage.get(i)+" XOR
("+bitStorage.get(i+1)+" AND "+sum.get(i+1));
            }
        }
    }
    return sum;
}
```

PS: BitStorage class is nothing but own implementation of BitSet with some additional methods .. like Add, Subtract, Shift etc

It has 2 memebers :

1. an integer (n) as max size (i dont want the vector's growing or shrinking to affect bitwise operations => therefore all operations are done wrt n) -> Eg: n is 4 then bits occupy the position o to 3 in BitSet
2. a BitSet object of size n passed into constructor

2 more points:

- **I thought of converting it to long or byte array and then adding but i need the solution ONLY in JDK 6 not 7**
- **I do not need the carry generated out of MSB, i want the answer in same no(bits) i.e n**

sorry for using "i want..." many times .. kinda tired . .tried many things! and uhmm i need this for a part of an algo.. looking forward to replies .. :) :)

java bit-manipulation bitwise-operators bits

edited Jul 29 '12 at 4:24 asked Jul 29 '12 at 4:10

riteshtt93
6,082 2 12 30

anybody.......? – riteshtt93 Jul 29 '12 at 4:50

While I applaud showing all your code like this... please explain why this "isn't quite right"... What answers are you getting that are incorrect? – billjamesdev Jul 29 '12 at 5:25

It took a while before I understood what what was going on, but I finally figured it out. It would help if next time a short example call is included. Your carry is broken, so you keep getting wrong bits set. – SilverbackNet Jul 29 '12 at 6:57

@BillJames thnx! . .and uhmm well many of the additions including carry had problems .. – riteshtt93 Jul 29 '12 at 11:46

@SilverbackNet Uhmm yes .. i figured that out .. after sometime when i posted the question .. – riteshtt93 Jul 29 '12 at 11:47

3 Answers

I'm tired, so forgive me if it's ugly. Your method of carrying is completely broken and setting wrong bits even when it shouldn't set anything. You should be counting up to carry up, and because of that, there's no reason to special case the last bit, the carry will just go away. By actually carrying up the last result into the next loop iteration, the logic is much simpler.

```
public static BitStorage Add(int n, BitStorage ...manyBitSets)
{
    BitStorage sum = new BitStorage(0, n);      //discarding carry out of MSB
    System.out.print("Addition of: ");
    for(BitStorage bitStorage:manyBitSets)
    {
        boolean carry = false;
        boolean lastcarry = false;
        //System.out.print(sum+"\t");
        //System.out.print(bitStorage+"\t");
        System.out.println("~~~~~");
        for(int i=0;i<n;i++)
        {
                System.out.println(sum + " + " +bitStorage+"\t"+(sum.get(i)?"1":"0"+"^"+
(bitStorage.get(i)?"1":"0")+"^"+(sum.get(i+1)?"1":"0"+"&"+
(bitStorage.get(i+1)?"1":"0"))));
                lastcarry = carry;
                carry = sum.get(i) && bitStorage.get(i);
                sum.set(i, lastcarry^sum.get(i)^bitStorage.get(i));      //carry taken
here
                //System.out.println(sum.get(i)+" XOR "+bitStorage.get(i)+" XOR
("+bitStorage.get(i+1)+" AND "+sum.get(i+1));
        }
    }
    return sum;
}
```

I used boolean for the variables because I built the class as a thin wrapper over BitSet, if you use int or whatever than change them.

answered Jul 29 '12 at 7:12
SilverbackNet
1,775 11 25

Thnx! for your post :) .. i was offline for sometime .. was having lunch .. while thinking about this .. . .i didnt try out your way .. but i guess i'll keep it as other options for future .. as of now i figured out the bug .. and implemented my own another logic .. and the answers were correct too! :) Anyways Thanks for your answer! :) – riteshtt93 Jul 29 '12 at 11:51

Well this was an interesting problem and kept me guessing for a while...

I wasn't able to derive the logic first but then I switched back to basics and derived the `boolean expression` for calculation the sum & carry for a 3 bit operation and here is the solution:

```
public static BitSet addBitSet(int n, List<BitSet> bitSetList){
    BitSet sumBitSet = new BitSet(n);
    for (BitSet firstBitSet : bitSetList) {
        BitSet secondBitSet = (BitSet) sumBitSet.clone();
        System.out.println("A:  " + printBitSet(firstBitSet, 6));
```

```
        System.out.println("B:  " + printBitSet(secondBitSet, 6));
        boolean carryForNext = false, sum,a,b,c;
        for (int i = n - 1; i >= 0; i--) {
            a=firstBitSet.get(i);
            b=secondBitSet.get(i);
            c=carryForNext;
            sum = a&!b&!c|!a&!b&c|!a&b&!c|a&b&c;
            carryForNext = a&b&!c|a&!b&c|!a&b&c|a&b&c;
            sumBitSet.set(i,sum);
        }
        System.out.println("SUM:" + printBitSet(sumBitSet, 6));
    }
    System.out.println(printBitSet(sumBitSet, 6));
    return sumBitSet;
}
```

and here is the code for `printBitSet` :

```
public static String printBitSet(BitSet bitSet, int size) {
    StringBuilder builder = new StringBuilder("");
    for (int i = 0; i < size; i++) {
        if (bitSet.get(i))
            builder.append("1");
        else
            builder.append("0");
    }
    return builder.toString();
}
```

answered Jul 29 '12 at 8:00

SiB
7,818 5 26 56

Thanks for your post! :) .. but i figured out the bug that was there .. and implemented my own anther logic as well ! :) .. and mine printing BitSet logic was same too :P .. except i had overriden toString! :P .. i'll keep your way as an option for future .. anyways thanks for your answer! :) – riteshtt93 Jul 29 '12 at 11:54

That's great. It kept me thinking for a while. Please accept an answer of this question. – SiB Jul 29 '12 at 13:42

I figured out the bug .. it was the problem of carry .. Instead using the same logic .. i settled onto another logic .. just XORed the 2 numbers (to get the sum) .. and again treated that as a number adding it to a number that has the carry(got from ANDing the original numbers).. The AND output(carry) is shifted left in every iteration .. because the LSB side bit additions don't have any carry to be added

We stop the loop when the carry bitset is all 0's(false)

An Example:

1. 0001
2. +0011
3. =0010 -->(XOR = sum)
4. +0010 -->(AND output = carry .. displayed after shifting left and now it will be added to XOR output)
5. =0000 -->(XOR = sum)
6. +0100 -->(AND output = carry .. displayed after shifting left and now it will be added to XOR output)
7. =0100 -->(XOR = sum .. **FINAL ANSWER**)
8. 0000 -->(AND output = carry .. displayed after shifting left .. WE STOP HERE .. stop because all are 0's))

A sample :)

```
import java.util.*;
public class BitSetAddition
{
static String nums[] =
{"0000","0001","0010","0011","0100","0101","0110","0111","1000","1001","1010","1011","1100",

public static void main(String args[])
{
    for(int q=0;q<nums.length;q++)
    {
        System.out.print(q+1+" -> ");
        BitSet b1 = new BitSet();
        String s = nums[q];
        b1.set(0, s.charAt(0)=='1'?true:false);
        b1.set(1, s.charAt(1)=='1'?true:false);
        b1.set(2, s.charAt(2)=='1'?true:false);
        b1.set(3, s.charAt(3)=='1'?true:false);

        for(int i=0;i<4;i++)
            System.out.print(b1.get(i)?"1":"0");
            System.out.print(" + ");

        BitSet b2 = new BitSet();
        String a = "0001";
        b2.set(0, a.charAt(0)=='1'?true:false);
        b2.set(1, a.charAt(1)=='1'?true:false);
        b2.set(2, a.charAt(2)=='1'?true:false);
        b2.set(3, a.charAt(3)=='1'?true:false);

        for(int i=0;i<4;i++)
            System.out.print(b2.get(i)?"1":"0");
            System.out.print(" = ");

        BitSet sum = new BitSet();
        BitSet carry = new BitSet();
        BitSet toAdd = new BitSet();
        BitSet tempSum = new BitSet();
        BitSet tempCarry = new BitSet();

        sum = b1;
        toAdd = b2;
        do
        {
            copy(4, tempSum, sum);
            copy(4, tempCarry, toAdd);
            tempSum.xor(toAdd);
            tempCarry.and(sum);
            copy(4, sum, tempSum);
            copy(4, carry, leftShift(4, tempCarry));
            copy(4, toAdd, carry);
            //sum.set(i, b1.get(i)^b2.get(i)^(b1.get(i+1)&b2.get(i+1)));
        }while(!carry.equals(new BitSet()));
            //if(i+2<=3)
                //sum.set(i, b1.get(i)^b2.get(i)^(b1.get(i+1)&b2.get(i+1)&(b1.get(i+2)&
(b2.get(i+2)))));
            //else if(i+1<=3)
                //sum.set(i, b1.get(i)^b2.get(i)^(b1.get(i+1)&b2.get(i+1)));
            //else
                //sum.set(i, b1.get(i)^b2.get(i));
        for(int i=0;i<4;i++)
            System.out.print(sum.get(i)?"1":"0");

        System.out.println();
    }
}
static void copy(int n,BitSet b, BitSet toCopy)
{
    for(int i=0;i<n;i++)
        b.set(i, toCopy.get(i));
}
static BitSet leftShift(int n, BitSet b)
{
    for(int i=0;i<n;i++)
        b.set(i, b.get(i+1));
        b.set(n-1, false);
        return b;
}
}
```

The comments in the program are of my previous logic .. i found my previous logic more complicated :P .. ignore the comments of the program if you want :)

Note: I didnt want the carry that comes out of MSB bit addition .. according to the algo (for which i needed this) .. so the answer (the sum) has the same num of bits .. anybody could adjust if they want :)

edited Jul 29 '12 at 12:43

answered Jul 29 '12 at 12:36

riteshT93
6,082 2 12 30