Code Jam 2011 - Qualification Round

Analysis: Candy Splitting

The main step needed to solve this problem is to understand Patrick's strange addition algorithm. For example, can we describe what happens when Patrick adds up several numbers instead of just two? It turns out we can: we should write all numbers in binary, align them by their least significant bit, and write 1 in those positions where we have an *odd* number of 1 bits in the summands (the numbers being added).

Consider this example: suppose Patrick needs to add 5, 7 and 9 together. First, he adds up 5 and 7 by writing them in binary and adding digit-by-digit without carry, as described in the problem statement:

```
  101
+ 111
-----
  010
```

The result is 010 in binary, which is 2. Now, he adds up 2 and 9:

```
  0010
+ 1001
------
  1011
```

The result is 1011 in binary, which is 11. It is most instructive to look at what happened to the least significant bit: after adding up two of the numbers, we had a 0 in the least significant bit since both of the summands had a 1 there. However, we have a 1 in the least significant bit of the overall result since the third number had a 1 there as well. It's not hard to see that this generalizes as described above: for any bit, it will be equal to 1 in the overall sum if and only if this bit is set to 1 in an odd number of summands.

Having established that, we can now understand Sean's task better. He needs to separate the given set of numbers into two parts in such a way that for every bit position, either:

- In both parts, an odd number of summands have this bit set to 1, so that the corresponding bit in the sum is 1 for both parts, or
- In both parts, an even number of summands have this bit set to 1, so that the corresponding bit in the sum is 0 for both parts.

But saying that we need two numbers to be either both odd, or both even, is equivalent to saying that their sum must be even!

That allows us to reformulate Sean's task simply as: he needs to separate the given set of numbers into two parts in such a way that for every bit position, an even number of summands have the bit set to 1 across both parts put together. Suddenly, we understand that this condition doesn't rely on the way we separate the numbers into two parts at all! Either it is true for every bit position that an even number of all summands have the bit set to 1, in which case any separation into two non-empty piles will make Patrick happy, or there is some bit which is 1 in an odd number of summands, in which case there's no way to make Patrick happy.

For example, suppose Sean has pieces of candy with values 5, 7, 9 and 11. If he separates them into 5 and 7 for himself, and 9 and 11 for Patrick, Patrick will add 5 and 7 as $5+7=101_2+111_2=010_2=2$, and 9 and 11 as $9+11=1001_2+1011_2=0010_2=2$, so Patrick is happy.

But even if Sean takes 7, 9 and 11 and leaves just 5 to Patrick, Patrick will add 7, 9 and 11 as $7+9+11=0111_2+1001_2+1011_2=0101_2=5$, so he is still happy! It's not difficult to verify that in all other cases Patrick is happy as well.

All the above reasoning can be made simpler if you notice that Patrick's strange addition is exactly bitwise exclusive or. The condition for Patrick's happiness can be reformulated as the bitwise exclusive or of all candy values being equal to zero. In many programming languages, bitwise exclusive or is already built in with the "^" operator, making this very easy to check!

Now, how does the overall solution for the problem work? First, we need to check if Patrick will be happy - as shown above, this does not depend on Sean's piles. If not, then we just output "NO" for this testcase. If yes, then Sean should maximize his pile, and this is achieved by taking all pieces of candy except the one with the smallest value.