题目描述：
Solo 和 koko 是两兄弟，妈妈给了他们一大堆积木，每块积木上都有自己的重量。现在他们想要将这些积木分成两堆。哥哥 Solo 负责分配，弟弟 koko 要求两个人获得的积木总重量“相等”（根据 Koko 的逻辑），个数可以不同，不然就会哭，但 koko 只会先将两个数转成二进制再进行加法，而且总会忘记进位（每个进位都忘记）。如当 25（11101）加 11（1011）时，koko 得到的计算结果是 18（10010）：

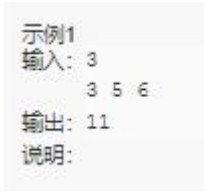
```
 11001
+01011
--------
 10010
```

Solo 想要尽可能使自己得到的积木总重量最大，且不让 koko 哭。
输入描述：
3

3 5 6

第一行是一个整数 N(2≤N≤100)，表示有多少块积木；第二行为空格分开的 N 个整数 Ci(1≤Ci≤106)，表示第 i 块积木的重量。
输出描述：
11
让 koko 不哭，输出 Solo 所能获得积木的最大总重量；否则输出“NO”。
补充说明：
如果能让 koko 不哭，输出 Solo 所能获得的积木的总重量，否则输出-1。
该样例输出为 11。

解释：Solo 能获得重量为 5 和 6 的两块积木，5 转成二级制为 101，6 转成二进制位 110，按照 koko 的计算方法（忘记进位），结果为 11(二进制)。Koko 获得重量为 3 的积木，转成二进制位 11(二进制)。Solo 和 koko 得到的积木的重量都是 11(二进制)。因此 Solo 可以获得的积木的总重量是 5+6=11（十进制）。



```python
import sys

n = int(sys.stdin.readline().strip())
weights = list(map(int, sys.stdin.readline().strip().split()))

def greedy(solo, koko, pool, possible):
    pool.sort()
    if pool:
        take = pool.pop()
        solo.append(take)
        if greedy(solo, koko, pool, possible):
            return True
        solo.pop()
        pool.append(take)
        take = pool.pop()
        koko.append(take)
        if greedy(solo, koko, pool, possible):
            return True
        koko.pop()
        pool.append(take)
        return False
    else:
        solo_total = 0
        koko_total = 0
        for w in solo:
            solo_total = solo_total ^ w
        for w in koko:
            koko_total = koko_total ^ w
        if solo_total == koko_total and len(solo)>0 and len(koko)>0:
            total = 0
            for i in solo:
                total += i
            possible.append(total)
            return True
        return False

possible = []
if greedy([], [], weights, possible):
    print(possible[0])
else:
    print(-1)
```