题目描述：

MELON 有一堆精美的雨花石（数量为 n，重量各异），准备送给 S 和 W。MELON 希望送给俩人的雨花石重量一致，请你设计一个程序，帮 MELON 确认是否能将雨花石平均分配。

输入描述：

第 1 行输入为雨花石个数：n， 0 < n < 31。

第 2 行输入为空格分割的各雨花石重量：m[0] m[1] ..... m[n-1]， 0 < m[k] < 1001。

不需要考虑异常输入的情况。

输出描述：

如果可以均分，从当前雨花石中最少拿出几块，可以使两堆的重量相等；如果不能均分，则输出 -1。

示例 1

输入：

```
4

1 1 2 2
```

输出：

```
2
```

说明：

输入第一行代表共 4 颗雨花石，第二行代表 4 颗雨花石重量分别为 1、1、2、2。

均分时只能分别为 1,2，需要拿出重量为 1 和 2 的两块雨花石，所以输出 2。

示例 2

输入：

```
10
1 1 1 1 1 9 8 3 7 10
```

输出：

```
3
```

说明：

输入第一行代表共 10 颗雨花石，第二行代表 4 颗雨花石重量分别为 1、1、1、1、1、9、8、3、7、10 。

均分时可以 1,1,1,1,1,9,7 和 10,8,3，也可以 1,1,1,1,9,8 和 10,7,3,1，或者其他均分方式，但第一种只需要拿出重量为 10,8,3 的 3 块雨花石，第二种需要拿出 4 块，所以输出 3(块数最少)。

```
import java.util.Scanner;
import java.util.HashMap;
import java.util.Arrays;

public class Main {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		int n = sc.nextInt();
		int[] nums = new int[n];
		int sum = 0;
		int max = Integer.MAX_VALUE;
		HashMap<Integer, Integer> map = new HashMap<>();
		int ans = 0, x;
		for (int i = 0; i < n; i++) {
			nums[i] = sc.nextInt();
			sum += nums[i];
//				if(map.getOrDefault(nums[i], null)==null){
//					map.put(nums[i], 1);
//				} else {

//				}

		}
		if(sum%2 != 0){
			System.out.println(-1);
			return ;
```

```
            }

//              Arrays.sort(nums);


//              help(nums, 0, n-1, sum/2);


            int amount = sum/2;
            int[] dp = new int[amount+1];

            for(int j = 0; j < dp.length; j++){
                  dp[j] = max;
            }
            dp[0] = 0;

            for(int i = 0; i < nums.length; i++){
                  for(int j = amount; j >= nums[i]; j--){
                        if(dp[j-nums[i]] != max) {
                              dp[j] = Math.min(dp[j], dp[j - nums[i]] + 1);
                        }
                  }
            }
            System.out.println(dp[amount] == max ? -1 : dp[amount]);
            return ;


      }


//          public static int help(int[] nums, int count, int idx, int sum){

//              if(sum < 0){
//                    return -1;
//              } else if(sum == 0){
//                    return count;
//              }

//              int min = Integer.MAX_VALUE;

//              for(int i = idx-1; i >= 0; i--){

//                    min = Math.min(min, help(nums, count+1, i,sum-nums[i]));
//              }
```

```
//                return min;
//        }
}
```