题目描述

又到了一年的末尾，项目组让小明负责新年晚会的小礼品发放工作。

为使得参加晚会的同事所获得的小礼品价值相对平衡，需要把小礼品根据价格进行分组，但每组最多只能包括两件小礼品，并且每个分组的价格总和**不能超过一个价格上限。**

为了保证发放小礼品的效率，小明需要找到分组数目最少的方案。

你的任务是写一个程序，找出分组数最少的分组方案，**并输出最少的分组数目。**

输入描述

第一行数据为分组礼品价格之和的上限

第二行数据为每个小礼品的价格，按照空格隔开，每个礼品价格不超过分组价格和的上限

输出描述

输出最小分组数量

用例

输入	5 1 2 5
输出	2
说明	无

题目解析

最少的分组方案，肯定是尽可能多的2件商品一组，避免产生1件商品一组。

对于价格大于等于上限的商品，只能独立一组，因此我们可以剔除这些商品。

然后对于价格不超过上限的商品，应该尽量用一个小价格和一个大价格组合，比如1，1，3，4，如果先组合1，1的话，那么3，4价格商品就必须独立成组了，如果1，3组合，1，4组合，那么就省了一组。

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 2) {
    const max = lines[0] - 0;
    const arr = lines[1].split(" ").map(Number);

    console.log(getResult(max, arr));

    lines.length = 0;
  }
});

```

```
function getResult(max, arr) {
  // 将商品按价格从小到大排序
  arr.sort((a, b) => a - b);

  let count = 0;

  let l = 0; // l指针指向最小价格的商品
  let r = arr.length - 1; // r指针指向最大价格的商品

  // 如果商品价格不超过上限，则优先最小价格和最大价格组合
  while (l < r) {
    const sum = arr[l] + arr[r];
    // 如果最小价格+最大价格 不超过上限，则组合，否则最大价格独立一组
    if (sum <= max) l++;
    r--;
    count++;
  }

  if (l === r) count++;

  return count;
}
```

Java算法源码

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

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int max = Integer.parseInt(sc.nextLine());
    Integer[] arr =
        Arrays.stream(sc.nextLine().split(" ")).map(Integer::parseInt).toArray(Integer[]::new);

    System.out.println(getResult(max, arr));
  }

  public static int getResult(int max, Integer[] arr) {
    // 将商品按价格从小到大排序
    Arrays.sort(arr);

    int count = 0;
    int l = 0; // l指针指向最小价格的商品
    int r = arr.length - 1; // r指针指向最大价格的商品

    // 如果商品价格不超过上限，则优先最小价格和最大价格组合
    while (l < r) {
      int sum = arr[l] + arr[r];

      // 如果最小价格+最大价格 不超过上限，则组合，否则最大价格独立一组
      if (sum <= max) l++;
      r--;
      count++;
    }

    if (l == r) count++;

    return count;
  }
}
```

Python算法源码

```python
# 输入获取
maxPrice = int(input())
prices = list(map(int, input().split()))


# 算法入口
def getResult():
    # 将商品按价格从小到大排序
    prices.sort()

    count = 0
    l = 0  # l指针指向最小价格的商品
    r = len(prices) - 1  # r指针指向最大价格的商品

    # 如果商品价格不超过上限，则优先最小价格和最大价格组合
    while l < r:
        sumPrice = prices[l] + prices[r]

        # 如果最小价格+最大价格 不超过上限，则组合，否则最大价格独立一组
        if sumPrice <= maxPrice:
            l += 1

        r -= 1
        count += 1

    if l == r:
        count += 1

    return count


# 算法调用
print(getResult())
```