题目描述

用数组代表每个人的能力 一个比赛活动要求参赛团队的最低能力值为N 每个团队可以由一人或者两人组成 且一个人只能参加一个团队 计算出最多可以派出多少只符合要求的队伍。

输入描述

- 第一行代表总人数，范围1-500000
- 第二行数组代表每个人的能力
  - 数组大小，范围1-500000
  - 元素取值，范围1-500000
- 第三行数值为团队要求的最低能力值，范围1-500000

输出描述

最多可以派出的团队数量

用例

输入	5 3 1 5 7 9 8
输出	3
说明	说明 3、5组成一队 1、7一队 9自己一队 输出3

输入	7 3 1 5 7 9 2 6 8
输出	4
说明	3、5组成一队，1、7一队，9自己一队，2、6一队，输出4

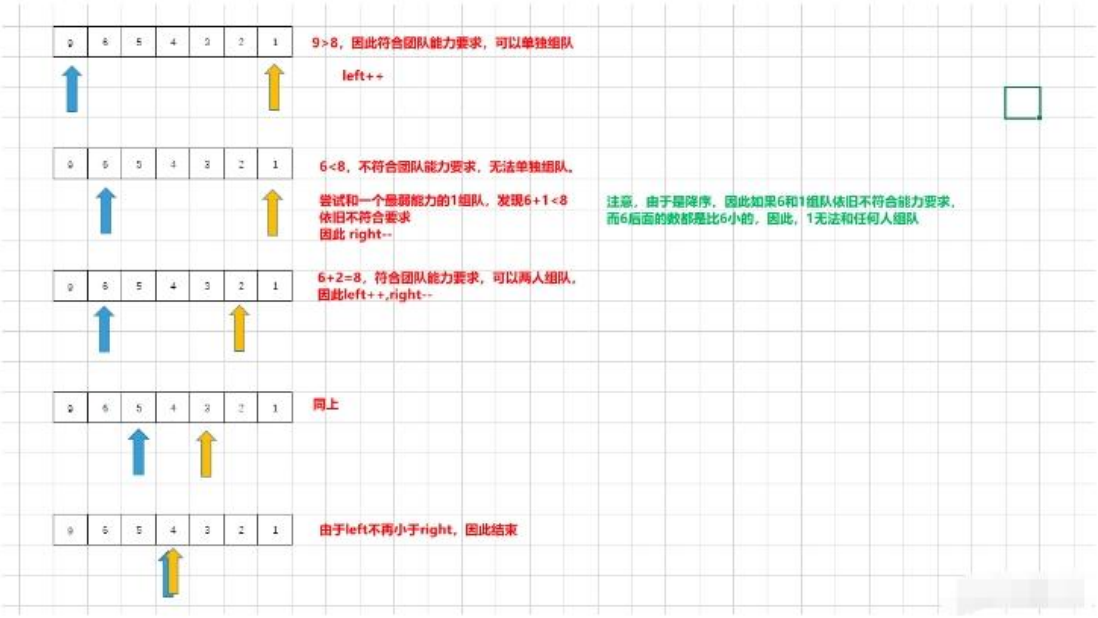
输入	3 1 1 9 8
输出	1
说明	9自己一队，输出1

题目解析

本题可以利用排序+双指针解决

首先我们将输入数列进行降序，比如3 1 5 4 9 2 6，就变为了9 6 5 4 3 2 1。

然后双指针运行逻辑如下



注意，我们需要处理边界情况，代表总人数可以只有1个，此时双指针left，right指向同一个元素，直接结束。

此时，我们需要特殊处理只有1个代表的情况，直接判断它的能力是否大于等于最低能力值，若是，则返回1，若不是则返回0

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 n = Integer.parseInt(sc.nextLine());

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

    int minCap = Integer.parseInt(sc.nextLine());

    System.out.println(getResult(n, capacities, minCap));
  }

  public static int getResult(int n, Integer[] capacities, int minCap) {
    if (capacities.length == 1) {
      return capacities[0] >= minCap ? 1 : 0;
    }

    Arrays.sort(capacities, (a, b) -> b - a);

```

```
    int l = 0;
    int r = capacities.length - 1;

    int maxTeamCount = 0;
    while (l < r) {
      if (capacities[l] >= minCap) {
        l++;
        maxTeamCount++;
      } else if (capacities[l] + capacities[r] >= minCap) {
        l++;
        r--;
        maxTeamCount++;
      } else {
        r--;
      }
    }

    return maxTeamCount;
  }
}
```

JS算法源码

```
/* 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 === 3) {
    let n = parseInt(lines[0]);
    let capacities = lines[1].split(" ").slice(0, n).map(Number);
    let minCap = parseInt(lines[2]);

    console.log(getMaxTeamCount(capacities, minCap));

    lines.length = 0;
  }
});

```

```
function getMaxTeamCount(capacities, minCap) {
  if (capacities.length === 1) {
    return capacities[0] >= minCap ? 1 : 0;
  }

  capacities.sort((a, b) => b - a);

  let left = 0;
  let right = capacities.length - 1;

  let maxTeamCount = 0;

  while (left < right) {
    if (capacities[left] >= minCap) {
      left++;
      maxTeamCount++;
    } else if (capacities[left] + capacities[right] >= minCap) {
      left++;
      right--;
      maxTeamCount++;
    } else {
      right--;
    }
  }

  return maxTeamCount;
}
```

Python算法源码

```python
# 输入获取
n = int(input())
capacities = list(map(int, input().split()))
minCap = int(input())


# 算法入口
def getResult():
    if len(capacities) == 1:
        return 1 if capacities[0] >= minCap else 0

    capacities.sort(reverse=True)

    l = 0
    r = len(capacities) - 1

    maxTeamCount = 0
    while l < r:
        if capacities[l] >= minCap:
            l += 1
            maxTeamCount += 1
        elif capacities[l] + capacities[r] >= minCap:
            l += 1
            r -= 1
            maxTeamCount += 1
        else:
            r -= 1

    return maxTeamCount


# 调用算法
print(getResult())
```