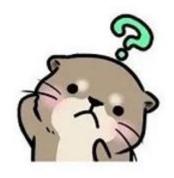
# C4-G 小水獭和蓝园

## 题目描述

小水獭想带走 n 个物品,每次可带走一个物品,或者两个重量之和不超过 m 的物品,问小水獭最少需要几次能带走全部物品。



#### 贪心算法:

如果某种取法一定不会使情况变得更坏,我们就采用这种方法。

关键:证明贪心选择的正确性

#### 贪心选择:

对于每次选择

- 当最大的元素和最小的元素的和大于 m 时, 我们只能单独取走最大的元素, ans + 1
- 否则, 我们将这个最大的元素和最小的元素捆绑, 一起取走, ans +1

#### 贪心的证明

如果某种取法不完全是按照我们构造的贪心取法取的,我们证明由它得到的解起码不会优于我们的贪心选择(下面情况都是 最大+最小 < m)

- 若最大的物品被单独拿走,最小的物品在之后跟着某个物品一起走;那么如果按照我们的贪心选择,取的次数不会增多
- 若最大的物品被单独拿走,最小的物品在之后也被单独拿走;按照我们的贪心选择,取的次数一定会减少
- 若最大的物品带着某个物品走,最小的物品也跟着某个物品走;按照我们的贪心选择,取的次数不会增多
- 若最大的物品带着某个物品走,最小的物品单独走;按照我们的贪心选择,取的次数不会增多

综上:构造的贪心的取法一定不会使结果变得更坏,所以贪心选择正确!

#### 另一种贪心的取法

最大的物品每次带着能带走的最大的物品走,即从小到大枚举,直到枚举到最后一个和最大的和大于 m 的物品,或者枚举完所有没取走的物品

通过上一页的证明,不难看出这种贪心的方法和以前构造的贪心的方法是一样优的,但是很明显这种贪心方法的代码更难打

相较于第一种贪心取法,第二种贪心取法的意义在本题中没有彰显

### 代码

```
1 ~ #include <stdio.h>
 2 #include <algorithm>
   using namespace std;
   const int maxn=1e5+5;
 6 int a[maxn];
8 ~ int main()
        int t,n,m;
        scanf("%d",&t);
12 ~
        while(t--)
13
14
            scanf("%d%d",&n,&m);
            for(int i=1;i<=n;i++) scanf("%d",&a[i]);</pre>
15
16
            sort(a+1,a+n+1);
17
            int i=1,j=n,ans=0;
18 ~
            while(i<=j)
19
                if((a[i]+a[j])<=m) i++;</pre>
20
21
                j--,ans++;
22
            printf("%d\n",ans);
24
25
        return 0;
```

#