信息学奥赛笔记02

暴力枚举课后习题 | 贪心算法

消消乐

题目描述

小Y最近在玩一款特殊的消消乐游戏，这个游戏是由若干个方块组成的，其中方块一共有`n`列，每一列的高度为a_i(高度可以为负），现在这个游戏有如下规则：

- 选中所有方块的列数中高度最低的那一列（如果有多个高度相同且最低，任选一个）对所有的列都消除a_i个方块。
- 被消除的那一列方块被清空，可以看成是被消除的那列方块将从数组中移除。
- 如果只剩`1`列，将结束操作。

现在小Y有多次操作的机会，求小Y能使得若干次消除后，方块列中剩下的方块中，使得最矮的那一列方块最多。

输入格式

第一行一个正整数n，表示列数。

第二行n个正整数，表示每一列初始方块的高度。

输出格式

一个整数，表示消除后最大的最小高度。

样例 #1

样例输入 #1

```
3
-1 2 0
```

样例输出 #1

```
2
```

样例 #2

样例输入 #2

```
5
3 2 -4 -2 0
```

样例输出 #2

```
2
```

提示

对于30%的数据，有 $1 \le n \le 1000, -10^6 \le a_i \le 10^6$，

对于100%的数据，有$1 \le n \le 2\times10^5, -10^9 \le a_i \le 10^9$。

30分做法（暴力枚举）

思路分析

> 本题的暴力枚举代码就是直接模拟即可，但是考虑到每次要取数组中的最小的数，我们可以设最小的数为a_0，次小的数为a_1，整个数组全部删去最小的数后，剩余的数变为$a_i - a_0$，数字与数字之间的差仍然不变，所以目前整个数组最小的数为$a_1 - a_0$。所以这道题想要通过暴力枚举来做，需要对原数组进行排序。

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n, ans = INT_MIN;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];
    sort(a.begin(), a.end());
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            a[j] -= a[i];
        }
        ans = max(ans, a[i + 1]);
    }
    cout << ans << endl;
    return 0;
}
```

100分做法

> 设数组为$[a_0, a_1, a_2 \dots a_{n-1}]$，并且是排序后的结果，将整个数组全部消去最小值$a_0$后，剩余数组变为$[a_1 - a_0, a_2 - a_0 \dots a_{n-1} - a_0]$，目前数组的最小值为$a_1 - a_0$。将目前数组的最小值$a_1 - a_0$消去后，整个数组变为$[a_2 - a_1，\ a_3 - a_1 \dots a_{n-1} - a_1]$，目前数组的最小值为$a_2 - a_1$。
>
> 将当前情况重复n次后，最终每次的最小值为$a_i - a_{i-1}$。所以我们只需要取排序后的数组相邻的值最大的结果即可。

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n;
```

```
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];
    sort(a.begin(), a.end());
    int mx = a[0];
    for (int i = 1; i < n; i++) {
        mx = max(mx, a[i] - a[i - 1]);
    }
    cout << mx << endl;
    return 0;
}
```

道路划分

题目描述

在一座小村上住着n个住民，政府兴修新的道路，由于这条道路会把村子一分为二，所以居民都产生了对这条道路的偏好，第i号住民的喜好值$a_i = 0$则表示他喜欢住在道路左边，否则$a_i = 1$则代表他喜欢住在道路的右边。

现在需要你对道路进行一次规划，我们认为，把路建在j处的含义是，将$a_0, a_1 \dots a_{j-1}$号居民划分在道路的左边。$a_j, a_{j+1} \dots a_{n-1}$号居民划分在道路的右边，我们认为道路的一侧是高满足度的一种划分指的是，获得满足的居民数量大于等于道路一侧居民的总数的一半，如道路的左侧有`3`个居民，其中有`2`个居民偏好于道路左侧，则我们认为道路的左侧是**高幸福度**的。现在需要你求出一种道路的划分位置，使得道路的两侧居民都是处于高幸福度的，如果有多个位置满足，则为了最终道路的美观，**输出一个距离中心位置(n / 2.0)** 最近的一个道路建设位置。

输入格式

第一行一个整数，表示居民的数量`n`。

第二行一个字符串，仅包含字符`0`和`1`，表示居民的偏好情况。

输出格式

一个整数，表示最佳划分位置。

如果需要将道路划分在第一个村庄的左侧，则输出0。

样例 #1

样例输入 #1

```
3
101
```

样例输出 #1

```
2
```

样例 #2

样例输入 #2

```
6
010111
```

样例输出 #2

```
3
```

提示

对于30%的数据，有$1 \leq n \leq 1000$。

对于100%的数据，有$1 \leq n \leq 10^6$。

30分做法(暴力枚举)

思路分析

> 我们可以枚举道路的位置，并且再统计一下道路左方幸福的人数是否达标，道路右方幸福的人数是否达标即可。

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n, ans = -1;
    string s;
    double mn = 100000;
    cin >> n >> s;
    for (int i = 0; i <= n; i++) {  // 枚举道路的位置
        int l = 0, r = 0;
        bool fl = i == 0 || 1.0 * l >= i / 2.0;        // 标记左边是否幸福
        bool fr = i == n || 1.0 * r >= (n - i) / 2.0; // 标记右边是否幸福
        if (fl && fr) {
            if (abs(n / 2.0 - i) < mn) {   // 如果当前位置更靠近中心点
                mn = abs(n / 2.0 - i);
                ans = i;  // 更新答案
            }
        }
    }
    cout << ans << endl;
    return 0;
}
```

100分做法(前缀和)

思路分析

> 设整个数组的和为$total$，设道路建设在位置i，道路的左边1的个数为sum，所以路的左边一共有i个人，其中sum个为1的人，有$i-sum$个0的人，这些人在路左侧是幸福的。道路i的右边有$n-i$个人，整个数组有$total$个为1的人，所以右边有$total-sum$个为1的人，那么每一次从左向右更新i的位置时，更新一下sum的值，就可以推导出道路左右两侧的情况了。这是利用了动态前缀和的思想。

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n, ans = -1, sum = 0, total = 0;
    string s;
    cin >> n >> s;
    for (int i = 0; i < n; i++) {
        total += s[i] - '0'; // 统计整个数组1的个数
    }
    for (int i = 0; i <= n; i++) {
        if (i - sum >= (i + 1) / 2 && total - sum >= (n - i + 1) / 2) {  // 如果左边幸福 && 右边幸福
            if (abs(n / 2.0 - i) < abs(n / 2.0 - ans)) ans = i;  // 如果答案更靠近中心点，则更新答案。
        }
        if (i < n) sum += s[i] - '0';
    }
    cout << ans << endl;
    return 0;
}
```

贪心算法

定义

> 在对一个问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，只做出在某种意义上的局部最优解。贪心算法不是对所有问题都能得到整体最优解，关键是贪心策略的选择，选择的贪心策略必须具备无后效性，即某个状态以前的过程不会影响以后的状态，只与当前状态有关。

贪心问题的求解步骤

- 建立数学模型来描述问题
- 把要求解的大问题分解为若干个子问题
- 对子问题逐个求解，得到局部最优解。
- 合并局部最优解获得全局最优解。

贪心问题常见的考点

贪心 + 排序

贪心 + 前缀和

贪心 + 二分

贪心 + 字符串

贪心 + 搜索

总结

贪心不太可以称之为它是一个算法，它是一种重要的思想，就好像人会本能的趋利避害，这也是一种贪心，想要最多的，最好的，这也是贪心，没有任何公式或者模板，可以说是该怎么去贪心，同学们只能通过做题来感受每一道题不同的**贪心策略**，敢于在考试中使用自己推导出的贪心策略来写题解题。

[NOIP2010 普及组] 接水问题

题目描述

学校里有一个水房，水房里一共装有 m 个龙头可供同学们打开水，每个龙头每秒钟的供水量相等，均为 1。

现在有 n 名同学准备接水，他们的初始接水顺序已经确定。将这些同学按接水顺序从 1 到 n 编号，i 号同学的接水量为 w_i。接水开始时，1 到 m 号同学各占一个水龙头，并同时打开水龙头接水。当其中某名同学 j 完成其接水量要求 w_j 后，下一名排队等候接水的同学 k 马上接替 j 同学的位置开始接水。这个换人的过程是瞬间完成的，且没有任何水的浪费。即 j 同学第 x 秒结束时完成接水，则 k 同学第 $x+1$ 秒立刻开始接水。若当前接水人数 n' 不足 m，则只有 n' 个龙头供水，其它 $m-n'$ 个龙头关闭。

现在给出 n 名同学的接水量，按照上述接水规则，问所有同学都接完水需要多少秒。

输入格式

第一行两个整数 n 和 m，用一个空格隔开，分别表示接水人数和龙头个数。

第二行 n 个整数 $w_1, w_2, \ldots, w_n$，每两个整数之间用一个空格隔开，w_i 表示 i 号同学的接水量。

输出格式

一个整数，表示接水所需的总时间。

样例 #1

样例输入 #1

```
5 3
4 4 1 2 1
```

样例输出 #1

```
4
```

样例 #2

样例输入 #2

```
8 4
23 71 87 32 70 93 80 76
```

样例输出 #2

```
163
```

提示

【输入输出样例 #1 说明】

第 1 秒，3 人接水。第 1 秒结束时，$1,2,3$ 号同学每人的已接水量为 $1,3$ 号同学接完水，4 号同学接替 3 号同学开始接水。

第 2 秒，3 人接水。第 2 秒结束时，$1,2$ 号同学每人的已接水量为 $2,4$ 号同学的已接水量为 1。

第 3 秒，3 人接水。第 3 秒结束时，$1,2$ 号同学每人的已接水量为 $3,4$ 号同学的已接水量为 2。4 号同学接完水，5 号同学接替 4 号同学开始接水。

第 4 秒，3 人接水。第 4 秒结束时，$1,2$ 号同学每人的已接水量为 $4,5$ 号同学的已接水量为 1。$1,2,5$ 号同学接完水，即所有人完成接水的总接水时间为 4 秒。

【数据范围】

$1 \le n \le 10^4$，$1 \le m \le 100$，$m \le n$；

$1 \le w_i \le 100$。

100分做法（贪心）

思路分析

> 为了保证总时间最短，这n个人来接水是有顺序的，所以只需要让所有水龙头尽可能的不浪费每一秒，换句话说就是让所有水龙头保持着开启状态，即可使得总时间最短，为了达成这一目的，我们等最早结束接水的同学走了的一瞬间，让下一个要接水的同学补到刚刚接完水的同学后面，在操作这个数组的时候，我们要统计当前水龙头的结束时间，那么两个人在交替的时候，被接力的水龙头的结束时间就从原本的a_i新增了现在新来的同学的$time_i$整体时间，就是这个水龙头的接水结束时间。

```cpp
#include <bits/stdc++.h>
using namespace std;
int a[10010], n, m;
int main() {
    cin >> n >> m;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
    for (int i = 1; i <= n - m; i++) {
        sort(a + 1, a + m + 1); // 将前 m 个水龙头排序
        a[1] += a[m + i]; // 最早结束的水龙头加上新来同学的接水时间成为这个水龙头的新结束时间
    }
```

```
    sort(a + 1, a + m + 1);  // 对所有水龙头最后一次排序
    cout << a[m] << endl;  // 输出最迟结束的水龙头就是答案
    return 0;
}
```