

第 8 套题目题解

T1 好朋友排排坐

我们假设三个人的位置有 $a < b < c$ 。

对于最小移动次数：

1. 如果 a, b, c 已经连续，答案显然为 0
2. 如果 a, b 或者 b, c 两个人有 1 个单位的间隔，那么答案为 1，例如 3, 4, 6 或 1, 14, 16
3. 其他情况，答案为 2。这种情况下，至少有两个人之间的空位大于等于 2，假设 b 和 c 之间的空格大于等于 2，我们可以将 a 移动到 $b + 2$ ， c 移动到 $b + 1$ 。

对于最大移动次数：

我们考虑 a, b 以及 b, c 之间的空位，每一次都会有一个人去占领空位，两个人之间的空格就会少一个，要想使得操作次数最大，肯定保证端点的一个人不动，另外两个人轮流占领中间的空格。最终答案就是 $\max(b - a, c - b) - 1$ 。

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int a, b, c;
    cin >> a >> b >> c;
    if(a > b) swap(a, b); // 冒泡排序
    if(b > c) swap(b, c);
    if(a > b) swap(a, b);
    if(c == a + 2) cout << 0 << endl;
    else if(b == a + 2 || c == b + 2) cout << 1 << endl;
    else cout << 2 << endl;
    cout << max(b - a, c - b) - 1;
    return 0;
}
```

T2 井字棋

本题，我们有一个井字格，我们要找出对于每一个字母或者每一对字母，是否这个字母或者这一对字母练成行，竖或者斜线。

因为范围很小，所以我们可以循环遍历每一个单独的字母，每一对字母。然后再循环行、列和对角线进行检查。

- 注意，这里再检查一对字母时，因为题目描述需要保证一队获胜时必须同时出现两个字母，所以我们还要检查不仅仅只出现了一个字母。

```

#include <bits/stdc++.h>
using namespace std;

char ch[4][4];

bool check1(char x) { // 判断x能够单独获胜
    for(int i = 1; i <= 3; ++i) { // 同一行或同一列
        if(ch[i][1] == x && ch[i][2] == x && ch[i][3] == x) return true;
        if(ch[1][i] == x && ch[2][i] == x && ch[3][i] == x) return true;
    }
    if(ch[1][1] == x && ch[2][2] == x && ch[3][3] == x) return true; //
    对角线
    if(ch[3][1] == x && ch[2][2] == x && ch[3][1] == x) return true;
    return false;
}

bool judge(char x, char y, char a, char b, char c) { // 判断(x,y)是否出现
    在a,b,c三个字符中
    // 三个字母必须是 (x,y) 其中一个
    if(a != x && a != y) return false;
    if(b != x && b != y) return false;
    if(c != x && c != y) return false;
    // (x, y) 至少出现一次
    if(a != x && b != x && c != x) return false;
    if(a != y && b != y && c != y) return false;
    return true;
}

bool check2(char x, char y) { // 判断 (x, y) 组队是否获胜
    for(int i = 1; i <= 3; ++i) { // 同一行或同一列
        if(judge(x, y, ch[1][i], ch[2][i], ch[3][i])) return true;
        if(judge(x, y, ch[i][1], ch[i][2], ch[i][3])) return true;
    }
    if(judge(x, y, ch[1][1], ch[2][2], ch[3][3])) return true; // 对角线
    if(judge(x, y, ch[1][3], ch[2][2], ch[3][1])) return true;
    return false;
}

```

```

int main() {
    for(int i = 1; i <= 3; ++i) {
        for(int j = 1; j <= 3; ++j) cin>>ch[i][j];
    }
    int ans1 = 0, ans2 = 0;
    for(char i = 'A'; i <= 'Z'; ++i) { // 单独获胜
        ans1 += check1(i);
    }
    for(char i = 'A'; i <= 'Z'; ++i) { // 两两组队获胜
        for(char j = i + 1; j <= 'Z'; ++j) {
            ans2 += check2(i, j);
        }
    }
    cout<<ans1<<endl<<ans2;
    return 0;
}

```

T3 老鹰捉小鸡

假设需要操作 k 次，那么只有前 k 个人的位置会发生改变，后面 $n - k$ 个人的位置不会变，所以这 $n - k$ 个人一定已经是排好序的。反之，如果有 $n - k$ 个人已经排好序了，在经过 k 次操作后，一定可以让整个序列有序。

那么，我们的问题就变成了，找到最长的单调递增的后缀，这可以在 $O(n)$ 时间内完成。

- 注意，当 $n - k$ 为 n ，也就是全部有序时，答案应该是 0。

```

#include <bits/stdc++.h>
using namespace std;

int main() {
    int n, a[105];
    cin>>n;
    for(int i = 1; i <= n; ++i) cin>>a[i];
    int ans = 0; // 整个序列都单调递增的话，答案为 0
    for(int i = n - 1; i >= 1; --i) {
        if(a[i] >= a[i + 1]) {
            ans = i;
            break;
        }
    }
    cout<<ans;
    return 0;
}

```

T4 选糖果

60 分做法

显然，每次选糖果肯定是贪心的从大到小选，那么我们对糖果从大到小排序，对于每一次询问，我们直接从大到小暴力枚举选择每一个糖果求和，知道和 $sum \geq x$ ，这就是答案。

时间复杂度为 $O(qn)$ 。

100分做法

我们选择糖果的时候，可以用一个前缀和 $sum[i]$ 记录前 i 个糖果的甜度总和，然后在这个前缀和数组中二分找到第一个大于等于 x 的位置，就是答案。

```

#include <bits/stdc++.h>
using namespace std;

const int maxn = 1e5 + 5;
int a[maxn], sum[maxn], n, q;

bool cmp(int x, int y) {
    return x > y;
}

int main() {
    cin>>n>>q;
    for(int i = 1; i <= n; ++i) cin>>a[i];
    sort(a + 1, a + 1 + n, cmp);
    for(int i = 1; i <= n; ++i) sum[i] = sum[i - 1] + a[i]; // 计算前缀和
    while(q--) {
        int x;
        cin>>x;
        if(sum[n] < x) cout<<"-1\n";
        else {
            int ans = lower_bound(sum + 1, sum + 1 + n, x) - sum; // 二分
            cout<<ans<<endl;
        }
    }
    return 0;
}

```