简况

比赛链接

AC 5题，Rank 26th

总结与反思

cmx

lpy

xsy

F题让输出id输出了序号C题加绝对值给自己加晕了导致白给WA

C题最开始的时候考虑到了改y但是咋就没想到改x弄了一个假的贪心乱WA

像个憨批。

题解

A.Ball

容易想到将给出的两个箱子当做一条边相连，于是形成了一个图。我们的目标是为每条边选择一侧点，并且点不会被重复选择。发现如果是一条l个点的链，那么就有l种选择，如果是一个基环树，那么就有2种选择，如果是其他形状，那么是不可能的，因为点数比边数还要少了，对应不起来。另外注意要额外判断单个点向自己连边的情况，这种虽然是环，但是种类是1。实现的时候善用并查集即可，维护点数和边数和是否含有单点环三个信息，没必要找环因为和环长无关。

（并查集开始居然忘了路径压缩，打自己）

B.Seal

C.Parade

题意：

给定$2N$个学生的坐标(互不相同)，他们最终要到$1\leq x \leq N, 1 \leq y \leq 2$这$2N$个位置上去，中

途不能有两个学生在一个格子上，问最少移动(曼哈顿)距离和。

题解：

仔细一想就知道两个学生能不能在一个格子上对答案根本没有影响，因为你一定能有一种方案让他们不重合。

那么我可以把$y>2$的学生先全部移到$(x,2)$把$y<1$的学生全部移到$(x,1)$这样一定不会使答案变劣，这一段距离时必走的。

同理也可以这样处理x坐标，处理完后所有学生的位置现在都在$1\leq x \leq N, 1 \leq y \leq 2$这个范围了，那么我们从左往右考虑：如果人多了就往右挪，不够就先在同一列找人(这样一定不劣，移动的总距离不会变多)，同一列不能够满足就再从后面列找人，在写代码时用负数表示还需要多少个学生，具体实现如下：

```
for (int i = 1; i <= n; ++i) {
    --num[i][1]; --num[i][2];
    if (1ll * num[i][1] * num[i][2] >= 0) {
        ans += abs(num[i][1] + num[i][2]);
        num[i + 1][1] += num[i][1];
        num[i + 1][2] += num[i][2];
    } else if (abs(num[i][1]) > abs(num[i][2])) {
        ans += abs(num[i][2]);
        num[i][1] += num[i][2];
        ans += abs(num[i][1]);
        num[i + 1][1] += num[i][1];
    } else {
        ans += abs(num[i][1]);
        num[i][2] += num[i][1];
        ans += abs(num[i][2]);
        num[i + 1][2] += num[i][2];
    }
}
```

D.Circuit

题意:

裸的FWT

题解:

generator和amplifier注意差分预处理,然后FWT得到receiver

最后预处理前缀和即可回答每组询问

by Hardict

E.Coprime

题意:

给定$\{a_{i}\}_{i=1}^{n}$,多组询问(l,r,x),求$\sum_{i=l}^{r}[gcd(a_{i},x)==1]$,强制在线

题解:

可以转变为求解$[1,r]$中满足的个数

考虑一个经典问题$1-n中与m互素的数的个数$,可以理题容斥解决

回到该题,可以知道判断素因子即可而且容斥利用的是$\mu(d)\neq 0$的数,针对多组询问,先预处理$1-1e5$每个数的约数中$\mu(d)\neq 0的d$

设$f[r][d]表示a_{1}\sim a_{r}中,\{i:d|a_{i}\}的集合大小$,即可针对每个询问,进行不超过约数个数$\sigma_{0}(x)\leq 128$次容斥即可

但$f[r][d]$实际转移量过大,注意到针对每个d,$f[r][d]$每次大小改变的r位置可知,且针对每个a_{r},至多有$\sigma_{0}(a_{r})$个d改变

针对每个d存储改变位置,查询时利用$upper_bound$即可得到对应的$f[r][d]$值,即可完成计算

时间复杂度为:全局预处理$O(1e5 log1e5)$,每组预处理$O(n\sigma_{0}(a_{i}))$,单次询问$O(\sigma_{0}(x)\sigma_{0}(a_{i}))$

by Hardict

F.Graduation

读题题，直接暴力枚举所有情况模拟即可。

by MountVoom

G.Go and Oreo

H.Homomorphism

J.Matrix of Determinants

K.Winner Winner, Chicken Dinner!

补题

I. Chamber of Braziers

很好的一道题，在比赛时居然死想没想出来TAT

考虑最终答案$[l,r]$里面，必然有一个1和一个最大值$t=r-l+1$，我们分t在1左边和右边两种情况讨论，第一种是第二种的对称。如果t在1右边的话，那么可以先找到1，然后枚举右端点r，顺便求这一段最大值mx，那么对于这个r，可能的l就已经唯一确定了——$r-mx+1$，剩下的工作实际上就是判断[l,r]之间，是否满足排序之后为1 2 3 4 5…

这个可以用对权值数组求字符串哈希来做。

但这里有个更方便有趣的方法，随机化+前缀异或。详见代码。如果我们需要比较的串的长度是一致，并且不考虑字符顺序的时候，可以使用这个技巧。

```
#include <bits/stdc++.h>

using namespace std;

const int maxn = 1000005;

int n, h[maxn], r[maxn], a[maxn], xo[maxn], ans;
char s[maxn];

void solve() {
    for (int i = 1; i <= n; i++)
        if (a[i] == 1) {
            int j, mx;
            ans = max(ans, 1);
            for (j = i + 1, mx = 1; j <= n && a[j] != 1; j++) {
                mx = max(mx, a[j]);
                if (j - mx >= 0 && (xo[j] ^ xo[j - mx]) == r[mx])
                    ans = max(ans, mx);
            }
            i = j - 1;
        }
}

int main() {
    int Tt;
    scanf("%d", &Tt);
    srand(time(0));
    for (int i = 1; i < maxn; i++)
        h[i] = RAND_MAX > maxn ? rand() : rand() * rand(), r[i] = r[i - 1] ^
h[i];
    while (Tt--) {
```

```
        scanf("%d", &n);
        for (int i = 1; i <= n; i++) {
            scanf("%d", &a[i]);
            xo[i] = xo[i - 1] ^ h[a[i]];
        }
        ans = 0;
        solve();
        reverse(a + 1, a + n + 1);
        for (int i = 1; i <= n; i++)
            xo[i] = xo[i - 1] ^ h[a[i]];
        solve();
        printf("%d\n", ans);
    }
    return 0;
}
```