2021.10.7-codeforce的C题及D题的训练

Problem - 1574D - Codeforces-The Strongest Build

题目大意：

首先给了n这个n代表的就是这个英雄有几个卡槽，然后我们可以对于每一个卡槽，我们会是给一个k代表这个卡槽里面有多少个选择，接下来n个卡槽都给搞完了之后，我们就是可以去输入一个m代表的是ban掉的序列

这个时候就是要我们来找这么一个问题就是让这些序列的值是最大的

实现思路：

首先假设我们一个ban掉的序列都没有的话，那么我们很容易得到这么一个事情，就是我们可以每一个都选择最大的，但是这时候我们发现我们有ban掉的序列，那么我么应该怎么办，我们可以拿一个map，把我们ban掉的序列存起来，然后我们枚举所有可能，如果找到第一个满足的没有被ban掉的序列，那么这个就是我们最后的最终的答案，然后我们怎么修改我们的答案呢，这里我们可以把我们这个每一行都给遍历一次，每次往前移动一个，首先我们得保证这个是没被ban掉的序列，这样的话我们判断m个其实这样的复杂度就是n*m的完全是可以接受的

代码实现：

```
/*
 * @Description: 电影和人生不一样，电影太仁慈了，人生太辛苦了
 * @Write Up: https://github.com/godhandsjoker/ACM-
 * @Author: godhands
 * @LastEditors: godhands
 * @LastEditTime: 2021-10-07 16:40:52
 */
#include <iostream>
#include <cstdio>
#include <cmath>
#include <algorithm>
#include <cstring>
```

```cpp
#include <vector>
#include <map>
using namespace std;

signed main()
{
#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin), freopen("out.txt", "w", stdout);
#endif
    ios::sync_with_stdio(false), cin.tie(nullptr),
cout.tie(nullptr);
    int n;
    while(cin >> n) {
        map<vector<int>, bool>mp;
        vector<vector<int>>a(n);
        vector<int> res;
        for (int i = 0; i < n; i++) {
            int c;
            cin >> c;
            a[i].resize(c);
            for (auto &it : a[i]) cin >> it;
            res.push_back(c - 1);
        }
        int m;
        cin >> m;
        for (int i = 0; i < m; i++) {
            vector<int> tmp;
            for (int j = 0; j < n; j++) {
                int x;
                cin >> x;
                tmp.push_back(x - 1);
            }
            mp[tmp] = true;
        }
        int ans = 0;
        if (mp[res]) {
            for (auto &it : mp) {
                int sum = 0;
                auto tmp = it.first;
                for (int i = 0; i < n; i++) sum += a[i][tmp[i]];
                if (sum <= ans) continue;
                for (int i = 0; i < n; i++) {
                    if (tmp[i]) {
```

```
                        tmp[i]--;
                        if (!mp[tmp]) if (sum + a[i][tmp[i]] - a[i]
[tmp[i] + 1] > ans) ans = sum + a[i][tmp[i]] - a[i][tmp[i] + 1],
res = tmp;
                        tmp[i]++;
                    }
                }
            }
        }
        for (auto &it : res) cout << it + 1 << " ";
        cout << '\n';
    }
    return 0;
}
```

Problem - 1574C - Codeforces-Slay the Dragon

题目大意：

首先你现在是有n个英雄，然后每一个英雄都是有一个自己的武力值，然后我们现在有m个boss，然后对于每一个boss我们会给定一个x和y，x是龙的防御力，然后y是龙的攻击力，我们要选择一个英雄来去把龙杀掉，剩下的要防御龙的攻击，然后你可以花费一个金币去把英雄的能力提高一点，然后提高的只在本回合有效，然后我们问杀掉每一只龙，最小的花费要是多少个金币

实现思路：

这个的话我们其实可以发现这么一个问题，其实我们现在的一个总和是一个定值，也就是我们每一次的花费是什么呢？就是（x - k），（y - sum + k）这两部分，其实我们可以发现这么一个问题，其实我们的x，y和sum每次都是一个定值，也就是说明我们要寻找的是这么一个问题，就是我们要寻找的是这个k的值，让这个式子的总和最小，然后如果我们把这两个式子分别来看，首先如果我们把这个英雄的能力值拍一个顺序的话，那么我们很容易其实就是可以发现，前一部分是单调递减的，后一部分是单调递增的，然后我们找到了这个的单调性，那么我们就是可以进行一个二分

代码实现：

```
/*
 * @Description: 电影和人生不一样，电影太仁慈了，人生太辛苦了
 * @Write Up: https://github.com/godhandsjoker/ACM-
```

```
 * @Author: godhands
 * @LastEditors: godhands
 * @LastEditTime: 2021-10-07 17:04:09
 */
#include <iostream>
#include <cstdio>
#include <cmath>
#include <algorithm>
#include <cstring>
#include <vector>
using namespace std;
#define int long long

signed main()
{
#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin), freopen("out.txt", "w", stdout);
#endif
    ios::sync_with_stdio(false), cin.tie(nullptr),
cout.tie(nullptr);
    int n, sum = 0;
    cin >> n;
    vector<int> a(n);
    for (auto &it : a) cin >> it, sum += it;
    sort(a.begin(), a.end());
    int T;
    cin >> T;
    while(T--) {
        int x, y;
        cin >> x >> y;
        int l = lower_bound(a.begin(), a.end(), x) - a.begin(), r =
l - 1;
        auto check = [](int x) -> int {
            return max(0ll, x);
        };
        cout << min(check(x - a[l]) + check(y - sum + a[l]),
(check(x - a[r]) + check(y - sum + a[r]))) << "\n";
    }
    return 0;
}
```

Problem - 1569D - Codeforces-Inconvenient Pairs

题目大意：

首先我们引入一个概念叫做曼哈顿距离，什么是曼哈顿距离呢，就是在一个笛卡尔坐标系下，我们两个点(x1, y1), (x2, y2)之间的曼哈顿距离就是|x1 - x2| + |y1 - y2|，然后我们首先是T组数据，然后接下来一个n，一个m，一个k，n和m分别代表的就是横纵的路，然后k代表的是有多少个人，然后我们输入一行代表的是n，一行代表的是m，然后输入k个点，这k个点就是我们k个人的坐标，最后我们要问的就是有多少个人不方便，什么样的两个人算是不方便的呢？意思就是如果两个人按照这个已经给定的路线走下去，如果他走过的这个路程要是大于我们刚才定义的曼哈顿距离的话，这两个人就是不方便的

实现思路：

~~其实我们画一下图，我们会发现这么一个有趣的事情，比如你看，如果他直接走，但是如果他没有往回走，然后不拐弯或者只拐一个弯，就是可以的，然后我们继续简化我们的思路，我们会发现其实出现问题的就是相邻的两条直线之间的点是不满足的点，但是只有这个点还是不够的，因为除了这个我们还有就是如果横着的相邻，然后如果他们在同一条竖着的直线上就是可以~~

好了，当我上面是废话，突然看到了一个特别妙的思路，比我的这个明白易懂很多，首先如果某个点正好落在了给定的横纵线的交点上面，那么我们会发现这么一个问题，就是这个点你去跟任何点都是曼哈顿距离，这个点你可以直接pass掉了，跟我们要找的答案没有任何关系，然后我们推断一下普遍的规律，假设我们一个点P落在了某条横线上面，并且是夹在了第i条喝第i + 1条的纵线之间，这时候我们会计算出来这么一个答案，就是所有夹在第i条和第i + 1条纵线之间的并且与点P不在同一个直线上面的都是可以记录进入答案的，然后我们就可以开辟几个map容器来夹出来这个答案

代码实现：

```
#include <iostream>
#include <cstdio>
#include <cmath>
#include <algorithm>
#include <cstring>
#include <vector>
#include <map>
using namespace std;
#define int long long

signed main()
```

```cpp
{
#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin), freopen("out.txt", "w", stdout);
#endif
    ios::sync_with_stdio(false), cin.tie(nullptr),
cout.tie(nullptr);
    int T;
    cin >> T;
    while(T--) {
        int x, y, k;
        cin >> x >> y >> k;
        vector<int> a(x), b(y);
        for (auto &it : a) cin >> it;
        for (auto &it : b) cin >> it;
        a.push_back(0x3f3f3f3f), b.push_back(0x3f3f3f3f);
        map<int, int> mpx, mpy;
        map<pair<int, int>, int> mpxy, mpyx;
        int res = 0;
        while(k--) {
            int xx, yy;
            cin >> xx >> yy;
            int posx = lower_bound(a.begin(), a.end(), xx) -
a.begin();
            int posy = lower_bound(b.begin(), b.end(), yy) -
b.begin();
            if (a[posx] == xx && b[posy] == yy) continue;
            if (a[posx] == xx) {
                res += mpy[posy] - mpxy[{posx, posy}], mpy[posy]++,
mpxy[{posx, posy}]++;
            } else {
                res += mpx[posx] - mpyx[{posy, posx}], mpx[posx]++,
mpyx[{posy, posx}]++;
            }
        }
        cout << res << '\n';
    }
    return 0;
}
```

Problem - 1572A - Codeforces-Book

题目大意：

首先是给你一个T组数据，然后给定一个n，代表的就是这本书里面一共有n章，然后接下来会有n行，每一行的第一个就是理解这章需要的章节数，接下来我们把需要阅读的章节读入进来，最后我们输出的就是如果可以理解书输出的就是你要阅读的一个次数，否则你输出的就是-1

实现思路：

在Codeforce给了两种做法，第一个是迭代nlogn的，然后第二个是建图DAG有向无环图，这个图论这里我还是不行，所以第二种方法更优他是n的复杂度，然后这里我只找到了第二种的做法，然后判断是否有环，然后重新学习了一下拓扑排序，因为这个可以对一张DAG（有向无环图）进行一个判环，然后我在这里学习了一下拓扑排序算法学习笔记(53): 拓扑排序

代码实现：

```
/*
 * @Description: 电影和人生不一样，电影太仁慈了，人生太辛苦了
 * @Write Up: https://github.com/godhandsjoker/ACM-
 * @Author: godhands
 * @LastEditors: godhands
 * @LastEditTime: 2021-10-08 17:54:17
 */
#include <iostream>
#include <cstdio>
#include <cmath>
#include <algorithm>
#include <cstring>
#include <vector>
#include <queue>
using namespace std;

signed main()
{
#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin), freopen("out.txt", "w", stdout);
#endif
    ios::sync_with_stdio(false), cin.tie(nullptr),
cout.tie(nullptr);
```

```
    int T;
    cin >> T;
    while(T--) {
        int n;
        cin >> n;
        vector<int> g[n + 1], in(n + 1);
        for (auto &it : in) it = 0;
        priority_queue<pair<int, int>, vector<pair<int, int>>,
greater<pair<int, int>>>p;
        for (int i = 1; i <= n; i++) {
            cin >> in[i];
            if (in[i] == 0) p.push({1, i});
            for (int j = 1; j <= in[i]; j++) {
                int tmp;
                cin >> tmp;
                g[tmp].push_back(i);
            }
        }
        int res = -1, cnt = 0;
        while(!p.empty()) {
            auto it = p.top();
            res = it.first;
            p.pop();
            cnt++;
            for (auto &ot : g[it.second]) {
                in[ot]--;
                if (in[ot] == 0) {
                    if (ot > it.second) p.push({it.first, ot});
                    else p.push({it.first + 1, ot});
                }
            }
        }
        if (cnt != n) res = -1;
        cout << res << "\n";
    }
    return 0;
}
```

Problem - 1592E - Codeforces-Bored Bakry

题目大意：

输入一个n，代表数组里面有n个数字，然后我们接下来输入n个数字，代表的就是这个数组里面的数字，然后我们要输出的就是好数组的最大的长度，这里给出好数组的一个定义，如果一段区间之内，他们相与的结果大于了这个区间他们相或的结果的话，那么我们就称这个区间为好数组区间，我们要找到的就是这个最长的好区间的一个长度

实现思路：

这题比较考验一个思维性，这里引用一下cf的一个题解，我们想这么一个问题，就是我们用And来表示从L到R的一个&和，而Xor来表示L到R的^的和，然后我们继续想，如果我们的这个区间的长度是一个奇数，那么我们可以确定的一个事情就是假设我们把这个区间内的所有的数字都用二进制来表示的话，我们就是可以发现这么的一个事情，就是他的每一位都是会出现奇数次，所以他一定不会是一个好的数组，因为And会是包含在了Xor里面，而如果这个区间的长度是一个偶数的话，那么他就是可以的，因为两个数异或的本身还是自己，而相与就是会有一个加法的进位，所以这里是我们可以明确的第一个点，然后接下来的点我引用一下知乎上的这个博主写的一个题解Codeforces Round #746 (Div. 2) A~E

- 当前位， $a_l, \dots a_r$ 中均为 1 。且区间含有偶数个元素

所以，要找寻的区间的长度一定是偶数。即满足 $2 \mid r-l+1$ 。

而 k 位之前的部分必须均取“等于”，即每一位 $and = xor = 0$ 。对于 k 位之前的部分设异或前缀和 $pref$ ，则位置 l 满足条件： $pref_{l-1} = pref_r$ 。

如果单纯的枚举合适的 l 需要 $O(n)$ 的时间，所以现在需要快速找到 $l-1$ 满足条件。

考虑到值域范围不大，我们采取下面的方式：

- 将 $pref_i = x$ 的位置 i 放到桶 $m[x]$ 中
- 对于右端点 r 对应的值 $pref_r$ ，在桶中查找合适位置。
- 更新答案。

代码实现：

这里这道题我想不太出来，参考了大佬的代码，这里放出来参考的大佬的博客的链接VP: Codeforces Round #746 (Div. 2) - Tony102's Blog这个讲的也是非常的好

然后又参考了另一位大佬的代码然后学了一下“平板电视”这个神奇的东西，真的是比STL还要STL，长见识了，这里放上链接CF1592E Bored Bakry

平板电视传送门

然后放上代码：

```
// 引用参考资料，请看我写的wp
// https://github.com/godhandsjoker/ACM-
#include <iostream>
#include <cstdio>
#include <bits/extc++.h>

using namespace std;
using namespace __gnu_pbds;

const int N = 1e6 + 10;
int a[N];
bool flag[N];

signed main() {
    ios::sync_with_stdio(false), cin.tie(nullptr),
cout.tie(nullptr);
    int n;
    while(cin >> n) {
        int res = 0;
        for (int i = 1; i <= n; i++) cin >> a[i];
        for (int x = 20; x >=0; x--) {
            memset(flag, 0, sizeof flag);
            for (int i = 1; i <= n; i++) if (a[i] & (1 << x))
flag[i] = true;
            int cnt = 0;
            gp_hash_table<int, int> odd, even;
            even[0] = -1;
            for (int i = 1; i <= n; i++) {
                if (!flag[i]){
                    cnt = 0;
                    odd.clear();
                    even.clear();
                    if (i & 1) odd[0] = i;
                    else even[0] = i;
                    continue;
                }
                cnt ^= (a[i] >> (x + 1));
                if (i & 1) {
                    if (!odd[cnt]) odd[cnt] = i;
```

```
                    else res = max(res, i - odd[cnt]);
                } else {
                    if (even[cnt] != -1 && !even[cnt]) even[cnt] =
i;
                    else res = max(res, even[cnt] == -1 ? i -
even[cnt] - 1 : i - even[cnt]);
                }
            }
        }
        cout << res << '\n';
    }
    return 0;
}
```

Problem - D - Codeforces-Hemose in ICPC ?

题目大意：

这个是个很经典标准的交互题，题目有点长难读，但是当我们看到这个上面写着最多不超过12次的时候我们就是应该会想要要二分，因为这样在这个数据范围内他才不会超过这个12次

给定一棵树，然后每条边上都有各自的一个权值，权值定义为的是两个点路径之间的一个gcd，我们每一次可以询问一个点集，然后返回任意两个点之间的最大值，然后让我们找到的就是最大的两点之间的一个距离

实现思路：

首先我参考了这个博主的视频，这场比赛的A~E这位博主都有更新Codeforces Round #746 div2我们第一次直接把所有点都给选择出来，然后找到一个最大值，然后一般的交互题就是三种思路，第一个二分，第二个分组分块，第三个二进制，然后这里引入了一个括号序和一个欧拉序，这里我们就引入了这个欧拉序，然后我们要找到最大的两个临边的话，我们就要把他转换成为一个一维的平面，然后讲On的复杂度降低到log级别，这就是我们引入的原理，欧拉序保证了两个点之间一定是有临边，然后我们直接二分查找就可以了

代码实现：

```
/*
 * @Description: 电影和人生不一样，电影太仁慈了，人生太辛苦了
 * @Write Up: https://github.com/godhandsjoker/ACM-
```

```
 * @Author: godhands
 * @LastEditors: godhands
 * @LastEditTime: 2021-10-08 16:17:49
 */
#include <iostream>
#include <cstdio>
#include <cmath>
#include <algorithm>
#include <cstring>
#include <vector>
#include <set>
using namespace std;

const int N = 1e3 + 10;
vector<int>s[N];
int cnt = 0, a[N << 1];
set<int> b;

int querry() {
    cout << "? " << b.size() << " ";
    for (auto &it : b) cout << it << " ";
    cout << endl;
    b.clear();
    int res;
    cin >> res;
    return res;
}

void dfs(int cur, int fa) {
    for (auto &it : s[cur]) {
        if (it == fa) continue;
        a[++cnt] = it;
        dfs(it, cur);
        a[++cnt] = cur;
    }
}

signed main()
{
#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin), freopen("out.txt", "w", stdout);
#endif
    int n;
```

```
    cin >> n;
    int x, y;
    for (int i = 1; i < n; i++) cin >> x >> y, s[x].push_back(y),
s[y].push_back(x);
    a[++cnt] = 1;
    dfs(1, 0);
    for (int i = 1; i <= n; i++) b.insert(i);
    int maxx = querry();
    int l = 1, r = cnt;
    while(l + 1 < r) {
        int mid = l + r >> 1;
        for (int i = 1; i <= mid; i++) b.insert(a[i]);
        if (querry() == maxx) r = mid;
        else l = mid;
    }
    cout << "! " << a[l] << " " << a[r] << endl;
    return 0;
}
```

Problem - 1592C - Codeforces-Bakry and Partitioning

题目大意：

首先是T组数据，然后给了一个n和一个k，n代表有n个点，每个点有相应的权值，然后让你删除最小一条边，最多k - 1条边，就是把他变成2到n个森林，然后你要使这几个森林的值相同的，然后问可不可以

实现思路：

其实就是异或的一个性质，我们可以想象这么一个问题，如果我们可以最后拆解成为六个，我们一定可以归为两个，如果可以拆解成为四个，那么我们一定可以归为两个，然后如果你能构造出来五个的话，那么你是一定可以构造出来三个的，这个就是一个异或的经典的例子，对于偶数来讲，我们要把他们分解成为两个部分，第一部分等于第二个部分的异或和，然后k大于等于三的时候我们就是要研究他是不是可以分解成为三部分，这三部分怎么去分，然后这三部分最后的异或值就是等于的最后的一个所有a[i]的异或和的值，然后这个题一开始一个炸空间MLE

代码实现：

```
/*
 * @Description: 电影和人生不一样，电影太仁慈了，人生太辛苦了
 * @Write Up: https://github.com/godhandsjoker/ACM-
 * @Author: godhands
 * @LastEditors: godhands
 * @LastEditTime: 2021-10-08 17:00:12
 */
#include <iostream>
#include <vector>

using namespace std;

const int N = 1e5 + 10;
int a[N], b[N], f[N], T, n, k, res, sum;
vector<int> tree[N];

void dfs(int u) {
    f[u] = 1;
    for (int i = 0; i < tree[u].size(); i++) {
        if (f[tree[u][i]] == 0) {
            dfs(tree[u][i]);
            if (b[tree[u][i]] != sum) b[u] = b[u] ^ b[tree[u][i]];
        }
    }
    if (b[u] == sum) res++;
}
signed main() {
    ios::sync_with_stdio(false), cin.tie(nullptr),
cout.tie(nullptr);
    cin >> T;
    while(T--) {
        res = 0, sum = 0;
        cin >> n >> k;
        for (int i = 1; i <= n; i++) {
            cin >> a[i];
            b[i] = a[i], tree[i].clear();
            sum = sum ^ a[i], f[i] = 0;
        }
        for (int i = 1; i < n; i++) {
            int u, v;
            cin >> u >> v;
```

```
            tree[u].push_back(v);
            tree[v].push_back(u);
        }
        if (sum == 0) cout << "YES\n";
        else if (k == 2) cout << "NO\n";
        else {
            dfs(1);
            res == 1 ? cout << "NO\n" : cout << "YES\n";
        }
    }
    return 0;
}
```