Codeforces Round #655 EF题解

E. Omkar and Last Floor

由于结果为每列1个数的平方，因此1需要尽可能多地集中到1列。

dp[l][r] 表示水平区间[l,r]内的最优解，如果一个片段一部分在[l,r]外则不计入结果。

dp[l][r] = max(dp[l, i-1] + dp[i+1, r] + 被i分割的片段数)

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

using namespace std;
int tem = 0;
int ll[200][200], rr[200][200];
int dp[200][200] = {};
int n;

int dfs(int l, int r) {
    if(l > r) return 0;
    if (dp[l][r] >= 0)return dp[l][r];
    int m = 0;
    for (int i = l; i <= r; ++i) {
        int tot = 0;
        for (int j = 0; j < n; ++j) {
            if (ll[j][i] >= l && rr[j][i] <= r) tot++;
        }
        m = max(m, tot * tot + dfs(l, i - 1) + dfs(i + 1, r));
    }
    return dp[l][r] = m;
}

int main() {
    int m;
    cin >> n >> m;
    for (int i = 0; i < n; ++i) {
        int k;
        scanf("%d", &k);
        for (int j = 0; j < k; ++j) {
            int l, r;
            scanf("%d%d", &l, &r);
            for (int i1 = l; i1 <= r; ++i1) {
                ll[i][i1] = l;
                rr[i][i1] = r;
            }
        }
    }
```

```
    for (int i = 0; i <= m; ++i) {
        for (int j = 0; j <= m; ++j) {
            dp[i][j] = -1;
        }
    }

    cout << dfs(1, m);
}
```

F. Omkar and Modes

显然需要用类似二分的方式询问，不过二分的次数为log k * k□log k可能会大于4

但是还有额外的信息可以利用：数列是递增的。相同的数字必然相邻。

对于一个询问的两个子区间，如果左子区间和父区间结果数字相同，且数字个数不同，则可以推断出该数字在父区间内的全部覆盖范围，搜索剩余片段即可。如果结果数字不同，则父区间询问结果与右区间完全相同，不再需要再次询问右区间。

另外，为了尽可能阻止出现左子区间和父区间结果完全相同，需要再次二分的情况，二分的中值设定为了父区间的数字最大重复个数。

因此大约每两次可以询问定位一个数字。

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

using namespace std;

int a[200005];
map<int, int> mp;

int xx = 0, ff = 0;

pair<int, int> chk(int l, int r, int prex, int pref) {
    if (r < l) return {-1, -1};
    if (!ff)printf("? %d %d\n", l, r);
    fflush(stdout);
    int x = xx, f = ff;

    if (!ff) {
        scanf("%d", &x);
        assert(x != -1);
        scanf("%d", &f);
    }

    xx = 0;
    ff = 0;

    if (f >= r - l + 1) {
```

```
        for (int i = l; i <= r; ++i) {
            a[i] = x;
        }
        if (!mp.count(x))mp[x] = 0;
        mp[x] += r - l + 1;
    } else if (x == prex) {
        for (int i = 0; i < f; ++i) {
            a[r] = x;
            r--;
            mp[x]++;
        }
        chk(l, r, 0, 0);
    } else {
        int oldx = mp.count(x) ? mp[x] : 0;
        chk(l, l + f - 1, x, f);
        int j = mp[x] - oldx;
        int i = l + f;
        if (!j) {
            xx = x;
            ff = f;
            chk(i, r, prex, pref);
        } else {
            int cnt = f - j;
            if (cnt < f) {
                if (!mp.count(x))mp[x] = 0;
                while (cnt-- > 0 && i <= r) {
                    a[i] = x;
                    mp[x]++;
                    i++;
                }
            }
            chk(i, r, prex, pref);
            return {x, f};
        }
    }
    return {x, f};
}

int main() {
    int n;
    cin >> n;
    chk(1, n, 0, 0);
    printf("! ");
    for (int i = 1; i <= n; ++i) {
        printf("%d ", a[i]);
    }
}
```