The 17th Chinese Northeast Collegiate Programming Contest

过题记录 AC (10/13)

星号（*）标出的是读过题面的题目

A	B	C	D	E	F	G	H	I	J	K	L	M
2/6	*		1/102	4/240	4/286	1/56	3/167	1/66	2/226	1/7	*	2/23

打的还可以，稍微中后期沉不住，罚时上来了。

题目分析及错误反思

D. Concrete Painting

题型：传统题

Tag：计数

考虑每个区间的贡献，对于每个区间如果被 k 段覆盖了，那么答案贡献可以得到是 $2^{n-k} \times (2^k - 1)$。

E. Triangle Pick

题型：传统题

Tag：三维计算几何（这是第一次遇到）

主要的就是要计算出射线和平面的交点，可以用参数方程和平面的方程求解出来。

Wa 2 的原因是 C 写的时候把读入写错了。

需要注意一下。

```
#include<bits/stdc++.h>
using namespace std;
const long double eps = 1e-9;
int n, m;
struct Tri{
    long double x1,y1,z1,x2,y2,z2,x3,y3,z3,A,B,C,D,S;
}tri[100005];
struct Vec{
    long double x,y,z;
};
int sgn(long double x) {
    if (fabs(x) < eps) {
        return 0;
```

```
    } else if (x > 0) {
        return 1;
    } else {
        return -1;
    }
}
long double get_Abs(Vec a){
    return sqrt(a.x * a.x + a.y * a.y + a.z * a.z);
}
Vec chaji(long double x1, long double y1, long double z1, long double x2, long
double y2, long double z2){
    return {y1 * z2 - y2 * z1, z1 * x2 - z2 * x1, x1 * y2 - x2 * y1};
}
long double get_S(long double x1, long double y1, long double z1, long double x2,
long double y2, long double z2){
    return get_Abs(chaji(x1, y1, z1, x2, y2, z2));
}
int main(){
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++){
        cin >>tri[i].x1 >> tri[i].y1 >> tri[i].z1 >> tri[i].x2 >> tri[i].y2 >>
tri[i].z2 >> tri[i].x3 >> tri[i].y3 >> tri[i].z3;
        Vec a = chaji(tri[i].x1 - tri[i].x2, tri[i].y1 - tri[i].y2, tri[i].z1 -
tri[i].z2, tri[i].x1 - tri[i].x3, tri[i].y1 - tri[i].y3, tri[i].z1 - tri[i].z3);
        tri[i].A = a.x;
        tri[i].B = a.y;
        tri[i].C = a.z;
        tri[i].D = -(tri[i].A * tri[i].x1 + tri[i].B * tri[i].y1 + tri[i].C *
tri[i].z1);
        tri[i].S = get_S(tri[i].x1 - tri[i].x2, tri[i].y1 - tri[i].y2, tri[i].z1 -
tri[i].z2, tri[i].x1 - tri[i].x3, tri[i].y1 - tri[i].y3, tri[i].z1 - tri[i].z3);

    }
    while (m--){
        long double a, b, c;
        cin >> a >> b >> c;
        int k = 0;
        long double ans = 1000000000000.0;
        for (int i = 1; i <= n; i++){
            if (sgn(tri[i].A * a + tri[i].B * b + tri[i].C * c) != 0){
                long double t = -tri[i].D / (tri[i].A * a + tri[i].B * b + tri[i].C
* c);
                long double x = a * t;
                long double y = b * t;
                long double z = c * t;
                long double S = get_S(x - tri[i].x1, y - tri[i].y1, z - tri[i].z1, x
- tri[i].x2, y - tri[i].y2, z - tri[i].z2) +
                get_S(x - tri[i].x2, y - tri[i].y2, z - tri[i].z2, x - tri[i].x3, y
- tri[i].y3, z - tri[i].z3) +
                get_S(x - tri[i].x3, y - tri[i].y3, z - tri[i].z3, x - tri[i].x1, y
- tri[i].y1, z - tri[i].z1);
                if (sgn(t) >= 0 && sgn(S - tri[i].S) == 0){
```

```
                    long double len = x * x + y * y + z * z;
                    if (len < ans){
                        ans = len;
                        k = i;
                    }
                }
            }
        }
        printf("%d\n", k);
    }
}
```

F. MPFT

题型：传统题

Tag：模拟，线段树

用线段树维护区间最小位置，同时单调队列维护T时间内cnt次数。

！！！注意卡常时间！！！

```
#include<bits/stdc++.h>
using namespace std;
struct node {
    int l, r;
    int min, pos;
} tree[4000005];
void up(int p) {
    tree[p].min = 1e9;
    if(tree[p * 2].min < tree[p].min) {
        tree[p].pos = tree[p * 2].pos;
        tree[p].min = tree[p * 2].min;
    }
    if(tree[p * 2 + 1].min < tree[p].min) {
        tree[p].pos = tree[p * 2 + 1].pos;
        tree[p].min = tree[p * 2 + 1].min;
    }
    if(tree[p * 2].min == 1e9 && tree[p * 2 + 1].min == 1e9) {
        tree[p].pos = -1;
    }
}
void build(int l, int r, int p) {
    tree[p].l = l;
    tree[p].r = r;
    tree[p].pos = -1;
    tree[p].min = 1e9;
    if(l == r) {
        return;
    }
```

```
    int mid = l + r >> 1;
    build(l , mid, p * 2);
    build(mid + 1, r, p * 2 + 1);
}
void update(int x, int d, int p) {
    if(tree[p].l == tree[p].r) {
        if(d == 0) {
            tree[p].min = 1e9;
            tree[p].pos = -1;
        } else {
            tree[p].min = d;
            tree[p].pos = tree[p].l;
        }
        return;
    }
    int mid = tree[p].l + tree[p].r >> 1;
    if(x <= mid)update(x, d, p * 2);
    else update(x, d, p * 2 + 1);
    up(p);
}
int P[1000005], T[1000005];
int cnt[1000005];
int net[1000005];
int pr[1000005];
int tot, num;
struct hc {
    int t, p;
} ans[2000005];
int read() {
    char ch = getchar();
    int s = 0;
    while (ch < '0' || ch > '9') ch = getchar();
    while (ch >= '0' && ch <= '9') {
        s = s * 10 + ch - 48;
        ch = getchar();
    }
    return s;
}
int main() {
    int n, m ,Tim, K;
    build(1, 1000000, 1);
    n = read();
    m = read();
    Tim = read();
    K = read();
    int j = 1;
    for (int i = 1, t, p; i <= m; i++) {
        T[i] = read();
        P[i] = read();
        while(T[i] - T[j] > Tim && j < i) {
            if(net[P[j]] && net[P[j]] <= T[j])cnt[P[j]]--;
            j++;
```

```
        }
        if(!net[P[i]]) {
            num++;
            net[P[i]] = T[i];
            cnt[P[i]] = 1;
            if(num > n) {
                int tmp = tree[1].pos;
                if(tmp != -1) {
                    num--;
                    ans[++tot] = hc {T[i], tmp};
                    net[tmp] = 0;
                    cnt[tmp] = 0;
                    update(tmp, 0, 1);
                }
            }
        } else {
            cnt[P[i]]++;
        }
        if(cnt[P[i]] >= K) {
            num--;
            ans[++tot] = hc {T[i], P[i]};
            update(P[i], 0, 1);
            net[P[i]] = 0;
            cnt[P[i]] = 0;
        } else {
            update(P[i], T[i], 1);
        }
    }
    int tot1 = 0;
    for (int i = 1; i <= 1000000; i++) {
        if(net[i])pr[++tot1] = i;
    }
    printf("%d %d\n", tot, tot1);
    for (int i = 1; i <= tot; i++) {
        printf("%d %d\n", ans[i].t, ans[i].p);
    }
    for (int i = 1; i <= tot1; i++) {
        printf("%d ", pr[i]);
    }
}
```

J. Less Time on the Road

题型：传统题

Tag：DP，Floyd 最短路

`f[i][j][k][t]` 表示的是当前处理到了第 i 个任务，并且第 i 个任务是由 $j(0 \to A/1 \to B)$ 完成的，另外一个人在 k 点，A 已经花费了 t 的时间时， B 所需要的最小时间。

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

using namespace std;

#define inf 0x3f3f3f3f

const int N = 80;

int n, m, q;
int d[N][N];
int dp[2][2][N][N * N + 1];
int qry[N + 1];

void chkmin(int& x, int y) {
    x = min(x, y);
}

void chkmax(int& x, int y) {
    x = max(x, y);
}

int main() {
    cin >> n >> m;
    memset(d, 0x3f, sizeof d);
    for (int i = 0; i < n; i++) {
        d[i][i] = 0;
    }
    for (int i = 0; i < m; i++) {
        int u, v;
        cin >> u >> v;
        u--, v--;
        d[u][v] = 1;
    }
    for (int k = 0; k < n; k++) {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (k != i && i != j && j != k) {
                    d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
                }
            }
        }
    }
    cin >> q;
    for (int i = 0; i < q; i++) {
        cin >> qry[i];
        qry[i]--;
    }
    memset(dp[0], inf, sizeof dp[0]);
    dp[0][0][0][d[0][qry[0]]] = 0;
    dp[0][1][0][0] = d[0][qry[0]];
    for (int cur = 0, i = 0; i < q - 1; i++) {
        int nxt = (cur ^ 1);
```

```
        memset(dp[nxt], inf, sizeof dp[nxt]);
        for (int u = 0; u < n; u++) {
            for (int ds = 0; ds < (i + 1) * N; ds++) {
                if (dp[cur][0][u][ds] != inf) {
                    chkmin(dp[nxt][0][u][ds + d[qry[i]][qry[i + 1]]], dp[cur][0][u]
[ds]);
                    chkmin(dp[nxt][1][qry[i]][ds], dp[cur][0][u][ds] + d[u][qry[i +
1]]);
                }
                if (dp[cur][1][u][ds] != inf) {
                    chkmin(dp[nxt][0][qry[i]][ds + d[u][qry[i + 1]]], dp[cur][1][u]
[ds]);
                    chkmin(dp[nxt][1][u][ds], dp[cur][1][u][ds] + d[qry[i]][qry[i +
1]]);
                }
            }
        }
        cur = nxt;
    }
    int ans = inf;
    int L = (q & 1) ^ 1;
    for (int j = 0; j < 2; j++) {
        for (int u = 0; u < n; u++) {
            for (int ds = 0; ds <= n * q; ds++) {
                chkmin(ans, max(ds, dp[L][j][u][ds]));
            }
        }
    }
    cout << ans << "\n";
    return 0;
}
```

G. Expected Sum

题型：传统题

Tag：计数

考虑每个数位的贡献，那就计算一下后面连续 k 个的不是 $+$ 的概率，加上下一个就是 $+$ 的概率，逆推就可以得到了。

赛后补题安排

补题安排

- H：none
- C：none
- Z：L

补题记录

问题及需要补的知识点

问题

读题准确性需要提高。

精度问题：特别是能改成乘法的除法。（两次G在这里了）

F 题卡常数，区间最小值返回下标，数据结构如果复杂度是比较极限。debug！！！特殊数据一定要记得处理

知识点

- 三维计算几何基础模板