20190817初露锋芒（二）题解

A.博弈-HDU2147

题意

有一个n*m大小的期盼，将一枚硬币放在棋盘右上角(1,m)处，每次一人使硬币向左、向下、向左下移动一格到空白位置，最后无法移动到人输。kiki先手，赢了输出Wonderful，输了输出What a pity。

分析

问题属博弈论中的组合博弈，使用P/N图分析法求解

> **组合博弈P/N图分析法**
>
> 必败点(P点) :前一个选手(Previous player)将取胜的位置称为必败点。
>
> 必胜点(N点) :下一个选手(Next player)将取胜的位置称为必胜点。
>
> 必败（必胜）点的属性：
>
> 1. 所有终结点是必败点（P点）；
> 2. 从任何必胜点（N点）操作，至少有一种方法可以进入必败点（P点）；
> 3. 无论如何操作， 从必败点（P点）都只能进入必胜点（N点）
>
> 由上面的属性得到该题的算法：
>
> - 步骤1:将所有终结位置标记为必败点（P点）；
> - 步骤2: 将所有一步操作能进入必败点（P点）的位置标记为必胜点（N点）
> - 步骤3:如果从某个点开始的所有一步操作都只能进入必胜点（N点），则将该点标记为必败点（P点）；
> - 步骤4: 如果在步骤3未能找到新的必败（P点），则算法终止；否则，返回到步骤2。
>
> **画图三规则**
>
> - 每个图的末状态均为必败点P
> - 所有能够一步到达必败点的都是必胜点N
> - 所有能够一步到达必胜点的都是必败点 P

由上述规则，画出7*7的PN图：

P	N	P	N	P	N	P
N	N	N	N	N	N	N
P	N	P	N	P	N	P
N	N	N	N	N	N	N
P	N	P	N	P	N	P
N	N	N	N	N	N	N
P	N	P	N	P	N	P

因为从右上角(1,m)点出发，由此可见，当n、m都为奇数的点是必败点，其他都为必胜点。

代码

```
#include <iostream>
using namespace std;

int main() {
    int n, m;
    while(cin >> n >> m) {
        if (n == 0 || m == 0) break;
        if(n % 2 == 0 || m % 2 == 0) {
            cout << "Wonderful!\n";
        }
        else
            cout << "What a pity!\n";
    }
    return 0;
}
```

B.贪心-HDU2037

题意

给出节目总数和开始时间结束时间，求最多可以看完多少个完整的节目

分析

贪心法，按照节目结束时间从先到后排列即可，从第一个开始看，然后判断前一个看的结束后能否赶上看当前遍历到的这个节目，如果能则计数并更新看完后的时间，直到遍历完所有的节目，输出结果。

代码

```
#include <iostream>
```

```cpp
#include <algorithm>
using namespace std;

struct Tv{
    int s, e;
};

Tv tv[110];

bool cmp(Tv a, Tv b) { return a.e < b.e; }

int main() {
    int n;
    while(cin >> n) {
        if (n == 0) continue;
        for (int i = 0; i < n; ++i) {
            cin >> tv[i].s >> tv[i].e;
        }
        sort(tv, tv + n, cmp);
        int ans = 1, pre = tv[0].e;
        for (int i = 1; i < n; ++i) {
            if (tv[i].s >= pre) {
                ans++;
                pre = tv[i].e;
            }
        }
        cout << ans << endl;
    }
    return 0;
}
```

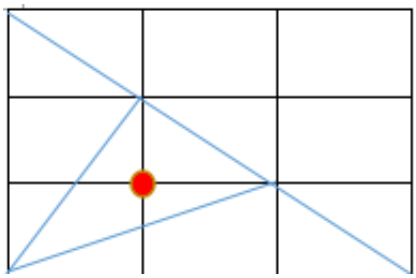
C.位运算-HDU5666

题意

给出q与P，判断直线x+y=q这条直线与x、y轴正方向围成的三角形内的坐标是整数的点，且原点与直线上整数点连接起来后不在所有连线上的点的个数，输出余P的结果。

分析

因为q题中已经说明一定是素数，所以通过画图不难发现：



q为素数时，在不连线的状况下所有的整数点即使连线后也不在线上，所以这一约束可以忽略。

可推出规律，在q>=3的时候，三角形内共有(q-1)*(q-2)/2个点，此式即为答案。但q范围为10^18，会爆long long，所以需要使用快速幂模版，修改为快速乘，输出mod P的结果即可。

代码

```
#include <iostream>
#include <algorithm>
typedef long long LL;
using namespace std;

LL qpow(LL a, LL n, LL m) {
    LL ans = 0;
    while(n) {
        if(n & 1) {
            ans = (ans + a) % m;
        }
        a = (a + a) % m;
        n >>= 1;
    }
    return ans;
}

int main() {
    LL q, p, t;
    cin >> t;
    while(t--) {
        cin >> q >> p;
        LL a = q - 2, b = q - 1;
        if(q % 2 == 0) a/= 2;
        else b /= 2;
        cout << qpow(a, b, p) << endl;
    }
    return 0;
}
```

D.并查集-HDU1233

题意

杭电畅通工程系列题，要求计算修公路的最小成本

分析

由题可见为最小生成树模版题，将成本存入用来存储道路信息的结构体作为边的权值。每个结构体中存储这段路的起点、终点以及成本。使用Kruskal最小生成树算法，将道路按权值从小到大升序排列，不成环即加入，并计算权值即可。

代码

```
#include <algorithm>
#include <iostream>
using namespace std;

int pre[110];

int find(int key) {
    if (pre[key] == key) return key;
    return pre[key] = find(pre[key]);
}

void init(int n) {
    for (int i = 0; i <= n; ++i) {
        pre[i] = i;
    }
}

struct Road {
    int s, e, cost;
    bool operator<(const Road &a) const { return cost < a.cost; }
};

Road r[6010];

int main() {
    ios::sync_with_stdio(false);
    int n;
    while(cin >> n && n) {
        int tmp = n * (n - 1) / 2;
        for(int i = 0; i < tmp; ++i) {
            cin >> r[i].s >> r[i].e >> r[i].cost;
        }
        sort(r, r + tmp);
        init(n);
        int ans = 0;
        for(int i = 0; i < tmp; ++i) {
            int ra = find(r[i].s);
            int rb = find(r[i].e);
            if(ra != rb) {
```

```
                pre[ra] = rb;
                ans += r[i].cost;
            }
        }
        cout << ans << endl;
    }
    return 0;
}
```

E.最小生成树-HDU4786

题意

给出每条边的起点终点以及其颜色（u、v、c），0代表黑色1代表白色。求问是否存在白色边数是斐波那契数的生成树。

分析

由题意可知构造出的生成树最后的权值之和即为白色边的数量，所以先按权值升序排列，使用Kruskal算法求出最小生成树权值和，再降序排列，求出最大生成树权值和，打出斐波那契数表，并判断在这两个数之间是否存在斐波那契数即可。因为在任何状态下，都可以使用一条白边替换黑边，所以最小值和最大值之间的白边数都可以达到。尤其注意的是，应先判断这棵树是否能够联通，如果不能联通应输出No。

代码

```
#include <iostream>
#include <algorithm>
using namespace std;

int pre[100010];
int fib[100];

int find(int key) {
    if(pre[key] == key) return key;
    return pre[key] = find(pre[key]);
}

void init(int n) {
    for(int i = 0; i <= n; ++i) pre[i] = i;
    return ;
}

void fib_init() {
    fib[1] = 1;
    fib[2] = 2;
    for(int i = 3; fib[i] <= 100000; ++i) {
        fib[i] = fib[i - 1] + fib[i - 2];
```

```
    }
    return ;
}

struct Edge {
    int u, v, c;
};

void merge(int x, int y) {
    int rx = find(x);
    int ry = find(y);
    if(rx != ry) {
        pre[rx] = ry;
    }
    return ;
}

Edge edge[100010];

bool cmp_low(Edge a, Edge b) {return a.c < b.c;}
bool cmp_high(Edge a, Edge b) {return a.c > b.c;}


int main() {
    ios::sync_with_stdio(false);
    int t, kase = 1;
    cin >> t;
    fib_init();
    while(t--) {
        int n, m;
        cin >> n >> m;
        init(n);
        for(int i = 0; i < m; ++i) {
            cin >> edge[i].u >> edge[i].v >> edge[i].c;
            merge(edge[i].u, edge[i].v);
        }

        int cnt = 0;
        for(int i = 1; i <= n; ++i) {
            if(pre[i] == i)cnt++;
        }

        cout << "Case #" << kase++ << ": ";
        if(cnt > 1) {
            cout << "No\n";
            continue;
        }
        int low = 0, high = 0;
        sort(edge, edge + m, cmp_low);
        init(n);
        for(int i = 0; i < m; ++i) {
            int ru = find(edge[i].u);
```

```
            int rv = find(edge[i].v);
            if(ru == rv) continue;
            low += edge[i].c;
            merge(edge[i].u, edge[i].v);
        }

        sort(edge, edge + m, cmp_high);
        init(n);
        for(int i = 0; i < m; ++i) {
            int ru = find(edge[i].u);
            int rv = find(edge[i].v);
            if(ru == rv) continue;
            high += edge[i].c;
            merge(edge[i].u, edge[i].v);
        }

        bool ans = false;
        for(int i = 1; fib[i] <= 100000; ++i) {
            if(fib[i] >= low && fib[i] <= high) {
                ans = true;
                break;
            }
        }
        if(ans) cout << "Yes\n";
        else cout << "No\n";
    }
    return 0;
}
```

F.模拟-HDU2525

题意

初始有n个刚克隆出来的克隆人，每个刚克隆出来的克隆人在克隆完后的a天中，每天都可以获得一份克隆材料，每个克隆人需要培养k天才能完成克隆，第k+1天开始造成伤害，每个克隆人可以存活d天，克隆成功后每天造成5点伤害，问第x天共造成了多少伤害

分析

题目中有几个需要注意的点：

1. 存活的天数d包括了还未被成功克隆的天数，也就是如果d=4，克隆需要2天，那么克隆成功后只能再存活2天。
2. 培养的第k天无法造成伤害，第k+1天才能开始攻击

也就是，今天（第i天）刚产生战斗力的克隆人，它所产生的“种子”将在第i + k + 1天开始产生战斗力，并且它将持续产生a天的种子。

所以我们只需用数组模拟每天克隆成功了多少人，再算出每天存活的人数，最后乘5加和即可算出最终结果。

代码

```
#include <cstring>
#include <iostream>
using namespace std;
int main() {
    long long live[1000], grow[1000];
    int t, n, d, a, k, x;
    cin >> t;
    while (t--) {
        memset(live, 0, sizeof(live));
        memset(grow, 0, sizeof(grow));
        cin >> n >> d >> a >> k >> x;
        grow[0] = n;
        long long ans = 0;
        for (int i = 0; i < x; i++) {
            for (int j = i; j < i + a; j++) {
                grow[j + k + 1] += grow[i];
            }
            for (int j = i; j < i + d; j++) {
                live[j] += grow[i];
            }
            ans += 5 * live[i];
        }
        cout << ans << endl;
    }
    return 0;
}
```

G.栈-HDU1509

题意

模拟消息队列

输入有两种指令，一种为GET，获取当前队列中最前的消息输出并弹出；一种为PUT + 消息名 + 参数 + 优先级，如果优先级相同则先进先出（FIFO），优先级数字越小优先级越高，模拟这个过程。

分析

使用STL库的优先队列，将信息的优先级、参数与信息名存入结构体，并重载小于号使优先级数字低的在保持在队首。然后检测输入，如果是PUT，则读入信息，插入队列；如果是GET，则获取队首打印信息，并弹出队列。

代码

```cpp
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;

struct Message {
    string name;
    int para, lev, id;

    bool operator < (const Message &a) const {
        if (lev == a.lev) return id > a.id;
        return lev > a.lev;
    }
};

priority_queue <Message> q;

Message msg[60010];

int main() {
    string op;
    int i = 0;
    while (cin >> op) {
        if(op == "PUT") {
            cin >> msg[i].name >> msg[i].para >> msg[i].lev;
            msg[i].id = i;
            q.push(msg[i++]);
        }
        else if(op == "GET") {
            if(q.empty()) {
                cout << "EMPTY QUEUE!\n";
                continue;
            }
            else {
                cout << q.top().name << ' ' << q.top().para << endl;
                q.pop();
            }
        }
    }
    return 0;
}
```

H.线性筛-HDU1788

题意

“现在有一个问题是这样的：”之前的内容全是废话，因为看前面的内容浪费了好久时间Orz

一个正整数N除以M1余（M1-a），除M2余（M2-a）.....除Mi余（Mi-a），求满足条件的最小的N。

输入i（M的个数）和a，然后输入M1～Mi。

分析

由题易知，N % Mi = Mi - a，则（N + a） % Mi == 0。

即n + a是所有M的最小公倍数，最终结果减去a即可。

代码

```
#include <iostream>
#include <algorithm>
using namespace std;
typedef long long LL;

LL gcd(LL a, LL b) {
    if(!b) return a;
    return gcd(b, a % b);
}

LL lcm(LL a, LL b) {
    return a / gcd(a, b) * b;
}

int main() {
    int I, a, m;
    while(cin >> I >> a && (I || a)) {
        LL ans = 1;
        for(int i = 0; i < I; ++i) {
            cin >> m;
            ans = lcm(ans, m);
        }
        cout << ans - a << endl;
    }
    return 0;
}
```