ACM/ICPC Template

AsZ VincentLDL

October 12, 2015

Contents

1 java TODO **2**
1.1 读写 2
1.2 高精 2

2 dp 优化 **3**
2.1 决策单调性优化 3
2.2 单调队列优化以及写仙人掌图 4
2.3 斜率优化 6
2.3.1 斜率以及 x 维都单调 6
2.3.2 随便什么情况：cdq 分治优化 7

3 图论 **9**
3.1 tarjan TODO 9
3.1.1 2-sat 9
3.1.2 割顶，点双联通分量 TODO 11
3.1.3 桥，边双联通分量 TODO 11
3.2 平面图 TODO 11
3.3 最佳追捕算法 11
3.4 网络流 TODO 11
3.4.1 dinic 11
3.4.2 费用流 TODO 13
3.4.3 常见模型 TODO 13
3.5 弦图 13
3.5.1 做法与常见问题 13
3.5.2 万不得已用线性作法 14
3.5.3 nlogn 好写得多 15
3.6 最小树形图 16
3.7 二分图 17
3.7.1 普通 KM 17
3.7.2 牛逼 KM 18
3.7.3 常见问题汇总 19
3.7.4 最小点覆盖输出方案 20
3.8 带花树 21
3.8.1 普通图最大匹配 21
3.8.2 普通图最优匹配 22
3.9 最大团 TODO 27
3.10 欧拉理论 TODO 27

4 数据结构 **28**
4.1 左偏树 TODO 28
4.2 splay TODO 28
4.3 lct TODO 28
4.4 可持久化线段树以及 LCA 不能再写错了!!! 28
4.5 点分治 29
4.6 树链剖分 TODO 32
4.7 qtree TODO 32

5 其他算法 **33**
5.1 pq 树 TODO 33
5.2 DLX TODO 33
5.3 对抗搜索 TODO 33
5.4 cdq 分治与读入优化 33

Chapter 1

java

```java
import java.io.*;
import java.math.*;
import java.util.*;
import java.text.*;


class point
{
    int A, B;
    double C;
    public point(int a, int b)
    {
        this.A = a; this.B = b;
        if (b == 0) this.C = 1e20;
        else this.C = 1.0 * a / b;
    }
};

public class Main
{
    public static int N, V;
    public static int[] A = new int[44];
    public static int[] B = new int[44];
    public static double[] C = new double[44];
    public static final int MAXN = 44;
    public static point[] g = new point[MAXN];

    public static void main(String[] args)
    {

        Comparator<point> comparator = new Comparator<point>(){
               public int compare(point s1, point s2) {
                   return (s1.C < s2.C) ? -1 : 1;
               }
        };

        Scanner cin = new Scanner(new BufferedInputStream(System.in));
  PrintWriter out = new PrintWriter(new OutputStreamWriter(System.out));
        int task = cin.nextInt();
        for (; task > 0; --task)
        {
            N = cin.nextInt();
            V = cin.nextInt();
            for (int i = 0; i < N; ++i)
                A[i] = cin.nextInt();
            int flag = 1;
            for (int i = 0; i < N; ++i)
            {
                B[i] = cin.nextInt();
                if (B[i] != 0 && A[i] >= V)
                    flag = 0;
            }
            if (flag == 0)
            {
                out.println(-1);
                continue;
            }
            for (int i = 0; i < N; ++i)
                g[i] = new point(A[i], B[i]);
            Arrays.sort(g, 0, N, comparator);
            BigDecimal ans = BigDecimal.ZERO;
            for (int i = N - 1; i >= 0; --i)
            {
                if (g[i].B == 0) continue;
                BigDecimal tmp = ans.multiply(BigDecimal.valueOf(1.0 * g[i].A));
                tmp = tmp.add(BigDecimal.valueOf(1.0 * g[i].B));
                //System.out.println(tmp + " " + (1.0 * V - g[i].A));
                tmp = tmp.divide(BigDecimal.valueOf(1.0 * V - g[i].A), 1000, BigDecimal.ROUND_HALF_UP);
                ans = ans.add(tmp);
            }
            out.println(ans.setScale(0, BigDecimal.ROUND_HALF_UP)); //保留0位小数
        //    Arrays.sort
        }
      out.flush();    }
}
```

Listing 1.1: Main.java

Chapter 2

dp 优化

2.1 决策单调性优化

- 形式：$f[i] = f[j] + w[j, i]$ 形式决策单调。
- 一般打表找规律看决策是否单调。
- 四边形不等式：$w[i,j]+w[i+1, j+1] <= w[i+1, j] + w[i+1, j]$, 则满足决策单调性。
- 有时候不满足决策单调性，但是去掉完全不合法状态之后却可以满足。

```cpp
#include <bits/stdc++.h>
#define MAXN 51234

using namespace std;
typedef long long arrayN[MAXN];

deque < pair< pair<int, int> , int> > deq;
arrayN f, sum, c;
long long L;

long long sqr(long long x)
{
    return x * x;
}

long long trans(int l, int r)
{
    return sqr(1LL * r - (l + 1) - L + sum[r] - sum[l]) + f[l];
}
int myLowBound(pair <int, int> pr, int ori, int now)
{
    int l = pr.first, r = pr.second;
    for (; l < r; )
    {
        int mid = l + r >> 1;
        if (trans(ori, mid) <= trans(now, mid)) l = mid + 1;
        else r = mid;
    }
    return l;
}

int main()
{
    int n;
    freopen("toys.in", "r", stdin);
    cin >> n >> L;
    for (int i = 1; i <= n; ++i)
    {
        cin >> c[i];
        sum[i] = sum[i - 1] + c[i];
    }
    deq.push_back(make_pair(make_pair(1, n), 0));
    for (int i = 1; i <= n; ++i)
    {
        for (; deq.front().first.second < i; deq.pop_front());
        f[i] = trans(deq.front().second, i);
        if (i == n) break;
        deq.front().first.first = i + 1;
        if (deq.front().first.second < i + 1) deq.pop_front();
        for (;!deq.empty() && trans(deq.back().second, deq.back().first.first) >= trans(i, deq.back().first.first); deq.pop_back());
        if (deq.empty()) deq.push_back(make_pair(make_pair(i + 1, n), i));

        else
        {
            int x = myLowBound(deq.back().first, deq.back().second, i);
            if (trans(i, x) >= (trans(deq.back().second, x))) x++;
            deq.back().first.second = x - 1;
            if (x <= n) deq.push_back(make_pair(make_pair(x, n), i));
        }
    }
    cout << f[n] << endl;
    return 0;
}
```

Listing 2.1: hnoi2008toys.cpp

2.2 单调队列优化以及写仙人掌图

- 题目背景：仙人掌图上最长链
- 形式：f[i] = max(g[j]) + w[i]，w[i] 单调, 可见，如果 j<k, g[j]<g[k], 则 j 可以直接不考虑，所以此时维护 g 单调减的队列即可。
- 仙人掌图找环：首先形成 bfs 树，发现有环，记 pt，ph，然后选 pt 沿着 pre 走到跟，一路打时间戳；再从 ph 沿着 pre 走，就可以找到 lca。pt，ph 到 lca 的路径，加上 pt–>ph 就是基环了。

```
#include <bits/stdc++.h>
#define MAXN 1123456
#define MAXM 2123456

typedef int arrayN[MAXN], arrayM[MAXM];

using namespace std;

arrayN fir, cost, t, pre, vis;
arrayM e, nxt, c;
long long ans, dst[MAXN];
int num, now, visNow;

void link(int u, int v, int w)
{
    e[++num] = v, nxt[num] = fir[u];
    fir[u] = num, c[num] = w;
}

vector <int> bfsFindCycle(int x)
{
    ++now;
    vector <int> cyc;
    deque <int> deq;
    int pt = 0, ph = 0, last;
    deq.push_back(x);
    t[x] = now;
    for (; !deq.empty() && !pt;)
    {
        int u = deq.front();
        deq.pop_front();
        for (int p = fir[u]; p && !pt; p = nxt[p])
            if (e[p] != pre[u])
                if (t[e[p]] == now)
                {
                    pt = u, ph = e[p];
                    last = c[p];
                }
                else
                {
                    t[e[p]] = now;
                    pre[e[p]] = u;
                    cost[e[p]] = c[p];
                    deq.push_back(e[p]);
                }
    }
    vector <int> cycTmp;
    if (pt)
    {
        ++now;
        int tmp = pt;
        for (; tmp != x; tmp = pre[tmp])
            t[tmp] = now;
        t[x] = now;
        int lca = ph;
        for (; t[lca] != now; lca = pre[lca]);
        for (tmp = pt; tmp != lca; tmp = pre[tmp])
        {
            swap(last, cost[tmp]);
            cyc.push_back(tmp);
        }
        cyc.push_back(lca);
        cost[lca] = last;

        for (tmp = ph; tmp != lca; tmp = pre[tmp])
            cycTmp.push_back(tmp);
        for (; !cycTmp.empty(); cycTmp.pop_back())
            cyc.push_back(cycTmp.back());
    } else cyc.push_back(x);

    ++now;
    for (int i = 0; i < cyc.size(); ++i)
        t[cyc[i]] = now;
    return cyc;
```

```cpp
}

struct node
{
    long long w;
    long long lst, f;}g[MAXN * 2];

long long bfsLongest(int rt, int &nrt)
{
    long long lst = 0;
    deque <int> deq;
    deq.push_back(rt);
    nrt = rt;
    vis[rt] = ++visNow;
    dst[rt] = 0;
    for (; !deq.empty(); )
    {
        int u = deq.front();
        deq.pop_front();
        for (int p = fir[u]; p; p = nxt[p])
            if (vis[e[p]] != visNow && t[e[p]] != now)
            {
                vis[e[p]] = visNow;
                dst[e[p]] = dst[u] + c[p];
                if (dst[e[p]] > lst)
                {
                    lst = dst[e[p]];
                    nrt = e[p];
                }
                deq.push_back(e[p]);
            }
    }
    return lst;
}

long long solve(int x)
{
    long long ans = 0;
    vector <int> cyc = bfsFindCycle(x);
    int n = cyc.size();
    for (int i = 0; i < n; ++i)
    {
        int pt, pp;
        t[cyc[i]] = 0;
        g[i].lst = bfsLongest(cyc[i], pt);
        ans = max(ans, bfsLongest(pt, pp));
        t[cyc[i]] = now;
        if (n == 1) return g[i].lst;
        g[i].w = cost[cyc[i]];
        g[i].f = 0;
        g[i + n] = g[i];
    }
    g[0].w = 0;
    for (int i = 1; i < 2 * n; ++i)
    g[i].w += g[i - 1].w;
    g[0].f = g[0].lst;
    deque <int> deq;
    deq.push_back(0);
    for (int i = 1; i < 2 * n; ++i)
    {
        for (; deq.front() + n <= i; deq.pop_front());
        g[i].f = g[i].lst + g[i].w + g[deq.front()].lst - g[deq.front()].w;
        for (; !deq.empty() && g[deq.back()].lst - g[deq.back()].w <= g[i].lst - g[i].w; deq.pop_back());
        deq.push_back(i);
    }
    for (int i = 0; i < 2 * n; ++i)
        ans = max(ans, g[i].f);
    return ans;
}

int main()
{
    freopen("island.in", "r", stdin);
    int n;
    num = 1;
    scanf("%d", &n);
    for (int i = 1; i <= n; ++i)
    {
        int v, len;
        scanf("%d%d", &v, &len);
        link(i, v, len);
        link(v, i, len);
    }
    long long ans = 0;
    for (int i = 1; i <= n; ++i)
        if (!vis[i])
            ans += solve(i);
    printf("%lld\n", ans);
    return 0;
```

```
}
```

Listing 2.2: ioi2008Island.cpp

2.3 斜率优化

- $f[i] = \min(a[i] * x[j] + b[i] * y[j])$
- 更好的理解：设 $P=f[i]$, 则 $y = (-a/b)x + P/b$. 求满足要求的最小截距。或者通过各种转化，最优决策就是从无穷远朝原点移动，第一个碰上的点为最优决策点。
- 很好的性质：所有最优决策一定在当前所有点构成的凸包上。(例如，在最优决策点划一条相应斜率的线，其余点均在该线上方，)

2.3.1 斜率以及 x 维都单调

想像斜率越来越大的直线往 y 正方向移动，第 i 次移动首次碰上 k。对于以后的决策，因为斜率更大，那么在 k 之前，第 i 次移动没有碰上的点必然再也用不上了，所以可以维护一个单调队列。下面例题是：把一个序列切开，每个部分权值是和平方加常数，求权值和最小值

```cpp
#include <deque>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <cstdlib>

#define MAXN 512345

using namespace std;
typedef long long arrayN[MAXN];

struct node
{
    long long x, y, f;
    node (long long tx = 0, long long ty = 0, long long tf = 0)
    {
        x = tx, y = ty, f = tf;
    }
    //y = f + sum^2, x = sum
}g[MAXN];

long long sqr(long long x)
{
    return x * x;
}

long long cross(long long x1, long long y1, long long x2, long long y2)
{
    return x1 * y2 - x2 * y1;
}


deque < int > deq;

int main()
{
    freopen("hdu3507.in", "r", stdin);
    int N, M;
    for (; scanf("%d%d", &N, &M) != EOF; )
    {
        deq.clear();
        g[0] = node(0, 0, 0);
        deq.push_back(0);
        for (int i = 1; i <= N; ++i)
        {
            int x;
            scanf("%d", &x);
            g[i].x = g[i - 1].x + x;
            long long lim = g[i].x << 1;
            for (; deq.size() > 1; deq.pop_front())
            {
                node u = g[deq[0]];
                node v = g[deq[1]];
                if ((v.y - u.y) > lim * (v.x - u.x))
                    break;
            }
            node pt = g[deq.front()];
            g[i].f = pt.f + sqr(g[i].x - pt.x) + M;
            g[i].y = sqr(g[i].x) + g[i].f;
            for (; deq.size() >= 2; deq.pop_back())
            {
                node A = g[deq[deq.size() - 2]];
                node B = g[deq[deq.size() - 1]];
                node C = g[i];
                if (cross(B.x - A.x, B.y - A.y, C.x - B.x, C.y - B.y) > 0) break;
            }
            deq.push_back(i);
        }
        cout << g[N].f << endl;
    }
    return 0;
}
```

Listing 2.3: hdu3507.cpp

2.3.2 随便什么情况：cdq 分治优化

- 排序的顺序，凸壳的方向写之前一定要画清楚。
- 这里归并排一维的序可以节省一个 log 的复杂度
- cdq 分治的顺序至关重要，千万不能乱。
- f[i] 表示第 i 天手上的券全换成现金最多多少, 其中 x[j],y[j] 分别表示用 f[j] 的钱换成 A，B 券分别能有多少。
- $f[i] = \max(\max(A[i] * x[j] + B[i] * x[j], f[j]))$
- 就是经典的斜率优化问题咯。不用平衡树的话可以离线用 cdq 分治。先按照 - A[i]/B[i] 排序（具体大小顺序画一画就知道了）。solve(l, r) 时需要按照下标 lab 大小分为两部分。然后 solve(l,mid), 同时主义归并把递散维 x 排好序。l ~ mid 至 mid ＋ 1 r 转移. 最后 solve(mid + 1, r)，接着归并排好 x 就行了。

```
#include <bits/stdc++.h>
#define MST(a, b) memset((a), (b), sizeof(a))
#define MAXN 112345
#define esp 1e-8

using namespace std;

struct node
{
    double A, B, rate; //A/B
    double x, y;
    double f;
}g[MAXN];

int lab[MAXN], a[MAXN];

int cmp(double x)
{
    if (x < -esp) return -1;
    if (x > esp) return 1;
    return 0;
}

int smaller(int u, int v)
{
    int tx = cmp(g[u].x - g[v].x);
    int ty = cmp(g[u].y - g[v].y);
    return tx < 0 || (tx == 0 && ty < 0);
}

void mergeSortX(int al, int ar, int bl, int br)
{
    int Na = 0;
    for (int i = al; i <= ar; ++i)
    {
        while (bl <= br && smaller(lab[bl], lab[i]))
            a[++Na] = lab[bl++];
        a[++Na] = lab[i];
    }
    for (; bl <= br; ++bl)
        a[++Na] = lab[bl];
    for (int i = 1; i <= Na; ++i)
        lab[al + i - 1] = a[i];
}


double cross(int A, int B, int C)
{
    return (g[B].x - g[A].x) * (g[C].y - g[B].y) - (g[B].y - g[A].y) * (g[C].x - g[B].x);
}

double comRate(int A, int B, int C)
{
    return (g[B].y - g[A].y) * g[C].B + g[C].A * (g[B].x - g[A].x);
}

void getRightPartF(int al, int ar, int bl, int br)
{
    int Na = 0;
    double lim = 0;
    for (int i = al; i <= ar; ++i)
    {
        lim = max(lim, g[lab[i]].f);
        while (Na >= 2 && cmp(cross(a[Na - 1], a[Na], lab[i])) >= 0)
```

```cpp
            --Na;
        a[++Na] = lab[i];
    }
    int La = 1;
    for (int i = bl; i <= br; ++i)
    {
        int p = lab[i];        g[p].f = max(g[p].f, lim);
        for (; La + 1 <= Na && cmp(comRate(a[La], a[La + 1], p)) >= 0; ++La);
        g[p].f = max(g[p].f, g[a[La]].x * g[p].A + g[a[La]].y * g[p].B);
    }
}

void solve(int l, int r)
{
    if (l == r)
    {
        int p = lab[l];
        //g[p].f = max(g[p].f, g[p - 1].f);
        g[p].x *= g[p].f;
        g[p].y *= g[p].f;
        return ;
    }
    int Na = r - l + 1;
    int upLim = 0, downLim = MAXN;
    for (int i = l; i <= r; ++i)
    {
        upLim = max(upLim, lab[i]);
        downLim = min(downLim, lab[i]);
    }
    int midLim = (upLim + downLim) >> 1;
    int pLow = 0;
    for (int i = l; i <= r; ++i)
        if (lab[i] <= midLim)
            a[++pLow] = lab[i];
    int pHigh = pLow;
    for (int i = l; i <= r; ++i)
        if (lab[i] > midLim)
            a[++pHigh] = lab[i];
    for (int i = 1; i <= Na; ++i)
        lab[i + l - 1] = a[i];
    pLow += l - 1;
    solve(l, pLow);
    getRightPartF(l, pLow, pLow + 1, r);
    solve(pLow + 1, r);
    mergeSortX(l, pLow, pLow + 1, r);
}

int com(int u, int v)
{
    node tu = g[u];
    node tv = g[v];
    return tu.A * tv.B < tv.A * tu.B;
}

int main()
{
    //  freopen("cash4.in", "r", stdin);
    int N, S;
    scanf("%d%d", &N, &S);
    for (int i = 1; i <= N; ++i)
    {
        scanf("%lf%lf%lf", &g[i].A, &g[i].B, &g[i].rate);
        g[i].y = 1.0 / (g[i].B + g[i].A * g[i].rate);
        g[i].x = g[i].y * g[i].rate;
        g[i].f = S;
        lab[i] = i;
    }
    g[1].f = S;
    sort(lab + 1, lab + N + 1, com);
    solve(1, N);
    double ans = 0;
    for (int i = 1; i <= N; ++i)
        ans = max(ans, g[i].f);
    printf("%.3f\n", ans);
    return 0;
}
```

Listing 2.4: cash.cpp

Chapter 3

图论

3.1 tarjan TODO

3.1.1 2-sat

如果没有产生矛盾, 把处在同一个强联通分量中的点和边缩成一个点, 得到新的有向图 G'. 然后, 把 G' 中的所有弧反向, 得到图 G". 现在观察 G", 由于已经进行了缩点操作, 所以是拓扑图.

把 G" 所以点标记未着色. 按照拓扑顺序重复下面操作: 1. 选择未着色的顶点 x. 把 x 染成红色. 2. 把所有与 x 矛盾的顶点 y 及其子孙全部染成蓝色 3. 重复操作 1 和 2, 知道不存在未着色的点位置. 此时 G" 中被染成红色的点在图 G 中对应的定点集合, 就是 2-SAT 的一组解

```cpp
//指定小写字母元音/辅音
//给出第i个位置是元音/辅音蕴涵j位置元音/辅音
//给定字符串st,求字典序不小于它的最小的合法2-sat方案
#include <bits/stdc++.h>
#define MAXN 500
#define MAXM 512345

using namespace std;
typedef int arrayN[MAXN], arrayM[MAXM];

char g[30], st[MAXN];
arrayN fir0, low, dfn, inVec, cnt, belong;
arrayN deg, con0, con1, fir1, topOrder, col;
arrayM e0, nxt0, e1, nxt1;
int num, now, tot, nextAlp[30][2], firAlp[2];
vector<int> vec;

int getKind(char ch) {
    if (ch == 'V') return 0;
    else return 1;
}

void link0(int u, int v) {
    e0[++num] = v, nxt0[num] = fir0[u];
    fir0[u] = num;
}

void link1(int u, int v) {
    e1[++num] = v, nxt1[num] = fir1[u];
    fir1[u] = num;
}

void tarjan(int x) {
    low[x] = dfn[x] = ++now;
    vec.push_back(x);
    for (int p = fir0[x], q; p; p = nxt0[p])
        if (!inVec[q = e0[p]])
            if (!dfn[e0[p]]) {
                tarjan(e0[p]);
                low[x] = min(low[x], low[e0[p]]);
            } else low[x] = min(low[x], dfn[e0[p]]);
    if (low[x] == dfn[x]) {
        cnt[belong[x] = ++tot] = 1;
        inVec[x] = 1;
        for (; vec.back() != x; vec.pop_back()) {
            int q = vec.back();
            inVec[q] = 1;
            cnt[belong[q] = tot]++;
        }
        vec.pop_back();
    }
}

void topSort() {
    int l = 1, r = 0;
    for (int i = 1; i <= tot; ++i)
        if (deg[i] == 0) topOrder[++r] = i;
    for (; l <= r; ++l) {
        int u = topOrder[l];
        for (int p = fir1[u]; p; p = nxt1[p]) {
            --deg[e1[p]];
            if (deg[e1[p]] == 0) topOrder[++r] = e1[p];
        }
    }
```

```
}
int getDAG(int n) {
    for (int i = 1; i <= n * 2; ++i)
        dfn[i] = low[i] = belong[i] = inVec[i] = deg[i] = 0;
    now = tot = num = 0;     for (int i = 1; i <= n * 2; ++i)
        if (!dfn[i]) tarjan(i);
    for (int i = 1; i <= n; ++i)
        if (belong[i] == belong[con0[i]]) return 0;
    for (int i = 1; i <= 2 * n; ++i) {
        for (int p = fir0[i]; p; p = nxt0[p]) {
            int q = e0[p];
            if (belong[i] == belong[q]) continue;
            link1(belong[q], belong[i]);
            deg[belong[i]]++;
        }
        con1[belong[i]] = belong[con0[i]];
        con1[belong[con0[i]]] = belong[i];
    }
    topSort();
    return 1;
}

int dye(int x, int co) {
    if (col[x]) {
        return (co == col[x]);
    }
    col[x] = co;
    for (int p = fir1[x]; p; p = nxt1[p])
        if (!dye(e1[p], co)) return 0;
    return 1;
}

int originDye(int p, int n) {
    int all = -1;
    if (firAlp[0] > 'z') all = 0;
    if (firAlp[1] > 'z') all = 1;
    if (all >= 0)
        for (int i = 1; i <= n; ++i) {
            int pos1 = i + all * n;
            int pos0 = con1[pos1];
            if (col[pos0] == 2) return 0;
            col[pos0] = 1;
            if (!dye(pos1, 2)) return 0;
        }
    for (int i = 1; i <= p + 1; ++i) {
        int pos = belong[i];
        if (getKind(g[st[i - 1] - 'a'])) pos = con1[pos];
        if (col[pos] == 2) return 0;
        col[pos] = 1;
        if (!dye(con1[pos], 2)) return 0;
    }
    return 1;
}
int DAGDye(int n) {
    for (int i = 1; i <= n; ++i) {
        int x = topOrder[i];
        if (!col[x]) {
            col[x] = 1;
            if (!dye(con1[x], 2)) return 0;
        }
    }
    return 1;
}

int finalCheck(int n, int p) {
    for (int i = 1; i <= n; ++i) {
        if (col[belong[i]] != 1 && col[belong[con0[i]]] != 1)
    return 0;
    }
    return 1;
}

int solve(int n, int p) {
    memset(col, 0, sizeof(col));
    if (!originDye(p, n)) return 0;
    if (!DAGDye(tot)) return 0;
    return finalCheck(n, p);
}

void getNextAlp() {
    int len = strlen(g);
    firAlp[1] = firAlp[0] = 'z' + 1;
    for (int i = 0; i < len; ++i) {
        nextAlp[i][0] = nextAlp[i][1] = 'z' + 1;
        int k = getKind(g[i]);
        firAlp[k] = min(firAlp[k], i + 'a');
        for (int j = i + 1; j < len; ++j) {
            int k = getKind(g[j]);
            nextAlp[i][k] = min(nextAlp[i][k], 'a' + j);
        }
        if (nextAlp[i][0] > nextAlp[i][1])
            swap(nextAlp[i][0], nextAlp[i][1]);
    }
}
int main() {
#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin);#endif
    scanf("%s", g);
    int n, m;
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; ++i) {
        con0[i] = i + n;
        con0[i + n] = i;
    }
    num = 0;
    for (int i = 1; i <= m; ++i) {
        char t1, t2;
        int pos1, pos2;
        scanf("%d %c %d %c\n", &pos1, &t1, &pos2, &t2);
        //  if (i == 50 && n == 50 && m == 50) printf("%d %c %d %c\n", pos1, t1, pos2, t2);
        int k1 = getKind(t1);
        int k2 = getKind(t2);
        pos1 += k1 * n;
        pos2 += k2 * n;
        link0(pos1, pos2);
        link0(con0[pos2], con0[pos1]);
    }
    scanf("%s", st);
    // if (n == 50 && m == 50) printf("%s\n", st);
    getNextAlp();
    if (getDAG(n) == 0) {
        printf("-1\n");
        return 0;
    }
    int flag = solve(n, n - 1);
    for (int i = n - 1; i >= 0 && !flag; --i) {
        int tmp = st[i] - 'a';
        for (int j = 0; j <= 1 && !flag; ++j)
            if (nextAlp[st[i] - 'a'][j] <= 'z') {
                st[i] = nextAlp[tmp][j];
                flag = solve(n, i);
                if (flag) {
                    for (int k = i + 1; k <= n - 1; ++k) {
                        int u = firAlp[0];
                        int v = firAlp[1];
                        if (u > v) swap(u, v);
                        st[k] = u;
                        if (solve(n, k)) continue;
                        st[k] = v;
                    }
                }
            }
    }
    if (!flag) printf("-1\n");
    else printf("%s\n", st);
    return 0;
}
```

Listing 3.1: cf568C.cpp

3.1.2 割顶，点双联通分量 TODO

- 每条边恰好属于一个双联通分量
- 不同双联通分量最多只有一个公共点, 且一定是割顶
- 任意割顶都是至少两个不同双联通分量的公共点

3.1.3 桥，边双联通分量 TODO

去掉桥之后求联通块即得边边双联通分量

3.2 平面图 TODO

farmland 那道题，平面图判定 hnoi

3.3 pufer 编码

一棵标号树的 Pufer 编码规则如下：找到标号最小的叶子节点，输出与它相邻的节点到 prufer 序列, 将该叶子节点删去，反复操作，直至剩余 2 个节点。

3.4 最佳追捕算法

问题描述: 逃犯若干, 在公路网上流窜, 最少派几名刑警, 才能保证抓获全部逃犯.

做法: 每次删除所有叶子, 分一层. 直到删除到只剩下一条链为止. 层数 (算上一条链那层) 就是答案.

3.5 网络流 TODO

3.5.1 dinic

uva11248 流量大于等于 C 的流是否存在。如果不存在，修改哪些边的流量可以使得存在。

```cpp
#include <bits/stdc++.h>
#define REP(i, n) for(int i = 0; i < (int) (n); ++i)
#define REPP(i, a, b) for (int i = (int) (a); i <= (int) (b); ++i)
#define MST(a, b) memset((a), (b), sizeof(a))
#define MAXN 205
#define MAXM 21234

using namespace std;

typedef int arrayN[MAXN], arrayM[MAXM];
int N, E, C, num;
const int INF = ~0U >> 1;
arrayN fir, d;
arrayM nxt, e;
long long c[MAXM], c0[MAXM];

struct edge
{
  int u, v, lab;
  edge(int u = 0, int v = 0, int lab = 0): u(u), v(v), lab(lab) {};
} g[MAXM], cand[MAXM];

void link(int u, int v, int w)
{
  e[++num] = v, nxt[num] = fir[u];
  fir[u] = num, c[num] = 1LL * w;
}

void copy(long long cs[], long long cd[])
{
  REPP(i, 1, num) cd[i] = cs[i];
}

bool bfs(int s)
{
  MST(d, 0x3f);
  d[s] = 0;
  queue<int> que;
  que.push(s);
  for (; !que.empty();)
  {
    int u = que.front();
    que.pop();
    for (int p = fir[u]; p; p = nxt[p])
      if (c[p] && d[e[p]] > d[u] + 1)
      {
        d[e[p]] = d[u] + 1;
        que.push(e[p]);
      }
  }
  return d[N] < d[0];
}

long long dfs(int x, long long low)
{
  long long flow = 0;
  if (x == N) return low;
  for (int p = fir[x]; p; p = nxt[p])
    if (c[p] && d[e[p]] == d[x] + 1)
    {
      long long tmp = dfs(e[p], min(low, c[p]));
      if (!tmp) d[e[p]] = d[0];
      c[p] -= tmp, c[p ^ 1] += tmp;
      flow += tmp, low -= tmp;
      if (!low) break;
    }
  return flow;
}

int com(edge A, edge B)
{
  return A.u < B.u || (A.u == B.u && A.v < B.v);
}
void findCutEdge(long long base)
{
  int tot = 0;
  REPP(i, 1, N)
    if (d[i] < d[0])
      for (int p = fir[i]; p; p = nxt[p])
        if (d[e[p]] >= d[0] && (!(p & 1)))
          cand[++tot] = edge(i, e[p], p);
  copy(c, c0);

  int ansTot = 0;
  REPP(i, 1, tot)
  {
```

```
  copy(c0, c);
  c[cand[i].lab] = C;
  long long ans = base;
  for (; ans < C && bfs(1); ans += dfs(1, C));
  if (ans >= C) g[++ansTot] = cand[i];  }
 if (ansTot == 0)
 {
  printf("not possible\n");
  return ;
 }
 sort(g + 1, g + ansTot + 1, com);
 printf("possible option:(%d,%d)", g[1].u, g[1].v);
 REPP(i, 2, ansTot)
  printf(",(%d,%d)", g[i].u, g[i].v);
 printf("\n");
}

int main()
{
 freopen("uva11248.in", "r", stdin);
 int task = 0;
 for (;;)
 {
  scanf("%d%d%d", &N, &E, &C);
  if (N + E + C == 0) break;
  num = 1;
  MST(fir, 0);
  REP(i, E)
  {
   int u, v, w;
   scanf("%d%d%d", &u, &v, &w);
   link(u, v, w);
   link(v, u, 0);
  }
  long long ans = 0;
  for (; ans < C && bfs(1); ans += dfs(1, C));
  ++task;
  printf("Case %d: ", task);
  if (ans >= C)
  {
   printf("possible\n");
   continue;
  }
  findCutEdge(ans);
 }
 return 0;
}
```

Listing 3.2: uva11248.cpp

3.5.2 费用流 TODO

3.5.3 常见模型 TODO

3.6 弦图

3.6.1 做法与常见问题

做法如下：

- 最大势算法求待验证完美消除序列
  1. 未被选的点中选被标记次数最多的点 i
  2. 把 i 相邻的点标记次数 + 1

- 判断是否为完美消除序列（下述扫描必需全部完成）
  1. 上述序列依次扫描，扫到 i
  2. 标号小于 seq[i] 的与 i 相邻点为 j1,j2,...jk
  3. 判断 jk 与 j1,j2...jk-1 相邻即可

常见问题如下：

- 色数：贪心按照完美消除序列产生顺序依次染最小的能染的颜色

- 最大独立集：贪心按照完美消除序列产生顺序倒着依次选，能选就选

- 最小团覆盖（用最少的团覆盖所有点）: 最大独立集带上下面的 N 集合

- 极大团：

  - $N(v) = w \mid w$ 与 v 相邻，且先加入
  - 团一定是 v union $N(v)$ 的形式
  - 现在需要判断每个 v union $N(v)$ 是否为极大团
  - $next[v]$ 是与 v 相邻的, 最靠近 v 的已经加入完美序列的点
  - $next[w] = v$ 且 $|N(v)| + 1 <= |N(w)|$, 则 v 不是极大团

- 最大团 = 最小染色，最大点独立集 = 最小团覆盖（对于弦图任何诱导子图成立，即完美图）

- 区间图的完美消除序列就是右端点排序。从大到小依次加入完美消除序列。选最多区间不重叠：(最大独立集), 从小到大排序依次加

3.6.2 万不得已用线性作法

这个是判断是否为弦图

```
#include <bits/stdc++.h>
#define MAXN 1123
#define MAXM 2123456

using namespace std;
typedef int arrayN[MAXN], arrayM[MAXM];

arrayN fir, firMcs, nxtMcs, mcsSeq, l;
arrayN vis, r, cnt, preMcs, lab;
arrayM nxt, e;
int num, flag[MAXN][MAXN];
int mx; // max

void link(int u, int v)
{
    e[++num] = v, nxt[num] = fir[u];
    fir[u] = num;
}

void delMcs(int pos, int pt)
{
    if (nxtMcs[pt] == pt)
    {
        r[l[pos]] = r[pos];
        l[r[pos]] = l[pos];
        if (pos == mx) mx = l[mx];
        firMcs[pos] = 0;
        return ;
    }
    preMcs[nxtMcs[pt]] = preMcs[pt];
    nxtMcs[preMcs[pt]] = nxtMcs[pt];
    if (firMcs[pos] == pt)
        firMcs[pos] = nxtMcs[pt];
}

void insMcs(int pos, int pt)
{
    if (firMcs[pos])
    {
        int tmp = firMcs[pos];
        nxtMcs[pt] = tmp;
        preMcs[pt] = preMcs[tmp];
        nxtMcs[preMcs[pt]] = pt;
        preMcs[nxtMcs[pt]] = pt;
        return;
    }
    preMcs[pt] = nxtMcs[pt] = firMcs[pos] = pt;
    if (firMcs[pos - 1]) //easy wrong
    {
        l[pos] = pos - 1;
        r[pos] = r[pos - 1];
    } else
    {
        if (l[pos - 1] == pos - 1)
            l[pos] = r[pos] = pos;
        else
        {
            l[pos] = l[pos - 1];
            r[pos] = r[pos - 1];
        }
    }
    r[l[pos]] = l[r[pos]] = pos;
    if (pos > mx) mx = pos;
}

void getMcsSeq(int n, int m)
{
    mx = 0;
    l[0] = 0, r[0] = 0;
    memset(firMcs, 0, sizeof(firMcs));
    memset(cnt, 0, sizeof(cnt));
    for (int i = 1; i <= n; ++i)
    {
        nxtMcs[i] = i + 1;
        preMcs[i] = i - 1;
    }
    nxtMcs[n] = 1, preMcs[1] = n;
    firMcs[0] = 1;
    memset(vis, 0, sizeof(vis));
    for (int i = 1; i <= n; ++i)
    {
        int tmp = (mcsSeq[i] = firMcs[mx]);
        delMcs(cnt[tmp], tmp);
        vis[tmp] = 1;
        for (int p = fir[tmp]; p; p = nxt[p])
            if (!vis[e[p]])
            {
                delMcs(cnt[e[p]], e[p]);
                ++cnt[e[p]];
                insMcs(cnt[e[p]], e[p]);
            }
    }
}

int checkMcs(int n)
{
    for (int i = 1; i <= n; ++i)
        lab[mcsSeq[i]] = i;
    memset(vis, 0, sizeof(vis));
    int now = 0;
    for (int i = 1; i <= n; ++i)
    {
        ++now;
        int pt = mcsSeq[i], cnt = 0, bgst = 0;
        for (int p = fir[pt]; p; p = nxt[p])
            if (lab[e[p]] < i)
            {
                vis[e[p]] = now;
                ++cnt;
                if (lab[e[p]] > bgst)
                    bgst = e[p];
            }
        if (bgst == 0) continue;
        for (int p = fir[bgst]; p; p = nxt[p])
        {
            if (lab[e[p]] < i && vis[e[p]] == now)
                --cnt;
        }
        if (cnt > 1) return 0;
    }
    return 1;
}

int main()
{
    //  freopen("in.txt", "r", stdin);
    //freopen("out.txt", "w", stdout);
    for (;;)
    {
        int n, m;
        scanf("%d%d", &n, &m);
        if (n + m == 0) break;
        num = 0;
        memset(fir, 0, sizeof(fir));
        memset(flag, 0, sizeof(flag));
        for (int i = 1; i <= m; ++i)
        {
            int u, v;
            scanf("%d%d", &u, &v);
            if (flag[u][v] || u == v) continue;
            link(u, v);
            link(v, u);
            flag[u][v] = flag[v][u] = 1;
        }
        getMcsSeq(n, m);
        if (checkMcs(n)) printf("Perfect\n\n");
        else printf("Imperfect\n\n");
    }
    return 0;
}
```

Listing 3.3: zoj1015.cpp

3.6.3 nlogn 好写得多

这个是求色数

```
#include <bits/stdc++.h>
#define MAXN 11234
#define MAXM 2123456

using namespace std;

typedef int arrayN[MAXN], arrayM[MAXM];

arrayN fir, mcsOrder, label, col;
arrayM e, nxt;
int num, n, base, seg[MAXN * 4];
set <int> s;

void link(int u, int v) {
    e[++num] = v, nxt[num] = fir[u];
    fir[u] = num;
}

int maxLab(int u, int v) {
    return label[u] > label[v] ? u : v;
}

void change(int x, int val) {
    label[x] = val;
    x += base;
    for (x >>= 1; x; x >>= 1) {
        seg[x] = maxLab(seg[x << 1], seg[x << 1 ^ 1]);
    }
}
void getMCS() {
    for (base = 1; base <= n + 1; base <<= 1);
    for (int i = 1; i <= n; ++i) seg[i + base] = i;
    label[0] = -1;
    for (int i = base - 1; i >= 1; --i)
        seg[i] = maxLab(seg[i << 1], seg[i << 1 ^ 1]);
    int tot = 0;
    for (int i = 1; i <= n; ++i) {
        int x = mcsOrder[++tot] = seg[1];
        change(x, -1);
        for (int p = fir[x]; p; p = nxt[p]) {
            if (label[e[p]] >= 0) change(e[p], label[e[p]] + 1);
        }
    }
}
int main()
{
#ifndef ONLINE_JUDGE
    freopen("in.txt", "r", stdin);
#endif
    int m;
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; ++i) {
        int u, v;
        scanf("%d%d", &u, &v);
        link(u, v);
        link(v, u);
    }
    getMCS();
    int ans = 0;
    for (int i = 1; i <= n; ++i)
        s.insert(i);
    for (int j = 1; j <= n; ++j)  {
        int i = mcsOrder[j];
        for (int p = fir[i]; p; p = nxt[p]) {
            set<int>::iterator it = s.find(col[e[p]]);
            if (it != s.end())
                s.erase(it);
        }
        col[i] = *s.begin();
        ans = max(ans, col[i]);
        for (int p = fir[i]; p; p = nxt[p]) {
            set<int>::iterator it = s.find(col[e[p]]);
            if (col[e[p]] && it == s.end())
                s.insert(col[e[p]]);
        }
    }
    printf("%d\n", ans);
    return 0;
}
```

Listing 3.4: hnoi2008.cpp

3.7 最小树形图

- 特别注意判断 root 的地方.
- 下面这题是二分，选择大于等于 bLowLim 的边才有效
- 这是指定了 root 为 0
- 不固定根的时候，只需要新加根节点。到每个点连边，边权大于所有边之和即可。

```
#include <bits/stdc++.h>
#define REP(i, n) for (int i = 0; i < (int) (n); ++i)
#define REPP(i, a, b) for(int i = (int) (a); i <= (int) (b); ++i)
#define MST(a, b) memset((a), (b), sizeof(a))
#define MAXN 66
#define MAXM 11234

using namespace std;
const int oo = ~0U>>1;
typedef int arrayN[MAXN], arrayM[MAXM];

int N, M, C;
arrayN vis, minW, belong, pre;

struct edge
{
  int u, v, b, c;
  edge(int u1 = 0, int v1 = 0, int b1 = 0, int c1 = 0)
    {
      u = u1, v = v1, b = b1,  c = c1;
    }
}edOri[MAXM], ed[MAXM];
```

```cpp

int zhuLiu(int bLowLim)
{
  int root = 0, tot = N, ntot;
  int ans = 0;
  REP(i, M) ed[i] = edOri[i];  for (;;)
  {
    REP(i, tot) minW[i] = oo, vis[i] = -1, belong[i] = -1;
    REP(i, M)
    {
      if (ed[i].u == ed[i].v || ed[i].b < bLowLim) continue;
      if (ed[i].c < minW[ed[i].v])
      {
        minW[ed[i].v] = ed[i].c;
        pre[ed[i].v] = ed[i].u;
      }
    }

    pre[root] = -1;
    minW[root] = 0;
    REP(i, tot)
      if(minW[i] >= oo) return oo;
      else ans += minW[i];
    ntot = 0;
    REP(i, tot)
      if (vis[i] == -1)
      {

        int h1 = i;
        for (; vis[h1] == -1; h1 = pre[h1])
        {
          vis[h1] = i;
          if (h1 == root) break;
        }
        if (h1 == root || vis[h1] != i) continue;
        int h2 = h1;
        for (h2 = pre[h1]; h2 != h1; h2 = pre[h2])
          belong[h2] = ntot;
        belong[h1] = ntot++;
      }
    REP(i, tot) if (belong[i] == -1) belong[i] = ntot++;
    REP(i, M)
    {
      ed[i].c -= minW[ed[i].v];
      ed[i].u = belong[ed[i].u];
      ed[i].v = belong[ed[i].v];
    }
    if (tot == ntot) return ans;
    tot = ntot;
    root = belong[root];
  }
}

int main()
{
  freopen("in.txt", "r", stdin);
  int task;
  for (scanf("%d", &task); task; --task)
  {
    int L = 1, R = 1;
    scanf("%d%d%d", &N, &M, &C);
    REP(i, M)
    {
      int u, v, b, c;
      scanf("%d%d%d%d", &u, &v, &b, &c);
      edOri[i] = edge(u, v, b, c);
      R = max(R, b);
    }
    L = 0;
    for (; L < R; )
    {
      int mid = (L + R + 1) >> 1;
      if (zhuLiu(mid) > C)
        R = mid - 1;
      else L = mid;
    }
    if (L == 0) printf("streaming not possible.\n");
    else printf("%d kbps\n", L);
  }
  return 0;
}
```

Listing 3.5: uva11865.cpp

3.8 二分图

3.8.1 普通 KM

```cpp
#include <bits/stdc++.h>
#define REP(i, n) for (int i = 0; i < (n); ++i)
#define REPP(i, a, b) for(int i = (a); i <= (b); ++i)
```

```
#define MST(a, b) memset((a), (b), sizeof(a))
#define MAXN 512
#define INF 0x3f3f3f3f

using namespace std;

typedef int arrayN[MAXN];

int n;
arrayN S, T, match, w[MAXN], lx, ly;

int dfs(int x)
{
  S[x] = 1;
  REPP(i, 1, n)
    if (lx[x] + ly[i] == w[x][i] && !T[i])
  {
    T[i] = 1;  //容易忽略
    if (!match[i] || dfs(match[i])) //dfs中别漏了match
    {
      match[i] = x;
      return 1;
    }
  }
  return 0;
}

void update()
{
  int minL = INF; //找最小
  REPP(i, 1, n)
    if (S[i])
      REPP(j, 1, n)
        if (!T[j])
          minL = min(minL, lx[i] + ly[j] - w[i][j]);
  REPP(i, 1, n)
  {
    if (S[i]) lx[i] -= minL;
    if (T[i]) ly[i] += minL;
  }
}
void KM()
{
  REPP(i, 1, n)
  {
    lx[i] = 0;
    ly[i] = 0;
    match[i] = 0;
    REPP(j, 1, n)
      lx[i] = max(lx[i], w[i][j]);
  }
  REPP(i, 1, n)
  {
    for (;;)
    {
      MST(S, 0);
      MST(T, 0);
      if (dfs(i)) break;
      else update();
    }
  }
}
int main()
{
  freopen("in.txt", "r", stdin);
  for (; scanf("%d", &n) != EOF; )
  {
    REPP(i, 1, n)
      REPP(j, 1, n)
      scanf("%d", &w[i][j]);
    KM();
    REPP(i, 1, n)
      printf("%d%c", lx[i], " \n"[i == n]);
    REPP(i, 1, n)
      printf("%d%c", ly[i], " \n"[i == n]);
    int ans = 0;
    REPP(i, 1, n)
      ans += w[match[i]][i];
    printf("%d\n", ans);
  }
  return 0;
}
```

Listing 3.6: uva11383.cpp

3.8.2 牛逼 KM

```
#include<vector>
#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>
```

```cpp
#include <cmath>
#include <cstdlib>
using namespace std;

const int N = 110 + 1;
const double INF = 1e12, EPS = 1e-6;

int n, p[N][N], fa[N];
bool used[N];
double w[N][N], u[N][N], v[N][N],  minv[N];
// smallest match
void km(int lev) {
  int i = lev;
  lev++;
  for (int j = 0; j <= n; ++j) {
    u[lev][j] = u[i][j];
    v[lev][j] = v[i][j];
    p[lev][j] = p[i][j];
    minv[j] = INF;
    used[j] = false;
  }
  p[lev][n] = i;
  int j0 = n;
  do {
    used[j0] = true;
    int i0 = p[lev][j0], j1;
        double delta = INF;
    for (int j = 0; j < n; ++j) {
      if (!used[j]) {
        double cur = w[i0][j] - u[lev][i0] - v[lev][j];
        if (cmp(cur - minv[j]) < 0) {
          minv[j] = cur;
          fa[j] = j0;
        }
        if (cmp(minv[j] - delta) < 0) {
          delta = minv[j];
          j1 = j;
        }
      }
    }
    for (int j = 0; j <= n; ++j) {
      if (used[j]) {
        u[lev][p[lev][j]] += delta, v[lev][j] -= delta;
      } else {
        minv[j] -= delta;
      }
    }
    j0 = j1;
  } while (p[lev][j0] != -1);
  do {
    int j1 = fa[j0];
    p[lev][j0] = p[lev][j1];
    j0 = j1;
  } while (j0 != n);
}

int main()
{
    for (int i = 0; i <= n; ++i) {
        u[0][i] = v[0][i] = 0;
        p[0][i] = -1, fa[i] = 0;
    }
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j)
            w[i][j] = 1.0 * dist(a[i], b[j]);
        w[i][n] = 0;
    }
    for (int i = 0; i < n; ++i) km(i);
    double ans = 0;
    for (int i = 0; i < n; ++i) {
        ans += w[p[n][i]][i];
        printf("%d\n", p[n][i] + 1);
    }
}
```

Listing 3.7: poj3565Better.cpp

3.8.3 常见问题汇总

- 最大独立集: 等于顶点数减去最大匹配。最大匹配中点全部去掉，剩余的点为独立集。此时共 |V|-2|M| 个点。接着从匹配边取一边加入独立集（这两个点不可能同时与非匹配点相邻，否则可以增广）。

- 最大团：补图的最大独立集

- 最小点覆盖: 即最大匹配。输出方案见代码

- 最小路径覆盖所有点

- DAG 最小不相交路径覆盖:

  把原图中的每个点 V 拆成 Vx 和 Vy，如果有一条有向边 A->B，那么就加边 Ax-By。这样就得到了一个二分图，最小路径覆盖 = 原图的节点数 -新图最大

匹配。证明: 一开始每个点都独立的为一条路径，总共有 n 条不相交路径。我们每次在二分图里加一条边就相当于把两条路径合成了一条路径，因为路径之间不能有公共点，所以加的边之间也不能有公共点，这就是匹配的定义。所以有：最小路径覆盖 = 原图的节点数 -新图最大匹配。

- 有向无环图最小可相交路径覆盖: 先用 floyd 求出原图的传递闭包，即如果 a 到 b 有路，那么就加边 a->b。然后就转化成了最小不相交路径覆盖问题。
- 稳定婚姻问题很有趣，见白书 P353。

3.8.4 最小点覆盖输出方案

```cpp
#include <bits/stdc++.h>
#define REP(i, n) for (int i = 0; i < (n); ++i)
#define REPP(i, a, b) for(int i = (a); i <= (b); ++i)
#define MAXN 1123
#define MST(a, b) memset((a), (b), sizeof(a))

using namespace std;

int n, m, tot, w[MAXN][MAXN], vis[MAXN], cok[MAXN], rok[MAXN], match[MAXN];

int dfs(int x)
{
  REPP(i, 1, n)
    if (w[x][i] && !vis[i])
    {
      vis[i] = 1; //容易忽略
      if (!match[i] || dfs(match[i]))
      {
        match[i] = x;
        return 1;
      }
    }
  return 0;
}

void dfs2(int x)
{
  rok[x] = 1;
  REPP(i, 1, n)
    if (w[x][i] && !cok[i])
    {
      cok[i] = 1;
      dfs2(match[i]);
    }
}

int main()
{
  freopen("in.txt", "r", stdin);
  for (;;)
  {
    scanf("%d%d%d", &n, &m, &tot);
    if (n + m + tot == 0) break;
    MST(w, 0);
    REPP(i, 1, tot)
    {
      int u, v;
      scanf("%d%d", &u, &v);
      w[u][v] = 1;
    }
    MST(match, 0);
    int ans = 0;
    REPP(i, 1, n)
    {
      MST(vis, 0);
      if (dfs(i)) ++ans;
    }
    printf("%d", ans);
    MST(vis, 0);
    MST(rok, 0);
    MST(cok, 0);
    REPP(i, 1, n)
      vis[match[i]] = 1;
    REPP(i, 1, n)
      if (!vis[i])
        dfs2(i);
    REPP(i, 1, n)
      if (!rok[i])
        printf(" r%d", i);
    REPP(i, 1, n)
      if (cok[i])
        printf(" c%d", i);
    printf("\n");
  }
  return 0;
}
```

Listing 3.8: uva11419.cpp

3.9 带花树

3.9.1 普通图最大匹配

```
/*
  解决一般图的最大匹配问题 O(N^3)
*/

#include <bits/stdc++.h>
#define MAXE 250*250*2
#define MAXN 250
#define SET(a,b) memset(a,b,sizeof(a))

using namespace std;
//g[i][j]存放关系图：i,j是否有边,match[i]存放i所匹配的点
bool g[MAXN][MAXN],inque[MAXN],inblossom[MAXN];
int match[MAXN],pre[MAXN],base[MAXN];

queue<int> Q;

//找公共祖先
int lca(int u,int v) {
    bool inpath[MAXN]= {false};
    while(1) {
        u=base[u];
        inpath[u]=true;
        if(match[u]==-1)break;
        u=pre[match[u]];
    }
    while(1) {
        v=base[v];
        if(inpath[v])return v;
        v=pre[match[v]];
    }
}

//压缩花
void reset(int u,int anc) {
    while(u!=anc) {
        int v=match[u];
        inblossom[base[u]]=1;
        inblossom[base[v]]=1;
        v=pre[v];
        if(base[v]!=anc)pre[v]=match[u];
        u=v;
    }
}

void contract(int u,int v,int n) {
    int anc = lca(u,v);
    //SET(inblossom,0);
    memset(inblossom,0,sizeof(inblossom));
    reset(u,anc);
    reset(v,anc);
    if(base[u]!=anc)pre[u]=v;
    if(base[v]!=anc)pre[v]=u;
    for(int i=1; i<=n; i++)
        if(inblossom[base[i]]) {
            base[i]=anc;
            if(!inque[i]) {
                Q.push(i);
                inque[i]=1;
            }
        }
}

bool dfs(int S,int n) {
    for(int i=0; i<=n; i++)
        pre[i]=-1 , inque[i]=0 , base[i]=i;
    while(Q.size())Q.pop();
    Q.push(S);
    inque[S]=1;
    while(!Q.empty()) {
        int u=Q.front();
        Q.pop();
        for(int v=1; v<=n; v++) {
            if(g[u][v]&&base[v]!=base[u]&&match[u]!=v) {
                if(v==S||(match[v]!=-1&&pre[match[v]]!=-1))
                    contract(u,v,n);
                else if(pre[v]==-1) {
                    pre[v]=u;
                    if(match[v]!=-1)
                        Q.push(match[v]),inque[match[v]]=1;
                    else {
                        u=v;
                        while(u!=-1) {
                            v=pre[u];
                            int w=match[v];
                            match[u]=v;
                            match[v]=u;
                            u=w;
                        }
                        return true;
                    }
                }
            }
        }    }
    return false;
}

int main() {

#ifndef ONLINE_JUDGE
    freopen("sum.in","r",stdin);
    //freopen("sum.out","w",stdout);
#endif

    int n,a,b,ans,i;
    while(scanf("%d",&n)!=EOF) {
        ans=0;          //最多有几对匹配
        memset(match,-1,sizeof(match));
        memset(g,0,sizeof(g));
        while(scanf("%d%d",&a,&b)!=EOF&&a!=0)
            g[a][b] = g[b][a] = 1;
        for(i=1; i<=n; i++)
            if(match[i]==-1&&dfs(i,n))
                ans++;
        cout<<ans*2<<endl;
        for(i=1; i<=n; i++)
            if(match[i]!=-1) {
                printf("%d %d\n",i,match[i]);
                match[i] = match[match[i]] = -1;
            }
    }
    return 0;
}
```

Listing 3.9: ural1099.cpp

3.9.2 普通图最优匹配

```
/*
  input
  第一行两个正整数，n,m。保证 n≥2。

接下来 m 行，每行三个整数 v,u,w 表示第 v 个男生和第 u 个男生愿意组成小组，且能写出 w 万万行的代码。保证 1≤v,u≤n，保证 v≠u，保证同一对 v,u 不会出现两次（这里是无序对）。
output

第一行一个整数，表示总代码量最多是多少（单位是万万行）。

接下来一行 n 个整数，描述一组最优方案。第 v 个整数表示 v 号男生所在小组的另一个男生的编号。如果 v 号男生没有小组请输出 0。
*/

#include <iostream>
#include <cstdio>
#include <algorithm>
#include <vector>
using namespace std;

typedef long long s64;

const int INF = 2147483647;

const int MaxN = 400;
const int MaxM = 79800;

template <class T>
inline void tension(T &a, const T &b)
{
  if (b < a)
    a = b;
}
template <class T>
inline void relax(T &a, const T &b)
{
  if (b > a)
    a = b;
}
template <class T>
inline int size(const T &a)
{
  return (int)a.size();
}

inline int getint()
{
  char c;
  while (c = getchar(), '0' > c || c > '9');

  int res = c - '0';
```

```cpp
  while (c = getchar(), '0' <= c && c <= '9')
    res = res * 10 + c - '0';
  return res;
}

const int MaxNX = MaxN + MaxN;

struct edge
{
  int v, u, w;

  edge(){}
  edge(const int &_v, const int &_u, const int &_w)
    : v(_v), u(_u), w(_w){}
};

int n, m;
edge mat[MaxNX + 1][MaxNX + 1];

int n_matches;
s64 tot_weight;
int mate[MaxNX + 1];
int lab[MaxNX + 1];

int q_n, q[MaxN];
int fa[MaxNX + 1], col[MaxNX + 1];
int slackv[MaxNX + 1];

int n_x;
int bel[MaxNX + 1], blofrom[MaxNX + 1][MaxN + 1];
vector<int> bloch[MaxNX + 1];

inline int e_delta(const edge &e) // does not work inside blossoms
{
  return lab[e.v] + lab[e.u] - mat[e.v][e.u].w * 2;
}
inline void update_slackv(int v, int x)
{
  if (!slackv[x] || e_delta(mat[v][x]) < e_delta(mat[slackv[x]][x]))
    slackv[x] = v;
}
inline void calc_slackv(int x)
{
  slackv[x] = 0;
  for (int v = 1; v <= n; v++)
    if (mat[v][x].w > 0 && bel[v] != x && col[bel[v]] == 0)
      update_slackv(v, x);
}

inline void q_push(int x)
{
  if (x <= n)
    q[q_n++] = x;
  else
  {
    for (int i = 0; i < size(bloch[x]); i++)
      q_push(bloch[x][i]);
  }
}
inline void set_mate(int xv, int xu)
{
  mate[xv] = mat[xv][xu].u;
  if (xv > n)
  {
    edge e = mat[xv][xu];
    int xr = blofrom[xv][e.v];
    int pr = find(bloch[xv].begin(), bloch[xv].end(), xr) - bloch[xv].begin();
    if (pr % 2 == 1)
    {
      reverse(bloch[xv].begin() + 1, bloch[xv].end());
      pr = size(bloch[xv]) - pr;
    }

    for (int i = 0; i < pr; i++)
      set_mate(bloch[xv][i], bloch[xv][i ^ 1]);
    set_mate(xr, xu);

    rotate(bloch[xv].begin(), bloch[xv].begin() + pr, bloch[xv].end());
  }
}
inline void set_bel(int x, int b)
{
  bel[x] = b;
  if (x > n)
  {
    for (int i = 0; i < size(bloch[x]); i++)
      set_bel(bloch[x][i], b);
  }
}
```

```

inline void augment(int xv, int xu)
{
  while (true)
  {    int xnu = bel[mate[xv]];
    set_mate(xv, xu);
    if (!xnu)
      return;
    set_mate(xnu, bel[fa[xnu]]);
    xv = bel[fa[xnu]], xu = xnu;
  }
}
inline int get_lca(int xv, int xu)
{
  static bool book[MaxNX + 1];
  for (int x = 1; x <= n_x; x++)
    book[x] = false;
  while (xv || xu)
  {
    if (xv)
    {
      if (book[xv])
        return xv;
      book[xv] = true;
      xv = bel[mate[xv]];
      if (xv)
        xv = bel[fa[xv]];
    }
    swap(xv, xu);
  }
  return 0;
}

inline void add_blossom(int xv, int xa, int xu)
{
  int b = n + 1;
  while (b <= n_x && bel[b])
    b++;
  if (b > n_x)
    n_x++;

  lab[b] = 0;
  col[b] = 0;

  mate[b] = mate[xa];

  bloch[b].clear();
  bloch[b].push_back(xa);
  for (int x = xv; x != xa; x = bel[fa[bel[mate[x]]]])
    bloch[b].push_back(x), bloch[b].push_back(bel[mate[x]]), q_push
    (bel[mate[x]]);
  reverse(bloch[b].begin() + 1, bloch[b].end());
  for (int x = xu; x != xa; x = bel[fa[bel[mate[x]]]])
    bloch[b].push_back(x), bloch[b].push_back(bel[mate[x]]), q_push
    (bel[mate[x]]);

  set_bel(b, b);

  for (int x = 1; x <= n_x; x++)
  {
    mat[b][x].w = mat[x][b].w = 0;
    blofrom[b][x] = 0;
  }
  for (int i = 0; i < size(bloch[b]); i++)
  {
    int xs = bloch[b][i];
    for (int x = 1; x <= n_x; x++)
      if (mat[b][x].w == 0 || e_delta(mat[xs][x]) < e_delta(mat[b][
    x]))
        mat[b][x] = mat[xs][x], mat[x][b] = mat[x][xs];
    for (int x = 1; x <= n_x; x++)
      if (blofrom[xs][x])
        blofrom[b][x] = xs;
  }
  calc_slackv(b);
}
inline void expand_blossom1(int b) // lab[b] == 1
{
  for (int i = 0; i < size(bloch[b]); i++)
    set_bel(bloch[b][i], bloch[b][i]);

  int xr = blofrom[b][mat[b][fa[b]].v];
  int pr = find(bloch[b].begin(), bloch[b].end(), xr) - bloch[b].
    begin();
  if (pr % 2 == 1)
  {
    reverse(bloch[b].begin() + 1, bloch[b].end());
    pr = size(bloch[b]) - pr;
  }

  for (int i = 0; i < pr; i += 2)
  {
    int xs = bloch[b][i], xns = bloch[b][i + 1];
    fa[xs] = mat[xns][xs].v;
    col[xs] = 1, col[xns] = 0;
    slackv[xs] = 0, calc_slackv(xns);
    q_push(xns);  }
  col[xr] = 1;
  fa[xr] = fa[b];
  for (int i = pr + 1; i < size(bloch[b]); i++)
  {
    int xs = bloch[b][i];
    col[xs] = -1;
    calc_slackv(xs);
  }

  bel[b] = 0;
}
inline void expand_blossom_final(int b) // at the final stage
{
  for (int i = 0; i < size(bloch[b]); i++)
  {
    if (bloch[b][i] > n && lab[bloch[b][i]] == 0)
      expand_blossom_final(bloch[b][i]);
    else
      set_bel(bloch[b][i], bloch[b][i]);
  }
  bel[b] = -1;
}

inline bool on_found_edge(const edge &e)
{
  int xv = bel[e.v], xu = bel[e.u];
  if (col[xu] == -1)
  {
    int nv = bel[mate[xu]];
    fa[xu] = e.v;
    col[xu] = 1, col[nv] = 0;
    slackv[xu] = slackv[nv] = 0;
    q_push(nv);
  }
  else if (col[xu] == 0)
  {
    int xa = get_lca(xv, xu);
    if (!xa)
    {
      augment(xv, xu), augment(xu, xv);
      for (int b = n + 1; b <= n_x; b++)
        if (bel[b] == b && lab[b] == 0)
          expand_blossom_final(b);
      return true;
    }
    else
      add_blossom(xv, xa, xu);
  }
  return false;
}

bool match()
{
  for (int x = 1; x <= n_x; x++)
    col[x] = -1, slackv[x] = 0;

  q_n = 0;
  for (int x = 1; x <= n_x; x++)
    if (bel[x] == x && !mate[x])
      fa[x] = 0, col[x] = 0, slackv[x] = 0, q_push(x);
  if (q_n == 0)
    return false;

  while (true)
  {
    for (int i = 0; i < q_n; i++)
    {
      int v = q[i];
      for (int u = 1; u <= n; u++)
        if (mat[v][u].w > 0 && bel[v] != bel[u])
        {
          int d = e_delta(mat[v][u]);
          if (d == 0)
          {
            if (on_found_edge(mat[v][u]))
              return true;
          }
          else if (col[bel[u]] == -1 || col[bel[u]] == 0)
            update_slackv(v, bel[u]);
        }
    }

    int d = INF;
    for (int v = 1; v <= n; v++)
      if (col[bel[v]] == 0)
        tension(d, lab[v]);
    for (int b = n + 1; b <= n_x; b++)
```

```
   if (bel[b] == b && col[b] == 1)
    tension(d, lab[b] / 2);
  for (int x = 1; x <= n_x; x++)
   if (bel[x] == x && slackv[x])
   {    if (col[x] == -1)
     tension(d, e_delta(mat[slackv[x]][x]));
    else if (col[x] == 0)
     tension(d, e_delta(mat[slackv[x]][x]) / 2);
   }

  for (int v = 1; v <= n; v++)
  {
   if (col[bel[v]] == 0)
    lab[v] -= d;
   else if (col[bel[v]] == 1)
    lab[v] += d;
  }
  for (int b = n + 1; b <= n_x; b++)
   if (bel[b] == b)
   {
    if (col[bel[b]] == 0)
     lab[b] += d * 2;
    else if (col[bel[b]] == 1)
     lab[b] -= d * 2;
   }

  q_n = 0;
  for (int v = 1; v <= n; v++)
   if (lab[v] == 0) // all unmatched vertices' labels are zero! cheers!
    return false;
  for (int x = 1; x <= n_x; x++)
   if (bel[x] == x && slackv[x] && bel[slackv[x]] != x && e_delta(mat[slackv[x]][x]) == 0)
   {
    if (on_found_edge(mat[slackv[x]][x]))
     return true;
   }
  for (int b = n + 1; b <= n_x; b++)
   if (bel[b] == b && col[b] == 1 && lab[b] == 0)
    expand_blossom1(b);
 }
 return false;
}

void calc_max_weight_match()
{
 for (int v = 1; v <= n; v++)
  mate[v] = 0;

 n_x = n;
 n_matches = 0;
 tot_weight = 0;

 bel[0] = 0;
 for (int v = 1; v <= n; v++)
  bel[v] = v, bloch[v].clear();
 for (int v = 1; v <= n; v++)
  for (int u = 1; u <= n; u++)
   blofrom[v][u] = v == u ? v : 0;

 int w_max = 0;
 for (int v = 1; v <= n; v++)
  for (int u = 1; u <= n; u++)
   relax(w_max, mat[v][u].w);
 for (int v = 1; v <= n; v++)
  lab[v] = w_max;

 while (match())
  n_matches++;

 for (int v = 1; v <= n; v++)
  if (mate[v] && mate[v] < v)
   tot_weight += mat[v][mate[v]].w;
}

int main()
{
 n = getint(), m = getint();

 for (int v = 1; v <= n; v++)
  for (int u = 1; u <= n; u++)
   mat[v][u] = edge(v, u, 0);

 for (int i = 0; i < m; i++)
 {
  int v = getint(), u = getint(), w = getint();
  mat[v][u].w = mat[u][v].w = w;
 }

 calc_max_weight_match();

```

```
printf("%lld\n", tot_weight);
for (int v = 1; v <= n; v++)
  printf("%d ", mate[v]);
printf("\n");
return 0;}
```

Listing 3.10: uoj81.cpp

3.10 最大团 TODO

3.11 欧拉理论 TODO

Chapter 4

数据结构

4.1 左偏树 TODO

4.2 splay TODO

4.3 lct TODO

4.4 可持久化线段树以及 LCA 不能再写错了 !!!

本题要求路径上 k 大

```cpp
#include <bits/stdc++.h>
#define MAXN 112345
#define MAXNODE 5012345

using namespace std;
typedef int arrayN[MAXN * 2];

arrayN e, nxt, fir;
int num, tot;

struct segmentNode
{
  segmentNode *l, *r;
  int low, up, num;
}tree[MAXNODE];

struct node
{
  int val, dep;
  int f[25];
  segmentNode *rt;
} a[MAXN];


void link(int u, int v)
{
  e[++num] = v, nxt[num] = fir[u];
  fir[u] = num;
}

segmentNode *build(int l, int r)
{
  segmentNode *tp = &tree[tot++];
  int mid = l + r >> 1;
  tp->low = l, tp->up = r;
  tp->num = 0;
  tp->l = tp->r = NULL;
  if (l == r) return tp;
  tp->l = build(l, mid);
  tp->r = build(mid + 1, r);
  return tp;
}

segmentNode *change(segmentNode *u, int x)
{
  segmentNode *tp = &tree[tot++];
  tp->l = u->l, tp->r = u->r;
  tp->num = u->num + 1;
  tp->low = u->low, tp->up = u->up;
  int mid = tp->up + tp->low >> 1;
  if (tp->low == tp->up) return tp;
  if (x <= mid) tp->l = change(u->l, x);
  else tp->r = change(u->r, x);
  return tp;
}

void dfs(int x, int fa, int depth)
{
  a[x].dep = depth;
  a[x].f[0] = fa;
  a[x].rt = change(a[fa].rt, a[x].val);
  for (int p = fir[x]; p; p = nxt[p])
    if (e[p] != fa)
      dfs(e[p], x, depth + 1);
}
```

```cpp

void initLCA(int n)
{
  for (int i = 1; i <= 20; ++i)    for (int j = 1; j <= n; ++j)
      a[j].f[i] = a[a[j].f[i - 1]].f[i - 1];
}

int getLCA(int u, int v)
{
  if (a[u].dep < a[v].dep) swap(u, v);
  int dt = a[u].dep - a[v].dep;
  for (int i = 20; i >= 0 && dt; i--)
    if (a[u].f[i] && ((1<< i) <= dt))
    {
      u = a[u].f[i];
      dt -= (1 << i);
    }
  if (u == v) return u;
  for (int i = 20; i >= 0; --i)
    if (a[u].f[i] != a[v].f[i])
      u = a[u].f[i], v = a[v].f[i];
  return a[u].f[0];
}

int ask(int u, int v, int lca, int k)
{
  int fa = a[lca].f[0];
  segmentNode *lk1l = a[u].rt, *lk1r = a[lca].rt;
  segmentNode *lk2l = a[v].rt, *lk2r = a[fa].rt;
  for (; ;)
  {
    if (lk1l->low == lk1l->up) return lk1l->low;
    int tmp = lk1l->l->num - lk1r->l->num + lk2l->l->num - lk2r->l->num;
    if (tmp >= k)
    {
      lk1l = lk1l->l, lk1r = lk1r->l;
      lk2l = lk2l->l, lk2r = lk2r->l;
    }else
    {
      k -= tmp;
      lk1l = lk1l->r, lk1r = lk1r->r;
      lk2l = lk2l->r, lk2r = lk2r->r;
    }
  }
}

vector<int> vec;

int main()
{
  freopen("in.txt", "r", stdin);
  int n, m;
  scanf("%d%d", &n, &m);
  for (int i = 1; i <= n; ++i)
  {
    scanf("%d", &a[i].val);
    vec.push_back(a[i].val);
  }
  sort(vec.begin(), vec.end());
  vec.resize(unique(vec.begin(), vec.end()) - vec.begin());
  for (int i = 1; i <= n; ++i)
    a[i].val = lower_bound(vec.begin(), vec.end(), a[i].val) - vec.begin();
  for (int i = 1; i < n; ++i)
  {
    int u, v;
    scanf("%d%d", &u, &v);
    link(u, v);
    link(v, u);
  }
  a[0].rt = build(0, n);
  dfs(1, 0, 1);
  initLCA(n);
  for (int i = 1; i <= m; ++i)
  {
    int u, v, k;
    scanf("%d%d%d", &u, &v, &k);
    int lca = getLCA(u, v);
    printf("%d\n", vec[ask(u, v, lca, k)]);
  }
  return 0;
}
```

Listing 4.1: COT.cpp

4.5 点分治

```cpp
#include <cstdlib>
#include <cstdio>
#include <iostream>
```

```cpp
#include <vector>
#include <cstring>
#include <algorithm>
#define REP(i, n) for(int i = 0; i < (int) (n); ++i)
#define REPP(i, a, b) for(int i = (int) (a); i <= (int) (b); ++i)
#define MST(a, b) memset(a, (b), sizeof(a))
#define MAXN 11111
//小于等于k的点对
using namespace std;
typedef int arrayN[MAXN *2];

arrayN fir, nxt, e, c, sizeN, vis;
int n, k, ans, num;
vector<int> stRoot, stEp;


void link(int u, int v, int w)
{
    e[++num] = v, nxt[num] = fir[u], fir[u] = num;
    c[num] = w;
}

int dfsSize(int x, int fa)
{
    sizeN[x] = 1;
    for (int p = fir[x]; p; p = nxt[p])
        if (e[p] != fa && !vis[e[p]])
            sizeN[x] += dfsSize(e[p], x);
    return sizeN[x];
}

int getRoot(int x, int fa, int totN)
{
    int maxSize = totN - sizeN[x];
    for (int p = fir[x]; p; p = nxt[p])
        if (e[p] != fa && !vis[e[p]])
        {
            maxSize = max(maxSize, sizeN[e[p]]);
            int tmp = getRoot(e[p], x, totN);
            if (tmp) return tmp;
        }
    if (maxSize <= totN / 2) return x;
    return 0;
}

void dfsSt(int x, int fa, int len)
{
    stEp.push_back(len);
    for (int p = fir[x]; p; p = nxt[p])
        if (!vis[e[p]] && e[p] != fa)
            dfsSt(e[p], x, len +c[p]);
}

int calc(vector<int> &st)
{
    int tmp = 0;
    sort(st.begin(), st.end());
    int L = 0, R = st.size() - 1;
    for (;L < R;)
    {
        if (st[L] +st[R] <= k) tmp += R - L, L++;
        else --R;
    }
    return tmp;
}

void solve(int x)
{
    int root = getRoot(x, x, dfsSize(x, x));
    vis[root] = 1;
    stRoot.clear();
    stRoot.push_back(0);
    for (int p = fir[root]; p; p = nxt[p])
        if (!vis[e[p]])
        {
            stEp.clear();
            dfsSt(e[p], root, c[p]);
            ans -= calc(stEp);
            REP(i, stEp.size())
                stRoot.push_back(stEp[i]);
        }
    ans += calc(stRoot);
    for (int p = fir[root]; p; p = nxt[p])
        if (!vis[e[p]]) solve(e[p]);
    vis[root] = 0;
}

int main()
{
    freopen("in.txt", "r", stdin);
    for (;;)
    {
        scanf("%d%d", &n, &k);
        if (n + k == 0) break;
        ans = 0;
        num = 0;
        MST(fir, 0);        REPP(i, 1, n - 1)
        {
            int u, v, w;
            scanf("%d%d%d", &u, &v, &w);
            link(u, v, w);
            link(v, u, w);
        }
        MST(vis, 0);
        ans = 0;
        solve(1);
        printf("%d\n", ans);
    }
    return 0;
}
```

Listing 4.2: poj1741.cpp

树上 A 权值不超过 lim 的 B 权值和最大的路径

```
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include <vector>
#include <cstring>
#include <algorithm>
#define REP(i, n) for(int i = 0; i < (int) (n); ++i)
#define REPP(i, a, b) for(int i = (int) (a); i <= (int) (b); ++i)
#define MST(a, b) memset(a, (b), sizeof(a))
#define MAXN 11111
//小于等于k的点对
using namespace std;
typedef int arrayN[MAXN *2];

arrayN fir, nxt, e, c, sizeN, vis;
int n, k, ans, num;
vector<int> stRoot, stEp;


void link(int u, int v, int w)
{
    e[++num] = v, nxt[num] = fir[u], fir[u] = num;
    c[num] = w;
}

int dfsSize(int x, int fa)
{
    sizeN[x] = 1;
    for (int p = fir[x]; p; p = nxt[p])
        if (e[p] != fa && !vis[e[p]])
            sizeN[x] += dfsSize(e[p], x);
    return sizeN[x];
}

int getRoot(int x, int fa, int totN)
{
    int maxSize = totN - sizeN[x];
    for (int p = fir[x]; p; p = nxt[p])
        if (e[p] != fa && !vis[e[p]])
        {
            maxSize = max(maxSize, sizeN[e[p]]);
            int tmp = getRoot(e[p], x, totN);
            if (tmp) return tmp;
        }
    if (maxSize <= totN / 2) return x;
    return 0;
}

void dfsSt(int x, int fa, int len)
{
    stEp.push_back(len);
    for (int p = fir[x]; p; p = nxt[p])
        if (!vis[e[p]] && e[p] != fa)
            dfsSt(e[p], x, len +c[p]);
}

int calc(vector<int> &st)
{
    int tmp = 0;
    sort(st.begin(), st.end());
    int L = 0, R = st.size() - 1;
    for (;L < R;)
    {
        if (st[L] +st[R] <= k) tmp += R - L, L++;
        else --R;
    }
    return tmp;
}

```

```cpp
void solve(int x)
{
    int root = getRoot(x, x, dfsSize(x, x));
    vis[root] = 1;
    stRoot.clear();    stRoot.push_back(0);
    for (int p = fir[root]; p; p = nxt[p])
        if (!vis[e[p]])
        {
            stEp.clear();
            dfsSt(e[p], root, c[p]);
            ans -= calc(stEp);
            REP(i, stEp.size())
                stRoot.push_back(stEp[i]);
        }
    ans += calc(stRoot);
    for (int p = fir[root]; p; p = nxt[p])
        if (!vis[e[p]]) solve(e[p]);
    vis[root] = 0;
}

int main()
{
    freopen("in.txt", "r", stdin);
    for (;;)
    {
        scanf("%d%d", &n, &k);
        if (n + k == 0) break;
        ans = 0;
        num = 0;
        MST(fir, 0);
        REPP(i, 1, n - 1)
        {
            int u, v, w;
            scanf("%d%d%d", &u, &v, &w);
            link(u, v, w);
            link(v, u, w);
        }
        MST(vis, 0);
        ans = 0;
        solve(1);
        printf("%d\n", ans);
    }
    return 0;
}
```

Listing 4.3: poj1741.cpp

4.6 树链剖分 TODO

4.7 qtree TODO

Chapter 5

其他算法

5.1 pq 树 TODO

5.2 DLX TODO

5.3 对抗搜索 TODO

5.4 cdq 分治与读入优化

- 不要排结构体，因为排结构体到时候还要排回来。
- 线段树打时间戳不要 memsize();
- 在严格小的限制下，第二维排序的时候一定要双关键字排序
- 这题是三维空间中，三个坐标都不减的最长链

```cpp
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <algorithm>
#define REP(i, n) for(int i = 0; i < (int) (n); ++i)
#define REPP(i, a, b) for(int i = (int) (a); i <= (int) (b); ++i)
#define REDD(i, a, b) for(int i = (int) (a); i >= (int) (b); —i)
#define MST(a, b) memset((a), (b), sizeof(a))
#define MAXN 111111
#include <vector>

using namespace std;
int zLim;

long long gTot[MAXN *4];
int t, g[MAXN *4], n, ti[MAXN *4], now;
struct node
{
    int x, y, z, f;
    long long tot;
} a[MAXN];

int comx(node A, node B)
{
    return (A.x < B.x) || ((A.x == B.x) && (A.y < B.y)) || ((A.x == B.x) && (A.y == B.y) && A.z < B.z);
}

int comy(node A, node B)
{
    return A.y < B.y;
}

void change(int pos, int x, long long cnt)
{
    pos += t;
    if (ti[pos] != now) g[pos] = gTot[pos] = 0;
    if (x < g[pos]) return ;
    if (x == g[pos]) gTot[pos] += cnt;
    else gTot[pos] = cnt, g[pos] = x;
    ti[pos] = now;

    for(pos >>= 1; pos; pos >>= 1)
    {
        if (ti[pos <<1] != now) g[pos <<1] = gTot[pos <<1] = 0;
        if (ti[pos <<1 ^1] != now) g[pos <<1 ^ 1] = gTot[pos <<1 ^1] = 0;
        ti[pos] = now;
        g[pos] = max(g[pos <<1], g[pos << 1 ^1]);
        gTot[pos] = 0;
        if (g[pos] == g[pos <<1]) gTot[pos] += gTot[pos <<1];
        if (g[pos] == g[pos <<1 ^1]) gTot[pos] += gTot[pos <<1 ^1];
    }
}

int ask(int l, int r, long long &cnt)
{
    if (l > r) return 0;
    int tmp = 0;
    cnt = 0;
    l += t - 1, r += t + 1;
    for (;(l ^ r) != 1; l >>= 1, r >>= 1)
    {        if (!(l &1))
        {
            if (ti[l +1] == now)
      {
            if (tmp == g[l +1]) cnt += gTot[l +1];
            else if (tmp <g[l +1])
            {
                tmp = g[l +1];
                cnt = gTot[l +1];
            }
      }
        }
        if (r &1)
        {
            if (ti[r - 1] == now)
      {
            if (tmp == g[r - 1]) cnt += gTot[r - 1];
            else if (tmp < g[r - 1])
            {
                tmp = g[r - 1];
                cnt = gTot[r - 1];
            }
      }
        }
    }
    return tmp;
}


void solve(int l, int r)
{
    if (l == r) return ;
    int mid = (l +r) >>1;
    solve(mid +1, r);

    sort(a +mid +1, a +r +1, comy);
    sort(a +l, a +mid +1, comy);

    // MST(g, 0);
    //MST(gTot, 0);
    ++now;
    int pos = r +1;
    REDD(i, mid, l)
    {
        for (;pos > mid +1 && a[pos - 1].y >= a[i].y; --pos)
        {
            change(a[pos - 1].z, a[pos - 1].f, a[pos - 1].tot);
        }

        long long tmpTot;
        int tmp = ask(a[i].z, zLim, tmpTot) +1;
        if (a[i].f == tmp) a[i].tot += tmpTot;
        else if (a[i].f < tmp)
        {
            a[i].f = tmp;
            a[i].tot = tmpTot;
        }
    }

    sort(a + l, a +r +1, comx);
    solve(l, mid);
}

int INT()
{
    int res;
    char ch;
    while (ch = getchar(), !isdigit(ch));
    for (res = ch - '0'; ch = getchar(), isdigit(ch);)
        res = res * 10 + ch - '0';
    return res;
}

int main()
{
    int task;
    freopen("in.txt", "r", stdin);
    now = 0;
    for (task = INT(); task; --task)
    {
        n = INT();
        vector <int> dataZ;
        REPP(i, 1, n)
        {
            a[i].x = INT();
            a[i].y = INT();
            a[i].z = INT();
            a[i].f = 1;
            a[i].tot = 1;
            dataZ.push_back(a[i].z);
        }
        sort(dataZ.begin(), dataZ.end());       dataZ.resize(unique(dataZ.begin(), dataZ.end()) - dataZ.begin());
        REPP(i, 1, n)
        {
            a[i].z = (lower_bound(dataZ.begin(), dataZ.end(), a[i].z) - dataZ.begin()) +1;
        }
        zLim = dataZ.size();
        for (t = 1; t <= zLim + 1; t <<= 1);
        sort(a +1, a + n +1, comx);
        solve(1, n);
        int ans = 0;
        long long cnt = 0;
        REPP(i, 1, n)
        {
            if (ans == a[i].f) cnt += a[i].tot;
            else if (ans < a[i].f) cnt = a[i].tot, ans = a[i].f;
        }
        printf("%d %lld\n", ans, cnt);
    }
    return 0;
}
```

Listing 5.1: hdu4742.cpp