

ACM/ICPC Template

AsZ VincentLDL

October 12, 2015

Contents

1	java TODO	2	3.6	最小树形图	16
1.1	读写	2	3.7	二分图	17
1.2	高精	2	3.7.1	普通 KM	17
2	dp 优化	3	3.7.2	牛逼 KM	18
2.1	决策单调性优化	3	3.7.3	常见问题汇总	19
2.2	单调队列优化以及写仙人掌图	4	3.7.4	最小点覆盖输出方案	20
2.3	斜率优化	6	3.8	带花树	21
2.3.1	斜率以及 x 维都单调	6	3.8.1	普通图最大匹配	21
2.3.2	随便什么情况: cdq 分治优化	7	3.8.2	普通图最优匹配	22
3	图论	9	3.9	最大团 TODO	27
3.1	tarjan TODO	9	3.10	欧拉理论 TODO	27
3.1.1	2-sat	9	4	数据结构	28
3.1.2	割顶, 点双联通分量 TODO	11	4.1	左偏树 TODO	28
3.1.3	桥, 边双联通分量 TODO	11	4.2	splay TODO	28
3.2	平面图 TODO	11	4.3	lct TODO	28
3.3	最佳追捕算法	11	4.4	可持久化线段树以及 LCA 不能再写错了 !!!	28
3.4	网络流 TODO	11	4.5	点分治	29
3.4.1	dinic	11	4.6	树链剖分 TODO	32
3.4.2	费用流 TODO	13	4.7	qtree TODO	32
3.4.3	常见模型 TODO	13	5	其他算法	33
3.5	弦图	13	5.1	pq 树 TODO	33
3.5.1	做法与常见问题	13	5.2	DLX TODO	33
3.5.2	万不得已用线性作法	14	5.3	对抗搜索 TODO	33
3.5.3	$n \log n$ 好写得	15	5.4	cdq 分治与读入优化	33

Chapter 1

java

```
1 import java.io.*;
2 import java.math.*;
3 import java.util.*;
4 import java.text.*;
5
6 class point
7 {
8     int A, B;
9     double C;
10    public point(int a, int b)
11    {
12        this.A = a; this.B = b;
13        if (b == 0) this.C = 1e20;
14        else this.C = 1.0 * a / b;
15    }
16 }
17 };
18
19 public class Main
20 {
21     public static int N, V;
22     public static int[] A = new int[44];
23     public static int[] B = new int[44];
24     public static double[] C = new double[44];
25     public static final int MAXN = 44;
26     public static point[] g = new point[MAXN];
27
28     public static void main(String[] args)
29     {
30
```

```
31         Comparator<point> comparator = new Comparator<point>(){
32             public int compare(point s1, point s2) {
33                 return (s1.C < s2.C) ? -1 : 1;
34             }
35         };
36
37         Scanner cin = new Scanner(new BufferedInputStream(System.in
38 ));
39         PrintWriter out = new PrintWriter(new OutputStreamWriter(System.
40 out));
41         int task = cin.nextInt();
42         for (; task > 0; --task)
43         {
44             N = cin.nextInt();
45             V = cin.nextInt();
46             for (int i = 0; i < N; ++i)
47                 A[i] = cin.nextInt();
48             int flag = 1;
49             for (int i = 0; i < N; ++i)
50             {
51                 B[i] = cin.nextInt();
52                 if (B[i] != 0 && A[i] >= V)
53                     flag = 0;
54             }
55             if (flag == 0)
56             {
57                 out.println(-1);
58                 continue;
59             }
60             for (int i = 0; i < N; ++i)
61                 g[i] = new point(A[i], B[i]);
62             Arrays.sort(g, 0, N, comparator);
63             BigDecimal ans = BigDecimal.ZERO;
64             for (int i = N - 1; i >= 0; --i)
65             {
66                 if (g[i].B == 0) continue;
67                 BigDecimal tmp = ans.multiply(BigDecimal.valueOf
68 (1.0 * g[i].A));
69                 tmp = tmp.add(BigDecimal.valueOf(1.0 * g[i].B));
70                 //System.out.println(tmp + " " + (1.0 * V - g[i].A)
71 );
72                 tmp = tmp.divide(BigDecimal.valueOf(1.0 * V - g[i].
73 A), 1000, BigDecimal.ROUND_HALF_UP);
74                 ans = ans.add(tmp);
75             }
76         }
77     }
78 }
```

```
71         out.println(ans.setScale(0, BigDecimal.ROUND_HALF_UP));
72         //保留0位小数
73         //    Arrays.sort
74         }
75     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： $w[i, j] + w[i + 1, j + 1] \leq w[i + 1, j] + w[i, j + 1]$, 则满足决策单调性。
- 有时候不满足决策单调性，但是去掉完全不合格状态之后可以满足。

```
1 #include <bits/stdc++.h>
2 #define MAXN 51234
3
4 using namespace std;
5 typedef long long arrayN[MAXN];
6
7 deque < pair< pair<int, int> , int> > deq;
8 arrayN f, sum, c;
9 long long L;
10
11 long long sqr(long long x)
12 {
13     return x * x;
14 }
15
16 long long trans(int l, int r)
17 {
18     return sqr(1LL * r - (l + 1) - L + sum[r] - sum[l]) + f[l];
19 }
```

```
20 int myLowBound(pair <int, int> pr, int ori, int now)
21 {
22     int l = pr.first, r = pr.second;
23     for (; l < r; )
24     {
25         int mid = l + r >> 1;
26         if (trans(ori, mid) <= trans(now, mid)) l = mid + 1;
27         else r = mid;
28     }
29     return l;
30 }
31
32 int main()
33 {
34     int n;
35     freopen("toys.in", "r", stdin);
36     cin >> n >> L;
37     for (int i = 1; i <= n; ++i)
38     {
39         cin >> c[i];
40         sum[i] = sum[i - 1] + c[i];
41     }
42     deq.push_back(make_pair(make_pair(1, n), 0));
43     for (int i = 1; i <= n; ++i)
44     {
45         for (; deq.front().first.second < i; deq.pop_front());
46         f[i] = trans(deq.front().second, i);
47         if (i == n) break;
48         deq.front().first.first = i + 1;
49         if (deq.front().first.second < i + 1) deq.pop_front();
50         for (; !deq.empty() && trans(deq.back().second, deq.back().
51 first.first) >= trans(i, deq.back().first.first); deq.pop_back
52 ());
53         if (deq.empty()) deq.push_back(make_pair(make_pair(i + 1, n
54 ), i));
55         else
56         {
57             int x = myLowBound(deq.back().first, deq.back().second,
58 i);
59             if (trans(i, x) >= (trans(deq.back().second, x))) x++;
60             deq.back().first.second = x - 1;
61             if (x <= n) deq.push_back(make_pair(make_pair(x, n), i)
62 );
63         }
64     }
65 }
```

```

61     cout << f[n] << endl;
62     return 0;
63 }

```

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 $pt \rightarrow ph$ 就是基环了。

```

1  #include <bits/stdc++.h>
2  #define MAXN 1123456
3  #define MAXM 2123456
4
5  typedef int arrayN[MAXN], arrayM[MAXM];
6
7  using namespace std;
8
9  arrayN fir, cost, t, pre, vis;
10 arrayM e, nxt, c;
11 long long ans, dst[MAXN];
12 int num, now, visNow;
13
14 void link(int u, int v, int w)
15 {
16     e[++num] = v, nxt[num] = fir[u];
17     fir[u] = num, c[num] = w;
18 }
19
20 vector<int> bfsFindCycle(int x)
21 {
22     ++now;
23     vector<int> cyc;
24     deque<int> deq;
25     int pt = 0, ph = 0, last;
26     deq.push_back(x);
27     t[x] = now;
28     for (; !deq.empty() && !pt;)

```

```

29 {
30     int u = deq.front();
31     deq.pop_front();
32     for (int p = fir[u]; p && !pt; p = nxt[p])
33         if (e[p] != pre[u])
34             if (t[e[p]] == now)
35             {
36                 pt = u, ph = e[p];
37                 last = c[p];
38             }
39             else
40             {
41                 t[e[p]] = now;
42                 pre[e[p]] = u;
43                 cost[e[p]] = c[p];
44                 deq.push_back(e[p]);
45             }
46     }
47     vector<int> cycTmp;
48     if (pt)
49     {
50         ++now;
51         int tmp = pt;
52         for (; tmp != x; tmp = pre[tmp])
53             t[tmp] = now;
54         t[x] = now;
55         int lca = ph;
56         for (; t[lca] != now; lca = pre[lca]);
57         for (tmp = pt; tmp != lca; tmp = pre[tmp])
58         {
59             swap(last, cost[tmp]);
60             cyc.push_back(tmp);
61         }
62         cyc.push_back(lca);
63         cost[lca] = last;
64
65         for (tmp = ph; tmp != lca; tmp = pre[tmp])
66             cycTmp.push_back(tmp);
67         for (; !cycTmp.empty(); cycTmp.pop_back())
68             cyc.push_back(cycTmp.back());
69     } else cyc.push_back(x);
70
71     ++now;
72     for (int i = 0; i < cyc.size(); ++i)
73         t[cyc[i]] = now;
74     return cyc;

```

```

75 }
76
77 struct node
78 {
79     long long w;
80     long long lst, f;}g[MAXN * 2];
81
82 long long bfsLongest(int rt, int &nrt)
83 {
84     long long lst = 0;
85     deque <int> deq;
86     deq.push_back(rt);
87     nrt = rt;
88     vis[rt] = ++visNow;
89     dst[rt] = 0;
90     for (; !deq.empty(); )
91     {
92         int u = deq.front();
93         deq.pop_front();
94         for (int p = fir[u]; p; p = nxt[p])
95             if (vis[e[p]] != visNow && t[e[p]] != now)
96             {
97                 vis[e[p]] = visNow;
98                 dst[e[p]] = dst[u] + c[p];
99                 if (dst[e[p]] > lst)
100                 {
101                     lst = dst[e[p]];
102                     nrt = e[p];
103                 }
104                 deq.push_back(e[p]);
105             }
106     }
107     return lst;
108 }
109
110 long long solve(int x)
111 {
112     long long ans = 0;
113     vector <int> cyc = bfsFindCycle(x);
114     int n = cyc.size();
115     for (int i = 0; i < n; ++i)
116     {
117         int pt, pp;
118         t[cyc[i]] = 0;
119         g[i].lst = bfsLongest(cyc[i], pt);
120         ans = max(ans, bfsLongest(pt, pp));

```

```

121         t[cyc[i]] = now;
122         if (n == 1) return g[i].lst;
123         g[i].w = cost[cyc[i]];
124         g[i].f = 0;
125         g[i + n] = g[i];
126     }
127     g[0].w = 0;
128     for (int i = 1; i < 2 * n; ++i)
129         g[i].w += g[i - 1].w;
130     g[0].f = g[0].lst;
131     deque <int> deq;
132     deq.push_back(0);
133     for (int i = 1; i < 2 * n; ++i)
134     {
135         for (; deq.front() + n <= i; deq.pop_front());
136         g[i].f = g[i].lst + g[i].w + g[deq.front()].lst - g[deq.
front()].w;
137         for (; !deq.empty() && g[deq.back()].lst - g[deq.back()].w
<= g[i].lst - g[i].w; deq.pop_back());
138         deq.push_back(i);
139     }
140     for (int i = 0; i < 2 * n; ++i)
141         ans = max(ans, g[i].f);
142     return ans;
143 }
144
145 int main()
146 {
147     freopen("island.in", "r", stdin);
148     int n;
149     num = 1;
150     scanf("%d", &n);
151     for (int i = 1; i <= n; ++i)
152     {
153         int v, len;
154         scanf("%d%d", &v, &len);
155         link(i, v, len);
156         link(v, i, len);
157     }
158     long long ans = 0;
159     for (int i = 1; i <= n; ++i)
160         if (!vis[i])
161             ans += solve(i);
162     printf("%lld\n", ans);
163     return 0;

```


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

```

1 #include <deque>
2 #include <cstdio>
3 #include <cstring>
4 #include <iostream>
5 #include <cstdlib>
6
7 #define MAXN 512345
8
9 using namespace std;
10 typedef long long arrayN[MAXN];
11
12 struct node
13 {
14     long long x, y, f;
15     node (long long tx = 0, long long ty = 0, long long tf = 0)
16     {
17         x = tx, y = ty, f = tf;
18     }
19     //y = f + sum^2, x = sum
20 }g[MAXN];
21
22 long long sqr(long long x)
23 {

```

```

24     return x * x;
25 }
26
27 long long cross(long long x1, long long y1, long long x2, long long
    y2)
28 {
29     return x1 * y2 - x2 * y1;
30 }
31
32 deque < int > deq;
33
34 int main()
35 {
36     freopen("hdu3507.in", "r", stdin);
37     int N, M;
38     for (; scanf("%d%d", &N, &M) != EOF; )
39     {
40
41         deq.clear();
42         g[0] = node(0, 0, 0);
43         deq.push_back(0);
44         for (int i = 1; i <= N; ++i)
45         {
46             int x;
47             scanf("%d", &x);
48             g[i].x = g[i - 1].x + x;
49             long long lim = g[i].x << 1;
50             for (; deq.size() > 1; deq.pop_front())
51             {
52                 node u = g[deq[0]];
53                 node v = g[deq[1]];
54                 if ((v.y - u.y) > lim * (v.x - u.x))
55                     break;
56             }
57             node pt = g[deq.front()];
58             g[i].f = pt.f + sqr(g[i].x - pt.x) + M;
59             g[i].y = sqr(g[i].x) + g[i].f;
60             for (; deq.size() >= 2; deq.pop_back())
61             {
62                 node A = g[deq[deq.size() - 2]];
63                 node B = g[deq[deq.size() - 1]];
64                 node C = g[i];
65                 if (cross(B.x - A.x, B.y - A.y, C.x - B.x, C.y - B.
y) > 0) break;
66             }
67             deq.push_back(i);

```

```

68     }
69     cout << g[N].f << endl;
70 }
71 return 0;
72 }

```

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。

```

1 #include <bits/stdc++.h>
2 #define MST(a, b) memset((a), (b), sizeof(a))
3 #define MAXN 112345
4 #define esp 1e-8
5
6 using namespace std;
7
8 struct node
9 {
10     double A, B, rate; //A/B
11     double x, y;
12     double f;
13 }g[MAXN];
14
15 int lab[MAXN], a[MAXN];
16
17 int cmp(double x)
18 {
19     if (x < -esp) return -1;
20     if (x > esp) return 1;

```

```

21     return 0;
22 }
23
24 int smaller(int u, int v)
25 {
26     int tx = cmp(g[u].x - g[v].x);
27     int ty = cmp(g[u].y - g[v].y);
28     return tx < 0 || (tx == 0 && ty < 0);
29 }
30
31 void mergeSortX(int al, int ar, int bl, int br)
32 {
33     int Na = 0;
34     for (int i = al; i <= ar; ++i)
35     {
36         while (bl <= br && smaller(lab[bl], lab[i]))
37             a[++Na] = lab[bl++];
38         a[++Na] = lab[i];
39     }
40     for (; bl <= br; ++bl)
41         a[++Na] = lab[bl];
42     for (int i = 1; i <= Na; ++i)
43         lab[al + i - 1] = a[i];
44 }
45
46 double cross(int A, int B, int C)
47 {
48     return (g[B].x - g[A].x) * (g[C].y - g[B].y) - (g[B].y - g[A].y)
49         * (g[C].x - g[B].x);
50 }
51
52 double comRate(int A, int B, int C)
53 {
54     return (g[B].y - g[A].y) * g[C].B + g[C].A * (g[B].x - g[A].x);
55 }
56
57 void getRightPartF(int al, int ar, int bl, int br)
58 {
59     int Na = 0;
60     double lim = 0;
61     for (int i = al; i <= ar; ++i)
62     {
63         lim = max(lim, g[lab[i]].f);
64         while (Na >= 2 && cmp(cross(a[Na - 1], a[Na], lab[i])) >=
0)

```

```

65     —Na;
66     a[++Na] = lab[i];
67 }
68 int La = 1;
69 for (int i = bl; i <= br; ++i)
70 {
71     int p = lab[i];      g[p].f = max(g[p].f, lim);
72     for (; La + 1 <= Na && cmp(comRate(a[La], a[La + 1], p)) >= 16
73         0; ++La);
74     g[p].f = max(g[p].f, g[a[La]].x * g[p].A + g[a[La]].y * g[p].B);
75 }
76
77 void solve(int l, int r)
78 {
79     if (l == r)
80     {
81         int p = lab[l];
82         //g[p].f = max(g[p].f, g[p - 1].f);
83         g[p].x *= g[p].f;
84         g[p].y *= g[p].f;
85         return ;
86     }
87     int Na = r - l + 1;
88     int upLim = 0, downLim = MAXN;
89     for (int i = l; i <= r; ++i)
90     {
91         upLim = max(upLim, lab[i]);
92         downLim = min(downLim, lab[i]);
93     }
94     int midLim = (upLim + downLim) >> 1;
95     int pLow = 0;
96     for (int i = l; i <= r; ++i)
97         if (lab[i] <= midLim)
98             a[++pLow] = lab[i];
99     int pHigh = pLow;
100    for (int i = l; i <= r; ++i)
101        if (lab[i] > midLim)
102            a[++pHigh] = lab[i];
103    for (int i = 1; i <= Na; ++i)
104        lab[i + l - 1] = a[i];
105    pLow += l - 1;
106    solve(l, pLow);
107    getRightPartF(l, pLow, pLow + 1, r);
108    solve(pLow + 1, r);
109
110    mergeSortX(l, pLow, pLow + 1, r);
111 }
112 int com(int u, int v)
113 {
114     node tu = g[u];
115     node tv = g[v];
116     return tu.A * tv.B < tv.A * tu.B;
117 }
118
119 int main()
120 {
121     // freopen("cash4.in", "r", stdin);
122     int N, S;
123     scanf("%d%d", &N, &S);
124     for (int i = 1; i <= N; ++i)
125     {
126         scanf("%lf%lf%lf", &g[i].A, &g[i].B, &g[i].rate);
127         g[i].y = 1.0 / (g[i].B + g[i].A * g[i].rate);
128         g[i].x = g[i].y * g[i].rate;
129         g[i].f = S;
130         lab[i] = i;
131     }
132     g[1].f = S;
133     sort(lab + 1, lab + N + 1, com);
134     solve(1, N);
135     double ans = 0;
136     for (int i = 1; i <= N; ++i)
137         ans = max(ans, g[i].f);
138     printf("%.3f\n", ans);
139     return 0;
140 }

```

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

```
1 //指定小写字母元音/辅音
2 //给出第i个位置是元音/辅音蕴涵j位置元音/辅音
3 //给定字符串st,求字典序不小于它的最小的合法2-sat方案
4 #include <bits/stdc++.h>
5 #define MAXN 500
6 #define MAXM 512345
7
8 using namespace std;
9 typedef int arrayN[MAXN], arrayM[MAXM];
10
11 char g[30], st[MAXN];
12 arrayN fir0, low, dfn, inVec, cnt, belong;
13 arrayN deg, con0, con1, fir1, topOrder, col;
14 arrayM e0, nxt0, e1, nxt1;
15 int num, now, tot, nextAlp[30][2], firAlp[2];
16 vector<int> vec;
17
18 int getKind(char ch) {
```

```
19     if (ch == 'V') return 0;
20     else return 1;
21 }
22
23 void link0(int u, int v) {
24     e0[++num] = v, nxt0[num] = fir0[u];
25     fir0[u] = num;
26 }
27
28 void link1(int u, int v) {
29     e1[++num] = v, nxt1[num] = fir1[u];
30     fir1[u] = num;
31 }
32
33 void tarjan(int x) {
34     low[x] = dfn[x] = ++now;
35     vec.push_back(x);
36     for (int p = fir0[x], q; p; p = nxt0[p])
37         if (!inVec[q = e0[p]])
38             if (!dfn[e0[p]]) {
39                 tarjan(e0[p]);
40                 low[x] = min(low[x], low[e0[p]]);
41             } else low[x] = min(low[x], dfn[e0[p]]);
42     if (low[x] == dfn[x]) {
43         cnt[belong[x] = ++tot] = 1;
44         inVec[x] = 1;
45         for (; vec.back() != x; vec.pop_back()) {
46             int q = vec.back();
47             inVec[q] = 1;
48             cnt[belong[q] = tot]++;
49         }
50         vec.pop_back();
51     }
52 }
53
54 void topSort() {
55     int l = 1, r = 0;
56     for (int i = 1; i <= tot; ++i)
57         if (deg[i] == 0) topOrder[++r] = i;
58     for (; l <= r; ++l) {
59         int u = topOrder[l];
60         for (int p = fir1[u]; p; p = nxt1[p]) {
61             --deg[e1[p]];
62             if (deg[e1[p]] == 0) topOrder[++r] = e1[p];
63         }
64     }
```

```

65 }
66 int getDAG(int n) {
67     for (int i = 1; i <= n * 2; ++i)
68         dfn[i] = low[i] = belong[i] = inVec[i] = deg[i] = 0;
69     now = tot = num = 0;    for (int i = 1; i <= n * 2; ++i)
70         if (!dfn[i]) tarjan(i);
71     for (int i = 1; i <= n; ++i)
72         if (belong[i] == belong[con0[i]]) return 0;
73     for (int i = 1; i <= 2 * n; ++i) {
74         for (int p = fir0[i]; p; p = nxt0[p]) {
75             int q = e0[p];
76             if (belong[i] == belong[q]) continue;
77             link1(belong[q], belong[i]);
78             deg[belong[i]]++;
79         }
80         con1[belong[i]] = belong[con0[i]];
81         con1[belong[con0[i]]] = belong[i];
82     }
83     topSort();
84     return 1;
85 }
86
87 int dye(int x, int co) {
88     if (col[x]) {
89         return (co == col[x]);
90     }
91     col[x] = co;
92     for (int p = fir1[x]; p; p = nxt1[p])
93         if (!dye(e1[p], co)) return 0;
94     return 1;
95 }
96
97 int originDye(int p, int n) {
98     int all = -1;
99     if (firAlp[0] > 'z') all = 0;
100    if (firAlp[1] > 'z') all = 1;
101    if (all >= 0)
102        for (int i = 1; i <= n; ++i) {
103            int pos1 = i + all * n;
104            int pos0 = con1[pos1];
105            if (col[pos0] == 2) return 0;
106            col[pos0] = 1;
107            if (!dye(pos1, 2)) return 0;
108        }
109    for (int i = 1; i <= p + 1; ++i) {
110        int pos = belong[i];

```

```

111        if (getKind(g[st[i - 1] - 'a'])) pos = con1[pos];
112        if (col[pos] == 2) return 0;
113        col[pos] = 1;
114        if (!dye(con1[pos], 2)) return 0;
115    }
116    return 1;
117 }
118 int DAGDye(int n) {
119     for (int i = 1; i <= n; ++i) {
120         int x = topOrder[i];
121         if (!col[x]) {
122             col[x] = 1;
123             if (!dye(con1[x], 2)) return 0;
124         }
125     }
126     return 1;
127 }
128
129 int finalCheck(int n, int p) {
130     for (int i = 1; i <= n; ++i) {
131         if (col[belong[i]] != 1 && col[belong[con0[i]]] != 1)
132             return 0;
133     }
134     return 1;
135 }
136
137 int solve(int n, int p) {
138     memset(col, 0, sizeof(col));
139     if (!originDye(p, n)) return 0;
140     if (!DAGDye(tot)) return 0;
141     return finalCheck(n, p);
142 }
143
144 void getNextAlp() {
145     int len = strlen(g);
146     firAlp[1] = firAlp[0] = 'z' + 1;
147     for (int i = 0; i < len; ++i) {
148         nextAlp[i][0] = nextAlp[i][1] = 'z' + 1;
149         int k = getKind(g[i]);
150         firAlp[k] = min(firAlp[k], i + 'a');
151         for (int j = i + 1; j < len; ++j) {
152             int k = getKind(g[j]);
153             nextAlp[i][k] = min(nextAlp[i][k], 'a' + j);
154         }
155         if (nextAlp[i][0] > nextAlp[i][1])
156             swap(nextAlp[i][0], nextAlp[i][1]);

```

```

156     }
157 }
158 int main() {
159 #ifndef ONLINE_JUDGE
160     freopen("in.txt", "r", stdin);#endif
161     scanf("%s", g);
162     int n, m;
163     scanf("%d%d", &n, &m);
164     for (int i = 1; i <= n; ++i) {
165         con0[i] = i + n;
166         con0[i + n] = i;
167     }
168     num = 0;
169     for (int i = 1; i <= m; ++i) {
170         char t1, t2;
171         int pos1, pos2;
172         scanf("%d %c %d %c\n", &pos1, &t1, &pos2, &t2);
173         // if (i == 50 && n == 50 && m == 50) printf("%d %c %d %c\n", pos1, t1, pos2, t2);
174         int k1 = getKind(t1);
175         int k2 = getKind(t2);
176         pos1 += k1 * n;
177         pos2 += k2 * n;
178         link0(pos1, pos2);
179         link0(con0[pos2], con0[pos1]);
180     }
181     scanf("%s", st);
182     // if (n == 50 && m == 50) printf("%s\n", st);
183     getNextAlp();
184     if (getDAG(n) == 0) {
185         printf("-1\n");
186         return 0;
187     }
188     int flag = solve(n, n - 1);
189     for (int i = n - 1; i >= 0 && !flag; --i) {
190         int tmp = st[i] - 'a';
191         for (int j = 0; j <= 1 && !flag; ++j)
192             if (nextAlp[st[i] - 'a'][j] <= 'z') {
193                 st[i] = nextAlp[tmp][j];
194                 flag = solve(n, i);
195                 if (flag) {
196                     for (int k = i + 1; k <= n - 1; ++k) {
197                         int u = firAlp[0];
198                         int v = firAlp[1];
199                         if (u > v) swap(u, v);
200                         st[k] = u;

```

```

201         if (solve(n, k)) continue;
202         st[k] = v;
203     }
204 }
205 }
206 }
207 if (!flag) printf("-1\n");
208 else printf("%s\n", st);
209 return 0;
210 }

```

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。

```
1 #include <bits/stdc++.h>
2 #define REP(i, n) for(int i = 0; i < (int) (n); ++i)
3 #define REPP(i, a, b) for (int i = (int) (a); i <= (int) (b); ++i)
4 #define MST(a, b) memset((a), (b), sizeof(a))
5 #define MAXN 205
6 #define MAXM 21234
7
8 using namespace std;
9
10 typedef int arrayN[MAXN], arrayM[MAXM];
11 int N, E, C, num;
12 const int INF = ~0U >> 1;
13 arrayN fir, d;
14 arrayM nxt, e;
15 long long c[MAXM], c0[MAXM];
16
17 struct edge
18 {
19     int u, v, lab;
20     edge(int u = 0, int v = 0, int lab = 0): u(u), v(v), lab(lab) {};
21 } g[MAXM], cand[MAXM];
22
23 void link(int u, int v, int w)
24 {
25     e[++num] = v, nxt[num] = fir[u];
26     fir[u] = num, c[num] = 1LL * w;
27 }
28
29 void copy(long long cs[], long long cd[])
30 {
31     REPP(i, 1, num) cd[i] = cs[i];
32 }
33
34 bool bfs(int s)
35 {
36     MST(d, 0x3f);
37     d[s] = 0;
38     queue<int> que;
39     que.push(s);
40     for (; !que.empty();)
```

```
41     {
42         int u = que.front();
43         que.pop();
44         for (int p = fir[u]; p; p = nxt[p])
45             if (c[p] && d[e[p]] > d[u] + 1)
46             {
47                 d[e[p]] = d[u] + 1;
48                 que.push(e[p]);
49             }
50     }
51     return d[N] < d[0];
52 }
53
54 long long dfs(int x, long long low)
55 {
56     long long flow = 0;
57     if (x == N) return low;
58     for (int p = fir[x]; p; p = nxt[p])
59         if (c[p] && d[e[p]] == d[x] + 1)
60         {
61             long long tmp = dfs(e[p], min(low, c[p]));
62             if (!tmp) d[e[p]] = d[0];
63             c[p] -= tmp, c[p ^ 1] += tmp;
64             flow += tmp, low -= tmp;
65             if (!low) break;
66         }
67     return flow;
68 }
69
70 int com(edge A, edge B)
71 {
72     return A.u < B.u || (A.u == B.u && A.v < B.v);
73 }
74
75 void findCutEdge(long long base)
76 {
77     int tot = 0;
78     REPP(i, 1, N)
79         if (d[i] < d[0])
80             for (int p = fir[i]; p; p = nxt[p])
81                 if (d[e[p]] >= d[0] && !(p & 1))
82                     cand[++tot] = edge(i, e[p], p);
83     copy(c, c0);
84
85     int ansTot = 0;
86     REPP(i, 1, tot)
```

```

87     copy(c0, c);
88     c[cand[i].lab] = C;
89     long long ans = base;
90     for (; ans < C && bfs(1); ans += dfs(1, C));
91     if (ans >= C) g[++ansTot] = cand[i]; }
92 if (ansTot == 0)
93 {
94     printf("not possible\n");
95     return ;
96 }
97 sort(g + 1, g + ansTot + 1, com);
98 printf("possible option:(%d,%d)", g[1].u, g[1].v);
99 REPP(i, 2, ansTot)
100     printf(" (%d,%d)", g[i].u, g[i].v);
101 printf("\n");
102 }
103
104 int main()
105 {
106     freopen("uva11248.in", "r", stdin);
107     int task = 0;
108     for (;;)
109     {
110         scanf("%d%d%d", &N, &E, &C);
111         if (N + E + C == 0) break;
112         num = 1;
113         MST(fir, 0);
114         REP(i, E)
115         {
116             int u, v, w;
117             scanf("%d%d%d", &u, &v, &w);
118             link(u, v, w);
119             link(v, u, 0);
120         }
121         long long ans = 0;
122         for (; ans < C && bfs(1); ans += dfs(1, C));
123         ++task;
124         printf("Case %d: ", task);
125         if (ans >= C)
126         {
127             printf("possible\n");
128             continue;
129         }
130         findCutEdge(ans);
131     }
132     return 0;

```

133 }

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 j_1, j_2, \dots, j_k
 3. 判断 j_k 与 j_1, j_2, \dots, j_{k-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：
 - $N(v) = w \mid w \text{ 与 } v \text{ 相邻}$ ，且先加入
 - 团一定是 $v \text{ union } N(v)$ 的形式
 - 现在需要判断每个 $v \text{ union } N(v)$ 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 $|N(v)| + 1 \leq |N(w)|$ ，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

```
1 #include <bits/stdc++.h>
2 #define MAXN 1123
3 #define MAXM 2123456
4
5 using namespace std;
6 typedef int arrayN[MAXN], arrayM[MAXM];
7
8 arrayN fir, firMcs, nxtMcs, mcsSeq, l;
9 arrayN vis, r, cnt, preMcs, lab;
10 arrayM nxt, e;
11 int num, flag[MAXN][MAXN];
12 int mx; // max
13
14 void link(int u, int v)
15 {
16     e[++num] = v, nxt[num] = fir[u];
17     fir[u] = num;
18 }
19
20 void delMcs(int pos, int pt)
21 {
22     if (nxtMcs[pt] == pt)
23     {
24         r[l[pos]] = r[pos];
25         l[r[pos]] = l[pos];
26         if (pos == mx) mx = l[mx];
27         firMcs[pos] = 0;
28         return ;
29     }
30     preMcs[nxtMcs[pt]] = preMcs[pt];
31     nxtMcs[preMcs[pt]] = nxtMcs[pt];
32     if (firMcs[pos] == pt)
33         firMcs[pos] = nxtMcs[pt];
34 }
35
36 void insMcs(int pos, int pt)
37 {
38     if (firMcs[pos])
39     {
40         int tmp = firMcs[pos];
41         nxtMcs[pt] = tmp;
42         preMcs[pt] = preMcs[tmp];
43         nxtMcs[preMcs[pt]] = pt;
```

```
44         preMcs[nxtMcs[pt]] = pt;
45         return;
46     }
47     preMcs[pt] = nxtMcs[pt] = firMcs[pos] = pt;
48     if (firMcs[pos - 1]) //easy wrong
49     {
50         l[pos] = pos - 1;
51         r[pos] = r[pos - 1];
52     } else
53     {
54         if (l[pos - 1] == pos - 1)
55             l[pos] = r[pos] = pos;
56         else
57         {
58             l[pos] = l[pos - 1];
59             r[pos] = r[pos - 1];
60         }
61     }
62     r[l[pos]] = l[r[pos]] = pos;
63     if (pos > mx) mx = pos;
64 }
65
66 void getMcsSeq(int n, int m)
67 {
68     mx = 0;
69     l[0] = 0, r[0] = 0;
70     memset(firMcs, 0, sizeof(firMcs));
71     memset(cnt, 0, sizeof(cnt));
72     for (int i = 1; i <= n; ++i)
73     {
74         nxtMcs[i] = i + 1;
75         preMcs[i] = i - 1;
76     }
77     nxtMcs[n] = 1, preMcs[1] = n;
78     firMcs[0] = 1;
79     memset(vis, 0, sizeof(vis));
80     for (int i = 1; i <= n; ++i)
81     {
82         int tmp = (mcsSeq[i] = firMcs[mx]);
83         delMcs(cnt[tmp], tmp);
84         vis[tmp] = 1;
85         for (int p = fir[tmp]; p; p = nxt[p])
86             if (!vis[e[p]])
87             {
88                 delMcs(cnt[e[p]], e[p]);
89                 ++cnt[e[p]];
```

```

90         insMcs(cnt[e[p]], e[p]);
91     }
92 }
93 }
94
95 int checkMcs(int n)
96 {
97     for (int i = 1; i <= n; ++i)
98         lab[mcsSeq[i]] = i;
99     memset(vis, 0, sizeof(vis));
100     int now = 0;
101     for (int i = 1; i <= n; ++i)
102     {
103         ++now;
104         int pt = mcsSeq[i], cnt = 0, bgst = 0;
105         for (int p = fir[pt]; p; p = nxt[p])
106             if (lab[e[p]] < i)
107             {
108                 vis[e[p]] = now;
109                 ++cnt;
110                 if (lab[e[p]] > bgst)
111                     bgst = e[p];
112             }
113         if (bgst == 0) continue;
114         for (int p = fir[bgst]; p; p = nxt[p])
115         {
116             if (lab[e[p]] < i && vis[e[p]] == now)
117                 --cnt;
118         }
119         if (cnt > 1) return 0;
120     }
121     return 1;
122 }
123
124 int main()
125 {
126     // freopen("in.txt", "r", stdin);
127     // freopen("out.txt", "w", stdout);
128     for (;;)
129     {
130         int n, m;
131         scanf("%d%d", &n, &m);
132         if (n + m == 0) break;
133         num = 0;
134         memset(fir, 0, sizeof(fir));
135         memset(flag, 0, sizeof(flag));

```

```

136         for (int i = 1; i <= m; ++i)
137         {
138             int u, v;
139             scanf("%d%d", &u, &v);
140             if (flag[u][v] || u == v) continue;
141             link(u, v);
142             link(v, u);
143             flag[u][v] = flag[v][u] = 1;
144         }
145         getMcsSeq(n, m);
146         if (checkMcs(n)) printf("Perfect\n\n");
147         else printf("Imperfect\n\n");
148     }
149     return 0;
150 }

```

Listing 3.3: zoj1015.cpp

3.6.3 nlogn 好写得多

这个是求色数

```

1 #include <bits/stdc++.h>
2 #define MAXN 11234
3 #define MAXM 2123456
4
5 using namespace std;
6
7 typedef int arrayN[MAXN], arrayM[MAXM];
8
9 arrayN fir, mcsOrder, label, col;
10 arrayM e, nxt;
11 int num, n, base, seg[MAXN * 4];
12 set <int> s;
13
14 void link(int u, int v) {
15     e[++num] = v, nxt[num] = fir[u];
16     fir[u] = num;
17 }
18
19 int maxLab(int u, int v) {
20     return label[u] > label[v] ? u : v;
21 }
22
23 void change(int x, int val) {
24     label[x] = val;
25     x += base;

```

```

26     for (x >= 1; x; x >= 1) {
27         seg[x] = maxLab(seg[x << 1], seg[x << 1 ^ 1]);
28     }
29 }
30 void getMCS() {
31     for (base = 1; base <= n + 1; base <= 1);
32     for (int i = 1; i <= n; ++i) seg[i + base] = i;
33     label[0] = -1;
34     for (int i = base - 1; i >= 1; --i)
35         seg[i] = maxLab(seg[i << 1], seg[i << 1 ^ 1]);
36     int tot = 0;
37     for (int i = 1; i <= n; ++i) {
38         int x = mcsOrder[++tot] = seg[1];
39         change(x, -1);
40         for (int p = fir[x]; p; p = nxt[p]) {
41             if (label[e[p]] >= 0) change(e[p], label[e[p]] + 1);
42         }
43     }
44 }
45 int main()
46 {
47     #ifndef ONLINE_JUDGE
48         freopen("in.txt", "r", stdin);
49     #endif
50     int m;
51     scanf("%d", &n, &m);
52     for (int i = 1; i <= m; ++i) {
53         int u, v;
54         scanf("%d", &u, &v);
55         link(u, v);
56         link(v, u);
57     }
58     getMCS();
59     int ans = 0;
60     for (int i = 1; i <= n; ++i)
61         s.insert(i);
62     for (int j = 1; j <= n; ++j) {
63         int i = mcsOrder[j];
64         for (int p = fir[i]; p; p = nxt[p]) {
65             set<int>::iterator it = s.find(col[e[p]]);
66             if (it != s.end())
67                 s.erase(it);
68         }
69         col[i] = *s.begin();
70         ans = max(ans, col[i]);
71         for (int p = fir[i]; p; p = nxt[p]) {

```

```

72             set<int>::iterator it = s.find(col[e[p]]);
73             if (col[e[p]] && it == s.end())
74                 s.insert(col[e[p]]);
75         }
76     }
77     printf("%d\n", ans);
78     return 0;
79 }

```

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。

```

1 #include <bits/stdc++.h>
2 #define REP(i, n) for (int i = 0; i < (int) (n); ++i)
3 #define REPP(i, a, b) for(int i = (int) (a); i <= (int) (b); ++i)
4 #define MST(a, b) memset((a), (b), sizeof(a))
5 #define MAXN 66
6 #define MAXM 11234
7
8 using namespace std;
9 const int oo = ~0U>>1;
10 typedef int arrayN[MAXN], arrayM[MAXM];
11
12 int N, M, C;
13 arrayN vis, minW, belong, pre;
14
15 struct edge
16 {
17     int u, v, b, c;
18     edge(int u1 = 0, int v1 = 0, int b1 = 0, int c1 = 0)
19     {
20         u = u1, v = v1, b = b1, c = c1;
21     }
22 }edOri[MAXM], ed[MAXM];

```

```

23
24 int zhuLiu(int bLowLim)
25 {
26     int root = 0, tot = N, ntot;
27     int ans = 0;
28     REP(i, M) ed[i] = edOri[i]; for (;;)
29     {
30         REP(i, tot) minW[i] = oo, vis[i] = -1, belong[i] = -1;
31         REP(i, M)
32         {
33             if (ed[i].u == ed[i].v || ed[i].b < bLowLim) continue;
34             if (ed[i].c < minW[ed[i].v])
35             {
36                 minW[ed[i].v] = ed[i].c;
37                 pre[ed[i].v] = ed[i].u;
38             }
39         }
40
41         pre[root] = -1;
42         minW[root] = 0;
43         REP(i, tot)
44             if (minW[i] >= oo) return oo;
45             else ans += minW[i];
46         ntot = 0;
47         REP(i, tot)
48             if (vis[i] == -1)
49             {
50
51                 int h1 = i;
52                 for (; vis[h1] == -1; h1 = pre[h1])
53                 {
54                     vis[h1] = i;
55                     if (h1 == root) break;
56                 }
57                 if (h1 == root || vis[h1] != i) continue;
58                 int h2 = h1;
59                 for (h2 = pre[h1]; h2 != h1; h2 = pre[h2])
60                     belong[h2] = ntot;
61                 belong[h1] = ntot++;
62             }
63         REP(i, tot) if (belong[i] == -1) belong[i] = ntot++;
64         REP(i, M)
65         {
66             ed[i].c -= minW[ed[i].v];
67             ed[i].u = belong[ed[i].u];
68             ed[i].v = belong[ed[i].v];

```

```

69     }
70     if (tot == ntot) return ans;
71     tot = ntot;
72     root = belong[root];
73 }
74 }
75
76 int main()
77 {
78     freopen("in.txt", "r", stdin);
79     int task;
80     for (scanf("%d", &task); task; --task)
81     {
82         int L = 1, R = 1;
83         scanf("%d%d%d", &N, &M, &C);
84         REP(i, M)
85         {
86             int u, v, b, c;
87             scanf("%d%d%d%d", &u, &v, &b, &c);
88             edOri[i] = edge(u, v, b, c);
89             R = max(R, b);
90         }
91         L = 0;
92         for (; L < R; )
93         {
94             int mid = (L + R + 1) >> 1;
95             if (zhuLiu(mid) > C)
96                 R = mid - 1;
97             else L = mid;
98         }
99         if (L == 0) printf("streaming not possible.\n");
100         else printf("%d kbps\n", L);
101     }
102     return 0;
103 }

```

Listing 3.5: uva11865.cpp

3.8 二分图

3.8.1 普通 KM

```

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

```

```

4 #define MST(a, b) memset((a), (b), sizeof(a))
5 #define MAXN 512
6 #define INF 0x3f3f3f3f
7
8 using namespace std;
9
10 typedef int arrayN[MAXN];
11
12 int n;
13 arrayN S, T, match, w[MAXN], lx, ly;
14
15 int dfs(int x)
16 {
17     S[x] = 1;
18     REPP(i, 1, n)
19         if (lx[x] + ly[i] == w[x][i] && !T[i])
20         {
21             T[i] = 1; //容易忽略
22             if (!match[i] || dfs(match[i])) //dfs中别漏了match
23             {
24                 match[i] = x;
25                 return 1;
26             }
27         }
28     return 0;
29 }
30
31 void update()
32 {
33     int minL = INF; //找最小
34     REPP(i, 1, n)
35         if (S[i])
36             REPP(j, 1, n)
37                 if (!T[j])
38                     minL = min(minL, lx[i] + ly[j] - w[i][j]);
39     REPP(i, 1, n)
40     {
41         if (S[i]) lx[i] -= minL;
42         if (T[i]) ly[i] += minL;
43     }
44 }
45 void KM()
46 {
47     REPP(i, 1, n)
48     {
49         lx[i] = 0;

```

```

50         ly[i] = 0;
51         match[i] = 0;
52         REPP(j, 1, n)
53             lx[i] = max(lx[i], w[i][j]);
54     }
55     REPP(i, 1, n)
56     {
57         for (;;)
58         {
59             MST(S, 0);
60             MST(T, 0);
61             if (dfs(i)) break;
62             else update();
63         }
64     }
65 }
66 int main()
67 {
68     freopen("in.txt", "r", stdin);
69     for (; scanf("%d", &n) != EOF; )
70     {
71         REPP(i, 1, n)
72             REPP(j, 1, n)
73                 scanf("%d", &w[i][j]);
74         KM();
75         REPP(i, 1, n)
76             printf("%d%c", lx[i], " \n"[i == n]);
77         REPP(i, 1, n)
78             printf("%d%c", ly[i], " \n"[i == n]);
79         int ans = 0;
80         REPP(i, 1, n)
81             ans += w[match[i]][i];
82         printf("%d\n", ans);
83     }
84     return 0;
85 }

```

Listing 3.6: uva11383.cpp

3.8.2 牛逼 KM

```

1 #include<vector>
2 #include<cstdio>
3 #include<cstring>
4 #include<iostream>
5 #include<algorithm>

```

```

6 #include <cmath>
7 #include <cstdlib>
8 using namespace std;
9
10 const int N = 110 + 1;
11 const double INF = 1e12, EPS = 1e-6;
12
13 int n, p[N][N], fa[N];
14 bool used[N];
15 double w[N][N], u[N][N], v[N][N], minv[N];
16 // smallest match
17 void km(int lev) {
18     int i = lev;
19     lev++;
20     for (int j = 0; j <= n; ++j) {
21         u[lev][j] = u[i][j];
22         v[lev][j] = v[i][j];
23         p[lev][j] = p[i][j];
24         minv[j] = INF;
25         used[j] = false;
26     }
27     p[lev][n] = i;
28     int j0 = n;
29     do {
30         used[j0] = true;
31         int i0 = p[lev][j0], j1;
32         double delta = INF;
33         for (int j = 0; j < n; ++j) {
34             if (!used[j]) {
35                 double cur = w[i0][j] - u[lev][i0] - v[lev][j];
36                 if (cmp(cur - minv[j]) < 0) {
37                     minv[j] = cur;
38                     fa[j] = j0;
39                 }
40             }
41             if (cmp(minv[j] - delta) < 0) {
42                 delta = minv[j];
43                 j1 = j;
44             }
45         }
46         for (int j = 0; j <= n; ++j) {
47             if (used[j]) {
48                 u[lev][p[lev][j]] += delta, v[lev][j] -= delta;
49             } else {
50                 minv[j] -= delta;
51             }

```

```

52         }
53         j0 = j1;
54     } while (p[lev][j0] != -1);
55     do {
56         int j1 = fa[j0];
57         p[lev][j0] = p[lev][j1];
58         j0 = j1;
59     } while (j0 != n);
60 }
61
62 int main()
63 {
64     for (int i = 0; i <= n; ++i) {
65         u[0][i] = v[0][i] = 0;
66         p[0][i] = -1, fa[i] = 0;
67     }
68     for (int i = 0; i < n; ++i) {
69         for (int j = 0; j < n; ++j)
70             w[i][j] = 1.0 * dist(a[i], b[j]);
71         w[i][n] = 0;
72     }
73     for (int i = 0; i < n; ++i) km(i);
74     double ans = 0;
75     for (int i = 0; i < n; ++i) {
76         ans += w[p[n][i]][i];
77         printf("%d\n", p[n][i] + 1);
78     }
79 }

```

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 V_x 和 V_y , 如果有一条有向边 $A \rightarrow B$, 那么就加边 $A_x - B_y$ 。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 $a \rightarrow b$ 。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

```
1 #include <bits/stdc++.h>
2 #define REP(i, n) for (int i = 0; i < (n); ++i)
3 #define REPP(i, a, b) for(int i = (a); i <= (b); ++i)
4 #define MAXN 1123
5 #define MST(a, b) memset((a), (b), sizeof(a))
6
7 using namespace std;
8
9 int n, m, tot, w[MAXN][MAXN], vis[MAXN], cok[MAXN], rok[MAXN],
    match[MAXN];
10
11 int dfs(int x)
12 {
13     REPP(i, 1, n)
14         if (w[x][i] && !vis[i])
15         {
16             vis[i] = 1; //容易忽略
17             if (!match[i] || dfs(match[i]))
18             {
19                 match[i] = x;
20                 return 1;
21             }
22         }
23     return 0;
24 }
25
26 void dfs2(int x)
27 {
28     rok[x] = 1;
29     REPP(i, 1, n)
30         if (w[x][i] && !cok[i])
31         {
32             cok[i] = 1;
33             dfs2(match[i]);
34         }
```

```
34     }
35 }
36
37 int main()
38 {
39     freopen("in.txt", "r", stdin);
40     for (;;)
41     {
42         scanf("%d%d%d", &n, &m, &tot);
43         if (n + m + tot == 0) break;
44         MST(w, 0);
45         REPP(i, 1, tot)
46         {
47             int u, v;
48             scanf("%d%d", &u, &v);
49             w[u][v] = 1;
50         }
51         MST(match, 0);
52         int ans = 0;
53         REPP(i, 1, n)
54         {
55             MST(vis, 0);
56             if (dfs(i)) ++ans;
57         }
58         printf("%d", ans);
59         MST(vis, 0);
60         MST(rok, 0);
61         MST(cok, 0);
62         REPP(i, 1, n)
63             vis[match[i]] = 1;
64         REPP(i, 1, n)
65             if (!vis[i])
66                 dfs2(i);
67         REPP(i, 1, n)
68             if (!rok[i])
69                 printf(" r%d", i);
70         REPP(i, 1, n)
71             if (cok[i])
72                 printf(" c%d", i);
73         printf("\n");
74     }
75     return 0;
76 }
```

Listing 3.8: uva11419.cpp

3.9 带花树

3.9.1 普通图最大匹配

```
1  /*
2   解决一般图的最大匹配问题 O(N^3)
3  */
4
5  #include <bits/stdc++.h>
6  #define MAXE 250*250*2
7  #define MAXN 250
8  #define SET(a,b) memset(a,b,sizeof(a))
9
10 using namespace std;
11 //g[i][j]存放关系图：i,j是否有边,match[i]存放i所匹配的点
12 bool g[MAXN][MAXN],inque[MAXN],inblossom[MAXN];
13 int match[MAXN],pre[MAXN],base[MAXN];
14
15 queue<int> Q;
16
17 //找公共祖先
18 int lca(int u,int v) {
19     bool inpath[MAXN]= {false};
20     while(1) {
21         u=base[u];
22         inpath[u]=true;
23         if(match[u]==-1)break;
24         u=pre[match[u]];
25     }
26     while(1) {
27         v=base[v];
28         if(inpath[v])return v;
29         v=pre[match[v]];
30     }
31 }
32
33 //压缩花
34 void reset(int u,int anc) {
35     while(u!=anc) {
36         int v=match[u];
37         inblossom[base[u]]=1;
38         inblossom[base[v]]=1;
39         v=pre[v];
40         if(base[v]!=anc)pre[v]=match[u];
41         u=v;
42     }
```

```
43 }
44
45 void contract(int u,int v,int n) {
46     int anc = lca(u,v);
47     //SET(inblossom,0);
48     memset(inblossom,0,sizeof(inblossom));
49     reset(u,anc);
50     reset(v,anc);
51     if(base[u]!=anc)pre[u]=v;
52     if(base[v]!=anc)pre[v]=u;
53     for(int i=1; i<=n; i++)
54         if(inblossom[base[i]]) {
55             base[i]=anc;
56             if(!inque[i]) {
57                 Q.push(i);
58                 inque[i]=1;
59             }
60         }
61 }
62
63 bool dfs(int S,int n) {
64     for(int i=0; i<=n; i++)
65         pre[i]=-1 , inque[i]=0 , base[i]=i;
66     while(Q.size())Q.pop();
67     Q.push(S);
68     inque[S]=1;
69     while(!Q.empty()) {
70         int u=Q.front();
71         Q.pop();
72         for(int v=1; v<=n; v++) {
73             if(g[u][v]&&base[v]!=base[u]&&match[u]!=v) {
74                 if(v==S|| (match[v]!=-1&&pre[match[v]]!=-1))
75                     contract(u,v,n);
76                 else if(pre[v]==-1) {
77                     pre[v]=u;
78                     if(match[v]!=-1)
79                         Q.push(match[v]),inque[match[v]]=1;
80                 }
81                 else {
82                     u=v;
83                     while(u!=-1) {
84                         v=pre[u];
85                         int w=match[v];
86                         match[u]=v;
87                         match[v]=u;
88                         u=w;
89                     }
```



```

89         return true;
90     }
91 }
92 }
93 }
94 return false;
95 }
96
97 int main() {
98
99 #ifndef ONLINE_JUDGE
100     freopen("sum.in", "r", stdin);
101     //freopen("sum.out", "w", stdout);
102 #endif
103
104     int n, a, b, ans, i;
105     while (scanf("%d", &n) != EOF) {
106         ans = 0; //最多有几对匹配
107         memset(match, -1, sizeof(match));
108         memset(g, 0, sizeof(g));
109         while (scanf("%d%d", &a, &b) != EOF && a != 0)
110             g[a][b] = g[b][a] = 1;
111         for (i = 1; i <= n; i++)
112             if (match[i] == -1 && dfs(i, n))
113                 ans++;
114         cout << ans * 2 << endl;
115         for (i = 1; i <= n; i++)
116             if (match[i] != -1) {
117                 printf("%d %d\n", i, match[i]);
118                 match[i] = match[match[i]] = -1;
119             }
120     }
121     return 0;
122 }

```

Listing 3.9: ural1099.cpp

3.9.2 普通图最优匹配

```

1 /*
2  input
3  第一行两个正整数，n,m。保证 n≥2。
4
5  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）。

```

```

6  output
7
8  第一行一个整数，表示总代码量最多是多少（单位是万万行）。
9
10 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。
11 */
12
13 #include <iostream>
14 #include <cstdio>
15 #include <algorithm>
16 #include <vector>
17 using namespace std;
18
19 typedef long long s64;
20
21 const int INF = 2147483647;
22
23 const int MaxN = 400;
24 const int MaxM = 79800;
25
26 template <class T>
27 inline void tension(T &a, const T &b)
28 {
29     if (b < a)
30         a = b;
31 }
32 template <class T>
33 inline void relax(T &a, const T &b)
34 {
35     if (b > a)
36         a = b;
37 }
38 template <class T>
39 inline int size(const T &a)
40 {
41     return (int)a.size();
42 }
43
44 inline int getint()
45 {
46     char c;
47     while (c = getchar(), '0' > c || c > '9');
48
49     int res = c - '0';

```

```

50 while (c = getchar(), '0' <= c && c <= '9')
51     res = res * 10 + c - '0';
52 return res;
53 }
54
55 const int MaxNX = MaxN + MaxN;
56
57 struct edge
58 {
59     int v, u, w;
60
61     edge(){}
62     edge(const int &_v, const int &_u, const int &_w)
63         : v(_v), u(_u), w(_w){}
64 };
65
66 int n, m;
67 edge mat[MaxNX + 1][MaxNX + 1];
68
69 int n_matches;
70 s64 tot_weight;
71 int mate[MaxNX + 1];
72 int lab[MaxNX + 1];
73
74 int q_n, q[MaxN];
75 int fa[MaxNX + 1], col[MaxNX + 1];
76 int slackv[MaxNX + 1];
77
78 int n_x;
79 int bel[MaxNX + 1], blofrom[MaxNX + 1][MaxN + 1];
80 vector<int> bloch[MaxNX + 1];
81
82 inline int e_delta(const edge &e) // does not work inside blossoms
83 {
84     return lab[e.v] + lab[e.u] - mat[e.v][e.u].w * 2;
85 }
86 inline void update_slackv(int v, int x)
87 {
88     if (!slackv[x] || e_delta(mat[v][x]) < e_delta(mat[slackv[x]][x]))
89         slackv[x] = v;
90 }
91 inline void calc_slackv(int x)
92 {
93     slackv[x] = 0;
94     for (int v = 1; v <= n; v++)
95         if (mat[v][x].w > 0 && bel[v] != x && col[bel[v]] == 0)
96             update_slackv(v, x);
97 }
98
99 inline void q_push(int x)
100 {
101     if (x <= n)
102         q[q_n++] = x;
103     else
104     {
105         for (int i = 0; i < size(bloch[x]); i++)
106             q_push(bloch[x][i]);
107     }
108 }
109 inline void set_mate(int xv, int xu)
110 {
111     mate[xv] = mat[xv][xu].u;
112     if (xv > n)
113     {
114         edge e = mat[xv][xu];
115         int xr = blofrom[xv][e.v];
116         int pr = find(bloch[xv].begin(), bloch[xv].end(), xr) - bloch[
xv].begin();
117         if (pr % 2 == 1)
118         {
119             reverse(bloch[xv].begin() + 1, bloch[xv].end());
120             pr = size(bloch[xv]) - pr;
121         }
122
123         for (int i = 0; i < pr; i++)
124             set_mate(bloch[xv][i], bloch[xv][i ^ 1]);
125         set_mate(xr, xu);
126
127         rotate(bloch[xv].begin(), bloch[xv].begin() + pr, bloch[xv].end
());
128     }
129 }
130 inline void set_bel(int x, int b)
131 {
132     bel[x] = b;
133     if (x > n)
134     {
135         for (int i = 0; i < size(bloch[x]); i++)
136             set_bel(bloch[x][i], b);
137     }
138 }

```

```

139
140 inline void augment(int xv, int xu)
141 {
142     while (true)
143     {
144         int xnu = bel[mate[xv]];
145         set_mate(xv, xu);
146         if (!xnu)
147             return;
148         set_mate(xnu, bel[fa[xnu]]);
149         xv = bel[fa[xnu]], xu = xnu;
150     }
151 }
152 inline int get_lca(int xv, int xu)
153 {
154     static bool book[MaxNX + 1];
155     for (int x = 1; x <= n_x; x++)
156         book[x] = false;
157     while (xv || xu)
158     {
159         if (xv)
160         {
161             if (book[xv])
162                 return xv;
163             book[xv] = true;
164             xv = bel[mate[xv]];
165             if (xv)
166                 xv = bel[fa[xv]];
167         }
168         swap(xv, xu);
169     }
170     return 0;
171 }
172 inline void add_blossom(int xv, int xa, int xu)
173 {
174     int b = n + 1;
175     while (b <= n_x && bel[b])
176         b++;
177     if (b > n_x)
178         n_x++;
179
180     lab[b] = 0;
181     col[b] = 0;
182
183     mate[b] = mate[xa];
184

```

```

185     bloch[b].clear();
186     bloch[b].push_back(xa);
187     for (int x = xv; x != xa; x = bel[fa[bel[mate[x]]]])
188         bloch[b].push_back(x), bloch[b].push_back(bel[mate[x]]), q_push
            (bel[mate[x]]);
189     reverse(bloch[b].begin() + 1, bloch[b].end());
190     for (int x = xu; x != xa; x = bel[fa[bel[mate[x]]]])
191         bloch[b].push_back(x), bloch[b].push_back(bel[mate[x]]), q_push
            (bel[mate[x]]);
192
193     set_bel(b, b);
194
195     for (int x = 1; x <= n_x; x++)
196     {
197         mat[b][x].w = mat[x][b].w = 0;
198         blofrom[b][x] = 0;
199     }
200     for (int i = 0; i < size(bloch[b]); i++)
201     {
202         int xs = bloch[b][i];
203         for (int x = 1; x <= n_x; x++)
204             if (mat[b][x].w == 0 || e_delta(mat[xs][x]) < e_delta(mat[b][
                x]))
205                 mat[b][x] = mat[xs][x], mat[x][b] = mat[x][xs];
206         for (int x = 1; x <= n_x; x++)
207             if (blofrom[xs][x])
208                 blofrom[b][x] = xs;
209     }
210     calc_slackv(b);
211 }
212 inline void expand_blossom1(int b) // lab[b] == 1
213 {
214     for (int i = 0; i < size(bloch[b]); i++)
215         set_bel(bloch[b][i], bloch[b][i]);
216
217     int xr = blofrom[b][mat[b][fa[b]].v];
218     int pr = find(bloch[b].begin(), bloch[b].end(), xr) - bloch[b].
        begin();
219     if (pr % 2 == 1)
220     {
221         reverse(bloch[b].begin() + 1, bloch[b].end());
222         pr = size(bloch[b]) - pr;
223     }
224
225     for (int i = 0; i < pr; i += 2)
226     {

```

```

227     int xs = bloch[b][i], xns = bloch[b][i + 1];
228     fa[xs] = mat[xns][xs].v;
229     col[xs] = 1, col[xns] = 0;
230     slackv[xs] = 0, calc_slackv(xns);
231     q_push(xns); }
232 col[xr] = 1;
233 fa[xr] = fa[b];
234 for (int i = pr + 1; i < size(bloch[b]); i++)
235 {
236     int xs = bloch[b][i];
237     col[xs] = -1;
238     calc_slackv(xs);
239 }
240 bel[b] = 0;
241 }
242 inline void expand_blossom_final(int b) // at the final stage
243 {
244     for (int i = 0; i < size(bloch[b]); i++)
245     {
246         if (bloch[b][i] > n && lab[bloch[b][i]] == 0)
247             expand_blossom_final(bloch[b][i]);
248         else
249             set_bel(bloch[b][i], bloch[b][i]);
250     }
251     bel[b] = -1;
252 }
253
254 inline bool on_found_edge(const edge &e)
255 {
256     int xv = bel[e.v], xu = bel[e.u];
257     if (col[xu] == -1)
258     {
259         int nv = bel[mate[xu]];
260         fa[xu] = e.v;
261         col[xu] = 1, col[nv] = 0;
262         slackv[xu] = slackv[nv] = 0;
263         q_push(nv);
264     }
265     else if (col[xu] == 0)
266     {
267         int xa = get_lca(xv, xu);
268         if (!xa)
269         {
270             augment(xv, xu), augment(xu, xv);
271             for (int b = n + 1; b <= n_x; b++)
272

```

```

273         if (bel[b] == b && lab[b] == 0)
274             expand_blossom_final(b);
275         return true;
276     }
277     else
278         add_blossom(xv, xa, xu);
279 }
280 return false;
281 }
282
283 bool match()
284 {
285     for (int x = 1; x <= n_x; x++)
286         col[x] = -1, slackv[x] = 0;
287
288     q_n = 0;
289     for (int x = 1; x <= n_x; x++)
290         if (bel[x] == x && !mate[x])
291             fa[x] = 0, col[x] = 0, slackv[x] = 0, q_push(x);
292     if (q_n == 0)
293         return false;
294
295     while (true)
296     {
297         for (int i = 0; i < q_n; i++)
298         {
299             int v = q[i];
300             for (int u = 1; u <= n; u++)
301                 if (mat[v][u].w > 0 && bel[v] != bel[u])
302                 {
303                     int d = e_delta(mat[v][u]);
304                     if (d == 0)
305                     {
306                         if (on_found_edge(mat[v][u]))
307                             return true;
308                     }
309                     else if (col[bel[u]] == -1 || col[bel[u]] == 0)
310                         update_slackv(v, bel[u]);
311                 }
312             }
313
314     int d = INF;
315     for (int v = 1; v <= n; v++)
316         if (col[bel[v]] == 0)
317             tension(d, lab[v]);
318     for (int b = n + 1; b <= n_x; b++)

```

```

319     if (bel[b] == b && col[b] == 1)
320         tension(d, lab[b] / 2);
321     for (int x = 1; x <= n_x; x++)
322         if (bel[x] == x && slackv[x])
323             {
324                 if (col[x] == -1)
325                     tension(d, e_delta(mat[slackv[x]][x]));
326                 else if (col[x] == 0)
327                     tension(d, e_delta(mat[slackv[x]][x]) / 2);
328             }
329     for (int v = 1; v <= n; v++)
330     {
331         if (col[bel[v]] == 0)
332             lab[v] -= d;
333         else if (col[bel[v]] == 1)
334             lab[v] += d;
335     }
336     for (int b = n + 1; b <= n_x; b++)
337         if (bel[b] == b)
338             {
339                 if (col[bel[b]] == 0)
340                     lab[b] += d * 2;
341                 else if (col[bel[b]] == 1)
342                     lab[b] -= d * 2;
343             }
344     q_n = 0;
345     for (int v = 1; v <= n; v++)
346         if (lab[v] == 0) // all unmatched vertices' labels are zero!
347             cheers!
348         return false;
349     for (int x = 1; x <= n_x; x++)
350         if (bel[x] == x && slackv[x] && bel[slackv[x]] != x &&
351             e_delta(mat[slackv[x]][x]) == 0)
352             {
353                 if (on_found_edge(mat[slackv[x]][x]))
354                     return true;
355             }
356     for (int b = n + 1; b <= n_x; b++)
357         if (bel[b] == b && col[b] == 1 && lab[b] == 0)
358             expand_blossom1(b);
359     return false;
360 }
361
362 void calc_max_weight_match()

```

```

363 {
364     for (int v = 1; v <= n; v++)
365         mate[v] = 0;
366
367     n_x = n;
368     n_matches = 0;
369     tot_weight = 0;
370
371     bel[0] = 0;
372     for (int v = 1; v <= n; v++)
373         bel[v] = v, bloch[v].clear();
374     for (int v = 1; v <= n; v++)
375         for (int u = 1; u <= n; u++)
376             blofrom[v][u] = v == u ? v : 0;
377
378     int w_max = 0;
379     for (int v = 1; v <= n; v++)
380         for (int u = 1; u <= n; u++)
381             relax(w_max, mat[v][u].w);
382     for (int v = 1; v <= n; v++)
383         lab[v] = w_max;
384
385     while (match())
386         n_matches++;
387
388     for (int v = 1; v <= n; v++)
389         if (mate[v] && mate[v] < v)
390             tot_weight += mat[v][mate[v]].w;
391 }
392
393 int main()
394 {
395     n = getint(), m = getint();
396
397     for (int v = 1; v <= n; v++)
398         for (int u = 1; u <= n; u++)
399             mat[v][u] = edge(v, u, 0);
400
401     for (int i = 0; i < m; i++)
402     {
403         int v = getint(), u = getint(), w = getint();
404         mat[v][u].w = mat[u][v].w = w;
405     }
406
407     calc_max_weight_match();
408

```

```
409 printf("%lld\n", tot_weight);  
410 for (int v = 1; v <= n; v++)  
411     printf("%d ", mate[v]);  
412 printf("\n");  
413 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

```
1 #include <bits/stdc++.h>
2 #define MAXN 112345
3 #define MAXNODE 5012345
4
5 using namespace std;
6 typedef int arrayN[MAXN * 2];
7
8 arrayN e, nxt, fir;
9 int num, tot;
10
11 struct segmentNode
12 {
13     segmentNode *l, *r;
14     int low, up, num;
15 }tree[MAXNODE];
16
17 struct node
18 {
19     int val, dep;
```

```
20     int f[25];
21     segmentNode *rt;
22 } a[MAXN];
23
24
25 void link(int u, int v)
26 {
27     e[++num] = v, nxt[num] = fir[u];
28     fir[u] = num;
29 }
30
31 segmentNode *build(int l, int r)
32 {
33     segmentNode *tp = &tree[tot++];
34     int mid = l + r >> 1;
35     tp->low = l, tp->up = r;
36     tp->num = 0;
37     tp->l = tp->r = NULL;
38     if (l == r) return tp;
39     tp->l = build(l, mid);
40     tp->r = build(mid + 1, r);
41     return tp;
42 }
43
44 segmentNode *change(segmentNode *u, int x)
45 {
46     segmentNode *tp = &tree[tot++];
47     tp->l = u->l, tp->r = u->r;
48     tp->num = u->num + 1;
49     tp->low = u->low, tp->up = u->up;
50     int mid = tp->up + tp->low >> 1;
51     if (tp->low == tp->up) return tp;
52     if (x <= mid) tp->l = change(u->l, x);
53     else tp->r = change(u->r, x);
54     return tp;
55 }
56
57 void dfs(int x, int fa, int depth)
58 {
59     a[x].dep = depth;
60     a[x].f[0] = fa;
61     a[x].rt = change(a[fa].rt, a[x].val);
62     for (int p = fir[x]; p; p = nxt[p])
63         if (e[p] != fa)
64             dfs(e[p], x, depth + 1);
65 }
```

```

66 void initLCA(int n)
67 {
68     for (int i = 1; i <= 20; ++i)    for (int j = 1; j <= n; ++j)
69         a[j].f[i] = a[a[j].f[i - 1]].f[i - 1];
70 }
71
72 int getLCA(int u, int v)
73 {
74     if (a[u].dep < a[v].dep) swap(u, v);
75     int dt = a[u].dep - a[v].dep;
76     for (int i = 20; i >= 0 && dt; i--)
77         if (a[u].f[i] && ((1<<i) <= dt))
78         {
79             u = a[u].f[i];
80             dt -= (1 << i);
81         }
82     if (u == v) return u;
83     for (int i = 20; i >= 0; --i)
84         if (a[u].f[i] != a[v].f[i])
85             u = a[u].f[i], v = a[v].f[i];
86     return a[u].f[0];
87 }
88
89 int ask(int u, int v, int lca, int k)
90 {
91     int fa = a[lca].f[0];
92     segmentNode *lk1l = a[u].rt, *lk1r = a[lca].rt;
93     segmentNode *lk2l = a[v].rt, *lk2r = a[fa].rt;
94     for (; ;)
95     {
96         if (lk1l->low == lk1l->up) return lk1l->low;
97         int tmp = lk1l->l->num - lk1r->l->num + lk2l->l->num - lk2r->l
98             ->num;
99         if (tmp >= k)
100         {
101             lk1l = lk1l->l, lk1r = lk1r->l;
102             lk2l = lk2l->l, lk2r = lk2r->l;
103         }else
104         {
105             k -= tmp;
106             lk1l = lk1l->r, lk1r = lk1r->r;
107             lk2l = lk2l->r, lk2r = lk2r->r;
108         }
109     }
110 }

```

```

111 vector<int> vec;
112
113 int main()
114 {
115     freopen("in.txt", "r", stdin);
116     int n, m;
117     scanf("%d%d", &n, &m);
118     for (int i = 1; i <= n; ++i)
119     {
120         scanf("%d", &a[i].val);
121         vec.push_back(a[i].val);
122     }
123     sort(vec.begin(), vec.end());
124     vec.resize(unique(vec.begin(), vec.end()) - vec.begin());
125     for (int i = 1; i <= n; ++i)
126         a[i].val = lower_bound(vec.begin(), vec.end(), a[i].val) - vec.
127             begin();
128     for (int i = 1; i < n; ++i)
129     {
130         int u, v;
131         scanf("%d%d", &u, &v);
132         link(u, v);
133         link(v, u);
134     }
135     a[0].rt = build(0, n);
136     dfs(1, 0, 1);
137     initLCA(n);
138     for (int i = 1; i <= m; ++i)
139     {
140         int u, v, k;
141         scanf("%d%d%d", &u, &v, &k);
142         int lca = getLCA(u, v);
143         printf("%d\n", vec[ask(u, v, lca, k)]);
144     }
145     return 0;
146 }

```

Listing 4.1: COT.cpp

4.5 点分治

```

1 #include <cstdlib>
2 #include <cstdio>
3 #include <iostream>

```



```

4 #include <vector>
5 #include <cstring>
6 #include <algorithm>
7 #define REP(i, n) for(int i = 0; i < (int) (n); ++i)
8 #define REPP(i, a, b) for(int i = (int) (a); i <= (int) (b); ++i)
9 #define MST(a, b) memset(a, (b), sizeof(a))
10 #define MAXN 11111
11 //小于等于k的点对
12 using namespace std;
13 typedef int arrayN[MAXN * 2];
14
15 arrayN fir, nxt, e, c, sizeN, vis;
16 int n, k, ans, num;
17 vector<int> stRoot, stEp;
18
19
20 void link(int u, int v, int w)
21 {
22     e[++num] = v, nxt[num] = fir[u], fir[u] = num;
23     c[num] = w;
24 }
25
26 int dfsSize(int x, int fa)
27 {
28     sizeN[x] = 1;
29     for (int p = fir[x]; p; p = nxt[p])
30         if (e[p] != fa && !vis[e[p]])
31             sizeN[x] += dfsSize(e[p], x);
32     return sizeN[x];
33 }
34
35 int getRoot(int x, int fa, int totN)
36 {
37     int maxSize = totN - sizeN[x];
38     for (int p = fir[x]; p; p = nxt[p])
39         if (e[p] != fa && !vis[e[p]])
40             {
41                 maxSize = max(maxSize, sizeN[e[p]]);
42                 int tmp = getRoot(e[p], x, totN);
43                 if (tmp) return tmp;
44             }
45     if (maxSize <= totN / 2) return x;
46     return 0;
47 }
48
49 void dfsSt(int x, int fa, int len)

```

```

50 {
51     stEp.push_back(len);
52     for (int p = fir[x]; p; p = nxt[p])
53         if (!vis[e[p]] && e[p] != fa)
54             dfsSt(e[p], x, len + c[p]);
55 }
56
57 int calc(vector<int> &st)
58 {
59     int tmp = 0;
60     sort(st.begin(), st.end());
61     int L = 0, R = st.size() - 1;
62     for (; L < R;)
63     {
64         if (st[L] + st[R] <= k) tmp += R - L, L++;
65         else --R;
66     }
67     return tmp;
68 }
69
70 void solve(int x)
71 {
72     int root = getRoot(x, x, dfsSize(x, x));
73     vis[root] = 1;
74     stRoot.clear();
75     stRoot.push_back(0);
76     for (int p = fir[root]; p; p = nxt[p])
77         if (!vis[e[p]])
78             {
79                 stEp.clear();
80                 dfsSt(e[p], root, c[p]);
81                 ans += calc(stEp);
82                 REP(i, stEp.size())
83                     stRoot.push_back(stEp[i]);
84             }
85     ans += calc(stRoot);
86     for (int p = fir[root]; p; p = nxt[p])
87         if (!vis[e[p]]) solve(e[p]);
88     vis[root] = 0;
89 }
90
91 int main()
92 {
93     freopen("in.txt", "r", stdin);
94     for (;;)
95     {

```

```

96     scanf("%d%d", &n, &k);
97     if (n + k == 0) break;
98     ans = 0;
99     num = 0;
100    MST(fir, 0);      REPP(i, 1, n - 1)
101    {
102        int u, v, w;
103        scanf("%d%d%d", &u, &v, &w);
104        link(u, v, w);
105        link(v, u, w);
106    }
107    MST(vis, 0);
108    ans = 0;
109    solve(1);
110    printf("%d\n", ans);
111 }
112 return 0;
113 }

```

Listing 4.2: poj1741.cpp

树上 A 权值不超过 \lim 的 B 权值和最大的路径

```

1 #include <cstdlib>
2 #include <cstdio>
3 #include <iostream>
4 #include <vector>
5 #include <cstring>
6 #include <algorithm>
7 #define REP(i, n) for(int i = 0; i < (int) (n); ++i)
8 #define REPP(i, a, b) for(int i = (int) (a); i <= (int) (b); ++i)
9 #define MST(a, b) memset(a, (b), sizeof(a))
10 #define MAXN 11111
11 //小于等于k的点
12 using namespace std;
13 typedef int arrayN[MAXN * 2];
14
15 arrayN fir, nxt, e, c, sizeN, vis;
16 int n, k, ans, num;
17 vector<int> stRoot, stEp;
18
19 void link(int u, int v, int w)
20 {
21     e[++num] = v, nxt[num] = fir[u], fir[u] = num;
22     c[num] = w;
23 }

```

```

24 }
25
26 int dfsSize(int x, int fa)
27 {
28     sizeN[x] = 1;
29     for (int p = fir[x]; p; p = nxt[p])
30         if (e[p] != fa && !vis[e[p]])
31             sizeN[x] += dfsSize(e[p], x);
32     return sizeN[x];
33 }
34
35 int getRoot(int x, int fa, int totN)
36 {
37     int maxSize = totN - sizeN[x];
38     for (int p = fir[x]; p; p = nxt[p])
39         if (e[p] != fa && !vis[e[p]])
40         {
41             maxSize = max(maxSize, sizeN[e[p]]);
42             int tmp = getRoot(e[p], x, totN);
43             if (tmp) return tmp;
44         }
45     if (maxSize <= totN / 2) return x;
46     return 0;
47 }
48
49 void dfsSt(int x, int fa, int len)
50 {
51     stEp.push_back(len);
52     for (int p = fir[x]; p; p = nxt[p])
53         if (!vis[e[p]] && e[p] != fa)
54             dfsSt(e[p], x, len + c[p]);
55 }
56
57 int calc(vector<int> &st)
58 {
59     int tmp = 0;
60     sort(st.begin(), st.end());
61     int L = 0, R = st.size() - 1;
62     for (; L < R;)
63     {
64         if (st[L] + st[R] <= k) tmp += R - L, L++;
65         else --R;
66     }
67     return tmp;
68 }
69

```

```

70 void solve(int x)
71 {
72     int root = getRoot(x, x, dfsSize(x, x));
73     vis[root] = 1;
74     stRoot.clear();    stRoot.push_back(0);
75     for (int p = fir[root]; p; p = nxt[p])
76         if (!vis[e[p]])
77             {
78                 stEp.clear();
79                 dfsSt(e[p], root, c[p]);
80                 ans -= calc(stEp);
81                 REP(i, stEp.size())
82                     stRoot.push_back(stEp[i]);
83             }
84     ans += calc(stRoot);
85     for (int p = fir[root]; p; p = nxt[p])
86         if (!vis[e[p]]) solve(e[p]);
87     vis[root] = 0;
88 }
89
90 int main()
91 {
92     freopen("in.txt", "r", stdin);
93     for (;;)
94     {
95         scanf("%d%d", &n, &k);
96         if (n + k == 0) break;
97         ans = 0;
98         num = 0;
99         MST(fir, 0);
100        REPP(i, 1, n - 1)
101        {
102            int u, v, w;
103            scanf("%d%d%d", &u, &v, &w);
104            link(u, v, w);
105            link(v, u, w);
106        }
107        MST(vis, 0);
108        ans = 0;
109        solve(1);
110        printf("%d\n", ans);
111    }
112    return 0;
113 }

```

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 `memset()`;
- 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

```
1 #include <iostream>
2 #include <cstring>
3 #include <cstdlib>
4 #include <cstdio>
5 #include <algorithm>
6 #define REP(i, n) for(int i = 0; i < (int) (n); ++i)
7 #define REPP(i, a, b) for(int i = (int) (a); i <= (int) (b); ++i)
8 #define REDD(i, a, b) for(int i = (int) (a); i >= (int) (b); --i)
9 #define MST(a, b) memset((a), (b), sizeof(a))
10 #define MAXN 111111
11 #include <vector>
12
```

```
13 using namespace std;
14 int zLim;
15
16 long long gTot[MAXN * 4];
17 int t, g[MAXN * 4], n, ti[MAXN * 4], now;
18 struct node
19 {
20     int x, y, z, f;
21     long long tot;
22 } a[MAXN];
23
24 int comx(node A, node B)
25 {
26     return (A.x < B.x) || ((A.x == B.x) && (A.y < B.y)) || ((A.x ==
27         B.x) && (A.y == B.y) && A.z < B.z);
28 }
29 int comy(node A, node B)
30 {
31     return A.y < B.y;
32 }
33
34 void change(int pos, int x, long long cnt)
35 {
36     pos += t;
37     if (ti[pos] != now) g[pos] = gTot[pos] = 0;
38     if (x < g[pos]) return;
39     if (x == g[pos]) gTot[pos] += cnt;
40     else gTot[pos] = cnt, g[pos] = x;
41     ti[pos] = now;
42
43     for(pos >>= 1; pos; pos >>= 1)
44     {
45         if (ti[pos << 1] != now) g[pos << 1] = gTot[pos << 1] = 0;
46         if (ti[pos << 1 ^ 1] != now) g[pos << 1 ^ 1] = gTot[pos << 1
47             ^ 1] = 0;
48         ti[pos] = now;
49         g[pos] = max(g[pos << 1], g[pos << 1 ^ 1]);
50         gTot[pos] = 0;
51         if (g[pos] == g[pos << 1]) gTot[pos] += gTot[pos << 1];
52         if (g[pos] == g[pos << 1 ^ 1]) gTot[pos] += gTot[pos << 1 ^ 1];
53     }
54
55 int ask(int l, int r, long long &cnt)
56 {
```

```

57     if (l > r) return 0;
58     int tmp = 0;
59     cnt = 0;
60     l += t - 1, r += t + 1;
61     for (; (l ^ r) != 1; l >>= 1, r >>= 1)
62     {
63         if (!(l & 1))
64         {
65             if (ti[l + 1] == now)
66             {
67                 if (tmp == g[l + 1]) cnt += gTot[l + 1];
68                 else if (tmp < g[l + 1])
69                 {
70                     tmp = g[l + 1];
71                     cnt = gTot[l + 1];
72                 }
73             }
74             if (r & 1)
75             {
76                 if (ti[r - 1] == now)
77                 {
78                     if (tmp == g[r - 1]) cnt += gTot[r - 1];
79                     else if (tmp < g[r - 1])
80                     {
81                         tmp = g[r - 1];
82                         cnt = gTot[r - 1];
83                     }
84                 }
85             }
86         }
87     }
88     return tmp;
89 }
90
91 void solve(int l, int r)
92 {
93     if (l == r) return ;
94     int mid = (l + r) >> 1;
95     solve(mid + 1, r);
96
97     sort(a + mid + 1, a + r + 1, comy);
98     sort(a + l, a + mid + 1, comy);
99
100     // MST(g, 0);
101     //MST(gTot, 0);
102     ++now;

```

```

103     int pos = r + 1;
104     REDD(i, mid, l)
105     {
106         for (; pos > mid + 1 && a[pos - 1].y >= a[i].y; --pos)
107         {
108             change(a[pos - 1].z, a[pos - 1].f, a[pos - 1].tot);
109         }
110
111         long long tmpTot;
112         int tmp = ask(a[i].z, zLim, tmpTot) + 1;
113         if (a[i].f == tmp) a[i].tot += tmpTot;
114         else if (a[i].f < tmp)
115         {
116             a[i].f = tmp;
117             a[i].tot = tmpTot;
118         }
119     }
120
121     sort(a + l, a + r + 1, comx);
122     solve(l, mid);
123 }
124
125 int INT()
126 {
127     int res;
128     char ch;
129     while (ch = getchar(), !isdigit(ch));
130     for (res = ch - '0'; ch = getchar(), isdigit(ch);)
131         res = res * 10 + ch - '0';
132     return res;
133 }
134
135 int main()
136 {
137     int task;
138     freopen("in.txt", "r", stdin);
139     now = 0;
140     for (task = INT(); task; --task)
141     {
142         n = INT();
143         vector<int> dataZ;
144         REPP(i, 1, n)
145         {
146             a[i].x = INT();
147             a[i].y = INT();
148             a[i].z = INT();

```

149	a[i].f = 1;	161	solve(1, n);
150	a[i].tot = 1;	162	int ans = 0;
151	dataZ.push_back(a[i].z);	163	long long cnt = 0;
152	}	164	REPP(i, 1, n)
153	sort(dataZ.begin(), dataZ.end());	165	{
	unique(dataZ.begin(), dataZ.end()) - dataZ.begin());	166	if (ans == a[i].f) cnt += a[i].tot;
154	REPP(i, 1, n)	167	else if (ans < a[i].f) cnt = a[i].tot, ans = a[i].f;
155	{	168	}
156	a[i].z = (lower_bound(dataZ.begin(), dataZ.end(), a[i].	169	printf("%d %lld\n", ans, cnt);
	z) - dataZ.begin()) + 1;	170	}
157	}	171	return 0;
158	zLim = dataZ.size();	172 }	
159	for (t = 1; t <= zLim + 1; t <= 1);		
160	sort(a + 1, a + n + 1, comx);		

Listing 5.1: hdu4742.cpp