

ACM/ICPC Template

QingyuZhang VincentLDL

October 20, 2015

Contents

1	java	
2	dp 优化	
2.1	决策单调性优化	4
2.2	单调队列优化以及写仙人掌图	5
2.3	斜率优化	7
2.3.1	斜率以及 x 维都单调	7
2.3.2	随便什么情况: cdq 分治优化	8
3	图论	10
3.1	tarjan	10
3.1.1	2-sat	10
3.1.2	割顶, 点双联通分量	12
3.1.3	桥, 边双联通分量	16
3.2	pufer 编码	17
3.3	最佳追捕算法	17
3.4	网络流	17
3.4.1	dinic	17
3.5	弦图	19
3.5.1	做法与常见问题	19
3.5.2	万不得已用线性作法	19
3.5.3	$n \log n$ 好写得多	21
3.6	最小树形图	22
3.7	二分图	23
3.7.1	普通 KM	23
2	3.7.2 牛逼 KM	24
	3.7.3 常见问题汇总	25
4	3.7.4 最小点覆盖输出方案	25
3.8	带花树	26
3.8.1	普通图最大匹配	26
3.8.2	普通图最优匹配	28
3.9	最大团	32
3.10	欧拉理论	32
	3.10.1 注意事项	32
	3.10.2 混合图欧拉回路构图	33
3.11	曼哈顿最小生成树	34
4	数据结构	36
4.1	kd-tree	36
4.2	lct TODO	37
4.3	可持久化线段树以及 LCA 不能再写错了 !!!	39
4.4	点分治	41
5	其他算法	45
5.1	pq 树	45
5.2	DLX	47
5.2.1	数独: 精确覆盖	47
5.2.2	可重复覆盖	48
5.3	k 短路	49
5.4	cdq 分治与读入优化	51

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;

```


164 }

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

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

- 每条边与非割顶的点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
- 求点双联通分支可以求割顶的时候顺便求出来

题目描述: n 个骑士, m 个敌对关系. 举办一次会议选奇数个人 (不包括 1 个) 坐在圆桌上. 相邻的人不仇恨就可以成功举办会议. 若某个骑士什么会议都不能参加, 则踢走他. 问最少踢走多少人.

做法: 找不能形成奇圈的点. 理由如下: 1. 如果一个双联通分量内的某些顶点在一个奇圈中 (即双联通分量含有奇圈), 那么这个双联通分量的其他顶点也在某个奇圈中; 2. 如果一个双联通分量含有奇圈, 则他必定不是一个二分图. 反过来也成立, 这是一个充要条件. 所以本题的做法, 就是对补图求点双联通分量. 然后对于求得的点双联通分量, 使用染色法判断是不是二分图, 不是二分图, 这个双联通分量的点是可以存在的.

```

1 #include <algorithm>
2 #include <cstdio>
3 #include <cstring>
4 using namespace std;
5 const int MAXN = 1010;
6 const int MAXM = 2000010;
7 struct Edge
8 {
9     int to, next;
10 } edge[MAXM];
11 int head[MAXN], tot;
12 int Low[MAXN], DFN[MAXN], Stack[MAXN], Belong[MAXN];

```

```

13
14 int Index,top;
15 int block;//点双连通分量的个数
16 bool Instack[MAXN];
17 bool can[MAXN];
18 bool ok[MAXN];//标记
19 int tmp[MAXN];//暂时存储双连通分量中的点
20 int cc;//tmp的计数
21 int color[MAXN];//染色
22 void addedge(int u,int v) {
23     edge[tot].to = v;edge[tot].next = head[u];head[u] = tot++;
24 }
25 bool dfs(int u,int col)//染色判断二分图
26 {
27     color[u] = col;
28     for(int i = head[u];i != -1;i = edge[i].next)
29     {
30         int v = edge[i].to;
31         if( !ok[v] )continue;
32         if(color[v] != -1)
33         {
34             if(color[v]==col)return false;
35             continue;
36         }
37         if(!dfs(v,!col))return false;
38     }
39     return true;
40 }
41 void Tarjan(int u,int pre) {
42     int v;
43     Low[u] = DFN[u] = ++Index;
44     Stack[top++] = u;
45     Instack[u] = true;
46     for(int i = head[u];i != -1;i = edge[i].next)
47     {
48         v = edge[i].to;
49         if(v == pre)continue;
50         if( !DFN[v] )
51         {
52             Tarjan(v,u);
53             if(Low[u] > Low[v])Low[u] = Low[v];
54             if( Low[v] >= DFN[u])
55             {
56                 block++;
57                 int vn;
58                 cc = 0;

```

```

59         memset(ok,false,sizeof(ok));
60         do
61         {
62             vn = Stack[--top];
63             Belong[vn] = block;
64             Instack[vn] = false;
65             ok[vn] = true;
66             tmp[cc++] = vn;
67         }
68         while( vn!=v );
69         ok[u] = 1;
70         memset(color,-1,sizeof(color));
71         if( !dfs(u,0) )
72         {
73             can[u] = true;
74             while(cc--)can[tmp[cc]]=true;
75         }
76     }
77 }
78 else if(Instack[v] && Low[u] > DFN[v])
79     Low[u] = DFN[v];
80 }
81 }
82 void solve(int n)
83 {
84     memset(DFN,0,sizeof(DFN));
85     memset(Instack,false,sizeof(Instack));
86     Index = block = top = 0;
87     memset(can,false,sizeof(can));
88     for(int i = 1;i <= n;i++)
89         if(!DFN[i])
90             Tarjan(i,-1);
91     int ans = n;
92     for(int i = 1;i <= n;i++)
93         if(can[i])
94             ans--;
95     printf("%d\n",ans);
96 }
97 void init()
98 {
99     tot = 0;
100     memset(head,-1,sizeof(head));
101 }
102 int g[MAXN][MAXN];
103 int main()
104 {

```

```

105 int n,m;
106 int u,v;
107 while(scanf("%d%d",&n,&m)==2)
108 {
109     if(n==0 && m==0)break;          init();
110     memset(g,0,sizeof(g));
111     while(m--)
112     {
113         scanf("%d%d",&u,&v);
114         g[u][v]=g[v][u]=1;
115     }
116     for(int i = 1;i <= n;i++)
117         for(int j = 1;j <= n;j++)
118             if(i != j && g[i][j]==0)
119                 addedge(i,j);
120     solve(n);
121 }
122 return 0;
123 }

```

Listing 3.2: poj2942.cpp

我自己写的割顶

```

1 typedef int arrayN[MAXN];
2
3 int n, m, top, tot, next[MAXN * MAXN], e[MAXN * MAXN], num;
4 VP slack, g[MAXN];
5 arrayN low, dfn, fir, ed[MAXN], color, isAns, belong;
6
7 void tarjan(int x, int fa)
8 {
9     low[x] = dfn[x] = ++top;
10    REPP(p, 1, n)
11        if (!ed[x][p] && x != p && p != fa && dfn[p] <= dfn[x])
12        {
13            slack.PB(MP(x, p));
14            if (!dfn[p])
15            {
16                tarjan(p, x);
17                low[x] = min(low[x], low[p]);
18                if (low[p] >= dfn[x])
19                {
20                    g[++tot].clear();
21                    for (;;)
22                    {
23                        int u = slack.back().FI;
24                        int v = slack.back().SE;

```

```

25                g[tot].PB(slack.back());
26                slack.pop_back();
27                if (u == x && v == p) break;
28            }
29        } else low[x] = min(low[x], dfn[p]);
30    }
31 }
32 }
33
34 bool dye(int x, int co)
35 {
36     color[x] = co;
37     for (int p = fir[x]; p; p = next[p])
38         if (!color[e[p]])
39             if (!dye(e[p], 3 - co)) return false; else ;
40         else if (color[x] + color[e[p]] != 3) return false;
41         return true;
42 }
43
44 void link(int u, int v)
45 {
46     e[++num] = v, next[num] = fir[u];
47     fir[u] = num;
48 }
49
50 int main()
51 {
52     int x, y, z, ca = 1;
53     for (; RII(n, m); )
54     {
55         mem(ed);
56         mem(dfn);
57         top = tot = 0;
58         if (n + m == 0) break;
59         REP(i, m)
60         {
61             int u, v;
62             RII(u, v);
63             ed[u][v] = ed[v][u] = 1;
64         }
65         REPP(i, 1, n)
66             if (!dfn[i])
67                 tarjan(i, 0);
68         mem(isAns);
69         REPP(i, 1, tot)
70         {

```



```

71     mem(color);
72     mem(belong);
73     mem(fir);
74     num = 0;
75     int sta = 0;          for (; !g[i].empty(); g[i].
pop_back())
76     {
77         belong[g[i].back().FI] = belong[g[i].back().SE] = i
;
78         link(sta = g[i].back().FI, g[i].back().SE);
79         link(sta = g[i].back().SE, g[i].back().FI);
80     }
81     if (!dye(sta, 1))
82         REPP(j, 1, n)
83         if (belong[j] == i)
84             isAns[j] = 1;
85     }
86     int ans = 0;
87     REPP(i, 1, n)
88         if (!isAns[i]) ans++;
89     PI(ans);
90 }
91 return 0;
92 }

```

Listing 3.3: poj2942my.cpp

割顶的 tarjan 特别注意树根那个点.

```

1 // * 求 无向图的割点和桥
2 // * 可以找出割点和桥,求删掉每个点后增加的连通块。
3 // * 需要注意重边的处理,可以先用矩阵存,再转邻接表,或者进行判重
4 const int MAXN = 10010;
5 const int MAXM = 100010;
6 struct Edge {
7     int to,next;
8     bool cut;//是否为桥的标记
9 }edge[MAXN];
10 int head[MAXN],tot;
11 int Low[MAXN],DFN[MAXN],Stack[MAXN];
12 int Index,top;
13 bool Instack[MAXN];
14 bool cut[MAXN];
15 int add_block[MAXN];//删除一个点后增加的连通块
16 int bridge;
17 void addedge(int u,int v) {

```

```

19     edge[tot].to = v;edge[tot].next = head[u];edge[tot].cut = false
;
20     head[u] = tot++;
21 }
22 void Tarjan(int u,int pre) {
23     int v;
24     Low[u] = DFN[u] = ++Index;
25     Stack[top++] = u;
26     Instack[u] = true;
27     int son = 0;
28     for(int i = head[u];i != -1;i = edge[i].next) {
29         v = edge[i].to;
30         if(v == pre)continue;
31         if( !DFN[v] ) {
32             son++;
33             Tarjan(v,u);
34             if(Low[u] > Low[v])Low[u] = Low[v];
35 //桥
36 //一条无向边(u,v)是桥,当且仅当(u,v)为树枝边,且满足DFS(u)<Low(v)。
37             if(Low[v] > DFN[u]) {
38                 bridge++;
39                 edge[i].cut = true;
40                 edge[i^1].cut = true;
41             }
42 //割点
43 //一个顶点u是割点,当且仅当满足(1)或(2) (1) u为树根,且u
有多于一个子树。
44 // (2) u不为树根,且满足存在(u,v)为树枝边(或称父子边,
45 //即u为v在搜索树中的父亲),使得DFS(u)<=Low(v)
46             if(u != pre && Low[v] >= DFN[u])//不是树根
47             {
48                 cut[u] = true;
49                 add_block[u]++;
50             }
51         }
52         else if( Low[u] > DFN[v])
53             Low[u] = DFN[v];
54     }
55 //树根,分支数大于1
56     if(u == pre && son > 1)cut[u] = true;
57     if(u == pre)add_block[u] = son - 1;
58     Instack[u] = false;
59     top--;
60 }

```

Listing 3.4: tarjan.cpp

3.1.3 桥，边双联通分量

- 去掉桥之后求联通块即得边边双联通分量
- 桥不属于任何边双联通分支，其余边和每个定点都属于恰好一个边双联通分支。
- 构造边双联通图。先求所有桥，删除桥边，剩余的连通块即为双联通子图，缩点，连回桥，变成树。添加 $(leaf+1)/2$ 条边即可。每次选最近公共祖先最远的叶子连边。注意判度数等于 1 的点的个数才是 leaf，有时候根有特殊情况。
- 注意处理好重边

```
1 // 图上动态添边求桥数目
2 // 注意图不连通的情况，此题不考虑
3 #include <cstdio>
4 #include <cstring>
5 #include <cstdlib>
6 #include <algorithm>
7 #include <string>
8 #define MAXN 412345
9
10 using namespace std;
11 typedef int arrayN[MAXN];
12 arrayN fir0, e0, nxt0, fir1, e1, nxt1, f, faT;
13 arrayN vis, isBri, belong, dfn, low, par, dep;
14 int now, tot, num, briCnt;
15
16 void link0(int u, int v) {
17     e0[++num] = v, nxt0[num] = fir0[u];
18     fir0[u] = num, vis[num] = isBri[num] = 0;
19 }
20 void link1(int u, int v) {
21     e1[++num] = v, nxt1[num] = fir1[u];
22     fir1[u] = num;
23 }
24 void tarjan(int u) {
25     dfn[u] = low[u] = ++now;
26     for (int p = fir0[u]; p; p = nxt0[p])
27         if (!vis[p]) {
28             vis[p] = vis[p ^ 1] = 1;
29             int v = e0[p];
30             if (!dfn[v]) {
31                 tarjan(v);
32                 par[v] = p;
33                 low[u] = min(low[u], low[v]);
34             } else low[u] = min(low[u], dfn[v]);
35         }
```

```
36 }
37
38 void dfs(int x, int bel) {
39     belong[x] = bel;
40     for (int p = fir0[x], q; p; p = nxt0[p])
41         if (!isBri[p] && !belong[q = e0[p]]) dfs(q, bel);
42 }
43 void compressAndBuildTree(int n) {
44     memset(belong, 0, sizeof(belong));
45     tot = 0;
46     for (int i = 1; i <= n; ++i)
47         if (!belong[i]) dfs(i, ++tot);
48     num = 1;
49     memset(fir1, 0, sizeof(fir1));
50     for (int i = 1; i <= n; ++i)
51         for (int p = fir0[i]; p; p = nxt0[p])
52             if (isBri[p]) link1(belong[i], belong[e0[p]]);
53 }
54 void bcc_edge(int n) {
55     memset(dfn, 0, sizeof(dfn));
56     now = 0;
57     for (int i = 1; i <= n; ++i)
58         if (!dfn[i]) tarjan(i);
59
60     //getbrige
61     briCnt = 0;
62     for (int i = 2; i <= n; ++i) {
63         int p = par[i] ^ 1;
64         int u = e0[p], v = i;
65         if (dfn[u] < low[v]) {
66             ++briCnt;
67             isBri[p] = isBri[p ^ 1] = 1;
68         }
69     }
70     compressAndBuildTree(n);
71 }
72
73 void dfsGetfaTree(int u, int fa, int depth) {
74     faT[u] = fa;
75     dep[u] = depth;
76     for (int p = fir1[u], q; p; p = nxt1[p])
77         if ((q = e1[p]) != fa)
78             dfsGetfaTree(q, u, depth + 1);
79 }
80
81 int getfa(int x) {
```

```

82     return x == f[x] ? x : (f[x] = getfa(f[x]));
83 }
84 int ask(int u, int v) {
85     int tu = belong[u], tv = belong[v];
86     tu = getfa(tu), tv = getfa(tv); for (; tu != tv;) {
87         if (dep[tu] < dep[tv]) swap(tu, tv);
88         briCnt--;
89         f[tu] = faT[tv];
90         tu = getfa(faT[tv]);
91     }
92     return briCnt;
93 }
94 int main() {
95     #ifndef ONLINE_JUDGE
96         freopen("in.txt", "r", stdin);
97     #endif
98     int n, m;
99     for (int ca = 1; scanf("%d%d", &n, &m) != EOF; ++ca) {
100         if (n + m == 0) break;
101         printf("Case %d:\n", ca);
102         num = 1;
103         memset(fir0, 0, sizeof(fir0));
104         for (int i = 1; i <= m; ++i) {
105             int u, v;
106             scanf("%d%d", &u, &v);
107             link0(u, v), link0(v, u);
108         }
109         bcc_edge(n);
110         dfsGetfaTree(1, 0, 1);
111         for (int i = 1; i <= tot; ++i)
112             f[i] = i;
113         int q;
114         scanf("%d", &q);
115         for (; q; --q) {
116             int u, v;
117             scanf("%d%d", &u, &v);
118             printf("%d\n", ask(u, v));
119         }
120         printf("\n");
121     }
122 }

```

Listing 3.5: poj3694.cpp

3.2 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。

3.3 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。

3.4 网络流

3.4.1 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  bool bfs(int s)
34  {
35      MST(d, 0x3f);
36      d[s] = 0;
37      queue<int> que;
38      que.push(s);
39      for (; !que.empty(); )
40      {
41          int u = que.front();
42          que.pop();
43          for (int p = fir[u]; p; p = nxt[p])
44              if (c[p] && d[e[p]] > d[u] + 1)
45              {
46                  d[e[p]] = d[u] + 1;
47                  que.push(e[p]);
48              }
49      }
50      return d[N] < d[0];
51  }
52
53  long long dfs(int x, long long low)
54  {
55      long long flow = 0;
56      if (x == N) return low;
57      for (int p = fir[x]; p; p = nxt[p])
58          if (c[p] && d[e[p]] == d[x] + 1)
59          {
60              long long tmp = dfs(e[p], min(low, c[p]));
61              if (!tmp) d[e[p]] = d[0];
62              c[p] -= tmp, c[p ^ 1] += tmp;
63              flow += tmp, low -= tmp;
64              if (!low) break;
65          }
66      return flow;
67  }
68
69  int com(edge A, edge B)
70  {
71      return A.u < B.u || (A.u == B.u && A.v < B.v);

```

```

72  }
73  void findCutEdge(long long base)
74  {
75      int tot = 0;
76      REPP(i, 1, N)
77          if (d[i] < d[0])
78              for (int p = fir[i]; p; p = nxt[p])
79                  if (d[e[p]] >= d[0] && !(p & 1))
80                      cand[++tot] = edge(i, e[p], p);
81      copy(c, c0);
82
83      int ansTot = 0;
84      REPP(i, 1, tot)
85      {
86          copy(c0, c);
87          c[cand[i].lab] = C;
88          long long ans = base;
89          for (; ans < C && bfs(1); ans += dfs(1, C));
90          if (ans >= C) g[++ansTot] = cand[i];
91      }
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));    ++task;
123 printf("Case %d: ", task);
124 if (ans >= C)
125 {
126     printf("possible\n");
127     continue;
128 }
129 findCutEdge(ans);
130 }
131 return 0;
132 }

```

Listing 3.6: uva11248.cpp

3.5 弦图

3.5.1 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：
 - $N(v) = w \mid w \text{ 与 } v \text{ 相邻}, \text{ 且先加入}$
 - 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

3.5.2 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));

```

```

79     for (int i = 1; i <= n; ++i)
80     {
81         int tmp = (mcsSeq[i] = firMcs[mx]);
82         delMcs(cnt[tmp], tmp);
83         vis[tmp] = 1;
84         for (int p = fir[tmp]; p; p = nxt[p])
85             if (!vis[e[p]])
86             {
87                 delMcs(cnt[e[p]], e[p]);
88                 ++cnt[e[p]];
89                 insMcs(cnt[e[p]], e[p]);
90             }
91     }
92 }
93
94 int checkMcs(int n)
95 {
96     for (int i = 1; i <= n; ++i)
97         lab[mcsSeq[i]] = i;
98     memset(vis, 0, sizeof(vis));
99     int now = 0;
100     for (int i = 1; i <= n; ++i)
101     {
102         ++now;
103         int pt = mcsSeq[i], cnt = 0, bgst = 0;
104         for (int p = fir[pt]; p; p = nxt[p])
105             if (lab[e[p]] < i)
106             {
107                 vis[e[p]] = now;
108                 ++cnt;
109                 if (lab[e[p]] > bgst)
110                     bgst = e[p];
111             }
112         if (bgst == 0) continue;
113         for (int p = fir[bgst]; p; p = nxt[p])
114         {
115             if (lab[e[p]] < i && vis[e[p]] == now)
116                 --cnt;
117         }
118         if (cnt > 1) return 0;
119     }
120     return 1;
121 }
122
123 int main()
124 {

```

```

125 // freopen("in.txt", "r", stdin);
126 //freopen("out.txt", "w", stdout);
127 for (;;)
128 {
129     int n, m;        scanf("%d%d", &n, &m);
130     if (n + m == 0) break;
131     num = 0;
132     memset(fir, 0, sizeof(fir));
133     memset(flag, 0, sizeof(flag));
134     for (int i = 1; i <= m; ++i)
135     {
136         int u, v;
137         scanf("%d%d", &u, &v);
138         if (flag[u][v] || u == v) continue;
139         link(u, v);
140         link(v, u);
141         flag[u][v] = flag[v][u] = 1;
142     }
143     getMcsSeq(n, m);
144     if (checkMcs(n)) printf("Perfect\n\n");
145     else printf("Imperfect\n\n");
146 }
147 return 0;
148 }

```

Listing 3.7: zoj1015.cpp

3.5.3 nlogn 好写得更多

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
31 void getMCS() {
32     for (base = 1; base <= n + 1; base <= 1);
33     for (int i = 1; i <= n; ++i) seg[i + base] = i;
34     label[0] = -1;
35     for (int i = base - 1; i >= 1; --i)
36         seg[i] = maxLab(seg[i << 1], seg[i << 1 ^ 1]);
37     int tot = 0;
38     for (int i = 1; i <= n; ++i) {
39         int x = mcsOrder[++tot] = seg[1];
40         change(x, -1);
41         for (int p = fir[x]; p; p = nxt[p]) {
42             if (label[e[p]] >= 0) change(e[p], label[e[p]] + 1);
43         }
44     }
45 }
46
47 int main()
48 {
49     #ifndef ONLINE_JUDGE
50         freopen("in.txt", "r", stdin);
51     #endif
52     int m;
53     scanf("%d%d", &n, &m);
54     for (int i = 1; i <= m; ++i) {
55         int u, v;
56         scanf("%d%d", &u, &v);
57         link(u, v);
58         link(v, u);
59     }
60     getMCS();
61     int ans = 0;
62     for (int i = 1; i <= n; ++i)
        s.insert(i);

```

```

63     for (int j = 1; j <= n; ++j) {
64         int i = mcsOrder[j];
65         for (int p = fir[i]; p; p = nxt[p]) {
66             set<int>::iterator it = s.find(col[e[p]]);
67             if (it != s.end())
68                 s.erase(it);
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     }
78     printf("%d\n", ans);
79     return 0;

```

Listing 3.8: hnoi2008.cpp

3.6 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];
29     for (;;)
30     {
31         REP(i, tot) minW[i] = oo, vis[i] = -1, belong[i] = -1;
32         REP(i, M)
33         {
34             if (ed[i].u == ed[i].v || ed[i].b < bLowLim) continue;
35             if (ed[i].c < minW[ed[i].v])
36             {
37                 minW[ed[i].v] = ed[i].c;
38                 pre[ed[i].v] = ed[i].u;
39             }
40         }
41
42         pre[root] = -1;
43         minW[root] = 0;
44         REP(i, tot)
45             if (minW[i] >= oo) return oo;
46         else ans += minW[i];
47         ntot = 0;
48         REP(i, tot)
49             if (vis[i] == -1)
50             {
51
52                 int h1 = i;
53                 for (; vis[h1] == -1; h1 = pre[h1])
54                 {
55                     vis[h1] = i;
56                     if (h1 == root) break;
57                 }
58                 if (h1 == root || vis[h1] != i) continue;

```



```

59     int h2 = h1;
60     for (h2 = pre[h1]; h2 != h1; h2 = pre[h2])
61         belong[h2] = ntot;
62     belong[h1] = ntot++;
63 }
64 REP(i, tot) if (belong[i] == -1) belong[i] = ntot++;
65 M)
66 {
67     ed[i].c -= minW[ed[i].v];
68     ed[i].u = belong[ed[i].u];
69     ed[i].v = belong[ed[i].v];
70 }
71 if (tot == ntot) return ans;
72 tot = ntot;
73 root = belong[root];
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 }

3.7 二分图

3.7.1 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 void KM()
45 {
46     REPP(i, 1, n)
47     {
48         lx[i] = 0;
49         ly[i] = 0;
50         match[i] = 0;
51         REPP(j, 1, n)
52             lx[i] = max(lx[i], w[i][j]);
53     }
54     REPP(i, 1, n)
55     {
56         for (;;)
57         {
58             MST(S, 0);
59             MST(T, 0);
60             if (dfs(i)) break;
61             else update();
62         }
63     }
64 }
65 int main()
66 {
67     freopen("in.txt", "r", stdin);
68     for (; scanf("%d", &n) != EOF; )
69     {
70         REPP(i, 1, n)
71             REPP(j, 1, n)
72                 scanf("%d", &w[i][j]);
73         KM();
74         REPP(i, 1, n)
75             printf("%d%c", lx[i], " \n"[i == n]);
76         REPP(i, 1, n)
77             printf("%d%c", ly[i], " \n"[i == n]);
78         int ans = 0;
79         REPP(i, 1, n)
80             ans += w[match[i]][i];
81         printf("%d\n", ans);
82     }
83     return 0;

```

```

84 }

```

Listing 3.10: uva11383.cpp

3.7.2 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) {

```

```

41     delta = minv[j];
42     j1 = j;
43 }
44 }
45 for (int j = 0; j <= n; ++j) {
46     if (used[j]) {
47         u[lev][p[lev][j]] += delta, v[lev][j] -= delta;
48     } else {
49         minv[j] -= delta;
50     }
51 }
52 j0 = j1;
53 while (p[lev][j0] != -1);
54 do {
55     int j1 = fa[j0];
56     p[lev][j0] = p[lev][j1];
57     j0 = j1;
58 } while (j0 != n);
59 }
60
61 int main()
62 {
63     for (int i = 0; i <= n; ++i) {
64         u[0][i] = v[0][i] = 0;
65         p[0][i] = -1, fa[i] = 0;
66     }
67     for (int i = 0; i < n; ++i) {
68         for (int j = 0; j < n; ++j)
69             w[i][j] = 1.0 * dist(a[i], b[j]);
70         w[i][n] = 0;
71     }
72     for (int i = 0; i < n; ++i) km(i);
73     double ans = 0;
74     for (int i = 0; i < n; ++i) {
75         ans += w[p[n][i]][i];
76         printf("%d\n", p[n][i] + 1);
77     }
78 }

```

Listing 3.11: poj3565Better.cpp

3.7.3 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。

3.7.4 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)    if (w[x][i] && !cok[i])
30     {
31         cok[i] = 1;
32         dfs2(match[i]);
33     }
34 }
35
36 int main()
37 {
38     freopen("in.txt", "r", stdin);
39     for (;;)
40     {
41         scanf("%d%d%d", &n, &m, &tot);
42         if (n + m + tot == 0) break;
43         MST(w, 0);
44         REPP(i, 1, tot)
45         {
46             int u, v;
47             scanf("%d%d", &u, &v);
48             w[u][v] = 1;
49         }
50         MST(match, 0);
51         int ans = 0;
52         REPP(i, 1, n)
53         {
54             MST(vis, 0);
55             if (dfs(i)) ++ans;
56         }
57         printf("%d", ans);
58         MST(vis, 0);
59         MST(rok, 0);
60         MST(cok, 0);
61         REPP(i, 1, n)
62         vis[match[i]] = 1;
63         REPP(i, 1, n)
64         if (!vis[i])
65             dfs2(i);
66         REPP(i, 1, n)
67         if (!rok[i])
68             printf(" r%d", i);
69         REPP(i, 1, n)
70         if (cok[i])
71             printf(" c%d", i);

```

```

72     printf("\n");
73 }
74 return 0;
75 }

```

Listing 3.12: uva11419.cpp

3.8 带花树

3.8.1 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];         if(base[v]!=anc)pre[v]=match[u];
40         u=v;
41     }
42 }
43
44 void contract(int u,int v,int n) {
45     int anc = lca(u,v);
46     //SET(inblossom,0);
47     memset(inblossom,0,sizeof(inblossom));
48     reset(u,anc);
49     reset(v,anc);
50     if(base[u]!=anc)pre[u]=v;
51     if(base[v]!=anc)pre[v]=u;
52     for(int i=1; i<=n; i++)
53         if(inblossom[base[i]]) {
54             base[i]=anc;
55             if(!inque[i]) {
56                 Q.push(i);
57                 inque[i]=1;
58             }
59         }
60 }
61
62 bool dfs(int S,int n) {
63     for(int i=0; i<=n; i++)
64         pre[i]=-1 , inque[i]=0 , base[i]=i;
65     while(Q.size())Q.pop();
66     Q.push(S);
67     inque[S]=1;
68     while(!Q.empty()) {
69         int u=Q.front();
70         Q.pop();
71         for(int v=1; v<=n; v++) {
72             if(g[u][v]&&base[v]!=base[u]&&match[u]!=v) {
73                 if(v==S || (match[v]!=-1&&pre[match[v]]!=-1))
74                     contract(u,v,n);
75                 else if(pre[v]==-1) {
76                     pre[v]=u;
77                     if(match[v]!=-1)
78                         Q.push(match[v]),inque[match[v]]=1;
79                     else {
80                         u=v;

```

```

81         while(u!=-1) {
82             v=pre[u];
83             int w=match[v];
84             match[u]=v;
85             match[v]=u;
86             u=w;
87         }
88         return true;
89     }
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 3.13: ural1099.cpp

3.8.2 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  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
5
6  第一行一个整数 , 表示总代码量最多是多少 ( 单位是万万行 ) 。
7
8  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。
9 */
10
11 #include <iostream>
12 #include <cstdio>
13 #include <algorithm>
14 #include <vector>
15 using namespace std;
16
17 typedef long long s64;
18
19 const int INF = 2147483647;
20
21 const int MaxN = 400;
22 const int MaxM = 79800;
23
24 template <class T>
25 inline void tension(T &a, const T &b)
26 {
27     if (b < a)
28         a = b;
29 }
30 template <class T>
31 inline void relax(T &a, const T &b)
32 {
33     if (b > a)
34         a = b;
35 }
36 template <class T>
37 inline int size(const T &a)
38 {
39     return (int)a.size();
40 }
```

```
41
42 inline int getint()
43 {
44     char c;
45     while (c = getchar(), '0' > c || c > '9');
46
47     int res = c - '0';
48     while (c = getchar(), '0' <= c && c <= '9')
49         res = res * 10 + c - '0';
50     return res;
51 }
52
53 const int MaxNX = MaxN + MaxN;
54
55 struct edge
56 {
57     int v, u, w;
58
59     edge(){}
60     edge(const int &_v, const int &_u, const int &_w)
61         : v(_v), u(_u), w(_w){}
62 };
63
64 int n, m;
65 edge mat[MaxNX + 1][MaxNX + 1];
66
67 int n_matches;
68 s64 tot_weight;
69 int mate[MaxNX + 1];
70 int lab[MaxNX + 1];
71
72 int q_n, q[MaxN];
73 int fa[MaxNX + 1], col[MaxNX + 1];
74 int slackv[MaxNX + 1];
75
76 int n_x;
77 int bel[MaxNX + 1], blofrom[MaxNX + 1][MaxN + 1];
78 vector<int> bloch[MaxNX + 1];
79
80 inline int e_delta(const edge &e) // does not work inside blossoms
81 {
82     return lab[e.v] + lab[e.u] - mat[e.v][e.u].w * 2;
83 }
84 inline void update_slackv(int v, int x)
85 {
```

```

86   if (!slackv[x] || e_delta(mat[v][x]) < e_delta(mat[slackv[x]][x]))
87       slackv[x] = v;
88 }
89 inline void calc_slackv(int x)
90 {   slackv[x] = 0;
91     for (int v = 1; v <= n; v++)
92         if (mat[v][x].w > 0 && bel[v] != x && col[bel[v]] == 0)
93             update_slackv(v, x);
94 }
95
96 inline void q_push(int x)
97 {
98     if (x <= n)
99         q[q_n++] = x;
100     else
101     {
102         for (int i = 0; i < size(bloch[x]); i++)
103             q_push(bloch[x][i]);
104     }
105 }
106 inline void set_mate(int xv, int xu)
107 {
108     mate[xv] = mat[xv][xu].u;
109     if (xv > n)
110     {
111         edge e = mat[xv][xu];
112         int xr = blofrom[xv][e.v];
113         int pr = find(bloch[xv].begin(), bloch[xv].end(), xr) - bloch[
114             xv].begin();
115         if (pr % 2 == 1)
116         {
117             reverse(bloch[xv].begin() + 1, bloch[xv].end());
118             pr = size(bloch[xv]) - pr;
119         }
120         for (int i = 0; i < pr; i++)
121             set_mate(bloch[xv][i], bloch[xv][i ^ 1]);
122         set_mate(xr, xu);
123
124         rotate(bloch[xv].begin(), bloch[xv].begin() + pr, bloch[xv].end
125             ());
126     }
127 }
128 inline void set_bel(int x, int b)
129 {
130     bel[x] = b;
131     if (x > n)
132     {
133         for (int i = 0; i < size(bloch[x]); i++)
134             set_bel(bloch[x][i], b);
135     }
136 }
137 inline void augment(int xv, int xu)
138 {
139     while (true)
140     {
141         int xnu = bel[mate[xv]];
142         set_mate(xv, xu);
143         if (!xnu)
144             return;
145         set_mate(xnu, bel[fa[xnu]]);
146         xv = bel[fa[xnu]], xu = xnu;
147     }
148 }
149 inline int get_lca(int xv, int xu)
150 {
151     static bool book[MaxNX + 1];
152     for (int x = 1; x <= n_x; x++)
153         book[x] = false;
154     while (xv || xu)
155     {
156         if (xv)
157         {
158             if (book[xv])
159                 return xv;
160             book[xv] = true;
161             xv = bel[mate[xv]];
162             if (xv)
163                 xv = bel[fa[xv]];
164         }
165         if (xu)
166             swap(xv, xu);
167     }
168     return 0;
169 }
170 inline void add_blossom(int xv, int xa, int xu)
171 {
172     int b = n + 1;
173     while (b <= n_x && bel[b])
174         b++;

```

```

175 if (b > n_x)
176     n_x++;
177
178 lab[b] = 0;
179 col[b] = 0;
180 mate[b] = mate[xa];
181
182 bloch[b].clear();
183 bloch[b].push_back(xa);
184 for (int x = xv; x != xa; x = bel[fa[bel[mate[x]]]])
185     bloch[b].push_back(x), bloch[b].push_back(bel[mate[x]]), q_push(
186         (bel[mate[x]]));
187 reverse(bloch[b].begin() + 1, bloch[b].end());
188 for (int x = xu; x != xa; x = bel[fa[bel[mate[x]]]])
189     bloch[b].push_back(x), bloch[b].push_back(bel[mate[x]]), q_push(
190         (bel[mate[x]]));
191
192 set_bel(b, b);
193
194 for (int x = 1; x <= n_x; x++)
195 {
196     mat[b][x].w = mat[x][b].w = 0;
197     blofrom[b][x] = 0;
198 }
199 for (int i = 0; i < size(bloch[b]); i++)
200 {
201     int xs = bloch[b][i];
202     for (int x = 1; x <= n_x; x++)
203         if (mat[b][x].w == 0 || e_delta(mat[xs][x]) < e_delta(mat[b][
204             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].
219         begin();
220     if (pr % 2 == 1)
221     {
222         reverse(bloch[b].begin() + 1, bloch[b].end());
223         pr = size(bloch[b]) - pr;
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     }
233     col[xr] = 1;
234     fa[xr] = fa[b];
235     for (int i = pr + 1; i < size(bloch[b]); i++)
236     {
237         int xs = bloch[b][i];
238         col[xs] = -1;
239         calc_slackv(xs);
240     }
241     bel[b] = 0;
242 }
243
244 inline void expand_blossom_final(int b) // at the final stage
245 {
246     for (int i = 0; i < size(bloch[b]); i++)
247     {
248         if (bloch[b][i] > n && lab[bloch[b][i]] == 0)
249             expand_blossom_final(bloch[b][i]);
250         else
251             set_bel(bloch[b][i], bloch[b][i]);
252     }
253     bel[b] = -1;
254 }
255
256 inline bool on_found_edge(const edge &e)
257 {
258     int xv = bel[e.v], xu = bel[e.u];
259     if (col[xu] == -1)
260     {
261         int nv = bel[mate[xu]];
262         fa[xu] = e.v;
263         col[xu] = 1, col[nv] = 0;
264         slackv[xu] = slackv[nv] = 0;
265         q_push(nv);
266     }

```



```

263 }
264 else if (col[xu] == 0)
265 {
266     int xa = get_lca(xv, xu);
267     if (!xa) {
268         augment(xv, xu), augment(xu, xv);
269         for (int b = n + 1; b <= n_x; b++)
270             if (bel[b] == b && lab[b] == 0)
271                 expand_blossom_final(b);
272         return true;
273     }
274     else
275         add_blossom(xv, xa, xu);
276 }
277 return false;
278 }
279
280 bool match()
281 {
282     for (int x = 1; x <= n_x; x++)
283         col[x] = -1, slackv[x] = 0;
284
285     q_n = 0;
286     for (int x = 1; x <= n_x; x++)
287         if (bel[x] == x && !mate[x])
288             fa[x] = 0, col[x] = 0, slackv[x] = 0, q_push(x);
289     if (q_n == 0)
290         return false;
291
292     while (true)
293     {
294         for (int i = 0; i < q_n; i++)
295         {
296             int v = q[i];
297             for (int u = 1; u <= n; u++)
298                 if (mat[v][u].w > 0 && bel[v] != bel[u])
299                 {
300                     int d = e_delta(mat[v][u]);
301                     if (d == 0)
302                     {
303                         if (on_found_edge(mat[v][u]))
304                             return true;
305                     }
306                     else if (col[bel[u]] == -1 || col[bel[u]] == 0)
307                         update_slackv(v, bel[u]);
308                 }

```

```

309     }
310
311     int d = INF;
312     for (int v = 1; v <= n; v++)
313         if (col[bel[v]] == 0)
314             tension(d, lab[v]);
315     for (int b = n + 1; b <= n_x; b++)
316         if (bel[b] == b && col[b] == 1)
317             tension(d, lab[b] / 2);
318     for (int x = 1; x <= n_x; x++)
319         if (bel[x] == x && slackv[x])
320         {
321             if (col[x] == -1)
322                 tension(d, e_delta(mat[slackv[x]][x]));
323             else if (col[x] == 0)
324                 tension(d, e_delta(mat[slackv[x]][x]) / 2);
325         }
326
327     for (int v = 1; v <= n; v++)
328     {
329         if (col[bel[v]] == 0)
330             lab[v] -= d;
331         else if (col[bel[v]] == 1)
332             lab[v] += d;
333     }
334     for (int b = n + 1; b <= n_x; b++)
335         if (bel[b] == b)
336         {
337             if (col[bel[b]] == 0)
338                 lab[b] += d * 2;
339             else if (col[bel[b]] == 1)
340                 lab[b] -= d * 2;
341         }
342
343     q_n = 0;
344     for (int v = 1; v <= n; v++)
345         if (lab[v] == 0) // all unmatched vertices' labels are zero!
346             cheers!
347             return false;
348     for (int x = 1; x <= n_x; x++)
349         if (bel[x] == x && slackv[x] && bel[slackv[x]] != x &&
350             e_delta(mat[slackv[x]][x]) == 0)
351         {
352             if (on_found_edge(mat[slackv[x]][x]))
353                 return true;
354         }

```

```

353     for (int b = n + 1; b <= n_x; b++)
354         if (bel[b] == b && col[b] == 1 && lab[b] == 0)
355             expand_blossom1(b);
356 }
357 return false;}
358
359 void calc_max_weight_match()
360 {
361     for (int v = 1; v <= n; v++)
362         mate[v] = 0;
363
364     n_x = n;
365     n_matches = 0;
366     tot_weight = 0;
367
368     bel[0] = 0;
369     for (int v = 1; v <= n; v++)
370         bel[v] = v, bloch[v].clear();
371     for (int v = 1; v <= n; v++)
372         for (int u = 1; u <= n; u++)
373             blofrom[v][u] = v == u ? v : 0;
374
375     int w_max = 0;
376     for (int v = 1; v <= n; v++)
377         for (int u = 1; u <= n; u++)
378             relax(w_max, mat[v][u].w);
379     for (int v = 1; v <= n; v++)
380         lab[v] = w_max;
381
382     while (match())
383         n_matches++;
384
385     for (int v = 1; v <= n; v++)
386         if (mate[v] && mate[v] < v)
387             tot_weight += mat[v][mate[v]].w;
388 }
389
390 int main()
391 {
392     n = getint(), m = getint();
393
394     for (int v = 1; v <= n; v++)
395         for (int u = 1; u <= n; u++)
396             mat[v][u] = edge(v, u, 0);
397
398     for (int i = 0; i < m; i++)

```

```

399 {
400     int v = getint(), u = getint(), w = getint();
401     mat[v][u].w = mat[u][v].w = w;
402 }
403
404 calc_max_weight_match();
405
406 printf("%lld\n", tot_weight);
407 for (int v = 1; v <= n; v++)
408     printf("%d ", mate[v]);
409 printf("\n");
410
411 return 0;
412 }

```

Listing 3.14: uoj81.cpp

3.9 最大团

- 度数和 $= 2m$
- 度数小于等于 $\sqrt{2m}$ 的为小点, 否则大点
- 最大团包含小点, 设为 v_0 , 则最大团 $\in N(v_0)$
- 最大团只包含大点, 大点个数 $\sqrt{2m}$ 级别
- 复杂度 $O(2^{\sqrt{2m}}nm)$

3.10 欧拉理论

3.10.1 注意事项

欧拉回路先判断连通, 注意多连通块的孤立点要去掉.

```

1 // 每个边有编号, 求字典序最小的欧拉回路, 每次走小边即可.
   起点时第一条边中小的节点
2 #include <cstring>
3 #include <algorithm>
4 #include <cstdio>
5 #include <vector>
6 #define MAXN 2000
7
8 using namespace std;
9 typedef int arrayN[MAXN];

```

```

10
11 vector <int> ans;
12 vector < pair <int, int> > vec[MAXN];
13 int n, m, vis[MAXN * MAXN], S;
14 arrayN pos, f, deg;
15
16 int getfa(int x) {
17     return f[x] == x ? x : (f[x] = getfa(f[x]));
18 }
19
20 void link(int u, int v, int id) {
21     vec[u].push_back(make_pair(id, v));
22     vec[v].push_back(make_pair(id, u));
23     int fu = getfa(u), fv = getfa(v);
24     if (fu != fv) f[fu] = fv;
25     deg[u]++, deg[v]++;
26     n = max(n, max(u, v));
27     vis[id] = 0;
28 }
29
30 void dfs(int x) {
31     for (; pos[x] < vec[x].size(); ) {
32         int p = pos[x]++;
33         int id = vec[x][p].first;
34         int v = vec[x][p].second;
35         if (!vis[id]) {
36             vis[id] = 1;
37             dfs(v);
38             ans.push_back(id);
39         }
40     }
41 }
42
43 void getEulerPath() {
44     for (int i = 1; i <= n; ++i) {
45         sort(vec[i].begin(), vec[i].end()); // only this problem
46         pos[i] = 0;
47     }
48     ans.clear(); // 记得清空
49     dfs(S);
50     for (int i = ans.size() - 1; i >= 0; --i)
51         printf("%d%c", ans[i], " \n"[i == 0]);
52 }
53 bool checkEuler() {
54     int block = 0;
55     for (int i = 1; i <= n; ++i) {

```

```

56         if (getfa(i) == i && deg[i]) ++block;
57         if (deg[i] & 1) return false;
58     }
59     if (block && deg[S] == 0) return false; // only for this
        problem
60     return block <= 1;
61 }
62
63 int main() {
64     #ifndef ONLINE_JUDGE
65         freopen("in.txt", "r", stdin);
66     #endif
67     for (int u, v, id; scanf("%d%d", &u, &v) != EOF;) {
68         if (u + v == 0) break;
69         if (u > v) swap(u, v);
70         S = u;
71         scanf("%d", &id);
72         for (int i = 1; i < MAXN; ++i) {
73             vec[i].clear();
74             deg[i] = 0;
75             f[i] = i;
76         }
77         link(u, v, id);
78         for (;;) {
79             scanf("%d%d", &u, &v);
80             if (u + v == 0) break;
81             scanf("%d", &id);
82             link(u, v, id);
83         }
84         if (!checkEuler()) printf("Round trip does not exist.\n");
85         else getEulerPath();
86     }
87 }

```

Listing 3.15: poj1041.cpp

3.10.2 混合图欧拉回路构图

把该图的无向边随便定向，计算每个点的入度和出度。如果有某个点出入度之差为奇数，那么肯定不存在欧拉回路。因为欧拉回路要求每点入度 = 出度，也就是总度数为偶数，存在奇数度点必不能有欧拉回路。好了，现在每个点入度和出度之差均为偶数。那么将这个偶数除以 2，得 x_i 。也就是说，对于每一个点，只要将 x_i 条边改变方向（入 > 出就是变入，出 > 入就是变出），就能保证出 = 入。如果每个点都是出 = 入，那么很明显，该图就存在欧拉回路。现在的问题就变成了：我该改变哪些边，可以让每个点出 = 入？构造网络流模型。首先，有向边是不能改

变方向的，要之无用，删。一开始不是把无向边定向了吗？定的是什么向，就把网络构建成为什么样，边长容量上限 1。另新建 s 和 t 。对于入 $>$ 出的点 u ，连接边 (u, t) 、容量为 x ，对于出 $>$ 入的点 v ，连接边 (s, v) ，容量为 x （注意对不同的点 x 不同）。之后，察看是否有满流的分配。有就是能有欧拉回路，没有就是没有。欧拉回路是哪个？查看流值分配，将所有流量非 0（上限是 1，流值不是 0 就是 1）的边反向，就能得到每点入度 = 出度的欧拉图。由于是满流，所以每个入 $>$ 出的点，都有 x 条边进来，将这些进来的边反向，OK，入 = 出了。对于出 $>$ 入的点亦然。那么，没和 s 、 t 连接的点怎么办？和 s 连接的条件是出 $>$ 入，和 t 连接的条件是入 $>$ 出，那么这个既没和 s 也没和 t 连接的点，自然早在开始就已经满足入 = 出了。那么在网络流过程中，这些点属于“中间点”。我们知道中间点流量不允许有累积的，这样，进去多少就出来多少，反向之后，自然仍保持平衡。

3.11 曼哈顿最小生成树

本题描述：二维平面那 n 个点，分成 k 个集合。使得集合内部至少有两个点曼哈顿距离小于等于 x ，求最小的 x 。（实际上就是求曼哈顿最小生成树的 k 大边）

```
1 #include <stdio>
2 #include <cstring>
3 #include <algorithm>
4 using namespace std;
5 const int maxn = 60000, maxzb = 1000;
6 struct node{int x, y, k[2];} a[maxn];
7 struct point{int a, b, c;} b[maxn * 16];
8 struct treepoint{int k[2];} d[maxzb * 10];
9 int i, n, tot, f[maxn], lim, h, bh[maxn], kkk;
10 int comx(int p, int q) {return a[p].x < a[q].x;}
11 int comy(int p, int q) {return a[p].y < a[q].y;}
12 int com1(const point &p, const point &q) {return p.c < q.c;}
13 int getfa(int x) {if (f[x] != x) f[x] = getfa(f[x]); return f[x];}
14 int dist(int p, int q) {return abs(a[p].x - a[q].x) + abs(a[p].y - a[q].y);}
15 int minbh(int p, int q, int k) {return a[p].k[k] < a[q].k[k] ? p : q;}
16 int maxbh(int p, int q, int k) {return a[p].k[k] > a[q].k[k] ? p : q;}
17 void addx(int x)
18 {
19     int mid = h + a[x].k[1];
20     d[mid].k[0] = minbh(d[mid].k[0], x, 0);
21     for (mid >= 1; mid; mid >= 1)
22         d[mid].k[0] = minbh(d[mid << 1].k[0], d[mid << 1 ^ 1].k[0], 0);
23     mid = h + a[x].k[0];
24     d[mid].k[1] = maxbh(d[mid].k[1], x, 1);
25     for (mid >= 1; mid; mid >= 1)
```

```
26     d[mid].k[1] = maxbh(d[mid << 1].k[1], d[mid << 1 ^ 1].k[1], 1);
27 }
28 void addy(int x)
29 {
30     int mid = h + a[x].k[1];
31     d[mid].k[0] = minbh(d[mid].k[0], x, 0);
32     for (mid >= 1; mid; mid >= 1)
33         d[mid].k[0] = minbh(d[mid << 1].k[0], d[mid << 1 ^ 1].k[0], 0);
34     mid = h + a[x].k[0];
35     d[mid].k[1] = minbh(d[mid].k[1], x, 1);
36     for (mid >= 1; mid; mid >= 1)
37         d[mid].k[1] = minbh(d[mid << 1].k[1], d[mid << 1 ^ 1].k[1], 1);
38 }
39 int ask(int l, int r, int k, int boss)
40 {
41     int mid = 0;
42     if (l > r) return 0;
43     for (l += h - 1, r += h + 1; (l ^ r) != 1; l >>= 1, r >>= 1)
44     {
45         if (!(l & 1)) mid = boss ? maxbh(mid, d[l + 1].k[k], k) : minbh(mid, d[l + 1].k[k], k);
46         if (r & 1) mid = boss ? maxbh(mid, d[r - 1].k[k], k) : minbh(mid, d[r - 1].k[k], k);
47     } return mid;
48 }
49 void link()
50 {
51     for (i = 1; i <= n; ++i) bh[i] = i;
52     a[0].k[0] = maxzb * 3, a[0].k[1] = -1;
53     sort(bh + 1, bh + n + 1, comx);
54     for (addx(bh[n]), i = n - 1; i; addx(bh[i]), --i)
55     {
56         b[++tot].a = bh[i], b[tot].b = ask(a[bh[i]].k[1], lim, 0, 0);
57         b[tot].c = dist(b[tot].a, b[tot].b);
58         if (b[tot].b == 0) --tot;
59         b[++tot].a = bh[i], b[tot].b = ask(1, a[bh[i]].k[0], 1, 1);
60         b[tot].c = dist(b[tot].a, b[tot].b);
61         if (b[tot].b == 0) --tot;
62     }
63     sort(bh + 1, bh + n + 1, comy);
64     memset(d, 0, sizeof(d));
65     a[0].k[1] = a[0].k[0];
66     for (addy(bh[n]), i = n - 1; i; addy(bh[i]), --i)
67     {
68         b[++tot].a = bh[i], b[tot].b = ask(1, a[bh[i]].k[1], 0, 0);
```

```

69     b[tot].c = dist(b[tot].a, b[tot].b);
70     if (b[tot].b == 0) —tot;
71     b[++tot].a = bh[i], b[tot].b = ask(1, a[bh[i]].k[0], 1, 0);
72     b[tot].c = dist(b[tot].a, b[tot].b);
73     if (b[tot].b == 0) —tot;    }
74 }
75 void kruskal()
76 {
77     int ans = n;
78     sort(b + 1, b + tot + 1, com1);
79     for (i = 1; i <= n; ++i) f[i] = i;
80     for (i = 1; i <= tot; ++i)
81     {
82         int f1 = getfa(b[i].a), f2 = getfa(b[i].b);
83         if (f1 != f2)
84         {
85             —ans;
86             if (ans == kkk) break;
87             f[f1] = f2;
88         }
89     }
90     printf("%d\n", b[i].c);
91 }
92 int main()
93 {
94     freopen("tree.in", "r", stdin);
95     freopen("tree.out", "w", stdout);
96     scanf("%d%d", &n, &kkk);
97     for (i = 1; i <= n; ++i)
98     {
99         scanf("%d%d", &a[i].x, &a[i].y);
100         a[i].k[0] = a[i].x + a[i].y + maxzb;
101         a[i].k[1] = a[i].y - a[i].x + maxzb;
102         lim = max(max(lim, a[i].k[0]), a[i].k[1]);
103     }
104     for (h = 1; h <= lim; h <= 1);
105     link();
106     kruskal();
107     return 0;
108 }

```

Listing 3.16: poj3241.cpp

Chapter 4

数据结构

4.1 kd-tree

求 k 维空间 m 近点对

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <iostream>
4 #include <algorithm>
5 #include <vector>
6 #include <queue>
7 #include <set>
8 #include <map>
9 #include <string>
10 #include <math.h>
11 #include <stdlib.h>
12 #include <time.h>
13 using namespace std;
14 const int MAXN = 50010;
15 const int DIM = 10;
16 inline double sqr(double x){return x*x;}
17 namespace KDTree{
18     int K;//维数
19     struct Point{
20         int x[DIM];
21         double distance(const Point &b)const{
22             double ret = 0;
23             for(int i = 0;i < K;i++)
24                 ret += sqr(x[i]-b.x[i]);
25             return ret;
26         }
27     }
```

```
27     void input(){
28         for(int i = 0;i < K;i++)scanf("%d",&x[i]);
29     }
30     void output(){
31         for(int i = 0;i < K;i++)
32             printf("%d%c",x[i],i < K-1?' ':'\n');
33     }
34 };
35 struct qnode{
36     Point p;
37     double dis;
38     qnode(){}
39     qnode(Point _p,double _dis){
40         p = _p; dis = _dis;
41     }
42     bool operator <(const qnode &b)const{
43         return dis < b.dis;
44     }
45 };
46 priority_queue<qnode>q;
47 struct cmpx{
48     int div;
49     cmpx(const int &_div){div = _div;}
50     bool operator()(const Point &a,const Point &b){
51         for(int i = 0;i < K;i++)
52             if(a.x[(div+i)%K] != b.x[(div+i)%K])
53                 return a.x[(div+i)%K] < b.x[(div+i)%K];
54         return true;
55     }
56 };
57 bool cmp(const Point &a,const Point &b,int div){
58     cmpx cp = cmpx(div);
59     return cp(a,b);
60 }
61 struct Node{
62     Point e;
63     Node *lc,*rc;
64     int div;
65 }pool[MAXN],*tail,*root;
66 void init(){
67     tail = pool;
68 }
69 Node* build(Point *a,int l,int r,int div){
70     if(l >= r)return NULL;
71     Node *p = tail++;
72     p->div = div;
```

```

73     int mid = (l+r)/2;
74     nth_element(a+l,a+mid,a+r,cmpx(div));
75     p->e = a[mid];
76     p->lc = build(a,l,mid,(div+1)%K);
77     p->rc = build(a,mid+1,r,(div+1)%K);    return p;
78 }
79 void search(Point p,Node *x,int div,int m){
80     if(!x)return;
81     if(cmp(p,x->e,div)){
82         search(p,x->lc,(div+1)%K,m);
83         if(q.size() < m){
84             q.push(qnode(x->e,p.distance(x->e)));
85             search(p,x->rc,(div+1)%K,m);
86         }
87     } else {
88         if(p.distance(x->e) < q.top().dis){
89             q.pop();
90             q.push(qnode(x->e,p.distance(x->e)));
91         }
92         if(sqr(x->e.x[div]-p.x[div]) < q.top().dis)
93             search(p,x->rc,(div+1)%K,m);
94     }
95 }
96 else {
97     search(p,x->rc,(div+1)%K,m);
98     if(q.size() < m){
99         q.push(qnode(x->e,p.distance(x->e)));
100        search(p,x->lc,(div+1)%K,m);
101    }
102    else {
103        if(p.distance(x->e) < q.top().dis){
104            q.pop();
105            q.push(qnode(x->e,p.distance(x->e)));
106        }
107        if(sqr(x->e.x[div]-p.x[div]) < q.top().dis)
108            search(p,x->lc,(div+1)%K,m);
109    }
110 }
111 }
112 void search(Point p,int m){
113     while(!q.empty())q.pop();
114     search(p,root,0,m);
115 }
116 };
117 KDTree::Point p[MAXN];
118 int main()

```

```

119 {
120     int n,k;
121     while(scanf("%d%d",&n,&k) == 2){
122         KDTree::K = k;
123         for(int i = 0;i < n;i++)p[i].input();
124         KDTree::init();
125         KDTree::root = KDTree::build(p,0,n,0);
126         int Q;
127         scanf("%d",&Q);
128         KDTree::Point o;
129         while(Q--){
130             o.input();
131             int m;
132             scanf("%d",&m);
133             KDTree::search(o,m);
134             printf("the closest %d points are:\n",m);
135             int cnt = 0;
136             while(!KDTree::q.empty()){
137                 p[cnt++] = KDTree::q.top().p;
138                 KDTree::q.pop();
139             }
140             for(int i = 0;i < m;i++)p[m-1-i].output();
141         }
142     }
143     return 0;
144 }

```

Listing 4.1: hdu4347.cpp

4.2 1ct TODO

```

1 #include <bits/stdc++.h>
2 // 依次加入数字作为点，向其约数连gcd大小的边，维护最大生成树
3 // 这里把边变成点，这样查询点更好做。
4 const int MAXNODE = 11234567;
5 const int INF = 11234567;
6 const int MAXN = 112345;
7
8 using namespace std;
9
10 long long ANS = 0, ans[MAXN];
11
12 struct node {
13     bool rev;
14     int minVal, val;

```

```

15  node *f, *son[2], *minPos;
16  } g[MAXNODE], *null, *tree[MAXN];
17
18  int cnt = 0;
19
20  node* newNode(int x) {
21      node *tmp = &g[cnt++];
22      tmp->f = tmp->son[0] = tmp->son[1] = null;
23      tmp->minVal = tmp->val = x;
24      tmp->minPos = tmp;
25      return tmp;
26  }
27
28  void origin() {
29      null = &g[cnt++];
30      null->f = null->son[0] = null->son[1] = null;
31      null->val = null->minVal = INF;
32      null->minPos = null;
33      null->rev = false;
34  }
35
36  void makeRev(node *x) {
37      if (x == null) return ;
38      swap(x->son[0], x->son[1]);
39      x->rev = !x->rev;
40  }
41
42  void update(node *x) {
43      if (x == null) return ;
44      x->minVal = x->val;
45      x->minPos = x;
46      for (int k = 0; k <= 1; ++k)
47          if (x->son[k]->minVal < x->minVal) {
48              x->minVal = x->son[k]->minVal;
49              x->minPos = x->son[k]->minPos;
50          }
51  }
52
53  void pushDown(node *x) {
54      if (x == null) return ;
55      if (x->rev) {
56          makeRev(x->son[0]);
57          makeRev(x->son[1]);
58          x->rev = false;
59      }
60  }

```

```

61
62  bool ifRoot(node *x) {
63      if (x->f == null) return true;
64      return (x->f->son[0] != x) && (x->f->son[1] != x);
65  }
66
67  void rotate(node *x, node *y, int p) {
68      node *z = y->f;
69      x->f = z;
70      if (!ifRoot(y)) z->son[z->son[1] == y] = x;
71      y->son[p] = x->son[1 - p];
72      x->son[1 - p] = y;
73      if (y->son[p] != null)
74          y->son[p]->f = y;
75      if (x->son[1 - p] != null)
76          x->son[1 - p]->f = x;
77      update(y);
78      update(x);
79      update(z);
80  }
81
82  void splay(node *x) {
83      if (x == null) return ;
84      pushDown(x);
85      for (; !ifRoot(x);) {
86          node *y = x->f, *z = y->f;
87          pushDown(z);
88          pushDown(y);
89          pushDown(x);
90          int p = (y->son[1] == x);
91          int q = (z->son[1] == y);
92          if (ifRoot(y)) rotate(x, y, p);
93          else if (p == q) rotate(y, z, p), rotate(x, y, p);
94          else rotate(x, y, p), rotate(x, z, q);
95      }
96  }
97
98  node *access(node *u) {
99      node *v = null;
100     for (; u != null; u = u->f) {
101         splay(u);
102         u->son[1] = v;
103         v = u;
104         update(u);
105     }
106     return v;

```



```

107 }
108
109 void evert(node *x) { //换根
110     node *tmp = access(x);
111     makeRev(tmp); splay(x);
112 }
113
114 node *getRoot(node *x) {
115     access(x);
116     splay(x);
117     node *tmp = x;
118     for (; tmp->son[0] != null; tmp = tmp->son[0])
119         pushDown(tmp);
120     return tmp;
121 }
122
123 void link(node *const x, node *const y) {
124     if (getRoot(x) == getRoot(y)) return ;
125     evert(x);
126     x->f = y;
127     access(x);
128 }
129
130 void cut(node *x, node *y) { //切断x到y的路径,从y相连的第一个点断开
131     if ((x == y) || (getRoot(x) != getRoot(y))) return ;
132     evert(x);
133     access(y);
134     splay(y);
135     y->son[0]->f = null;
136     y->son[0] = null;
137     update(y);
138 }
139
140 void addEdge(int u, int v, int val) {
141     node *x = tree[u];
142     node *y = tree[v];
143     node *rx = getRoot(x);
144     node *ry = getRoot(y);
145     if (rx == ry) {
146         evert(x);
147         node *tmp = access(y);
148
149         node *delNode = tmp->minPos;
150         if (delNode->val >= val) return ;
151         ANS -= 1LL * delNode->minVal;
152         cut(x, delNode);

```

```

153         cut(y, delNode);
154     }
155     node *mid = newNode(val);
156     ANS += 1LL * val;
157     link(x, mid);
158     link(mid, y);
159 }
160
161 int main() {
162     origin();
163     for (int i = 1; i <= 100000; ++i) {
164         tree[i] = newNode(INF);
165         if (i == 1) continue;
166         for (int j = 1; 1LL * j * j <= i; ++j)
167             if (i % j == 0) {
168                 addEdge(i, j, j);
169                 if (j * j < i) addEdge(i, i/j, i/j);
170             }
171         ans[i] = ANS;
172     }
173     int n;
174     for (; scanf("%d", &n) != EOF; ) {
175         cout << ans[n] << endl;
176     }
177     return 0;
178 }

```

Listing 4.2: hdu5398.cpp

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

```

```

60   a[x].rt = change(a[fa].rt, a[x].val);
61   for (int p = fir[x]; p; p = nxt[p])
62     if (e[p] != fa)
63       dfs(e[p], x, depth + 1);
64 }
65
66 void initLCA(int n)
67 {
68   for (int i = 1; i <= 20; ++i)
69     for (int j = 1; j <= n; ++j)
70       a[j].f[i] = a[a[j].f[i - 1]].f[i - 1];
71 }
72
73 int getLCA(int u, int v)
74 {
75   if (a[u].dep < a[v].dep) swap(u, v);
76   int dt = a[u].dep - a[v].dep;
77   for (int i = 20; i >= 0 && dt; i--)
78     if (a[u].f[i] && ((1 << i) <= dt))
79     {
80       u = a[u].f[i];
81       dt -= (1 << i);
82     }
83   if (u == v) return u;
84   for (int i = 20; i >= 0; --i)
85     if (a[u].f[i] != a[v].f[i])
86       u = a[u].f[i], v = a[v].f[i];
87   return a[u].f[0];
88 }
89
90 int ask(int u, int v, int lca, int k)
91 {
92   int fa = a[lca].f[0];
93   segmentNode *lk1l = a[u].rt, *lk1r = a[lca].rt;
94   segmentNode *lk2l = a[v].rt, *lk2r = a[fa].rt;
95   for (; )
96   {
97     if (lk1l->low == lk1l->up) return lk1l->low;
98     int tmp = lk1l->l->num - lk1r->l->num + lk2l->l->num - lk2r->l->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.
begin();
127     for (int i = 1; i < n; ++i)
128     {
129         int u, v;
130         scanf("%d%d", &u, &v);
131         link(u, v);
132         link(v, u);
133     }
134     a[0].rt = build(0, n);
135     dfs(1, 0, 1);
136     initLCA(n);
137     for (int i = 1; i <= m; ++i)
138     {
139         int u, v, k;
140         scanf("%d%d%d", &u, &v, &k);
141         int lca = getLCA(u, v);
142         printf("%d\n", vec[ask(u, v, lca, k)]);
143     }
144     return 0;
145 }

```

Listing 4.3: COT.cpp

4.4 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 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 int dfsSize(int x, int fa)
26 {
27     sizeN[x] = 1;
28     for (int p = fir[x]; p; p = nxt[p])
29         if (e[p] != fa && !vis[e[p]])
30             sizeN[x] += dfsSize(e[p], x);
31     return sizeN[x];
32 }
33
34 int getRoot(int x, int fa, int totN)
35 {
36     int maxSize = totN - sizeN[x];
37     for (int p = fir[x]; p; p = nxt[p])
38         if (e[p] != fa && !vis[e[p]])
39         {
40             maxSize = max(maxSize, sizeN[e[p]]);
41             int tmp = getRoot(e[p], x, totN);
42             if (tmp) return tmp;
43         }
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);
101         REPP(i, 1, n - 1)
102         {
103             int u, v, w;
104             scanf("%d%d%d", &u, &v, &w);
105             link(u, v, w);
106             link(v, u, w);
107         }
108         MST(vis, 0);
109         ans = 0;
110         solve(1);
111         printf("%d\n", ans);
112     }
113     return 0;
114 }

```

Listing 4.4: 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;    c[num] = w;
23 }
24
25 int dfsSize(int x, int fa)
26 {
27     sizeN[x] = 1;
28     for (int p = fir[x]; p; p = nxt[p])
29         if (e[p] != fa && !vis[e[p]])
30             sizeN[x] += dfsSize(e[p], x);
31     return sizeN[x];
32 }
33
34 int getRoot(int x, int fa, int totN)
35 {
36     int maxSize = totN - sizeN[x];
37     for (int p = fir[x]; p; p = nxt[p])
38         if (e[p] != fa && !vis[e[p]])
39             {
40                 maxSize = max(maxSize, sizeN[e[p]]);
41                 int tmp = getRoot(e[p], x, totN);
42                 if (tmp) return tmp;
43             }
44     if (maxSize <= totN / 2) return x;
45     return 0;
46 }
47
48 void dfsSt(int x, int fa, int len)
49 {
50     stEp.push_back(len);
51     for (int p = fir[x]; p; p = nxt[p])
52         if (!vis[e[p]] && e[p] != fa)
53             dfsSt(e[p], x, len + c[p]);
54 }
55
56 int calc(vector<int> &st)
57 {
58     int tmp = 0;
59     sort(st.begin(), st.end());
60     int L = 0, R = st.size() - 1;
61     for (; L < R; )
62     {
63         if (st[L] + st[R] <= k) tmp += R - L, L++;

```

```

64         else —R;
65     }
66     return tmp;
67 }
68
69 void solve(int x)
70 {
71     int root = getRoot(x, x, dfsSize(x, x));
72     vis[root] = 1;
73     stRoot.clear();
74     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 4.5: poj1741.cpp

Chapter 5

其他算法

5.1 pq 树

题目. 一列一列得移动. 使得每一行的 1 连续

```
1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4 using namespace std;
5
6 const int maxn=505;
7
8 int N, cnt[maxn], left[maxn], right[maxn], lcnt, rcnt, onel[maxn],
   oner[maxn];
9 bool start[maxn], done[maxn], empty[maxn];
10 char A[maxn][maxn], TMP[maxn][maxn];
11
12 inline void find_term()
13 {
14     for (int i=0;i<N;i++)
15     {
16         left[i]=N, right[i]=0;
17         for (int j=0;j<N;j++) if (A[i][j]=='1')
18         {
19             left[i]=min(left[i],j);
20             right[i]=max(right[i],j);
21         }
22     }
23 }
24
25 inline void copy_col(char A[][maxn], int c1, char B[][maxn], int c2
```

```

    )
26 {
27     for (int i=0;i<N;i++) B[i][c2]=A[i][c1];
28 }
29
30 int drop(int *begin, int *end)
31 {
32     memset(empty,0,sizeof empty);
33     int n(0);
34     for (int *i=begin;i<end;i++)
35     {
36         copy_col(A,*i,TMP,n++);
37         empty[*i]=true;
38     }
39     return n;
40 }
41
42 void pull_left(int L, int R, int *begin, int *end)
43 {
44     int n(drop(begin,end));
45     for (int i=R-1,j=R-1;i>=L+n;i--,j--)
46     {
47         while (empty[j]) j--;
48         if (i!=j) copy_col(A,j,A,i);
49     }
50     for (int i=0;i<n;i++)
51         copy_col(TMP,i,A,L+i);
52     start[L+n]=true;
53     find_term();
54 }
55
56 void pull_right(int L, int R, int *begin, int *end)
57 {
58     int n(drop(begin,end));
59     for (int i=L,j=L;i<R-n;i++,j++)
60     {
61         while (empty[j]) j++;
62         if (i!=j) copy_col(A,j,A,i);
63     }
64     for (int i=0;i<n;i++)
65         copy_col(TMP,i,A,R-n+i);
66     start[R-n]=true;
67     find_term();
68 }
69
70 inline void find_range(int row, int &begin, int &end, int &cnt, int
```

```

    *one)
71 {
72     end++;
73     while (!start[begin]) begin--;
74     while (!start[end]) end++; cnt=0;
75     for (int i=begin;i<end;i++) if (A[row][i]=='1')
76         one[cnt++]=i;
77 }
78
79 void Split(int L, int R)
80 {
81     int row(-1);
82     for (int i=0;i<N;i++)
83         if (!done[i] && left[i]>=L && right[i]<R && (!~row || cnt[row]<=
84             cnt[i]))
85             row=i;
86     if (!~row) return;
87     bool ok(false), jump(true);
88     lcnt=0;
89     for (int i=L;i<R;i++)
90         if (A[row][i]=='1') onel[lcnt++]=i;
91     if (lcnt==R-L)
92         jump=false;
93     else
94         pull_left(L,R,one1,one1+lcnt);
95     done[row]=true;
96     while (!ok)
97     {
98         ok=true;
99         for (int i=0;i<N;i++)
100             if (
101                 !done[i] && left[i]>=L && right[i]<R &&
102                 (!start[left[i]] || !start[right[i]+1] || right[i]-left[i]
103                 +1!=cnt[i])
104                 {
105                     int lb(left[i]), le(left[i]), rb(right[i]), re(right[i]);
106                     find_range(i,lb,le,lcnt,one1);
107                     find_range(i,rb,re,rcnt,oner);
108                     if (lb==rb) continue;
109                     if (re==R && lb==L && (le-lb==lcnt || lcnt+rcnt==cnt[i]) &&
110                         jump)
111                     {
112                         for (int k=rb;k>=lb;k--) start[k+rcnt]=start[k], start[k
113                             ]=false;
114                         start[lb]=true;
115
116                         pull_left(lb,re,oner,oner+rcnt);
117                         rb=re=right[i];
118                         find_range(i,rb,re,rcnt,oner);
119                         pull_left(rb,re,oner,oner+rcnt);
120                     }
121                 }
122             else
123             {
124                 pull_right(lb,le,one1,one1+lcnt);
125                 pull_left(rb,re,oner,oner+rcnt);
126             }
127         for (int k=0;k<N;k++)
128             if (done[k] && A[k][R-1]=='1') jump=false;
129         if (right[i]-left[i]+1!=cnt[i])
130             puts("NO"), exit(0);
131         done[i]=true, ok=false;
132         break;
133     }
134 }
135
136
137 for (int i=L,j=i+1;i<R;i=j,j++)
138 {
139     while (!start[j]) j++;
140     Split(i,j);
141 }
142
143 int main()
144 {
145     scanf("%d",&N);
146     for (int i=0;i<N;i++)
147     {
148         scanf("%s",A[i]);
149         for (int j=0;j<N;j++)
150             if (A[i][j]=='1') cnt[i]++;
151         if (!cnt[i]) done[i]=true;
152     }
153     find_term();
154     start[0]=start[N]=true;
155     Split(0,N);
156     puts("YES");
157     for (int i=0;i<N;i++)
158         puts(A[i]);
159     return 0;
160 }

```

Listing 5.1: cf243E.cpp

5.2 DLX

n 行选若干行使得每一列一个 1.

5.2.1 数独: 精确覆盖

- 9*9*9 行, 表示 r 行 c 列填数字 k
- 9*9*4 列. 分别表示 r 行有 k, c 列有 k, t 格子有 k, r 行 c 列有数字
- 即转化成选若干行使得每一列有一个 1 的问题.

```
1 #include <stdio>
2 #include <algorithm>
3 #include <cstring>
4 using namespace std;
5 const int N = 9; //3*3数独
6 const int MaxN = N*N*N + 10;
7 const int MaxM = N*N*4 + 10;
8 const int maxnode = MaxN*4 + MaxM + 10;
9 char g[MaxN];
10 struct DLX {
11     int n,m,size;
12     int
13     U[maxnode],D[maxnode],R[maxnode],L[maxnode],Row[maxnode],Col[
14     maxnode];
15     int H[MaxN],S[MaxM]; //S[i] 该列剩余1得数目, 每次取最小删除, H[
16     i]行得头节点
17     int ansd,ans[MaxN];
18     void init(int _n,int _m) {
19         n = _n;
20         m = _m;
21         for(int i = 0;i <= m;i++) {
22             S[i] = 0;
23             U[i] = D[i] = i;
24             L[i] = i - 1;
25             R[i] = i + 1;
26         }
27         R[m] = 0; L[0] = m;
28         size = m;
29         for(int i = 1;i <= n;i++) H[i] = -1;
30     }
31     void Link(int r,int c) {
32         ++S[Col[++size]=c];
33         Row[size] = r;
```

```
34         D[size] = D[c];
35         U[D[c]] = size;
36         U[size] = c;
37         D[c] = size;
38         if(H[r] < 0)H[r] = L[size] = R[size] = size;
39         else {
40             R[size] = R[H[r]];
41             L[R[H[r]]] = size;
42             L[size] = H[r];
43             R[H[r]] = size;
44         }
45     }
46     void remove(int c) {
47         L[R[c]] = L[c]; R[L[c]] = R[c];
48         for(int i = D[c];i != c;i = D[i])
49             for(int j = R[i];j != i;j = R[j]) {
50                 U[D[j]] = U[j];
51                 D[U[j]] = D[j];
52                 --S[Col[j]];
53             }
54     }
55     void resume(int c) {
56         for(int i = U[c];i != c;i = U[i])
57             for(int j = L[i];j != i;j = L[j])
58                 ++S[Col[U[D[j]]=D[U[j]]=j]];
59         L[R[c]] = R[L[c]] = c;
60     }
61     bool Dance(int d) {
62         if(R[0] == 0) {
63             for(int i = 0;i < d;i++)g[(ans[i]-1)/9] = (ans[i]-1)%9
64             + '1';
65             for(int i = 0;i < N*N;i++)printf("%c",g[i]);
66             printf("\n");
67             return true;
68         }
69         int c = R[0];
70         for(int i = R[0];i != 0;i = R[i])
71             if(S[i] < S[c])
72                 c = i;
73         remove(c);
74         for(int i = D[c];i != c;i = D[i])
75             {
76                 ans[d] = Row[i];
77                 for(int j = R[i];j != i;j = R[j])remove(Col[j]);
78                 if(Dance(d+1))return true;
79                 for(int j = L[i];j != i;j = L[j])resume(Col[j]);
80             }
```

```

77     }
78     resume(c);
79     return false;
80 }
81 };void place(int &r,int &c1,int &c2,int &c3,int &c4,int i,int j,int
      k)
82 {
83     r = (i*N+j)*N + k; c1 = i*N+j+1; c2 = N*N+i*N+k;
84     c3 = N*N*2+j*N+k; c4 = N*N*3+((i/3)*3+(j/3))*N+k;
85 }
86 DLX dlx;
87 int main() {
88     while(scanf("%s",g) == 1) {
89         if(strcmp(g,"end") == 0) break;
90         dlx.init(N*N*N,N*N*4);
91         int r,c1,c2,c3,c4;
92         for(int i = 0;i < N;i++)
93             for(int j = 0;j < N;j++)
94                 for(int k = 1;k <= N;k++)
95                     if(g[i*N+j] == '.' || g[i*N+j] == '0'+k) {
96                         place(r,c1,c2,c3,c4,i,j,k);
97                         dlx.Link(r,c1);
98                         dlx.Link(r,c2);
99                         dlx.Link(r,c3);
100                        dlx.Link(r,c4);
101                    }
102                dlx.Dance(0);
103            }
104            return 0;
105 }

```

Listing 5.2: poj3074.cpp

5.2.2 可重复覆盖

题目: $n*m$ 的 01 矩阵,1 表示有怪物. 每次可以消灭 $n1*m1$ 的矩阵内的东西. 最少多少次可以消灭所有怪物.

```

1 #include <stdio>
2 #include <algorithm>
3 #include <cstring>
4 using namespace std;
5
6 const int MaxM = 15*15+10;
7 const int MaxN = 15*15+10;
8 const int maxnode = MaxN * MaxM;

```

```

9 const int INF = 0x3f3f3f3f;
10 struct DLX
11 {
12     int n,m,size;
13     int
14     U[maxnode],D[maxnode],R[maxnode],L[maxnode],Row[maxnode],Col[
15     maxnode];
16     int H[MaxN],S[MaxM];
17     int ansd;
18     void init(int _n,int _m) {
19         n = _n;
20         m = _m;
21         for(int i = 0;i <= m;i++) {
22             S[i] = 0;
23             U[i] = D[i] = i;
24             L[i] = i-1;
25             R[i] = i+1;
26         }
27         R[m] = 0; L[0] = m;
28         size = m;
29         for(int i = 1;i <= n;i++)H[i] = -1;
30     }
31     void Link(int r,int c) {
32         ++S[Col[++size]=c];
33         Row[size] = r;
34         D[size] = D[c];
35         U[D[c]] = size;
36         U[size] = c;
37         D[c] = size;
38         if(H[r] < 0)H[r] = L[size] = R[size] = size;
39         else {
40             R[size] = R[H[r]];
41             L[R[H[r]]] = size;
42             L[size] = H[r];
43             R[H[r]] = size;
44         }
45     }
46     void remove(int c) {
47         for(int i = D[c];i != c;i = D[i])
48             L[R[i]] = L[i], R[L[i]] = R[i];
49     }
50     void resume(int c) {
51         for(int i = U[c];i != c;i = U[i])
52             L[R[i]] = R[L[i]] = i;
53     }
54     bool v[MaxM];

```

```

54 int f() {
55     int ret = 0;
56     for(int c = R[0]; c != 0; c = R[c])v[c] = true;
57     for(int c = R[0]; c != 0; c = R[c])
58         if(v[c]) { ret++;
59             v[c] = false;
60             for(int i = D[c]; i != c; i = D[i])
61                 for(int j = R[i]; j != i; j = R[j])
62                     v[Col[j]] = false;
63         }
64     return ret;
65 }
66 void Dance(int d) {
67     if(d + f() >= ansd)return;
68     if(R[0] == 0) {
69         if(d < ansd)ansd = d;
70         return;
71     }
72     int c = R[0];
73     for(int i = R[0]; i != 0; i = R[i])
74         if(S[i] < S[c])
75             c = i;
76     for(int i = D[c]; i != c; i = D[i]) {
77         remove(i);
78         for(int j = R[i]; j != i; j = R[j])remove(j);
79         Dance(d+1);
80         for(int j = L[i]; j != i; j = L[j])resume(j);
81         resume(i);
82     }
83 }
84 };
85 DLX g;
86 int a[20][20];
87 int id[20][20];
88 int main() {
89     int n,m;
90     while(scanf("%d%d",&n,&m) == 2) {
91         int sz = 0;
92         memset(id,0,sizeof(id));
93         for(int i = 0; i < n; i++)
94             for(int j = 0; j < m; j++) {
95                 scanf("%d",&a[i][j]);
96                 if(a[i][j] == 1)id[i][j] = (++sz);
97             }
98         g.init(n*m,sz);
99         sz = 1;

```

```

100     int n1,m1;
101     scanf("%d%d",&n1,&m1);
102     for(int i = 0; i < n; i++)
103         for(int j = 0; j < m; j++) {
104             for(int x = 0; x < n1 && i + x < n; x++)
105                 for(int y = 0; y < m1 && j + y < m; y++)
106                     if(id[i+x][j+y])
107                         g.Link(sz,id[i+x][j+y]);
108             sz++;
109         }
110     g.ansd = INF;
111     g.Dance(0);
112     printf("%d\n",g.ansd);
113 }
114 return 0;
115 }

```

Listing 5.3: fzu1686.cpp

5.3 k 短路

```

1 #include <queue>
2 #include <cstdio>
3 #include <cstring>
4 #include <iostream>
5 #include <algorithm>
6 #define rep(i,n) for(int i=0;i<n;i++)
7 #define rep1(i,n) for(int i=1;i<=n;i++)
8 #define REP(i,a,b) for(int i=a;i<=b;i++)
9 #define foreach(i,vec) for(unsigned i=0;i<vec.size();i++)
10 #define pb push_back
11 #define RD3(x,y,z) scanf("%d%d%d",&x,&y,&z)
12 #define MP make_pair
13 #define Clear(x) memset(x,0,sizeof(x))
14 #define PII pair<int,int>
15
16 using namespace std;
17 //g(n) 是在状态空间中从初始节点到n节点的实际代价,
18 //h(n) 是从n到目标节点最佳路径的估计代价。
19 //t 第k次出堆即得k短
20
21 using namespace std;
22
23 const int MAXN = 1e3+5;
24 const int INF = 1e9;

```

```

25 vector< PII > adj[MAXN];
26 vector< PII > radj[MAXN];
27 int dis[MAXN], id[MAXN], n,m;
28 bool use[MAXN];
29
30 struct node{
31     int x,g,h;
32     node(){}
33     node(int a,int b,int c){
34         x = a;
35         g = b;
36         h = c;
37     }
38     friend bool operator < (node a,node b){
39         return a.g+a.h>b.g+b.h;
40     }
41 };
42
43 void spfa(int s){
44     queue<int> q;
45     q.push(s);
46     rep1(i,n)
47         dis[i] = INF;
48     dis[s] = 0;
49     memset(use,false,sizeof(use));
50     while(q.size()){
51         int x = q.front();
52         q.pop();
53         use[x] = false;
54         foreach(i,radj[x]){
55             int y = radj[x][i].first;
56             int t = radj[x][i].second + dis[x];
57             if(dis[y]>t){
58                 dis[y] = t;
59                 if(!use[y]){
60                     use[y] = true;
61                     q.push(y);
62                 }
63             }
64         }
65     }
66 }
67
68 int Astar(int s,int t,int k){
69     if(dis[s]==INF)return -1;

```

```

71     if(s==t)k ++;
72
73     Clear(id);
74
75     priority_queue< node > q;
76     q.push( node( s,0,dis[s] ) );
77
78     while(q.size()){
79         node p = q.top();
80         q.pop();
81         int x = p.x;
82
83         id[x] ++;
84         if(id[x]>k)continue;
85         if(id[t]==k)return p.g+p.h;
86
87         foreach(i,adj[x]){
88             int y = adj[x][i].first;
89             int w = adj[x][i].second;
90             q.push( node( y,p.g+w,dis[y] ) );
91         }
92     }
93     return -1;
94 }
95
96 int main(){
97     while(cin>>n>>m){
98         rep1(i,n){
99             adj[i].clear();
100             radj[i].clear();
101         }
102         int x,y,z;
103         while(m--){
104             RD3(x,y,z);
105             adj[x].pb( MP(y,z) );
106             radj[y].pb( MP(x,z) );
107         }
108
109         int s,t,k;
110         RD3(s,t,k);
111         spfa(t);
112         cout<<Astar(s,t,k)<<endl;
113     }
114     return 0;
115 }

```

Listing 5.4: poj2449.cpp

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
30 int comy(node A, node B)
31 {
32     return A.y < B.y;
33 }
34
35 void change(int pos, int x, long long cnt)
```

```
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         }
75         if (r & 1)
76         {
77             if (ti[r - 1] == now)
78             {
79                 if (tmp == g[r - 1]) cnt += gTot[r - 1];
80                 else if (tmp < g[r - 1])
```

```

81         {
82             tmp = g[r - 1];
83             cnt = gTot[r - 1];
84         }
85     }
86 }
87 return tmp;
88 }
89
90 void solve(int l, int r)
91 {
92     if (l == r) return ;
93     int mid = (l + r) >> 1;
94     solve(mid + 1, r);
95
96     sort(a + mid + 1, a + r + 1, comy);
97     sort(a + l, a + mid + 1, comy);
98
99     // MST(g, 0);
100    //MST(gTot, 0);
101    ++now;
102    int pos = r + 1;
103    REDD(i, mid, l)
104    {
105        for (; pos > mid + 1 && a[pos - 1].y >= a[i].y; --pos)
106        {
107            change(a[pos - 1].z, a[pos - 1].f, a[pos - 1].tot);
108        }
109
110        long long tmpTot;
111        int tmp = ask(a[i].z, zLim, tmpTot) + 1;
112        if (a[i].f == tmp) a[i].tot += tmpTot;
113        else if (a[i].f < tmp)
114        {
115            a[i].f = tmp;
116            a[i].tot = tmpTot;
117        }
118    }
119
120    sort(a + l, a + r + 1, comx);
121    solve(l, mid);
122 }
123
124 int INT()
125 {
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;
150             a[i].tot = 1;
151             dataZ.push_back(a[i].z);
152         }
153         sort(dataZ.begin(), dataZ.end());
154         dataZ.resize(unique(dataZ.begin(), dataZ.end()) - dataZ.
155         begin());
156         REPP(i, 1, n)
157         {
158             a[i].z = (lower_bound(dataZ.begin(), dataZ.end(), a[i].
159             z) - dataZ.begin()) + 1;
160         }
161         zLim = dataZ.size();
162         for (t = 1; t <= zLim + 1; t <= 1);
163         sort(a + 1, a + n + 1, comx);
164         solve(1, n);
165         int ans = 0;
166         long long cnt = 0;
167         REPP(i, 1, n)
168         {
169             if (ans == a[i].f) cnt += a[i].tot;
170             else if (ans < a[i].f) cnt = a[i].tot, ans = a[i].f;
171         }
172         printf("%d %lld\n", ans, cnt);

```

```
171 }
```

```
172 return 0;  
173 }
```

Listing 5.5: hdu4742.cpp