

ICPC Templates For Grooming

ZZY

November 7, 2020

Contents

1	图论	5
1.1	最短路	5
1.1.1	堆优化 Dijkstra	5
1.1.2	spfa	6
1.1.3	floyd 求传递闭包	7
1.1.4	floyd 求最小环	8
1.1.5	johnson 全源最短路	9
1.1.6	差分约束系统	11
1.1.7	经典例题	12
1.1.8	SCOI2011 糖果	13
1.1.9	16 ccpc final G	15
1.1.10	倍杀测量者	21
1.2	网络流	24
1.2.1	DICNIC	24
1.2.2	ISAP	25
1.2.3	MCMF	28
1.2.4	常见思路	29
1.3	匹配问题	30
1.3.1	匈牙利	30
1.3.2	HK	31
1.3.3	KM-DFS	33
1.3.4	KM-BFS	35
1.3.5	带花树	36
1.3.6	稳定婚姻问题	38
1.3.7	常见思路	38
1.4	2-SAT	39
1.4.1	输出任意解	39
1.4.2	输出字典序最小解	41
1.4.3	思路	42
1.4.4	经典例题	42
1.4.5	UVA 11930	43
1.4.6	cf27D	45
1.5	强连通	48
1.5.1	有向可有环图	48
1.6	双连通	49
1.6.1	割点桥	49

1.6.2	船新版本	51
1.6.3	思路	52
1.6.4	经典例题	53
1.6.5	POJ 2942	54
1.6.6	UVA 10972	58
1.6.7	T103492	59
1.6.8	P1407	61
1.7	欧拉回路	63
1.7.1	模板	63
1.7.2	知识点	64
1.7.3	经典例题	65
1.7.4	cf 21D	65
1.8	LCA	66
1.8.1	ST 表	66
1.8.2	离线	68
1.9	最大团	69
1.9.1	Bron-Kerbosch	69
1.9.2	常见思路	70
1.10	拓扑排序	70
1.10.1	toposort	70
2	计算几何	71
2.1	点	71
2.2	线	73
2.3	圆	76
2.4	多边形	77
2.5	半平面交	80
2.6	function	81
2.7	注意点	82
2.8	常出现的模型	83
2.9	奇怪的技巧	83
2.10	凸包	83
2.10.1	判断是否是稳定凸包	83
2.11	旋转卡壳	84
2.11.1	SCOI2007 最大土地面积	84
2.12	扫描线	90
2.12.1	矩形面积交 hdu1255	90
2.12.2	面积并坐标 double 版	92
2.12.3	面积并坐标 i64 版 $O(n \lg n)$	94
2.12.4	周长并 $O(n \lg n)$	95
2.13	半平面交	97
2.13.1	经典例题	97
2.13.2	codechef ALLPOLY	97
2.14	不知道分在哪一类	104
2.14.1	经典例题	104
2.14.2	cf 598C	104
2.14.3	2020ZJ 省赛 H	105

3 数据结构	108
3.1 线段树	108
3.1.1 线段树 __ 区间合并 hotel	108
3.2 树状数组	110
3.2.1 逆序对	110
3.3 主席树	111
3.3.1 静态查询区间第 k 大	111
3.3.2 动态查询区间第 k 大	112
3.4 字典树	114
3.4.1 trie 树	114
4 字符串	115
4.1 KMP	115
4.2 exKMP	116
4.3 Manacher	117
5 dp	118
5.1 树形 dp	118
5.1.1 树的重心	118
5.1.2 树上最远距离	119
6 树上问题	120
6.1 树的直径	120
7 STL	121
7.1 自定义排序	121
7.2 nth_element	122
8 其他问题	122
8.1 ST 表	122
8.1.1 ST 表	122
8.2 莫队	123
8.2.1 复杂度	123
8.2.2 普通莫队 LOJ 1188 $O(\sqrt{n} \times q)$	123
8.2.3 带修改莫队 cf 940F	124
8.3 二分注意点	127
8.4 LIS	127
8.5 尺取法	128
8.6 单调队列	128
8.7 一句话去重并生成新数组	129
8.8 输入一行看有多少个数	130
8.9 最小 (大) 表示法	130
8.10 随机数	130
8.11 输入日期输出周几	131
9 黑科技	131
9.1 IO	131
9.1.1 快读模板	131
9.1.2 __int128 输入输出模板	132
9.2 istringstream	132

9.3	unordered_map 防 hack 模板	133
9.4	杜教 BM	133
9.5	模拟退火	135
10	大数	136
10.1	java	136
10.1.1	坑点	136
10.1.2	输入	137
10.1.3	注意点	137
10.2	python	137
10.2.1	python	137

1 图论

1.1 最短路

1.1.1 堆优化 Dijkstra

```

1 //不能求最长路，不能处理带负权的边 用spfa搞
2 // stl优先队列是O(e log e)，手写二叉堆是O(e log v)，斐波那契堆是O(v log v + e)
3 //点的编号从0开始
4 #define i64 long long
5 const int N = (int)1e5+1000; //点数
6 const i64 INF = 0x3f3f3f3f3f3f3fLL;
7 struct node {
8     int id;
9     i64 w;
10     node(){}
11     node(int a, i64 b) : id(a), w(b) {} //hdu6805 美好的回忆:>
12     friend bool operator < (node a, node b) {return a.w > b.w;}
13 };
14 vector<node> G[N];
15 bool vis[N];
16 i64 dis[N];
17
18 void dij(int s, int n) {
19     priority_queue<node> q;
20     while (!q.empty()) q.pop();
21     node cur;
22     for (int i = 0; i <= n; ++i) { //另外,memset比for快哦
23         dis[i] = INF;
24         vis[i] = 0;
25     }
26     dis[s] = 0;
27     q.push(node(s, dis[s]));
28     while (!q.empty()) {
29         cur = q.top();
30         q.pop(); //另外,能return要的值就return哦(ccpcfinal 2016 G 枚举m条边跑dij那题),会
                快很多
31         if (vis[cur.id]) continue;
32         vis[cur.id] = 1;
33         for (node to : G[cur.id]) {
34             if (!vis[to.id] && dis[to.id] > dis[cur.id] + to.w) { //dis[to.id] > to.w
                就变成了堆优化prim
35                 dis[to.id] = dis[cur.id] + to.w;
36                 q.push(node(to.id, dis[to.id]));
37             }
38         }
39     }
40 }
41 void init(int n) {
42     for (int i = 0; i <= n; ++i) G[i].clear();
43 }
44 void addEdge(int a, int b, int c){
45     G[a].eb(node(b,c));

```

```

46     G[b].eb(node(a,c));
47 }
48
49 int main() {
50     int n, m, s, t;
51     cin >> n >> m >> s >> t; //输入
52     init(n); //初始化
53     forn(i, m) {
54         int u, v, w;
55         cin >> u >> v >> w; //输入
56         G[u].emplace_back(node(v, w)); //建图
57         G[v].emplace_back(node(u, w));
58     }
59     dij(s, n); //跑dij
60     cout << dis[t] << '\n';
61     return 0;
62 }

```

1.1.2 spfa

```

1 //最长路 or 判断正环: dis变成-INF, 松弛改成<
2 //判负环跑最短路, 判正环跑最长路
3 //一般时间复杂度 $O(k(\text{常数}) * E)$  最差复杂度 $O(V * E)$ 
4 //SPFA总的期望时间复杂度为 $O(n * \log n * \log(m/n) + m)$ 
5 #define i64 long long
6 const int N = 点数;
7 const i64 INF = 0x3f3f3f3f3f3f3f3f;
8 int n,m;
9 struct node {
10     int id;
11     i64 w;
12     node() {}
13     node(int a, i64 b) : id(a), w(b) {}
14 };
15 vector<node> G[N];
16 i64 dis[N],cnt[N];
17 bool vis[N];
18
19 bool spfa(int s, int n) { //s是起点,n是上界点数,点的编号从0开始
20     queue<node> q;
21     node cur;
22     for (int i = 0; i <= n; ++i){
23         dis[i] = INF;
24         vis[i]=cnt[i]=0;
25     }
26     vis[s] = 1;
27     dis[s] = 0;
28     q.push(node(s, dis[s]));
29     while (!q.empty()) {
30         cur = q.front();
31         q.pop();
32         //判断负 (正) 环在这++ , 如果>n(n为这张图的点数, 对应题目要求修改)就return true

```

```

33     ++cnt[cur.id];
34     if(cnt[cur.id]>n) return false;
35     vis[cur.id] = 0;
36     for (node to : G[cur.id]) {
37         if (dis[to.id] > dis[cur.id]+to.w) {
38
39             dis[to.id] = dis[cur.id]+to.w;
40             if (!vis[to.id]) {
41                 q.push(node(to.id, dis[to.id]));
42                 vis[to.id] = 1;
43             }
44         }
45     }
46     return true;
47 }
48
49 void init(int n) {
50     for (int i = 0; i <= n; ++i) G[i].clear();
51 }
52
53 int main() {
54
55     int n, m, s, t;
56     cin >> n >> m >> s >> t;
57     init(n);
58     forn(i, m) {
59         int u, v, w;
60         cin >> u >> v >> w;
61         G[u].pb(node(v, w));
62         G[v].pb(node(u, w));
63     }
64     spfa(s, n); //s是起点, n是点数
65     cout << dis[t] << '\n';
66
67     return 0;
68 }

```

1.1.3 floyd 求传递闭包

```

1  // hdu 1704
2  const int maxn = 600;
3  int tc, n, m, G[maxn][maxn];
4
5  void floyd() {
6      for (int k = 1; k <= n; ++k) {
7          for (int i = 1; i <= n; ++i) {
8              if (G[i][k]) {
9                  for (int j = 1; j <= n; ++j) {
10                     if (G[k][j]) {
11                         G[i][j] = 1;
12                     }
13                 }
14             }
15         }
16     }
17 }

```

```

14     }
15     }
16 }
17 }
18
19 int main() {
20
21     cin >> tc;
22     while (tc--) {
23         ms(G, 0);
24         cin >> n >> m;
25         forn(i, m) {
26             int a, b;
27             cin >> a >> b;
28             G[a][b] = 1;
29         }
30         floyd();
31         int cnt = 0;
32         for (int i = 1; i <= n; ++i) {
33             for (int j = i+1; j <= n; ++j) {
34                 if (!G[i][j] && !G[j][i]) ++cnt;
35             }
36         }
37         cout << cnt << '\n';
38     }
39
40     return 0;
41 }

```

1.1.4 floyd 求最小环

```

1 //AcWing 344 floyd求最小环并且按顺序输出最小环中的点
2 //hdu 1599
3 const int maxn = 110;
4 const i64 INF = 0x3f3f3f3f;
5 i64 G[maxn][maxn], dis[maxn][maxn], road[maxn][maxn], ans;
6 vi res;
7
8 void floyd(int n) {
9     int i, j, k;
10    for (k = 1; k <= n; ++k) {
11        for (i = 1; i < k; ++i) {
12            if (G[k][i] == INF) continue;
13            for (j = i+1; j < k; ++j) {
14                if (G[k][j] == INF) continue;
15                if (G[k][i]+G[k][j]+dis[i][j] < ans) {
16                    ans = G[k][i]+G[k][j]+dis[i][j];
17                    /* -----记录路径部分----- */
18                    res.clear();
19                    for (int temp = i; temp != j; temp = road[temp][j]) res.eb(temp);
20                    res.eb(j);
21                    res.eb(k);

```



```

22         /* ----- */
23     }
24 }
25 }
26 for (i = 1; i <= n; ++i) {
27     if (dis[i][k] == INF) continue;
28     for (j = 1; j <= n; ++j) {
29         if (dis[k][j] == INF) continue;
30         if (dis[i][k]+dis[k][j] < dis[i][j]) {
31             dis[i][j] = dis[i][k]+dis[k][j];
32             /* -----记录路径部分----- */
33             road[i][j] = road[i][k];
34             /* ----- */
35         }
36     }
37 }
38 }
39 }
40 void init(int n) {
41     for (int i = 1; i <= n; ++i) for (int j = 1; j <= n; ++j) G[i][j] = INF;
42     for (int i = 1; i <= n; ++i) for (int j = 1; j <= n; ++j) dis[i][j] = INF;
43     ans = INF;
44 }
45 int n, m;
46
47 int main() {
48     cin >> n >> m;
49     init(n);
50     forn(i, m) {
51         i64 u, v, val;
52         cin >> u >> v >> val;
53         G[u][v] = G[v][u] = dis[u][v] = dis[v][u] = min(dis[u][v], val);
54         road[u][v] = v; road[v][u] = u;
55     }
56     floyd(n);
57     if (ans == INF) cout << "No solution." << '\n';
58     else {
59         cout << ans << endl;
60         for (int x : res) cout << x << ' ';
61         cout << '\n';
62     }
63     return 0;
64 }

```

1.1.5 johnson 全源最短路

```

1 //洛谷 johnson全源最短路模板题
2 Johnson 算法则通过另外一种方法来给每条边重新标注边权。
3 我们新建一个虚拟节点（在这里我们就设它的编号为0）。从这个点向其他所有点连一条边权为 0 的边。
4 接下来用 Bellman-Ford 算法求出从 0 号点到其他所有点的最短路，记为dis[i]
5 假如存在一条从 u 点到 v 点，边权为 w 的边，则我们将该边的边权重新设置为 w+dis[u]-dis[v]
6 接下来以每个点为起点，跑 n 轮 Dijkstra 算法即可求出任意两点间的最短路了(要把偏移量dis[v]-

```

```

    dis[u]加上)。
7
8  const int N = 3100;
9  const i64 INF = 0x3f3f3f3f3f3f3fLL;
10 int n,m,u[6100],v[6100],w[6100];
11 struct node {
12     int id;
13     i64 w;
14     node(){}
15     node(int a, i64 b) : id(a), w(b) {} //hdu6805 美好的回忆:>
16     friend bool operator < (node a, node b) {return a.w > b.w;}
17 };
18 vector<node> G[N];
19 i64 dis[N],cnt[N],d2[N];
20 bool vis[N];
21 bool spfa(int s, int n) { //s是起点,n是点数,点的编号从0开始
22     queue<node> q;
23     node cur;
24     ms(vis,0);
25     ms(cnt,0);
26     for (int i = 0; i <= n; ++i) dis[i] = INF;
27     vis[s] = 1;
28     dis[s] = 0;
29     q.push(node(s, dis[s]));
30     while (!q.empty()) {
31         cur = q.front();
32         q.pop();
33         vis[cur.id] = 0;
34         //判断负(正)环在这++, 如果>n(n为这张图的点数,对应题目要求修改)就return true
35         ++cnt[cur.id];
36         if(cnt[cur.id]>n) return false;
37         for (node to : G[cur.id]) {
38             if (dis[to.id] > dis[cur.id]+to.w) {
39                 dis[to.id] = dis[cur.id]+to.w;
40                 if (!vis[to.id]) {
41                     q.push(node(to.id, dis[to.id]));
42                     vis[to.id] = 1;
43                 }
44             }
45         }
46     }
47     return true;
48 }
49 void dij(int s, int n) {
50     priority_queue<node> q;
51     while (!q.empty()) q.pop();
52     node cur;
53     for (int i = 0; i <= n; ++i) {
54         d2[i] = INF;
55         vis[i] = 0;
56     }
57     d2[s] = 0;
58     q.push(node(s, d2[s]));

```

```

59     while (!q.empty()) {
60         cur = q.top();
61         q.pop();
62         if (vis[cur.id]) continue;
63         vis[cur.id] = 1;
64         for (node to : G[cur.id]) {
65             if (!vis[to.id] && d2[to.id] > d2[cur.id] + to.w) { //dis[to.id] > to.w 就
                变成了堆优化prim
66                 d2[to.id] = d2[cur.id] + to.w;
67                 q.push(node(to.id, d2[to.id]));
68             }
69         }
70     }
71 }
72 void init(int n) {
73     for (int i = 0; i <= n; ++i) G[i].clear();
74 }
75 signed main() {
76     cin >> n >> m;
77     init(n+5);
78     for (int i = 0; i < m; ++i) {
79         cin >> u[i] >> v[i] >> w[i];
80         G[u[i]].eb(node(v[i], w[i]));
81     }
82     for (int i = 1; i <= n; ++i) G[0].eb(node(i, 0));
83     if (!spfa(0, n)) {
84         cout << -1 << '\n';
85         return 0;
86     }
87     for (int i = 1; i <= n; ++i) {
88         for (node &now : G[i]) {
89             now.w += dis[i] - dis[now.id];
90         }
91         for (int i = 1; i <= n; ++i) {
92             dij(i, n);
93             i64 ans = 0;
94             for (i64 j = 1; j <= n; ++j) {
95                 if (d2[j] == INF) ans += j * 1000000000LL; //1e9
96                 else ans += j * (d2[j] + dis[j] - dis[i]);
97             }
98             cout << ans << '\n';
99         }
100     return 0;
101 }

```

1.1.6 差分约束系统

- 1 若要使所有量的两两量最接近，则将源点到每个点的距离初始化为 0 。若要使得某一变量与其他变量的差最大，则将源点到各点的距离初始化为正无穷，其中之一为 0 。
- 2 若求最小方案则跑最长路(SCOI2011 糖果)，否则跑最短路
- 3 最长路建图只要把最短路建图边交换端点，值取个反就行。
- 4 跑最短路解法：a向b连一条权值为c的有向边表示 $b - a \leq c$ ，然后建一个超级汇点向所有点连权值为 0 的边，

```

    用SPFA判断是否存在负环，存在即无解。
5  别忘了特判自环答案是-1。
6  技巧 & 注意点：
7  1. 别忘了要自建一个虚拟源点向所有点连0的边！
8  2. 注意a a关系的特判，比如a>a, a-a>c之类..,不然会超时(SCOI2011 糖果)
9  3. a-b=c 可以转换成 a-b>=c && a-b<=c
10
11 最短路建法：
12 a-b<=c b->a (c)
13 a-b<c b->a (c-1)
14
15 a-b>=c a->b (-c)
16 a-b>c a->b (-c-1)
17
18 a==b b->a (0) && a->b (0)
19
20 a<=c S->a (c)
21 a<c S->a (c-1) ??
22
23 a>=c a->S (-c)
24 a>c a->S (-c-1) ??
25
26 a>=c*b b->a (log(c)) (跑最长路)
27 c*a>=b b->a (-log(c)) (跑最长路)

```

1.1.7 经典例题

```

1 最短路
2 Usaco2006 Dec Wormholes 虫洞
3 题意：
4 有n个点，m个边1，w个边2，边1就是普通的有一段长度T的路，边2是可以回到之前T时间的虫洞，现在
   问你能不能回到过去（在出发时刻之前回到出发点）？
5 思路：
6 注意边1要建双向边（正边），边2就建单向边（负边），然后就跑一边普通的spfa判负环就行。
7
8 16 ccpc final G Pandaland
9 时限:3000ms
10 题意：
11 给你一张图(不一定连通),求最小环。给你m行x1,y1,x2,y2,w,表示点(x1,y1)和点(x2,y2)之间有一条w
   的边。
12 1<=T<=50,1<=m<=4000,-10000<=xi,yi<=10000,1<=w<=1e5
13 思路：
14 n^3的floyd最小环肯定不行。
15 解法一：
16 枚举删每条边，删除这条边，然后跑最短路，这两点的距离+这个删掉边的权值就是最小环，维护答案。
17 堆优化dij里面还要剪两次支，就很玄学。
18 时间复杂度 O(m*m*log(m)) result: 900ms
19 解法2：
20 **最短的环除去一条边后一定是在这个图的最小生成树上**。先kruskal求最短路，然后通过枚举不
   在树上的边，求树上这条边的两点间距离，并加上边的权值维护答案即可。
21 注意:题目给的图可能有多个联通块。
22 时间复杂度O(m*log(m)) result: 200ms

```

23
 24 拆分约束
 25 SCOI2011 糖果
 26 题意：
 27 有 n 个小朋友，要满足 k 个需求。
 28 接下来 k 行，表示这些点需要满足的关系，每行三个数字 x, a, b 。
 29 如果 $x=1$ ，表示第 A 个小朋友分到的糖果必须和第 B 个小朋友分到的糖果一样多。
 30 如果 $x=2$ ，表示第 A 个小朋友分到的糖果必须少于第 B 个小朋友分到的糖果。
 31 如果 $x=3$ ，表示第 A 个小朋友分到的糖果必须不少于第 B 个小朋友分到的糖果。
 32 如果 $x=4$ ，表示第 A 个小朋友分到的糖果必须多于第 B 个小朋友分到的糖果。
 33 如果 $x=5$ ，表示第 A 个小朋友分到的糖果必须不多于第 B 个小朋友分到的糖果。
 34 问至少需要准备多少个糖果，才能使得每个小朋友都能够分到糖果，并且满足小朋友们所有的要求？
 35 思路：
 36 因为要方案数(和)最小，所以跑最长路(lbn讲的)。
 37
 38 倍杀测量者
 39 题意：
 40 第一行三个整数 n, s, t ，分别表示机房内选手人数，选手立下的 **flag** 总数和已知的选手分数的数量。 n 位选手从 1 开始编号至 n ，编号为 k 的选手被称为选手 k 。
 41 接下来 s 行，每行四个整数 o, A, B, k 。其中 $o=1$ 表示选手 A 立下了“我没 k 倍杀选手 B 就女装”的 **flag**， $o=2$ 表示选手 A 立下了“选手 B 把我 k 倍杀我就女装”的 **flag**。
 42 接下来 t 行，每行两个整数 C, x ，表示已知选手 C 的分数为 x 。
 43 若存在能保证赛后有选手女装的最大的 T ，则输出 T ，只有当输出与答案的绝对误差不超过 $(1e-4)$ 时才被视作正确输出。若不存在，输出 **"-1"**
 44 $1 \leq n, s \leq 1000, 1 \leq A, B, C, t \leq n, 1 \leq k \leq 10, 1 \leq x \leq 1e9$ 。保证输入中的 c 两辆不同。
 45 思路：
 46 二分+拆分约束
 47 我跑的最长路，最长路建边其实就是最短路建边，然后边调个头 && 边权取反就行。

1.1.8 SCOI2011 糖果

```

1  #include <bits/stdc++.h>
2  #define mp make_pair
3  #define fi first
4  #define se second
5  #define pb push_back
6  #define eb emplace_back
7  #define all(x) (x).begin(), (x).end()
8  #define rall(x) (x).rbegin(), (x).rend()
9  #define forn(i, n) for (int i = 0; i < (int)(n); ++i)
10 #define for1(i, n) for (int i = 1; i <= (int)(n); ++i)
11 #define ford(i, a, b) for (int i = (int)(a); i >= (int)b; --i)
12 #define fore(i, a, b) for (int i = (int)(a); i <= (int)(b); ++i)
13 #define rep(i, l, r) for (int i = (l); i <= (r); i++)
14 #define per(i, r, l) for (int i = (r); i >= (l); i--)
15 #define ms(x, y) memset(x, y, sizeof(x))
16 #define SZ(x) int(x.size())
17 using namespace std;
18 typedef pair<int, int> pii;
19 typedef vector<int> vi;
20 typedef vector<pii> vpi;
21 typedef vector<vi> vvi;
```

```

22 typedef long long i64;
23 typedef vector<i64> vi64;
24 typedef vector<vi64> vvi64;
25 typedef pair<i64, i64> pi64;
26 typedef double ld;
27 template<class T> bool uin(T &a, T b) { return a > b ? (a = b, true) : false; }
28 template<class T> bool uax(T &a, T b) { return a < b ? (a = b, true) : false; }
29 //1.integer overflow (1e5 * 1e5) (2e9 + 2e9)
30 //2.runtime error
31 //3.boundary condition
32 const int N = (int)1e5+100;
33 const i64 INF = 0x3f3f3f3f3f3f3f3fLL;
34 int n,m;
35 struct node {
36     int id;
37     i64 w;
38     node() {}
39     node(int a, i64 b) : id(a), w(b) {}
40 };
41 vector<node> G[N];
42 i64 dis[N],cnt[N];
43 bool vis[N];
44
45 bool spfa(int s, int n) { //s是起点,n是上界点数,点的编号从0开始
46     queue<node> q;
47     node cur;
48     for (int i = 0; i <= n; ++i){
49         dis[i] = -INF;
50         vis[i]=cnt[i]=0;
51     }
52     vis[s] = 1;
53     dis[s] = 0;
54     q.push(node(s, dis[s]));
55     while (!q.empty()) {
56         cur = q.front();
57         q.pop();
58         //判断负(正)环在这++ , 如果>n(n为这张图的点数,对应题目要求修改)就return true
59         ++cnt[cur.id];
60         if(cnt[cur.id]>n+1) return false;
61         vis[cur.id] = 0;
62         for (node to : G[cur.id]) {
63             if (dis[to.id] < dis[cur.id]+to.w) {
64                 dis[to.id] = dis[cur.id]+to.w;
65                 if (!vis[to.id]) {
66                     q.push(node(to.id, dis[to.id]));
67                     vis[to.id] = 1;
68                 }
69             }
70         }
71     }
72     return true;
73 }
74 void init(int n) {

```

```

75     for (int i = 0; i <= n; ++i) G[i].clear();
76 }
77 int main() {
78     ios::sync_with_stdio(false);
79     cin.tie(0);
80     cout.precision(10);
81     cout << fixed;
82 #ifdef LOCAL_DEFINE
83     freopen("input.txt", "r", stdin);
84 #endif
85     cin>>n>>m;
86     init(n+5);
87     int S=0;
88     bool pre=false;
89     forn(i, m){
90         int op,a,b;
91         cin>>op>>a>>b;
92         if(a==b) if(op==2 || op==4) pre=true;
93         if(op==1){
94             G[a].eb(node(b,0));
95             G[b].eb(node(a,0));
96         } else if(op==2){
97             G[a].eb(node(b,1));
98         } else if(op==3){
99             G[b].eb(node(a,0));
100        } else if(op==4){
101            G[b].eb(node(a,1));
102        } else if(op==5){
103            G[a].eb(node(b,0));
104        }
105    }
106    if(pre){ //特判，自己不可能比自己多或者少
107        cout<<-1<<'\n';
108        return 0;
109    }
110    for1(i, n) G[S].eb(node(i,1)); //因为每个儿童至少需要一个糖果，可以先连1的边，然后跑最
    长路
111    if(spfa(S,n)){
112        i64 sum=0;
113        for1(i, n) sum+=dis[i];
114        cout<<sum<<'\n';
115    } else cout<<-1<<'\n';
116 #ifdef LOCAL_DEFINE
117     cerr << "Time elapsed: " << 1.0 * clock() / CLOCKS_PER_SEC << " s.\n";
118 #endif
119     return 0;
120 }

```

1.1.9 16 ccpc final G

```

1 16 ccpc final G Pandaland
2 时限:3000ms

```

```

3  题意：
4  给你一张图(不一定连通)，求最小环。给你m行x1,y1,x2,y2,w,表示点(x1,y1)和点(x2,y2)之间有一条w
   的边。
5  1<=T<=50,1<=m<=4000,-10000<=xi,yi<=10000,1<=w<=1e5
6  思路：
7  n^3的floyd最小环肯定不行。
8  解法一：
9  枚举删每条边，删除这条边，然后跑最短路，这两点的距离+这个删掉边的权值就是最小环，维护答案。
10 堆优化dij里面还要剪两次支，就很玄学。
11 时间复杂度 O(m*m*log(m)) result: 900ms
12 解法2：
13 **最短的环除去一条边后一定是在这个图的最小生成树上**。先kruskal求最短路，然后通过枚举不在
   树上的边，求树上这条边的两点间距离，并加上边的权值维护答案即可。
14 注意：题目给的图可能有多个联通块。
15 时间复杂度O(m*log(m)) result: 200ms
16
17 堆优化dij写法：
18 #define i64 long long
19 const int N=1*(int)1e4+100;
20 const i64 INF=0x3f3f3f3f3f3f3fLL;
21 int tc,m,head[N],tot;
22 i64 ans;
23 vector< pair<int, int> > b;
24 struct ED{
25     int to,nxt,flag;
26     i64 val;
27 }star[N];
28 struct Edge{
29     int x1,y1,x2,y2;
30     i64 w;
31 }e[N];
32 struct node {
33     int id;
34     i64 w;
35     node(){}
36     node(int a, i64 b) : id(a), w(b) {} //hdu6805 美好的回忆:>
37     friend bool operator < (node a, node b) {return a.w > b.w;}
38 };
39 vector<node> G[N];
40 bool vis[N];
41 i64 dis[N];
42 void addEdge(int a,int b,int c){
43     star[tot].to=b;
44     star[tot].val=c;
45     star[tot].flag=0;
46     star[tot].nxt=head[a];
47     head[a]=tot++;
48 }
49 i64 dij(int s,int n,int ed){
50     priority_queue<node> q;
51     while(!q.empty()) q.pop();
52     node cur;
53     memset(dis,INF,sizeof(dis));

```



```

54     memset(vis,0,sizeof(vis));
55     dis[s]=0;
56     q.push(node(s,dis[s]));
57     while(!q.empty()){
58         cur=q.top();
59         if(cur.id == ed) return dis[ed]; //剪枝1
60         if(cur.w>ans) return INF; //剪枝2
61         q.pop();
62         if(vis[cur.id]) continue;
63         vis[cur.id]=1;
64         for(int i=head[cur.id]; ~i; i=star[i].nxt){
65             if(star[i].flag){
66                 continue;
67             }
68             if(!vis[star[i].to] && dis[star[i].to]>dis[cur.id]+star[i].val){
69                 dis[star[i].to]=dis[cur.id]+star[i].val;
70                 q.push(node(star[i].to,dis[star[i].to]));
71             }
72         }
73     }
74     return dis[ed];
75 }
76 void init(int n){
77     for(int i=0; i<=n; ++i) G[i].clear();
78 }
79 unordered_map<int,int> idx;
80 int main(){
81     int kase=1;
82     cin>>tc;
83     while(tc--){
84         b.clear();
85         tot=0;
86         memset(head,-1,sizeof(head));
87         cin>>m;
88         for(int i=0; i<m; ++i){
89             cin>>e[i].x1>>e[i].y1>>e[i].x2>>e[i].y2>>e[i].w;
90             b.emplace_back(make_pair(e[i].x1,e[i].y1));
91             b.emplace_back(make_pair(e[i].x2,e[i].y2));
92         }
93         sort(b.begin(),b.end());
94         b.resize(unique(b.begin(),b.end())-b.begin());
95         for(int i=0; i<m; ++i){
96             int u=lower_bound(b.begin(),b.end(),make_pair(e[i].x1,e[i].y1))-b.begin
97                 ();
98             int v=lower_bound(b.begin(),b.end(),make_pair(e[i].x2,e[i].y2))-b.begin
99                 ();
100             addEdge(u,v,e[i].w);
101             addEdge(v,u,e[i].w);
102         }
103         ans=(i64)1e10;
104         for(int i=0; i<m; ++i){
105             int st=lower_bound(b.begin(),b.end(),make_pair(e[i].x1,e[i].y1))-b.begin
106                 ();

```

```

104         int ed=lower_bound(b.begin(),b.end(),make_pair(e[i].x22,e[i].y22))-b.begin
            ();
105         star[i*2+1].flag=1;
106         star[i*2].flag=1;
107         i64 temp=e[i].w+dij(st,2*m+5,ed);
108         ans=min(ans,temp);
109         star[i*2+1].flag=0;
110         star[i*2].flag=0;
111     }
112     if(ans==(i64)1e10) cout<<"Case #"<<kase++<<": "<<0<<'\n';
113     else cout<<"Case #"<<kase++<<": "<<ans<<'\n';
114 }
115 return 0;
116 }
117
118
119
120
121 LCA写法:
122 #define i64 long long
123 const int N = 2*(int)10000+100; //要开两倍点数量的大小 (欧拉序长度)
124 vector< pair<int,int> > b;
125 int n,fa[N];
126 int x11[N],y11[N],x22[N],y22[N],w11[N],chk1ca[N];
127 struct node{
128     int a,b,used;
129     i64 w;
130     node() {
131         used=0;
132     }
133     node(int _a,int _b,i64 _w){
134         a=_a;
135         b=_b;
136         w=_w;
137         used=0;
138     }
139     bool operator < (const node &b) const{
140         return w<b.w;
141     }
142 }nd[N];
143 struct LCA
144 {
145     #define type long long
146     struct node{int to;type w;node(){}}node(int _to,type _w):to(_to),w(_w){};
147     type dist[N];
148     int path[N],dep[N],loc[N],len[N],LOG[N],all,n;
149     int dp[25][N], point[25][N]; //2^20 == 1e6 2^25 == 3e7
150     vector<node> G[N];
151     void dfs(int u, int now) {
152         chk1ca[u]=1; //因为有多棵不连通的树
153         path[++all] = u;
154         loc[u] = all;
155         dep[all] = now;

```

```

156     for (node cur : G[u]) {
157         int v = cur.to;
158         if (loc[v]) continue;
159         len[v] = now+1;
160         dist[v] = dist[u]+cur.w;
161         dfs(v, now+1);
162         path[++all] = u;
163         dep[all] = now;
164     }
165 }
166 void initRMQ(int n)
167 {
168     LOG[0] = -1;
169     for (int i = 1; i <= all; ++i) {
170         dp[0][i] = dep[i];
171         point[0][i] = path[i];
172         LOG[i] = ((i&(i-1)) == 0 ? LOG[i-1]+1 : LOG[i-1]);
173     }
174     for (int i = 1; (1<<i) <= all; ++i) {
175         for (int j = 1; j+(1<<i)-1 <= all; ++j) {
176             if (dp[i-1][j] < dp[i-1][j+(1<<(i-1))]) {
177                 dp[i][j] = dp[i-1][j];
178                 point[i][j] = point[i-1][j];
179             } else {
180                 dp[i][j] = dp[i-1][j+(1<<(i-1))];
181                 point[i][j] = point[i-1][j+(1<<(i-1))];
182             }
183         }
184     }
185 }
186 int queryLCA(int l,int r)
187 {
188     l = loc[l]; r = loc[r];
189     if(l>r) swap(l,r);
190     int k = LOG[r-l+1];
191     /*
192     貌似下面这种写法对于某些数据情况更快，对于某些数据也更慢- -
193     记得把上面预处理的LOG删了
194     P 3379
195     int k=0;
196     while((1<<k)<=r-l+1) k++;
197     k--;
198     */
199     if(dp[k][l] < dp[k][r-(1<<k)+1]) return point[k][l];
200     else return point[k][r-(1<<k)+1];
201 }
202
203 type getDist(int a,int b){return dist[a]+dist[b]-2*dist[queryLCA(a,b)];}
204 int getLen(int a,int b){return len[a]+len[b]-2*len[queryLCA(a,b)];}
205 void init(int _n)
206 {
207     n = _n;
208     all = 0;

```

```

209     for(int i = 0; i <= n; i++)
210     {
211         loc[i] = 0;
212         dist[i] = 0;
213         len[i] = 0;
214         G[i].clear();
215     }
216 }
217 void addEdge(int a, int b, type w=1)
218 {
219     G[a].emplace_back(node(b,w));
220     G[b].emplace_back(node(a,w));
221 }
222 void solve(int root)
223 {
224     dfs(root, 1);
225     initRMQ(all);
226 }
227 #undef type
228 }lca;
229 int findRoot(int x){return (x==fa[x]?x:fa[x]=findRoot(fa[x]));}
230 int main() {
231     int tc, kase=1;
232     cin>>tc;
233     while(tc--){
234         b.clear();
235         cin>>n;
236         for(int i=0; i<n; i++){
237             cin>>x11[i]>>y11[i]>>x22[i]>>y22[i]>>w11[i];
238             b.eb(mp(x11[i],y11[i]));
239             b.eb(mp(x22[i],y22[i]));
240         }
241         sort(all(b));
242         b.resize(unique(all(b))-b.begin());
243         for(int i=0; i<n; ++i){
244             int ai=lower_bound(all(b),mp(x11[i],y11[i]))-b.begin();
245             int bi=lower_bound(all(b),mp(x22[i],y22[i]))-b.begin();
246             nd[i]=node(ai,bi,w11[i]);
247         }
248         lca.init(SZ(b)+5);
249         for(int i=0; i<SZ(b)+5; ++i){
250             chklca[i]=0;
251             fa[i]=i;
252         }
253         sort(nd, nd+n);
254         int rt=-1;
255         for(int i=0; i<n; ++i){
256             int aa=findRoot(nd[i].a);
257             int bb=findRoot(nd[i].b);
258             if(aa!=bb){
259                 fa[bb]=aa;
260                 lca.addEdge(nd[i].a, nd[i].b, nd[i].w);
261                 nd[i].used=1;

```

```

262     }
263 }
264 for(int i=0; i<SZ(b); ++i){
265     if(!chk1ca[i]){
266         lca.solve(i);
267     }
268 }
269 i64 ans=(i64)1e10;
270 for(int i=0; i<n; ++i){
271     if(nd[i].used) continue;
272     ans=min(ans,lca.getDist(nd[i].a,nd[i].b)+nd[i].w);
273 }
274 if(ans==(i64)1e10) ans=0; //cannot find cycle
275 cout<<"Case #"<<kase++<<": "<<ans<<'\n';
276 }
277 return 0;
278 }

```

1.1.10 倍杀测量者

```

1  倍杀测量者
2  题意：
3  第一行三个整数n,s,t,分别表示机房内选手人数，选手立下的flag总数和已知的选手分数的数量。n位选
   手从1开始编号至n，编号为k的选手被称为选手k。
4  接下来s行，每行四个整数o,A,B,k。其中 o=1 表示选手A立下了“我没k倍杀选手B就女装”的flag，o
   =2 表示选手A立下了“选手B把我k倍杀我就女装”的flag。
5  接下来t行，每行两个整数C,x,表示已知选手C的分数为x。
6  若存在能保证赛后有选手女装的最大的T，则输出T，只有当输出与答案的绝对误差不超过(1e-4)时才被
   视作正确输出。若不存在，输出"-1"
7  1<=n,s<=1000,1<=A,B,C,t<=n,1<=j<=10,1<=x<=1e9。 保证输入中的c两辆不同。
8  思路：
9  二分+拆分约束
10 我跑的最长路，最长路建边其实就是最短路建边，然后边调个头 && 边权取反就行。
11
12 #include <bits/stdc++.h>
13 #define mp make_pair
14 #define fi first
15 #define se second
16 #define pb push_back
17 #define eb emplace_back
18 #define all(x) (x).begin(), (x).end()
19 #define rall(x) (x).rbegin(), (x).rend()
20 #define forn(i, n) for (int i = 0; i < (int)(n); ++i)
21 #define for1(i, n) for (int i = 1; i <= (int)(n); ++i)
22 #define ford(i, a, b) for (int i = (int)(a); i >= (int)b; --i)
23 #define fore(i, a, b) for (int i = (int)(a); i <= (int)(b); ++i)
24 #define rep(i, l, r) for (int i = (l); i <= (r); i++)
25 #define per(i, r, l) for (int i = (r); i >= (l); i--)
26 #define ms(x, y) memset(x, y, sizeof(x))
27 #define SZ(x) int(x.size())
28 using namespace std;
29 typedef pair<int, int> pii;

```

```

30 typedef vector<int> vi;
31 typedef vector<pii> vpi;
32 typedef vector<vi> vvi;
33 typedef long long i64;
34 typedef vector<i64> vi64;
35 typedef vector<vi64> vvi64;
36 typedef pair<i64, i64> pi64;
37 typedef double ld;
38 template<class T> bool uin(T &a, T b) { return a > b ? (a = b, true) : false; }
39 template<class T> bool uax(T &a, T b) { return a < b ? (a = b, true) : false; }
40 //1.integer overflow (1e5 * 1e5) (2e9 + 2e9)
41 //2.runtime error
42 //3.boundary condition
43 const int N = 2000;
44 const i64 INF = 0x3f3f3f3f3f3f3f3fLL;
45 int n,s,t;
46 struct node {
47     int id,type;
48     double w;
49     node() {}
50     node(int a, double b, int _type=-1) : id(a), w(b), type(_type) {}
51 };
52 vector<node> G[N];
53 double dis[N];
54 int cnt[N];
55 bool vis[N];
56
57 bool spfa(int s, int n, double now) { //s是起点,n是上界点数,点的编号从0开始
58     queue<node> q;
59     node cur;
60     for (int i = 0; i <= n; ++i){
61         dis[i] = -INF;
62         vis[i]=cnt[i]=0;
63     }
64     vis[s] = 1;
65     dis[s] = 0;
66     q.push(node(s, dis[s]));
67     while (!q.empty()) {
68         cur = q.front();
69         q.pop();
70         //判断负(正)环在这++, 如果>n(n为这张图的点数,对应题目要求修改)就return true
71         ++cnt[cur.id];
72         if(cnt[cur.id]>n+1) return false;
73         vis[cur.id] = 0;
74         for (node to : G[cur.id]) {
75             double ew=to.w;
76             if(to.type==1) ew=log2(ew-now);
77             else if(to.type==2) ew=-log2(ew+now);
78             if (dis[to.id] < dis[cur.id]+ew) {
79                 dis[to.id] = dis[cur.id]+ew;
80                 if (!vis[to.id]) {
81                     q.push(node(to.id, dis[to.id]));
82                     vis[to.id] = 1;

```

```

83         }
84     }
85 }
86 }
87     return true;
88 }
89 void init(int n) {
90     for (int i = 0; i <= n; ++i) G[i].clear();
91 }
92 int main() {
93     ios::sync_with_stdio(false);
94     cin.tie(0);
95     cout.precision(10);
96     cout<<fixed;
97 #ifdef LOCAL_DEFINE
98     freopen("input.txt", "r", stdin);
99 #endif
100     cin>>n>>s>>t;
101     double l=0,r=10,ans; //下面二分的准备
102     int S=0; //源点
103     forn(i, s){
104         int o,a,b;
105         double k;
106         cin>>o>>a>>b>>k;
107         G[b].eb(node(a,k,o));
108         if(o==1) uin(r,k); //K-T要>0
109     }
110     forn(i, t){
111         int c,x;
112         cin>>c>>x;
113         G[c].eb(node(S,-log2(x),3));
114         G[S].eb(node(c,log2(x),3));
115     }
116     for(int i=1; i<=n; ++i) G[S].eb(node(i,0,3));
117     if(spfa(S,n,0)){ //不管k怎么取，都没人女装
118         cout<<-1<<'\n';
119         return 0;
120     }
121     forn(i,50){
122         double mid=(l+r)*0.5;
123         if(!spfa(S,n,mid)){ //有人立的flag假了，要女装
124             l=mid;
125             ans=mid;
126         } else r=mid;
127     }
128     cout<<ans<<'\n';
129 #ifdef LOCAL_DEFINE
130     cerr << "Time elapsed: " << 1.0 * clock() / CLOCKS_PER_SEC << " s.\n";
131 #endif
132     return 0;
133 }

```

1.2 网络流

1.2.1 DICNIC

```

1  /*
2  网络流dinic复杂度
3  上届 $O((n^2)m)$ 
4  若所有边容量为1, $O(\min(n^{1/3}, m^{1/2})m)$ 
5  二分图 $O(n^{1/2}m)$ 
6  */
7  //如果是无向图, 加的反向边流量也为w, 而不是0
8  //要开long long不如直接#define int long long, 注意要把int()改成(int)()
9  const int N = EDIT+100; //点数
10 const int INF = 0x3f3f3f3f; //ll const i64 INF =0x8个3fLL; .1.
11 struct Edge{
12     int from, to, cap, flow; //如果要开ll的话, 这边也要开ll .2.
13     Edge(int u, int v, int c, int f) //如果要开ll的话, 这边也要开ll .3.
14         : from(u), to(v), cap(c), flow(f) {}
15 };
16 struct Dicnic {
17     #define Type int
18     int n, m, s, t; //节点数, 边数 (包括反向弧), 源点编号和汇点编号
19     vector<Edge> edges; //边表. edge[e]和edge[e^1]互为反向弧
20     vector<int> G[N]; //邻接表, G[i][j]表示节点i的第j条边在e数组中的编号
21     bool vis[N]; //BFS使用
22     Type d[N]; //从起点到i的距离
23     int cur[N]; //当前弧下标
24     void init(int n) {
25         this->n = n;
26         for (int i = 0; i <= n; ++i) G[i].clear();
27         edges.clear();
28     }
29     void addEdge(int from, int to, Type cap) {
30         edges.emplace_back(Edge(from, to, cap, 0));
31         edges.emplace_back(Edge(to, from, 0, 0));
32         m = int(edges.size());
33         G[from].emplace_back(m-2);
34         G[to].emplace_back(m-1);
35     }
36     bool BFS() {
37         memset(vis, 0, sizeof(vis));
38         memset(d, 0, sizeof(d));
39         queue<int> q;
40         while (!q.empty()) q.pop();
41         q.push(s);
42         d[s] = 0;
43         vis[s] = 1;
44         while (!q.empty()) {
45             int x = q.front();
46             q.pop();
47             for (int i = 0; i < int(G[x].size()); ++i) {
48                 Edge &e = edges[G[x][i]];
49                 if (!vis[e.to] && e.cap > e.flow) {

```



```

50         vis[e.to] = 1;
51         d[e.to] = d[x] + 1;
52         q.push(e.to);
53     }
54 }
55 }
56 return vis[t];
57 }
58 Type DFS(int x, Type a) {
59     if (x == t || a == 0) return a;
60     Type flow = 0, f;
61     for (int &i = cur[x]; i < int(G[x].size()); ++i) { //从上次考虑的弧
62         Edge &e = edges[G[x][i]];
63         if (d[x]+1 == d[e.to] && (f = DFS(e.to, min(a, e.cap-e.flow))) > 0) {
64             e.flow += f;
65             edges[G[x][i]^1].flow -= f;
66             flow += f;
67             a -= f;
68             if (a == 0) break;
69         }
70     }
71     return flow;
72 }
73 Type Maxflow(int s, int t) {
74     this->s = s; this->t = t;
75     Type flow = 0;
76     while (BFS()) {
77         memset(cur, 0, sizeof(cur));
78         flow += DFS(s, INF);
79     }
80     return flow;
81 }
82 #undef Type
83 }dicnic;
84
85 int main() {
86     cin >> n >> m >> s >> t; //V, E, 原点, 汇点
87     dicnic.init(n); //初始化
88     while (m--) {
89         int u, v, w;
90         cin >> u >> v >> w;
91         dicnic.addEdge(u, v, w); //建图
92     }
93     cout << dicnic.Maxflow(s, t) << '\n'; //跑最大流
94     return 0;
95 }

```

1.2.2 ISAP

```

1 //时间复杂度  $O(V^2 \cdot E)$ 
2 const int maxn = "EDIT"; //点数
3 const int INF = 0x3f3f3f3f;

```

```

4 struct Edge{
5     int from, to, cap, flow;
6     Edge(int u, int v, int c, int f)
7         : from(u), to(v), cap(c), flow(f) {}
8 };
9 struct ISAP {
10     int n, m, s, t; //节点数, 边数 (包括反向弧), 原点编号和汇点编号
11     vector<Edge> edges; //边表。edges[e]和edges[e^1]互为反向弧
12     vector<int> G[maxn]; //邻接表, G[i][j]表示节点i的第j条边在e数组中的序号
13     bool vis[maxn]; //BFS使用
14     int d[maxn]; //起点到i的距离
15     int cur[maxn]; //当前弧下标
16     int p[maxn]; //可增广路上的一条弧
17     int num[maxn]; //距离标号计数
18     void init(int n) {
19         this->n = n;
20         for (int i = 0; i <= n; ++i) {
21             d[i] = INF;
22             num[i] = vis[i] = cur[i] = 0;
23             G[i].clear();
24         }
25         edges.clear();
26     }
27     void addEdge(int from, int to, int cap) {
28         edges.emplace_back(from, to, cap, 0);
29         edges.emplace_back(to, from, 0, 0);
30         int m = int(edges.size());
31         G[from].emplace_back(m - 2);
32         G[to].emplace_back(m - 1);
33     }
34     int Augument() {
35         int x = t, a = INF;
36         while (x != s) {
37             Edge &e = edges[p[x]];
38             a = min(a, e.cap - e.flow);
39             x = edges[p[x]].from;
40         }
41         x = t;
42         while (x != s) {
43             edges[p[x]].flow += a;
44             edges[p[x] ^ 1].flow -= a;
45             x = edges[p[x]].from;
46         }
47         return a;
48     }
49     void BFS() {
50         queue<int> q;
51         while (!q.empty()) q.pop();
52         q.push(t);
53         d[t] = 0;
54         vis[t] = 1;
55         while (!q.empty()) {
56             int x = q.front();

```

```

57         q.pop();
58         int len = int(G[x].size());
59         for (int i = 0; i < len; ++i) {
60             Edge &e = edges[G[x][i] ^ 1];
61             if (!vis[e.from] && e.cap > e.flow) {
62                 vis[e.from] = 1;
63                 d[e.from] = d[x] + 1;
64                 q.push(e.from);
65             }
66         }
67     }
68 }
69 int Maxflow(int s, int t) {
70     this->s = s;
71     this->t = t;
72     int flow = 0;
73     BFS();
74     if (d[s] >= n) return 0;
75     for (int i = 1; i <= n; ++i)
76         if (d[i] < INF) num[d[i]]++;
77     int x = s;
78     while (d[s] < n) {
79         if (x == t) {
80             flow += Augument();
81             x = s;
82         }
83         int ok = 0;
84         for (int i = cur[x]; i < int(G[x].size()); ++i) {
85             Edge &e = edges[G[x][i]];
86             if (e.cap > e.flow && d[x] == d[e.to] + 1) {
87                 ok = 1;
88                 p[e.to] = G[x][i];
89                 cur[x] = i;
90                 x = e.to;
91                 break;
92             }
93         }
94         if (!ok) { //Retreat
95             int m = n - 1;
96             for (int i = 0; i < int(G[x].size()); ++i) {
97                 Edge &e = edges[G[x][i]];
98                 if (e.cap > e.flow) m = min(m, d[e.to]);
99             }
100             if (--num[d[x]] == 0) break; //gap优化
101             num[d[x] = m + 1]++;
102             cur[x] = 0;
103             if (x != s) x = edges[p[x]].from;
104         }
105     }
106     return flow;
107 }
108 }isap;
109

```

```

110 int main() {
111
112     cin >> n >> m >> s >> t;
113     isap.init(n);
114     while (m--) {
115         int u, v, w;
116         cin >> u >> v >> w;
117         isap.addEdge(u, v, w);
118     }
119     cout << isap.Maxflow(s, t) << '\n';
120
121     return 0;
122 }

```

1.2.3 MCMF

```

1 //洛谷P3381
2 #define ll long long
3 const int maxn = 5000+100;
4 const int INF = INT_MAX;
5 struct Edge{
6     int from, to, cap, flow, cost;
7     Edge(int u, int v, int c, int f, int cc)
8         : from(u), to(v), cap(c), flow(f), cost(cc) {}
9 };
10 struct MCMF {
11     int n, m;
12     vector<Edge> edges;
13     vector<int> G[maxn];
14     int inq[maxn]; //是否在队列中
15     int d[maxn]; //bellmanford
16     int p[maxn]; //上一条弧
17     int a[maxn]; //可改进量
18     void init(int n) {
19         this->n = n;
20         for (int i = 0; i <= n; ++i) G[i].clear();
21         edges.clear();
22     }
23     void addEdge(int from, int to, int cap, int cost) {
24         edges.emplace_back(from, to, cap, 0, cost);
25         edges.emplace_back(to, from, 0, 0, -cost);
26         m = int(edges.size());
27         G[from].push_back(m - 2);
28         G[to].push_back(m - 1);
29     }
30     bool spfa(int s, int t, int &flow, ll &cost) {
31         for (int i = 1; i <= n; ++i) d[i] = INF;
32         memset(inq, 0, sizeof(inq));
33         d[s] = 0;
34         inq[s] = 1;
35         p[s] = 0;
36         a[s] = INF;

```

```

37     queue<int> q;
38     q.push(s);
39     while (!q.empty()) {
40         int u = q.front();
41         q.pop();
42         inq[u] = 0;
43         for (int i = 0; i < int(G[u].size()); ++i) {
44             Edge &e = edges[G[u][i]];
45             if (e.cap > e.flow && d[e.to] > d[u] + e.cost) {
46                 d[e.to] = d[u] + e.cost;
47                 p[e.to] = G[u][i];
48                 a[e.to] = min(a[u], e.cap - e.flow);
49                 if (!inq[e.to]) {
50                     q.push(e.to);
51                     inq[e.to] = 1;
52                 }
53             }
54         }
55     }
56     if (d[t] == INF) return false;
57     flow += a[t];
58     cost += (ll)d[t] * (ll)a[t];
59     for (int u = t; u != s; u = edges[p[u]].from) {
60         edges[p[u]].flow += a[t];
61         edges[p[u] ^ 1].flow -= a[t];
62     }
63     return true;
64 }
65 int MincostMaxflow(int s, int t, ll &cost) {
66     int flow = 0;
67     cost = 0;
68     while (spfa(s, t, flow, cost));
69     return flow;
70 }
71 }mcmf;
72 int n, m, s, t;
73
74 int main() {
75     cin >> n >> m >> s >> t;
76     mcmf.init(n); //初始化
77     for (int i = 0; i < m; ++i) {
78         int u, v, w, f;
79         cin >> u >> v >> w >> f;
80         mcmf.addEdge(u, v, w, f); //建图
81     }
82     ll cost = 0;
83     cout << mcmf.MincostMaxflow(s, t, cost) << ' ' << cost << '\n';
84     return 0;
85 }

```

1.2.4 常见思路

- 1 如果是无向图，那么建边的时候，反向边的流量不是0，而是和正向边的流量相同。
- 2
- 3 最大权闭合子图：
- 4 有一个有向图，每一个点都有一个权值（可以为正或负或0），选择一个权值和最大的子图，使得每个点的后继都在子图里面，这个子图就叫最大权闭合子图。
- 5 最大权子图一个经典的网络流问题，如果一个点被选择了则后继必须被选择，那么称该图是 闭合的，因此该问题叫做最大权闭合子图问题。可以使用最小割解决。
- 6 具体的建图方法为：
- 7 源点向所有正权点连接一条容量为权值的边
- 8 保留原图中所有的边，容量为正无穷
- 9 所有负权点向汇点连接一条容量为权值绝对值的边
- 10 则原图的最大权闭合子图的点权和即为所有正权点权值之和减去建出的网络流图的最小割。
- 11 以下约定源点为 **ss**，汇点为 **tt**。
- 12 在最小割图上，如果割掉 **ss** 和 **uu** 之间的边，代表不选择 **uu** 进入子图，如果割掉 **vv** 和 **tt** 之间的边，代表选择 **vv** 进入子图。
- 13 小技巧：**dicnic**里的**d**数组不为0，就说明那个点要取
- 14 求完最小割后，如果点 **ss** 与 **ii** 相连，那么子图上会选择点 **ii**，如果 **ii** 与 **tt** 相连，则不选择点 **ii**。
- 15
- 16 二分图最大点权独立集 = 所有的点权 - 二分图最小点权覆盖集（最小割） 方格取数问题
- 17 最小点权覆盖集的建图方法：
- 18 1、增加源点 **s**，连接 **s** 到 **x** 集合中所有点，边权是相应点的点权
- 19 2、增加汇点 **t**，连接 **y** 集合中所有点到 **t**，边权是相应点的点权
- 20 3、对原图中的边，将边权变成无穷大
- 21
- 22 最小割割边唯一性判断：
- 23 在残余网络上（非满流的边）跑**tarjan**求出所有SCC，记**id[u]**为点**u**所在SCC的编号。显然有**id[s]!=id[t]**（否则**s**到**t**有通路，能继续增广）。
- 24 ①对于任意一条满流边**(u,v)**，**(u,v)**能够出现在某个最小割集中，当且仅当**id[u]!=id[v]**；
- 25 ②对于任意一条满流边**(u,v)**，**(u,v)**必定出现在最小割集中，当且仅当**id[u]==id[s]**且**id[v]==id[t]**。

1.3 匹配问题

1.3.1 匈牙利

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  const int maxn = 600;
6
7  int nl, nr, m;
8  int mx[maxn], my[maxn], vis[maxn];
9  vector<int> G[maxn];
10
11 bool aug(int x) {
12     for (int to : G[x]) {
13         if (vis[to] == -1) {
14             vis[to] = 1;
15             if (my[to] == -1 || aug(my[to])) {
16                 my[to] = x;
17                 mx[x] = to;

```

```

18         return true;
19     }
20 }
21 }
22 return false;
23 }
24
25 int main() {
26     scanf("%d%d", &n1, &nr, &m);
27     for (int i = 0; i < m; ++i) {
28         int u, v;
29         scanf("%d", &u, &v);
30         G[u].emplace_back(v);
31     }
32     int maxMatch = 0;
33     memset(my, -1, sizeof(my));
34     memset(mx, -1, sizeof(mx));
35     for (int i = 1; i <= n1; ++i) {
36         memset(vis, -1, sizeof(vis));
37         if (aug(i)) ++maxMatch;
38     }
39     printf("%d\n", maxMatch);
40     for (int i = 1; i <= n1; ++i) {
41         printf("%d ", mx[i] == -1 ? 0 : mx[i]);
42     } puts("");
43     return 0;
44 }

```

1.3.2 HK

```

1  /*
2     别忘了给un赋值为左端点的个数 在主函数里赋值un=...
3     如果从0~ (n-1)开始编号, 就改00处地方
4     如果是多组数据别忘了清空G
5  */
6  /*****hdu2389*****/
7  int tc, t, m, n;
8  struct coordinate {
9      int x, y, speed;
10 }human[3100], umb[3100];
11 /*****/
12 const int maxn = 6100;
13 const int INF = 0x3f3f3f3f;
14 vector< vector<int> > G(maxn);
15 int un; //un为左端的顶点数, 编号1~(un)
16 int mx[maxn], my[maxn];
17 int dx[maxn], dy[maxn];
18 int dis;
19 bool vis[maxn];
20 bool SearchP() {
21     queue<int> q;
22     dis = INF;

```

```

23     memset(dx, -1, sizeof(dx));
24     memset(dy, -1, sizeof(dy));
25     for (int i = 1; i <= un; ++i) { //编号1~(un) ①
26         if(mx[i] == -1) {
27             q.push(i);
28             dx[i] = 0;
29         }
30     }
31     while (!q.empty()) {
32         int u = q.front();
33         q.pop();
34         if (dx[u] > dis) break;
35         int sz = int(G[u].size());
36         for (int i = 0; i < sz; ++i) {
37             int v = G[u][i];
38             if (dy[v] == -1) {
39                 dy[v] = dx[u]+1;
40                 if (my[v] == -1) dis = dy[v];
41             } else {
42                 dx[my[v]] = dy[v]+1;
43                 q.push(my[v]);
44             }
45         }
46     }
47 }
48 return (dis!=INF);
49 }
50 bool dfs(int u) {
51     int sz = int(G[u].size());
52     for (int i = 0; i < sz; ++i) {
53         int v = G[u][i];
54         if (!vis[v] && dy[v] == dx[u]+1) {
55             vis[v] = true;
56             if (my[v] != -1 && dy[v] == dis) continue;
57             if (my[v] == -1 || dfs(my[v])) {
58                 my[v] = u;
59                 mx[u] = v;
60                 return true;
61             }
62         }
63     }
64     return false;
65 }
66 int MaxMatch() {
67     int res = 0;
68     memset(mx, -1, sizeof(mx));
69     memset(my, -1, sizeof(my));
70     while (SearchP()) {
71         memset(vis, false, sizeof(vis));
72         for (int i = 1; i <= un; ++i) { //②
73             if (mx[i] == -1 && dfs(i)) res++;
74         }
75     }

```



```

76     return res;
77 }
78 int main() {
79     int kase = 1;
80     cin >> tc;
81     while (tc--) {
82         cin >> t;
83         cin >> m;
84         un = m; ////!!给un赋值左端点的个数
85         for (int i = 0; i <= m; ++i) G[i].clear(); //多组数据，清空G
86         for (int i = 1; i <= m; ++i) cin >> human[i].x >> human[i].y >> human[i].
            speed;
87         cin >> n;
88         for (int i = 1; i <= n; ++i) cin >> umb[i].x >> umb[i].y;
89         for (int i = 1; i <= m; ++i) {
90             for (int j = 1; j <= n; ++j) {
91                 if (t * t * human[i].speed * human[i].speed >=
92                     (umb[j].x - human[i].x) * (umb[j].x - human[i].x) +
93                     (umb[j].y - human[i].y) * (umb[j].y - human[i].y)) {
94                     G[i].emplace_back(j); //加边
95                 }
96             }
97         }
98         cout << "Scenario #" << kase++ << ":" << '\n';
99         cout << MaxMatch() << '\n';
100        cout << '\n';
101    }
102    return 0;
103 }

```

1.3.3 KM-DFS

```

1  //hdu 2255 点和点之间都有边
2  //hdu 2426 点和点之间有可能没有边
3  //n <= m
4
5  const int maxn = 510;
6  const int INF = INT_MAX;
7  int n, m, link[maxn][maxn], num_a[maxn], num_b[maxn];
8  int match[maxn], slack[maxn], vis_a[maxn], vis_b[maxn];
9
10 bool dfs(int x) {
11     vis_a[x] = 1;
12     for (int i = 0; i < m; ++i) {
13         if (link[x][i] != -1) { //if写外面比里面写continue快? (hdu2426)
14             //两点之间不一定有边的情况要加这句
15             if (vis_b[i]) continue;
16             int diff = num_a[x] + num_b[i] - link[x][i];
17             if (!diff) {
18                 vis_b[i] = 1;
19                 if (match[i] == -1 || dfs(match[i])) {
20                     match[i] = x;

```

```

21         return true;
22     }
23     } else {
24         slack[i] = min(slack[i], diff);
25     }
26 }
27 }
28 return false;
29 }
30 int KM() {
31     ms(match, -1); ms(num_b, 0);
32     for (int i = 0; i < n; ++i) {
33         num_a[i] = INT_MIN;
34         for (int j = 0; j < m; ++j) {
35             num_a[i] = max(num_a[i], link[i][j]);
36         }
37     }
38     for (int i = 0; i < n; ++i) {
39         fill(slack, slack+m, INF);
40         int flag = 0;
41         for (int j = 0; j < m; ++j) if (link[i][j] != -1) flag = 1;
42         or
43         if (num_a[i] != -1) flag = 1;
44         //两点之间不一定有边的要加这句
45         //如果两点之间都有边的上面那句可以不写, flag=1, 这样快一点? (hdu2255)
46         while (flag) {
47             ms(vis_a, 0); ms(vis_b, 0);
48             if (dfs(i)) break;
49             int d = INF;
50             for (int j = 0; j < m; ++j) if (!vis_b[j]) d = min(d, slack[j]);
51             if (d == INF) break;
52             for (int j = 0; j < max(n, m); ++j) {
53                 if (j < n) {
54                     if (vis_a[j]) num_a[j] -= d;
55                 }
56                 if (j < m) {
57                     if (vis_b[j]) num_b[j] += d;
58                     else slack[j] -= d;
59                 }
60             }
61         }
62     }
63     int res = 0;
64     for (int i = 0; i < m; ++i) {
65         if (match[i] != -1) {
66             res += link[match[i]][i];
67         }
68     }
69     return res;
70 }
71
72 int main() {
73

```

```

74     int e;
75     while (cin >> n >> m >> e) {
76         ms(link, -1);
77         forn(i, e) {
78             int s, r, v;
79             cin >> s >> r >> v;
80             if (v >= 0) link[s][r] = v;
81         }
82         int ans = KM();
83         cout << ans << '\n';
84     }
85
86     return 0;
87 }

```

1.3.4 KM-BFS

```

1  /*KM by lbn O(n^3)
2  UOJ80 左nl 右nr 二分图最大权匹配
3  给定每两个点间权值，求一个匹配使权值和最大，不存在的边权开成 -1，时间复杂度 O(n^3)。
4  用法：两个点间权值 wi,j, lxi 和 lyi 为顶标，随时满足 lxi + lyi ≥ wi,j, lki 为右部第 i
    个点匹配的左部点。如果要求最小值全部取反即可。
5  #define INF 1e9 不存在的边权开到 -n*(|maxv|+1),inf为 3n*(|maxv|+1) (点数 最大边权)
6  lx左, ly右
7  点从1~n编号*/
8
9  #define ll long long
10
11 const int N = 410; //二分图某一边的最大点数
12 const int INF = 0x3f3f3f3f;
13 int lx[N],ly[N],w[N][N],lk[N],slk[N],pre[N],vy[N],
14     py,d,p;
15 int nl, nr, m, n, x, y, z;
16
17 int main() {
18
19     /*初始化别忘了*/
20     memset(lx, 0, sizeof(lx));
21     memset(ly, 0, sizeof(ly));
22     memset(lk, 0, sizeof(lk));
23     memset(w, 0, sizeof(w));
24     cin >> nl >> nr >> m;
25     /*二分图的大小*/
26     n = max(nl,nr);
27
28     /* 负权图初始化要加 hdu2813 */
29     for (int i = 1; i <= n; ++i) {
30         for (int j = 1; j <= n; ++j) w[i][j] = -INF;
31         ly[i] = -INF;
32     }
33     /* -----*/
34

```

```

35     for (int i = 1; i <= m; ++i)
36     {
37         cin >> x >> y >> z;
38         w[y][x]=z; //注意是[y][x]!! 负权的话z是负数! 相应的最后的ans也是负数!
39         lx[y]=max(lx[y],z); //负权可以不写?但我一直写的
40     }
41     /* -----KM----- */
42     for (int i = 1; i <= n; ++i)
43     {
44         for (int j = 1; j <= n; ++j) slk[j]=INF,vy[j]=0;
45         for(lk[py=0]=i;lk[py];py=p)
46         {
47             vy[py]=1; d=INF; x=lk[py];
48             for (int y = 1; y <= n; ++y) {
49                 if(!vy[y])
50                 {
51                     if(lx[x]+ly[y]-w[x][y]<slk[y]) slk[y]=lx[x]+ly[y]-w[x][y],pre[y]=py;
52                     if(slk[y]<d) d=slk[y],p=y;
53                 }
54             }
55             for (int y = 0; y <= n; ++y) {
56                 if(vy[y]) lx[lk[y]]-=d,ly[y]+=d;
57                 else slk[y]-=d;
58             }
59         }
60     }
61     for(;py;py=pre[py]) lk[py]=lk[pre[py]];
62 }
63 ll ans = 0;
64 正权 : for (int i = 1; i <= n; ++i) ans+=lx[i]+ly[i];
65 负权 : for (int i = 1; i <= n; ++i) if (w[lk[i]][i] != -INF) ans+=w[lk[i]][i];
66 printf("%lld\n",ans);
67 for (int i = 1; i <= nl; ++i) {
68     if(w[lk[i]][i]) printf("%d ",lk[i]);
69     else printf("0 ");
70 }
71 puts("");
72     //输出方案
73     /*-----*/
74     return 0;
75 }

```

1.3.5 带花树

```

1 //时间复杂度O(n^3)
2 #include <bits/stdc++.h>
3
4 using namespace std;
5
6 const int maxn = 1100;
7 int n, m, x, y;
8 vector<int> G[maxn];

```

```

9 namespace Blossom {
10     int mate[maxn], n, ret, nxt[maxn], f[maxn], mark[maxn], vis[maxn], t;
11     queue<int> q;
12     int F(int x) {return x == f[x] ? x : f[x] = F(f[x]);}
13     void Merge(int a, int b) {f[F(a)] = F(b);}
14     int lca(int x, int y) {
15         for (t++; swap(x, y)) {
16             if (~x) {
17                 if (vis[x = F(x)] == t) return x;
18                 vis[x] = t;
19                 x = (mate[x] != -1 ? nxt[mate[x]] : -1);
20             }
21         }
22     }
23     void group(int a, int p) {
24         for (int b, c; a != p; Merge(a, b), Merge(b, c), a = c) {
25             b = mate[a], c = nxt[b];
26             if (F(c) != p) nxt[c] = b;
27             if (mark[b] == 2) mark[b] = 1, q.push(b);
28             if (mark[c] == 2) mark[c] = 1, q.push(c);
29         }
30     }
31     void aug(int s, const vector<int> G[]) {
32         for (int i = 0; i < n; ++i) {
33             nxt[i] = vis[i] = -1;
34             f[i] = i;
35             mark[i] = 0;
36         }
37         while (!q.empty()) q.pop();
38         q.push(s);
39         mark[s] = 1;
40         while (mate[s] == -1 && !q.empty()) {
41             int x = q.front();
42             q.pop();
43             for (int i = 0, y; i < int(G[x].size()); ++i) {
44                 if ((y = G[x][i]) != mate[x] && F(x) != F(y) && mark[y] != 2) {
45                     if (mark[y] == 1) {
46                         int p = lca(x, y);
47                         if (F(x) != p) nxt[x] = y;
48                         if (F(y) != p) nxt[y] = x;
49                         group(x, p); group(y, p);
50                     } else if (mate[y] == -1) {
51                         nxt[y] = x;
52                         for (int j = y, k, l; ~j; j = l) {
53                             k = nxt[j];
54                             l = mate[k];
55                             mate[j] = k;
56                             mate[k] = j;
57                         }
58                         break;
59                     } else {
60                         nxt[y] = x;
61                         q.push(mate[y]);

```

```

62         mark[mate[y]] = 1;
63         mark[y] = 2;
64     }
65 }
66 }
67 }
68 }
69 int solve(int _n, vector<int> G[]) {
70     n = _n;
71     memset(mate, -1, sizeof(mate));
72     for (int i = t = 0; i < n; ++i) if (mate[i] == -1) aug(i, G);
73     for (int i = ret = 0; i < n; ++i) ret += (mate[i] > i);
74     printf("%d\n", ret);
75     for (int i = 0; i < n; ++i) printf("%d ", mate[i] + 1);
76     return ret;
77 }
78 }
79
80 int main() {
81     //for (int i = 0; i <= maxV; ++i) G[i].clear(); //多组数据的话别忘了从 0~最大点数
      清空一下vector
82     scanf("%d%d", &n, &m);
83     for (int i = 0; i < m; ++i) {
84         scanf("%d%d", &x, &y);
85         --x; --y; //顶点0~(n-1)编号
86         G[x].emplace_back(y);
87         G[y].emplace_back(x);
88     }
89     Blossom::solve(n, G);
90     return 0;
91 }

```

1.3.6 稳定婚姻问题

- 1 洛谷 p1407
- 2 题意：
- 3 给你所有人的关系，问每对情侣之间的是否稳定。
- 4 解法：
- 5 强连通分量。
- 6 先连正房之间的关系，女->男。
- 7 再连二奶之间的关系，男->女。
- 8 最后看每对情侣关系是否稳定，那就看这两人是不是在同一个强连通分量中。
- 9 在同一个强连通分量中，就说明不稳定。否则稳定。

1.3.7 常见思路

- 1 最小点覆盖 = 最大匹配 //选最少的点覆盖整个二分图 poj 3041 Asteroids
- 2 最小边覆盖 = 点数 - 最大匹配 //选最少的边覆盖整个二分图
- 3 最大独立集 = 点数 - 最大匹配
- 4
- 5 如果一图是二分图，那么他一定没有奇环。如果一图没有奇环，那么它可以是二分图。

```

6
7 二分图的判断：染色法
8 假设dfs初始点a涂黑色，并与他相邻的点就涂白色。
9 如果搜到某一个点u的相邻点v已经涂色并且与u同色，就不可能是二分图了。
10
11 匹配：给定一个二分图G，在G的一个子图M中，M的边集{E}中的任意两条边都不依附
12 于同一个顶点，则称M是一个匹配。
13 最大匹配：包含边数最多的匹配。
14 完美匹配（完备匹配）：所有点都在匹配边上的匹配。
15 最佳匹配：如果G为加权二分图，则权值和最大的完备匹配称为最佳匹配。
16
17 匈牙利算法  $O(V * E)$ 
18 HK算法  $O(\sqrt{V} * E)$ 

```

1.4 2-SAT

1.4.1 输出任意解

```

1 //洛谷 P4782
2 //从0开始的偶数为false，从1开始的奇数为true
3 //O(n+m)
4 const int N = 2*(int)EDIT+100; //点数*2
5 int scc, top, tot;
6 vector<int> G[N];
7 int low[N], dfn[N], belong[N];
8 int stk[N], vis[N];
9 void init(int n) {
10     for (int i = 0; i <= 2*n; ++i) {
11         G[i].clear();
12         low[i] = 0;
13         dfn[i] = 0;
14         stk[i] = 0;
15         vis[i] = 0;
16     }
17     scc = top = tot = 0;
18 }
19 void tarjan(int x) {
20     stk[top++] = x;
21     low[x] = dfn[x] = ++tot;
22     vis[x] = 1;
23     for (int to : G[x]) {
24         if (!dfn[to]) {
25             tarjan(to);
26             low[x] = min(low[x], low[to]);
27         } else if (vis[to]) low[x] = min(low[x], dfn[to]);
28     }
29     if (low[x] == dfn[x]) {
30         ++scc;
31         int temp;
32         do {
33             temp = stk[--top];
34             belong[temp] = scc;

```

```

35     vis[temp] = 0;
36     } while (temp != x);
37 }
38 }
39 void twoSat(int n) {
40     for (int i=0; i<2*n; ++i) {
41         if (!dfn[i]) tarjan(i);
42     }
43     for (int i=0; i<2*n; i+=2) {
44         if (belong[i] == belong[i^1]) {
45             cout<<"IMPOSSIBLE"<<'\n';
46             return;
47         }
48     }
49     cout<<"POSSIBLE"<<'\n';
50     for (int i = 0; i < 2*n; i+=2) { //因为强连通用了栈，所以强连通编号是反拓扑序
51         if (belong[i] > belong[i^1]) { //false->true 也就是只能为真
52             cout<<1<<' ';
53         } else cout<<0<<' ';
54     }
55     cout<<'\n';
56 }
57 void addEdge(int a, int b) {
58     G[a].emplace_back(b);
59 }
60 int n,m,a,ai,b,bi;
61 int main() {
62     cin>>n>>m;
63     init(n+5);
64     for(int i=0; i<m; ++i){
65         cin>>a>>ai>>b>>bi;
66         --a;--b;
67         a*=2;b*=2;
68         if(ai==0 && bi==0){
69             addEdge(a^1,b);
70             addEdge(b^1,a);
71         } else if(ai==0 && bi==1){
72             addEdge(a^1,b^1);
73             addEdge(b,a);
74         } else if(ai==1 && bi==0){
75             addEdge(a,b);
76             addEdge(b^1,a^1);
77         } else{
78             addEdge(a,b^1);
79             addEdge(b,a^1);
80         }
81     }
82     twoSat(n);
83     return 0;
84 }

```


1.4.2 输出字典序最小解

```

1 // hdu 1814
2 1<=n<=8000,0<=m<=20000,1<=a<b<=2n
3 // O(N*(N+M))
4 // 有1~2*n个人,第i个和第i+1个是同一对的,现在有m对不喜欢关系,现在要每个队选一个人出来,相互之间不会不喜欢,问最小字典序解。
5 // 从0开始的偶数为false,从1开始的奇数为true
6 const int N=2*EDIT+100;
7 vector< vector<int> > G(N);
8 int n,m;
9 bool vis[N]; //染色标记, true表示选择
10 int stk[N],top; //栈
11 void init(int n){
12     for(int i=0; i<=2*n; ++i){
13         vis[i]=false;
14         G[i].clear();
15     }
16 }
17 void addEdge(int u,int v){
18     G[u].emplace_back(v);
19 }
20 bool dfs(int now){
21     if(vis[now^1]) return false;
22     if(vis[now]) return true;
23     vis[now]=true;
24     stk[top++]=now;
25     for(int to:G[now]){
26         if(!dfs(to)) return false;
27     }
28     return true;
29 }
30 bool twoSat(int n){
31     // memset(vis,false,sizeof(vis));
32     for(int i=0; i<2*n; i+=2){
33         if(vis[i] || vis[i^1]) continue;
34         top=0;
35         if(!dfs(i)){
36             while(top){
37                 vis[stk[--top]]=false;
38             }
39             if(!dfs(i^1)) return false;
40         }
41     }
42     return true;
43 }
44 int main() {
45     while(cin>>n>>m){
46         init(n); //多组数据要清空
47         for(int i=0; i<m; ++i){
48             int u,v;
49             cin>>u>>v;
50             --u; --v;

```

```

51         addEdge(u,v^1);
52         addEdge(v,u^1);
53     }
54     if(twoSat(n)){
55         for(int i=0; i<2*n; ++i) if(vis[i]) cout<<i+1<<'\\n';
56     } else{
57         cout<<"NIE"<<'\\n';
58     }
59 }
60 return 0;
61 }

```

1.4.3 思路

- 1 我们发现每个点要么取0, 要么取1, 因此我们对ai建两个点i和i', 分别表示ai取1和ai取0。
- 2 然后我们考虑建边来表示这些关系, 我们令一条有向边的意义, $x \rightarrow y$ 表示如果选择x就必须选择y。
- 3 总结一下连边的规律: (用i'表示i的反面)
- 4 1. i, j必须同时选, 那么就有 $i \rightarrow j, j \rightarrow i$ 。
- 5 2. i, j不能同时选, 那么就有 $i \rightarrow j', j \rightarrow i'$ 。
- 6 3. i, j至少选一个, 那么就有 $i' \rightarrow j, j' \rightarrow i$ 。
- 7 4. 必须选i, 那么就有 $i' \rightarrow i$ 。

1.4.4 经典例题

```

1 UVA 11930
2 题意:
3 给你n个矩形, 每个矩形给你四个顶点(无序, 所以要先排序), 现在要求每个矩形都要选一个对角线, 使得
   这些对角线都不相交。如果存在解的话就YES, 不然NO。
4 1000ms n<=1000, -1e9<=xi,yi<=1e9
5 思路:
6 预处理出2*n条对角线, 然后遍历。
7 1. 同一个矩形的对角线continue,
8 2. i与j不相交, i与j^1相交 -> 选i一定选j -> i向j连边。
9 3. i与j相交, i与j^1不相交 -> 不可能选i, 只能选i^1 -> i向i^1连边
10 sgn和叉积那边要开ll(490ms) or 直接#define int long long(430ms 还快一点??)
11
12 cf 27D
13 问题:
14 给你一个环, 然后给你m条边, 表示要连这两条边, 现在只能在里面连或者在外面连两种方式, 问是否可
   行
15 关键要判环内两边是否一定规范相交(相交但不是端点相交)
16 环的编号是顺序的1~n, 输入时已经保证每个seg的l < r。
17 4<=n<=100, 1<=m<=100, 1<=ai,bi<=n, ai!=bi
18 思路:
19 下面的是判线段相交的代码
20 bool within(int x,int y,int z){
21     return (x<z && z<y);
22 }
23 bool intersect(int i1,int i2){
24     if(seg[i1].l==seg[i2].l || seg[i1].l==seg[i2].r) return false;
25     if(seg[i1].r==seg[i2].l || seg[i1].r==seg[i2].r) return false;

```

```

26     return (within(seg[i2].l, seg[i2].r, seg[i1].l) != within(seg[i2].l, seg[i2].r, seg[i1]
27     });

```

1.4.5 UVA 11930

```

1  UVA 11930
2  题意：
3  给你n个矩形，每个矩形给你四个顶点(无序,所以要先排序),现在要求每个矩形都要选一个对角线，使得
   这些对角线都不相交。如果存在解的话就YES,不然NO。
4  1000ms n<=1000, -1e9<=xi,yi<=1e9
5  思路：
6  预处理出2*n条对角线，然后遍历。
7  1. 同一个矩形的对角线continue,
8  2. i与j不相交,i与j^1相交 -> 选i一定选j -> i向j连边。
9  3. i与j相交,i与j^1不相交 -> 不可能选i,只能选i^1 -> i向i^1连边
10 sgn和叉积那边要开ll(490ms) or 直接#define int long long(430ms 还快一点??)
11
12 const int N=(int)1e5;
13 int scc, top, tot;
14 vector<int> G[N];
15 int low[N], dfn[N], belong[N];
16 int stk[N], vis[N];
17 void init(int n) {
18     for (int i = 0; i <= 2*n; ++i) {
19         G[i].clear();
20         low[i] = 0;
21         dfn[i] = 0;
22         stk[i] = 0;
23         vis[i] = 0;
24     }
25     scc = top = tot = 0;
26 }
27 void tarjan(int x) {
28     stk[top++] = x;
29     low[x] = dfn[x] = ++tot;
30     vis[x] = 1;
31     for (int to : G[x]) {
32         if (!dfn[to]) {
33             tarjan(to);
34             low[x] = min(low[x], low[to]);
35         } else if (vis[to]) low[x] = min(low[x], dfn[to]);
36     }
37     if (low[x] == dfn[x]) {
38         ++scc;
39         int temp;
40         do {
41             temp = stk[--top];
42             belong[temp] = scc;
43             vis[temp] = 0;
44         } while (temp != x);
45     }

```

```

46 }
47 void twoSat(int n) {
48     for (int i=0; i<2*n; ++i) {
49         if (!dfn[i]) tarjan(i);
50     }
51     for (int i=0; i<2*n; i+=2) {
52         if (belong[i] == belong[i^1]) {
53             cout<<"NO"<<"\n";
54             return;
55         }
56     }
57     cout<<"YES"<<"\n";
58 }
59 void addEdge(int a, int b) {
60     G[a].emplace_back(b);
61 }
62 int n,x11,y11,x22,y22,x33,y33,x44,y44;
63 int sgn(i64 x){
64     if(x==0LL) return 0;
65     if(x>0LL) return 1;
66     else return -1;
67 }
68 struct Point{
69     i64 x,y;
70     Point(){}
71     Point(i64 _x,i64 _y){
72         x=_x;
73         y=_y;
74     }
75     bool operator < (Point b) const{
76         return x==b.x?y<b.y:x<b.x;
77     }
78     Point operator - (const Point &b) const{
79         return Point(x-b.x,y-b.y);
80     }
81     i64 operator * (const Point &b) const{
82         return (x*b.x+y*b.y);
83     }
84     i64 operator ^ (const Point &b) const{
85         return (x*b.y-y*b.x);
86     }
87 };
88 struct Line{
89     Point s,e;
90     Line() {}
91     Line(Point _s,Point _e){
92         s=_s;
93         e=_e;
94     }
95     int segcrossseg(Line v){
96         int d1=sgn((e-s)^(v.s-s));
97         int d2=sgn((e-s)^(v.e-s));
98         int d3=sgn((v.e-v.s)^(s-v.s));

```

```

99     int d4=sgn((v.e-v.s)^(e-v.s));
100    if((d1^d2)==-2 && (d3^d4)==-2) return 2;
101    return (d1==0 && sgn((v.s-s)*(v.s-e))<=0) ||
102           (d2==0 && sgn((v.e-s)*(v.e-e))<=0) ||
103           (d3==0 && sgn((s-v.s)*(s-v.e))<=0) ||
104           (d4==0 && sgn((e-v.s)*(e-v.e))<=0);
105    }
106 }l[N];
107 signed main() {
108     while(cin>>n){
109         if(!n) break;
110         int tot=0;
111         forn(i, n){
112             vector<Point> temp;
113             temp.clear();
114             i64 a11,b11,a22,b22,a33,b33,a44,b44;
115             cin>>a11>>b11>>a22>>b22>>a33>>b33>>a44>>b44;
116             temp.eb(Point(a11,b11));
117             temp.eb(Point(a22,b22));
118             temp.eb(Point(a33,b33));
119             temp.eb(Point(a44,b44));
120             sort(all(temp));
121             x11=temp[0].x; y11=temp[0].y;
122             x22=temp[1].x; y22=temp[1].y;
123             x33=temp[3].x; y33=temp[3].y;
124             x44=temp[2].x; y44=temp[2].y;
125             l[tot++]=Line(Point(x11,y11),Point(x33,y33));
126             l[tot++]=Line(Point(x22,y22),Point(x44,y44));
127         }
128         init(2*n);
129         for(int i=0; i<tot; ++i){
130             for(int j=0; j<tot; ++j){
131                 if((i/2)==(j/2)) continue;
132                 if(l[i].segcrossseg(l[j])==0 && l[i].segcrossseg(l[j^1])!=0){
133                     addEdge(i,j);
134                 }
135                 if(j%2==0 && l[i].segcrossseg(l[j])!=0 && l[i].segcrossseg(l[j^1])!=0){
136                     addEdge(i,i^1);
137                 }
138             }
139         }
140         twoSat(n);
141     }
142     return 0;
143 }

```

1.4.6 cf27D

- 1 cf 27D
- 2 问题:
- 3 给你一个环, 然后给你m条边, 表示要连这两条边, 现在只能在里面连或者在外面连两种方式, 问是否可行

```

4  关键要判环内两边是否一定规范相交(相交但不是端点相交)
5  环的编号是顺序的1~n,输入时已经保证每个seg的l < r。
6  4<=n<=100,1<=m<=100,1<=ai,bi<=n,ai!=bi
7  思路:
8  下面的是判线段相交的代码
9  bool within(int x,int y,int z){
10     return (x<z && z<y);
11 }
12 bool intersect(int i1,int i2){
13     if(seg[i1].l==seg[i2].l || seg[i1].l==seg[i2].r) return false;
14     if(seg[i1].r==seg[i2].l || seg[i1].r==seg[i2].r) return false;
15     return (within(seg[i2].l,seg[i2].r,seg[i1].l)!=within(seg[i2].l,seg[i2].r,seg[i1].r));
16 }
17
18 const int N = 2*(int)1000+100; //点数*2
19 int n,m;
20 struct SE{
21     int l,r;
22     SE(){}
23     SE(int _l,int _r){
24         l=_l;
25         r=_r;
26     }
27 }seg[N];
28 int scc, top, tot;
29 vector<int> G[N];
30 int low[N], dfn[N], belong[N];
31 int stk[N], vis[N];
32 void init(int n) {
33     for (int i = 0; i <= 2*n; ++i) {
34         G[i].clear();
35         low[i] = 0;
36         dfn[i] = 0;
37         stk[i] = 0;
38         vis[i] = 0;
39     }
40     scc = top = tot = 0;
41 }
42 void tarjan(int x) {
43     stk[top++] = x;
44     low[x] = dfn[x] = ++tot;
45     vis[x] = 1;
46     for (int to : G[x]) {
47         if (!dfn[to]) {
48             tarjan(to);
49             low[x] = min(low[x], low[to]);
50         } else if (vis[to]) low[x] = min(low[x], dfn[to]);
51     }
52     if (low[x] == dfn[x]) {
53         ++scc;
54         int temp;
55         do {

```

```

56         temp = stk[--top];
57         belong[temp] = scc;
58         vis[temp] = 0;
59     } while (temp != x);
60 }
61 }
62 void twoSat(int n) {
63     for (int i=0; i<2*n; ++i) {
64         if (!dfn[i]) tarjan(i);
65     }
66     for (int i=0; i<2*n; i+=2) {
67         if (belong[i] == belong[i^1]) {
68             cout<<"Impossible"<<"\n";
69             return;
70         }
71     }
72     for (int i = 0; i < 2*n; i+=2) { //因为强连通用了栈，所以强连通编号是反拓扑序
73         if (belong[i] > belong[i^1]) { //false->true 也就是只能为真
74             cout<<"i";
75         } else cout<<"o";
76     }
77     cout<<"\n";
78 }
79 void addEdge(int a, int b) { //建双向边
80     G[a].emplace_back(b);
81     G[b].emplace_back(a);
82 }
83 bool within(int x,int y,int z){
84     return (x<z && z<y);
85 }
86 bool intersect(int i1,int i2){
87     if(seg[i1].l==seg[i2].l || seg[i1].l==seg[i2].r) return false;
88     if(seg[i1].r==seg[i2].l || seg[i1].r==seg[i2].r) return false;
89     return (within(seg[i2].l,seg[i2].r,seg[i1].l)!=within(seg[i2].l,seg[i2].r,seg[i1].r));
90 }
91 int main() {
92     cin>>n>>m;
93     forn(i, m){
94         int u,v;
95         cin>>u>>v;
96         if(u>v) swap(u,v);
97         seg[i]=SE(u,v);
98     }
99     init(1000);
100     for(int i=0; i<m; ++i){
101         for(int j=i+1; j<m; ++j){
102             if(intersect(i,j)){
103                 addEdge(i*2,(j*2)^1);
104                 addEdge(j*2,(i*2)^1);
105             }
106         }
107     }

```

```
108     twoSat(m);
109     return 0;
110 }
```

1.5 强连通

1.5.1 有向可有环图

```
1  // hdu 3836 求最少加几条边使图变成强连通图
2  // 时间复杂度  $O(V + E)$ 
3  // 有向可有环图->有向无环图(DAG)
4  const int N = EDIT+100; //点数
5  int scc, top, tot;
6  vector<int> G[N];
7  int low[N], dfn[N], belong[N];
8  int stk[N], vis[N];
9  void init(int n) {
10     for (int i = 0; i <= n; ++i) {
11         G[i].clear();
12         low[i] = 0;
13         dfn[i] = 0;
14         stk[i] = 0;
15         vis[i] = 0;
16     }
17     scc = top = tot = 0;
18 }
19 void tarjan(int x) {
20     stk[top++] = x;
21     low[x] = dfn[x] = ++tot;
22     vis[x] = 1;
23     for (int to : G[x]) {
24         if (!dfn[to]) {
25             tarjan(to);
26             low[x] = min(low[x], low[to]);
27         } else if (vis[to]) {
28             low[x] = min(low[x], dfn[to]);
29         }
30     }
31     if (low[x] == dfn[x]) {
32         ++scc;
33         int temp;
34         do {
35             temp = stk[--top];
36             belong[temp] = scc;
37             vis[temp] = 0;
38         } while (temp != x);
39     }
40 }
41
42 int n, m;
43 vi in, out;
44
```



```

45 int main() {
46     while (cin >> n >> m) {
47         init(n); //初始化别忘了
48         forn(i, m) {
49             int u, v;
50             cin >> u >> v;
51             G[u].eb(v); //建图
52         }
53         for (int i = 1; i <= n; ++i) if (!dfn[i]) tarjan(i); //tarjan求强连通
54         /* solving */
55         if (scc == 1) {
56             cout << 0 << '\n';
57             continue;
58         }
59         in = vi(scc+1, 0);
60         out = vi(scc+1, 0);
61         for (int i = 1; i <= n; ++i) {
62             for (int j : G[i]) {
63                 if (belong[i] == belong[j]) continue; //缩点，同一个强连通分量之间不用连边
64                 ++out[belong[i]];
65                 ++in[belong[j]];
66             }
67         }
68         int in0 = 0, out0 = 0;
69         for (int i = 1; i <= scc; ++i) {
70             if (!in[i]) ++in0;
71             if (!out[i]) ++out0;
72         }
73         cout << max(in0, out0) << '\n';
74         /* end of solving */
75     }
76     return 0;
77 }

```

1.6 双连通

1.6.1 割点桥

```

1 //时间复杂度O(V+E)
2 //add_block[i]:割掉i以后会多产生的连通块数，即割掉后图中的连通块数为add_block[i]+1
3 const int N = edit+100; //点数
4 vector<int> E[N];
5 struct BCC {
6     int n, bcc, top, tot;
7     vector<int> G[N];
8     vector< pair<int, int> > bridge;
9     int low[N], dfn[N], belong[N], fa[N];
10    int stk[N];
11    int cut[N], add_block[N];
12
13    void dfs(int x, int pre) {
14        stk[top++] = x;

```

```

15     low[x] = dfn[x] = ++tot;
16     fa[x] = pre;
17     int son = 0;
18     for (int to : G[x]) {
19         if (to == pre) continue;
20         if (!dfn[to]) {
21             ++son;
22             dfs(to, x);
23             low[x] = min(low[x], low[to]);
24             if (x != pre && low[to] >= dfn[x]) {
25                 cut[x] = 1;
26                 add_block[x]++;
27             }
28             if (low[to] > dfn[x]) bridge.push_back(make_pair(x, to));
29         }
30         else if (dfn[to] < dfn[x]) low[x] = min(low[x], dfn[to]);
31     }
32     if (x == pre && son > 1) {
33         cut[x] = 1;
34         add_block[x] = son-1;
35     }
36     if (low[x] == dfn[x]) {
37         ++bcc;
38         int temp;
39         do {
40             temp = stk[--top];
41             belong[temp] = bcc;
42         } while (temp != x);
43     }
44 }
45 void solve(int _n, vector<int> E[]) {
46     n = _n;
47     for (int i = 1; i <= n; ++i) {
48         belong[i]=0;
49         G[i] = E[i];
50         low[i] = dfn[i] = stk[i] = fa[i] = 0;
51         cut[i] = add_block[i] = 0;
52     }
53     bcc = top = tot = 0;
54     bridge.clear();
55     for (int i = 1; i <= n; ++i) if (!dfn[i]) dfs(i, i);
56 }
57 void rebuild(vector<int> E[]) {
58     for (int i = 1; i <= n; ++i) E[i].clear();
59     for (int i = 1; i <= n; ++i) {
60         int t = fa[i];
61         if (belong[i] != belong[t]) {
62             E[belong[i]].push_back(belong[t]);
63             E[belong[t]].push_back(belong[i]);
64         }
65     }
66 }
67 }bcc;

```

```

68 int n, m;
69
70 int main() {
71     cin >> n >> m;
72     for(int i=0; i<=n; ++i) E[i].clear(); //初始化
73     forn(i, m) {
74         int x, y;
75         cin >> x >> y; //顶点编号从1~n
76         E[x].eb(y);
77         E[y].eb(x);
78     }
79     bcc.solve(n, E);
80     vi res;
81     for1(i, n) if (bcc.cut[i]) res.eb(i);
82     cout << SZ(res) << '\n';
83     for (int x : res) cout << x << ' ';
84     cout << '\n';
85     return 0;
86 }

```

1.6.2 船新版本

```

1  https://ac.nowcoder.com/acm/contest/7501/D
2  牛客小米icpc邀请赛第一场 D
3  upd:完善了上一个版本求add_block不适用于一开始不连通的图的情况。
4  //时间复杂度 $O(V+E)$ 
5  //建边建双向边
6  //顶点编号1~n
7  //add_block[i]表示割掉i后图中的连通块数
8
9  const int N = EDIT+100; //点数
10 vector<int> E[N];
11 struct BCC {
12     int n, bcc, top, tot, sum;
13     vector<int> G[N];
14     vector< pair<int, int> > bridge;
15     int low[N], dfn[N], belong[N], fa[N];
16     int stk[N];
17     int cut[N], add_block[N];
18
19     void dfs(int x, int pre) {
20         stk[top++] = x;
21         low[x] = dfn[x] = ++tot;
22         fa[x] = pre;
23         int son = 0;
24         for (int to : G[x]) {
25             if (to == pre) continue;
26             if (!dfn[to]) {
27                 ++son;
28                 dfs(to, x);
29                 low[x] = min(low[x], low[to]);
30                 if (x != pre && low[to] >= dfn[x]) cut[x] = 1;

```

```

31         if(low[to]>=dfn[x]) ++add_block[x];
32         if (low[to]>dfn[x]) bridge.push_back(make_pair(x, to));
33     }
34     else if(dfn[to] < dfn[x]) low[x] = min(low[x], dfn[to]);
35 }
36 if (x == pre && son > 1) cut[x] = 1;
37 if (low[x] == dfn[x]) {
38     ++bcc;
39     int temp;
40     do {
41         temp = stk[--top];
42         belong[temp] = bcc;
43     } while (temp != x);
44 }
45 if(x!=pre) ++add_block[x];
46 }
47 void solve(int _n, vector<int> E[]) {
48     n = _n;
49     for (int i = 0; i <= n; ++i) { //0~n-1
50         G[i] = E[i];
51         low[i] = dfn[i] = stk[i] = fa[i] = 0;
52         cut[i] = add_block[i] = 0;
53     }
54     bcc = top = tot = sum = 0;
55     bridge.clear();
56     for (int i = 1; i <= n; ++i){
57         if (!dfn[i]){
58             dfs(i, i); //0~n-1
59             ++sum;
60         }
61     }
62     for (int i=1; i<=n; ++i) add_block[i]+=sum-1;
63     // for(int i=1; i<=n; ++i) cout<<add_block[i]<<' ';
64     // cout<<'\n';
65 }
66 void rebuild(vector<int> E[]) {
67     for (int i = 1; i <= n; ++i) E[i].clear();
68     for (int i = 1; i <= n; ++i) {
69         int t = fa[i];
70         if (belong[i] != belong[t]) {
71             E[belong[i]].push_back(belong[t]);
72             E[belong[t]].push_back(belong[i]);
73         }
74     }
75 }
76 }bcc;

```

1.6.3 思路

- 1 1.有割点不一定有桥，有桥一定存在割点
- 2 2.桥一定是割点依附的边。
- 3

4 边双连通：若一个无向图中去掉任意一条边都不会改变此图的
 5 联通性，即不存在桥，则称作边双连通图。
 6
 7 边双连通分量：无向图中，删除任意边仍然能联通的块
 8 那么再此分量中的边一定不是桥
 9 同时，不在此分量中的边一定是桥
 10
 11 所以用这个可以用来判桥
 12 对于每个连通分量，删除任意一边连通性不变，其中可能含有割点，
 13 且其中环与环不保证有公共边，但至少有一个公共点。
 14
 15 的连通性，即不存在割点，则称作点双连通图。
 16 点双连通，若一个无向图中去掉任意一个节点都不会改变此图

1.6.4 经典例题

1 洛谷 T103492
 2 题意：
 3 给你一个 n 个点， m 条边的无向图，让你输出共 $pbccCnt$ 行， $pbccCnt$ 为点双连通分量数量。对于第 i 行，
 4 输出第 i 个点双连通分量的每个点。（顺序不分前后）
 5 $n \leq 5e4$, $m \leq 3e5$
 6 思路：
 7 无，直接看代码。
 8 hdu 3836
 9 题意：
 10 最少加几条边使得整张图 scc 。
 11 思路：
 12 先求出强连通分量对 scc 重新建图，预处理出新图每个点的入度和出度，统计出新图中出度为 0 和入度为 0
 13 的点的个数分别为 $out0$ 和 $in0$ ，答案就是 $\max(out0, in0)$ 。
 14 poj 2186
 15 题意：
 16 有 m 对奶牛之间的喜欢关系，喜欢关系有传递性，现在问有多少牛被所有的牛喜欢？（或有哪些牛）
 17 思路：
 18 先求出强连通分量对 scc 重新建图，预处理出新图每个点的出度，设出度为 0 的点数为 cnt ，如果 $cnt > 1$ ，就
 19 无解，否则 cnt 就是 1 ，有解，遍历每个点属于的 scc 如果这个 scc 在新图中出度为 0 ，就 $+1$ 。（把这个
 20 点加入答案）
 21 poj 2553
 22 题意：
 23 给你一个图，现在定义一个点 u 是牛逼的，如果 u 可以到达的所有点也都可以到达 u ，问有哪些点是牛逼的。
 24 思路：
 25 先求出强连通分量对 scc 重新建图，预处理出新图的每个点的出度，出度为 0 的 scc 里面的点就是牛逼的。
 26 poj 3352（一开始就连通）
 27 题意：
 28 最少加几条边使整张图 bcc 。
 29 思路：
 30 前提：当前图已经连通。
 31 先求出 bcc 然后对 bcc 重新建图，预处理出新图每个点的度数，设度数为 1 的点数为 cnt ，最少需要加（
 $cnt + 1$ ）/ 2 条边就能使整条 bcc 。

```

32
33 UVA 10972 (一开始不连通)
34 题意:
35 一开始给你一个无向图, 现在要让这个无向图变成有向图, 现在问你最少加几条边才能让有向图强连通?
36 思路:
37 其实就是要让原无向图边双连通, 先边双连通重新建图, 因为有向图强连通, 所以无向图中每个点的度至少为2, 设ans=0, 新图中点度为1就ans++, 点度为2就ans+=2, 最后答案是(ans+1)/2。
38
39 poj 2942 边双连通+染色法判二分图
40 题意:
41 一些骑士, 他们有些人之间有矛盾, 现在要求选出一些骑士围成一圈。
42 圈要满足如下条件:
43 1. 人数大于1。
44 2. 总人数为奇数。
45 3. 有仇恨的骑士不能挨着坐。
46 问有几个骑士不能和任何人形成任何的圆圈?
47 思路:
48 首先反向建立补图, 然后问题转换成在图中找奇圈, 圈肯定出现在双联通分量中, 则求出图的双联通分量, 又通过特性知道, 一个双联通分量有奇圈则其中的点都可以出现在一个奇圈中。而对于奇圈的判定可以用交叉染色判断是非为二分图, 二分图中肯定无奇圈。
49
50 p1407 稳定婚姻问题
51 题意:
52 先输入一个n, 下面n对正房之间的关系(先女后男), 再输一个m, 表示m对二奶(保证二奶都在正房里出现过)之间的关系(先女后男)。如果一对婚姻破裂后, 通过婚外情能再次组成n对婚姻, 这对婚姻就是不稳定的。求出每对婚姻稳定与否。
53 n<=4000, m<=20000。
54 思路:
55 强连通分量。
56 先连正房之间的关系, 女->男。
57 再连二奶之间的关系, 男->女。
58 最后看每对情侣关系是否稳定, 那就看这两人是不是在同一个强连通分量中。
59 在同一个强连通分量中, 就说明不稳定。否则稳定。

```

1.6.5 POJ 2942

```

1  #include <stdio.h>
2  #include <vector>
3  #include <algorithm>
4  #include <string.h>
5  #include <limits.h>
6  #include <string>
7  #include <iostream>
8  #include <queue>
9  #include <math.h>
10 #include <map>
11 #include <stack>
12 #include <sstream>
13 #include <set>
14 #include <iterator>
15 #include <list>
16 #include <cstdio>

```

```

17 #include <iomanip>
18 #include <climits>
19 #define mp make_pair
20 #define fi first
21 #define se second
22 #define pb push_back
23 #define eb emplace_back
24 #define all(x) (x).begin(), (x).end()
25 #define rall(x) (x).rbegin(), (x).rend()
26 #define forn(i, n) for (int i = 0; i < (int)(n); ++i)
27 #define for1(i, n) for (int i = 1; i <= (int)(n); ++i)
28 #define ford(i, a, b) for (int i = (int)(a); i >= (int)b; --i)
29 #define fore(i, a, b) for (int i = (int)(a); i <= (int)(b); ++i)
30 #define rep(i, l, r) for (int i = (l); i <= (r); i++)
31 #define per(i, r, l) for (int i = (r); i >= (l); i--)
32 #define ms(x, y) memset(x, y, sizeof(x))
33 #define SZ(x) int(x.size())
34 using namespace std;
35 typedef pair<int, int> pii;
36 typedef vector<int> vi;
37 typedef vector<pii> vpi;
38 typedef vector<vi> vvi;
39 typedef long long i64;
40 typedef vector<i64> vi64;
41 typedef vector<vi64> vvi64;
42 typedef pair<i64, i64> pi64;
43 typedef double ld;
44 template<class T> bool uin(T &a, T b) { return a > b ? (a = b, true) : false; }
45 template<class T> bool uax(T &a, T b) { return a < b ? (a = b, true) : false; }
46 //1.integer overflow (1e5 * 1e5) (2e9 + 2e9)
47 //2.runtime error
48 //3.boundary condition
49 const int N = 1000+100; //点数
50 int n,m,mep[N][N];
51 vector<int> E[N];
52 struct BCC {
53     int n, bcc, top, tot;
54     vector<int> G[N];
55     vector< pair<int, int> > bridge;
56     int low[N], dfn[N], belong[N], fa[N];
57     int stk[N];
58     int cut[N], add_block[N];
59     int col[N], vis[N];
60
61     bool findOddCircle(int now,int c){ //找奇环
62         col[now]=c;
63         for(int i=0; i<int(G[now].size()); ++i){
64             int to=G[now][i];
65             if(belong[now]!=belong[to]) continue;
66             if(col[to]==-1){
67                 if(findOddCircle(to,c^1)) return true;
68             }
69             if(col[to]==col[now]) return true;

```

```

70     }
71     return false;
72 }
73 void dfs(int x, int pre) {
74     stk[top++] = x;
75     low[x] = dfn[x] = ++tot;
76     fa[x] = pre;
77     int son = 0;
78     for (int i=0; i<int(G[x].size()); ++i) {
79         int to=int(G[x][i]);
80         if (to == pre) continue;
81         if (!dfn[to]) {
82             ++son;
83             dfs(to, x);
84             low[x] = min(low[x], low[to]);
85             if (low[to] >= dfn[x]) {
86                 cut[x] = 1;
87                 add_block[x]++;
88                 ++bcc;
89                 int temp;
90                 vector<int> p; p.clear();
91                 do {
92                     temp = stk[--top];
93                     p.pb(temp);
94                     belong[ temp] = bcc;
95                 } while (temp != to); //魔改bcc, 得出所有的bcc(包括大bcc包含的小的bcc)
96                 belong[x]=bcc;
97                 p.pb(x);
98                 if(int(p.size())>=3){
99                     ms(col,-1);
100                     if(findOddCircle(temp,0)) for(int j=0; j<int(p.size()); ++j) vis
                        [p[j]]=1;
101                 }
102             }
103             if (low[to] > dfn[x]) bridge.push_back(make_pair(x, to));
104         }
105         else low[x] = min(low[x], dfn[to]);
106     }
107     if (x == pre && son > 1) {
108         cut[x] = 1;
109         add_block[x] = son-1;
110     }
111 }
112 void solve(int _n, vector<int> E[]) {
113     n = _n;
114     for (int i = 1; i <= n; ++i) {
115         belong[i]=0;
116         G[i] = E[i];
117         low[i] = dfn[i] = stk[i] = fa[i] = 0;
118         cut[i] = add_block[i] = 0;
119         vis[i]=0;
120     }
121     bcc = top = tot = 0;

```



```

122     bridge.clear();
123     for (int i = 1; i <= n; ++i) if (!dfn[i]) dfs(i, i);
124 }
125 void rebuild(vector<int> E[]) {
126     for (int i = 1; i <= n; ++i) E[i].clear();
127     for (int i = 1; i <= n; ++i) {
128         int t = fa[i];
129         if (belong[i] != belong[t]) {
130             E[belong[i]].push_back(belong[t]);
131             E[belong[t]].push_back(belong[i]);
132         }
133     }
134 }
135 }bcc;
136 signed main() {
137     ios::sync_with_stdio(false);
138     cin.tie(0);
139     cout.precision(10);
140     cout << fixed;
141 #ifdef LOCAL_DEFINE
142     freopen("input.txt", "r", stdin);
143 #endif
144     while(~scanf("%d%d",&n,&m)){
145         if(!n && !m) break;
146         for(int i=0; i<=n; ++i){
147             E[i].clear();
148             for(int j=0; j<=n; ++j) mep[i][j]=0;
149         }
150         forn(i, m){
151             int u,v;
152             scanf("%d%d",&u,&v);
153             mep[u][v]=mep[v][u]=1;
154         }
155         for(int i=1; i<=n; ++i){
156             for(int j=1; j<=n; ++j){
157                 if(i==j) continue; //别tm建自环
158                 if(!mep[i][j]){
159                     E[i].pb(j);
160                 }
161             }
162         }
163         bcc.solve(n,E);
164         int sum=n;
165         for(int i=1; i<=n; ++i) if(bcc.vis[i]) --sum;
166         printf("%d\n",sum);
167     }
168 #ifdef LOCAL_DEFINE
169     cerr << "Time elapsed: " << 1.0 * clock() / CLOCKS_PER_SEC << " s.\n";
170 #endif
171     return 0;
172 }

```

1.6.6 UVA 10972

```

1  const int N = 1000+100; //点数
2  int n,m;
3  vector<int> E[N];
4  struct BCC {
5      int n, bcc, top, tot;
6      vector<int> G[N];
7      vector< pair<int, int> > bridge;
8      int low[N], dfn[N], belong[N], fa[N];
9      int stk[N];
10     int cut[N], add_block[N];
11     int deg[N];
12
13     void dfs(int x, int pre) {
14         stk[top++] = x;
15         low[x] = dfn[x] = ++tot;
16         fa[x] = pre;
17         int son = 0;
18         for (int to : G[x]) {
19             if (to == pre) continue;
20             if (!dfn[to]) {
21                 ++son;
22                 dfs(to, x);
23                 low[x] = min(low[x], low[to]);
24                 if (x != pre && low[to] >= dfn[x]) {
25                     cut[x] = 1;
26                     add_block[x]++;
27                 }
28                 if (low[to] > dfn[x]) bridge.push_back(make_pair(x, to));
29             }
30             else if(dfn[to] < dfn[x]) low[x] = min(low[x], dfn[to]);
31         }
32         if (x == pre && son > 1) {
33             cut[x] = 1;
34             add_block[x] = son-1;
35         }
36         if (low[x] == dfn[x]) {
37             ++bcc;
38             int temp;
39             do {
40                 temp = stk[--top];
41                 belong[temp] = bcc;
42             } while (temp != x);
43         }
44     }
45     void solve(int _n, vector<int> E[]) {
46         n = _n;
47         for (int i = 1; i <= n; ++i) {
48             belong[i]=0;
49             G[i] = E[i];
50             low[i] = dfn[i] = stk[i] = fa[i] = 0;
51             cut[i] = add_block[i] = 0;

```

```

52         deg[i]=0;
53     }
54     bcc = top = tot = 0;
55     bridge.clear();
56     for (int i = 1; i <= n; ++i) if (!dfn[i]) dfs(i, i);
57 }
58 void rebuild(vector<int> E[]) {
59     for (int i = 1; i <= n; ++i) E[i].clear();
60     for (int i = 1; i <= n; ++i) {
61         int t = fa[i];
62         if (belong[i] != belong[t]) {
63             ++deg[belong[i]]; ++deg[belong[t]];
64             E[belong[i]].push_back(belong[t]);
65             E[belong[t]].push_back(belong[i]);
66         }
67     }
68 }
69 int gao(){
70     if(bcc==1) return 0;
71     int ans=0;
72     for(int i=1; i<=bcc; ++i){
73         if(deg[i]<2) ans+=2-deg[i];
74     }
75     return (ans+1)/2;
76 }
77 }bcc;
78 signed main() {
79     while(cin>>n>>m){
80         for(int i=0; i<=n; ++i) E[i].clear();
81         for(int i=0; i<m; ++i){
82             int u,v;
83             cin>>u>>v;
84             E[u].eb(v);
85             E[v].eb(u);
86         }
87         bcc.solve(n,E);
88         bcc.rebuild(E);
89         cout<<bcc.gao()<<'\\n';
90     }
91     return 0;
92 }

```

1.6.7 T103492

- 1 题意:
- 2 给你一个 n 个点, m 条边的无向图, 让你输出共 $pbccCnt$ 行, $pbccCnt$ 为点双连通分量数量。对于第 i 行,
输出第 i 个点双连通分量的每个点。(顺序不分前后)
- 3 $n \leq 5e4, m \leq 3e5$
- 4 思路:
- 5 无, 直接看代码。
- 6
- 7 `const int N = 5*(int)1e4+100; //点数`

```

8  int n,m;
9  vector<int> E[N];
10 struct BCC {
11     int n, bcc, pbcc, top, ptop, tot;
12     vector<int> G[N];
13     vector<int> ans[N];
14     vector< pair<int, int> > bridge;
15     int low[N], dfn[N], belong[N], fa[N], pbelong[N];
16     int stk[N], pstk[N];
17     int cut[N], add_block[N];
18
19     void dfs(int x, int pre) {
20         stk[top++] = x;
21         pstk[ptop++] = x;
22         low[x] = dfn[x] = ++tot;
23         fa[x] = pre;
24         int son = 0;
25         if(SZ(E[x])==0){
26             ans[++pbcc].eb(x);
27             //pbelong[x]=pbcc;
28             return;
29         }
30         for (int to : G[x]) {
31             if (to == pre) continue;
32             if (!dfn[to]) {
33                 ++son;
34                 dfs(to, x);
35                 low[x] = min(low[x], low[to]);
36                 if (x != pre && low[to] >= dfn[x]) {
37                     cut[x] = 1;
38                     add_block[x]++;
39                 }
40                 if(low[to]>=dfn[x]){
41                     ++pbcc;
42                     int temp;
43                     do{
44                         temp=psk[--ptop];
45                         ans[pbcc].eb(temp);
46                         // pbelong[temp]=pbcc;
47                     } while(temp!=to);
48                     ans[pbcc].eb(x);
49                 }
50                 if (low[to] > dfn[x]) bridge.push_back(make_pair(x, to));
51             }
52             else if(dfn[to] < dfn[x]) low[x] = min(low[x], dfn[to]);
53         }
54         if (x == pre && son > 1) {
55             cut[x] = 1;
56             add_block[x] = son-1;
57         }
58         if (low[x] == dfn[x]) {
59             ++bcc;
60             int temp;

```

```

61         do {
62             temp = stk[--top];
63             belong[temp] = bcc;
64         } while (temp != x);
65     }
66 }
67 void solve(int _n, vector<int> E[]) {
68     n = _n;
69     for (int i = 1; i <= n; ++i) {
70         ans[i].clear();
71         belong[i]=pbelong[i]=0;
72         G[i] = E[i];
73         low[i] = dfn[i] = stk[i] = pstk[i] = fa[i] = 0;
74         cut[i] = add_block[i] = 0;
75     }
76     bcc = pbcc = top = ptop = tot = 0;
77     bridge.clear();
78     for (int i = 1; i <= n; ++i) if (!dfn[i]) dfs(i, i);
79     for1(i, pbcc){
80         for(int x:ans[i]) cout<<x<<' ';
81         cout<<'\n';
82     }
83 }
84 void rebuild(vector<int> E[]) {
85     for (int i = 1; i <= n; ++i) E[i].clear();
86     for (int i = 1; i <= n; ++i) {
87         int t = fa[i];
88         if (belong[i] != belong[t]) {
89             E[belong[i]].push_back(belong[t]);
90             E[belong[t]].push_back(belong[i]);
91         }
92     }
93 }
94 }bcc;
95 int main() {
96     cin>>n>>m;
97     forn(i, m){
98         int a,b;
99         cin>>a>>b;
100         E[a].eb(b);
101         E[b].eb(a);
102     }
103     //for(int i=1; i<=n; ++i) reverse(all(E[i])); 傻逼题没加spj
104     bcc.solve(n,E);
105     return 0;
106 }

```

1.6.8 P1407

- 1 题意:
- 2 先输入一个n, 下面n对正房之间的关系(先女后男), 再输一个m, 表示m对二奶(保证二奶都在正房里出现过)之间的关系(先女后男)。如果一对婚姻破裂后, 通过婚外情能再次组成n对婚姻, 这对婚姻就是

不稳定的。求出每对婚姻稳定与否。

```
3  n<=4000, m<=20000.
4  思路:
5  强连通分量。
6  先连正房之间的关系, 女->男。
7  再连二奶之间的关系, 男->女。
8  最后看每对情侣关系是否稳定, 那就看这两人是不是在同一个强连通分量中。
9  在同一个强连通分量中, 就说明不稳定。否则稳定。
10
11  const int N = 8000+100; //点数
12  //有n对, 每对有两个, 所以要乘2
13  int n,m,idx=0;
14  map<string,int> hs;
15  int scc, top, tot;
16  vector<int> G[N];
17  int low[N], dfn[N], belong[N];
18  int stk[N], vis[N];
19  void init(int n) {
20      for (int i = 0; i <= 2*n; ++i) {
21          G[i].clear();
22          low[i] = 0;
23          dfn[i] = 0;
24          stk[i] = 0;
25          vis[i] = 0;
26      }
27      scc = top = tot = 0;
28  }
29  void tarjan(int x) {
30      stk[top++] = x;
31      low[x] = dfn[x] = ++tot;
32      vis[x] = 1;
33      for (int to : G[x]) {
34          if (!dfn[to]) {
35              tarjan(to);
36              low[x] = min(low[x], low[to]);
37          } else if (vis[to]) {
38              low[x] = min(low[x], dfn[to]);
39          }
40      }
41      if (low[x] == dfn[x]) {
42          ++scc;
43          int temp;
44          do {
45              temp = stk[--top];
46              belong[temp] = scc;
47              vis[temp] = 0;
48          } while (temp != x);
49      }
50  }
51
52  int main() {
53      cin>>n;
54      init(2*n+5); //有n对, 每对有两个, 所以要乘2
```

```

55     string s1,s2;
56     for(int i=0; i<n; ++i){
57         cin>>s1>>s2;
58         if(!hs[s1]) hs[s1]=idx++;
59         if(!hs[s2]) hs[s2]=idx++;
60         G[hs[s1]].eb(hs[s2]);
61     }
62     cin>>m;
63     forn(i,m){
64         cin>>s1>>s2;
65         G[hs[s2]].eb(hs[s1]);
66     }
67     for(int i=0; i<idx; ++i) if(!dfn[i]) tarjan(i);
68     bool ok=true;
69     for(int i=0; i<idx; i+=2){
70         if(belong[i]!=belong[i+1]){
71             cout<<"Safe"<<"\n";
72         } else{
73             cout<<"Unsafe"<<"\n";
74         }
75     }
76     return 0;
77 }

```

1.7 欧拉回路

1.7.1 模板

```

1  //下面O(n+m)求欧拉回路的代码中，n为点数，m为边数。
2  //若有解则一次输出经过的边的编号。
3  //若是无向图，则正数表示x到y，负数表示y到x。
4  const int N=点数;
5  const int M=边数;
6  namespace UndirectedGraph{
7      int n,m,i,x[M],y[M],d[N],g[N],v[M<<1],w[M<<1],vis[M<<1],nxt[M<<1],ed;
8      int ans[M],cnt;
9      void addEdge(int x,int y,int z){
10         d[x]++;
11         v[++ed]=y; w[ed]=z; nxt[ed]=g[x]; g[x]=ed;
12     }
13     void dfs(int x){
14         for(int &i=g[x];i;){
15             if(vis[i]){i=nxt[i];continue;}
16             vis[i]=vis[i^1]=1;
17             int j=w[i];
18             dfs(v[i]);
19             ans[++cnt]=j;
20         }
21     }
22     void solve(){
23         scanf("%d%d",&n,&m);
24         ed=1;

```

```

25     for(int i=0; i<=n; ++i) cnt=d[i]=g[i]=0;
26     for(int i=m+1; i<=2*m+10; ++i) vis[i]=0;
27     for(int i=0; i<=m; ++i) vis[i]=ans[i]=0;
28     for(i=1; i<=m; ++i) scanf("%d%d",&x[i],&y[i]),addEdge(x[i],y[i],i),addEdge(y[
        i],x[i],-i);
29     for(i=1; i<=n; ++i) if(d[i]&1) {puts("NO");return;}
30     for(i=1; i<=n; ++i) if(g[i]) {dfs(i);break;}
31     for(i=1; i<=n; ++i) if(g[i]) {puts("NO");return;}
32     puts("YES");
33     for(i=m; i; i--) printf("%d ",ans[i]);
34 }
35 }
36 namespace DirectedGraph{
37     int n,m,i,x,y,d[N],g[N],v[M],vis[M],nxt[M],ed;
38     int ans[M],cnt;
39     void addEdge(int x,int y){
40         d[x]++; d[y]--;
41         v[++ed]=y; nxt[ed]=g[x]; g[x]=ed;
42     }
43     void dfs(int x){
44         for(int &i=g[x];i;){
45             if(vis[i]){i=nxt[i];continue;}
46             vis[i]=1;
47             int j=i;
48             dfs(v[i]);
49             ans[++cnt]=j;
50         }
51     }
52     void solve(){
53         scanf("%d%d",&n,&m);
54         ed=0;
55         for(int i=0; i<=n; ++i) cnt=d[i]=g[i]=0;
56         for(int i=0; i<=m+10; ++i) vis[i]=ans[i]=0;
57         for(i=1; i<=m; ++i) scanf("%d%d",&x,&y),addEdge(x,y);
58         for(i=1; i<=n; ++i) if(d[i]) {puts("NO");return;}
59         for(i=1; i<=n; ++i) if(g[i]) {dfs(i);break;}
60         for(i=1; i<=n; ++i) if(g[i]) {puts("NO");return;}
61         puts("YES");
62         for(i=m; i; i--) printf("%d ",ans[i]);
63     }
64 }

```

1.7.2 知识点

- 1 欧拉回路:
- 2 无向图: 每个顶点的度数都是偶数, 则存在欧拉回路。
- 3 有向图: 每个顶点的入度=出度, 则存在欧拉回路。
- 4
- 5 欧拉路径:
- 6 无向图: 当且仅当该图的度数为偶数或者除了两个度数为奇数外其余的全是偶数。
- 7 有向图: 当且仅当改图的所有出度=入度 或者 一个顶点出度=入度+1, 另一个顶点入度=出度+1, 其他顶点出度=入度。

1.7.3 经典例题

```

1 cf 21D
2 题意：
3 给你n个点m条边的无向图，求从点1开始经过每条边至少一次最后回到点1的最小路程。
4 就是找** 一条路径可重复的欧拉回路 **。
5 input:
6 cin>>n>>m;
7 for(int i=0; i<m; ++i) cin>>x>>y>>z;
8 n是点数(这个图点的编号1~n)，m是边数，x,y,z表示x和y之间有一条长度为z的无向边
9 1<=n<=15,0<=m<=2000,1<=x,y<=n,1<=w<=10000
10 思路：
11 首先对于欧拉回路：所有点的度数都为偶数。因为所有点至少经过一次，那么可以把题意转换成最少加多
    少条边使得图满足以上结论。
12 而加边的目的是为了把奇度数转化为偶度数，先floyd一下得到全源最短路。dp[i]表示状态i下度数为偶
    数的最小花费，因为n<=15,想到状压dp,挑两个奇度数的点转移即可。详情可见代码。

```

1.7.4 cf 21D

```

1 cf 21D
2 题意：
3 给你n个点m条边的无向图，求从点1开始经过每条边至少一次最后回到点1的最小路程。
4 就是找** 一条路径可重复的欧拉回路 **。
5 input:
6 cin>>n>>m;
7 for(int i=0; i<m; ++i) cin>>x>>y>>z;
8 n是点数(这个图点的编号1~n)，m是边数，x,y,z表示x和y之间有一条长度为z的无向边
9 1<=n<=15,0<=m<=2000,1<=x,y<=n,1<=w<=10000
10 思路：
11 首先对于欧拉回路：所有点的度数都为偶数。因为所有点至少经过一次，那么可以把题意转换成最少加多
    少条边使得图满足以上结论。
12 而加边的目的是为了把奇度数转化为偶度数，先floyd一下得到全源最短路。dp[i]表示状态i下度数为偶
    数的最小花费，因为n<=15,想到状压dp,挑两个奇度数的点转移即可。详情可见代码。
13
14 const int N=100;
15 const int INF=0x3f3f3f3f;
16 int n,m,x,y,w,dis[N][N],deg[N],dp[(1<<20)];
17 int main() {
18     cin>>n>>m;
19     for(int i=0; i<=n+5; ++i){
20         deg[i]=0;
21         for(int j=0; j<=n+5; ++j) dis[i][j]=INF;
22     }
23     int sum=0;
24     for(int i=0; i<m; ++i){
25         cin>>x>>y>>w;
26         uin(dis[x][y],w);
27         uin(dis[y][x],w);
28         ++deg[x]; ++deg[y];
29         sum+=w;
30     }

```

```

31     for(int k=1; k<=n; ++k) for(int i=1; i<=n; ++i) for(int j=1; j<=n; ++j) uin(dis[
        i][j],dis[i][k]+dis[k][j]);
32     for(int i=2; i<=n; ++i){
33         if(deg[i]>0 && dis[i][1]==INF){ //条件一定有deg[i]>0,因为这题要求的是经过所有边,
            不是所有点。
34             cout<<-1<<'\n';
35             return 0;
36         }
37     }
38     int now=0;
39     for(int i=1; i<=n; ++i) if(deg[i]&1) now|=(1<<(i-1));
40     for(int i=0; i<now+10; ++i) dp[i]=INF;
41     dp[now]=0;
42     int ans=INT_MAX;
43     for(int i=now; i>0; --i){
44         for(int j=1; j<=n; ++j){
45             if(i&(1<<(j-1))){
46                 for(int k=j+1; k<=n; ++k){
47                     if(i&(1<<(k-1))){
48                         uin(dp[i^(1<<(j-1))^(1<<(k-1))],dp[i]+dis[j][k]);
49                     }
50                 }
51             }
52         }
53     }
54     cout<<sum+dp[0]<<'\n';
55     return 0;
56 }

```

1.8 LCA

1.8.1 ST 表

```

1 //预处理O(nlogn) 在线查询O(1)
2 const int maxn = 2*(int)+100; //要开两倍点数量的大小 (欧拉序长度)
3 struct LCA
4 {
5     #define type int
6     struct node{int to;type w;node(){}node(int _to,type _w):to(_to),w(_w){}};
7     type dist[maxn];
8     int path[maxn],dep[maxn],loc[maxn],len[maxn],LOG[maxn],all,n;
9     int dp[25][maxn], point[25][maxn]; //2^20 == 1e6 2^25 == 3e7
10    vector<node> G[maxn];
11    void dfs(int u, int now) {
12        path[++all] = u;
13        loc[u] = all;
14        dep[all] = now;
15        for (node cur : G[u]) {
16            int v = cur.to;
17            if (loc[v]) continue;
18            len[v] = now+1;
19            dist[v] = dist[u]+cur.w;

```

```

20     dfs(v, now+1);
21     path[++all] = u;
22     dep[all] = now;
23 }
24 }
25 void initRMQ(int n)
26 {
27     LOG[0] = -1;
28     for (int i = 1; i <= all; ++i) {
29         dp[0][i] = dep[i];
30         point[0][i] = path[i];
31         LOG[i] = ((i&(i-1)) == 0 ? LOG[i-1]+1 : LOG[i-1]);
32     }
33     for (int i = 1; (1<<i) <= all; ++i) {
34         for (int j = 1; j+(1<<i)-1 <= all; ++j) {
35             if (dp[i-1][j] < dp[i-1][j+(1<<(i-1))]) {
36                 dp[i][j] = dp[i-1][j];
37                 point[i][j] = point[i-1][j];
38             } else {
39                 dp[i][j] = dp[i-1][j+(1<<(i-1))];
40                 point[i][j] = point[i-1][j+(1<<(i-1))];
41             }
42         }
43     }
44 }
45 int queryLCA(int l, int r)
46 {
47     l = loc[l]; r = loc[r];
48     if(l>r) swap(l,r);
49     int k = LOG[r-l+1];
50     /*
51     貌似下面这种写法对于某些数据情况更快, 对于某些数据也更慢 - -
52     记得把上面预处理的LOG删了
53     P 3379
54     int k=0;
55     while((1<<k)<=r-l+1) k++;
56     k--;
57     */
58     if(dp[k][l] < dp[k][r-(1<<k)+1]) return point[k][l];
59     else return point[k][r-(1<<k)+1];
60 }
61
62 type getDist(int a, int b){return dist[a]+dist[b]-2*dist[queryLCA(a,b)];}
63 int getLen(int a, int b){return len[a]+len[b]-2*len[queryLCA(a,b)];}
64 void init(int _n)
65 {
66     n = _n;
67     all = 0;
68     for(int i = 0; i <= n; i++)
69     {
70         loc[i] = 0;
71         dist[i] = 0;
72         len[i] = 0;

```

```

73         G[i].clear();
74     }
75 }
76 void addEdge(int a,int b,type w=1)
77 {
78     G[a].emplace_back(node(b,w));
79     G[b].emplace_back(node(a,w));
80 }
81 void solve(int root)
82 {
83     dfs(root, 1);
84     initRMQ(all);
85 }
86 #undef type
87 }lca;
88
89 int main() {
90
91     n = read();
92     lca.init(n);
93     for (int i = 0; i < n-1; ++i) {
94         int a, b;
95         a = read(); b = read();
96         lca.addEdge(a, b, 1);
97     }
98     lca.solve(1);
99     q = read();
100    while (q--) {
101        int a, b;
102        a = read(); b = read();
103        printf("%d\n", lca.queryLCA(a, b));
104        printf("%d\n", lca.getLen(a, b)); //深度 1
105        printf("%d\n", lca.getDist(a, b)); //长度 w
106    }
107
108    return 0;
109 }

```

1.8.2 离线

```

1  //P3379
2  //时间复杂度O(V + E)
3  const int maxn = 500000+1000;
4  int n, m, s, ans[maxn], vis[maxn], fa[maxn];
5  vvi g(maxn);
6  vector< vector< pair<int, int> > > q(maxn);
7
8  int find_root(int x) {
9      if (x == fa[x]) return x;
10     else {
11         return (fa[x] = find_root(fa[x]));
12     }

```

```

13 }
14 void uni_root(int a, int b) {
15     int aa = find_root(a);
16     int bb = find_root(b);
17     if (aa != bb) fa[bb] = aa;
18 }
19 void LCA(int u, int par) {
20     for (int to : g[u]) {
21         if (to == par) continue;
22         LCA(to, u);
23         uni_root(u, to);
24     }
25     vis[u] = 1;
26     for (auto it : q[u]) {
27         if (vis[it.fi]) {
28             ans[it.se] = find_root(it.fi);
29         }
30     }
31 }
32
33 int main() {
34
35     cin >> n >> m >> s;
36     forn(i, n-1) {
37         int u, v;
38         cin >> u >> v;
39         g[u].eb(v);
40         g[v].eb(u);
41     }
42     forn(i, m) {
43         int u, v;
44         cin >> u >> v;
45         q[u].pb({v, i});
46         q[v].pb({u, i});
47     }
48     ms(vis, 0);
49     for1(i, n) fa[i] = i;
50     LCA(s, 0);
51     forn(i, m) cout << ans[i] << '\n';
52
53     return 0;
54 }

```

1.9 最大团

1.9.1 Bron-Kerbosch

```

1 //O(3^(n/3))
2 #include <bits/stdc++.h>
3 using namespace std;
4 int n;
5 const int maxn = 60;

```

```

6  int ma[maxn], g[maxn][maxn], f[maxn][maxn], ans;
7  int dfs(int cur, int tot) {
8      if (!cur) {
9          if (tot > ans) return ans = tot, 1;
10         return 0;
11     }
12     for (int i = 0, j, u, nxt; i < cur; ++i) {
13         if (cur - i + tot <= ans) return 0;
14         u = f[tot][i], nxt = 0;
15         if (ma[u] + tot <= ans) return 0;
16         for (int j = i + 1; j < cur; ++j) if (g[u][f[tot][j]]) f[tot + 1][nxt++] = f[tot][j];
17         if (dfs(nxt, tot + 1)) return 1;
18     }
19     return 0;
20 }
21 int main() {
22     while (cin >> n) {
23         if (!n) break;
24         ans = 0; //初始化
25         for (int i = 0; i < n; ++i) {
26             for (int j = 0; j < n; ++j) {
27                 int x;
28                 cin >> x;
29                 g[i][j] = g[j][i] = x;
30             }
31         }
32         int k, j;
33         for (int i = n - 1; ~i; dfs(k, 1), ma[i--] = ans) {
34             for (k = 0, j = i + 1; j < n; ++j) {
35                 if (g[i][j]) f[1][k++] = j;
36             }
37         }
38         cout << ans << '\n';
39     }
40     return 0;
41 }

```

1.9.2 常见思路

- 1 给你一个无向图G,
- 2 G的最大独立集是G中两两顶点之间都不相连的最多个数的集合。
- 3 G的最大团是G中两两顶点之间都相连的最多个数的集合。
- 4 1.最大独立集不是唯一的。
- 5 2.性质：无向图的最大团 == 该无向图补图的最大独立集

1.10 拓扑排序

1.10.1 toposort

```

1  bool toposort(vvi &g, vi &inDeg) {
2      queue<int> q;

```

```
3   while (!q.empty()) q.pop();
4   forn(i, n) {
5       if (inDeg[i] == 0) {
6           q.push(i);
7       }
8   }
9   int cnt = 0;
10  while (!q.empty()) {
11      int now = q.front();
12      q.pop();
13      ++cnt;
14      for (auto it : g[now]) {
15          --inDeg[it];
16          if (inDeg[it] == 0) q.push(it);
17      }
18  }
19  if (cnt == n) return true;
20  return false;
21 }
22
23 int main() {
24     cin >> t;
25     while (t--) {
26         cin >> n >> m;
27         vvi g(n);
28         vi inDeg(n, 0);
29         forn(i, m) {
30             int e1, e2;
31             cin >> e1 >> e2;
32             --e1; --e2;
33             g[e1].eb(e2);
34             ++inDeg[e2];
35         }
36         if (toposort(g, inDeg)) cout << "Correct" << '\n';
37         else cout << "Wrong" << '\n';
38     }
39     return 0;
40 }
```

2 计算几何

2.1 点

```
1  const double eps = 1e-5;
2  const double inf = 1e20;
3  const double pi = acos(-1.0);
4  const int maxp = 1010;
5  //Compares a double to zero
6  int sgn(double x) {
7      if (fabs(x) < eps) return 0;
8      if (x < 0) return -1;
9      else return 1;
```

```

10 }
11 //square of a double
12 inline double sqr(double x) {return x * x;}
13 struct Point {
14     double x, y;
15     Point() {}
16     Point(double _x, double _y) {
17         x = _x;
18         y = _y;
19     }
20     void input() {
21         scanf("%lf%lf", &x, &y);
22     }
23     void output() {
24         printf("%.8f %.8f\n", x, y);
25     }
26     bool operator == (Point b) const {
27         return sgn(x - b.x) == 0 && sgn(y - b.y) == 0;
28     }
29     bool operator < (Point b) const {
30         return sgn(x - b.x) == 0 ? sgn(y - b.y) < 0 : x < b.x;
31     }
32     Point operator - (const Point &b) const {
33         return Point(x - b.x, y - b.y);
34     }
35     //点积
36     double operator * (const Point &b) const {
37         return x * b.x + y * b.y;
38     }
39     //叉积
40     double operator ^ (const Point &b) const {
41         return x * b.y - y * b.x;
42     }
43     //返回长度
44     double len() {
45         return hypot(x, y); //库函数
46     }
47     //返回长度的平方
48     double len2() {
49         return x * x + y * y;
50     }
51     //返回两点的距离
52     double distance(Point p) {
53         return hypot(x - p.x, y - p.y);
54     }
55     //返回两点距离的平方
56     double distance2(Point p) {
57         return (x-p.x)*(x-p.x)+(y-p.y)*(y-p.y);
58     }
59     Point operator + (const Point &b) const {
60         return Point(x + b.x, y + b.y);
61     }
62     Point operator * (const double &k) const {

```



```

63     return Point(x * k, y * k);
64 }
65 Point operator / (const double &k) const {
66     return Point(x / k, y / k);
67 }
68 //计算 pa 和 pb 的夹角
69 //就是求这个点看a, b的夹角
70 //测试LightOJ 1203
71 double rad(Point a, Point b) {
72     Point p = *this;
73     return fabs(atan2(fabs((a - p) ^ (b - p)), (a - p) * (b - p)));
74 }
75 //化为长度为r的向量
76 Point trunc(double r) {
77     double l = len();
78     if (!sgn(l)) return *this;
79     r /= l;
80     return Point(x * r, y * r);
81 }
82 //逆时针旋转 90 度
83 Point rotleft() {
84     return Point(-y, x);
85 }
86 //顺时针旋转 90 度
87 Point rotright() {
88     return Point(y, -x);
89 }
90 //绕着 p 点逆时针旋转 angle
91 Point rotat(Point p, double angle) {
92     Point v = (*this) - p;
93     double c = cos(angle), s = sin(angle);
94     return Point(p.x + v.x * c - v.y * s, p.y + v.x * s + v.y * c);
95 }
96 };
97 //叉积
98 double cross(Point A, Point B, Point C){
99     return (B-A)^(C-A);
100 }
101 //点积
102 double dot(Point A, Point B, Point C){
103     return (B-A)*(C-A);
104 }

```

2.2 线

```

1 struct Line {
2     Point s, e;
3     Line() {}
4     Line(Point _s, Point _e) {
5         s = _s;
6         e = _e;
7     }

```

```

8   bool operator == (Line v) {
9       return (s == v.s) && (e == v.e);
10  }
11  //根据一个点和倾斜角 angle 确定直线,  $0 \leq \text{angle} < \pi$ 
12  Line (Point p, double angle) {
13      s = p;
14      if (sgn(angle - pi / 2) == 0) {
15          e = (s + Point(0, 1));
16      } else {
17          e = (s + Point(1, tan(angle)));
18      }
19  }
20  //ax + by + c == 0
21  Line(double a, double b, double c) {
22      if (sgn(a) == 0) {
23          s = Point(0, -c / b);
24          e = Point(1, -c / b);
25      } else if (sgn(b) == 0) {
26          s = Point(-c / a, 0);
27          e = Point(-c / a, 1);
28      } else {
29          s = Point(0, -c / b);
30          e = Point(1, (-c - a) / b);
31      }
32  }
33  void input() {
34      s.input();
35      e.input();
36  }
37  void adjust() {
38      if (e < s) swap(s, e);
39  }
40  //求线段长度
41  double length() {
42      return s.distance(e);
43  }
44  //返回直线倾斜角
45  double angle() {
46      double k = atan2(e.y - s.y, e.x - s.x);
47      if (sgn(k) < 0) k += pi;
48      if (sgn(k - pi) == 0) k -= pi;
49      return k;
50  //点和直线关系 要保证s.y>e.y, 如<要swap
51  }
52  //1 线在点的左侧
53  //2 线在点的右侧
54  //3 点在直线上
55  int relation(Point p) {
56      int c = sgn((p - s) ^ (e - s));
57      if (c < 0) return 1;
58      else if (c > 0) return 2;
59      else return 3;
60  }

```

```

61 //点在线段上的判断
62 bool pointonseg(Point p) {
63     return sgn((p - s) ^ (e-s)) == 0 && sgn((p - s) * (p - e)) <= 0;
64 }
65 //两向量平行 (对应直线平行或重合)
66 bool parallel(Line v) {
67     return sgn((e - s) ^ (v.e - v.s)) == 0;
68 }
69 //两线段相交判断
70 //2 规范相交
71 //1 非规范相交
72 //0 不相交
73 int segcrossseg(Line v) {
74     int d1 = sgn((e - s) ^ (v.s - s));
75     int d2 = sgn((e - s) ^ (v.e - s));
76     int d3 = sgn((v.e - v.s) ^ (s - v.s));
77     int d4 = sgn((v.e - v.s) ^ (e - v.s));
78     if ((d1 ^ d2) == -2 && (d3 ^ d4) == -2) return 2;
79     return (d1 == 0 && sgn((v.s - s) * (v.s - e)) <= 0) ||
80         (d2 == 0 && sgn((v.e - s) * (v.e - e)) <= 0) ||
81         (d3 == 0 && sgn((s - v.s) * (s - v.e)) <= 0) ||
82         (d4 == 0 && sgn((e - v.s) * (e - v.e)) <= 0);
83 }
84 //直线和线段相交判断
85 //-*this line -v seg
86 //2 规范相交
87 //1 非规范相交
88 //0 不相交
89 int linecrossseg(Line v) {
90     int d1 = sgn((e - s) ^ (v.s - s));
91     int d2 = sgn((e - s) ^ (v.e - s));
92     if ((d1 ^ d2) == -2) return 2;
93     return (d1 == 0 || d2 == 0);
94 }
95 //两直线关系
96 //0 平行
97 //1 重合
98 //2 相交
99 int linecrossline(Line v) {
100     if ((*this).parallel(v))
101         return v.relation(s) == 3;
102     return 2;
103 }
104 //求两直线的交点
105 //要保证两直线不平行或重合
106 Point crosspoint(Line v) {
107     double a1 = (v.e - v.s) ^ (s - v.s);
108     double a2 = (v.e - v.s) ^ (e - v.s);
109     return Point((s.x * a2 - e.x * a1) / (a2 - a1),
110                 (s.y * a2 - e.y * a1) / (a2 - a1));
111 }
112 //点到直线的距离
113 //测试 : cf614C

```

```

114 double dispointtoline(Point p) {
115     return fabs((p - s) ^ (e - s)) / length();
116 }
117 //点到线段的距离
118 //测试 : cf614C
119 double dispointtoseg(Point p) {
120     if (sgn((p - s) * (e - s)) < 0 || sgn((p - e) * (s - e)) < 0)
121         return min(p.distance(s), p.distance(e));
122     return dispointtoline(p);
123 }
124 //返回线段到线段的距离
125 //前提是两线段不相交, 相交距离就是0了
126 double dissegtoseg(Line v) {
127     return min(min(dispointtoseg(v.s), dispointtoseg(v.e)),
128               min(v.dispointtoseg(s), dispointtoseg(e)));
129 }
130 //返回p在直线上的投影
131 Point lineprog(Point p) {
132     return s + ( ((e - s) * ((e - s) * (p - s))) / ((e - s).len2()) );
133 }
134 //返回点p关于直线的对称点
135 Point symmetrypoint(Point p) {
136     Point q = lineprog(p);
137     return Point(2 * q.x - p.x, 2 * q.y - p.y);
138 }
139 };

```

2.3 圆

```

1 struct circle{
2     Point p;
3     double r;
4     circle(){}
5     circle(Point _p, double _r){
6         p=_p;
7         r=_r;
8     }
9     circle(double x, double y, double _r){
10        p=Point(x,y);
11        r=_r;
12    }
13    //面积
14    double area(){
15        return pi*r*r;
16    }
17    //周长
18    double circumference(){
19        return 2*pi*r;
20    }
21    //两圆的关系
22    //5 相离
23    //4 外切

```

```

24 //3 相交
25 //2 内切
26 //1 内含
27 //需要Point的distance
28 //测试: UVA12304
29 int relationcircle(circle v){
30     double d=p.distance(v.p);
31     if(sgn(d-r-v.r)>0) return 5;
32     if(sgn(d-r-v.r)==0) return 4;
33     double l=fabs(r-v.r);
34     if(sgn(d-r-v.r)<0 && sgn(d-l)>0) return 3;
35     if(sgn(d-l)==0) return 2;
36     if(sgn(d-l)<0) return 1;
37 }
38 //求两圆相交的面积
39 //cf 600D 毒瘤题, 要开long double
40 double areacircle(circle v){
41     int rel=relationcircle(v);
42     if(rel>=4) return 0.0;
43     if(rel<=2) return min(area(),v.area());
44     double d=p.distance(v.p);
45     double a1=2*acos((1.0*r*r+d*d-1.0*v.r*v.r)/(2.0*r*d));
46     double a2=2*acos((1.0*v.r*v.r+d*d-1.0*r*r)/(2.0*v.r*d));
47     double ans=1.0*r*r*a1/2.0 + 1.0*v.r*v.r*a2/2.0 - 1.0*r*r*sin(a1)/2.0 - 1.0*v.
        r*v.r*sin(a2)/2.0;
48     return ans;
49 }
50 };

```

2.4 多边形

```

1 struct polygon{
2     int n;
3     Point p[maxp];
4     Line l[maxp];
5     void input(int _n){
6         n=_n;
7         for(int i=0; i<n; ++i) p[i].input();
8         //半平面交有时要加上reverse(p,p+n)把它们变成逆时针顺序
9     }
10    void add(Point q){
11        p[n++]=q;
12    }
13    void getline(){
14        for(int i=0; i<n; ++i) l[i]=Line(p[i],p[(i+1)%n]);
15    }
16    struct cmp{
17        Point p;
18        cmp(const Point &p0){p=p0;}
19        bool operator()(const Point &aa,const Point &bb){
20            Point a=aa,b=bb;
21            int d=sgn((a-p)^(b-p));

```

```

22         if(d==0){
23             return sgn(a.distance(p)-b.distance(p))<0;
24         }
25         return d>0;
26     }
27 };
28 //进行极角排序
29 //首先需要找到最左下角的点
30 //需要重载号好Point的<操作符 (min函数要用)
31 void norm(){
32     Point mi=p[0];
33     for(int i=1;i<n;++i) mi=min(mi,p[i]);
34     sort(p,p+n,cmp(mi));
35 }
36 //得到凸包
37 //得到凸包里面的点编号是0~n-1的
38 //两种凸包的方法
39 //注意如果有影响,要特判下所有点共点,或者共线的特殊情况
40 //测试 Light0J1203 Light1239
41 void getconvex(polygon &convex){
42     sort(p,p+n);
43     convex.n=n;
44     for(int i=0; i<min(n,2); ++i) convex.p[i]=p[i];
45     if(convex.n==2 && (convex.p[0]==convex.p[1])) convex.n--; //特判
46     if(n<=2) return;
47     int &top=convex.n;
48     top=1;
49     for(int i=2; i<n; ++i){
50         while(top && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i]))<=0) --top;
51         convex.p[++top]=p[i];
52     }
53     int temp=top;
54     convex.p[++top]=p[n-2];
55     for(int i=n-3; i>=0; --i){
56         while(top!=temp && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i]))<=0)
57             top--;
58         convex.p[++top]=p[i];
59     }
60     if(convex.n==2 && (convex.p[0]==convex.p[1])) convex.n--; //特判
61     convex.norm(); //原来的得到的是顺时针的点,排序后逆时针。
62 }
63 //得到凸包的另外一种方法
64 //测试 Light0J1203 Light0J1239
65 void Graham(polygon &convex){
66     norm();
67     int &top=convex.n;
68     top=0;
69     if(n==1){
70         top=1;
71         convex.p[0]=p[0];
72         return;
73     }
74     if(n==2){

```

```

74         top=2;
75         convex.p[0]=p[0];
76         convex.p[1]=p[1];
77         if(convex.p[0]==convex.p[1]) --top;
78         return;
79     }
80     convex.p[0]=p[0];
81     convex.p[1]=p[1];
82     top=2;
83     for(int i=2; i<n; ++i){
84         while(top>1 && sgn((convex.p[top-1]-convex.p[top-2])^
85             (p[i]-convex.p[top-2]))<=0) --top; //这边 <= 改成 < 就能把凸
            包上共线的点也加进去
86         convex.p[top++]=p[i];
87     }
88     if(convex.n==2 && convex.p[0]==convex.p[1]) convex.n--; //特判
89 }
90 //判断是不是凸的
91 bool isconvex(){
92     bool s[3];
93     memset(s, false, sizeof(s));
94     for(int i=0; i<n; ++i){
95         int j=(i+1)%n;
96         int k=(j+1)%n;
97         s[sgn((p[j]-p[i])^(p[k]-p[i]))+1)=true;
98         if(s[0] && s[2]) return false;
99     }
100     return true;
101 }
102 //旋转卡壳
103 double rotate_calipers(){
104     double ans=0;
105     if(n==1) return ans;
106     else if(n==2){
107         ans=max(ans, p[0].distance(p[1]));
108         return ans;
109     }
110     else if(n==3){
111         ans=max(ans, p[0].distance(p[1]));
112         ans=max(ans, p[0].distance(p[2]));
113         ans=max(ans, p[1].distance(p[2]));
114         return ans;
115     } else{
116         int i, j=1;
117         p[n++]=p[0];
118         for(int i=0; i<n; ++i){
119             for(; cross(p[i+1], p[j+1], p[i])>cross(p[i+1], p[j], p[i]); j=(j+1)%n);
120             ans=max(ans, max(p[i].distance(p[j]), p[i+1].distance(p[j]))); //最远点对
121         }
122         return ans;
123     }
124 }
125 //得到周长

```

```

126 //测试 LightOj1239
127 double getcircumference(){
128     double sum=0;
129     for(int i=0; i<n; ++i){
130         sum+=p[i].distance(p[(i+1)%n]);
131     }
132     return sum;
133 }
134 //得到面积
135 double getarea(){
136     double sum=0;
137     for(int i=0; i<n; ++i) sum+=(p[i]^p[(i+1)%n]);
138     return fabs(sum)/2;
139 }
140 };

```

2.5 半平面交

```

1 //半平面交 O(nlogn)
2 //测试 POJ3335 POJ1474 POJ1279
3 struct halfplane:public Line{
4     double angle;
5     halfplane(){}
6     //表示向量s->e逆时针(左侧)的半平面
7     //所以要把所有点都变为逆时针排序
8     halfplane(Point _s,Point _e){
9         s=_s;
10        e=_e;
11    }
12    halfplane(Line v){
13        s=v.s;
14        e=v.e;
15    }
16    void calcangle(){
17        angle=atan2(e.y-s.y,e.x-s.x);
18    }
19    bool operator <(const halfplane &b) const{
20        return angle<b.angle;
21    }
22 };
23 struct halfplanes{
24     int n; //别忘了给n赋值!
25     halfplane hp[maxp];
26     Point p[maxp];
27     int que[maxp];
28     int st,ed;
29     void push(halfplane tmp){
30         hp[n++]=tmp;
31     }
32     //去重
33     void unik(){
34         int m=1;

```



```

35     for(int i=1; i<n; ++i){
36         if(sgn(hp[i].angle-hp[i-1].angle)!=0){
37             hp[m++]=hp[i];
38         } else if(sgn((hp[m-1].e-hp[m-1].s)^(hp[i].s-hp[m-1].s))>0){
39             hp[m-1]=hp[i];
40         }
41     }
42     n=m;
43 }
44 bool halfplaneinsert(){
45     for(int i=0; i<n; ++i) hp[i].calcangle();
46     sort(hp, hp+n);
47     unik();
48     que[st=0]=0;
49     que[ed=1]=1;
50     p[1]=hp[0].crosspoint(hp[1]);
51     for(int i=2; i<n; ++i){
52         while(st<ed && sgn((hp[i].e-hp[i].s)^(p[ed]-hp[i].s))<0) --ed;
53         while(st<ed && sgn((hp[i].e-hp[i].s)^(p[st+1]-hp[i].s))<0) ++st;
54         que[++ed]=i;
55         if(hp[i].parallel(hp[que[ed-1]])) return false;
56         p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
57     }
58     while(st<ed && sgn((hp[que[st]].e-hp[que[st]].s)^(p[ed]-hp[que[st]].s))<0) --ed;
59     while(st<ed && sgn((hp[que[ed]].e-hp[que[ed]].s)^(p[st+1]-hp[que[ed]].s))<0) ++st;
60     if(st+1>=ed) return false;
61     return true;
62 }
63 //得到最后半平面交得到的凸多边形
64 //需要先调用halfplaneinsert()且返回true
65 void getconvex(polygon &con){
66     p[st]=hp[que[st]].crosspoint(hp[que[ed]]);
67     con.n=ed-st+1;
68     for(int j=st, i=0; j<=ed; ++i, ++j)
69         con.p[i]=p[j];
70 }
71 };
72
73 scanf("%d", &n);
74 plo.input(n);
75 plo.getline();
76 hfp.n=n;
77 for(int i=0; i<n; ++i){
78     hfp.hp[i]=plo.l[i];
79 }
80 if(hfp.halfplaneinsert()) puts("YES");
81 else puts("NO");

```

2.6 function

```

1  bool sameline(Point a, Point b, Point c){ //判断三点是否共线
2      double ax=c.x-a.x, ay=c.y-a.y;
3      double bx=c.x-b.x, by=c.y-b.y;
4      if(sgn(ax*by-ay*bx)==0)return true;
5      return false;
6  }
7  bool squ(Point a,Point b,Point c,Point d){ //判断正方形 cf136D
8      vector<double> temp;
9      double l11=a.distance(b); temp.eb(l11);
10     double l22=b.distance(c); temp.eb(l22);
11     double l33=c.distance(d); temp.eb(l33);
12     double l44=d.distance(a); temp.eb(l44);
13     double l55=a.distance(c); temp.eb(l55);
14     double l66=b.distance(d); temp.eb(l66);
15     sort(all(temp));
16     if(!sgn(temp[0]-temp[1]) && !sgn(temp[1]-temp[2]) && !sgn(temp[2]-temp[3]) && !
        sgn(temp[4]-temp[5]) && sgn(temp[3]-temp[4])==-1) return true;
17     return false;
18 }
19 bool rec(Point a,Point b,Point c,Point d){ //判断矩形 cf136D
20     Point ct=Point((a.x+b.x+c.x+d.x)/4,(a.y+b.y+c.y+d.y)/4); //求质心
21     double l11=ct.distance(a);
22     double l22=ct.distance(b);
23     double l33=ct.distance(c);
24     double l44=ct.distance(d);
25     if(!sgn(l11-l22) && !sgn(l11-l33) && !sgn(l11-l44)) return true;
26     return false;
27 }

```

2.7 注意点

```

1  1. 计算几何题本地有可能开不了那么多点，交上去的时候一定别忘了改N的值啊！！
2  2. 如果你觉得思路没问题，但一直wa，那很有可能是精度问题，把1e-5改成1e-3 或 1e-5改成1e-8试
   试
3  3. 要多注意n=1,2.. 这种小值时候是否要特判
4  4. 对于某些题，要注意所有点共线的情况
5  5. 求直角三角形的斜边用hypot的时候，要注意如果要判断斜边是不是整数先要加一个eps再向下取整，
   比如3,4求出来的可能会是4.999999 (cf40A)
6  如
7  const double eps=1e-8;
8  int tmp=hypot(x,y)+eps;
9  if(tmp*tmp==x*x+y*y){
10     puts("black");
11     return 0;
12 }
13 或者
14 const double eps=1e-8;
15 double d2=floor(dist+eps);
16 if(!sgn(dist-d2)){
17     puts("black");
18     return 0;
19 }

```

- 20 6. 如果出现点坐标都是 $[-1e9, 1e9]$ && 用到 $(a-b)^{(c-d)}$ && 用的`int`存点 -> 会爆`int`, 要开`1l(UVA 11930)`, 不放心就`#define int long long`

2.8 常出现的模型

- 1 1. 给你一个多边形, 看给出的点是按顺时针还是按逆时针给的
 2 对于凸多边形, 看一个点和旁边的点的叉积就行了
 3 对于 n 个点的普通多边形, 要做 n 次 i 点和 $i+1$ 点 (n 点是 0 点) 的叉积, 然后加起来, 如果是负数就为逆时针, 如果是正数就为顺时针

2.9 奇怪的技巧

- 1 1. 输出`long double "%Lf" "#define __USE_MINGW_ANSI_STDIO 1"`

2.10 凸包

2.10.1 判断是否是稳定凸包

- 1 POJ 1228 数据水
 2 所谓稳定凸包就是不存在凸包外加入一个点使得形成的新凸包还包含原凸包的所有点。
 3 所以求完凸包后的每条边上至少要有三个点
 4 题意:
 5 有一个多边形的凸包, 但这个凸包上一条边有可能除了两条端点还有点, 现在拿掉不知道多少个点, 问剩下的点是不是稳定凸包?
 6 既判断这 n 个点连成的多边形是不是稳定凸包?
 7 思路:
 8 1. 如果点数 <6 , 那么肯定不是稳定凸包。
 9 2. 如果所有点都共线, 那么肯定不是稳定凸包。
 10 3. 对剩下的这 n 个点求凸包, 然后枚举凸包的每条边, 看原多边形是否有一个不是这条边的两个端点的点在这条线上。如果每条边都有, 那就是稳定凸包了。
- ```

11
12 signed main() {
13 int tc;
14 scanf("%d",&tc);
15 while(tc--){
16 scanf("%d",&n);
17 plo.input(n);
18 if(n<6){
19 puts("NO");
20 continue;
21 }
22 bool oneline=true;
23 for(int i=0; i<n-2; ++i){
24 if(!sameline(plo.p[i],plo.p[i+1],plo.p[i+2])){
25 oneline=false;
26 break;
27 }
28 }
29 if(oneline){
30 puts("NO");

```

```

31 continue;
32 }
33 plo.Graham(cv);
34 bool ok=true;
35 cv.getline();
36 for(int i=0; i<cv.n; ++i){
37 bool exec=false;
38 for(int j=0; j<plo.n; ++j){
39 if(cv.l[i].s==plo.p[j]) continue;
40 if(cv.l[i].e==plo.p[j]) continue;
41 if(sameline(cv.l[i].s,cv.l[i].e,plo.p[j])){
42 exec=true;
43 break;
44 }
45 }
46 if(!exec){
47 ok=false;
48 break;
49 }
50 }
51 puts(ok?"YES":"NO");
52 }
53 return 0;
54 }

```

## 2.11 旋转卡壳

### 2.11.1 SCOI2007 最大土地面积

```

1 1. SCOI 2007 最大土地面积 / cf gym102460 L
2 //给你n个点，让你取四个点形成一个四边形，问最大的四边形面积是多少？
3 //凸包后枚举对踵点 $O(n^2)$ ，然后旋转卡壳，因为是在枚举对踵点里面卡壳，所以卡壳的复杂度是 $O(1)$ ，
 总复杂度 $O(n^2)$
4 //n=2000, x,y是double, x,y<=1e5 0.2s
5 #define i64 long long
6 const double eps = 1e-8;
7 const double inf = 1e20;
8 const double pi = acos(-1.0);
9 const int maxp = 40100;
10 int n;
11 //Compares a double to zero
12 int sgn(double x) {
13 if (fabs(x) < eps) return 0;
14 if (x < 0) return -1;
15 else return 1;
16 }
17 struct Point {
18 double x, y;
19 int idx;
20 Point() {}
21 Point(double _x, double _y, int _idx) {
22 x = _x;

```

```

23 y = _y;
24 idx = _idx;
25 }
26 void input() {
27 scanf("%lf%lf", &x, &y);
28 }
29 void output() {
30 printf("%.8f %.8f\n", x, y);
31 }
32 bool operator == (Point b) const {
33 return sgn(x - b.x) == 0 && sgn(y - b.y) == 0;
34 }
35 bool operator < (Point b) const {
36 return sgn(x - b.x) == 0 ? sgn(y - b.y) < 0 : x < b.x;
37 }
38 Point operator - (const Point &b) const {
39 return Point(x - b.x, y - b.y, idx - b.idx);
40 }
41 //点积
42 double operator * (const Point &b) const {
43 return x * b.x + y * b.y;
44 }
45 //叉积
46 double operator ^ (const Point &b) const {
47 return x * b.y - y * b.x;
48 }
49 //返回两点的距离
50 double distance(Point p) {
51 return hypot(x - p.x, y - p.y);
52 }
53 }fp;
54 struct polygon{
55 int n;
56 Point p[maxp];
57 void input(int _n){
58 n=_n;
59 for(int i=0; i<n; ++i){
60 p[i].input();
61 p[i].idx=i;
62 }
63 }
64 void add(Point q){
65 p[n++]=q;
66 }
67 struct cmp{
68 Point p;
69 cmp(const Point &p0){p=p0;}
70 bool operator()(const Point &aa,const Point &bb){
71 Point a=aa,b=bb;
72 int d=sgn((a-p)^(b-p));
73 if(d==0){
74 return sgn(a.distance(p)-b.distance(p))<0;
75 }

```

```

76 return d>0;
77 }
78 };
79 //进行极角排序
80 //首先需要找到最左下角的点
81 //需要重载号好Point的<操作符 (min函数要用)
82 void norm(){
83 Point mi=p[0];
84 for(int i=1;i<n;++i) mi=min(mi,p[i]);
85 sort(p,p+n,cmp(mi));
86 }
87 //得到凸包
88 //得到凸包里面的点编号是0~n-1的
89 //两种凸包的方法
90 //注意如果有影响,要特判下所有点共点,或者共线的特殊情况
91 //测试 LightOJ1203 Light1239
92 void getconvex(polygon &convex){
93 sort(p,p+n);
94 convex.n=n;
95 for(int i=0; i<min(n,2); ++i) convex.p[i]=p[i];
96 if(convex.n==2 && (convex.p[0]==convex.p[1])) convex.n--; //特判
97 if(n<=2) return;
98 int &top=convex.n;
99 top=1;
100 for(int i=2; i<n; ++i){
101 while(top && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i]))<=0) --top;
102 convex.p[++top]=p[i];
103 }
104 int temp=top;
105 convex.p[++top]=p[n-2];
106 for(int i=n-3; i>=0; --i){
107 while(top!=temp && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i]))<=0)
108 top--;
109 convex.p[++top]=p[i];
110 }
111 if(convex.n==2 && (convex.p[0]==convex.p[1])) convex.n--; //特判
112 convex.norm(); //原来的得到的是顺时针的点,排序后逆时针。
113 }
114 bool OK(Point a,Point b,Point c,Point d)
115 {
116 double A=fabs((b-a)^(c-a));
117 double B=fabs((b-a)^(d-a));
118 return A<B;
119 }
120 void Rotating_Calipers(polygon &plo)
121 {
122 double ans=0,now;
123 if (n<=2) ans=0;
124 else if (n==3)
125 {
126 now=~(1LL<<63); //LLONG_MAX
127 ans=abs((p[0]-p[2])^(p[1]-p[2]));
128 for (int i=0;i<plo.n;i++)

```

```

128 {
129 if (p[0].idx==plo.p[i].idx || p[1].idx==plo.p[i].idx || p[2].idx==plo.p
 [i].idx) continue;
130 double s1=fabs((p[0]-plo.p[i])^(p[1]-plo.p[i]));
131 double s2=fabs((p[1]-plo.p[i])^(p[2]-plo.p[i]));
132 double s3=fabs((p[2]-plo.p[i])^(p[0]-plo.p[i]));
133 now=min(min(now,s1),min(s2,s3));
134 }
135 if (now!=~(1LL<<63)) ans-=now;
136 else ans=0;
137 }
138 else
139 {
140 for (int i=0;i<n;i++)
141 {
142 int x=(i+1)%n,y=(i+2)%n;
143 for (int j=(i+2)%n;j!=i; (++j)%=n)
144 {
145 while (x!=j && OK(p[i],p[j],p[x],p[x+1])) (++x)%=n;
146 while (y!=i && OK(p[i],p[j],p[y],p[y+1])) (++y)%=n;
147 now=fabs((p[x]-p[i])^(p[j]-p[i]))+fabs((p[y]-p[i])^(p[j]-p[i]));
148 if (now>ans) ans=now;
149 }
150 }
151 }
152 printf("%.3lf\n",ans/2.0);
153 return ;
154 }
155 }plo;
156 int main() {
157 int tc;
158 scanf("%d",&tc);
159 while(tc--){
160 scanf("%d",&n);
161 plo.input(n);
162 polygon cv; plo.getconvex(cv); //得到凸包
163 cv.Rotating_Calipers(plo); //旋转卡壳
164 }
165 return 0;
166 }
167
168
169 //n<=4000, x,y是整数, x,y<=1e9 5s
170 #define i64 long long
171 const double eps = 1e-8;
172 const double inf = 1e20;
173 const double pi = acos(-1.0);
174 const int maxp = 40100;
175 int n;
176 //Compares a double to zero
177 int sgn(double x) {
178 if (fabs(x) < eps) return 0;
179 if (x < 0) return -1;

```

```

180 else return 1;
181 }
182 struct Point {
183 i64 x, y;
184 int idx;
185 Point() {}
186 Point(i64 _x, i64 _y, int _idx) {
187 x = _x;
188 y = _y;
189 idx = _idx;
190 }
191 void input() {
192 scanf("%lld%lld", &x, &y);
193 }
194 void output() {
195 printf("%.8f %.8f\n", x, y);
196 }
197 bool operator == (Point b) const {
198 return sgn(x - b.x) == 0 && sgn(y - b.y) == 0;
199 }
200 bool operator < (Point b) const {
201 return sgn(x - b.x) == 0 ? sgn(y - b.y) < 0 : x < b.x;
202 }
203 Point operator - (const Point &b) const {
204 return Point(x - b.x, y - b.y, idx - b.idx);
205 }
206 //点积
207 //叉积
208 i64 operator ^ (const Point &b) const {
209 return x * b.y - y * b.x;
210 }
211 //返回两点的距离
212 double distance(Point p) {
213 return hypot(x - p.x, y - p.y);
214 }
215 }fp;
216 struct polygon{
217 int n;
218 Point p[maxp];
219 void input(int _n){
220 n=_n;
221 for(int i=0; i<n; ++i){
222 p[i].input();
223 p[i].idx=i;
224 }
225 }
226 void add(Point q){
227 p[n++]=q;
228 }
229 struct cmp{
230 Point p;
231 cmp(const Point &p0){p=p0;}
232 bool operator()(const Point &aa,const Point &bb){

```



```

233 Point a=aa,b=bb;
234 int d=sgn((a-p)^(b-p));
235 if(d==0){
236 return sgn(a.distance(p)-b.distance(p))<0;
237 }
238 return d>0;
239 }
240 };
241 //进行极角排序
242 //首先需要找到最左下角的点
243 //需要重载号好Point的<操作符 (min函数要用)
244 void norm(){
245 Point mi=p[0];
246 for(int i=1;i<n;++i) mi=min(mi,p[i]);
247 sort(p,p+n,cmp(mi));
248 }
249 //得到凸包
250 //得到凸包里面的点编号是0~n-1的
251 //两种凸包的方法
252 //注意如果有影响,要特判下所有点共点,或者共线的特殊情况
253 //测试 Light0J1203 Light1239
254 void getconvex(polygon &convex){
255 sort(p,p+n);
256 convex.n=n;
257 for(int i=0; i<min(n,2); ++i) convex.p[i]=p[i];
258 if(convex.n==2 && (convex.p[0]==convex.p[1])) convex.n--; //特判
259 if(n<=2) return;
260 int &top=convex.n;
261 top=1;
262 for(int i=2; i<n; ++i){
263 while(top && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i]))<=0) --top;
264 convex.p[++top]=p[i];
265 }
266 int temp=top;
267 convex.p[++top]=p[n-2];
268 for(int i=n-3; i>=0; --i){
269 while(top!=temp && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i]))<=0)
 top--;
270 convex.p[++top]=p[i];
271 }
272 if(convex.n==2 && (convex.p[0]==convex.p[1])) convex.n--; //特判
273 convex.norm(); //原来的得到的是顺时针的点,排序后逆时针。
274 }
275 bool OK(Point a,Point b,Point c,Point d)
276 {
277 double A=fabs((b-a)^(c-a));
278 double B=fabs((b-a)^(d-a));
279 return A<B;
280 }
281 void Rotating_Calipers(polygon &plo)
282 {
283 i64 ans=0,now;
284 if (n<=2) ans=0;

```

```

285 else if (n==3)
286 {
287 now=~(1LL<<63); //LLONG_MAX
288 ans=abs((p[0]-p[2])^(p[1]-p[2]));
289 for (int i=0;i<plo.n;i++)
290 {
291 if (p[0].idx==plo.p[i].idx || p[1].idx==plo.p[i].idx || p[2].idx==plo.p
 [i].idx) continue;
292 i64 s1=abs((p[0]-plo.p[i])^(p[1]-plo.p[i]));
293 i64 s2=abs((p[1]-plo.p[i])^(p[2]-plo.p[i]));
294 i64 s3=abs((p[2]-plo.p[i])^(p[0]-plo.p[i]));
295 now=min(min(now,s1),min(s2,s3));
296 }
297 if (now!=~(1LL<<63)) ans-=now;
298 else ans=0;
299 }
300 else
301 {
302 for (int i=0;i<n;i++)
303 {
304 int x=(i+1)%n,y=(i+2)%n;
305 for (int j=(i+2)%n;j!=i; (++j)%=n)
306 {
307 while (x!=j && OK(p[i],p[j],p[x],p[x+1])) (++x)%=n;
308 while (y!=i && OK(p[i],p[j],p[y],p[y+1])) (++y)%=n;
309 now=abs((p[x]-p[i])^(p[j]-p[i]))+fabs((p[y]-p[i])^(p[j]-p[i]));
310 if (now>ans) ans=now;
311 }
312 }
313 }
314 i64 temp=ans/2;
315 if(ans%2==0) printf("%lld\n",temp);
316 else printf("%lld.5\n",temp);
317 return ;
318 }
319 }plo;
320 int main() {
321 int tc;
322 scanf("%d",&tc);
323 while(tc--){
324 scanf("%d",&n);
325 plo.input(n);
326 polygon cv; plo.getconvex(cv); //得到凸包
327 cv.Rotating_Calipers(plo); //旋转卡壳
328 }
329 return 0;
330 }

```

## 2.12 扫描线

### 2.12.1 矩形面积交 hdu1255

```

1 hdu 1255
2 多组数据输入,n<=1000, x,y是浮点数, 范围[0,1e5]
3
4 const int N=2100;
5 const double eps=1e-5;
6 int n,maxNode;
7 double v[N];
8 int sgn(double x){
9 if(fabs(x)<eps) return 0;
10 if(x>0) return 1;
11 else return -1;
12 }
13 struct LI{
14 double x,y11,y22;
15 int state;
16 LI(double _x=0, double _y11=0, double _y22=0, int _state=0){
17 x=_x;
18 y11=_y11;
19 y22=_y22;
20 state=_state;
21 }
22 bool operator < (const LI &rhs) const{
23 return sgn(x-rhs.x)==-1;
24 }
25 }l[N];
26 struct SE{
27 double l,r,len,more;
28 int cover;
29 }seg[N*8];
30 bool cmp(double a,double b){
31 return sgn(a-b)==-1;
32 }
33 void pushUp(int node){
34 if(seg[node].cover>=2){
35 seg[node].more=seg[node].r-seg[node].l;
36 } else if(seg[node].cover==1){
37 seg[node].more=seg[node<<1].len+seg[node<<1|1].len;
38 seg[node].len=seg[node].r-seg[node].l;
39 } else{
40 seg[node].more=seg[node<<1].more+seg[node<<1|1].more;
41 seg[node].len=seg[node<<1].len+seg[node<<1|1].len;
42 }
43 }
44 void build(int node,int l,int r){
45 uax(maxNode,node);
46 seg[node].l=v[l]; seg[node].r=v[r];
47 seg[node].cover=0;
48 seg[node].len=seg[node].more=0;
49 if(r-l<=1) return;
50 int mid=(l+r)>>1;
51 build(node<<1,l,mid);
52 build(node<<1|1,mid,r);
53 }

```

```

54 void update(int node,double ul,double ur,int nowState){ //注意这两个double!
55 if(node>maxNode) return;
56 double l=seg[node].l,r=seg[node].r; //注意这两个double!
57 if(ul<=l && r<=ur){
58 seg[node].cover+=nowState;
59 pushUp(node);
60 return;
61 }
62 if(ul<seg[node<<1].r) update(node<<1,ul,ur,nowState);
63 if(ur>seg[node<<1|1].l) update(node<<1|1,ul,ur,nowState);
64 pushUp(node);
65 }
66 int main() {
67 int tc;
68 scanf("%d",&tc);
69 while(tc--){
70 maxNode=-1;
71 scanf("%d",&n);
72 for(int i=1; i<=n; ++i){
73 double a,b,c,d;
74 scanf("%lf%lf%lf%lf",&a,&b,&c,&d);
75 l[i]=LI(a,b,d,1);
76 l[i+n]=LI(c,b,d,-1);
77 v[i]=b;
78 v[i+n]=d;
79 }
80 sort(l+1,l+1+2*n);
81 sort(v+1,v+1+2*n,cmp);
82 build(1,1,2*n);
83 double sum=0;
84 for(int i=1; i<=2*n; ++i){
85 sum+=seg[1].more*(l[i].x-l[i-1].x);
86 update(1,l[i].y1,l[i].y2,l[i].state);
87 }
88 printf("%.2f\n",sum);
89 }
90 return 0;
91 }

```

### 2.12.2 面积并坐标 double 版

```

1 const int maxn = 2*(int)1e6+1000;
2 int n;
3 double v[maxn];
4 struct Line {
5 double x_;
6 double y_, y__;
7 int state;
8 bool operator < (const Line &oth) const {return x_<oth.x_;}
9 }line[maxn];
10 struct s1 {
11 double l, r;

```

```

12 int cover;
13 double len;
14 }sgt[maxn<<3];
15
16 void pushUp(int node) {
17 if (sgt[node].cover) sgt[node].len = sgt[node].r-sgt[node].l;
18 else sgt[node].len = sgt[node<<1].len+sgt[node<<1|1].len;
19 }
20 void build(int node, int l, int r) {
21 sgt[node].l = v[l]; sgt[node].r = v[r];
22 sgt[node].cover = 0;
23 sgt[node].len = 0;
24 if (r-l <= 1) return;
25 int mid = (l+r)>>1;
26 build(node<<1, l, mid);
27 build(node<<1|1, mid, r);
28 }
29 void update(int node, double ul, double ur, int val) {
30 double l = sgt[node].l, r = sgt[node].r;
31 if (ul <= l && r <= ur) {
32 sgt[node].cover += val;
33 pushUp(node);
34 return;
35 }
36 if (ul < sgt[node<<1].r) update(node<<1, ul, ur, val);
37 if (ur > sgt[node<<1|1].l) update(node<<1|1, ul, ur, val);
38 pushUp(node);
39 }
40
41 int main() {
42
43 while (cin >> n) {
44 if (!n) break;
45 for1(i, n) {
46 double x_, y_, x__, y__;
47 cin >> x_ >> y_ >> x__ >> y__;
48 v[i] = y_; v[i+n] = y__;
49 line[i] = (Line){x_, y_, y__, 1};
50 line[i+n] = (Line){x__, y_, y__, -1};
51 }
52 sort(v+1, v+1+(n<<1));
53 sort(line+1, line+1+(n<<1));
54 build(1, 1, n<<1);
55 double ans = 0;
56 for (int i = 1; i <= (n<<1); ++i) {
57 ans += sgt[1].len*(line[i].x_-line[i-1].x_);
58 update(1, line[i].y_, line[i].y__, line[i].state);
59 }
60 cout << ans << '\n';
61 }
62
63 return 0;
64 }

```

2.12.3 面积并坐标 i64 版  $O(n \lg n)$ 

```

1 洛谷 p5490
2 给你n个矩形，每个矩形给你 左上 和 右下 两点
3
4 const int N=2*(int)1e5+100;
5 int n,v[N],maxNode;
6 struct LI{
7 i64 x;
8 int y1,y2,state;
9 LI(i64 _x=0, int _y1=0, int _y2=0, int _state=0){
10 x=_x;
11 y1=_y1;
12 y2=_y2;
13 state=_state;
14 }
15 bool operator < (const LI &rhs) const{
16 return x<rhs.x;
17 }
18 }l[N];
19 struct SE{
20 int l,r,cover;
21 i64 len;
22 }seg[N*8];
23 void pushUp(int node){
24 if(seg[node].cover>0) seg[node].len=seg[node].r-seg[node].l;
25 else seg[node].len=seg[node<<1].len+seg[node<<1|1].len;
26 }
27 void build(int node, int l, int r){
28 uax(maxNode,node);
29 seg[node].l=v[l]; seg[node].r=v[r];
30 seg[node].cover=seg[node].len=0;
31 if(r-l<=1) return;
32 int mid=(l+r)>>1;
33 build(node<<1,l,mid);
34 build(node<<1|1,mid,r);
35 }
36 void update(int node, int ul, int ur, int nowState){
37 if(node>maxNode) return;
38 int l=seg[node].l,r=seg[node].r;
39 if(ul<=l && r<=ur){
40 seg[node].cover+=nowState;
41 pushUp(node);
42 return;
43 }
44 if(ul<seg[node<<1].r) update(node<<1,ul,ur,nowState);
45 if(ur>seg[node<<1|1].l) update(node<<1|1,ul,ur,nowState);
46 pushUp(node);
47 }
48 int main() {
49 maxNode=-1;
50 scanf("%d",&n);
51 for(int i=1; i<=n; ++i){

```

```

52 i64 a,c;
53 int b,d;
54 scanf("%lld%lld%lld", &a, &b, &c, &d);
55 l[i]=LI(a,b,d,1);
56 l[i+n]=LI(c,b,d,-1);
57 v[i]=b;
58 v[i+n]=d;
59 }
60 sort(l+1,l+1+2*n);
61 sort(v+1,v+1+2*n);
62 build(1,1,2*n);
63 i64 ans=0;
64 for(int i=1; i<=2*n; ++i){
65 ans+=seg[1].len*(l[i].x-l[i-1].x);
66 update(1,l[i].y1,l[i].y2,l[i].state);
67 }
68 printf("%lld\n",ans);
69 return 0;
70 }

```

#### 2.12.4 周长并 $O(n \lg n)$

```

1 const int maxn = 2*(int)1e5+1000;
2 int n;
3 int v[maxn];
4 struct Line {
5 int x_;
6 int y_, y__;
7 int state;
8 bool operator < (const Line &oth) const {return x_<oth.x_;}
9 }line[maxn];
10 struct s1 {
11 int l, r;
12 int cover;
13 i64 len;
14 int lcover, rcover;
15 int diff;
16 }sgt[maxn<<3]; //注意这个大小
17 void pushUp(int node) {
18 if (sgt[node].cover) {
19 sgt[node].len = sgt[node].r-sgt[node].l;
20 sgt[node].lcover = sgt[node].rcover = 1;
21 sgt[node].diff = 1;
22 } else {
23 sgt[node].len = sgt[node<<1].len+sgt[node<<1|1].len;
24 sgt[node].lcover = sgt[node<<1].lcover;
25 sgt[node].rcover = sgt[node<<1|1].rcover;
26 sgt[node].diff = sgt[node<<1].diff+sgt[node<<1|1].diff-(sgt[node<<1].rcover&
 sgt[node<<1|1].lcover);
27 }
28 }
29 void build(int node, int l, int r) {

```

```

30 sgt[node].l = v[l]; sgt[node].r = v[r];
31 sgt[node].cover = 0;
32 sgt[node].len = 0;
33 sgt[node].lcover = sgt[node].rcover = 0;
34 sgt[node].diff = 0;
35 if (r-l <= 1) return;
36 int mid = (l+r)>>1;
37 build(node<<1, l, mid);
38 build(node<<1|1, mid, r);
39 }
40 void update(int node, int ul, int ur, int val) {
41 int l = sgt[node].l, r = sgt[node].r;
42 if (ul <= l && r <= ur) {
43 sgt[node].cover += val;
44 pushUp(node);
45 return;
46 }
47 if (ul < sgt[node<<1].r) update(node<<1, ul, ur, val);
48 if (ur > sgt[node<<1|1].l) update(node<<1|1, ul, ur, val); //或者ur>sgt[node<<1].
 r 因为sgt[node<<1].r==sgt[node<<1|1].l
49 pushUp(node);
50 }
51 i64 myabs(i64 a) {return (a>0?a:-a);}
52
53 int main() {
54
55 cin >> n;
56 for1(i, n) {
57 int x_, y_, x__, y__;
58 cin >> x_ >> y_ >> x__ >> y__; //左下角坐标 右上角坐标
59 v[i] = y_; v[i+n] = y__;
60 line[i] = (Line){x_, y_, y__, 1};
61 line[i+n] = (Line){x__, y_, y__, -1};
62 }
63 sort(v+1, v+1+(n<<1));
64 sort(line+1, line+1+(n<<1));
65 build(1, 1, n<<1);
66 i64 ans = 0;
67 i64 lst = 0;
68 for (int i = 1; i <= (n<<1); ++i) {
69 ans += (line[i].x_-line[i-1].x_)*2*sgt[1].diff; //vertical
70 update(1, line[i].y_, line[i].y__, line[i].state);
71 ans += myabs(sgt[1].len-lst); //horizontal
72 lst = sgt[1].len;
73 }
74 cout << ans << '\n';
75
76 return 0;
77 }

```



## 2.13 半平面交

### 2.13.1 经典例题

```

1 codechef ALLPOLY
2 题意：
3 顺时针顺序给你n个点，让你输出这n个点形成多边形的核的面积/1e14。
4 思路：
5 傻逼卡精题，不想搞了，后面有过了的代码。

```

### 2.13.2 codechef ALLPOLY

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define N 100111
4 typedef long double D;
5 typedef long long L;
6 #define INF 1e111
7 #define EPS 1e-11
8 #define BEPS 1e-6
9
10 bool between(D a, D b, D c) {
11 return a-BEPS <= b && b <= c+BEPS || a+BEPS >= b && b >= c-BEPS;
12 }
13
14
15
16 template<class T>
17 struct pt {
18 T x, y;
19 pt() {}
20 pt(T x, T y): x(x), y(y) {}
21 pt operator+(const pt p) const {
22 return pt(x + p.x, y + p.y);
23 }
24 pt operator-(const pt p) const {
25 return pt(x - p.x, y - p.y);
26 }
27 pt scale(T s) const {
28 return pt(x * s, y * s);
29 }
30 T dot(pt p) {
31 return x * p.x + y * p.y;
32 }
33 T cross(pt p) {
34 return x * p.y - y * p.x;
35 }
36 pt operator-() const {
37 return scale(-1);
38 }
39 double mag() {
40 return hypot(x, y);
41 }

```

```

42 bool operator==(const pt p) const {
43 return (x == p.x && y == p.y);
44 }
45 bool operator!=(const pt p) const {
46 return !(*this == p);
47 }
48 bool operator<(const pt p) const {
49 return x < p.x || x == p.x && y < p.y;
50 }
51 int quad() {
52 return x > 0 ? (y > 0 ? 2 : y < 0 ? 8 : 1):
53 x < 0 ? (y > 0 ? 4 : y < 0 ? 6 : 5):
54 (y > 0 ? 3 : y < 0 ? 7 : 0);
55 }
56 };
57
58 typedef pt<D> ptD;
59 typedef pt<L> ptL;
60
61 ptD normalize(ptD p) {
62 return p.scale(100./p.mag());
63 }
64
65 ptD normalize(ptL p) {
66 return normalize(ptD(p.x, p.y));
67 }
68
69
70
71
72 template<class T>
73 struct seg {
74 pt<T> a, b;
75 seg() {}
76 seg(pt<T> a, pt<T> b): a(a), b(b) {}
77
78 seg operator-() const {
79 return seg(-a, -b);
80 }
81 seg swap() {
82 return seg(b, a);
83 }
84 pt<T> vec() {
85 return b - a;
86 }
87 bool operator==(const seg p) const {
88 return (a == p.a && b == p.b);
89 }
90 bool operator!=(const seg p) const {
91 return !(*this == p);
92 }
93 bool operator<(const seg p) const {
94 return a < p.a || a == p.a && b < p.b;

```

```

95 }
96 T area() {
97 return a.cross(b);
98 }
99 };
100
101 typedef seg<D> segD;
102 typedef seg<L> segL;
103
104 ptD line_intersect(segD a, segD b) {
105 ptD p1 = a.a;
106 ptD v1 = normalize(a.vec());
107 ptD p2 = b.a;
108 ptD v2 = normalize(b.vec());
109
110 D den = v1.cross(v2);
111 assert(fabs(den) > EPS);
112 D num = (p2 - p1).cross(v2);
113 D t = num / den;
114 return p1 + v1.scale(t);
115 }
116
117
118
119
120 template<class T>
121 struct ray {
122 seg<T> s;
123 bool rayl, rayr;
124 ray() {}
125 ray(seg<T> s, bool rayl, bool rayr): s(s), rayl(rayl), rayr(rayr) {}
126 ray operator-() const {
127 return ray(-s, rayl, rayr);
128 }
129 ray swap() {
130 return ray(s.swap(), rayr, rayl);
131 }
132 pt<T> vec() {
133 return s.vec();
134 }
135 };
136
137 typedef ray<D> rayD;
138
139 bool box_contains(rayD r, ptD p) {
140 segD s = r.s;
141 assert(s.a.x < s.b.x);
142 if (r.rayl) {
143 if (r.rayr) {
144 return true;
145 } else {
146 return p.x <= s.b.x+BEPS && between(INF * (s.a.y - s.b.y), p.y, s.b.y);
147 }
148 }

```

```

148 } else {
149 if (r.rayr) {
150 return s.a.x-BEPS <= p.x && between(s.a.y, p.y, INF * (s.b.y - s.a.y));
151 } else {
152 return s.a.x-BEPS <= p.x && p.x <= s.b.x+BEPS && between(s.a.y, p.y, s.b.y
153);
154 }
155 }
156
157
158
159
160
161 bool slopecomp(segD& a, segD& b) {
162 return a.vec().cross(b.vec()) < 0;
163 }
164 bool is_hill(segD a, segD b, segD c) {
165 return line_intersect(a, b).x < line_intersect(b, c).x - EPS;
166 }
167 void hull(vector<segD>& s) {
168 // lower hull
169 sort(s.begin(), s.end(), slopecomp);
170 vector<segD> hull;
171 for (int i = 0; i < s.size(); i++) {
172 while (hull.size() >= 2 && !is_hill(hull[hull.size()-2], hull.back(), s[i]))
173 {
174 hull.pop_back();
175 }
176 hull.push_back(s[i]);
177 }
178 // segmentify
179 for (int i = 0, j = 1; j < hull.size(); i++, j++) {
180 ptD p = line_intersect(hull[i], hull[j]);
181 hull[i] = i == 0 ? segD(p - normalize(hull[i].vec()), p) : segD(hull[i].a, p)
182 ;
183 hull[j] = segD(p, p + normalize(hull[j].vec()));
184 }
185 // cleanse & replace
186 s.clear();
187 for (int i = 0; i < hull.size(); i++) {
188 if (i == 0 || i == hull.size()-1 || hull[i].a.x < hull[i].b.x - BEPS) {
189 s.push_back(hull[i]);
190 }
191 }
192 }
193
194
195
196
197 rayD res;

```

```

198 bool chip_to(rayD r, ptD p) {
199 if (r.rayl) {
200 res = rayD(segD(p - normalize(r.vec()), p), true, false);
201 return true;
202 }
203 if (r.s.a.x < p.x) {
204 res = rayD(segD(r.s.a, p), false, false);
205 return true;
206 }
207 return false;
208 }
209
210 void tie(vector<rayD>& s, vector<rayD>& t) {
211 for (int i = 0; i < s.size(); i++) assert(s[i].s.a.x < s[i].s.b.x);
212 for (int i = 0; i < t.size(); i++) assert(t[i].s.a.x < t[i].s.b.x);
213 while (!s.empty() && !t.empty()) {
214 rayD sb = s.back(); s.pop_back();
215 rayD tb = t.back(); t.pop_back();
216 if (fabs(normalize(sb.vec()).cross(normalize(tb.vec()))) < EPS) continue; //
 parallel
217 ptD p = line_intersect(sb.s, tb.s);
218 bool scont = box_contains(sb, p);
219 bool tcont = box_contains(tb, p);
220 if (scont && tcont) {
221 if (chip_to(sb, p)) s.push_back(res);
222 if (chip_to(tb, p)) t.push_back(res);
223 break;
224 }
225 if (scont) s.push_back(sb);
226 if (tcont) t.push_back(tb);
227 }
228 if (s.empty() || t.empty()) {
229 s.clear();
230 t.clear();
231 }
232 }
233
234
235
236 D area(vector<segD>& s) {
237 D A = 0;
238 for (int i = 0; i < s.size(); i++) A += s[i].area();
239 return A;
240 }
241
242
243
244
245 int n;
246 ptL ptLs[N];
247 ptD ptDs[N];
248 segL segLs[N];
249 segD segDs[N];

```

```

250 L planecomp(segL a, segL b) {
251 return a.vec().cross(b.a - a.a);
252 }
253 bool dircomp(segL a, segL b) {
254 ptL da = a.vec();
255 ptL db = b.vec();
256 int qa = da.quad();
257 int qb = db.quad();
258 if (qa != qb) return qa < qb;
259 return db.cross(da) < 0;
260 }
261 void compute_segs() {
262 // removes unneeded half planes
263 for (int i = 0, j = n-1; i < n; j = i++) {
264 segL s = segL(ptLs[j], ptLs[i]);
265 segLs[i] = s;
266 }
267 // bounds
268 #define BOUND 10000000
269 segLs[n++] = segL(ptL(-(BOUND+11),0),ptL(-BOUND,BOUND));
270 segLs[n++] = segL(ptL(-BOUND,BOUND),ptL(BOUND,BOUND));
271 segLs[n++] = segL(ptL(BOUND,BOUND),ptL((BOUND+11),0));
272 segLs[n++] = segL(ptL((BOUND+11),0),ptL(BOUND,-BOUND));
273 segLs[n++] = segL(ptL(BOUND,-BOUND),ptL(-BOUND,-BOUND));
274 segLs[n++] = segL(ptL(-BOUND,-BOUND),ptL(-(BOUND+11),0));
275
276 sort(segLs, segLs + n, dircomp);
277
278 int m = 1;
279 for (int i = 1; i < n; i++) {
280 if (fabs(normalize(segLs[i].vec()).cross(normalize(segLs[m-1].vec())) <
281 0.01) {
282 L comp = planecomp(segLs[m-1], segLs[i]);
283 if (comp < 0) {
284 segLs[m-1] = segLs[i];
285 }
286 } else {
287 segLs[m++] = segLs[i];
288 }
289 n = m;
290 if (fabs(normalize(segLs[0].vec()).cross(normalize(segLs[n-1].vec())) < 0.01) {
291 L comp = planecomp(segLs[0], segLs[n-1]);
292 if (comp < 0) {
293 segLs[0] = segLs[n-1];
294 }
295 n--;
296 }
297 }
298 void rot() {
299 // removes vertical lines
300 D s = 1e-9;
301 D c = sqrt(1 - s*s);

```

```

302 for (int i = 0; i < n; i++) {
303 segDs[i].a.x = segLs[i].a.x * c - segLs[i].a.y * s;
304 segDs[i].a.y = segLs[i].a.x * s + segLs[i].a.y * c;
305 segDs[i].b.x = segLs[i].b.x * c - segLs[i].b.y * s;
306 segDs[i].b.y = segLs[i].b.x * s + segLs[i].b.y * c;
307 }
308 }
309
310
311
312 D solve() {
313 // preprocess
314 compute_segs();
315 rot();
316
317 // split
318 vector<segD> tops, bots;
319 for (int i = 0; i < n; i++) {
320 (segDs[i].a.x < segDs[i].b.x ? tops : bots).push_back(segDs[i]);
321 }
322
323 // hull
324 #define negall(s) do { for (int i = 0; i < s.size(); i++) s[i] = -s[i]; } while
325 (0)
326 #define swapall(s) do { for (int i = 0; i < s.size(); i++) s[i] = s[i].swap(); }
327 while (0)
328 #define ROT(x) do { reverse((x).begin(), (x).end()); swapall(x); negall(x); }
329 while (0)
330 hull(tops);
331 negall(bots);
332 hull(bots);
333 ROT(bots);
334
335 // combine
336 vector<rayD> rtops, rbots;
337 for (int i = 0; i < tops.size(); i++) rtops.push_back(rayD(tops[i], i == 0, i ==
338 tops.size()-1));
339 for (int i = 0; i < bots.size(); i++) rbots.push_back(rayD(bots[i], i == 0, i ==
340 bots.size()-1));
341 tie(rtops, rbots); ROT(rtops); ROT(rbots);
342 tie(rtops, rbots); ROT(rtops); ROT(rbots);
343 tops.clear();
344 bots.clear();
345 for (int i = 0; i < rtops.size(); i++) tops.push_back(rtops[i].s);
346 for (int i = 0; i < rbots.size(); i++) bots.push_back(rbots[i].s);
347
348 // normalize
349 ptD ave = ptD(0, 0);
350 for (int i = 0; i < tops.size(); i++) ave = ave + tops[i].a + tops[i].b;
351 for (int i = 0; i < bots.size(); i++) ave = ave + bots[i].a + bots[i].b;
352 ave = ave.scale(1./(2*(tops.size() + bots.size())));
353 for (int i = 0; i < tops.size(); i++) tops[i] = segD(tops[i].a - ave, tops[i].b
354 - ave);

```

```

349 for (int i = 0; i < bots.size(); i++) bots[i] = segD(bots[i].a - ave, bots[i].b
 - ave);
350
351 // area & proportion
352 return (area(bots) - area(tops)) * 0.5 / 4e14;
353 }
354
355
356
357 int main() {
358 int z;
359 scanf("%d", &z);
360 while (z--) {
361 scanf("%d", &n);
362 for (int i = 0; i < n; i++) {
363 scanf("%lld%lld", &ptls[i].x, &ptls[i].y);
364 }
365 printf("%.20Lf\n", solve());
366 }
367 }

```

## 2.14 不知道分在哪一类

### 2.14.1 经典例题

- 1 2020ZJ省赛 H-Huge Clouds
- 2 题意:
- 3 在二维平面上给定 $n$ 个点 $m$ 条线段, 定义在 $x$ 轴上一个点 $u$ , 如果存在一个点 $u$ 和某个给定点 $v$ 的连线 and 所有  
给定线段都不相交 (包括端点), 则 $u$ 不在阴影中。问 $x$ 轴上阴影的长度 ( $>1e9$ 则输出 $-1$ )。
- 4  $n \leq 500$ ,  $m \leq 500$ ,  $-1e4 \leq x \leq 1e4$ ,  $1 \leq y \leq 1e4$
- 5 思路:
- 6 首先 $O(n^2)$ 对每个点预处理出被线段遮挡的范围, 产生的阴影是一条线段或者一个射线, 因为题目规定超  
过 $1e9$ 就输出 $-1$ , 所以我们可以把射线的边界设为 $+-INF$ , 一个点被遮挡的范围就是它所有线段遮挡产  
生的线段和射线的并。
- 7 求出每个点遮挡范围之后, 对所有点的遮挡范围取交集, 交集的长度就是答案, 关键就是如何求线段并  
和线段集合的交
- 8
- 9 cf 598C
- 10 题意:
- 11 给你 $n$ 个源点在 $(0,0)$ , 终点在 $(x,y)$ 的向量, 问你哪两个向量之间的夹角最小?
- 12  $2 \leq n \leq 100000$ ,  $|x|, |y| \leq 10000$ ,  $x*x+y*y > 0$
- 13 GNU G++14 6.4.0
- 14 思路:
- 15 用 $\text{atan2}$ 函数算出每条边的角度 (弧度制), 然后排序, 然后 $O(n)$ 遍历每条相邻的边。
- 16 毒瘤题, 一定要`long double`才能过的了。

### 2.14.2 cf 598C

```

1 #include <bits/stdc++.h>
2 #define double long double //double->long double
3 using namespace std;
4 vector<pair<double,int> > a;

```



```

5 int n,ansx,ansy;
6 double sum=1e5;
7 int main(){
8 scanf("%d",&n);
9 for(int i=0;i<n;i++){
10 int x,y;
11 scanf("%d%d",&x,&y);
12 a.push_back({atan2(y,x),i+1});
13 }
14 sort(a.begin(),a.end());
15 a.push_back(a[0]);
16 for(int i=0;i<n;i++){
17 double x=a[i+1].first-a[i].first;
18 if(x<0)x+=2*acos(-1);
19 if(x<sum)sum=x,ansx=a[i].second,ansy=a[i+1].second;
20 }
21 printf("%d %d",ansx,ansy);
22 return 0;
23 }

```

### 2.14.3 2020ZJ 省赛 H

```

1 https://codeforces.com/gym/102770
2 题意：
3 在二维平面上给定n个点m条线段，定义在x轴上一个点u，如果存在一个点u和某个给定点v的连线 and 所有
 给定线段都不相交（包括端点），则u不在阴影中。问x轴上阴影的长度（>1e9则输出-1）。
4 n<=500, m<=500, -1e4<=x<=1e4, 1<=y<=1e4
5 思路：
6 首先O(n^2)对每个点预处理出被线段遮挡的范围，产生的阴影是一条线段或者一个射线，因为题目规定超
 过1e9就输出-1，所以我们可以把射线的边界设为+-INF，一个点被遮挡的范围就是它所有线段遮挡产
 生的线段和射线的并。
7 求出每个点遮挡范围之后，对所有点的遮挡范围取交集，交集的长度就是答案，关键就是如何求线段并
 和线段集合的交
8
9 #include <bits/stdc++.h>
10 using namespace std;
11 const int N=510;
12 const double eps = 1e-3;
13 const double INF=1e15;
14 int sgn(double x) {
15 if(fabs(x) < eps) return 0;
16 if(x < 0) return -1;
17 return 1;
18 }
19 struct Point {
20 double x,y;
21 Point() {}
22 Point(double _x,double _y) {
23 x = _x;
24 y = _y;
25 }
26 Point operator -(const Point &b)const {

```

```

27 return Point(x - b.x,y - b.y);
28 }
29 Point operator +(const Point &b)const {
30 return Point(x + b.x,y +b.y);
31 }
32 double operator ^(const Point &b)const {
33 return x*b.y - y*b.x;
34 }
35 double operator *(const Point &b)const {
36 return x*b.x + y*b.y;
37 }
38 }p[N];
39 struct Line{
40 Point s,e;
41 Line(){}
42 Line(Point _s, Point _e) {
43 s = _s;
44 e = _e;
45 }
46 bool pointonseg(Point p) {
47 return sgn((p - s) ^ (e-s)) == 0 && sgn((p - s) * (p - e)) <= 0;
48 }
49 Point crosspoint(Line v) {
50 double a1 = (v.e - v.s) ^ (s - v.s);
51 double a2 = (v.e - v.s) ^ (e - v.s);
52 return Point((s.x * a2 - e.x * a1) / (a2 - a1),
53 (s.y * a2 - e.y * a1) / (a2 - a1));
54 }
55 int relation(Point p) {
56 int c = sgn((p - s) ^ (e - s));
57 if (c < 0) return 1;
58 else if (c > 0) return 2;
59 else return 3;
60 }
61 }l[N];
62 struct seg{
63 double l,r;
64 };
65 seg getShadow(Point a,Line b){
66 if(b.pointonseg(a)) return {-INF,INF}; //因为这里要看的是点和线段的关系，不能用看点
 和直线关系的relation函数
67 if(b.s.y>=a.y && b.e.y>=a.y) return {0,0};
68 Point pp[2];
69 pp[0]=b.s; pp[1]=b.e;
70 if(pp[0].y>pp[1].y) swap(pp[0],pp[1]);
71 if(a.y>pp[1].y){
72 Line lONE=Line(a,pp[0]);
73 Line lTWO=Line(a,pp[1]);
74 Line lTHREE=Line(Point(-1,0),Point(1,0));
75 Point pONE=lONE.crosspoint(lTHREE);
76 Point pTWO=lTWO.crosspoint(lTHREE);
77 if(pONE.x>pTWO.x) return {pTWO.x,pONE.x};
78 else return {pONE.x,pTWO.x};

```

```

79 }
80 if(a.y>pp[0].y){
81 Line lONE=Line(a,pp[0]);
82 Line lTHREE=Line(Point(0,0),Point(1000,0));
83 Point pONE=lONE.crosspoint(lTHREE);
84 if(b.s.y<b.e.y) swap(b.s,b.e);
85 if(b.relation(a)==1){
86 return {-INF,pONE.x};
87 } else if(b.relation(a)==2){
88 return {pONE.x,INF};
89 }
90 }
91 }
92 bool cmp(seg a,seg b){
93 return a.l<b.l;
94 }
95 int main () {
96 int tc;
97 scanf("%d",&tc);
98 while(tc--){
99 int n,m;
100 scanf("%d%d",&n,&m);
101 for(int i=1;i<=n;i++){
102 scanf("%lf%lf",&p[i].x,&p[i].y);
103 }
104 for(int i=1;i<=m;i++){
105 scanf("%lf%lf%lf%lf",&l[i].s.x,&l[i].s.y,&l[i].e.x,&l[i].e.y);
106 }
107 map<double,int>m2;
108 for(int i=1;i<=n;i++){
109 map<double,int>m1;
110 for(int j=1;j<=m;j++){
111 seg temp=getShadow(p[i],l[j]);
112 m1[temp.l]++;
113 m1[temp.r]--;
114 }
115 int temp=0,flag=0;
116 double pre=0;
117 for(auto x:m1){ //差分求线段并
118 temp+=x.second;
119 if(temp>0 && !flag){
120 pre=x.first;
121 flag=1;
122 }
123 else if(temp==0 && flag){
124 m2[pre]++;
125 m2[x.first]--;
126 flag=0;
127 }
128 }
129 }
130 int temp=0,flag=0;
131 double pre=0,ans=0;

```

```

132 for(auto x:m2){ //差分求线段集合的交
133 temp+=x.second;
134 if(temp==n && !flag){
135 pre=x.first;
136 flag=1;
137 }
138 else if(temp<n && flag){
139 ans+=x.first-pre;
140 flag=0;
141 }
142 }
143 if(ans>1e9) puts("-1");
144 else printf("%.5f\n",ans);
145 }
146 }

```

### 3 数据结构

#### 3.1 线段树

##### 3.1.1 线段树 \_\_ 区间合并 hotel

```

1 #include <bits/stdc++.h>
2 #define ms(x, n) memset(x,n,sizeof(x));
3 #define forn(i, n) for(int i = 0; i < (int)n; i++)
4 #define For(i, a, b) for(int i = (a); i <= (int)(b); i++)
5 #define INF 0x3f3f3f3f
6 #define PI acos(-1.0)
7 #define mod(x) (x % 1000003)
8 typedef long long int ll;
9 typedef unsigned long long int ull;
10 using namespace std;
11
12 #define maxn 51000
13 #define lson l,mid,rt<<1
14 #define rson mid+1,r,rt<<1|1
15 int n, m, a[maxn], sum[maxn << 2], lsum[maxn << 2], rsum[maxn << 2], cover[maxn <<
 2], op, o, p;
16
17 void build_tree(int l, int r, int rt) {
18 sum[rt] = lsum[rt] = rsum[rt] = r - l + 1;
19 cover[rt] = -1;
20 if (l == r) return;
21 int mid = (l + r) >> 1;
22 build_tree(lson);
23 build_tree(rson);
24 }
25 void pushdown(int rt, int len) {
26 if (cover[rt] != -1) {
27 cover[rt << 1] = cover[rt << 1 | 1] = cover[rt];
28 lsum[rt << 1] = rsum[rt << 1] = sum[rt << 1] = cover[rt] ? 0 : len - (len >>
 1);

```

```

29 lsum[rt << 1 | 1] = rsum[rt << 1 | 1] = sum[rt << 1 | 1] = cover[rt] ? 0 : (
 len >> 1);
30 cover[rt] = -1;
31 }
32 }
33 void pushup(int rt, int len) {
34 lsum[rt] = lsum[rt << 1];
35 rsum[rt] = rsum[rt << 1 | 1];
36 if (lsum[rt] == len - (len >> 1)) {
37 lsum[rt] += lsum[rt << 1 | 1];
38 }
39 if (rsum[rt] == (len >> 1)) {
40 rsum[rt] += rsum[rt << 1];
41 }
42 sum[rt] = max(lsum[rt << 1 | 1] + rsum[rt << 1], max(sum[rt << 1], sum[rt << 1 |
 1]));
43 }
44 void update_tree(int l, int r, int rt, int sig, int L, int R) {
45 if (L <= l && r <= R) {
46 sum[rt] = lsum[rt] = rsum[rt] = sig ? 0 : r - l + 1;
47 cover[rt] = sig;
48 return;
49 }
50 pushdown(rt, r - l + 1);
51 int mid = (l + r) >> 1;
52 if (L <= mid) {
53 update_tree(lson, sig, L, R);
54 }
55 if (mid < R) {
56 update_tree(rson, sig, L, R);
57 }
58 pushup(rt, r - l + 1);
59 }
60 int query_tree(int l, int r, int rt, int w) {
61 if (l == r) return l;
62 pushdown(rt, r - l + 1);
63 int mid = (l + r) >> 1;
64 if (sum[rt << 1] >= w) {
65 return query_tree(lson, w);
66 } else if (rsum[rt << 1] + lsum[rt << 1 | 1] >= w) {
67 return mid - rsum[rt << 1] + 1;
68 } else {
69 return query_tree(rson, w);
70 }
71 }
72
73 int main()
74 {
75 ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
76 cin >> n >> m;
77 build_tree(1, n, 1);
78 while (m--) {
79 cin >> op;

```

```

80 if (op == 1) { //check in
81 cin >> o;
82 if (o > sum[1]) {
83 cout << 0 << endl;
84 } else {
85 int pos = query_tree(1, n, 1, o);
86 cout << pos << endl;
87 update_tree(1, n, 1, 1, pos, pos + o - 1);
88 }
89 } else if (op == 2) { //check out
90 cin >> o >> p;
91 update_tree(1, n, 1, 0, o, o + p - 1);
92 }
93 }
94 return 0;
95 }

```

## 3.2 树状数组

### 3.2.1 逆序对

```

1 // 兔子的逆序对 https://ac.nowcoder.com/acm/problem/20861
2 // 每交换一对数字，就会改变数组逆序对个数的奇偶性
3 const int maxn = (int)1e5 + 100;
4 const int cst = (int)1e5;
5 int n, m, a[maxn], b[maxn], ft[maxn];
6
7 void init() {
8 memset(ft, 0, sizeof(ft));
9 }
10 int query(int x) {
11 ll res = 0;
12 for (int i = x; i >= 1; i -= lowbit(i)) res += ft[i];
13 return res;
14 }
15 void update(int x, int val) {
16 for (int i = x; i <= cst; i += lowbit(i)) ft[i] += val;
17 }
18
19 int main() {
20 init();
21 cin >> n;
22 for (int i = 1; i <= n; ++i) cin >> a[i];
23 ll sum = 0;
24 for (int i = 1; i <= n; ++i) {
25 sum += query(cst) - query(a[i]);
26 update(a[i], 1);
27 }
28 bool odd = (sum & 1);
29 cin >> m;
30 for (int i = 0; i < m; ++i) {
31 int l, r;

```

```

32 cin >> l >> r;
33 int len = r - l + 1;
34 len /= 2;
35 if (len & 1) {
36 if (odd) odd = false;
37 else odd = true;
38 }
39 cout << (odd ? "dislike" : "like") << '\n';
40 }
41 return 0;
42 }

```

### 3.3 主席树

#### 3.3.1 静态查询区间第 k 大

```

1 //P 3834
2 const int maxn = 2*(int)1e5+1000;
3 int n, m, a[maxn], b[maxn], tVal[maxn*40], lt[maxn*40], rt[maxn*40], tot
 ;
4
5 int build(int l, int r) {
6 int node = ++tot;
7 tVal[node] = 0;
8 int mid = (l + r) >> 1;
9 if (l < r) {
10 lt[node] = build(l, mid);
11 rt[node] = build(mid + 1, r);
12 }
13 return node;
14 }
15 int update(int l, int r, int par, int p) {
16 int node = ++tot;
17 lt[node] = lt[par]; rt[node] = rt[par]; tVal[node] = tVal[par] + 1;
18 int mid = (l + r) >> 1;
19 if (l < r) {
20 if (p <= mid) lt[node] = update(l, mid, lt[par], p); //lt[par]!!
21 else rt[node] = update(mid + 1, r, rt[par], p); //rt[par]!!
22 }
23 return node; //node!!
24 }
25 int query(int n1, int n2, int l, int r, int k) {
26 if (l == r) return l;
27 int mid = (l + r) >> 1;
28 int sum = tVal[lt[n2]] - tVal[lt[n1]];
29 if (sum >= k) return query(lt[n1], lt[n2], l, mid, k);
30 else return query(rt[n1], rt[n2], mid + 1, r, k - sum);
31 }
32 void init() {
33 tot = 0;
34 }
35

```

```

36 int main() {
37
38 init(); //初始化
39 cin >> n >> m;
40 for (int i = 1; i <= n; ++i) {
41 cin >> a[i];
42 b[i] = a[i];
43 }
44 sort(b + 1, b + 1 + n);
45 int len = unique(b + 1, b + 1 + n) - b - 1;
46 t[0] = build(1, len);
47 for (int i = 1; i <= n; ++i) {
48 int p = lower_bound(b + 1, b + 1 + len, a[i]) - b;
49 t[i] = update(1, len, t[i - 1], p);
50 }
51 for (int i = 0; i < m; ++i) {
52 int l, r, k;
53 cin >> l >> r >> k;
54 int p = query(t[l - 1], t[r], 1, len, k);
55 cout << b[p] << '\n';
56 }
57 return 0;
58 }

```

### 3.3.2 动态查询区间第 k 大

```

1 //p2617
2 #define lowbit(x) (x&(-x))
3 const int maxn = (int)1e5+100;
4 int n, m, a[maxn], b[maxn<<1], len; //len: 离散化后数组的长度
5 int t[maxn*60], lt[maxn*360], rt[maxn*360], tVal[maxn*360], tot; //tot: 动态开点
6 int n1, n2, t1[maxn], t2[maxn];
7 struct qq {
8 int l, r, ra, val;
9 char op;
10 }q[maxn];
11
12 void update(int &node, int l, int r, int p, int val) {
13 if (!node) node = ++tot;
14 tVal[node] += val;
15 int mid = (l + r) >> 1;
16 if (l == r) return;
17 if (p <= mid) update(lt[node], l, mid, p, val);
18 else update(rt[node], mid + 1, r, p, val);
19 }
20 void change(int idx, int val) {
21 int p = lower_bound(b + 1, b + 1 + len, a[idx]) - b;
22 for (int i = idx; i <= n; i += lowbit(i)) update(t[i], 1, len, p, val);
23 }
24 int kTh(int l, int r, int k) {
25 if (l == r) return l;
26 int sum = 0;

```



```

27 for (int i = 0; i < n1; ++i) sum -= tVal[lt[t1[i]]];
28 for (int i = 0; i < n2; ++i) sum += tVal[lt[t2[i]]]; //lt[t1[i]]!!!
29 int mid = (l + r) >> 1;
30 if (sum >= k) {
31 for (int i = 0; i < n1; ++i) t1[i] = lt[t1[i]];
32 for (int i = 0; i < n2; ++i) t2[i] = lt[t2[i]];
33 return kTh(l, mid, k);
34 } else {
35 for (int i = 0; i < n1; ++i) t1[i] = rt[t1[i]];
36 for (int i = 0; i < n2; ++i) t2[i] = rt[t2[i]];
37 return kTh(mid + 1, r, k - sum);
38 }
39 }
40 int kPre(int l, int r, int k) { //l, r
41 n1 = 0, n2 = 0;
42 for (int i = l - 1; i >= 1; i -= lowbit(i)) t1[n1++] = t[i];
43 for (int i = r; i >= 1; i -= lowbit(i)) t2[n2++] = t[i];
44 return kTh(1, len, k); //1, len !!
45 }
46 void init() {
47 memset(t, 0, sizeof(t));
48 memset(lt, 0, sizeof(lt));
49 memset(rt, 0, sizeof(rt));
50 memset(tVal, 0, sizeof(tVal));
51 tot = 0; //这个0蛮有学问的，你后面查询logn裸树的时候，万一你要找的那个地方没有
52 //就变成空节点，对结果没影响，你赋值成当前仅当-1的时候会有问题
53 //因为0节点并不是空节点，而是你开的第一个点 :->
54 }
55
56 int main() {
57 init(); //初始化
58 int cnt = 0;
59 cin >> n >> m;
60 for (int i = 1; i <= n; ++i) {
61 cin >> a[i];
62 b[++cnt] = a[i];
63 }
64 for (int i = 0; i < m; ++i) {
65 cin >> q[i].op;
66 if (q[i].op == 'Q') cin >> q[i].l >> q[i].r >> q[i].ra;
67 else {
68 cin >> q[i].l >> q[i].val;
69 b[++cnt] = q[i].val;
70 }
71 }
72 sort(b + 1, b + 1 + cnt);
73 len = unique(b + 1, b + 1 + cnt) - b - 1;
74 for (int i = 1; i <= n; ++i) change(i, 1);
75
76 for (int i = 0; i < m; ++i) {
77 if (q[i].op == 'C') {
78 change(q[i].l, -1);
79 a[q[i].l] = q[i].val;

```

```

80 change(q[i].l, 1);
81 } else {
82 cout << b[kPre(q[i].l, q[i].r, q[i].ra)] << '\n';
83 }
84 }
85 return 0;
86 }

```

### 3.4 字典树

#### 3.4.1 trie 树

```

1 //UVA644 判断输入的多组字符串中有没有一个串是另一个串的子串
2 const int maxn = 500100;
3 int trie[maxn][26], sum[maxn], ed[maxn], root, len, tot;
4 char s[maxn];
5 bool flag;
6
7 void Insert(char s[]) {
8 root = 0;
9 len = strlen(s);
10 for (int i = 0; i < len; ++i) {
11 int id = int(s[i] - 'a');
12 if (!trie[root][id]) trie[root][id] = ++tot;
13 /*if (ed[trie[root][id]]) {
14 flag = true;
15 }
16 if (i == len-1) {
17 ++ed[trie[root][id]];
18 if (sum[trie[root][id]]) {
19 flag = true;
20 }
21 }*/
22 sum[trie[root][id]]++;
23 root = trie[root][id];
24 }
25 }
26 int Find(char s[]) {
27 root = 0;
28 len = strlen(s);
29 for (int i = 0; i < len; ++i) {
30 int id = int(s[i] - 'a');
31 if (!trie[root][id]) return 0;
32 root = trie[root][id];
33 }
34 return sum[root];
35 }
36 void init() { //多组数据慎用, 要么把字典树开小一点
37 memset(trie, 0, sizeof(trie));
38 memset(sum, 0, sizeof(sum));
39 memset(ed, 0, sizeof(ed));
40 tot = 0;

```

```

41 flag = false;
42 }
43
44 int main() {
45 int kase = 1;
46 init();
47 while (~scanf("%s", s)) {
48 if (strlen(s) == 1 && s[0] == '9') {
49 if (flag == false) printf("Set %d is immediately decodable\n", kase++);
50 else printf("Set %d is not immediately decodable\n", kase++);
51 init();
52 } else {
53 Insert(s);
54 }
55 }
56 return 0;
57 }

```

## 4 字符串

### 4.1 KMP

```

1 const int maxn = 1000000 + 1000;
2 int t, nxt[maxn];
3 char x[maxn], y[maxn];
4
5 //ps:2019ccpc秦皇岛的那题爆longlong了 XD
6
7 void kmp_pre(char x[], int m, int nxt[]) {
8 int i, j;
9 j = nxt[0] = -1;
10 i = 0;
11 while (i <= m) { //求最小循环节:i <= m 最小循环节:m - nxt[m] 周期性字符串m % (m -
12 //nxt[m]) == 0
13 while (-1 != j && x[i] != x[j]) j = nxt[j];
14 nxt[++i] = ++j;
15 }
16 }
17
18 int kmp_count(char x[], int m, char y[], int n) {
19 int i, j;
20 int ans = 0;
21 kmp_pre(x, m, nxt);
22 i = j = 0;
23 while (i < n) {
24 while (-1 != j && y[i] != x[j]) j = nxt[j];
25 ++i;
26 ++j;
27 if (j >= m) {
28 ++ans;
29 j = nxt[j];
30 }
31 }
32 }

```

```

30 return ans;
31 }
32
33 int main() {
34
35 cin >> t;
36 while (t--) {
37 cin >> x;
38 cin >> y;
39 cout << kmp_count(x, strlen(x), y, strlen(y)) << '\n';
40 }
41
42 return 0;
43 }

```

## 4.2 exKMP

```

1 const int maxn = 50000 + 1000;
2 int nxt[maxn], extend[maxn];
3 char x[maxn], y[maxn];
4 //x是模式串, y是主串
5 //(0~m-1)是有效部分, 和kmp不同
6 void pre_EKMP(char x[], int m, int nxt[]) {
7 nxt[0] = m;
8 int j = 0;
9 while (j + 1 < m && x[j] == x[j + 1]) ++j;
10 nxt[1] = j;
11 int k = 1;
12 for (int i = 2; i < m; ++i) {
13 int p = nxt[k] + k - 1;
14 int L = nxt[i - k];
15 if (i + L < p + 1) nxt[i] = L;
16 else {
17 j = max(0, p - i + 1);
18 while (i + j < m && x[i + j] == x[j]) ++j;
19 nxt[i] = j;
20 k = i;
21 }
22 }
23 }
24 void EKMP(char x[], int m, char y[], int n, int nxt[], int extend[]) {
25 pre_EKMP(x, m, nxt);
26 int j = 0;
27 while (j < n && j < m && x[j] == y[j]) ++j;
28 extend[0] = j;
29 int k = 0;
30 for (int i = 1; i < n; ++i) {
31 int p = extend[k] + k - 1;
32 int L = nxt[i - k];
33 if (i + L < p + 1) extend[i] = L;
34 else {
35 j = max(0, p - i + 1);

```

```

36 while (i + j < n && j < m && y[i + j] == x[j]) ++j;
37 extend[i] = j;
38 k = i;
39 }
40 }
41 }

```

### 4.3 Manacher

```

1 #include <bits/stdc++.h>
2 #define maxn 2000005
3 using namespace std;
4 int mp[maxn];
5 string str;
6 char c[maxn];
7 void Manacher(string s,int len){
8 int l=0,R=0,C=0;;
9 c[l++]='$', c[l++]='#';
10 for(int i=0;i<len;i++){
11 c[l++]=s[i], c[l++]='#';
12 }
13 for(int i=0;i<l;i++){
14 mp[i]=R>i?min(mp[2*C-i],R-i):1;
15 while(i+mp[i]<l&&i-mp[i]>0){
16 if(c[i+mp[i]]==c[i-mp[i]]) mp[i]++;
17 else break;
18 }
19 if(i+mp[i]>R){
20 R=i+mp[i], C=i;
21 }
22 }
23 }
24 int main()
25 {
26 int cnt=0;
27 while(cin>>str){
28 if(str=="END") break;
29 int len=str.length();
30 Manacher(str,len);
31 int ans=0;
32 for(int i=0;i<2*len+4;i++){
33 ans=max(ans,mp[i]-1);
34 }
35 printf("Case %d: %d\n",++cnt,ans);
36 }
37 return 0;
38 }

```

## 5 dp

### 5.1 树形 dp

#### 5.1.1 树的重心

```
1 //poj 1655
2 //树的重心:
3 //若树上的一个节点满足其所有的子树中最大的子树节点数最少, 那么这个点就是这棵树的重心。
4 const int maxn = 20000+100;
5 int tc, n, sz[maxn], fa[maxn], res[maxn];
6 vector<int> G[maxn];
7
8 void init(int n) {
9 memset(fa, 0, sizeof(fa));
10 memset(sz, 0, sizeof(sz));
11 for (int i = 1; i <= n; ++i) G[i].clear();
12 }
13 void dfs(int x, int par) {
14 sz[x] = 1;
15 fa[x] = par;
16 for (int i = 0; i < int(G[x].size()); ++i) {
17 int to = G[x][i];
18 if (to == par) continue;
19 dfs(to, x);
20 sz[x] += sz[to];
21 }
22 }
23
24 int main() {
25
26 scanf("%d", &tc);
27 while (tc--) {
28 scanf("%d", &n);
29 init(n);
30 for (int i = 0; i < n - 1; ++i) {
31 int u, v;
32 scanf("%d%d", &u, &v);
33 G[u].push_back(v);
34 G[v].push_back(u);
35 }
36 dfs(1, 0);
37
38 for (int i = 1; i <= n; ++i) {
39 int maxx = n - sz[i];
40 for (int j = 0; j < int(G[i].size()); ++j) {
41 int to = G[i][j];
42 if (to == fa[i]) continue; ///!
43 maxx = max(maxx, sz[to]);
44 }
45 res[i] = maxx;
46 }
47 }
```

```

48 int ans = INT_MAX, node;
49 for (int i = 1; i <= n; ++i) {
50 if (res[i] < ans) {
51 ans = res[i];
52 node = i;
53 }
54 }
55
56 printf("%d %d\n", node, ans);
57 }
58
59 return 0;
60 }

```

### 5.1.2 树上最远距离

```

1 //hdu 2196
2 const int maxn = (int)1e4+100;
3 int n, f[maxn], g[maxn], fa[maxn];
4 vector< pair<int, int> > G[maxn];
5
6 void init(int n) {
7 memset(f, 0, sizeof(f));
8 memset(g, 0, sizeof(g));
9 memset(fa, 0, sizeof(fa));
10 for (int i = 1; i <= n; ++i) G[i].clear(); ///!
11 }
12 void dfs(int x, int par) {
13 fa[x] = par;
14 for (pair<int, int> pii : G[x]) {
15 int to = pii.first;
16 if (to == par) continue;
17 dfs(to, x);
18 f[x] = max(f[x], f[to] + pii.second);
19 }
20 }
21 void dfs2(int x, int par) {
22 int temp = 0;
23 g[x] = g[par];
24 for (pair<int, int> pii : G[par]) {
25 int to = pii.first;
26 if (to == fa[par]) continue;
27 if (to == x) temp = pii.second;
28 else {
29 g[x] = max(g[x], f[to] + pii.second);
30 }
31 }
32 g[x] += temp;
33 for (pair<int, int> pii : G[x]) {
34 int to = pii.first;
35 if (to == par) continue;
36 dfs2(to, x);

```

```
37 }
38 }
39
40 int main() {
41 while (cin >> n) {
42 init(n);
43 for (int i = 2; i <= n; ++i) {
44 int u, val;
45 cin >> u >> val;
46 G[i].emplace_back(make_pair(u, val));
47 G[u].emplace_back(make_pair(i, val));
48 }
49 dfs(1, 0);
50 dfs2(1, 0);
51 for (int i = 1; i <= n; ++i) {
52 cout << max(f[i], g[i]) << '\n';
53 }
54 }
55 return 0;
56 }
```

## 6 树上问题

### 6.1 树的直径

```
1 int n;
2 vvi g;
3 vi p;
4
5 pii dfs(int v, int par = -1, int dist = 0) {
6 p[v] = par;
7 pii res = mp(dist, v);
8 for (auto to : g[v]) {
9 if (to == par) continue;
10 res = max(res, dfs(to, v, dist + 1));
11 }
12 return res;
13 }
14
15 int main() {
16
17 cin >> n;
18 g = vvi(n);
19 p = vi(n);
20 forn(i, n - 1) {
21 int v1, v2;
22 cin >> v1 >> v2;
23 --v1; --v2;
24 g[v1].eb(v2);
25 g[v2].eb(v1);
26 }
27 pii da = dfs(0);
```



```

28 pii db = dfs(da.se);
29 vi diam;
30 int v = db.se;
31 while (v != -1) {
32 diam.eb(v);
33 v = p[v];
34 }
35 if (int(diam.size()) == n) { //直径就是整个图，也就是整个图就是一条链
36 cout << n - 1 << '\n';
37 cout << diam[0] + 1 << " " << diam[1] + 1 << " " << diam.back() + 1 << '\n';
38 return 0;
39 }
40 queue<int> q;
41 vi d(n, -1);
42 for (int u : diam) {
43 d[u] = 0;
44 q.push(u);
45 }
46 while (!q.empty()) {
47 int v = q.front();
48 q.pop();
49 for (auto to : g[v]) {
50 if (d[to] == -1) {
51 d[to] = d[v] + 1;
52 q.push(to);
53 }
54 }
55 }
56 pii mx = mp(d[0], 0);
57 forn(i, n) {
58 mx = max(mx, mp(d[i], i));
59 }
60 cout << int(diam.size()) - 1 + mx.fi << '\n';
61 cout << diam[0] + 1 << " " << diam.back() + 1 << " " << mx.se + 1 << '\n';
62 return 0;
63 }

```

## 7 STL

### 7.1 自定义排序

```

1 //区间长度由大到小排序，若长度相同，则按左端点坐标由小到大排序
2 //(multi)set和priority_queue都有empty()函数
3 struct node {
4 int l, r;
5 node(int _l, int _r) : l(_l), r(_r) {}
6 };
7 struct cmp {
8 bool operator () (node n1, node n2) const {
9 int l1 = n1.r - n1.l + 1;
10 int l2 = n2.r - n2.l + 1;
11 if (l1 == l2) return n1.l < n2.l;

```

```

12 return l1 > l2;
13 }
14 };
15 (multi)set<node, cmp> st;
16 /*****
17 struct node {
18 int l, r;
19 node(int _l, int _r) : l(_l), r(_r) {}
20 friend bool operator < (node n1, node n2) { //一个堆，越在顶端(队顶)的越大
21 int l1 = n1.r - n1.l + 1; //优先队列只能重载 < 号
22 int l2 = n2.r - n2.l + 1;
23 if (l1 == l2) return n1.l > n2.l;
24 return l1 < l2;
25 }
26 };
27 priority_queue<node> pp;

```

## 7.2 nth\_element

```

1 int a[n];
2 //求第k小的数
3 nth_element(a, a + k, a + n);
4 //求第k大的数
5 nth_element(a, a + k, a + n, greater<>());

```

# 8 其他问题

## 8.1 ST 表

### 8.1.1 ST 表

```

1 //初始化O(logn) 询问O(1)
2 //hdu 5443
3 //询问最小，把两个max改成min就行
4 const int maxn = 2000;
5 int dp[maxn][35], LOG[maxn]; //2^30 == 1e9
6 int tc, n, m, a[maxn];
7
8 void initRMQ() {
9 LOG[0] = -1;
10 for (int i = 1; i <= n; ++i) {
11 LOG[i] = ((i & (i-1)) == 0) ? LOG[i-1] + 1 : LOG[i-1];
12 dp[i][0] = a[i];
13 }
14 for (int j = 1; j <= LOG[n]; ++j) {
15 for (int i = 1; i + (1 << j) - 1 <= n; ++i) {
16 dp[i][j] = min(dp[i][j-1], dp[i + (1<<(j-1))][j-1]);
17 }
18 }
19 }
20 int rmqQuery(int x, int y) {

```

```

21 int k = LOG[y-x+1];
22 return max(dp[x][k], dp[y-(1<<k)+1][k]);
23 }

```

## 8.2 莫队

### 8.2.1 复杂度

- 1 带修改莫队的复杂度为 $O(n^{5/3})$
- 2 不带修改莫队复杂度为 $O(m \log m + n \sqrt{m})$
- 3 另外还要加上暴力求解的复杂度

### 8.2.2 普通莫队 LOJ 1188 $O(\sqrt{n} \times q)$

```

1 //复杂度 $O(\sqrt{n} \times q)$
2 #include <bits/stdc++.h>
3
4 using namespace std;
5
6 const int maxn = (int)1e5 + 1000;
7 int t, n, qu, cnt[maxn], res, a[maxn], blo, ans[maxn];
8 struct node {
9 int l, r, id;
10 }q[maxn];
11
12 int read() {
13 int ans = 0, f = 1; char c = getchar();
14 for (; c < '0' || c > '9'; c = getchar()) if (c == '-') f = -1;
15 for (; c >= '0' && c <= '9'; c = getchar()) ans = ans * 10 + c - '0';
16 return ans * f;
17 }
18 bool cmp(node a, node b) {
19 return (a.l / blo == b.l / blo ? (a.l / blo) & 1 ? a.r < b.r : a.r > b.r : a.l <
20 b.l);
21 }
22 void add(int pos) {
23 int num = a[pos];
24 if (cnt[num] == 0) ++res;
25 ++cnt[num];
26 }
27 void cut(int pos) {
28 int num = a[pos];
29 --cnt[num];
30 if (cnt[num] == 0) --res;
31 }
32 int main() {
33 ios::sync_with_stdio(false); cin.tie(0);
34 t = read();
35 for (int kase = 1; kase <= t; ++kase) {
36 memset(cnt, 0, sizeof(cnt));
37 n = read();

```

```

38 qu = read();
39 blo = sqrt(n * 1.0 * 2 / 3);
40 for (int i = 1; i <= n; ++i) {
41 a[i] = read();
42 }
43 for (int i = 0; i < qu; ++i) {
44 q[i].l = read();
45 q[i].r = read();
46 q[i].id = i;
47 }
48 sort(q, q + qu, cmp);
49 int l = 1, r = 0;
50 res = 0;
51 for (int i = 0; i < qu; ++i) {
52 while (r < q[i].r) add(++r);
53 while (l > q[i].l) add(--l);
54 while (r > q[i].r) cut(r--);
55 while (l < q[i].l) cut(l++);
56 ans[q[i].id] = res;
57 }
58 printf("Case %d:\n", kase);
59 for (int i = 0; i < qu; ++i) {
60 printf("%d\n", ans[i]);
61 }
62 }
63 return 0;
64 }

```

### 8.2.3 带修改莫队 cf 940F

```

1 #include <bits/stdc++.h>
2
3 #define mp make_pair
4 #define mt make_tuple
5 #define fi first
6 #define se second
7 #define pb push_back
8 #define all(x) (x).begin(), (x).end()
9 #define rall(x) (x).rbegin(), (x).rend()
10 #define forn(i, n) for (int i = 0; i < (int)(n); ++i)
11 #define for1(i, n) for (int i = 1; i <= (int)(n); ++i)
12 #define ford(i, n) for (int i = (int)(n) - 1; i >= 0; --i)
13 #define fore(i, a, b) for (int i = (int)(a); i <= (int)(b); ++i)
14
15 using namespace std;
16
17 typedef pair<int, int> pii;
18 typedef vector<int> vi;
19 typedef vector<pii> vpi;
20 typedef vector<vi> vvi;
21 typedef long long i64;
22 typedef vector<i64> vi64;

```

```

23 typedef vector<vi64> vvi64;
24 typedef pair<i64, i64> pi64;
25 typedef double ld;
26
27 template<class T> bool uin(T &a, T b) { return a > b ? (a = b, true) : false; }
28 template<class T> bool uax(T &a, T b) { return a < b ? (a = b, true) : false; }
29
30 const int maxn = (int)2 * 1e6 + 1000;
31 int a[maxn], b[maxn], c[maxn], n, m, qdx = 0, mdx = 0, bsz, res = 0;
32 int ans[maxn], times[maxn], cnt[maxn], rl[maxn], rr[maxn], d[maxn];
33 struct query {
34 int l, r, md, id;
35 void set(int _l, int _r, int _md, int _id) {
36 l = _l, r = _r, md = _md, id = _id;
37 }
38 bool operator < (const query &b) const {
39 if (l / bsz != b.l / bsz) return l < b.l;
40 if (r / bsz != b.r / bsz) return r < b.r;
41 return id < b.id;
42 }
43 } que[maxn];
44 struct modify {
45 int wz, x, y;
46 void set(int _wz, int _x, int _y) {
47 wz = _wz, x = _x, y = _y;
48 }
49 } mod[maxn];
50
51 int read() {
52 int ans = 0, f = 1; char c = getchar();
53 for(; c < '0' || c > '9'; c = getchar()) if (c == '-') f = -1;
54 for(; c >= '0' && c <= '9'; c = getchar()) ans = ans * 10 + c - '0';
55 return ans * f;
56 }
57 template <class T>
58 void write(T x) {
59 if (x < 0) x = -x, putchar('-');
60 if (x >= 10) write(x / 10);
61 putchar('0' + x % 10);
62 }
63 void add(int x) {
64 --cnt[times[x]];
65 ++times[x];
66 ++cnt[times[x]];
67 }
68 void cut(int x) {
69 --cnt[times[x]];
70 --times[x];
71 ++cnt[times[x]];
72 }
73 void upd(int l, int r, int t) {
74 if (l <= mod[t].wz && mod[t].wz <= r) {
75 cut(mod[t].x), add(mod[t].y);

```

```

76 }
77 d[mod[t].wz] = mod[t].y;
78 }
79 void del(int l, int r, int t) {
80 if (l <= mod[t].wz && mod[t].wz <= r) {
81 add(mod[t].x), cut(mod[t].y);
82 }
83 d[mod[t].wz] = mod[t].x;
84 }
85 int find_ans() {
86 int now = 1;
87 while (cnt[now] != 0) {
88 ++now;
89 }
90 return now;
91 }
92 void work() {
93 bsz = (int)pow(n, 2.0 / 3);
94 sort(que + 1, que + 1 + qdx);
95 int l = 1, r = 0, t = 0;
96 res = 0;
97 for1(i, qdx) {
98 while (t < que[i].md) ++t, upd(l, r, t);
99 while (t > que[i].md) del(l, r, t), --t;
100 while (l < que[i].l) cut(d[l]), ++l;
101 while (l > que[i].l) --l, add(d[l]);
102 while (r < que[i].r) ++r, add(d[r]);
103 while (r > que[i].r) cut(d[r]), --r;
104 ans[que[i].id] = find_ans();
105 }
106 }
107
108 int main() {
109 ios::sync_with_stdio(false);
110 cin.tie(nullptr);
111 cout.precision(10);
112 cout << fixed;
113 #ifdef LOCAL_DEFINE
114 freopen("in", "r", stdin);
115 #endif
116
117 memset(times, 0, sizeof(times));
118 memset(cnt, 0, sizeof(cnt));
119 n = read(); m = read();
120 for1(i, n) {
121 a[i] = read();
122 b[i] = a[i];
123 }
124 int tot = n;
125 forn(i, m) {
126 int op, l, r;
127 op = read(); l = read(); r = read();
128 if (op == 1)

```

```

129 ++qdx, que[qdx].set(1, r, mdx, qdx);
130 if (op == 2) {
131 ++mdx;
132 rl[mdx] = 1;
133 ++tot;
134 rr[mdx] = a[tot] = b[tot] = r;
135 }
136 }
137 sort(b + 1, b + 1 + tot);
138 int dig = unique(b + 1, b + 1 + tot) - b - 1;
139 for1(i, n) {
140 d[i] = lower_bound(b + 1, b + 1 + dig, a[i]) - b;
141 c[i] = d[i];
142 }
143 for1(i, mdx) {
144 int tmp = lower_bound(b + 1, b + 1 + dig, rr[i]) - b;
145 mod[i].set(rl[i], 0, tmp);
146 }
147 for1(i, mdx) {
148 mod[i].x = c[mod[i].wz];
149 c[mod[i].wz] = mod[i].y;
150 }
151 work();
152 for1(i, qdx) {
153 write(ans[i]);
154 puts("");
155 }
156
157
158 #ifdef LOCAL_DEFINE
159 cerr << "Time elapsed: " << 1.0 * clock() / CLOCKS_PER_SEC << " s.\n";
160 #endif
161 return 0;
162 }

```

### 8.3 二分注意点

- 1 1. `while(>=) +1 -1`
- 2 2. `double`的二分最好用`forn(i,xx)`, 当误差 $<1e-6$ , `l=0` && `r=1e6`, `xx`可以是200 (cf653D)

### 8.4 LIS

```

1 int LIS(int a[]) { //lis数组从0开始
2 int len = 0;
3 for1(i, n) {
4 int x = lower_bound(lis, lis+len, a[i]) - lis;
5 lis[x] = a[i];
6 len = max(len, x+1);
7 }
8 return len;
9 }

```

```

10 int LIS(int a[]) { //lis数组从1开始
11 int len = 0;
12 for1(i, n) {
13 int x = lower_bound(lis+1, lis+1+len, a[i])-lis;
14 lis[x] = a[i];
15 len = max(len, x);
16 }
17 return len;
18 }
19 lower_bound : a1 < a2 < ... < an
20 upper_bound : a1 <= a2 <= ..<= an

```

## 8.5 尺取法

```

1 //尺取法：反复推进区间的开头和末尾，来求满足条件的最小区间的方法被称作尺取法。
2 #include<bits/stdc++.h>
3 //题意：给你一个长度为n的数列，再给你一个数s，让你找出数列中连续元素和>=s的最短长度。
4 #define ll long long
5 using namespace std;
6 const ll maxn=(1l)1e5+100;
7 ll tc,n,s,a[maxn];
8 int main() {
9 cin>>tc;
10 while(tc--){
11 cin>>n>>s;
12 for(int i=1; i<=n; ++i) cin>>a[i];
13 ll ans=LLONG_MAX;
14 ll l=1,r=1,sum=0;
15 for(;;){
16 while(r<=n && sum<s){
17 sum+=a[r++];
18 }
19 if(sum<s) break;
20 ans=min(ans, r-l);
21 sum-=a[l++];
22 }
23 if(ans==LLONG_MAX) ans=0;
24 cout<<ans<<'\n';
25 }
26 return 0;
27 }

```

## 8.6 单调队列

```

1 //2020牛客多校第二场 F
2 #include <bits/stdc++.h>
3 using namespace std;
4 const int N=5050;
5 int n,m,k,a[N][N],b[N][N];
6 deque<int> dq;
7 int main() {

```



```

8 ios::sync_with_stdio(false);cin.tie(0);cout.precision(10);cout << fixed;
9 #ifdef LOCAL_DEFINE
10 freopen("input.txt", "r", stdin);
11 #endif
12 cin>>n>>m>>k;
13 memset(a, 0, sizeof(a));
14 memset(b, 0, sizeof(b));
15 for(int i=1; i<=n; ++i){
16 for(int j=1; j<=m; ++j){
17 if(!a[i][j]){
18 for(int k=1; k*i<=n && k*j<=m; ++k){
19 a[i*k][j*k]=k; b[i*k][j*k]=i*j*k;
20 }
21 }
22 }
23 }
24 //维护队首最大的单调队列
25 memset(a, 0, sizeof(a));
26 for(int i=1; i<=n; ++i){
27 while(!dq.empty()) dq.pop_front();
28 dq.push_back(0);
29 for(int j=1; j<=m; ++j){
30 while(!dq.empty() && j-dq.front()>=k) dq.pop_front();
31 while(!dq.empty() && b[i][j]>=b[i][dq.back()]) dq.pop_back(); //因为队首最
 大, 所以>=
32 dq.push_back(j);
33 a[i][j]=b[i][dq.front()];
34 }
35 }
36 long long ans=0;
37 for(int j=1; j<=m; ++j){
38 while(!dq.empty()) dq.pop_front();
39 dq.push_back(0);
40 for(int i=1; i<=n; ++i){
41 while(!dq.empty() && i-dq.front()>=k) dq.pop_front();
42 while(!dq.empty() && a[i][j]>=a[dq.back()][j]) dq.pop_back();
43 dq.push_back(i);
44 if(i>=k && j>=k) ans+=a[dq.front()][j];
45 }
46 }
47 cout<<ans<<'\n';
48 #ifdef LOCAL_DEFINE
49 cerr << "Time elapsed: " << 1.0 * clock() / CLOCKS_PER_SEC << " s.\n";
50 #endif
51 return 0;
52 }

```

## 8.7 一句话去重并生成新数组

```

1 sort(all(ans)); //一定要先排序
2 ans.resize(unique(all(ans)) - ans.begin());

```

## 8.8 输入一行看有多少个数

```

1 //结果存在op数组中(0 ~ n-1)
2 vector<int> op(maxn);
3
4 int input(){
5 string str, ss;
6 getline(cin, str);
7 if (str[0] == '-')
8 return -1;
9 istringstream s(str);
10 vector<int> v;
11 v.clear();
12 while(s >> ss){
13 int tmp = 0;
14 for (int i = 0; i < int(ss.size()); i++)
15 tmp = tmp * 10 + (ss[i] - '0');
16 v.push_back(tmp);
17 }
18 for (int i = 0; i < int(v.size()); i++)
19 op[i] = v[i];
20 return v.size();
21 }

```

## 8.9 最小（大）表示法

```

1 //s是两个s顺序拼接起来的字符串， len是原来一个s的长度， 返回的是起点的下标
2 // 传入的值s是个两个s， len是一个s
3 int min_string(char *s, int len) {
4 int i = 0, j = 1, k = 0;
5 while (i < len && j < len && k < len) {
6 if (s[i + k] == s[j + k]) ++k;
7 else if (s[i + k] < s[j + k]) j += k + 1, k = 0;
8 else if (s[i + k] > s[j + k]) i += k + 1, k = 0;
9 if (i == j) ++j;
10 }
11 return min(i, j);
12 }
13 int max_string(char *s, int len) {
14 int i = 0, j = 1, k = 0;
15 while (i < len && j < len && k < len) {
16 if (s[i + k] == s[j + k]) ++k;
17 else if (s[i + k] < s[j + k]) i += k + 1, k = 0;
18 else if (s[i + k] > s[j + k]) j += k + 1, k = 0;
19 if (i == j) ++j;
20 }
21 return min(i, j);
22 }

```

## 8.10 随机数

```

1 【随机函数】
2 使用mt19937而不是rand()
3 #include <chrono>
4 #include <random>
5 mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
6 ll ans = rng();
7 printf("%I64d\n",ans);
8 范围为0 - 4294967295 (2的32次方 - 1)
9
10 【随机生成整数】
11 int randInt(int l,int r){ //生成l到r的整数,l <= r
12 return (rng() % (r - l + 1)) + l;
13 }

```

## 8.11 输入日期输出周几

```

1 输入年月日，例如2010-08-15,就调用calc(2010,8,15)
2 输出0代表周日,1代表周一.....,6代表周日
3
4 int calc(int y,int m,int d)
5 {
6 if(m==1 || m==2)
7 {
8 m+=12;
9 --y;
10 }
11 int w=(d+1+2*m+3*(m+1)/5+y+y/4-y/100+y/400)%7;
12 return w;
13 }

```

# 9 黑科技

## 9.1 IO

### 9.1.1 快读模板

```

1 //读不了浮点数
2 //前面四个快读效率都差不多 快写比printf快一点
3 int read() {
4 int x=0, f=1; char ch=getchar();
5 while(ch<'0' || ch>'9') {if(ch=='-') f = -1;ch = getchar();}
6 while(ch>='0' && ch<='9') x=(x<<3)+(x<<1)+(ch^48),ch = getchar();
7 return x*f;
8 }
9
10 int read() {
11 int ans = 0, f = 1; char c = getchar();
12 for (;c < '0' || c > '9'; c = getchar()) if (c == '-') f = -1;
13 for (;c >= '0' && c <= '9'; c = getchar()) ans = ans * 10 + c - '0';
14 return ans * f;

```

```

15 }
16
17 i64 read() {
18 i64 ans = 0, f = 1; char c = getchar();
19 for (; c < '0' || c > '9'; c = getchar()) if (c == '-') f = -1;
20 for (; c >= '0' && c <= '9'; c = getchar()) ans = ans * 10 + c - '0';
21 return ans * f;
22 }
23
24 template<class T>inline void read(T &res)
25 {
26 char c; T flag=1;
27 while((c=getchar())<'0' || c>'9')if(c=='-')flag=-1;res=c-'0';
28 while((c=getchar())>='0'&&c<='9')res=res*10+c-'0';res*=flag;
29 }
30
31 template <class T>
32 void write(T x){
33 if(x < 0) x = -x, putchar('-');
34 if(x >= 10) write(x / 10);
35 putchar('0' + x % 10);
36 }

```

### 9.1.2 \_\_int128 输入输出模板

```

1 //必须搭配scanf printf使用
2 __int128 read(){
3 __int128 x=0,f=1;
4 char ch=getchar();
5 while(ch<'0' || ch>'9'){
6 if(ch=='-')
7 f=-1;
8 ch=getchar();
9 }
10 while(ch>='0'&&ch<='9'){
11 x=x*10+ch-'0';
12 ch=getchar();
13 }
14 return x*f;
15 }
16 void print(__int128 x){
17 if(x<0){
18 putchar('-');
19 x=-x;
20 }
21 if(x>9)
22 print(x/10);
23 putchar(x%10+'0');
24 }

```

## 9.2 istringstream

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 string str, s;
4 int main() {
5 while (true) {
6 getline(cin, str);
7 if (int(str.size()) == 1 && str[0] == '#') break;
8 istringstream all(str);
9 while (all >> s) {
10 cout << "s : " << s << endl;
11 }
12 }
13 return 0;
14 }

```

### 9.3 unordered\_map 防 hack 模板

```

1 头文件 #include <bits/stdc++.h>
2 struct custom_hash {
3 static uint64_t splitmix64(uint64_t x) {
4 x += 0x9e3779b97f4a7c15;
5 x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
6 x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
7 return x ^ (x >> 31);
8 }
9
10 size_t operator()(uint64_t x) const {
11 static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().
 time_since_epoch().count();
12 return splitmix64(x + FIXED_RANDOM);
13 }
14 };
15
16 unordered_map<XXX, XXX, custom_hash> you_name_it;
17
18 两组数据:
19 unordered_map中的insert (优化过):
20 x = 107897: 0.035 seconds
21 x = 126271: 0.031 seconds
22
23 (未优化过):
24 x = 107897: 0.014 seconds
25 x = 126271: 2.787 seconds

```

### 9.4 杜教 BM

```

1 //建议放 >= 2阶数量的表到vector中
2 #include<bits/stdc++.h>
3 using namespace std;
4 #define rep(i,a,n) for (int i=a;i<n;i++)

```

```

5 #define pb push_back
6 typedef long long i64;
7 #define SZ(x) ((i64)(x).size())
8 typedef vector<i64> vi64;
9 typedef pair<i64,i64> PII;
10 const i64 mod=(i64)1e9 + 7;
11 i64 powmod(i64 a,i64 b) {
12 i64 res=1;
13 a%=mod;
14 assert(b>=0);
15 for(; b; b>>=1) {
16 if(b&1)res=res*a%mod;
17 a=a*a%mod;
18 }
19 return res;
20 }
21 i64 _,n;
22 namespace linear_seq {
23 const i64 N=10010;
24 i64 res[N],base[N],_c[N],_md[N];
25
26 vector<i64> Md;
27 void mul(i64 *a,i64 *b,i64 k) {
28 rep(i,0,k+k) _c[i]=0;
29 rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
30 for (i64 i=k+k-1; i>=k; i--) if (_c[i])
31 rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
32 rep(i,0,k) a[i]=_c[i];
33 }
34 i64 solve(i64 n,vi64 a,vi64 b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]+...
35 i64 ans=0,pnt=0;
36 i64 k=SZ(a);
37 assert(SZ(a)==SZ(b));
38 rep(i,0,k) _md[k-1-i]=-a[i];
39 _md[k]=1;
40 Md.clear();
41 rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
42 rep(i,0,k) res[i]=base[i]=0;
43 res[0]=1;
44 while ((1ll<pnt)<=n) pnt++;
45 for (i64 p=pnt; p>=0; p--) {
46 mul(res,res,k);
47 if ((n>p)&1) {
48 for (i64 i=k-1; i>=0; i--) res[i+1]=res[i];
49 res[0]=0;
50 rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
51 }
52 }
53 rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
54 if (ans<0) ans+=mod;
55 return ans;
56 }
57 vi64 BM(vi64 s) {

```

```

58 vi64 C(1,1),B(1,1);
59 i64 L=0,m=1,b=1;
60 rep(n,0,SZ(s)) {
61 i64 d=0;
62 rep(i,0,L+1) d=(d+(i64)C[i]*s[n-i])%mod;
63 if (d==0) ++m;
64 else if (2*L<=n) {
65 vi64 T=C;
66 i64 c=mod-d*powmod(b,mod-2)%mod;
67 while (SZ(C)<SZ(B)+m) C.pb(0);
68 rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
69 L=n+1-L;
70 B=T;
71 b=d;
72 m=1;
73 } else {
74 i64 c=mod-d*powmod(b,mod-2)%mod;
75 while (SZ(C)<SZ(B)+m) C.pb(0);
76 rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
77 ++m;
78 }
79 }
80 return C;
81 }
82 i64 gao(vi64 a,i64 n) {
83 vi64 c=BM(a);
84 c.erase(c.begin());
85 rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
86 return solve(n,c,vi64(a.begin(),a.begin()+SZ(c)));
87 }
88 };
89
90 i64 m, f[300];
91
92 int main() {
93 ios::sync_with_stdio(false);
94 cin.tie(0);
95 cin >> n >> m;
96 for (int i = 0; i < m; ++i) f[i] = 1;
97 for (int i = m; i <= 250; ++i) {
98 f[i] = (f[i - 1] + f[i - m]) % mod;
99 }
100 vi64 v;
101 for (int i = 1; i <= 250; ++i) {
102 v.emplace_back(f[i]);
103 }
104 cout << linear_seq::gao(v,n-1)%mod << '\n';
105 return 0;
106 }

```

## 9.5 模拟退火

```

1 JSOI2004 平衡点
2 可以优化玄学的几点:
3 1. 初始温度T的值
4 2. 降温系数的范围0.985~0.999
5 3. 多做几次退火减小误差
6 4. T>... (终止温度)
7 最好只改一个值, 比如T的初始值或多做几次退火, 不然就是在瞎jb交
8 const int N=1100;
9 int n,i;
10 double anx,any,vx,vy,dis,nx,ny,T,nv,x[N],y[N],w[N];
11 double Rand() {return (double)(rand()%20000)/20000.0;}
12 double dist(double xx,double yy){
13 nv=0;
14 for(int i=1; i<=n; ++i) nv+=sqrt((xx-x[i])*(xx-x[i])+(yy-y[i])*(yy-y[i]))*w[i];
15 if(nv<dis) dis=nv,anx=xx,any=yy;
16 return nv;
17 }
18 void SA(){
19 T=8000; //初始温度
20 for(;T>0.001;){//小于给定系数就退出
21 nx=vx; ny=vy;
22 nx=nx+T*(Rand()*2-1);//在当前位置的变化幅度内随机取一点
23 ny=ny+T*(Rand()*2-1);
24 nv=dist(vx,vy)-dist(nx,ny);//计算当前解
25 if(nv>0 || exp(nv/T)>rand()){//如果当前解比之前的最优解好那么取当前解
26 vx=nx; //否则以exp|当前解-最优解| / T的概率接受当前解
27 vy=ny;
28 }
29 T*=0.996; //降低搜索范围 (降温)
30 }
31 }
32 signed main() {
33 scanf("%d",&n);
34 for(i=1; i<=n; ++i){
35 scanf("%lf%lf%lf",&x[i],&y[i],&w[i]);
36 anx+=x[i];
37 any+=y[i];
38 }
39 anx/=(double)n; any/=(double)n;
40 vx=anx=vy=any; //(vx,vy)当前位置, (anx,any)最优解位置
41 dis=dist(anx,any);
42 /*for(int i=0; i<10; ++i)* SA();
43 printf("%.3f %.3f\n",anx,any);
44 return 0;
45 }

```

## 10 大数

### 10.1 java

#### 10.1.1 坑点



- 1 1. **for**里面做大数操作不要太多，因为大数做的每一个操作都是新生成一个大数的，**for**太多有可能会MLE。
- 2 17ccpc 秦皇岛的java题，64M，t=20，每次for3000次就会爆。

### 10.1.2 输入

```

1 输入
2 1.1 申明一个输入对象cin
3 Scanner cin=new Scanner(System.in);
4
5 1.2 输入一个int值
6 Int a=cin.nextInt();
7
8 1.3 输入一个大数
9 BigDecimal a=cin.nextBigDecimal();
10
11 1.4 EOF结束
12 while(cin.hasNext()) ...{}
13
14 输出
15 2.1 输出任意类型的str
16 System.out.println(str); //有换行
17 System.out.print(str) //无换行
18 System.out.println(" "+str); //输出字符串str
19 System.out.println("Hello,%s.Next year,you'll be %d",name,age);
20
21 大数类
22 3.1 赋值
23 BigInteger a=BigInteger.valueOf(12);
24 BigInteger b=new BigInteger(String.valueOf(12));
25 BigDecimal c=BigDecimal.valueOf(12.0);
26 BigDecimal d=new BigDecimal("12.0");//建议使用字符串以防止double类型导致的误差
27
28 也可以用上述方法构造一个临时对象用于参与运算
29 b.add(BigInteger.valueOf(105));

```

### 10.1.3 注意点

- 1 1. **for**里面做大数操作不要太多，因为大数做的每一个操作都是新生成一个大数的，**for**太多有可能会MLE。
- 2 17ccpc 秦皇岛的java题，64M，t=20，每次for3000次就会爆。

## 10.2 python

### 10.2.1 python

```

1 //https://ac.nowcoder.com/acm/contest/5670/E
2 def gcd(a, b):
3 if b == 0 : return a
4 else : return gcd(b, a % b)

```

```
5 n = int(input())
6 a = [1]
7 vis = [0] * (n + 1)
8 temp = input().split()
9 for x in temp :
10 tx = int(x)
11 a.append(tx)
12 ans = 1
13 for i in range(1, 1 + n) :
14 u = i
15 cnt = 0
16 if vis[u] == 1 : continue
17 while vis[u] == 0 :
18 vis[u] = 1;
19 cnt += 1
20 u = a[u]
21 ans = (ans * cnt) // gcd(ans, cnt)
22 ans = int(ans)
23 print(ans)
```