ICPC Templates For Grooming

ZZY

January 2, 2021

Contents

1 图论

1.1 最短路

1.1.1 堆优化 Dijkstra

```cpp
//不能求最长路，不能处理带负权的边 用spfa搞
// stl优先队列是O(eloge),手写二叉堆是O(elogv),斐波那契堆是O(vlogv + e)
//点的编号从0开始
#define i64 long long
const int N = EDIT+1000; //点数
const i64 INF = 0x3f3f3f3f3f3f3f3fLL;
struct node {
    int id;
    i64 w;
    node(){}
    node(int a, i64 b) : id(a), w(b) {} //hdu6805 美好的回忆:>
    friend bool operator < (node a, node b) {return a.w > b.w;}
};
vector<node> G[N];
bool vis[N];
i64 dis[N];

void dij(int s, int n) {
    priority_queue<node> q;
    while (!q.empty()) q.pop();
    node cur;
    for (int i = 0; i <= n; ++i) { //另外,memset比for快哦
        dis[i] = INF;
        vis[i] = 0;
    }
    dis[s] = 0;
    q.push(node(s, dis[s]));
    while (!q.empty()) {
        cur = q.top();
        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
        if (vis[cur.id]) continue;
        vis[cur.id] = 1;
        for (node to : G[cur.id]) {
            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
                dis[to.id] = dis[cur.id] + to.w;
                q.push(node(to.id, dis[to.id]));
            }
        }
    }
}
void init(int n) {
    for (int i = 0; i <= n; ++i) G[i].clear();
}
void addEdge(int a, int b, int c){
    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(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*logn*log(m/n) + m)
5   #define i64 long long
6   const int N = EDIT+100;
7   const i64 INF = 0x3f3f3f3f3f3f3f3fLL;
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 false
```

```
33          ++cnt[cur.id];
34          if(cnt[cur.id]>n) return false; //一定要是>n
35          vis[cur.id] = 0;
36          for (node to : G[cur.id]) {
37              if (dis[to.id] > dis[cur.id]+to.w) {
38                  dis[to.id] = dis[cur.id]+to.w;
39                  if (!vis[to.id]) {
40                      q.push(node(to.id, dis[to.id]));
41                      vis[to.id] = 1;
42                  }
43              }
44          }
45      }
46      return true;
47  }
48  void init(int n) {
49      for (int i = 0; i <= n; ++i) G[i].clear();
50  }
51
52  int main() {
53
54      int n, m, s, t;
55      cin >> n >> m >> s >> t;
56      init(n);
57      forn(i, m) {
58          int u, v, w;
59          cin >> u >> v >> w;
60          G[u].pb(node(v, w));
61          G[v].pb(node(u, w));
62      }
63      spfa(s, n); //s是起点, n是点数
64      cout << dis[t] << '\n';
65
66      return 0;
67  }
```

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              }
```

```
        }
    }
}

int main() {

    cin >> tc;
    while (tc--) {
        ms(G, 0);
        cin >> n >> m;
        forn(i, m) {
            int a, b;
            cin >> a >> b;
            G[a][b] = 1;
        }
        floyd();
        int cnt = 0;
        for (int i = 1; i <= n; ++i) {
            for (int j = i+1; j <= n; ++j) {
                if (!G[i][j] && !G[j][i]) ++cnt;
            }
        }
        cout << cnt << '\n';
    }

    return 0;
}
```

1.1.4 floyd 求最小环

```
//AcWing 344 floyd求最小环并且按顺序输出最小环中的点
//hdu 1599
const int maxn = 110;
const i64 INF = 0x3f3f3f3f;
i64 G[maxn][maxn], dis[maxn][maxn], road[maxn][maxn], ans;
vi res;

void floyd(int n) {
    int i, j, k;
    for (k = 1; k <= n; ++k) {
        for (i = 1; i < k; ++i) {
            if (G[k][i] == INF) continue;
            for (j = i+1; j < k; ++j) {
                if (G[k][j] == INF) continue;
                if (G[k][i]+G[k][j]+dis[i][j] < ans) {
                    ans = G[k][i]+G[k][j]+dis[i][j];
                    /* -----记录路径部分----- */
                    res.clear();
                    for (int temp = i; temp != j; temp = road[temp][j]) res.eb(temp);
                    res.eb(j);
                    res.eb(k);
                    /* ------------------ */
```

```
            }
        }
    }
    for (i = 1; i <= n; ++i) {
        if (dis[i][k] == INF) continue;
        for (j = 1; j <= n; ++j) {
            if (dis[k][j] == INF) continue;
            if (dis[i][k]+dis[k][j] < dis[i][j]) {
                dis[i][j] = dis[i][k]+dis[k][j];
                /* -----记录路径部分----- */
                road[i][j] = road[i][k];
                /* ------------------- */
            }
        }
    }
  }
}
void init(int n) {
  for (int i = 1; i <= n; ++i) for (int j = 1; j <= n; ++j) G[i][j] = INF;
  for (int i = 1; i <= n; ++i) for (int j = 1; j <= n; ++j) dis[i][j] = INF;
  ans = INF;
}
int n, m;

int main() {
  cin >> n >> m;
  init(n);
  forn(i, m) {
    i64 u, v, val;
    cin >> u >> v >> val;
    G[u][v] = G[v][u] = dis[u][v] = dis[v][u] = min(dis[u][v], val);
    road[u][v] = v; road[v][u] = u;
  }
  floyd(n);
  if (ans == INF) cout << "No solution." << '\n';
  else {
    cout << ans << endl;
    for (int x : res) cout << x << ' ';
    cout << '\n';
  }
  return 0;
}
```

1.1.5 johnson 全源最短路

```
//洛谷 johnson全源最短路模板题
Johnson 算法则通过另外一种方法来给每条边重新标注边权。
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。
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]
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]
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 = 0x3f3f3f3f3f3f3f3fLL;
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 (!v is[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≤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。
```

```
别忘了特判自环答案是-1。
技巧 & 注意点:
1. 别忘了要自建一个虚拟源点向所有点连0的边!
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)
3. a-b=c 可以转换成 a-b>=c && a-b<=c

最短路建法:
a-b<=c b->a (c)
a-b<c b->a (c-1)

a-b>=c a->b (-c)
a-b>c a->b (-c-1)

a==b b->a (0) && a->b (0)

a<=c S->a (c)
a<c S->a (c-1) ??

a>=c a->S (-c)
a>c a->S (-c-1) ??

a>=c*b b->a (log(c)) (跑最长路)
c*a>=b b->a (-log(c)) (跑最长路)
```

1.1.7 经典例题

```
最短路
Usaco2006 Dec Wormholes 虫洞
题意:
有n个点，m个边1，w个边2，边一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）？
思路:
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。

LightOJ 1108 Instant View of Big Bang
题意:
给你一张有向带权图，现在问你有哪些点可以走到负环，有的话就输出所有点，没的话就输出impossible
t<=125,1<=n<=1000,0<=m<=2000,0<=x,y<n,-1000<=t(权值)<=1000
思路:
1. 单纯用spfa搞，直接建反向图，然后跑负环，如果跑到负环就dfs那个点能到哪些点，然后存起来最后输出就行，注意spfa里和for里要判vv! 不然会超时
别问为什么不能建正向图跑负环 + 跑反向图看能到哪些点，问就是毒瘤题。
2. 建反向图，先跑强连通，然后继续for所有点spfa，这里spfa能走边的前提是两个点属于同一个强连通分量，因为负环肯定是一个强连通里的嘛，如果发现是负环也就加入点集里。
最后对点集跑个bfs看能到哪些点就行

LightOJ 1221 Travel Company
题意:
现在给你一张有向图（没写保证连通就是不连通），每条边都有收入和支出，现在要找到一个环让 环内总收入/环内总支出 > p,问你存不存在。
t<=100,2<=n(点数)<=100,0<=r(边数)<=9900,1<=p(要求的比例)<=100, 0<=a,b(顶点)<n, 0<=l(收
```

```
    入)<=5000,1<=e(支出)<=5000
思路:
总收入/总支出 > p 可以转换成 p*总支出 - 总收入 < 0,所以设a到b有一条收入l，支出e的边，那就a到b建一条 p*e-l 的边就行。
然后因为起点不知道，并且也不知道是不是连通，所以就对每个点跑spfa，没跑过的就跑，可以看代码里的vv数组，没了。

LightOJ 1002
题意:
给你一张带权无向图，可以有重边，给你起点，现在问你从起点分别到其他每个点的路上最大路径的最小值是多少？
思路:
改一下堆优化dij的松弛方式就行
if (!vis[to.id] && dis[to.id] > max(dis[cur.id],to.w)) {
   dis[to.id] = max(dis[cur.id],to.w);

16 ccpc final G Pandaland
时限:3000ms
题意:
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。
1<=T<=50,1<=m<=4000,-10000<=xi,yi<=10000,1<=w<=1e5
思路:
n^3的floyd最小环肯定不行。
解法一:
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。
堆优化dij里面还要剪两次支，就很玄学。
时间复杂度 O(m*m*log(m)) result: 900ms
解法2:
**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，然后可以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。
注意:题目给的图可能有多个联通块。
时间复杂度O(m*log(m)) result: 200ms

拆分约束
SCOI2011 糖果
题意:
有n个小朋友，要满足k个需求。
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。
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。
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。
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。
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。
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。
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？
思路:
因为要方案数(和)最小，所以跑最长路(lbn讲的)。

倍杀测量者
题意:
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。
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。
68  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。
69  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"
70  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。
71  思路:
72  二分+拆分约束
73  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];
```

```cpp
i64 dis[N],cnt[N];
bool vis[N];

bool spfa(int s, int n) { //s是起点,n是上界点数,点的编号从0开始
    queue<node> q;
    node cur;
    for (int i = 0; i <= n; ++i){
        dis[i] = -INF;
        vis[i]=cnt[i]=0;
    }
    vis[s] = 1;
    dis[s] = 0;
    q.push(node(s, dis[s]));
    while (!q.empty()) {
        cur = q.front();
        q.pop();
        //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
        ++cnt[cur.id];
        if(cnt[cur.id]>n+1) return false;
        vis[cur.id] = 0;
        for (node to : G[cur.id]) {
            if (dis[to.id] < dis[cur.id]+to.w) {
                dis[to.id] = dis[cur.id]+to.w;
                if (!vis[to.id]) {
                    q.push(node(to.id, dis[to.id]));
                    vis[to.id] = 1;
                }
            }
        }
    }
    return true;
}
void init(int n) {
    for (int i = 0; i <= n; ++i) G[i].clear();
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.precision(10);
    cout << fixed;
#ifdef LOCAL_DEFINE
    freopen("input.txt", "r", stdin);
#endif
    cin>>n>>m;
    init(n+5);
    int S=0;
    bool pre=false;
    forn(i, m){
        int op,a,b;
        cin>>op>>a>>b;
        if(a==b) if(op==2 || op==4) pre=true;
        if(op==1){
            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，然后可以通过枚举不
       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=0x3f3f3f3f3f3f3f3fLL;
```

```
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 x11,y11,x22,y22;
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].x11>>e[i].y11>>e[i].x22>>e[i].y22>>e[i].w;
90              b.emplace_back(make_pair(e[i].x11,e[i].y11));
91              b.emplace_back(make_pair(e[i].x22,e[i].y22));
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].x11,e[i].y11))-b.begin
                    ();
97              int v=lower_bound(b.begin(),b.end(),make_pair(e[i].x22,e[i].y22))-b.begin
                    ();
98              addEdge(u,v,e[i].w);
99              addEdge(v,u,e[i].w);
100         }
101         ans=(i64)1e10;
102         for(int i=0; i<m; ++i){
103             int st=lower_bound(b.begin(),b.end(),make_pair(e[i].x11,e[i].y11))-b.begin
                    ();
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],chklca[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         chklca[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;
```

```cpp
int findRoot(int x){return (x==fa[x]?x:fa[x]=findRoot(fa[x]));}
int main() {
    int tc,kase=1;
    cin>>tc;
    while(tc--){
        b.clear();
        cin>>n;
        for(int i=0; i<n; i++){
            cin>>x11[i]>>y11[i]>>x22[i]>>y22[i]>>w11[i];
            b.eb(mp(x11[i],y11[i]));
            b.eb(mp(x22[i],y22[i]));
        }
        sort(all(b));
        b.resize(unique(all(b))-b.begin());
        for(int i=0; i<n; ++i){
            int ai=lower_bound(all(b),mp(x11[i],y11[i]))-b.begin();
            int bi=lower_bound(all(b),mp(x22[i],y22[i]))-b.begin();
            nd[i]=node(ai,bi,w11[i]);
        }
        lca.init(SZ(b)+5);
        for(int i=0; i<SZ(b)+5; ++i){
            chklca[i]=0;
            fa[i]=i;
        }
        sort(nd,nd+n);
        int rt=-1;
        for(int i=0; i<n; ++i){
            int aa=findRoot(nd[i].a);
            int bb=findRoot(nd[i].b);
            if(aa!=bb){
                fa[bb]=aa;
                lca.addEdge(nd[i].a,nd[i].b,nd[i].w);
                nd[i].used=1;
            }
        }
        for(int i=0; i<SZ(b); ++i){
            if(!chklca[i]){
                lca.solve(i);
            }
        }
        i64 ans=(i64)1e10;
        for(int i=0; i<n; ++i){
            if(nd[i].used) continue;
            ans=min(ans,lca.getDist(nd[i].a,nd[i].b)+nd[i].w);
        }
        if(ans==(i64)1e10) ans=0; //cannot find cycle
        cout<<"Case #"<<kase++<<": "<<ans<<'\n';
    }
    return 0;
}
```

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 }
```

1.1.11 LightOJ 1208

```
1  LightOJ 1108 Instant View of Big Bang
2  题意:
3  给你一张有向带权图，现在问你有哪些点可以走到负环，有的话就输出所有点，没的话就输出
       impossible
4  t<=125,1<=n<=1000,0<=m<=2000,0<=x,y<n,-1000<=t(权值)<=1000
5  思路:
6  1. 单纯用spfa搞，直接建反向图，然后跑负环，如果跑到负环就dfs那个点能到哪些点，然后存起来最
       后输出就行，注意spfa里和for里要判vv! 不然会超时
7  别问为什么不能建正向图跑负环 + 跑反向图看能到哪些点，问就是毒瘤题。
8  2. 建反向图，先跑强连通，然后继续for所有点spfa，这里spfa能走边的前提是两个点属于同一个强连
       通分量，因为负环肯定是一个强连通里的嘛，如果发现是负环也就加入点集里。
9  最后对点集跑个bfs看能到哪些点就行
10
11 解法1:
12 const int N = 2000+100;
13 const i64 INF = 0x3f3f3f3f3f3f3f3fLL;
14 int n,m;
15 int tc,x[N],y[N],vv[N];
```

```
16 i64 t[N];
17 set<int> st;
18 bool have_cycle;
19 struct node {
20    int id;
21    i64 w;
22    node() {}
23    node(int a, i64 b) : id(a), w(b) {}
24 };
25 vector<node> G[N],E[N];
26 i64 dis[N],cnt[N];
27 bool vis[N];
28
29 void dfs(int x){
30    vv[x]=1;
31    st.insert(x);
32    for(node to:E[x]){
33       if(vv[to.id]) continue;
34       dfs(to.id);
35    }
36 }
37 bool spfa(int s, int n) { //s是起点,n是上界点数,点的编号从0开始
38    queue<node> q;
39    node cur;
40    for (int i = 0; i <= n; ++i){
41       dis[i] = INF;
42       vis[i]=cnt[i]=0;
43    }
44    vis[s] = 1;
45    dis[s] = 0;
46    q.push(node(s, dis[s]));
47    while (!q.empty()) {
48       cur = q.front();
49       q.pop();
50       //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
51       ++cnt[cur.id];
52       if(cnt[cur.id]>n){
53          have_cycle=true;
54          dfs(cur.id);
55          return false;
56       }
57       vis[cur.id] = 0;
58       for (node to : E[cur.id]) {
59          if (dis[to.id] > dis[cur.id]+to.w) {
60             if(vv[to.id]) continue;
61             dis[to.id] = dis[cur.id]+to.w;
62             if (!vis[to.id]) {
63                q.push(node(to.id, dis[to.id]));
64                vis[to.id] = 1;
65
66             }
67          }
68       }
```

```
    }
    return true;
}
void init(int n) {
    st.clear();
    have_cycle=false;
    for (int i = 0; i <= n; ++i){
        G[i].clear();
        E[i].clear();
        vv[i]=0;
    }
}
int main() {
    int kase=1;
    scanf("%d",&tc);
    while(tc--){
        scanf("%d%d",&n,&m);
        init(n+5);
        forn(i, m){
            scanf("%d%d%lld",&x[i],&y[i],&t[i]);
            G[x[i]].eb(node(y[i],t[i]));
            E[y[i]].eb(node(x[i],t[i]));
        }
        forn(i, n){
            if(vv[i]) continue;
            spfa(i, n);
        }
        printf("Case %d:",kase++);
        if(!have_cycle) printf(" impossible\n");
        else{
            for(int x:st) printf(" %d",x);
            puts("");
        }
    }
    return 0;
}

解法2:
const int N = 3000+100; //点数
const i64 INF = 0x3f3f3f3f3f3f3f3fLL;
vi neg,ans;
int final[N];
int tc,n,m;
int scc, top, tot;
int been[N];
vector<int> G[N];
int low[N], dfn[N], belong[N];
int stk[N], vis[N];
struct node {
    int id;
    i64 w;
    node() {}
    node(int a, i64 b) : id(a), w(b) {}
```

```
};
vector<node> G2[N];
i64 dis[N],cnt[N];
bool vv[N];
bool spfa(int s, int n) { //s是起点,n是上界点数,点的编号从0开始
    queue<node> q;
    node cur;
    for (int i = 0; i <= n; ++i){
        dis[i] = INF;
        vv[i] = cnt[i] = 0;
    }
    vv[s] = 1;
    dis[s] = 0;
    q.push(node(s, dis[s]));
    while (!q.empty()) {
        cur = q.front();
        q.pop();
        //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 false
        been[cur.id]=1;
        ++cnt[cur.id];
        if(cnt[cur.id]>n){
            neg.eb(cur.id);
            return false; //一定要是>n
        }
        vv[cur.id] = 0;
        for (node to : G2[cur.id]) {
            if(belong[to.id] != belong[cur.id]) continue;
            if (dis[to.id] > dis[cur.id]+to.w) {
                dis[to.id] = dis[cur.id]+to.w;
                if (!vv[to.id]) {
                    q.push(node(to.id, dis[to.id]));
                    vv[to.id] = 1;
                }
            }
        }
    }
    return true;
}
void init2(int n) {
    for (int i = 0; i <= n; ++i){
        G2[i].clear();
        been[i]=0;
        final[i]=0;
    }
}
void init(int n) {
    for (int i = 0; i <= n; ++i) {
        G[i].clear();
        low[i] = 0;
        dfn[i] = 0;
        stk[i] = 0;
        vis[i] = 0;
    }
```

```
175     scc = top = tot = 0;
176 }
177 void tarjan(int x) {
178     stk[top++] = x;
179     low[x] = dfn[x] = ++tot;
180     vis[x] = 1;
181     for (int to : G[x]) {
182         if (!dfn[to]) {
183             tarjan(to);
184             low[x] = min(low[x], low[to]);
185         } else if (vis[to]) {
186             low[x] = min(low[x], dfn[to]);
187         }
188     }
189     if (low[x] == dfn[x]) {
190         ++scc;
191         int temp;
192         do {
193             temp = stk[--top];
194             belong[temp] = scc;
195             vis[temp] = 0;
196         } while (temp != x);
197     }
198 }
199 void bfs(){
200     queue<int> q;
201     for(int x:neg){
202         q.push(x);
203         final[x]=1;
204         ans.eb(x);
205     }
206     while(!q.empty()){
207         int x = q.front();
208         q.pop();
209         for(int to:G[x]){
210             if(!final[to]){
211                 final[to]=1;
212                 q.push(to);
213                 ans.eb(to);
214             }
215         }
216     }
217 }
218 int main() {
219     scanf("%d",&tc);
220     int kase=1;
221     while(tc--){
222         scanf("%d%d",&n,&m);
223         neg.clear();
224         ans.clear();
225         init(n+5);
226         init2(n+5);
227         forn(i, m){
```

```
228             int x,y,t;
229             scanf("%d%d%d",&x,&y,&t);
230             ++x; ++y;
231             //printf("x:%d y:%d\n",x,y);
232             G[y].eb(x);
233             G2[y].eb(node(x,t));
234         }
235         for1(i, n) if(!dfn[i]) tarjan(i);
236         for1(i, n){
237             if(been[i]) continue;
238             spfa(i, n);
239         }
240         bfs();
241         printf("Case %d:",kase++);
242         if(!SZ(ans)) printf(" impossible\n");
243         else{
244             sort(all(ans));
245             for(int x:ans) printf(" %d",x-1);
246             puts("");
247         }
248     }
249     return 0;
250 }
```

1.1.12 LightOJ 1221

```
1  LightOJ 1221 Travel Company
2  题意:
3  现在给你一张有向图（没写保证连通就是不连通），每条边都有收入和支出，现在要找到一个环让 环内
       总收入/环内总支出 > p,问你存不存在。
4  t<=100,2<=n(点数)<=100,0<=r(边数)<=9900,1<=p(要求的比例)<=100, 0<=a,b(顶点)<n, 0<=l(收
       入)<=5000,1<=e(支出)<=5000
5  思路:
6  总收入/总支出 > p 可以转换成 p*总支出 - 总收入 < 0,所以设a到b有一条收入l，支出e的边，那就a
       到b建一条 p*e-l 的边就行。
7  然后因为起点不知道，并且也不知道是不是连通，所以就对每个点跑spfa，没跑过的就跑，可以看代码里
       的vv数组，没了。
8
9  const int N = 100+100;
10 const i64 INF = 0x3f3f3f3f3f3f3f3fLL;
11 int tc,r,p;
12 int a,b,l,e;
13 int n,m,vv[N];
14 struct node {
15     int id;
16     i64 w;
17     node() {}
18     node(int a, i64 b) : id(a), w(b) {}
19 };
20 vector<node> G[N];
21 i64 dis[N],cnt[N];
22 bool vis[N];
```

```
23
24 bool spfa(int s, int n) { //s是起点,n是上界点数,点的编号从0开始
25     queue<node> q;
26     node cur;
27     for (int i = 0; i <= n; ++i){
28         dis[i] = INF;
29         vis[i] = cnt[i] = 0;
30     }
31     vis[s] = 1;
32     dis[s] = 0;
33     q.push(node(s, dis[s]));
34     while (!q.empty()) {
35         cur = q.front();
36         q.pop();
37         vv[cur.id]=1; //这个点跑过spfa咯 1.
38         //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 false
39         ++cnt[cur.id];
40         if(cnt[cur.id]>n) return false; //一定要是>n
41         vis[cur.id] = 0;
42         for (node to : G[cur.id]) {
43             if (dis[to.id] > dis[cur.id]+to.w) {
44                 dis[to.id] = dis[cur.id]+to.w;
45                 if (!vis[to.id]) {
46                     q.push(node(to.id, dis[to.id]));
47                     vis[to.id] = 1;
48                 }
49             }
50         }
51     }
52     return true;
53 }
54 void init(int n) {
55     for (int i = 0; i <= n; ++i){
56         G[i].clear();
57         vv[i] = 0;
58     }
59 }
60
61 int main() {
62     int kase=1;
63     scanf("%d",&tc);
64     while(tc--){
65         scanf("%d%d%d",&n,&r,&p);
66         init(n+10);
67         forn(i, r){
68             scanf("%d%d%d%d",&a,&b,&l,&e);
69             G[a].eb(node(b,p*e-l));
70         }
71         bool ok=false;
72         forn(i, n){
73             if(vv[i]) continue; //这个点跑过spfa咯 2.
74             if(!spfa(i, n)){
75                 ok=true;
```

```
76              break;
77          }
78      }
79      printf("Case %d: ",kase++);
80      printf("%s\n",ok?"YES":"NO");
81   }
82   return 0;
83}
```

1.2 生成树

1.2.1 最小树形图固定根

```
1  洛谷模板题 复杂度O(n*m)?? 朱刘
2  点编号从0~n-1
3  题意:
4  给你n个点，m条边，指定一个根r，问你最小树形图的权值和是多少
5  1<=n<=100, 1<=m<=1e4, 1<=w<=1e6
6
7  const int N=200;
8  const int INF=0x3f3f3f3f;
9  int n,m,r;
10 //固定根的最小树形图，邻接矩阵写法
11 struct DMST{
12    int n;
13    int w[N][N]; // 边权
14    int vis[N]; // 访问标记，仅用来判断无解
15    int ans; // 计算答案
16    int removed[N];// 每个点是否被删除
17    int cid[N]; // 所在圈编号
18    int pre[N]; // 最小入边的起点
19    int iw[N]; // 最小入边的权值
20    int max_cid; // 最大圈编号
21    void init(int n){
22       this->n=n;
23       for(int i=0; i<n; ++i)
24          for(int j=0; j<n; ++j) w[i][j]=INF;
25    }
26    void addEdge(int u, int v, int cost){
27       w[u][v]=min(w[u][v],cost); //重边取边权最小的
28    }
29    //从S出发能到达多少个节点
30    int dfs(int s){
31       vis[s]=1;
32       int ans=1;
33       for(int i=0; i<n; ++i)
34          if(!vis[i] && w[s][i]<INF) ans+=dfs(i);
35       return ans;
36    }
37    //从u除法沿着pre指针找环
38    bool cycle(int u){
39       ++max_cid;
```

```
40      int v=u;
41      while(cid[v]!=max_cid){
42          cid[v]=max_cid;
43          v=pre[v];
44      }
45      return v==u;
46  }
47  //计算u的最小入弧，入弧起点不得在圈C中
48  void update(int u){
49      iw[u]=INF;
50      for(int i=0; i<n; ++i){
51          if(!removed[i] && w[i][u]<iw[u]){
52              iw[u]=w[i][u];
53              pre[u]=i;
54          }
55      }
56  }
57  //根节点为S，如果失败则返回false
58  bool solve(int s){
59      memset(vis,0,sizeof(vis));
60      if(dfs(s)!=n) return false;
61      memset(removed,0,sizeof(removed));
62      memset(cid,0,sizeof(cid));
63      for(int u=0; u<n; ++u) update(u);
64      pre[s]=s;
65      iw[s]=0;
66      ans=max_cid=0;
67      for(;;){
68          bool have_cycle=false;
69          for(int u=0; u<n; ++u){
70              if(u!=s && !removed[u] && cycle(u)){
71                  have_cycle=true;
72                  //以下代码缩环，环上除了u之外的节点均删除
73                  int v=u;
74                  do{
75                      if(v!=u) removed[v]=1;
76                      ans+=iw[v];
77                      //对于圈外点i,把i->v改成i->u(并调整权值);v->i改为u->i
78                      //注意圈上可能还有一个v'使得i->v'或者v'->i存在
79                      //因此只保留权值最小的i->u和u->i
80                      for(int i=0; i<n; ++i){
81                          if(cid[i]!=cid[u] && !removed[i]){
82                              if(cid[i]!=cid[u] && !removed[i]){
83                                  if(w[i][v]<INF) w[i][u]=min(w[i][u],w[i][v]-iw[v]);
84                                  w[u][i]=min(w[u][i],w[v][i]);
85                                  if(pre[i]==v) pre[i]=u;
86                              }
87                          }
88                      }
89                      v=pre[v];
90                  }while(v!=u);
91                  update(u);
92                  break;
```

```
            }
        }
        if(!have_cycle) break;
    }
    for(int i=0; i<n; ++i) if(!removed[i]) ans+=iw[i];
    return true;
  }
}dmst;
signed main() {
  cin>>n>>m>>r;
  dmst.init(n);//别忘初始化，只能赋值n(点数)，因为这个n是给到mdst里的n
  forn(i, m){
    int a,b,c;
    cin>>a>>b>>c;
    --a; --b;
    dmst.addEdge(a,b,c);
  }
  if(!dmst.solve(--r)) cout<<-1<<'\n';
  else cout<<dmst.ans<<'\n';
  return 0;
}
```

1.2.2 最小树形图固定根输出方案

```
https://judge.yosupo.jp/problem/directedmst
题意:
给你一个n个点，m条边的图，固定根s，现在让你求出这张图的最小树形图，让你输出最小花费以及每个点的父亲节点。
节点编号0~n-1, 1<=n<=2e5, n-1<=m<=2e5, 0<=w(边权)<=1e9

#include<bits/stdc++.h>
using namespace std;
const int SZ = 200100;
struct DSU{
  int p[SZ+1],sz[SZ+1];
  stack<pair<int,int> > st;
  DSU(){I(SZ);}
  void I(int n){for(int i=0;i<=n;i++) sz[p[i]=i]=1;}
  int F(int x){return x==p[x]?x:F(p[x]);}
  bool U(int x,int y){
    x=F(x),y=F(y);
    if(x==y) return st.emplace(-1,-1), false;
    if(sz[x]<sz[y]) swap(x,y);
    return p[y]=x,sz[x]+=sz[y],st.emplace(x,y),true;
  }
  void R(){
    if(st.empty()) return;
    auto [x,y]=st.top();st.pop();
    if(x==-1) return;
    sz[x]-=sz[y],p[y]=y;
  }
};
```

```cpp
typedef long long int T;
struct E{int a,b;T w;};
struct Node{
    E key;Node *l,*r;T d;E top(){return push(),key;}
    void push(){key.w+=d;if(l) l->d+=d;if(r) r->d+=d;d=0;}
};
Node* merge(Node *a,Node *b){
    if(!a||!b) return a?a:b;
    a->push(),b->push();if(a->key.w>b->key.w) swap(a,b);
    return swap(a->l,(a->r=merge(b,a->r))),a;
}
void pop(Node *&a){a->push(),a=merge(a->l,a->r);}
pair<T,vector<int> > DMST(int n,int r,const vector<E>& g){
    DSU D;vector<Node*> heap(n);vector<pair<int,int> > in(n,{-1,-1});
    T res=0;vector<int> used(n,-1);used[r]=r;
    vector<pair<int,vector<E> > > cycs;
    for(auto &e:g) heap[e.b]=merge(heap[e.b],new Node{e});
    for(int s=0,u=s,w;s<n;s++,u=s){
        vector<pair<int,E> > path;
        while(used[u]==-1){
            if(!heap[u]) return {-1,{}};
            used[u]=s;auto e=heap[u]->top();path.emplace_back(u,e);
            heap[u]->d-=e.w,pop(heap[u]),res+=e.w;u=D.F(e.a);
            if(used[u]==s){
                Node *o=0;cycs.emplace_back();
                do{
                    o=merge(o,heap[w=path.back().first]);
                    cycs.back().second.emplace_back(path.back().second);
                    path.pop_back();
                }while(D.U(u,w));
                u=D.F(u),heap[u]=o,used[u]=-1,cycs.back().first=u;
            }
        }
        for(auto &t:path) in[D.F(t.second.b)]={t.second.a,t.second.b};
    }
    while(!cycs.empty()){
        auto c=cycs.back();cycs.pop_back();
        pair<int,int> inedge = in[c.first];
        for(auto &t:c.second) D.R();
        for(auto &t:c.second) in[D.F(t.b)]={t.a,t.b};
        in[D.F(inedge.second)]=inedge;
    }
    vector<int> inv;for(int i=0;i<n;i++) inv.emplace_back(in[i].first);
    return {res,inv};
}


int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int n,m,s;
    cin>>n>>m>>s;
    vector<E> e(m);
```

```
    for(auto &it:e)
        cin>>it.a>>it.b>>it.w;
    auto res=DMST(n,s,e);
    cout<<res.first<<endl;
    for(int i=0;i<n;i++)
        if(res.second[i]==-1)
            cout<<i<<' ';
        else
            cout<<res.second[i]<<' ';
    cout<<endl;
}
```

1.2.3 最小树形图不固定根

```
hdu 2121
给你一个n个点，m条边的有向图，让你选一个点作为首都，这个首都到其他点的权值和最小，如果有的话
    就输出权值和以及作为首都的那个点，不存在的话就输出impossible

const int N=10050,M=50050,inf=0x7fffffff; //N顶点数，M边数
int n,m;
int ROOT,st,tot;
struct DMST{
    int n,sz,pre[N],id[N],vis[N],in[N];
    struct EDGE{
        int u,v,cost;
        EDGE(){}
        EDGE(int a,int b,int c):u(a),v(b),cost(c){}
    }E[M];
    void init(int _n){
        n=_n;
        sz=0;
    }
    void add(int u,int v,int w){
        E[sz++]=EDGE(u,v,w);
    }
    int dmst(int root){
        int u,v,cnt,ret=0;
        while(1){
            for(int i=0; i<n; ++i) in[i]=inf;
            for(int i=0; i<sz; ++i){
                u=E[i].u,v=E[i].v;
                if(E[i].cost<in[v] && u!=v){
                    pre[v]=u;
                    in[v]=E[i].cost;
                    if(u==root) ROOT=i;
                }
            }
            for(int i=0; i<n; ++i){
                if(i!=root && in[i]==inf) return -1;
            }
            cnt=in[root]=0;
            for(int i=0; i<n; ++i) id[i]=vis[i]=-1;
```

```
38          for(int i=0; i<n; ++i){
39              ret+=in[i],v=i;
40              while(vis[v]!=i && id[v]==-1 && v!=root) {
41                  vis[v]=i;
42                  v=pre[v];
43              }
44              if(v!=root && id[v]==-1){
45                  for(u=pre[v]; u!=v; u=pre[u]) id[u]=cnt;
46                  id[v]=cnt++;
47              }
48          }
49          if(!cnt) break;
50          for(int i=0; i<n; ++i) if(id[i]==-1) id[i]=cnt++;
51          for(int i=0; v=E[i].v,i<sz; ++i){
52              E[i].u=id[E[i].u],E[i].v=id[E[i].v];
53              if(E[i].u!=E[i].v) E[i].cost-=in[v];
54          }
55          n=cnt,root=id[root];
56      }
57      return ret;
58  }
59  }G;
60  void variable(int &cost, int &root){ //Variable Root
61      for(int i=0; i<n; ++i) G.add(st,i,tot); //st=n,tot=sum of Edge Wight+1
62      int ans=G.dmst(st);
63      if(ans==-1 || ans-tot>=tot){
64          cout<<"impossible"<<'\n';
65          cout<<'\n';
66          return;
67      }
68      cost=ans-tot,root=ROOT-m;
69      cout<<cost<<' '<<root<<'\n';
70      cout<<'\n';
71      return;
72  }
73  int main() {
74      while(cin>>n>>m){
75          G.init(n+1); //+1是给虚点准备的,fk cls
76          tot=1;
77          for(int i=0; i<m; ++i){
78              int a,b,c;
79              cin>>a>>b>>c;
80              tot+=c;
81              G.add(a,b,c);
82          }
83          st=n;
84          int cost=0,root=0;
85          variable(cost,root);
86      }
87      return 0;
88  }
```

1.3 网络流

1.3.1 DICNIC

```
/*
网络流dinic复杂度
上届O((n^2)m)
若所有边容量为1,O(min(n^(1/3),m^(1/2))m)
二分图O(n^(1/2)m)
*/
//如果是无向图，加的反向边流量也为w，而不是0
//要开long long不如直接#define int long long,注意要把int()改成(int)()
const int N = EDIT+100; //点数
const int INF = 0x3f3f3f3f; //ll const i64 INF =0x8个3fLL; .1.
struct Edge{
    int from, to, cap, flow; //如果要开ll的话，这边也要开ll .2.
    Edge(int u, int v, int c, int f) //如果要开ll的话，这边也要开ll .3.
        : from(u), to(v), cap(c), flow(f) {}
};
struct Dicnic {
    #define Type int
    int n, m, s, t; //节点数，边数（包括反向弧），源点编号和汇点编号
    vector<Edge> edges; //边表。edge[e]和edge[e^1]互为反向弧
    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
    bool vis[N]; //BFS使用
    Type d[N]; //从起点到i的距离
    int cur[N]; //当前弧下标
    void init(int n) {
        this->n = n;
        for (int i = 0; i <= n; ++i) G[i].clear();
        edges.clear();
    }
    void addEdge(int from, int to, Type cap) {
        edges.emplace_back(Edge(from, to, cap, 0));
        edges.emplace_back(Edge(to, from, 0, 0));
        m = int(edges.size());
        G[from].emplace_back(m-2);
        G[to].emplace_back(m-1);
    }
    bool BFS() {
        memset(vis, 0, sizeof(vis));
        memset(d, 0, sizeof(d));
        queue<int> q;
        while (!q.empty()) q.pop();
        q.push(s);
        d[s] = 0;
        vis[s] = 1;
        while (!q.empty()) {
            int x = q.front();
            q.pop();
            for (int i = 0; i < int(G[x].size()); ++i) {
                Edge &e = edges[G[x][i]];
                if (!vis[e.to] && e.cap > e.flow) {
```

```
                vis[e.to] = 1;
                d[e.to] = d[x] + 1;
                q.push(e.to);
            }
        }
    }
    return vis[t];
}
Type DFS(int x, Type a) {
    if (x == t || a == 0) return a;
    Type flow = 0, f;
    for (int &i = cur[x]; i < int(G[x].size()); ++i) { //从上次考虑的弧
        Edge &e = edges[G[x][i]];
        if (d[x]+1 == d[e.to] && (f = DFS(e.to, min(a, e.cap-e.flow))) > 0) {
            e.flow += f;
            edges[G[x][i]^1].flow -= f;
            flow += f;
            a -= f;
            if (a == 0) break;
        }
    }
    return flow;
}
Type Maxflow(int s, int t) {
    this->s = s; this->t = t;
    Type flow = 0;
    while (BFS()) {
        memset(cur, 0, sizeof(cur));
        flow += DFS(s, INF);
    }
    return flow;
}
#undef Type
}dicnic;

int main() {
    cin >> n >> m >> s >> t; //V, E, 原点, 汇点
    dicnic.init(n); //初始化
    while (m--) {
        int u, v, w;
        cin >> u >> v >> w;
        dicnic.addEdge(u, v, w); //建图
    }
    cout << dicnic.Maxflow(s, t) << '\n'; //跑最大流
    return 0;
}
```

1.3.2 ISAP

```
//时间复杂度 O(v^2*E)
const int maxn = "EDIT"; //点数
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
```

```
int main() {

    cin >> n >> m >> s >> t;
    isap.init(n);
    while (m--) {
        int u, v, w;
        cin >> u >> v >> w;
        isap.addEdge(u, v, w);
    }
    cout << isap.Maxflow(s, t) << '\n';

    return 0;
}
```

1.3.3 MCMF

```
//洛谷P3381
#define ll long long
const int maxn = 5000+100;
const int INF = INT_MAX;
struct Edge{
    int from, to, cap, flow, cost;
    Edge(int u, int v, int c, int f, int cc)
        : from(u), to(v), cap(c), flow(f), cost(cc) {}
};
struct MCMF {
    int n, m;
    vector<Edge> edges;
    vector<int> G[maxn];
    int inq[maxn]; //是否在队列中
    int d[maxn]; //bellmanford
    int p[maxn]; //上一条弧
    int a[maxn]; //可改进量
    void init(int n) {
        this->n = n;
        for (int i = 0; i <= n; ++i) G[i].clear();
        edges.clear();
    }
    void addEdge(int from, int to, int cap, int cost) {
        edges.emplace_back(from, to, cap, 0, cost);
        edges.emplace_back(to, from, 0, 0, -cost);
        m = int(edges.size());
        G[from].push_back(m - 2);
        G[to].push_back(m - 1);
    }
    bool spfa(int s, int t, int &flow, ll &cost) {
        for (int i = 1; i <= n; ++i) d[i] = INF;
        memset(inq, 0, sizeof(inq));
        d[s] = 0;
        inq[s] = 1;
        p[s] = 0;
        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 }
```

1.3.4 常见思路

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。

最大权闭合子图：
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。
最大闭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。
具体的建图方法为：
源点向所有正权点连结一条容量为权值的边
保留原图中所有的边，容量为正无穷
所有负权点向汇点连结一条容量为权值绝对值的边
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。
以下约定源点为 ss，汇点为 tt。
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。
小技巧：dicnic里的d数组不为0，就说明那个点要取
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。

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
最小点权覆盖集的建图方法：
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
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
3、对原图中的边，将边权变成无穷大

最小割割边唯一性判断：
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）。
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];
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]。

1.4 匹配问题

1.4.1 匈牙利

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

using namespace std;

const int maxn = 600;

int nl, nr, m;
int mx[maxn], my[maxn], vis[maxn];
vector<int> G[maxn];

bool aug(int x) {
   for (int to : G[x]) {
      if (vis[to] == -1) {
         vis[to] = 1;
         if (my[to] == -1 || aug(my[to])) {
            my[to] = x;
            mx[x] = to;
```

```
            return true;
        }
    }
  }
  return false;
}

int main() {
  scanf("%d%d%d", &nl, &nr, &m);
  for (int i = 0; i < m; ++i) {
    int u, v;
    scanf("%d%d", &u, &v);
    G[u].emplace_back(v);
  }
  int maxMatch = 0;
  memset(my, -1, sizeof(my));
  memset(mx, -1, sizeof(mx));
  for (int i = 1; i <= nl; ++i) {
    memset(vis, -1, sizeof(vis));
    if (aug(i)) ++maxMatch;
  }
  printf("%d\n", maxMatch);
  for (int i = 1; i <= nl; ++i) {
    printf("%d ", mx[i] == -1 ? 0 : mx[i]);
  }puts("");
  return 0;
}
```

1.4.2 HK

```
/*
    别忘了给un赋值为左端点的个数 在主函数里赋值un=...
    如果从0~(n-1)开始编号，就改①②处地方
    如果是多组数据别忘了清空G
*/
/**********hdu2389*********/
int tc, t, m, n;
struct coordinate {
  int x, y, speed;
}human[3100], umb[3100];
/*************************/
const int maxn = 6100;
const int INF = 0x3f3f3f3f;
vector< vector<int> > G(maxn);
int un; //un为左端的顶点数，编号1~(un)
int mx[maxn], my[maxn];
int dx[maxn], dy[maxn];
int dis;
bool vis[maxn];
bool SearchP() {
  queue<int> q;
  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 }
```

1.4.3 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;
```

```
                return true;
            }
        } else {
            slack[i] = min(slack[i], diff);
        }
    }
}
return false;
}
int KM() {
    ms(match, -1); ms(num_b, 0);
    for (int i = 0; i < n; ++i) {
        num_a[i] = INT_MIN;
        for (int j = 0; j < m; ++j) {
            num_a[i] = max(num_a[i], link[i][j]);
        }
    }
    for (int i = 0; i < n; ++i) {
        fill(slack, slack+m, INF);
        int flag = 0;
        for (int j = 0; j < m; ++j) if (link[i][j] != -1) flag = 1;
                    or
        if (num_a[i] != -1) flag = 1;
        //两点之间不一定有边的要加这句
        //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)
        while (flag) {
            ms(vis_a, 0); ms(vis_b, 0);
            if (dfs(i)) break;
            int d = INF;
            for (int j = 0; j < m; ++j) if (!vis_b[j]) d = min(d, slack[j]);
            if (d == INF) break;
            for (int j = 0; j < max(n, m); ++j) {
                if (j < n) {
                    if (vis_a[j]) num_a[j] -= d;
                }
                if (j < m) {
                    if (vis_b[j]) num_b[j] += d;
                    else slack[j] -= d;
                }
            }
        }
    }
    int res = 0;
    for (int i = 0; i < m; ++i) {
        if (match[i] != -1) {
            res += link[match[i]][i];
        }
    }
    return res;
}

int main() {

```

```
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  }
```

1.4.4 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  }
```

1.4.5 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];
```

```
namespace Blossom {
   int mate[maxn], n, ret, nxt[maxn], f[maxn], mark[maxn], vis[maxn], t;
   queue<int> q;
   int F(int x) {return x == f[x] ? x : f[x] = F(f[x]);}
   void Merge(int a, int b) {f[F(a)] = F(b);}
   int lca(int x, int y) {
      for (t++;;swap(x, y)) {
         if (~x) {
            if (vis[x = F(x)] == t) return x;
            vis[x] = t;
            x = (mate[x] != -1 ? nxt[mate[x]] : -1);
         }
      }
   }
   void group(int a, int p) {
      for (int b, c; a != p; Merge(a, b), Merge(b, c), a = c) {
         b = mate[a], c = nxt[b];
         if (F(c) != p) nxt[c] = b;
         if (mark[b] == 2) mark[b] = 1, q.push(b);
         if (mark[c] == 2) mark[c] = 1, q.push(c);
      }
   }
   void aug(int s, const vector<int> G[]) {
      for (int i = 0; i < n; ++i) {
         nxt[i] = vis[i] = -1;
         f[i] = i;
         mark[i] = 0;
      }
      while (!q.empty()) q.pop();
      q.push(s);
      mark[s] = 1;
      while (mate[s] == -1 && !q.empty()) {
         int x = q.front();
         q.pop();
         for (int i = 0, y; i < int(G[x].size()); ++i) {
            if ((y = G[x][i]) != mate[x] && F(x) != F(y) && mark[y] != 2) {
               if (mark[y] == 1) {
                  int p = lca(x, y);
                  if (F(x) != p) nxt[x] = y;
                  if (F(y) != p) nxt[y] = x;
                  group(x, p); group(y, p);
               } else if (mate[y] == -1) {
                  nxt[y] = x;
                  for (int j = y, k, l; ~j; j = l) {
                     k = nxt[j];
                     l = mate[k];
                     mate[j] = k;
                     mate[k] = j;
                  }
                  break;
               } else {
                  nxt[y] = x;
                  q.push(mate[y]);
```

```
                    mark[mate[y]] = 1;
                    mark[y] = 2;
                }
            }
        }
    }
    int solve(int _n, vector<int> G[]) {
        n = _n;
        memset(mate, -1, sizeof(mate));
        for (int i = t = 0; i < n; ++i) if (mate[i] == -1) aug(i, G);
        for (int i = ret = 0; i < n; ++i) ret += (mate[i] > i);
        printf("%d\n", ret);
        for (int i = 0; i < n; ++i) printf("%d ", mate[i] + 1);
        return ret;
    }
}

int main() {
    //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
    scanf("%d%d", &n, &m);
    for (int i = 0; i < m; ++i) {
        scanf("%d%d", &x, &y);
        --x; --y; //顶点0~(n-1)编号
        G[x].emplace_back(y);
        G[y].emplace_back(x);
    }
    Blossom::solve(n, G);
    return 0;
}
```

1.4.6 稳定婚姻问题

```
洛谷 p1407
题意:
给你所有人的关系，问每对情侣之间的是否稳定。
解法:
强连通分量。
先连正房之间的关系，女->男。
再连二奶之间的关系，男->女。
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。
在同一个强连通分量中，就说明不稳定。否则稳定。
```

1.4.7 常见思路

```
最小点覆盖 = 最大匹配 //选最少的点覆盖整个二分图 poj 3041 Asteroids
//无向图的最小点覆盖 gym 102835 F {点数<=2e5,边数<=2e5} 建双向边，建超级源点和超级汇点，然后跑最大流，得到的结果除以2就是最小点覆盖数
最小边覆盖 = 点数 - 最大匹配 //选最少的边覆盖整个二分图
最大独立集 = 点数 - 最大匹配
```

```
5
6  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。
7
8  二分图的判断：染色法
9  假设dfs初始点a涂黑色，并与他相邻的点就涂白色。
10 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。
11
12 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
13 于同一个顶点，则称M是一个匹配。
14 最大匹配：包含边数最多的匹配。
15 完美匹配（完备匹配）：所有点都在匹配边上的匹配。
16 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。
17
18 匈牙利算法 O(V * E)
19 HK算法 O(sqrt(V) * E)
```

1.5 二分图博弈

1.5.1 二分图博弈

```
1  一类博弈问题，基于以下条件：
2
3  1.博弈者人数为两人，双方轮流进行决策。
4  2.博弈状态（对应点）可分为两类（状态空间可分为两个集合），对应二分图两边（X集和Y集）。任意合
       法的决策（对应边）使状态从一类跳转到另一类。（正是由于这个性质使得问题可以用二分图描述）
5  3.不可以转移至已访问的状态。（不可重复访问点）
6  4.无法转移者判负。
7
8  这类问题相当于从二分图指定起点开始轮流移动，不可重复访问点，无法移动判负 。
9
10 结论：先手如果在最大匹配的非必要点，必输，如果在最大匹配的必要点，必胜。
11
12 bzoj 1443 JSOI2009
13 题意：
14 有一个(100*100)的矩形，现在矩形格子里的点有两种状态，一种是可以走，还有一种是不能走，现在yy
       先手，mm后手，问这个矩形中有哪些点让yy先手，结果是mm赢。
15 思路：
16 看清题意，规定yy先手，问你的是哪些点mm赢，也就是yy输。
17 yy输就是要找最大匹配的非必要点，所以这题就转化为了找出所有二分图最大匹配的非必要点。
18
19 gym 102832 H 2020ccpc长春
20 题意：
21 有一个最多5位数的密码锁，现在告诉你初始状态，以及有n个数不能达到，到达就输，现在Alice先手，
       Bob后手，每个人都是最优策略，问你最后谁能赢？
22 思路：
23 因为密码锁动一位，位的数字和的奇偶性就改变，就可以用二分图来表示之间的转换关系，想到二分图博
       弈。
24 因为Alice先手，如果Alice待在最大匹配的必要点，就是WIN。
25 注意建图过程，比较烦!
```

1.5.2 bzoj 1443 JSOI2009

```
bzoj 1443 JSOI2009
题意:
有一个(100*100)的矩形，现在矩形格子里的点有两种状态，一种是可以走，还有一种是不能走，现在yy先手，mm后手，问这个矩形中有哪些点让yy先手，结果是mm赢。
思路:
看清题意，规定yy先手，问你的是哪些点mm赢，也就是yy输。
yy输就是要找最大匹配的非必要点，所以这题就转化为了找出所有二分图最大匹配的非必要点。

const int N=(int)1e4+100;
vvi G(N);
int n,m,lk[N],vis[N],idx[110][110],dir[4][2]={{-1,0},{1,0},{0,1},{0,-1}};
int chk[N]; //判断某个点是否为最大匹配的非必要点
int visNum=0; //用在匈牙利里判断这次找增光路时有没有访问过，不用每次初始化vis数组,实验证明也没快多少..快了100ms以内
char mat[110][110];
bool gao(int x){ //匈牙利
   for(int to:G[x]){
      if(vis[to]!=visNum){
         vis[to]=visNum;
         if(lk[to]==-1 || gao(lk[to])){
            lk[x]=to;
            lk[to]=x;
            return true;
         }
      }
   }
   return false;
}
void dfs(int x){ //找出哪些点不是最大匹配的必要点
   chk[x]=1;
   for(int to:G[x]){
     if(!chk[lk[to]]){//注意是lk[to]
         dfs(lk[to]);
      }
   }
}
int main() {
   ms(lk,-1); ms(idx,0); ms(chk,0);
   cin>>n>>m;
   int tot=0;//二分图中点的下标计数
   for(int i=1; i<=n; ++i){
      for(int j=1; j<=m; ++j){
         cin>>mat[i][j];
         if(mat[i][j]=='.') idx[i][j]=++tot;
      }
   }
   for(int i=1; i<=n; ++i){
      for(int j=1; j<=m; ++j){
         if(mat[i][j]!='.') continue;
         for(int k=0; k<4; ++k){
            int tx=i+dir[k][0];
            int ty=j+dir[k][1];
```

```
51              if(tx<1 || tx>n || ty<1 || ty>m || mat[tx][ty]!='.') continue;
52              G[idx[i][j]].eb(idx[tx][ty]);
53              G[idx[tx][ty]].eb(idx[i][j]);
54          }
55       }
56    }
57    int hungary=0;
58    for1(i, tot){
59       ++visNum;
60       if(lk[i]==-1 && gao(i)) ++hungary;
61    }
62    if(hungary*2==tot){
63       cout<<"LOSE"<<'\n';
64       return 0;
65    }
66    for(int i=1; i<=n; ++i){
67       for(int j=1; j<=m; ++j){
68          //没被匹配并且没被确定不是必要点的那些点
69          if(mat[i][j]=='.' && lk[idx[i][j]]==-1 && !chk[idx[i][j]]){
70             dfs(idx[i][j]);
71          }
72       }
73    }
74    cout<<"WIN"<<'\n';
75    for(int i=1; i<=n; ++i){
76       for(int j=1; j<=m; ++j){
77          if(mat[i][j]=='.' && chk[idx[i][j]]){
78             cout<<i<<' '<<j<<'\n';
79          }
80       }
81    }
82    return 0;
83 }
```

1.6 2-SAT

1.6.1 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
61  int n,m,a,ai,b,bi;
62  int main() {
63      cin>>n>>m;
64      init(n+5);
65      for(int i=0; i<m; ++i){
66          cin>>a>>ai>>b>>bi;
67          --a;--b;
```

```
68        a*=2;b*=2; //*2!!
69        if(ai==0 && bi==0){
70            addEdge(a^1,b);
71            addEdge(b^1,a);
72        } else if(ai==0 && bi==1){
73            addEdge(a^1,b^1);
74            addEdge(b,a);
75        } else if(ai==1 && bi==0){
76            addEdge(a,b);
77            addEdge(b^1,a^1);
78        } else{
79            addEdge(a,b^1);
80            addEdge(b,a^1);
81        }
82    }
83    twoSat(n);
84    return 0;
85 }
```

1.6.2 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){
```

```
  // memset(vis,false,sizeof(vis));
    for(int i=0; i<2*n; i+=2){
        if(vis[i] || vis[i^1]) continue;
        top=0;
        if(!dfs(i)){
            while(top){
                vis[stk[--top]]=false;
            }
            if(!dfs(i^1)) return false;
        }
    }
    return true;
}
int main() {
    while(cin>>n>>m){
        init(n); //多组数据要清空
        for(int i=0; i<m; ++i){
            int u,v;
            cin>>u>>v;
            --u; --v;
            addEdge(u,v^1);
            addEdge(v,u^1);
        }
        if(twoSat(n)){
            for(int i=0; i<2*n; ++i) if(vis[i]) cout<<i+1<<'\n';
        } else{
            cout<<"NIE"<<'\n';
        }
    }
    return 0;
}
```

1.6.3 思路

```
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。
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->y表示如果选择x就必须选择y。
总结一下连边的规律：（用i'表示i的反面）
1. i，j必须同时选，那么就有i->j，j->i。
2. i，j不能同时选，那么就有i->j',j->i'。
3. i，j至少选一个，那么就有i'->j,j'->i。
4. 必须选i，那么就有i'->i。
```

1.6.4 经典例题

```
UVA 11930
题意：
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。
1000ms n<=1000, -1e9<=xi,yi<=1e9
思路：
预处理出2*n条对角线，然后遍历。
```

```
1. 同一个矩形的对角线continue,
2. i与j不相交,i与j^1相交 -> 选i一定选j ->i向j连边。
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
sgn和叉积那边要开ll(490ms) or 直接#define int long long(430ms 还快一点??)

cf 27D
问题:
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
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)
环的编号是顺序的1~n,输入时已经保证每个seg的l < r。
4<=n<=100,1<=m<=100,1<=ai,bi<=n,ai!=bi
思路:
下面的是判线段相交的代码
bool within(int x,int y,int z){
   return (x<z && z<y);
}
bool intersect(int i1,int i2){
   if(seg[i1].l==seg[i2].l || seg[i1].l==seg[i2].r) return false;
   if(seg[i1].r==seg[i2].l || seg[i1].r==seg[i2].r) return false;
    return (within(seg[i2].l,seg[i2].r,seg[i1].l)!=within(seg[i2].l,seg[i2].r,seg[i1].r));
}

POJ 3678
题意:
给你m个限制a,b,c(OR,AND,XOR)，表示a点和b点(操作)后的结果是c，点的值只能是0或1,问你有没有可行解。
思路:
如果操作是 OR
** c=0, a true -> a false, b true -> b false
  c=1, a false -> b true, b false -> a true
如果操作是 AND
  c=0, a true -> b false, b true -> a false
** c=1, a false -> a true, b false -> b true
如果操作是 XOR
  c=0, a false -> b false, b false -> a false
       a true -> b true, b true -> a true
  c=1, a false -> b true, b true -> a false
       a true -> b false, b false -> a true
```

1.6.5 UVA 11930

```
UVA 11930
题意:
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。
1000ms n<=1000, -1e9<=xi,yi<=1e9
思路:
预处理出2*n条对角线，然后遍历。
1. 同一个矩形的对角线continue,
2. i与j不相交,i与j^1相交 -> 选i一定选j ->i向j连边。
```

```
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
sgn和叉积那边要开ll(490ms) or 直接#define int long long(430ms 还快一点??)

const int N=(int)1e5;
int scc, top, tot;
vector<int> G[N];
int low[N], dfn[N], belong[N];
int stk[N], vis[N];
void init(int n) {
    for (int i = 0; i <= 2*n; ++i) {
        G[i].clear();
        low[i] = 0;
        dfn[i] = 0;
        stk[i] = 0;
        vis[i] = 0;
    }
    scc = top = tot = 0;
}
void tarjan(int x) {
    stk[top++] = x;
    low[x] = dfn[x] = ++tot;
    vis[x] = 1;
    for (int to : G[x]) {
        if (!dfn[to]) {
            tarjan(to);
            low[x] = min(low[x], low[to]);
        } else if (vis[to]) low[x] = min(low[x], dfn[to]);
    }
    if (low[x] == dfn[x]) {
        ++scc;
        int temp;
        do {
            temp = stk[--top];
            belong[temp] = scc;
            vis[temp] = 0;
        } while (temp != x);
    }
}
void twoSat(int n) {
    for (int i=0; i<2*n; ++i) {
        if (!dfn[i]) tarjan(i);
    }
    for (int i=0; i<2*n; i+=2) {
        if (belong[i] == belong[i^1]) {
            cout<<"NO"<<'\n';
            return;
        }
    }
    cout<<"YES"<<'\n';
}
void addEdge(int a, int b) {
    G[a].emplace_back(b);
}
```

```
int n,x11,y11,x22,y22,x33,y33,x44,y44;
int sgn(i64 x){
    if(x==0LL) return 0;
    if(x>0LL) return 1;
    else return -1;
}
struct Point{
    i64 x,y;
    Point(){}
    Point(i64 _x,i64 _y){
        x=_x;
        y=_y;
    }
    bool operator < (Point b) const{
        return x==b.x?y<b.y:x<b.x;
    }
    Point operator - (const Point &b) const{
        return Point(x-b.x,y-b.y);
    }
    i64 operator * (const Point &b) const{
        return (x*b.x+y*b.y);
    }
    i64 operator ^ (const Point &b) const{
        return (x*b.y-y*b.x);
    }
};
struct Line{
    Point s,e;
    Line() {}
    Line(Point _s,Point _e){
        s=_s;
        e=_e;
    }
    int segcrossseg(Line v){
        int d1=sgn((e-s)^(v.s-s));
        int d2=sgn((e-s)^(v.e-s));
        int d3=sgn((v.e-v.s)^(s-v.s));
        int d4=sgn((v.e-v.s)^(e-v.s));
        if((d1^d2)==-2 && (d3^d4)==-2) return 2;
        return (d1==0 && sgn((v.s-s)*(v.s-e))<=0) ||
        (d2==0 && sgn((v.e-s)*(v.e-e))<=0) ||
        (d3==0 && sgn((s-v.s)*(s-v.e))<=0) ||
        (d4==0 && sgn((e-v.s)*(e-v.e))<=0);
    }
}l[N];
signed main() {
    while(cin>>n){
        if(!n) break;
        int tot=0;
        forn(i, n){
            vector<Point> temp;
            temp.clear();
            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 }
```

1.6.6 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
```

```cpp
int n,m;
struct SE{
   int l,r;
   SE(){}
   SE(int _l,int _r){
      l=_l;
      r=_r;
   }
}seg[N];
int scc, top, tot;
vector<int> G[N];
int low[N], dfn[N], belong[N];
int stk[N], vis[N];
void init(int n) {
   for (int i = 0; i <= 2*n; ++i) {
      G[i].clear();
      low[i] = 0;
      dfn[i] = 0;
      stk[i] = 0;
      vis[i] = 0;
   }
   scc = top = tot = 0;
}
void tarjan(int x) {
   stk[top++] = x;
   low[x] = dfn[x] = ++tot;
   vis[x] = 1;
   for (int to : G[x]) {
      if (!dfn[to]) {
         tarjan(to);
         low[x] = min(low[x], low[to]);
      } else if (vis[to]) low[x] = min(low[x], dfn[to]);
   }
   if (low[x] == dfn[x]) {
      ++scc;
      int temp;
      do {
         temp = stk[--top];
         belong[temp] = scc;
         vis[temp] = 0;
      } while (temp != x);
   }
}
void twoSat(int n) {
   for (int i=0; i<2*n; ++i) {
      if (!dfn[i]) tarjan(i);
   }
   for (int i=0; i<2*n; i+=2) {
      if (belong[i] == belong[i^1]) {
         cout<<"Impossible"<<'\n';
         return;
      }
   }
```

```
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 }
```

1.7 强连通

1.7.1 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 }
```

1.8 双连通

1.8.1 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]);
```

```
        }
        if (x == pre && son > 1) {
            cut[x] = 1;
            add_block[x] = son-1;
        }
        if (low[x] == dfn[x]) {
            ++bcc;
            int temp;
            do {
                temp = stk[--top];
                belong[temp] = bcc;
            } while (temp != x);
        }
    }
    void solve(int _n, vector<int> E[]) {
        n = _n;
        for (int i = 1; i <= n; ++i) {
            belong[i]=0;
            G[i] = E[i];
            low[i] = dfn[i] = stk[i] = fa[i] = 0;
            cut[i] = add_block[i] = 0;
        }
        bcc = top = tot = 0;
        bridge.clear();
        for (int i = 1; i <= n; ++i) if (!dfn[i]) dfs(i, i);
    }
    void rebuild(vector<int> E[]) {
        for (int i = 1; i <= n; ++i) E[i].clear();
        for (int i = 1; i <= n; ++i) {
            int t = fa[i];
            if (belong[i] != belong[t]) {
                E[belong[i]].push_back(belong[t]);
                E[belong[t]].push_back(belong[i]);
            }
        }
    }
}bcc;
int n, m;

int main() {
    cin >> n >> m;
    //多组数据别忘情况E[i]
    for(int i=0; i<=n; ++i) E[i].clear(); //初始化
    forn(i, m) {
        int x, y;
        cin >> x >> y; //顶点编号从1~n
        E[x].eb(y);
        E[y].eb(x);
    }
    bcc.solve(n, E);
    vi res;
    for1(i, n) if (bcc.cut[i]) res.eb(i);
    cout << SZ(res) << '\n';
```

```
    for (int x : res) cout << x << ' ';
    cout << '\n';
    return 0;
}
```

1.8.2 割点桥 v2

```
const int N = EDIT+100; //点数
vector<int> E[N];
int n,m;
struct BCC {
    int n, bcc, top, tot;
    vector<int> G[N];
    vector< pair<int, int> > bridge;
    int low[N], dfn[N], belong[N], fa[N];
    int stk[N];
    int cut[N], final_block[N]; //final_block[i]表示割掉i以后图中有几个连通块
    int cntBlock; //一开始有几个连通块

    void dfs(int x, int pre) {
        stk[top++]=x;
        low[x]=dfn[x]=++tot;
        fa[x]=pre;
        int son=0;
        for (int to:G[x]) {
            if (to==pre) continue;
            if (!dfn[to]) {
                ++son;
                dfs(to, x);
                low[x]=min(low[x],low[to]);
                if (x!=pre && low[to]>=dfn[x]) {
                    cut[x]= 1;
                    final_block[x]++;
                }
                if (low[to]>dfn[x]) bridge.emplace_back(make_pair(x, to));
            }
            else if(dfn[to]<dfn[x]) low[x]=min(low[x],dfn[to]);
        }
        if (x==pre) {
            cut[x]=1;
            final_block[x]=son-1;
        }
        if (low[x]==dfn[x]) {
            ++bcc;
            int temp;
            do {
                temp=stk[--top];
                belong[temp]=bcc;
            } while (temp!=x);
        }
    }
    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]=final_block[i]=0;
52        }
53        bcc=top=tot=cntBlock= 0;
54        bridge.clear();
55        for (int i=1; i<=n; ++i){
56           if (!dfn[i]){
57              ++cntBlock;
58              dfs(i, i);
59           }
60        }
61        for(int i=1; i<=n; ++i) final_block[i]+=cntBlock;
62     }
63     void rebuild(vector<int> E[]) {
64        for (int i=1; i<=n; ++i) E[i].clear();
65        for (int i=1; i<=n; ++i){
66           int t=fa[i];
67           if (belong[i]!=belong[t]) {
68              E[belong[i]].emplace_back(belong[t]);
69              E[belong[t]].emplace_back(belong[i]);
70           }
71        }
72     }
73  }bcc;
74  int main() {
75     cin>>n>>m;
76     forn(i, m){
77        int u,v;
78        cin>>u>>v;
79        E[u].eb(v);
80        E[v].eb(u);
81     }
82     bcc.solve(n,E);
83     for1(i, n) cout<<bcc.final_block[i]<<' ';
84     cout<<'\n';
85     return 0;
86  }
```

1.8.3 割点桥 v3

```
1  const int N=(int)1e4+100;
2  int n,m;
3  vector<int> E[N],sccv[N];
4  struct BCC {
5     int n, bcc, top, top2, tot, cntCut, sccIdx; //sccIdx:点双联通分量编号
6     vector<int> G[N];
7     vector< pair<int, int> > bridge;
8     int low[N], dfn[N], belong[N], fa[N], val[N], sz[N];
```

```
9     //val[i]表示i点tarjan出去多少个**多的(懂得都懂)**点
10    //sz[i]表示编号为i的点双里有多少条边
11    int stk[N],stk2[N];
12    int cut[N], final_block[N]; //final_block[i]表示割掉i以后图中有几个连通块
13    int cntBlock; //一开始有几个连通块
14
15    void dfs(int x, int pre) {
16        stk[top++]=x; stk2[top2++]=x;
17        low[x]=dfn[x]=++tot;
18        fa[x]=pre;
19        int son=0;
20        for (int to:G[x]) {
21            if (to==pre) continue;
22            if (!dfn[to]) {
23                ++son;
24                dfs(to, x);
25                low[x]=min(low[x],low[to]);
26                if (x!=pre && low[to]>=dfn[x]) {
27                    ++cntCut;
28                    cut[x]= 1;
29                    final_block[x]++;
30                }
31                if(low[to]>=dfn[x]){
32                    int temp;
33                    ++sccIdx;
34                    sccv[sccIdx].clear();
35                    sz[sccIdx]=0;//记录当前点双里面的边数
36                    do{ //记录当前点双里面的点
37                        temp=stk[--top];
38                        sccv[sccIdx].eb(temp);
39                        sz[sccIdx]+=val[temp];
40                    }while(temp!=to);
41                    sccv[sccIdx].eb(x);
42                    sz[sccIdx]+=int(sccv[sccIdx].size())-1;
43                }
44                if (low[to]>dfn[x]) bridge.emplace_back(make_pair(x, to));
45            } else if(dfn[to]<dfn[x]){
46                low[x]=min(low[x],dfn[to]);
47                ++val[x]; //注意这里，用于后面处理点双连通里面有多少边
48            }
49        }
50        if (x==pre) {
51            final_block[x]=son-1;
52            if(son>1){
53                ++cntCut;
54                cut[x]=1;//gym 102835
55            }
56        }
57        if(low[x] == dfn[x]){
58            ++bcc;
59            int temp;
60            do{
61                temp=stk2[--top2];
```

```
62              belong[temp]=bcc;
63          }while(temp!=x);
64      }
65  }
66  void solve(int _n, vector<int> E[]) {
67      n=_n;
68      for (int i=1; i<=n; ++i) {
69          val[i]=sz[i]=0; //:)
70          belong[i]=0;
71          G[i]=E[i];
72          low[i]=dfn[i]=stk[i]=fa[i] = 0;
73          cut[i]=final_block[i]=0;
74      }
75      bcc=top=top2=tot=cntBlock=cntCut=sccIdx=0;
76      bridge.clear();
77      for (int i=1; i<=n; ++i){
78          if (!dfn[i]){
79              ++cntBlock;
80              dfs(i, i);
81          }
82      }
83      for(int i=1; i<=n; ++i) final_block[i]+=cntBlock;
84  }
85  void rebuild(vector<int> E[]) {
86      for (int i=1; i<=n; ++i) E[i].clear();
87      for (int i=1; i<=n; ++i){
88          int t=fa[i];
89          if (belong[i]!=belong[t]) {
90              E[belong[i]].emplace_back(belong[t]);
91              E[belong[t]].emplace_back(belong[i]);
92          }
93      }
94  }
95  }bcc;
96  int main() {
97      scanf("%d",&tc);
98      while(tc--){
99          scanf("%d%d",&n,&m);
100         for(int i=0; i<=n+5; ++i){
101             E[i].clear();
102             sccv[i].clear();
103         }
104         forn(i, m){
105             int u,v;
106             scanf("%d%d",&u,&v);
107             ++u; ++v; //如果下标从0开始计数的话
108             E[u].eb(v);
109             E[v].eb(u);
110         }
111         bcc.solve(n,E);
112     }
113     return 0;
114 }
```

1.8.4 船新版本

```
https://ac.nowcoder.com/acm/contest/7501/D
牛客小米icpc邀请赛第一场 D
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。
//时间复杂度O(V+E)
//建边建双向边
//顶点编号1~n
//add_block[i]表示割掉i后图中的连通块数

const int N = EDIT+100; //点数
vector<int> E[N];
struct BCC {
    int n, bcc, top, tot, sum;
    vector<int> G[N];
    vector< pair<int, int> > bridge;
    int low[N], dfn[N], belong[N], fa[N];
    int stk[N];
    int cut[N], add_block[N];

    void dfs(int x, int pre) {
        stk[top++] = x;
        low[x] = dfn[x] = ++tot;
        fa[x] = pre;
        int son = 0;
        for (int to : G[x]) {
            if (to == pre) continue;
            if (!dfn[to]) {
                ++son;
                dfs(to, x);
                low[x] = min(low[x], low[to]);
                if (x != pre && low[to] >= dfn[x]) cut[x] = 1;
                if(low[to]>=dfn[x]) ++add_block[x];
                if (low[to]>dfn[x]) bridge.push_back(make_pair(x, to));
            }
            else if(dfn[to] < dfn[x]) low[x] = min(low[x], dfn[to]);
        }
        if (x == pre && son > 1) cut[x] = 1;
        if (low[x] == dfn[x]) {
            ++bcc;
            int temp;
            do {
                temp = stk[--top];
                belong[temp] = bcc;
            } while (temp != x);
        }
        if(x!=pre) ++add_block[x];
    }
    void solve(int _n, vector<int> E[]) {
        n = _n;
        for (int i = 0; i <= n; ++i) { //0~n-1
            G[i] = E[i];
            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;
```

1.8.5 思路

```
1.有割点不一定有桥，有桥一定存在割点
2.桥一定是割点依附的边。

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
联通性，即不存在桥，则称作边双联通图。

边双连通分量：无向图中，删除任意边仍然能联通的块
那么再此分量中的边一定不是桥
同时，不在此分量中的边一定是桥

所以用这个可以用来判桥
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，
且其中环与环不保证有公共边，但至少有一个公共点。

的连通性，即不存在割点，则称作点双连通图。
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
```

1.8.6 经典例题

```
洛谷 T103492
题意：
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）
```

n<=5e4, m<=3e5
思路:
无，直接看代码。

hdu 3836
题意:
最少加几条边使得整张图scc。
思路:
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的点的个数分别为out0和in0,答案就是max(out0,in0)。

poj 2186
题意:
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）
思路:
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。(把这个点加入答案)

poj 2553
题意:
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。
思路:
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。

poj 3352 (一开始就连通)
题意:
最少加几条边使整张图边bcc。
思路:
前提：当前图已经连通。
先求出边bcc然后对边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 条边就能使整条边bcc。

```
for1(i, bcc.bcc) if(deg[i]==1) ++sum;
sum=(sum+1)/2;
printf("Case %d: %d\n",kase++,sum);
```

UVA 10972 (一开始不连通)
题意:
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？
思路:
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。

poj 2942 边双连通+染色法判二分图
题意:
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。
圈要满足如下条件:
1.人数大于1。
2.总人数为奇数。
3.有仇恨的骑士不能挨着坐。
问有几个骑士不能和任何人形成任何的圆圈？
思路:
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。
52
53  p1407 稳定婚姻问题
54  题意:
55  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。
56  n<=4000, m<=20000.
57  思路:
58  强连通分量。
59  先连正房之间的关系，女->男。
60  再连二奶之间的关系，男->女。
61  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。
62  在同一个强连通分量中，就说明不稳定。否则稳定。
63
64  LightOJ 1308 割点+分类讨论
65  题意:
66  t组数据，每组数据给你一个n和一个m，表示有n个点和m条双向边，保证原图没有重边，没有自环，连
        通。现在规定要炸一个点，被炸以后每个点都要往地表冲，现在问你至少要加多少通往地表的通道，
        能达成无论炸什么点，所有点都有通往地表的通道。让你输出最少加的通道数和在加最少通道数这个
        前提下的方案数。
67  t<=30, 2<=n<=10000, 0<=m<=20000, 0<=u,v<n
68  思路:
69  如果原图没有割点，那就最少要加两条通道，如果被炸了一个点还可以走另一个点，方案数是Cn2 = n*(
        n-1)/2。
70  如果有割点，就开始O(n)走完每个点连通分量，如果一个点双里面有 >1 的割点那就不用加通道，因为
        如果炸了一边还有另一边可以走，所以对答案没贡献。如果一个点双里面只有一个割点，设这个点双
        里面除了割点有num个点，那就最少要加的通道+1,方案数*num。
71  坑点/技巧:
72  结果要求对 2^64 取模，其实就是要求结果开unsigned long long，输出%llu。
73  用set存一个连通分量里的割点。
```

1.8.7 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>
```

```cpp
#include <climits>
#define mp make_pair
#define fi first
#define se second
#define pb push_back
#define eb emplace_back
#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define forn(i, n) for (int i = 0; i < (int)(n); ++i)
#define for1(i, n) for (int i = 1; i <= (int)(n); ++i)
#define ford(i, a, b) for (int i = (int)(a); i >= (int)b; --i)
#define fore(i, a, b) for (int i = (int)(a); i <= (int)(b); ++i)
#define rep(i, l, r) for (int i = (l); i <= (r); i++)
#define per(i, r, l) for (int i = (r); i >= (l); i--)
#define ms(x, y) memset(x, y, sizeof(x))
#define SZ(x) int(x.size())
using namespace std;
typedef pair<int, int> pii;
typedef vector<int> vi;
typedef vector<pii> vpi;
typedef vector<vi> vvi;
typedef long long i64;
typedef vector<i64> vi64;
typedef vector<vi64> vvi64;
typedef pair<i64, i64> pi64;
typedef double ld;
template<class T> bool uin(T &a, T b) { return a > b ? (a = b, true) : false; }
template<class T> bool uax(T &a, T b) { return a < b ? (a = b, true) : false; }
//1.integer overflow (1e5 * 1e5) (2e9 + 2e9)
//2.runtime error
//3.boundary condition
const int N = 1000+100; //点数
int n,m,mep[N][N];
vector<int> E[N];
struct BCC {
    int n, bcc, top, tot;
    vector<int> G[N];
    vector< pair<int, int> > bridge;
    int low[N], dfn[N], belong[N], fa[N];
    int stk[N];
    int cut[N], add_block[N];
    int col[N], vis[N];

    bool findOddCircle(int now,int c){ //找奇环
        col[now]=c;
        for(int i=0; i<int(G[now].size()); ++i){
            int to=G[now][i];
            if(belong[now]!=belong[to]) continue;
            if(col[to]==-1){
                if(findOddCircle(to,c^1)) return true;
            }
            if(col[to]==col[now]) return true;
        }
```

```
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 }
```

1.8.8 UVA 10972

```cpp
const int N = 1000+100; //点数
int n,m;
vector<int> E[N];
struct BCC {
    int n, bcc, top, tot;
    vector<int> G[N];
    vector< pair<int, int> > bridge;
    int low[N], dfn[N], belong[N], fa[N];
    int stk[N];
    int cut[N], add_block[N];
    int deg[N];

    void dfs(int x, int pre) {
        stk[top++] = x;
        low[x] = dfn[x] = ++tot;
        fa[x] = pre;
        int son = 0;
        for (int to : G[x]) {
            if (to == pre) continue;
            if (!dfn[to]) {
                ++son;
                dfs(to, x);
                low[x] = min(low[x], low[to]);
                if (x != pre && low[to] >= dfn[x]) {
                    cut[x] = 1;
                    add_block[x]++;
                }
                if (low[to] > dfn[x]) bridge.push_back(make_pair(x, to));
            }
            else if(dfn[to] < dfn[x]) low[x] = min(low[x], dfn[to]);
        }
        if (x == pre && son > 1) {
            cut[x] = 1;
            add_block[x] = son-1;
        }
        if (low[x] == dfn[x]) {
            ++bcc;
            int temp;
            do {
                temp = stk[--top];
                belong[temp] = bcc;
            } while (temp != x);
        }
    }
    void solve(int _n, vector<int> E[]) {
        n = _n;
        for (int i = 1; i <= n; ++i) {
            belong[i]=0;
            G[i] = E[i];
            low[i] = dfn[i] = stk[i] = fa[i] = 0;
            cut[i] = add_block[i] = 0;
            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 }
```

1.8.9 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<=5e4, m<=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;
```

```
vector<int> E[N];
struct BCC {
    int n, bcc, pbcc, top, ptop, tot;
    vector<int> G[N];
    vector<int> ans[N];
    vector< pair<int, int> > bridge;
    int low[N], dfn[N], belong[N], fa[N], pbelong[N];
    int stk[N], pstk[N];
    int cut[N], add_block[N];

    void dfs(int x, int pre) {
        stk[top++] = x;
        pstk[ptop++] = x;
        low[x] = dfn[x] = ++tot;
        fa[x] = pre;
        int son = 0;
        if(SZ(E[x])==0){
            ans[++pbcc].eb(x);
            //pbelong[x]=pbcc;
            return;
        }
        for (int to : G[x]) {
            if (to == pre) continue;
            if (!dfn[to]) {
                ++son;
                dfs(to, x);
                low[x] = min(low[x], low[to]);
                if (x != pre && low[to] >= dfn[x]) {
                    cut[x] = 1;
                    add_block[x]++;
                }
                if(low[to]>=dfn[x]){
                    ++pbcc;
                    int temp;
                    do{
                        temp=pstk[--ptop];
                        ans[pbcc].eb(temp);
                // pbelong[temp]=pbcc;
                    } while(temp!=to);
                    ans[pbcc].eb(x);
                }
                if (low[to] > dfn[x]) bridge.push_back(make_pair(x, to));
            }
            else if(dfn[to] < dfn[x]) low[x] = min(low[x], dfn[to]);
        }
        if (x == pre && son > 1) {
            cut[x] = 1;
            add_block[x] = son-1;
        }
        if (low[x] == dfn[x]) {
            ++bcc;
            int temp;
            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 }
```

1.8.10 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。
```

```
n<=4000, m<=20000.
思路:
强连通分量。
先连正房之间的关系，女->男。
再连二奶之间的关系，男->女。
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。
在同一个强连通分量中，就说明不稳定。否则稳定。

const int N = 8000+100; //点数
//有n对，每对有两个，所以要乘2
int n,m,idx=0;
map<string,int> hs;
int scc, top, tot;
vector<int> G[N];
int low[N], dfn[N], belong[N];
int stk[N], vis[N];
void init(int n) {
    for (int i = 0; i <= 2*n; ++i) {
        G[i].clear();
        low[i] = 0;
        dfn[i] = 0;
        stk[i] = 0;
        vis[i] = 0;
    }
    scc = top = tot = 0;
}
void tarjan(int x) {
    stk[top++] = x;
    low[x] = dfn[x] = ++tot;
    vis[x] = 1;
    for (int to : G[x]) {
        if (!dfn[to]) {
            tarjan(to);
            low[x] = min(low[x], low[to]);
        } else if (vis[to]) {
            low[x] = min(low[x], dfn[to]);
        }
    }
    if (low[x] == dfn[x]) {
        ++scc;
        int temp;
        do {
            temp = stk[--top];
            belong[temp] = scc;
            vis[temp] = 0;
        } while (temp != x);
    }
}

int main() {
    cin>>n;
    init(2*n+5); //有n对，每对有两个，所以要乘2
    string s1,s2;
```

```
    for(int i=0; i<n; ++i){
        cin>>s1>>s2;
        if(!hs[s1]) hs[s1]=idx++;
        if(!hs[s2]) hs[s2]=idx++;
        G[hs[s1]].eb(hs[s2]);
    }
    cin>>m;
    forn(i,m){
        cin>>s1>>s2;
        G[hs[s2]].eb(hs[s1]);
    }
    for(int i=0; i<idx; ++i) if(!dfn[i]) tarjan(i);
    bool ok=true;
    for(int i=0; i<idx; i+=2){
        if(belong[i]!=belong[i+1]){
            cout<<"Safe"<<'\n';
        } else{
            cout<<"Unsafe"<<'\n';
        }
    }
    return 0;
}
```

1.8.11 gym 102835

```
The 2020 ICPC Asia Taipei-Hsinchu Site Programming Contest
cf gym 102835
problem I Critical Structures
题意:
给你一个n个点，m条边的无向图，保证没有自环和重边，现在让你求四个值。
割点数目，桥数目，点双连通分量数目(设这个值为a),包含最多边的点双的边数(设这个值为b)，a和b要除以他们的gcd
思路:
tarjan，主要是求点双连通分量数目,包含最多边的点双的边数这两个之前没做过，更新了一波bcc模板，详情见代码
3<=n<=1000, n-1<=m<=n*(n-1)/2


const int N=(int)1e3+100;
vector<int> E[N],sccv[N];
int n,m;
struct BCC {
    int n, bcc, top, tot, cntCut, sccIdx; //sccIdx:点双联通分量编号
    vector<int> G[N];
    vector< pair<int, int> > bridge;
    int low[N], dfn[N], belong[N], fa[N], val[N], sz[N];
    //val[i]表示i点tarjan出去多少个点
    //sz[i]表示编号为i的点双里有多少条边
    int stk[N];
    int cut[N], final_block[N]; //final_block[i]表示割掉i以后图中有几个连通块
    int cntBlock; //一开始有几个连通块

```

```
26  void dfs(int x, int pre) {
27      stk[top++]=x;
28      low[x]=dfn[x]=++tot;
29      fa[x]=pre;
30      int son=0;
31      for (int to:G[x]) {
32          if (to==pre) continue;
33          if (!dfn[to]) {
34              ++son;
35              dfs(to, x);
36              low[x]=min(low[x],low[to]);
37              if (x!=pre && low[to]>=dfn[x]) {
38                  cut[x]= 1;
39                  final_block[x]++;
40              }
41              if(low[to]>=dfn[x]){
42                  int temp;
43                  ++sccIdx;
44                  sccv[sccIdx].clear();
45                  sz[sccIdx]=0;//记录当前点双里面的边数
46                  do{ //记录当前点双里面的点
47                      temp=stk[--top];
48                      sccv[sccIdx].eb(temp);
49                      sz[sccIdx]+=val[temp]; //操作1
50                  }while(temp!=to);
51                  sccv[sccIdx].eb(x);
52                  sz[sccIdx]+=int(sccv[sccIdx].size())-1; //操作2
53              }
54              if (low[to]>dfn[x]) bridge.emplace_back(make_pair(x, to));
55          } else if(dfn[to]<dfn[x]){
56              low[x]=min(low[x],dfn[to]);
57              ++val[x]; //注意这里，用于后面处理点双连通里面有多少边
58              //val[x]用来储存x节点除了一个儿子还有多少个儿子
59          }
60      }
61      if (x==pre) {
62          final_block[x]=son-1;
63          if(son>1){
64              cut[x]=1;//gym 102835
65          }
66      }
67  }
68  void solve(int _n, vector<int> E[]) {
69      n=_n;
70      for (int i=1; i<=n; ++i) {
71          val[i]=sz[i]=0; //:)
72          belong[i]=0;
73          G[i]=E[i];
74          low[i]=dfn[i]=stk[i]=fa[i] = 0;
75          cut[i]=final_block[i]=0;
76      }
77      bcc=top=tot=cntBlock=cntCut=sccIdx=0;
78      bridge.clear();
```

```
79          for (int i=1; i<=n; ++i){
80              if (!dfn[i]){
81                  ++cntBlock;
82                  dfs(i, i);
83              }
84          }
85          for(int i=1; i<=n; ++i) final_block[i]+=cntBlock;
86      }
87      void rebuild(vector<int> E[]) {
88          for (int i=1; i<=n; ++i) E[i].clear();
89          for (int i=1; i<=n; ++i){
90              int t=fa[i];
91              if (belong[i]!=belong[t]) {
92                  E[belong[i]].emplace_back(belong[t]);
93                  E[belong[t]].emplace_back(belong[i]);
94              }
95          }
96      }
97  }bcc;
98  int main() {
99      int tc;
100     cin>>tc;
101     while(tc--){
102         cin>>n>>m;
103         for(int i=0; i<=n; ++i){
104             E[i].clear();
105             sccv[i].clear();//?
106         }
107         forn(i, m){
108             int a,b;
109             cin>>a>>b;
110             E[a].eb(b);
111             E[b].eb(a);
112         }
113         bcc.solve(n,E);
114         for1(i, n) if(bcc.cut[i]) bcc.cntCut++;
115         cout<<bcc.cntCut<<' '<<int(bcc.bridge.size())<<' ';
116         int bb=-1; //最多边的那个点双的边数
117         for(int i=1; i<=bcc.sccIdx; ++i) bb=max(bb, bcc.sz[i]);
118         int aa=bcc.sccIdx; //有几个点双
119         int temp=__gcd(aa,bb);
120         aa/=temp;
121         bb/=temp;
122         cout<<aa<<' '<<bb<<'\n';
123     }
124     return 0;
125 }
```

1.8.12 LightOJ 1308

```
1  LightOJ 1308 割点+分类讨论
2  题意:
```

t组数据，每组数据给你一个n和一个m，表示有n个点和m条双向边，保证原图没有重边，没有自环，连通。现在规定要炸一个点，被炸以后每个点都要往地表冲，现在问你至少要加多少通往地表的通道，能达成无论炸什么点，所有点都有通往地表的通道。让你输出最少加的通道数和在加最少通道数这个前提下的方案数。

t<=30, 2<=n<=10000, 0<=m<=20000, 0<=u,v<n

思路:

如果原图没有割点，那就最少要加两条通道，如果被炸了一个点还可以走另一个点，方案数是Cn2 = n*(n-1)/2。

如果有割点，就开始O(n)走完每个点连通分量，如果一个点双里面有 >1 的割点那就不用加通道，因为如果炸了一边还有另一边可以走，所以对答案没贡献。如果一个点双里面只有一个割点，设这个点双里面除了割点有num个点，那就最少要加的通道+1,方案数*num。

坑点/技巧:

结果要求对 2^64 取模，其实就是要求结果开**unsigned long long**，输出%llu。

用set存一个连通分量里的割点。

```
const int N=(int)1e4+100;
int tc,ans,vis[N];
int n,m;
i64 num;
set<int> meet_cut;
vector<int> E[N],sccv[N];
struct BCC {
    int n, bcc, top, top2, tot, cntCut, sccIdx; //sccIdx:点双联通分量编号
    vector<int> G[N];
    vector< pair<int, int> > bridge;
    int low[N], dfn[N], belong[N], fa[N], val[N], sz[N];
    //val[i]表示i点tarjan出去多少个**多的(懂得都懂)**点
    //sz[i]表示编号为i的点双里有多少条边
    int stk[N],stk2[N];
    int cut[N], final_block[N]; //final_block[i]表示割掉i以后图中有几个连通块
    int cntBlock; //一开始有几个连通块

    void dfs(int x, int pre) {
        stk[top++]=x; stk2[top2++]=x;
        low[x]=dfn[x]=++tot;
        fa[x]=pre;
        int son=0;
        for (int to:G[x]) {
            if (to==pre) continue;
            if (!dfn[to]) {
                ++son;
                dfs(to, x);
                low[x]=min(low[x],low[to]);
                if (x!=pre && low[to]>=dfn[x]) {
                    ++cntCut;
                    cut[x]= 1;
                    final_block[x]++;
                }
                if(low[to]>=dfn[x]){
                    int temp;
                    ++sccIdx;
                    sccv[sccIdx].clear();
                    sz[sccIdx]=0;//记录当前点双里面的边数
```

```
50                do{ //记录当前点双里面的点
51                    temp=stk[--top];
52                    sccv[sccIdx].eb(temp);
53                    sz[sccIdx]+=val[temp];
54                }while(temp!=to);
55                sccv[sccIdx].eb(x);
56                sz[sccIdx]+=int(sccv[sccIdx].size())-1;
57            }
58            if (low[to]>dfn[x]) bridge.emplace_back(make_pair(x, to));
59        } else if(dfn[to]<dfn[x]){
60            low[x]=min(low[x],dfn[to]);
61            ++val[x]; //注意这里，用于后面处理点双连通里面有多少边
62        }
63    }
64    if (x==pre) {
65        final_block[x]=son-1;
66        if(son>1){
67            ++cntCut;
68            cut[x]=1;//gym 102835
69        }
70    }
71    if(low[x] == dfn[x]){
72        ++bcc;
73        int temp;
74        do{
75            temp=stk2[--top2];
76            belong[temp]=bcc;
77        }while(temp!=x);
78    }
79  }
80  void solve(int _n, vector<int> E[]) {
81    n=_n;
82    for (int i=1; i<=n; ++i) {
83        val[i]=sz[i]=0; //:)
84        belong[i]=0;
85        G[i]=E[i];
86        low[i]=dfn[i]=stk[i]=fa[i] = 0;
87        cut[i]=final_block[i]=0;
88    }
89    bcc=top=top2=tot=cntBlock=cntCut=sccIdx=0;
90    bridge.clear();
91    for (int i=1; i<=n; ++i){
92        if (!dfn[i]){
93            ++cntBlock;
94            dfs(i, i);
95        }
96    }
97    for(int i=1; i<=n; ++i) final_block[i]+=cntBlock;
98  }
99  void rebuild(vector<int> E[]) {
100   for (int i=1; i<=n; ++i) E[i].clear();
101   for (int i=1; i<=n; ++i){
102       int t=fa[i];
```

```
103             if (belong[i]!=belong[t]) {
104                 E[belong[i]].emplace_back(belong[t]);
105                 E[belong[t]].emplace_back(belong[i]);
106             }
107         }
108     }
109 }bcc;
110 void dfs(int x){
111     vis[x]=1;
112     ++num;
113     for(int to:E[x]){
114         if(bcc.cut[to]){
115             meet_cut.insert(to);
116             continue;
117         }
118         if(vis[to]) continue;
119 
120         dfs(to);
121     }
122 }
123 int main() {
124     scanf("%d",&tc);
125     int kase=1;
126     while(tc--){
127         ans=1;
128         scanf("%d%d",&n,&m);
129         for(int i=0; i<=n+5; ++i){
130             E[i].clear();
131             sccv[i].clear();
132             vis[i]=0;
133         }
134         forn(i, m){
135             int u,v;
136             scanf("%d%d",&u,&v);
137             ++u; ++v;
138             E[u].eb(v);
139             E[v].eb(u);
140         }
141         bcc.solve(n,E);
142         int cnt_shaft=0;
143         unsigned long long ans=1;
144         if(!bcc.cntCut){
145             printf("Case %d: %d %d\n",kase++,2,n*(n-1)/2);
146         } else{
147             for1(i, n){
148                 if(vis[i] || bcc.cut[i]) continue;
149                 num=0;
150                 meet_cut.clear();
151                 dfs(i);
152                 if(SZ(meet_cut)==1){
153                     ans=ans*num;
154                     ++cnt_shaft;
155                 }
```

```
156             }
157             printf("Case %d: %d %llu\n",kase++,cnt_shaft,ans);
158         }
159     }
160     return 0;
161 }
```

1.9 欧拉回路

1.9.1 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  }
```

1.9.2 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。
```

1.9.3 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。
```

1.9.4 cf 21D

```
cf 21D
题意:
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。
就是找** 一条路径可重复的欧拉回路 **。
input:
cin>>n>>m;
for(int i=0; i<m; ++i) cin>>x>>y>>z;
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
1<=n<=15,0<=m<=2000,1<=x,y<=n,1<=w<=10000
思路:
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。
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。

const int N=100;
const int INF=0x3f3f3f3f;
int n,m,x,y,w,dis[N][N],deg[N],dp[(1<<20)];
int main() {
   cin>>n>>m;
   for(int i=0; i<=n+5; ++i){
      deg[i]=0;
      for(int j=0; j<=n+5; ++j) dis[i][j]=INF;
   }
   int sum=0;
   for(int i=0; i<m; ++i){
      cin>>x>>y>>w;
      uin(dis[x][y],w);
      uin(dis[y][x],w);
      ++deg[x]; ++deg[y];
      sum+=w;
   }
   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]);
   for(int i=2; i<=n; ++i){
      if(deg[i]>0 && dis[i][1]==INF){ //条件一定要有deg[i]>0,因为这题要求的是经过所有边,不是所有点。
         cout<<-1<<'\n';
         return 0;
      }
   }
   int now=0;
   for(int i=1; i<=n; ++i) if(deg[i]&1) now|=(1<<(i-1));
   for(int i=0; i<now+10; ++i) dp[i]=INF;
   dp[now]=0;
   int ans=INT_MAX;
   for(int i=now; i>0; --i){
      for(int j=1; j<=n; ++j){
         if(i&(1<<(j-1))){
```

```
            for(int k=j+1; k<=n; ++k){
                if(i&(1<<(k-1))){
                    uin(dp[i^(1<<(j-1))^(1<<(k-1))],dp[i]+dis[j][k]);
                }
            }
        }
    }
}
cout<<sum+dp[0]<<'\n';
return 0;
}
```

1.10 LCA

1.10.1 ST 表

```
//预处理O(nlogn) 在线查询O(1)
const int maxn = 2*EDIT+100; //要开两倍点数量的大小（欧拉序长度）
struct LCA
{
    #define type int
    struct node{int to;type w;node(){}node(int _to,type _w):to(_to),w(_w){}};
    type dist[maxn];
    int path[maxn],dep[maxn],loc[maxn],len[maxn],LOG[maxn],all,n;
    int dp[25][maxn], point[25][maxn]; //2^20 == 1e6 2^25 == 3e7
    vector<node> G[maxn];
    void dfs(int u, int now) {
        path[++all] = u;
        loc[u] = all;
        dep[all] = now;
        for (node cur : G[u]) {
            int v = cur.to;
            if (loc[v]) continue;
            len[v] = now+1;
            dist[v] = dist[u]+cur.w;
            dfs(v, now+1);
            path[++all] = u;
            dep[all] = now;
        }
    }
    void initRMQ(int n)
    {
        LOG[0] = -1;
        for (int i = 1; i <= all; ++i) {
            dp[0][i] = dep[i];
            point[0][i] = path[i];
            LOG[i] = ((i&(i-1)) == 0 ? LOG[i-1]+1 : LOG[i-1]);
        }
        for (int i = 1; (1<<i) <= all; ++i) {
            for (int j = 1; j+(1<<i)-1 <= all; ++j) {
              if (dp[i-1][j] < dp[i-1][j+(1<<(i-1))]) {
                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 }
```

1.10.2 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  }
```

1.11 最大团

1.11.1 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() {
```

```
    while (cin >> n) {
        if (!n) break;
        ans = 0; //初始化
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                int x;
                cin >> x;
                g[i][j] = g[j][i] = x;
            }
        }
        int k, j;
        for (int i = n - 1; ~i; dfs(k, 1), ma[i--] = ans) {
            for (k = 0, j = i + 1; j < n; ++j) {
                if (g[i][j]) f[1][k++] = j;
            }
        }
        cout << ans << '\n';
    }
    return 0;
}
```

1.11.2 常见思路

```
给你一个无向图G,
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。
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。
1.最大独立集不是唯一的.
2.性质: 无向图的最大团 == 该无向图补图的最大独立集
```

1.12 拓扑排序

1.12.1 toposort

```
bool toposort(vvi &g, vi &inDeg) {
    queue<int> q;
    while (!q.empty()) q.pop();
    forn(i, n) {
        if (inDeg[i] == 0) {
            q.push(i);
        }
    }
    int cnt = 0;
    while (!q.empty()) {
        int now = q.front();
        q.pop();
        ++cnt;
        for (auto it : g[now]) {
            --inDeg[it];
            if (inDeg[it] == 0) q.push(it);
        }
    }
    if (cnt == n) return true;
```

```
    return false;
}

int main() {
    cin >> t;
    while (t--) {
        cin >> n >> m;
        vvi g(n);
        vi inDeg(n, 0);
        forn(i, m) {
            int e1, e2;
            cin >> e1 >> e2;
            --e1; --e2;
            g[e1].eb(e2);
            ++inDeg[e2];
        }
        if (toposort(g, inDeg)) cout << "Correct" << '\n';
        else cout << "Wrong" << '\n';
    }
    return 0;
}
```

1.13 相关题集

1.13.1 牛逼的图论题

```
cf 1364D
题目:
给你n,m,k，表示n个点，m条边，现在让你搞出满足下面两组之一的集合。
1. 一组不相关的点集，点集的大小为 k/2 向上取整。
2. 找出大小 <=k 的圆。

思路:
如果是一颗树，那么条件1肯定可以满足。
输入边的时候只需要处理1~k这些点的边，如果里面有环，那环的大小肯定<=k，符合条件2，不然就是颗
    树，满足条件1。
```

1.13.2 cf 1364D

```
// cf 1364D
#include <bits/stdc++.h>

using namespace std;

const int maxn = (int)1e5+100;
int n, m, k, h[maxn], fa[maxn];
vector< vector<int> > G(maxn);

void dfs(int x, int par) {
    fa[x] = par;
    for (int to : G[x]) {
        if (to == par) continue;
```

```
14          if (!h[to]) {
15              h[to] = h[x] + 1;
16              dfs(to, x);
17          } else {
18              vector<int> ans = {x}; //找到了环
19              int now = x;
20              while (now != to) {
21                  now = fa[now];
22                  ans.emplace_back(now);
23              }
24              cout << 2 << '\n';
25              cout << int(ans.size()) << '\n';
26              for (int x : ans) cout << x << ' ';
27              cout << '\n';
28              exit(0);
29          }
30      }
31  }
32
33  int main() {
34
35      cin >> n >> m >> k;
36      for (int i = 0; i < m; ++i) {
37          int u, v;
38          cin >> u >> v;
39          if (u > k || v > k) continue;
40          G[u].emplace_back(v);
41          G[v].emplace_back(u);
42      }
43      memset(h, 0, sizeof(h));
44      for (int i = 1; i <= k; ++i) {
45          if (!h[i]) {
46              h[i] = 1;
47              dfs(i, -1);
48          }
49      }
50      vector<int> team[2];
51      for (int i = 1; i <= k; ++i) { //是一棵树，找符合条件1的点
52          team[h[i] & 1].emplace_back(i);
53      }
54      if (int(team[0].size()) < int(team[1].size())) swap(team[0], team[1]);
55      cout << 1 << '\n';
56      for (int i = 0; i < (k + 1) / 2; ++i) { //注意这里是输出到(k+1)/2为止
57                                              //不能输出全部，比如有6个点但没有边的情况
58          cout << team[0][i] << ' ';
59      } cout <<'\n';
60      return 0;
61  }
```

2 计算几何

2.1 点

```
const double eps = 1e-5;
const double inf = 1e20;
const double pi = acos(-1.0);
const int maxp = 1010;
//Compares a double to zero
int sgn(double x) {
    if (fabs(x) < eps) return 0;
    if (x < 0) return -1;
    else return 1;
}
//square of a double
inline double sqr(double x) {return x * x;}
struct Point {
    double x, y;
    Point() {}
    Point(double _x, double _y) {
        x = _x;
        y = _y;
    }
    void input() {scanf("%lf%lf", &x, &y);}
    void output() {printf("%.8f %.8f\n", x, y);}
    bool operator == (Point b) const {return sgn(x-b.x)==0 && sgn(y-b.y)==0;}
    bool operator < (Point b) const {return sgn(x-b.x)==0 ? sgn(y-b.y)<0:x<b.x;}
    Point operator - (const Point &b) const {return Point(x-b.x, y-b.y);}
    Point operator + (const Point &b) const {return Point(x+b.x, y+b.y);}
    Point operator * (const double &k) const {return Point(x*k, y*k);}
    Point operator / (const double &k) const {return Point(x/k, y/k);}
    double operator * (const Point &b) const {return x*b.x+y*b.y;}//点积
    double operator ^ (const Point &b) const {return x*b.y-y*b.x;}//叉积
    double len() {return hypot(x, y);}//返回长度
    double len2() {return x*x + y*y;}//返回长度的平方
    double distance(Point p) {return hypot(x - p.x, y - p.y);}//返回两点的距离
    double distance2(Point p) {return (x-p.x)*(x-p.x)+(y-p.y)*(y-p.y);}//返回两点距离
        的平方

    //计算 pa 和 pb 的夹角
    //就是求这个点看a, b的夹角
    //测试LightOJ 1203
    double rad(Point a, Point b) {
        Point p = *this;
        return fabs(atan2(fabs((a - p) ^ (b - p)), (a - p) * (b - p)));
    }
    //化为长度为r的向量
    Point trunc(double r) {
        double l = len();
        if (!sgn(l)) return *this;
        r /= l;
        return Point(x * r, y * r);
    }
```

```
49      Point rotleft() {return Point(-y, x);}//逆时针旋转 90 度
50      Point rotright() {return Point(y, -x);}//顺时针旋转 90 度
51      //绕着 p 点逆时针旋转 angle
52      Point rotat(Point p, double angle) {
53          Point v = (*this) - p;
54          double c = cos(angle), s = sin(angle);
55          return Point(p.x + v.x * c - v.y * s, p.y + v.x * s + v.y * c);
56      }
57  };
58  double cross(Point A,Point B,Point C){return (B-A)^(C-A);}//叉积
59  double dot(Point A,Point B,Point C){return (B-A)*(C-A);}//点积
```

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) {return (s == v.s) && (e == v.e);}
9       //根据一个点和倾斜角 angle 确定直线，0 <= angle < pi
10      Line (Point p, double angle) {
11          s = p;
12          if (sgn(angle - pi / 2) == 0) {
13              e = (s + Point(0, 1));
14          } else {
15              e = (s + Point(1, tan(angle)));
16          }
17      }
18      //ax + by + c == 0
19      Line(double a, double b, double c) {
20          if (sgn(a) == 0) {
21              s = Point(0, -c / b);
22              e = Point(1, -c / b);
23          } else if (sgn(b) == 0) {
24              s = Point(-c / a, 0);
25              e = Point(-c / a, 1);
26          } else {
27              s = Point(0, -c / b);
28              e = Point(1, (-c - a) / b);
29          }
30      }
31      void input() { s.input();e.input();}
32      void adjust() {if (e < s) swap(s, e);}
33      //求线段长度
34      double length() {return s.distance(e);}
35      //返回直线倾斜角
36      double angle() {
37          double k = atan2(e.y - s.y, e.x - s.x);
38          if (sgn(k) < 0) k += pi;
```

```
39      if (sgn(k - pi) == 0) k -= pi;
40      return k;
41  //点和直线关系 要保证s.y>e.y，如<要swap
42  }
43  //1 线在点的左侧
44  //2 线在点的右侧
45  //3 点在直线上
46  int relation(Point p) {
47      int c = sgn((p - s) ^ (e - s));
48      if (c < 0) return 1;
49      else if (c > 0) return 2;
50      else return 3;
51  }
52  //点在线段上的判断
53  bool pointonseg(Point p) {return sgn((p - s) ^ (e-s)) == 0 && sgn((p - s) * (p -
        e)) <= 0;}
54  //两向量平行（对应直线平行或重合）
55  bool parallel(Line v) {return sgn((e - s) ^ (v.e - v.s)) == 0;}
56  //两线段相交判断
57  //2 规范相交
58  //1 非规范相交
59  //0 不相交
60  int segcrossseg(Line v) {
61      int d1 = sgn((e - s) ^ (v.s - s));
62      int d2 = sgn((e - s) ^ (v.e - s));
63      int d3 = sgn((v.e - v.s) ^ (s - v.s));
64      int d4 = sgn((v.e - v.s) ^ (e - v.s));
65      if ((d1 ^ d2) == -2 && (d3 ^ d4) == -2) return 2;
66      return (d1 == 0 && sgn((v.s - s) * (v.s - e)) <= 0) ||
67          (d2 == 0 && sgn((v.e - s) * (v.e - e)) <= 0) ||
68          (d3 == 0 && sgn((s - v.s) * (s - v.e)) <= 0) ||
69          (d4 == 0 && sgn((e - v.s) * (e - v.e)) <= 0);
70  }
71  //直线和线段相交判断
72  //-*this line -v seg
73  //2 规范相交
74  //1 非规范相交
75  //0 不相交
76  int linecrossseg(Line v) {
77      int d1 = sgn((e - s) ^ (v.s - s));
78      int d2 = sgn((e - s) ^ (v.e - s));
79      if ((d1 ^ d2) == -2) return 2;
80      return (d1 == 0 || d2 == 0);
81  }
82  //两直线关系
83  //0 平行
84  //1 重合
85  //2 相交
86  int linecrossline(Line v) {
87      if ((*this).parallel(v))
88          return v.relation(s) == 3;
89      return 2;
90  }
```

```
91      //求两直线的交点
92      //要保证两直线不平行或重合
93      Point crosspoint(Line v) {
94          double a1 = (v.e - v.s) ^ (s - v.s);
95          double a2 = (v.e - v.s) ^ (e - v.s);
96          return Point((s.x * a2 - e.x * a1) / (a2 - a1),
97                       (s.y * a2 - e.y * a1) / (a2 - a1));
98      }
99      //点到直线的距离
100     //测试 : cf614C
101     double dispointtoline(Point p) {return fabs((p - s) ^ (e - s)) / length();}
102     //点到线段的距离
103     //测试 : cf614C
104     double dispointtoseg(Point p) {
105         if (sgn((p - s) * (e - s)) < 0 || sgn((p - e) * (s - e)) < 0)
106             return min(p.distance(s), p.distance(e));
107         return dispointtoline(p);
108     }
109     //返回线段到线段的距离
110     //前提是两线段不相交，相交距离就是0了
111     double dissegtoseg(Line v) {
112         return min(min(dispointtoseg(v.s), dispointtoseg(v.e)),
113                    min(v.dispointtoseg(s), dispointtoseg(e)));
114     }
115     //返回p在直线上的投影
116     Point lineprog(Point p) {return s + ( ((e - s) * ((e - s) * (p - s))) / ((e - s)
            .len2()) );}
117     //返回点p关于直线的对称点
118     Point symmetrypoint(Point p) {
119         Point q = lineprog(p);
120         return Point(2 * q.x - p.x, 2 * q.y - p.y);
121     }
122 };
```

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      }
```

```
    //周长
    double circumference(){
        return 2*pi*r;
    }
    //两圆的关系
    //5 相离
    //4 外切
    //3 相交
    //2 内切
    //1 内含
    //需要Point的distance
    //测试: UVA12304
    int relationcircle(circle v){
        double d=p.distance(v.p);
        if(sgn(d-r-v.r)>0) return 5;
        if(sgn(d-r-v.r)==0) return 4;
        double l=fabs(r-v.r);
        if(sgn(d-r-v.r)<0 && sgn(d-l)>0) return 3;
        if(sgn(d-l)==0) return 2;
        if(sgn(d-l)<0) return 1;
    }
    //求两圆相交的面积
    //cf 600D 毒瘤题, 要开long double
    double areacircle(circle v){
        int rel=relationcircle(v);
        if(rel>=4) return 0.0;
        if(rel<=2) return min(area(),v.area());
        double d=p.distance(v.p);
        double a1=2*acos((1.0*r*r+d*d-1.0*v.r*v.r)/(2.0*r*d));
        double a2=2*acos((1.0*v.r*v.r+d*d-1.0*r*r)/(2.0*v.r*d));
        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;
        return ans;
    }
};
```

2.4 多边形

```
struct polygon{
    int n;
    Point p[maxp];
    Line l[maxp];
    void input(int _n){
        n=_n;
        for(int i=0; i<n; ++i) p[i].input();
        //半平面交有时要加上reverse(p,p+n)把它们变成逆时针顺序
    }
    void add(Point q){
        p[n++]=q;
    }
    void getline(){
        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 LightOJ1203 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)
                    top--;
57              convex.p[++top]=p[i];
58          }
59          if(convex.n==2 && (convex.p[0]==convex.p[1])) convex.n--;//特判
60          convex.norm(); //原来的得到的是顺时针的点，排序后逆时针。
61      }
62      //得到凸包的另外一种方法
63      //测试 LightOJ1203 LightOJ1239
64      void Graham(polygon &convex){
65          norm();
66          int &top=convex.n;
```

```
67          top=0;
68          if(n==1){
69              top=1;
70              convex.p[0]=p[0];
71              return;
72          }
73          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){
```

```
            for(;cross(p[i+1],p[j+1],p[i])>cross(p[i+1],p[j],p[i]);j=(j+1)%n);
            ans=max(ans, max(p[i].distance(p[j]),p[i+1].distance(p[j])));//最远点对
        }
        return ans;
    }
}
//得到周长
//测试 LightOj1239
double getcircumference(){
    double sum=0;
    for(int i=0; i<n; ++i){
        sum+=p[i].distance(p[(i+1)%n]);
    }
    return sum;
}
//得到面积
double getarea(){
    double sum=0;
    for(int i=0; i<n; ++i) sum+=(p[i]^p[(i+1)%n]);
    return fabs(sum)/2;
}
};
```

2.5 半平面交

```
//半平面交 O(nlogn)
//测试 POJ3335 POJ1474 POJ1279
struct halfplane:public Line{
    double angle;
    halfplane(){}
    //表示向量s->e逆时针(左侧)的半平面
    //所以要把所有点都变为逆时针排序
    halfplane(Point _s,Point _e){
        s=_s;
        e=_e;
    }
    halfplane(Line v){
        s=v.s;
        e=v.e;
    }
    void calcangle(){
        angle=atan2(e.y-s.y,e.x-s.x);
    }
    bool operator <(const halfplane &b) const{
        return angle<b.angle;
    }
};
struct halfplanes{
    int n; //别忘了给n赋值!
    halfplane hp[maxp];
    Point p[maxp];
    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)^
59                (p[ed]-hp[que[st]].s))<0) --ed;
60          while(st<ed && sgn((hp[que[ed]].e-hp[que[ed]].s)^
61                             (p[st+1]-hp[que[ed]].s))<0) ++st;
62          if(st+1>=ed) return false;
63          return true;
64      }
65      //得到最后半平面交得到的凸多边形
66      //需要先调用halfplaneinsert()且返回true
67      void getconvex(polygon &con){
68          p[st]=hp[que[st]].crosspoint(hp[que[ed]]);
69          con.n=ed-st+1;
70          for(int j=st,i=0; j<=ed; ++i,++j)
71              con.p[i]=p[j];
72      }
73  };
74
75  scanf("%d",&n);
76  plo.input(n);
77  plo.getline();
78  hfp.n=n;
79  for(int i=0; i<n; ++i){
80      hfp.hp[i]=plo.l[i];
```

```
}
if(hfp.halfplaneinsert()) puts("YES");
else puts("NO");
```

2.6 function

```
bool sameline(Point a, Point b, Point c){ //判断三点是否共线
    double ax=c.x-a.x, ay=c.y-a.y;
    double bx=c.x-b.x, by=c.y-b.y;
    if(sgn(ax*by-ay*bx)==0)return true;
    return false;
}
bool squ(Point a,Point b,Point c,Point d){ //判断正方形 cf136D
    vector<double> temp;
    double l11=a.distance(b); temp.eb(l11);
    double l22=b.distance(c); temp.eb(l22);
    double l33=c.distance(d); temp.eb(l33);
    double l44=d.distance(a); temp.eb(l44);
    double l55=a.distance(c); temp.eb(l55);
    double l66=b.distance(d); temp.eb(l66);
    sort(all(temp));
    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;
    return false;
}
bool rec(Point a,Point b,Point c,Point d){ //判断矩形 cf136D
    Point ct=Point((a.x+b.x+c.x+d.x)/4,(a.y+b.y+c.y+d.y)/4); //求质心
    double l11=ct.distance(a);
    double l22=ct.distance(b);
    double l33=ct.distance(c);
    double l44=ct.distance(d);
    if(!sgn(l11-l22) && !sgn(l11-l33) && !sgn(l11-l44)) return true;
    return false;
}
```

2.7 注意点

```
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. 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. 要多注意n=1,2.. 这种小值时候是否要特判
4. 对于某些题，要注意所有点共线的情况
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)
如
const double eps=1e-8;
int tmp=hypot(x,y)+eps;
if(tmp*tmp==x*x+y*y){
    puts("black");
    return 0;
}
```

```
或者
const double eps=1e-8;
double d2=floor(dist+eps);
if(!sgn(dist-d2)){
    puts("black");
    return 0;
}
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ll(UVA
    11930),不放心就#define int long long
```

2.8 常出现的模型

```
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
对于凸多边形，看一个点和旁边的点的叉积就行了
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. 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

```
POJ 1228 数据水
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。
所以求完凸包后的每条边上至少要有三个点
题意:
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？
既判断这n个点连成的多边形是不是稳定凸包？
思路:
1. 如果点数<6，那么肯定不是稳定凸包。
2. 如果所有点都共线，那么肯定不是稳定凸包。
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。

signed main() {
    int tc;
    scanf("%d",&tc);
    while(tc--){
        scanf("%d",&n);
        plo.input(n);
        if(n<6){
            puts("NO");
            continue;
        }
        bool oneline=true;
        for(int i=0; i<n-2; ++i){
```

```
            if(!sameline(plo.p[i],plo.p[i+1],plo.p[i+2])){
                oneline=false;
                break;
            }
        }
        if(oneline){
            puts("NO");
            continue;
        }
        plo.Graham(cv);
        bool ok=true;
        cv.getline();
        for(int i=0; i<cv.n; ++i){
            bool exec=false;
            for(int j=0; j<plo.n; ++j){
                if(cv.l[i].s==plo.p[j]) continue;
                if(cv.l[i].e==plo.p[j]) continue;
                if(sameline(cv.l[i].s,cv.l[i].e,plo.p[j])){
                    exec=true;
                    break;
                }
            }
            if(!exec){
                ok=false;
                break;
            }
        }
        puts(ok?"YES":"NO");
    }
    return 0;
}
```

2.11 旋转卡壳

2.11.1 SCOI2007 最大土地面积

```
1. SCOI 2007 最大土地面积 / cf gym102460 L
//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？
//凸包后枚举对锺点O(n^2)，然后旋转卡壳，因为是在枚举对锺点里面卡壳，所以卡壳的复杂度是O(1)，总复杂度O(n^2)
//n=2000, x,y是double，x,y<=1e5 0.2s
#define i64 long long
const double eps = 1e-8;
const double inf = 1e20;
const double pi = acos(-1.0);
const int maxp = 40100;
int n;
//Compares a double to zero
int sgn(double x) {
    if (fabs(x) < eps) return 0;
    if (x < 0) return -1;
    else return 1;
```

```
}
struct Point {
    double x, y;
    int idx;
    Point() {}
    Point(double _x, double _y, int _idx) {
        x = _x;
        y = _y;
        idx = _idx;
    }
    void input() {
        scanf("%lf%lf", &x, &y);
    }
    void output() {
        printf("%.8f %.8f\n", x, y);
    }
    bool operator == (Point b) const {
        return sgn(x - b.x) == 0 && sgn(y - b.y) == 0;
    }
    bool operator < (Point b) const {
        return sgn(x - b.x) == 0 ? sgn(y - b.y) < 0 : x < b.x;
    }
    Point operator - (const Point &b) const {
        return Point(x - b.x, y - b.y, idx - b.idx);
    }
    //点积
    double operator * (const Point &b) const {
        return x * b.x + y * b.y;
    }
    //叉积
    double operator ^ (const Point &b) const {
        return x * b.y - y * b.x;
    }
    //返回两点的距离
    double distance(Point p) {
        return hypot(x - p.x, y - p.y);
    }
}fp;
struct polygon{
    int n;
    Point p[maxp];
    void input(int _n){
        n=_n;
        for(int i=0; i<n; ++i){
            p[i].input();
            p[i].idx=i;
        }
    }
    void add(Point q){
        p[n++]=q;
    }
    struct cmp{
        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)
                top--;
108         convex.p[++top]=p[i];
109     }
110     if(convex.n==2 && (convex.p[0]==convex.p[1])) convex.n--;//特判
111     convex.norm(); //原来的得到的是顺时针的点，排序后逆时针。
112 }
113 bool OK(Point a,Point b,Point c,Point d)
114 {
115     double A=fabs((b-a)^(c-a));
116     double B=fabs((b-a)^(d-a));
117     return A<B;
118 }
119 void Rotating_Calipers(polygon &plo)
120 {
```

```
121         double ans=0,now;
122         if (n<=2) ans=0;
123         else if (n==3)
124         {
125             now=~(1LL<<63); //LLONG_MAX
126             ans=abs((p[0]-p[2])^(p[1]-p[2]));
127             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("%.31f\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;
```

```cpp
const double pi = acos(-1.0);
const int maxp = 40100;
int n;
//Compares a double to zero
int sgn(double x) {
    if (fabs(x) < eps) return 0;
    if (x < 0) return -1;
    else return 1;
}
struct Point {
    i64 x, y;
    int idx;
    Point() {}
    Point(i64 _x, i64 _y, int _idx) {
        x = _x;
        y = _y;
        idx = _idx;
    }
    void input() {
        scanf("%lld%lld", &x, &y);
    }
    void output() {
        printf("%.8f %.8f\n", x, y);
    }
    bool operator == (Point b) const {
        return sgn(x - b.x) == 0 && sgn(y - b.y) == 0;
    }
    bool operator < (Point b) const {
        return sgn(x - b.x) == 0 ? sgn(y - b.y) < 0 : x < b.x;
    }
    Point operator - (const Point &b) const {
        return Point(x - b.x, y - b.y, idx - b.idx);
    }
    //点积
    //叉积
    i64 operator ^ (const Point &b) const {
        return x * b.y - y * b.x;
    }
    //返回两点的距离
    double distance(Point p) {
        return hypot(x - p.x, y - p.y);
    }
}fp;
struct polygon{
    int n;
    Point p[maxp];
    void input(int _n){
        n=_n;
        for(int i=0; i<n; ++i){
            p[i].input();
            p[i].idx=i;
        }
    }
```

```
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 LightOJ1203 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];
```

```
    seg[node].cover=0;
    seg[node].len=seg[node].more=0;
    if(r-l<=1) return;
    int mid=(l+r)>>1;
    build(node<<1,l,mid);
    build(node<<1|1,mid,r);
}
void update(int node,double ul,double ur,int nowState){ //注意这两个double!
    if(node>maxNode) return;
    double l=seg[node].l,r=seg[node].r; //注意这两个double!
    if(ul<=l && r<=ur){
        seg[node].cover+=nowState;
        pushUp(node);
        return;
    }
    if(ul<seg[node<<1].r) update(node<<1,ul,ur,nowState);
    if(ur>seg[node<<1|1].l) update(node<<1|1,ul,ur,nowState);
    pushUp(node);
}
int main() {
    int tc;
    scanf("%d",&tc);
    while(tc--){
        maxNode=-1;
        scanf("%d",&n);
        for(int i=1; i<=n; ++i){
            double a,b,c,d;
            scanf("%lf%lf%lf%lf",&a,&b,&c,&d);
            l[i]=LI(a,b,d,1);
            l[i+n]=LI(c,b,d,-1);
            v[i]=b;
            v[i+n]=d;
        }
        sort(l+1,l+1+2*n);
        sort(v+1,v+1+2*n,cmp);
        build(1,1,2*n);
        double sum=0;
        for(int i=1; i<=2*n; ++i){
            sum+=seg[1].more*(l[i].x-l[i-1].x);
            update(1,l[i].y11,l[i].y22,l[i].state);
        }
        printf("%.2f\n",sum);
    }
    return 0;
}
```

2.12.2 面积并坐标 double 版

```
const int maxn = 2*(int)1e6+1000;
int n;
double v[maxn];
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(nlgn)

```
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 y11,y22,state;
9     LI(i64 _x=0, int _y11=0, int _y22=0, int _state=0){
10       x=_x;
11       y11=_y11;
12       y22=_y22;
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;
```

```
    }
    if(ul<seg[node<<1].r) update(node<<1,ul,ur,nowState);
    if(ur>seg[node<<1|1].l) update(node<<1|1,ul,ur,nowState);
    pushUp(node);
}
int main() {
    maxNode=-1;
    scanf("%d",&n);
    for(int i=1; i<=n; ++i){
        i64 a,c;
        int b,d;
        scanf("%lld%d%lld%d",&a,&b,&c,&d);
        l[i]=LI(a,b,d,1);
        l[i+n]=LI(c,b,d,-1);
        v[i]=b;
        v[i+n]=d;
    }
    sort(l+1,l+1+2*n);
    sort(v+1,v+1+2*n);
    build(1,1,2*n);
    i64 ans=0;
    for(int i=1; i<=2*n; ++i){
        ans+=seg[1].len*(l[i].x-l[i-1].x);
        update(1,l[i].y11,l[i].y22,l[i].state);
    }
    printf("%lld\n",ans);
    return 0;
}
```

2.12.4 周长并 O(nlgn)

```
const int maxn = 2*(int)1e5+1000;
int n;
int v[maxn];
struct Line {
    int x_;
    int y_, y__;
    int state;
    bool operator < (const Line &oth) const {return x_<oth.x_;}
}line[maxn];
struct s1 {
    int l, r;
    int cover;
    i64 len;
    int lcover, rcover;
    int diff;
}sgt[maxn<<3]; //注意这个大小
void pushUp(int node) {
    if (sgt[node].cover) {
        sgt[node].len = sgt[node].r-sgt[node].l;
        sgt[node].lcover = sgt[node].rcover = 1;
        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;
```

```
    }
    cout << ans << '\n';

    return 0;
}
```

2.13 半平面交

2.13.1 经典例题

```
codechef ALLPOLY
题意:
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。
思路:
傻逼卡精题，不想搞了，后面有过了的代码。
```

2.13.2 codechef ALLPOLY

```
#include <bits/stdc++.h>
using namespace std;
#define N 100111
typedef long double D;
typedef long long L;
#define INF 1e111
#define EPS 1e-11
#define BEPS 1e-6

bool between(D a, D b, D c) {
    return a-BEPS <= b && b <= c+BEPS || a+BEPS >= b && b >= c-BEPS;
}



template<class T>
struct pt {
    T x, y;
    pt() {}
    pt(T x, T y): x(x), y(y) {}
    pt operator+(const pt p) const {
        return pt(x + p.x, y + p.y);
    }
    pt operator-(const pt p) const {
        return pt(x - p.x, y - p.y);
    }
    pt scale(T s) const {
        return pt(x * s, y * s);
    }
    T dot(pt p) {
        return x * p.x + y * p.y;
    }
    T cross(pt p) {
        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     } 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           hull.pop_back();
174        }
175        hull.push_back(s[i]);
176     }
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        hull[j] = segD(p, p + normalize(hull[j].vec()));
183     }
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()))) <
           0.01) {
281          L comp = planecomp(segLs[m-1], segLs[i]);
282          if (comp < 0) {
283             segLs[m-1] = segLs[i];
284          }
285       } else {
286          segLs[m++] = segLs[i];
287       }
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
          (0)
325     #define swapall(s) do { for (int i = 0; i < s.size(); i++) s[i] = s[i].swap(); }
           while (0)
326     #define ROT(x) do { reverse((x).begin(), (x).end()); swapall(x); negall(x); }
          while (0)
327     hull(tops);
328     negall(bots);
329     hull(bots);
330     ROT(bots);
331
332     // combine
333     vector<rayD> rtops, rbots;
334     for (int i = 0; i < tops.size(); i++) rtops.push_back(rayD(tops[i], i == 0, i ==
           tops.size()-1));
335     for (int i = 0; i < bots.size(); i++) rbots.push_back(rayD(bots[i], i == 0, i ==
           bots.size()-1));
336     tie(rtops, rbots); ROT(rtops); ROT(rbots);
337     tie(rtops, rbots); ROT(rtops); ROT(rbots);
338     tops.clear();
339     bots.clear();
340     for (int i = 0; i < rtops.size(); i++) tops.push_back(rtops[i].s);
341     for (int i = 0; i < rbots.size(); i++) bots.push_back(rbots[i].s);
342
```

```
343     // normalize
344     ptD ave = ptD(0, 0);
345     for (int i = 0; i < tops.size(); i++) ave = ave + tops[i].a + tops[i].b;
346     for (int i = 0; i < bots.size(); i++) ave = ave + bots[i].a + bots[i].b;
347     ave = ave.scale(1./(2*(tops.size() + bots.size())));
348     for (int i = 0; i < tops.size(); i++) tops[i] = segD(tops[i].a - ave, tops[i].b
            - 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的连线和所有
       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，交集中的长度就是答案，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<=n<=100000, |x|,|y|<=10000, x*x+y*y>0
13 GNU G++14 6.4.0
14 思路:
15 用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。
```

```
2018 ccpc桂林 H
题意:
给你两条线段，一条白线一条黑线，现在问你有多少区域面积看得到白线看不到黑线，注意黑白线不包括断线，先输入白线两端点再输入黑线两端点。
-1000<=x,y<=1000
思路:
先分三种情况讨论，两条线段规范相交，不规范相交，不相交。
然后里面在细致的讨论。

反思:
1. 两线段ab,cd形成的区域是ac,bd还是ad,bc不能判断ac，ad谁小就确定，可以先随意形成两条线段，如果规范相交（这种情况只有规范相交和不相交两种情况），就不是这两条线段，换另一种就行。
2. 考虑不相交的情况漏考虑了两线段共线的情况。


LightOJ 1208
题目:
给你n个pair<点，点>，表示某个pair里这两个点可以连边，代价是长度*2,现在给你一头野猪的坐标，问你能不能用最小花费把野猪围起来，并且围栏是一个凸多边形，如果可以就输出最小花费，不可以就输出-1。
t<=125,1<=n<=100,-1e4<=x,y<=1e4
思路:
先处理输出边，让野猪在边的右边，然后n^2建边，然后floyd。

LightOJ 1292
题意:
t组数据，每次给你n个点，问你最多多少个点共线。暴力里的sgn要判0才能卡过。。。
时限:4000ms
t<=30,1<=n<=700,-10000<=x,y<=10000
思路:
1. N^3 暴力枚举直线然后数点,不剪枝3500ms，剪了3100ms
2. N^2*log(n) 暴力枚举直线,一边枚举一边记录相同斜率的计数(map) 3100ms
方法2注意点:
1. 不能用map<double,int>记录相同斜率的边的计数，因为在算斜率的时候有可能除以0。
2. 可以用pair<int,int>来描述斜率，要注意两点，1.x<0时要对x,y都取反，2.要除以他们绝对值的gcd
你有可能要问那<0,1>和<0,3>最后不是存的两个点吗，其实不是，因为__gcd(0,x)==__gcd(x,0)==x,所以最后都存成<0,1>这一个点。

LightOJ 1230
题意:
t组数据，每次给你n个条，以及一个矩形的右上角坐标，左下角坐标是(0,0),保证这n条线两两不重合，现在问你依次添加这n条边，最后分成了多少块区域。
t<=100, 0<=n<=100, 5<=L,W<=100(矩形右上角), x1,y1,x2,y2都在矩形的边界上
思路:
每次加线时开始遍历之前加过线，和他们求交点，如果交点是端点或没有端点就对答案贡献+1,有交点就把交点放set里面，最后最答案的贡献就是SZ(set)+1。
```

2.14.2 cf 598C

```
#include<bits/stdc++.h>
```

```
#define double long double //double->long double
using namespace std;
vector<pair<double,int> > a;
int n,ansx,ansy;
double sum=1e5;
int main(){
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        int x,y;
        scanf("%d%d",&x,&y);
        a.push_back({atan2(y,x),i+1});
    }
    sort(a.begin(),a.end());
    a.push_back(a[0]);
    for(int i=0;i<n;i++){
        double x=a[i+1].first-a[i].first;
        if(x<0)x+=2*acos(-1);
        if(x<sum)sum=x,ansx=a[i].second,ansy=a[i+1].second;
    }
    printf("%d %d",ansx,ansy);
    return 0;
}
```

2.14.3 2020ZJ 省赛 H

```
https://codeforces.com/gym/102770
题意:
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的连线和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）。
n<=500, m<=500, -1e4<=x<=1e4, 1<=y<=1e4
思路:
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。
求出每个点遮挡范围之后，对所有点的遮挡范围取交集，交集中的长度就是答案，关键就是如何求线段并和线段集合的交

#include <bits/stdc++.h>
using namespace std;
const int N=510;
const double eps = 1e-3;
const double INF=1e15;
int sgn(double x) {
    if(fabs(x) < eps) return 0;
    if(x < 0) return -1;
    return 1;
}
struct Point {
    double x,y;
    Point() {}
    Point(double _x,double _y) {
        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 }
```

2.14.4 2018 ccpc 桂林 H

```
1  题意:
2  给你两条线段，一条白线一条黑线，现在问你有多少区域面积看得到白线看不到黑线，注意黑白线不包括
       断线，先输入白线两端点再输入黑线两端点。
3  -1000<=x,y<=1000
4  思路:
5  先分三种情况讨论，两条线段规范相交，不规范相交，不相交。
6  然后里面在细致的讨论。
7
8  反思:
9  1. 两线段ab,cd形成的区域是ac,bd还是ad,bc不能判断ac，ad谁小就确定，可以先随意形成两条线段，
       如果规范相交（这种情况只有规范相交和不相交两种情况），就不是这两条线段，换另一种就行。
10 2. 考虑不相交的情况漏考虑了两线段共线的情况。
11 3. 求三角形面积可以用叉积或者海伦公式，推荐叉积，因为叉积精度高，求面积别忘了叉积取绝对值再
       /2。
12
13 int tc;
14 scanf("%d",&tc);
15 int kase=1;
16 while(tc--){
17     scanf("%lf%lf%lf%lf%lf%lf%lf%lf",&po[0].x,&po[0].y,&po[1].x,&po[1].y,&po[2].x,&
           po[2].y,&po[3].x,&po[3].y);
18     Line l11=Line(po[0],po[1]),l22=Line(po[2],po[3]);
19     int rel=l11.segcrossseg(l22);
20     if(rel==2) printf("Case %d: 0.000\n",kase++);
21     else if(rel==1){
22         if(l11.parallel(l22)) printf("Case %d: 0.000\n",kase++);
23         else if(po[0]==po[2] || po[0]==po[3] || po[1]==po[2] || po[1]==po[3]) printf(
               "Case %d: inf\n",kase++);
24         else{
25             Point cp=l11.crosspoint(l22);
26             if(cp==l11.s || cp==l11.e) printf("Case %d: 0.000\n",kase++);
```

```
27          else printf("Case %d: inf\n",kase++);
28       }
29    } else{
30       Point cp=l11.crosspoint(l22);
31       Line newl1,newl2;
32       if(l11.linecrossline(l22)==1) printf("Case %d: 0.000\n",kase++); //共线
33       else if(l11.pointonseg(cp)){
34          printf("Case %d: inf\n",kase++);
35       }
36       else if(l22.pointonseg(cp)) printf("Case %d: 0.000\n",kase++);
37       else{
38          newl1=Line(po[0],po[2]);newl2=Line(po[1],po[3]);
39          if(newl1.segcrossseg(newl2)!=0) {newl1=Line(po[0],po[3]);newl2=Line(po[1],
                po[2]);}
40          Point cp2=newl1.crosspoint(newl2);
41          if(newl1.parallel(newl2)){printf("Case %d: inf\n",kase++);}
42          else{
43             if(l11.relation(cp2)==l11.relation(l22.s)){
44                printf("Case %d: inf\n",kase++);
45             }
46             //else printf("Case %d: %.8f\n",kase++,gao(cp2.distance(l11.s),cp2.
                  distance(l11.e),l11.s.distance(l11.e)));
47             else printf("Case %d: %.8f\n",kase++,fabs((l11.e-cp2)^(l11.s-cp2))/2);
48          }
49       }
50    }
51 }
```

2.14.5 LightOj 1208

```
1  题目:
2  给你n个pair<点，点>，表示某个pair里这两个点可以连边，代价是长度*2,现在给你一头野猪的坐标，
       问你能不能用最小花费把野猪围起来，并且围栏是一个凸多边形，如果可以就输出最小花费，不可以
       就输出-1。
3  t<=125,1<=n<=100,-1e4<=x,y<=1e4
4  思路:
5  先处理输出边，让野猪在边的右边，然后n^2建边，然后floyd。
6
7  const int N=1000;
8  double mat[N][N];
9  int tc,n,x,y;
10 double l[N],dp[N][N];
11 struct vec{
12    int x,y;
13    vec(int _x=0,int _y=0){x=_x,y=_y;}
14    vec operator - (vec v) {return vec(x-v.x,y-v.y);}
15 };
16 struct node{
17    int ix1,iy1,ix2,iy2;
18 }nd[N];
19 map< pii,vector<int> > G;
20 double len(vec v){return hypot(v.x,v.y);}
```

```
int cross(vec a,vec b,vec c){return (a.x-c.x)*(b.y-c.y) - (a.y-c.y)*(b.x-c.x);}
bool is_right(vec a,vec b,vec c){return cross(a,b,c)>=0;}
int main() {
    int kase=1;
    scanf("%d",&tc);
    while(tc--){
        G.clear();
        scanf("%d%d%d",&n,&x,&y);
        for(int i=0; i<=n; ++i)
            for(int j=0; j<=n; ++j)
                dp[i][j]=1e9;
        forn(i, n){
            scanf("%d%d%d%d",&nd[i].ix1,&nd[i].iy1,&nd[i].ix2,&nd[i].iy2);
            if(!is_right(vec(x,y),vec(nd[i].ix2,nd[i].iy2),vec(nd[i].ix1,nd[i].iy1))){
                swap(nd[i].ix1,nd[i].ix2);
                swap(nd[i].iy1,nd[i].iy2);
            }
            G[mp(nd[i].ix1,nd[i].iy1)].eb(i);
            l[i]=len(vec(nd[i].ix2-nd[i].ix1,nd[i].iy2-nd[i].iy1));
        }
        for(int i=0; i<n; ++i){
            for(int j:G[mp(nd[i].ix2,nd[i].iy2)]){
                if(is_right(vec(nd[j].ix2,nd[j].iy2),vec(nd[i].ix2,nd[i].iy2),vec(nd[i
                    ].ix1,nd[i].iy1))){ //因为最后围成的是一个凸多边形
                    dp[i][j]=l[i]+l[j];
                }
            }
        }
        for(int k=0; k<n; ++k){
            for(int i=0; i<n; ++i){
                for(int j=0; j<n; ++j){
                    uin(dp[i][j],dp[i][k]+dp[k][j]);
                }
            }
        }
        printf("Case %d: ",kase++);
        double ans=1e9;
        forn(i, n) uin(ans,dp[i][i]);
        if(ans==1e9) puts("-1");
        else printf("%.7f\n",ans);
    }
    return 0;
}
```

2.14.6 LightOJ 1292

```
sol1:
const double eps=1e-8;
int sgn(double x){
    if(fabs(x) < eps) return 0;
    if(x < 0) return -1;
    return 1;
```

```
}
const int N=710;
int tc,n,m,vis[N][N];
struct vec{
    double x,y;
    vec(double _x=0,double _y=0){x=_x,y=_y;}
    vec operator + (vec v) {return vec(x+v.x,y+v.y);}
    vec operator - (vec v) {return vec(x-v.x,y-v.y);}
    vec operator * (double v) {return vec(x*v,y*v);}
    vec operator / (double v) {return vec(x/v,y/v);}

    double len() {return hypot(x,y);}
    double operator * (vec v) {return x*v.x+y*v.y;}
}po[1000];
double cross(vec a,vec b) {return a.x*b.y - a.y*b.x;}
int main() {
    int kase=1;
    scanf("%d",&tc);
    while(tc--){
        scanf("%d",&n);
        forn(i, n){
            int tx,ty;
            scanf("%d%d",&tx,&ty);
            po[i]=vec(tx,ty);
        }
        int ans;
        if(n<3){
            printf("Case %d: %d\n",kase++,n);
            continue;
        }

        ms(vis,0);
        ans=0;
        for(int i=0; i<n; ++i){
            for(int j=i+1; j<n; ++j){
                if(vis[i][j]) continue;
                int res=2;
                for(int k=j+1; k<n; ++k){
                    if(!sgn(cross(po[i]-po[k],po[j]-po[k]))){
                        ++res;
                        vis[k][i]=vis[k][j]=1;
                        vis[i][j]=vis[i][k]=1;
                        vis[j][i]=vis[j][k]=1;
                    }
                }
                uax(ans,res);
            }
        }
        printf("Case %d: %d\n",kase++,ans);
    }
    return 0;
}

```

```
sol2:
const int N=710;
int tc,n,m,vis[N][N];
struct vec{
    double x,y;
    vec(double _x=0,double _y=0){x=_x,y=_y;}
}po[1000];
int main() {
    int kase=1;
    scanf("%d",&tc);
    while(tc--){
        scanf("%d",&n);
        map< pair<int,int>,int > cnt;
        forn(i, n){
            scanf("%lf%lf",&po[i],&po[i].y);
        }
        if(n<3){
            printf("Case %d: %d\n",kase++,n);
            continue;
        }
        int ans=0;
        for(int i=0; i<n; ++i){
            cnt.clear();
            for(int j=0; j<n; ++j){
                if(i==j) continue;
                int ty=po[j].y-po[i].y;
                int tx=po[j].x-po[i].x;
                if(tx<0) {tx=-tx; ty=-ty;}
                int tgcd=__gcd(abs(tx),abs(ty));
                tx/=tgcd; ty/=tgcd;
                ++cnt[mp(tx,ty)];
                uax(ans,cnt[mp(tx,ty)]+1);
            }
        }
        printf("Case %d: %d\n",kase++,ans);
    }
    return 0;
}
```

2.14.7 LightOJ 1230

```
LightOJ 1230
题意:
t组数据，每次给你n个条，以及一个矩形的右上角坐标，左下角坐标是(0,0),保证这n条线两两不重合，
    现在问你依次添加这n条边，最后分成了多少块区域。
t<=100, 0<=n<=100, 5<=L,W<=100(矩形右上角), x1,y1,x2,y2都在矩形的边界上
思路:
每次加线时开始遍历之前加过线，和他们求交点，如果交点是端点或没有端点就对答案贡献+1,有交点就
    把交点放set里面，最后最答案的贡献就是SZ(set)+1。

const double eps=1e-8;
const int N=200;
```

```
10 int sgn(double x){
11    if(fabs(x) < eps) return 0;
12    if(x < 0) return -1;
13    return 1;
14 }
15 struct vec{
16    double x,y;
17    vec(double _x=0,double _y=0){x=_x;y=_y;}
18
19    vec operator * (double v) {return vec(x*v,y*v);}
20    vec operator / (double v) {return vec(x/v,y/v);}
21    vec operator + (vec v) {return vec(x+v.x,y+v.y);}
22    vec operator - (vec v) {return vec(x-v.x,y-v.y);}
23
24    double operator * (vec v) {return x*v.x+y*v.y;}
25 }p11[N],p22[N];
26 double cross(vec a,vec b) {return a.x*b.y-a.y*b.x;}
27 bool point_on_segment(vec p,vec a,vec b){
28    return sgn( cross(b-a,p-a) ) == 0 && sgn( (p-a)*(p-b) ) <= 0;
29 }
30 //判断线段ab,pq间是否有交点
31 int has_intersection(vec a,vec b,vec p,vec q){
32    int d1 = sgn( cross(b-a,p-a) ),d2 = sgn( cross(b-a,q-a) );
33    int d3 = sgn( cross(q-p,a-p) ),d4 = sgn( cross(q-p,b-p) );
34    if( d1 * d2 < 0 && d3 * d4 < 0) return 1;//有交点，且交点不在端点
35    if( ( d1 == 0 && point_on_segment(p,a,b) )||
36       ( d2 == 0 && point_on_segment(q,a,b) ) ||
37       ( d3 == 0 && point_on_segment(a,p,q) ) ||
38       ( d4 == 0 && point_on_segment(b,p,q) ))
39        return -1; //重合或交点在端点上
40    return 0;
41 }
42 //直线求交点,需保证p!=a,a!=b
43 int line_intersection(vec a,vec b,vec p,vec q,vec &o,double *t=0){
44    double U=cross(p-a,q-p);
45    double D=cross(b-a,q-p);
46    if(sgn(D) == 0) return sgn(U)==0 ? 2 : 0; //重叠|平行
47    o = a + (b-a) * (U/D);
48    if(t) *t = U/D;
49    return 1;
50 }
51
52 int main() {
53    int tc,n,kase=1;
54    double l,w;
55    scanf("%d",&tc);
56    while(tc--){
57       scanf("%d%lf%lf",&n,&l,&w);
58       int ans=1;
59       forn(i, n){
60          scanf("%lf%lf%lf%lf",&p11[i].x,&p11[i].y,&p22[i].x,&p22[i].y);
61          int res=1;
62          set< pair<double,double> > st;
```

```
63          for(int j=0; j<i; ++j){
64              int outcome=has_intersection(p11[i],p22[i],p11[j],p22[j]);
65              if(!outcome) continue; //没交点 贡献为1
66              if(outcome==-1) continue; //交点是端点 贡献为1
67              vec tmp;
68              line_intersection(p11[i],p22[i],p11[j],p22[j],tmp);
69              st.insert(mp(tmp.x,tmp.y));
70          }
71          ans+=SZ(st)+1;
72      }
73      printf("Case %d: %d\n",kase++,ans);
74  }
75  return 0;
76 }
```

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 树剖

3.3.1 重链剖分

```
1  const int N=(int)1e5+100;
2  int n,m,r,p;
3  int size[N],f[N],son[N],d[N],st[N],en[N],top[N],dfn,pos[N];
4  vector<int> G[N];
5  void dfs(int x){
6      size[x]=1;
7      for(int to:G[x]){
8          if(to!=f[x]){
9              f[to]=x;
10             d[to]=d[x]+1;
11             dfs(to);
12             size[x]+=size[to];
13             if(size[to]>size[son[x]]) son[x]=to;
14         }
15     }
16 }
17 void dfs2(int x,int y){
18     st[x]=++dfn; top[x]=y;
19     pos[dfn]=x; //新编号dfn位置的点对应原位置x的点，build线段树的时候要用(if(l==r) tv[
           node]=a[pos[l]]%p;)
20     if(son[x]) dfs2(son[x],y);
21     for(int to:G[x]){
22         if(to!=son[x] && to!=f[x]) dfs2(to,to);
23         en[x]=dfn;
24     }
25 }
26 //查询x,y两点的lca
27 int lca(int x,int y){
28     for(;top[x]!=top[y];x=f[top[x]]){
29         if(d[top[x]]<d[top[y]]) swap(x,y);
30     }
31     return d[x]<d[y]?x:y;
32 }
33 //x是y的祖先,查询x到y方向的第一个点
34 int lca2(int x,int y){
35     int t;
```

```
36      while(top[x]!=top[y]){
37          t=top[y];
38          y=f[top[y]];
39      }
40      return x==y?t:son[x];
41  }
42  //对x到y路径上的点进行操作
43  void chain(int x,int y){
44      for(;top[x]!=top[y];x=f[top[x]]){
45          if(d[top[x]]<d[top[y]]) swap(x,y);
46          change(st[top[x]],st[x]);
47      }
48      if(d[x]<d[y]) swap(x,y);
49      change(st[y],st[x]);
50   }
```

3.3.2 题单

```
1  洛谷 P3384
2  题意:
3  已知一棵包含n个节点的树(连通且无环)，每个节点上包含一个数值，需要支持以下操作。
4  第一行包含4个正整数n,m,r,p,分别表示树的节点个数，操作个数，节点序号和取模数(即所有的输出结
       果均对此取模)。
5  接下来包含n个非负整数，分别依次表示各个节点上初始的数值。
6  接下来n-1行每行包含两个整数x,y表示点x和y之间连有一条边(保证无环且连通)。
7  接下来m行每行包含若干个正整数，每行表示一个操作，格式如下:
8  操作1: 1 x y z 表示将树从x到y节点最短路径上所有节点的值都加上z。
9  操作2: 2 x y 表示求从树x到y节点最大路径上所有节点的值之和。
10 操作3: 3 x z 表示将以x为根节点的子树内所有节点值都加上z。
11 操作4: 4 x 表示求以x为根节点的子树内所有节点值之和。
12 1<=n<=1e5,1<=m<=1e5,1<=p<=2^31-1。
```

3.3.3 P3384

```
1  洛谷 P3384
2  题意:
3  已知一棵包含n个节点的树(连通且无环)，每个节点上包含一个数值，需要支持以下操作。
4  第一行包含4个正整数n,m,r,p,分别表示树的节点个数，操作个数，节点序号和取模数(即所有的输出结
       果均对此取模)。
5  接下来包含n个非负整数，分别依次表示各个节点上初始的数值。
6  接下来(n-1)行每行包含两个整数x,y表示点x和y之间连有一条边(保证无环且连通)。
7  接下来m行每行包含若干个正整数，每行表示一个操作，格式如下:
8  操作1: 1 x y z 表示将树从x到y节点最短路径上所有节点的值都加上z。
9  操作2: 2 x y 表示求从树x到y节点最大路径上所有节点的值之和。
10 操作3: 3 x z 表示将以x为根节点的子树内所有节点值都加上z。
11 操作4: 4 x 表示求以x为根节点的子树内所有节点值之和。
12 1<=n<=1e5,1<=m<=1e5,1<=p<=2^31-1。
13
14 const int N=(int)1e5+100;
15 int n,m,r,p;
16 int size[N],f[N],son[N],d[N],st[N],en[N],top[N],dfn,pos[N];
```

```
17 vector<int> G[N];
18 int tv[N*4],laz[N*4],a[N];
19 void dfs(int x){
20    size[x]=1;
21    for(int to:G[x]){
22       if(to!=f[x]){
23          f[to]=x;
24          d[to]=d[x]+1;
25          dfs(to);
26          size[x]+=size[to];
27          if(size[to]>size[son[x]]) son[x]=to;
28       }
29    }
30 }
31 void dfs2(int x,int y){
32    st[x]=++dfn; top[x]=y;
33    pos[dfn]=x;
34    if(son[x]) dfs2(son[x],y);
35    for(int to:G[x]){
36       if(to!=son[x] && to!=f[x]) dfs2(to,to);
37       en[x]=dfn;
38    }
39 }
40 //查询x,y两点的lca
41 int lca(int x,int y){
42    for(;top[x]!=top[y];x=f[top[x]]){
43       if(d[top[x]]<d[top[y]]) swap(x,y);
44    }
45    return d[x]<d[y]?x:y;
46 }
47 //x是y的祖先,查询x到y方向的第一个点
48 int lca2(int x,int y){
49    int t;
50    while(top[x]!=top[y]){
51       t=top[y];
52       y=f[top[y]];
53    }
54    return x==y?t:son[x];
55 }
56 void pushdown(int node,int l,int r){
57    if(laz[node]!=0){
58       int mid=(l+r)>>1;
59       tv[node<<1]=(tv[node<<1]+laz[node]*(mid-l+1)%p)%p; //fuck
60       tv[node<<1|1]=(tv[node<<1|1]+laz[node]*(r-mid)%p)%p; //me
61       laz[node<<1]=(laz[node<<1]+laz[node])%p;
62       laz[node<<1|1]=(laz[node<<1|1]+laz[node])%p;
63       laz[node]=0;
64    }
65 }
66 void build(int node,int l,int r){
67    laz[node]=0;
68    if(l==r){
69       tv[node]=a[pos[l]]%p;
```

```
70        return;
71    }
72    int mid=(l+r)>>1;
73    build(node<<1,l,mid);
74    build(node<<1|1,mid+1,r);
75    tv[node]=(tv[node<<1]+tv[node<<1|1])%p;
76 }
77 void modify(int node,int l,int r,int ml,int mr,int mval){
78    if(ml<=l && r<=mr){
79        tv[node]=(tv[node]+mval%p*(r-l+1)%p)%p;
80        laz[node]=(laz[node]+mval%p)%p;
81        return;
82    }
83    pushdown(node,l,r);
84    int mid=(l+r)>>1;
85    if(ml<=mid) modify(node<<1,l,mid,ml,mr,mval);
86    if(mr>mid) modify(node<<1|1,mid+1,r,ml,mr,mval);
87    tv[node]=(tv[node<<1]+tv[node<<1|1])%p;
88 }
89 int query(int node,int l,int r,int ql,int qr){
90    if(ql<=l && r<=qr){
91        return tv[node]%p;
92    }
93    pushdown(node,l,r);
94    int mid=(l+r)>>1;
95    int sum=0;
96    if(ql<=mid) sum=(sum+query(node<<1,l,mid,ql,qr))%p;
97    if(qr>mid) sum=(sum+query(node<<1|1,mid+1,r,ql,qr))%p;
98    return sum%p;
99 }
100 int qu(int x,int y){
101    int sum=0;
102    for(;top[x]!=top[y];x=f[top[x]]){
103        if(d[top[x]]<d[top[y]]) swap(x,y);
104        sum=(sum+query(1,1,n,st[top[x]],st[x]))%p;
105    }
106    if(d[x]<d[y]) swap(x,y);
107    sum=(sum+query(1,1,n,st[y],st[x]))%p;
108    return sum;
109 }
110 void modi(int x,int y,int z){
111    for(;top[x]!=top[y];x=f[top[x]]){
112        if(d[top[x]]<d[top[y]]) swap(x,y);
113        modify(1,1,n,st[top[x]],st[x],z);
114    }
115    if(d[x]<d[y]) swap(x,y);
116    modify(1,1,n,st[y],st[x],z);
117 }
118 signed main() {
119    cin>>n>>m>>r>>p;
120    for1(i, n) cin>>a[i];
121    forn(i, n-1){
122        int a,b;
```

```
123         cin>>a>>b;
124         G[a].eb(b);
125         G[b].eb(a);
126     }
127     dfs(r);
128     dfs2(r,r);
129     build(1,1,n);
130     forn(i, m){
131         int op,x,y,z;
132         cin>>op;
133         if(op==1){
134             cin>>x>>y>>z;
135             modi(x,y,z);
136         } else if(op==2){
137             cin>>x>>y;
138             cout<<qu(x,y)<<'\n';
139         } else if(op==3){
140             cin>>x>>z;
141             modify(1,1,n,st[x],en[x],z);
142         } else{
143             cin>>x;
144             cout<<query(1,1,n,st[x],en[x])<<'\n';
145         }
146     }
147     return 0;
148 }
```

3.4 主席树

3.4.1 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], t[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  }
```

3.4.2 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棵树的时候，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  }
```

3.5 字典树

3.5.1 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  }
```

```
int Find(char s[]) {
    root = 0;
    len = strlen(s);
    for (int i = 0; i < len; ++i) {
        int id = int (s[i] - 'a');
        if (!trie[root][id]) return 0;
        root = trie[root][id];
    }
    return sum[root];
}
void init() { //多组数据慎用，要么把字典树开小一点
    memset(trie, 0, sizeof(trie));
    memset(sum, 0, sizeof(sum));
    memset(ed, 0, sizeof (ed));
    tot = 0;
    flag = false;
}

int main() {
    int kase = 1;
    init();
    while (~scanf("%s", s)) {
        if (strlen(s) == 1 && s[0] == '9') {
            if (flag == false) printf("Set %d is immediately decodable\n", kase++);
            else printf("Set %d is not immediately decodable\n", kase++);
            init();
        } else {
            Insert(s);
        }
    }
    return 0;
}
```

4 字符串

4.1 KMP

```
const int maxn = 1000000 + 1000;
int t, nxt[maxn];
char x[maxn], y[maxn];

//ps:2019ccpc秦皇岛的那题爆longlong了 XD

void kmp_pre(char x[], int m, int nxt[]) {
    int i, j;
    j = nxt[0] = -1;
    i = 0;
    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 -
        nxt[m]) == 0
        while (-1 != j && x[i] != x[j]) j = nxt[j];
        nxt[++i] = ++j;
    }
```

```
}
int kmp_count(char x[], int m, char y[], int n) {
    int i, j;
    int ans = 0;
    kmp_pre(x, m, nxt);
    i = j = 0;
    while (i < n) {
        while (-1 != j && y[i] != x[j]) j = nxt[j];
        ++i;
        ++j;
        if (j >= m) {
            ++ans;
            j = nxt[j];
        }
    }
    return ans;
}

int main() {

    cin >> t;
    while (t--) {
        cin >> x;
        cin >> y;
        cout << kmp_count(x, strlen(x), y, strlen(y)) << '\n';
    }

    return 0;
}
```

4.2 exKMP

```
const int maxn = 50000 + 1000;
int nxt[maxn], extend[maxn];
char x[maxn], y[maxn];
 //x是模式串，y是主串
//(0~m-1)是有效部分，和kmp不同
void pre_EKMP(char x[], int m, int nxt[]) {
    nxt[0] = m;
    int j = 0;
    while (j + 1 < m && x[j] == x[j + 1]) ++j;
    nxt[1] = j;
    int k = 1;
    for (int i = 2; i < m; ++i) {
        int p = nxt[k] + k - 1;
        int L = nxt[i - k];
        if (i + L < p + 1) nxt[i] = L;
        else {
            j = max(0, p - i + 1);
            while (i + j < m && x[i + j] == x[j]) ++j;
            nxt[i] = j;
            k = i;
```

```
        }
    }
}
void EKMP(char x[], int m, char y[], int n, int nxt[], int extend[]) {
    pre_EKMP(x, m, nxt);
    int j = 0;
    while (j < n && j < m && x[j] == y[j]) ++j;
    extend[0] = j;
    int k = 0;
    for (int i = 1; i < n; ++i) {
        int p = extend[k] + k - 1;
        int L = nxt[i - k];
        if (i + L < p + 1) extend[i] = L;
        else {
            j = max(0, p - i + 1);
            while (i + j < n && j < m && y[i + j] == x[j]) ++j;
            extend[i] = j;
            k = i;
        }
    }
}
```

4.3 Manacher

```
#include <bits/stdc++.h>
#define maxn 2000005
using namespace std;
int mp[maxn];
string str;
char c[maxn];
void Manacher(string s,int len){
    int l=0,R=0,C=0;;
    c[l++]='$', c[l++]='#';
    for(int i=0;i<len;i++){
        c[l++]=s[i], c[l++]='#';
    }
    for(int i=0;i<l;i++){
        mp[i]=R>i?min(mp[2*C-i],R-i):1;
        while(i+mp[i]<l&&i-mp[i]>0){
            if(c[i+mp[i]]==c[i-mp[i]]) mp[i]++;
            else break;
        }
        if(i+mp[i]>R){
            R=i+mp[i], C=i;
        }
    }
}
int main()
{
    int cnt=0;
    while(cin>>str){
        if(str=="END") break;
```

```
        int len=str.length();
        Manacher(str,len);
        int ans=0;
        for(int i=0;i<2*len+4;i++){
            ans=max(ans,mp[i]-1);
        }
        printf("Case %d: %d\n",++cnt,ans);
    }
    return 0;
}
```

5 dp

5.1 树形 dp

5.1.1 树的重心

```
//poj 1655
//树的重心:
//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。
const int maxn = 20000+100;
int tc, n, sz[maxn], fa[maxn], res[maxn];
vector<int> G[maxn];

void init(int n) {
    memset(fa, 0, sizeof(fa));
    memset(sz, 0, sizeof(sz));
    for (int i = 1; i <= n; ++i) G[i].clear();
}
void dfs(int x, int par) {
    sz[x] = 1;
    fa[x] = par;
    for (int i = 0; i < int(G[x].size()); ++i) {
        int to = G[x][i];
        if (to == par) continue;
        dfs(to, x);
        sz[x] += sz[to];
    }
}

int main() {

    scanf("%d", &tc);
    while (tc--) {
        scanf("%d", &n);
        init(n);
        for (int i = 0; i < n - 1; ++i) {
            int u, v;
            scanf("%d%d", &u, &v);
            G[u].push_back(v);
            G[v].push_back(u);
        }
```

```
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

```
//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
//(multi)set和priority_queue都有empty()函数
struct node {
    int l, r;
    node(int _l, int _r) : l(_l), r(_r) {}
};
struct cmp {
    bool operator () (node n1, node n2) const {
        int l1 = n1.r - n1.l + 1;
        int l2 = n2.r - n2.l + 1;
        if (l1 == l2) return n1.l < n2.l;
        return l1 > l2;
    }
};
(multi)set<node, cmp> st;
/*********************************************/
struct node {
    int l, r;
    node(int _l, int _r) : l(_l), r(_r) {}
    friend bool operator < (node n1, node n2) { //一个堆，越在顶端(队顶)的越大
        int l1 = n1.r - n1.l + 1; //优先队列只能重载 < 号
        int l2 = n2.r - n2.l + 1;
        if (l1 == l2) return n1.l > n2.l;
        return l1 < l2;
    }
};
priority_queue<node> pp;
```

7.2 nth_element

```
int a[n];
//求第k小的数
nth_element(a, a + k, a + n);
//求第k大的数
nth_element(a, a + k, a + n, greater<>());
```

8 其他问题

8.1 ST 表

8.1.1 ST 表

```
//初始化O(logn) 询问O(1)
//hdu 5443
//询问最小，把两个max改成min就行
const int maxn = 2000;
int dp[maxn][35], LOG[maxn]; //2^30 == 1e9
```

```
int tc, n, m, a[maxn];

void initRMQ() {
   LOG[0] = -1;
   for (int i = 1; i <= n; ++i) {
      LOG[i] = ((i & (i-1)) == 0) ? LOG[i-1] + 1 : LOG[i-1];
      dp[i][0] = a[i];
   }
   for (int j = 1; j <= LOG[n]; ++j) {
      for (int i = 1; i + (1 << j) - 1 <= n; ++i) {
         dp[i][j] = max(dp[i][j-1], dp[i + (1<<(j-1))][j-1]);
      }
   }
}
int rmqQuery(int x, int y) {
   int k = LOG[y-x+1];
   return max(dp[x][k], dp[y-(1<<k)+1][k]);
}
```

8.2 莫队

8.2.1 复杂度

```
带修改莫队的复杂度为O(n^(5/3))
不带修改莫队复杂度为O(m*logm + n*sqrt(m))
另外还要加上暴力求解的复杂度
```

8.2.2 普通莫队 LOJ 1188 O(根号 n 乘 q)

```
//复杂度O(根号n * q)
#include <bits/stdc++.h>

using namespace std;

const int maxn = (int)1e5 + 1000;
int t, n, qu, cnt[maxn], res, a[maxn], blo, ans[maxn];
struct node {
   int l, r, id;
}q[maxn];

int read() {
   int ans = 0, f = 1; char c = getchar();
   for (;c < '0' || c > '9'; c = getchar()) if (c == '-') f = -1;
   for (;c >= '0' && c <= '9'; c = getchar()) ans = ans * 10 + c - '0';
   return ans * f;
}
bool cmp(node a, node b) {
   return (a.l / blo == b.l / blo ? (a.l / blo) & 1 ? a.r < b.r : a.r > b.r : a.l <
        b.l);
}
void add(int pos) {
   int num = a[pos];
```

```
    if (cnt[num] == 0) ++res;
    ++cnt[num];
}
void cut(int pos) {
    int num = a[pos];
    --cnt[num];
    if (cnt[num] == 0) --res;
}

int main() {
    ios::sync_with_stdio(false); cin.tie(0);
    t = read();
    for (int kase = 1; kase <= t; ++kase) {
        memset(cnt, 0, sizeof(cnt));
        n = read();
        qu = read();
        blo = sqrt(n * 1.0 * 2 / 3);
        for (int i = 1; i <= n; ++i) {
            a[i] = read();
        }
        for (int i = 0; i < qu; ++i) {
            q[i].l = read();
            q[i].r = read();
            q[i].id = i;
        }
        sort(q, q + qu, cmp);
        int l = 1, r = 0;
        res = 0;
        for (int i = 0; i < qu; ++i) {
            while (r < q[i].r) add(++r);
            while (l > q[i].l) add(--l);
            while (r > q[i].r) cut(r--);
            while (l < q[i].l) cut(l++);
            ans[q[i].id] = res;
        }
        printf("Case %d:\n", kase);
        for (int i = 0; i < qu; ++i) {
            printf("%d\n", ans[i]);
        }
    }
    return 0;
}
```

8.2.3 带修改莫队 cf 940F

```
#include <bits/stdc++.h>

#define mp make_pair
#define mt make_tuple
#define fi first
#define se second
#define pb push_back
```

```cpp
#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define forn(i, n) for (int i = 0; i < (int)(n); ++i)
#define for1(i, n) for (int i = 1; i <= (int)(n); ++i)
#define ford(i, n) for (int i = (int)(n) - 1; i >= 0; --i)
#define fore(i, a, b) for (int i = (int)(a); i <= (int)(b); ++i)

using namespace std;

typedef pair<int, int> pii;
typedef vector<int> vi;
typedef vector<pii> vpi;
typedef vector<vi> vvi;
typedef long long i64;
typedef vector<i64> vi64;
typedef vector<vi64> vvi64;
typedef pair<i64, i64> pi64;
typedef double ld;

template<class T> bool uin(T &a, T b) { return a > b ? (a = b, true) : false; }
template<class T> bool uax(T &a, T b) { return a < b ? (a = b, true) : false; }

const int maxn = (int)2 * 1e6 + 1000;
int a[maxn], b[maxn], c[maxn], n, m, qdx = 0, mdx = 0, bsz, res = 0;
int ans[maxn], times[maxn], cnt[maxn], rl[maxn], rr[maxn], d[maxn];
struct query {
    int l, r, md, id;
    void set(int _l, int _r, int _md, int _id) {
        l = _l, r = _r, md = _md, id = _id;
    }
    bool operator < (const query &b) const {
        if (l / bsz != b.l / bsz) return l < b.l;
        if (r / bsz != b.r / bsz) return r < b.r;
        return id < b.id;
    }
}que[maxn];
struct modify {
    int wz, x, y;
    void set(int _wz, int _x, int _y) {
        wz = _wz, x = _x, y = _y;
    }
}mod[maxn];

int read() {
    int ans = 0, f = 1; char c = getchar();
    for(;c < '0' || c > '9'; c = getchar()) if (c == '-') f = -1;
    for(;c >= '0' && c <= '9'; c = getchar()) ans = ans * 10 + c - '0';
    return ans * f;
}
template <class T>
void write(T x) {
    if (x < 0) x = -x, putchar('-');
    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(l, r, mdx, qdx);
130         if (op == 2) {
131             ++mdx;
132             rl[mdx] = l;
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 母函数

8.3.1 hdu 1028

```
题意:
给你一个数n，现在有无限个1~n的数，现在问你要组成n有多少种不同的组成方案，顺序无所谓。
比如 4 有5种组成方法:
4=1+1+1+1;
4=2+1+1;
4=2+2;
4=3+1;
4=4;
思路:
母函数，详情见注释

while(cin >> n) {
    for(int i = 0; i <= n; i++) { //初始化第一个括号
        c1[i] = 1;
        c2[i] = 0;
    }
    for(int i = 2; i <= n; i++) { //操作第i个括号，从第2个开始
        for(int j = 0; j <= n; j++) {//对于当前式子中指数为j的项进行操作
            for(int k = 0; k + j <= n; k += i) {//第i个口号中指数k的项与当前式子指数j项相乘
                c2[j + k] += c1[j]; //乘积是一个指数j + k的项，将其系数累加到该项
            }
        }//一轮计算结果保存在c2[]数组中
        for(int j = 0; j <= n; j++) { //把c2[]的数据赋值给c1[]，并将c2[]清零
            c1[j] = c2[j];
            c2[j] = 0;
        }
    }
    cout << c1[n] << endl;
}
```

8.4 二分注意点

```
1. while(>=) +1 -1
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)
```

8.5 LIS

```
int LIS(int a[]) { //lis数组从0开始
    int len = 0;
    for1(i, n) {
        int x = lower_bound(lis, lis+len, a[i])-lis;
        lis[x] = a[i];
        len = max(len, x+1);
    }
    return len;
}
```

```
int LIS(int a[]) { //lis数组从1开始
   int len = 0;
   for1(i, n) {
      int x = lower_bound(lis+1, lis+1+len, a[i])-lis;
      lis[x] = a[i];
      len = max(len, x);
   }
   return len;
}
lower_bound : a1 < a2 < ... < an
upper_bound : a1 <= a2 <= ..<= an
```

8.6 尺取法

```
//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。
#include<bits/stdc++.h>
//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。
#define ll long long
using namespace std;
const ll maxn=(ll)1e5+100;
ll tc,n,s,a[maxn];
int main() {
   cin>>tc;
   while(tc--){
      cin>>n>>s;
      for(int i=1; i<=n; ++i) cin>>a[i];
      ll ans=LLONG_MAX;
      ll l=1,r=1,sum=0;
      for(;;){
         while(r<=n && sum<s){
            sum+=a[r++];
         }
         if(sum<s) break;
         ans=min(ans, r-l);
         sum-=a[l++];
      }
      if(ans==LLONG_MAX) ans=0;
      cout<<ans<<'\n';
   }
   return 0;
}
```

8.7 单调队列

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

```
    ios::sync_with_stdio(false);cin.tie(0);cout.precision(10);cout << fixed;
#ifdef LOCAL_DEFINE
    freopen("input.txt", "r", stdin);
#endif
    cin>>n>>m>>k;
    memset(a, 0, sizeof(a));
    memset(b, 0, sizeof(b));
    for(int i=1; i<=n; ++i){
        for(int j=1; j<=m; ++j){
            if(!a[i][j]){
                for(int k=1; k*i<=n && k*j<=m; ++k){
                    a[i*k][j*k]=k; b[i*k][j*k]=i*j*k;
                }
            }
        }
    }
    //维护队首最大的单调队列
    memset(a, 0, sizeof(a));
    for(int i=1; i<=n; ++i){
        while(!dq.empty()) dq.pop_front();
        dq.push_back(0);
        for(int j=1; j<=m; ++j){
            while(!dq.empty() && j-dq.front()>=k) dq.pop_front();
            while(!dq.empty() && b[i][j]>=b[i][dq.back()]) dq.pop_back(); //因为队首最
                大，所以>=
            dq.push_back(j);
            a[i][j]=b[i][dq.front()];
        }
    }
    long long ans=0;
    for(int j=1; j<=m; ++j){
        while(!dq.empty()) dq.pop_front();
        dq.push_back(0);
        for(int i=1; i<=n; ++i){
            while(!dq.empty() && i-dq.front()>=k) dq.pop_front();
            while(!dq.empty() && a[i][j]>=a[dq.back()][j]) dq.pop_back();
            dq.push_back(i);
            if(i>=k && j>=k) ans+=a[dq.front()][j];
        }
    }
    cout<<ans<<'\n';
#ifdef LOCAL_DEFINE
    cerr << "Time elapsed: " << 1.0 * clock() / CLOCKS_PER_SEC << " s.\n";
#endif
    return 0;
}
```

8.8 一句话去重并生成新数组

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

8.9 输入一行看有多少个数

```cpp
//结果存在op数组中(0 ~ n-1)
vector<int> op(maxn);

int input(){
    string str, ss;
    getline(cin, str);
    if (str[0] == '-')
        return -1;
    istringstream s(str);
    vector<int> v;
    v.clear();
    while(s >> ss){
        int tmp = 0;
        for (int i = 0;i < int(ss.size()); i++)
            tmp = tmp * 10 + (ss[i] - '0');
        v.push_back(tmp);
    }
    for (int i = 0;i < int(v.size()); i++)
        op[i] = v[i];
    return v.size();
}
```

8.10 最小（大）表示法

```cpp
//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
// 传入的值s是个两个s， len是一个s
int min_string(char *s, int len) {
    int i = 0, j = 1, k = 0;
    while (i < len && j < len && k < len) {
        if (s[i + k] == s[j + k]) ++k;
        else if (s[i + k] < s[j + k]) j += k + 1, k = 0;
        else if (s[i + k] > s[j + k]) i += k + 1, k = 0;
        if (i == j) ++j;
    }
    return min(i, j);
}
int max_string(char *s, int len) {
    int i = 0, j = 1, k = 0;
    while (i < len && j < len && k < len) {
        if (s[i + k] == s[j + k]) ++k;
        else if (s[i + k] < s[j + k]) i += k + 1, k = 0;
        else if (s[i + k] > s[j + k]) j += k + 1, k = 0;
        if (i == j) ++j;
    }
    return min(i, j);
}
```

8.11 随机数

```
【随机函数】
使用mt19937而不是rand()
#include <chrono>
#include <random>
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
ll ans = rng();
printf("%I64d\n",ans);
范围为0 - 4294967295 (2的32次方 - 1)

【随机生成整数】
int randInt(int l,int r){    //生成l到r的整数,l <= r
    return (rng() % (r - l + 1)) + l;
}
```

8.12 输入日期输出周几

```
输入年月日，例如2010-08-15,就调用calc(2010,8,15)
输出0代表周日,1代表周一....,6代表周日

int calc(int y,int m,int d)
{
   if(m==1 || m==2)
      {
      m+=12;
      --y;
   }
   int w=(d+1+2*m+3*(m+1)/5+y+y/4-y/100+y/400)%7;
   return w;
}
```

9 黑科技

9.1 IO

9.1.1 快读模板

```
//读不了浮点数
//前面四个快读效率都差不多 快写比printf快一点
int read() {
   int x=0, f=1; char ch=getchar();
   while(ch<'0' || ch>'9') {if(ch=='-') f = -1;ch = getchar();}
   while(ch>='0' && ch<='9') x=(x<<3)+(x<<1)+(ch^48),ch = getchar();
   return x*f;
}

int read() {
   int ans = 0, f = 1; char c = getchar();
   for (;c < '0' || c > '9'; c = getchar()) if (c == '-') f = -1;
   for (;c >= '0' && c <= '9'; c = getchar()) ans = ans * 10 + c - '0';
   return ans * f;
```

```
}

i64 read() {
    i64 ans = 0, f = 1; char c = getchar();
    for (;c < '0' || c > '9'; c = getchar()) if (c == '-') f = -1;
    for (;c >= '0' && c <= '9'; c = getchar()) ans = ans * 10 + c - '0';
    return ans * f;
}

template<class T>inline void read(T &res)
{
    char c;T flag=1;
    while((c=getchar())<'0'||c>'9')if(c=='-')flag=-1;res=c-'0';
    while((c=getchar())>='0'&&c<='9')res=res*10+c-'0';res*=flag;
}

template <class T>
void write(T x){
    if(x < 0) x = -x, putchar('-');
    if(x >= 10) write(x / 10);
    putchar('0' + x % 10);
}
```

9.1.2 ___int128 输入输出模板

```
//必须搭配scanf printf使用
__int128 read(){
    __int128 x=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9'){
        if(ch=='-')
            f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9'){
        x=x*10+ch-'0';
        ch=getchar();
    }
    return x*f;
}
void print(__int128 x){
    if(x<0){
        putchar('-');
        x=-x;
    }
    if(x>9)
        print(x/10);
    putchar(x%10+'0');
}
```

9.2 istringstream

```cpp
#include <bits/stdc++.h>
using namespace std;
string str, s;
int main() {
    while (true) {
        getline(cin, str);
        if (int(str.size()) == 1 && str[0] == '#') break;
        istringstream all(str);
        while (all >> s) {
            cout << "s : " << s << endl;
        }
    }
    return 0;
}
```

9.3 unordered_map 防 hack 模板

```cpp
头文件 #include <bits/stdc++.h>
struct custom_hash {
    static uint64_t splitmix64(uint64_t x) {
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }

    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().
            time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};

unordered_map<XXX, XXX, custom_hash> you_name_it;

两组数据:
unordered_map中的insert (优化过) :
x = 107897: 0.035 seconds
x = 126271: 0.031 seconds

(未优化过) :
x = 107897: 0.014 seconds
x = 126271: 2.787 seconds
```

9.4 杜教 BM

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

```
#define pb push_back
typedef long long i64;
#define SZ(x) ((i64)(x).size())
typedef vector<i64> vi64;
typedef pair<i64,i64> PII;
const i64 mod=(i64)1e9 + 7;
i64 powmod(i64 a,i64 b) {
    i64 res=1;
    a%=mod;
    assert(b>=0);
    for(; b; b>>=1) {
        if(b&1)res=res*a%mod;
        a=a*a%mod;
    }
    return res;
}
i64 _,n;
namespace linear_seq {
    const i64 N=10010;
    i64 res[N],base[N],_c[N],_md[N];

    vector<i64> Md;
    void mul(i64 *a,i64 *b,i64 k) {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for (i64 i=k+k-1; i>=k; i--) if (_c[i])
                rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
        rep(i,0,k) a[i]=_c[i];
    }
    i64 solve(i64 n,vi64 a,vi64 b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]+...
        i64 ans=0,pnt=0;
        i64 k=SZ(a);
        assert(SZ(a)==SZ(b));
        rep(i,0,k) _md[k-1-i]=-a[i];
        _md[k]=1;
        Md.clear();
        rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
        rep(i,0,k) res[i]=base[i]=0;
        res[0]=1;
        while ((1ll<<pnt)<=n) pnt++;
        for (i64 p=pnt; p>=0; p--) {
            mul(res,res,k);
            if ((n>>p)&1) {
                for (i64 i=k-1; i>=0; i--) res[i+1]=res[i];
                res[0]=0;
                rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
            }
        }
        rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
    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

```
JSOI2004 平衡点
可以<del>优化</del>玄学的几点:
1. 初始温度T的值
2. 降温系数的范围0.985~0.999
3. 多做几次退火减小误差
4. T>... (终止温度)
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
const int N=1100;
int n,i;
double anx,any,vx,vy,dis,nx,ny,T,nv,x[N],y[N],w[N];
double Rand() {return (double)(rand()%20000)/20000.0;}
double dist(double xx,double yy){
    nv=0;
    for(int i=1; i<=n; ++i) nv+=sqrt((xx-x[i])*(xx-x[i])+(yy-y[i])*(yy-y[i]))*w[i];
    if(nv<dis) dis=nv,anx=xx,any=yy;
    return nv;
}
void SA(){
    T=8000; //初始温度
    for(;T>0.001;){//小于给定系数就退出
        nx=vx; ny=vy;
        nx=nx+T*(Rand()*2-1);//在当前位置的变化幅度内随机取一点
        ny=ny+T*(Rand()*2-1);
        nv=dist(vx,vy)-dist(nx,ny);//计算当前解
        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
            vx=nx; //否则以exp|当前解-最优解| / T的概率接受当前解
            vy=ny;
        }
        T*=0.996; //降低搜索范围 (降温)
    }
}
signed main() {
    scanf("%d",&n);
    for(i=1; i<=n; ++i){
        scanf("%lf%lf%lf",&x[i],&y[i],&w[i]);
        anx+=x[i];
        any+=y[i];
    }
    anx/=(double)n; any/=(double)n;
    vx=anx=vy=any;//(vx,vy)当前位置, (anx,any)最优解位置
    dis=dist(anx,any);
    /*for(int i=0; i<10; ++i)*/ SA();
    printf("%.3f %.3f\n",anx,any);
    return 0;
}
```

10 大数

10.1 java

10.1.1 输入

```
输入
1.1 申明一个输入对象cin
Scanner cin=new Scanner(System.in);

1.2 输入一个int值
Int a=cin.nextInt();

1.3 输入一个大数
BigDecimal a=cin.nextBigdecimal();

1.4 EOF结束
while(cin.hasNext()) ...{}

输出
2.1 输出任意类型的str
System.out.println(str); //有换行
System.out.print(str) //无换行
System.out.println"( "str);//输出字符串str
System.out.println("Hello,%s.Next year,you'll be %d",name,age);

大数类
3.1 赋值
BigInteger a=BigInteger.valueOf(12);
BigInteger b=new BigInteger(String.valueOf(12));
BigDecimal c=BigDecimal.valueOf(12.0);
BigDecimal d=new BigDecimal("12.0");//建议使用字符串以防止double类型导致的误差

也可以用上述方法构造一个临时对象用于参与运算
b.add(BigInteger.valueOf(105));
```

10.1.2 申明变量

```
申明数组
int[] cnt=new int[10];
```

10.1.3 String 操作

```
https://ac.nowcoder.com/acm/contest/9004/C
牛客C
题意:
给你一个长度1e5并且只包含(1~9)的字符串，把这n个数分成k块，顺序可以随意搞，让这k块数的和最大。
思路:
先把String转成array然后sort一下（默认从小到大），用StringBuilder（据说快一点）从后往前贪心取即可。

import java.util.*;
import java.math.BigInteger;
public class Solution {
    public String Maxsumforknumers (String x, int k) {
        char[] a=x.toCharArray();
```

```
13          Arrays.sort(a);
14          StringBuilder s=new StringBuilder();
15          for(int i=a.length-1; i>k-2; --i){
16              s.append(String.valueOf(a[i]));
17          }
18          BigInteger res=new BigInteger(s.toString());
19          for(int i=k-2; i>=0; --i){
20              res=res.add(new BigInteger(String.valueOf(a[i])));
21          }
22          return res.toString();
23      }
24  }
```

10.1.4 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)
```