ICPC Templates

-zzy-

October 16, 2020

Contents

1 图论

1.1 最短路

1.1.1 Dijkstra

```cpp
//不能求最长路 最长路用spfa搞
//时间复杂度 O(e*log(e))
#define i64 long long

const int maxn = ; //点数
const i64 INF = 0x3f3f3f3f3f3f3f3fLL;
struct node {
    int id;
    i64 w;
    node(){}
    node(int a, int b) : id(a), w(b) {}
    friend bool operator < (node a, node b) {return a.w > b.w;}
};
vector<node> G[maxn];
bool vis[maxn];
i64 dis[maxn];

void dij(int s, int n) {
    priority_queue<node> q;
    while (!q.empty()) q.pop();
    node cur;
    for (int i = 1; i <= n; ++i) {
        dis[i] = INF;
        vis[i] = 0;
    }
    dis[s] = 0;
    q.push(node(s, dis[s]));
    while (!q.empty()) {
        cur = q.top();
        q.pop();
        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();
}

int main() {

```

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

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   #define i64 long long
5   const int N = 点数;
6   const i64 INF = 0x3f3f3f3f3f3f3f3fLL;
7   int n,m;
8   struct node {
9       int id;
10      i64 w;
11      node() {}
12      node(int a, i64 b) : id(a), w(b) {}
13  };
14  vector<node> G[N];
15  i64 dis[N],cnt[N];
16  bool vis[N];
17
18  bool spfa(int s, int n) { //s是起点,n是上界点数,点的编号从0开始
19      queue<node> q;
20      node cur;
21      ms(vis,0);
22      ms(cnt,0);
23      for (int i = 0; i <= n; ++i) dis[i] = INF;
24      vis[s] = 1;
25      dis[s] = 0;
26      q.push(node(s, dis[s]));
27      while (!q.empty()) {
28          cur = q.front();
29          q.pop();
30          //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
31          ++cnt[cur.id];
32          if(cnt[cur.id]>n) return false;
33          vis[cur.id] = 0;
34          for (node to : G[cur.id]) {
35              if (dis[to.id] > dis[cur.id]+to.w) {
```

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

1.1.3 floyd 求传递闭包

```
1  // hdu 1704
2  const int maxn = 600;
3  int tc, n, m, G[maxn][maxn];
4
5  void floyd() {
6      for (int k = 1; k <= n; ++k) {
7          for (int i = 1; i <= n; ++i) {
8              if (G[i][k]) {
9                  for (int j = 1; j <= n; ++j) {
10                     if (G[k][j]) {
11                         G[i][j] = 1;
12                     }
13                 }
14             }
15         }
16     }
17 }
18
```

```
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) {
```

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

1.1.5 johnson 全源最短路

```
1  //洛谷 johnson全源最短路模板题
2  Johnson 算法则通过另外一种方法来给每条边重新标注边权。
3  我们新建一个虚拟节点（在这里我们就设它的编号为0）。从这个点向其他所有点连一条边权为 0 的边。
4  接下来用 Bellman-Ford 算法求出从 0 号点到其他所有点的最短路，记为dis[i]
5  假如存在一条从 u 点到 v 点，边权为 w 的边，则我们将该边的边权重新设置为 w+dis[u]-dis[v]
6  接下来以每个点为起点，跑 n 轮 Dijkstra 算法即可求出任意两点间的最短路了(要把偏移量dis[v]-
       dis[u]加上)。
7
8  const int N = 3100;
9  const i64 INF = 0x3f3f3f3f3f3f3f3fLL;
10 int n,m,u[6100],v[6100],w[6100];
```

```cpp
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];
i64 dis[N],cnt[N],d2[N];
bool vis[N];
bool spfa(int s, int n) { //s是起点,n是点数,点的编号从0开始
    queue<node> q;
    node cur;
    ms(vis,0);
    ms(cnt,0);
    for (int i = 0; i <= n; ++i) dis[i] = INF;
    vis[s] = 1;
    dis[s] = 0;
    q.push(node(s, dis[s]));
    while (!q.empty()) {
        cur = q.front();
        q.pop();
        vis[cur.id] = 0;
        //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) return false;
        for (node to : G[cur.id]) {
            if (dis[to.id] > dis[cur.id]+to.w) {
                dis[to.id] = dis[cur.id]+to.w;
                if (!v is[to.id]) {
                    q.push(node(to.id, dis[to.id]));
                    vis[to.id] = 1;
                }
            }
        }
    }
    return true;
}
void dij(int s, int n) {
    priority_queue<node> q;
    while (!q.empty()) q.pop();
    node cur;
    for (int i = 0; i <= n; ++i) {
        d2[i] = INF;
        vis[i] = 0;
    }
    d2[s] = 0;
    q.push(node(s, d2[s]));
    while (!q.empty()) {
        cur = q.top();
        q.pop();
        if (vis[cur.id]) continue;
        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  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。
2  a-b<=c b->a (c)
3  a-b<c b->a (c)
4  a==b b->a (0) && a->b (0)
5  a<=c S->a (c)
6  a>=c a->S (-c)
7  a/b>=c a->b (-log(c))
```

1.2 网络流

1.2.1 DICNIC

```cpp
/*
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)
*/
const int maxn = (int)1e4+1000;
const int INF = INT_MAX;
struct Edge{
    int from, to, cap, flow;
    Edge(int u, int v, int c, int f)
        : from(u), to(v), cap(c), flow(f) {}
};
struct Dicnic {
    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[maxn]; //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[maxn]; //BFS使用
    int d[maxn]; //从起点到i的距离
    int cur[maxn]; //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, int cap) {
        edges.emplace_back(from, to, cap, 0);
        edges.emplace_back(to, from, 0, 0);
        m = int(edges.size());
        G[from].push_back(m-2);
        G[to].push_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];
}
int DFS(int x, int a) {
    if (x == t || a == 0) return a;
    int 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;
}
int Maxflow(int s, int t) {
    this->s = s; this->t = t;
    int flow = 0;
    while (BFS()) {
        memset(cur, 0, sizeof(cur));
        flow += DFS(s, INF);
    }
    return flow;
}
}dicnic;

int main() {

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

    return 0;
}
```

1.2.2 ISAP

```
//时间复杂度 O(v^2*E)
const int maxn = "EDIT"; //点数
const int INF = 0x3f3f3f3f;
struct Edge{
    int from, to, cap, flow;
```

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

```
59          for (int i = 0; i < len; ++i) {
60              Edge &e = edges[G[x][i] ^ 1];
61              if (!vis[e.from] && e.cap > e.flow) {
62                  vis[e.from] = 1;
63                  d[e.from] = d[x] + 1;
64                  q.push(e.from);
65              }
66          }
67      }
68  }
69  int Maxflow(int s, int t) {
70      this->s = s;
71      this->t = t;
72      int flow = 0;
73      BFS();
74      if (d[s] >= n) return 0;
75      for (int i = 1; i <= n; ++i)
76          if (d[i] < INF) num[d[i]]++;
77      int x = s;
78      while (d[s] < n) {
79          if (x == t) {
80              flow += Augument();
81              x = s;
82          }
83          int ok = 0;
84          for (int i = cur[x]; i < int(G[x].size()); ++i) {
85              Edge &e = edges[G[x][i]];
86              if (e.cap > e.flow && d[x] == d[e.to] + 1) {
87                  ok = 1;
88                  p[e.to] = G[x][i];
89                  cur[x] = i;
90                  x = e.to;
91                  break;
92              }
93          }
94          if (!ok) { //Retreat
95              int m = n - 1;
96              for (int i = 0; i < int(G[x].size()); ++i) {
97                  Edge &e = edges[G[x][i]];
98                  if (e.cap > e.flow) m = min(m, d[e.to]);
99              }
100             if (--num[d[x]] == 0) break; //gap优化
101             num[d[x] = m + 1]++;
102             cur[x] = 0;
103             if (x != s) x = edges[p[x]].from;
104         }
105     }
106     return flow;
107 }
108 }isap;
109
110 int main() {
111
```

```
    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;
}
```

1.2.3 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;
        queue<int> q;
        q.push(s);
```

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

1.2.4 常见思路

```
1 如果是无向图，那么建边的时候，反向边的流量不是0，而是和正向边的流量相同。
2
```

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]。

1.3 匹配问题

1.3.1 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;
}
```

1.3.2 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;
    memset(dx, -1, sizeof(dx));
    memset(dy, -1, sizeof(dy));
```

```
25      for (int i = 1; i <= un; ++i) { //编号1~(un) ①
26          if(mx[i] == -1) {
27              q.push(i);
28              dx[i] = 0;
29          }
30      }
31      while (!q.empty()) {
32          int u = q.front();
33          q.pop();
34          if (dx[u] > dis) break;
35          int sz = int(G[u].size());
36          for (int i = 0; i < sz; ++i) {
37              int v = G[u][i];
38              if (dy[v] == -1) {
39                  dy[v] = dx[u]+1;
40                  if (my[v] == -1) dis = dy[v];
41                  else {
42                      dx[my[v]] = dy[v]+1;
43                      q.push(my[v]);
44                  }
45              }
46          }
47      }
48      return (dis!=INF);
49  }
50  bool dfs(int u) {
51      int sz = int(G[u].size());
52      for (int i = 0; i < sz; ++i) {
53          int v = G[u][i];
54          if (!vis[v] && dy[v] == dx[u]+1) {
55              vis[v] = true;
56              if (my[v] != -1 && dy[v] == dis) continue;
57              if (my[v] == -1 || dfs(my[v])) {
58                  my[v] = u;
59                  mx[u] = v;
60                  return true;
61              }
62          }
63      }
64      return false;
65  }
66  int MaxMatch() {
67      int res = 0;
68      memset(mx, -1, sizeof(mx));
69      memset(my, -1, sizeof(my));
70      while (SearchP()) {
71          memset(vis, false, sizeof(vis));
72          for (int i = 1; i <= un; ++i) { //②
73              if (mx[i] == -1 && dfs(i)) res++;
74          }
75      }
76      return res;
77  }
```

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

1.3.3 KM-DFS

```
1   //hdu 2255 点和点之间都有边
2   //hdu 2426 点和点之间有可能没有边
3   //n <= m
4
5   const int maxn = 510;
6   const int INF = INT_MAX;
7   int n, m, link[maxn][maxn], num_a[maxn], num_b[maxn];
8   int match[maxn], slack[maxn], vis_a[maxn], vis_b[maxn];
9
10  bool dfs(int x) {
11      vis_a[x] = 1;
12      for (int i = 0; i < m; ++i) {
13          if (link[x][i] != -1) { //if写外面比里面写continue快？(hdu2426)
14          //两点之间不一定有边的情况要加这句
15              if (vis_b[i]) continue;
16              int diff = num_a[x]+num_b[i]-link[x][i];
17              if (!diff) {
18                  vis_b[i] = 1;
19                  if (match[i] == -1 || dfs(match[i])) {
20                      match[i] = x;
21                      return true;
22                  }
```

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

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

1.3.4 KM-BFS

```
1   /*KM by lbn O(n^3)
2   UOJ80 左nl 右nr 二分图最大权匹配
3   给定每两个点间权值，求一个匹配使权值和最大，不存在的边权开成 ?1，时间复杂度 O(n^3)。
4   用法：两个点间权值 wi;j， lxi 和 lyi 为顶标，随时满足 lxi + lyi ≥ wi;j， lki 为右部第 i
        个点匹配的左部点。如果要求最小值全部取反即可。
5   #define INF 1e9 不存在的边权开到-n*(|maxv|+1),inf为3n*(|maxv|+1) （点数 最大边权）
6   lx左，ly右
7   点从1~n编号*/
8
9   #define ll long long
10
11  const int N = 410; //二分图某一边的最大点数
12  const int INF = 0x3f3f3f3f;
13  int lx[N],ly[N],w[N][N],lk[N],slk[N],pre[N],vy[N],
14     py,d,p;
15  int nl, nr, m, n, x, y, z;
16
17  int main() {
18
19     /*初始化别忘了*/
20     memset(lx, 0, sizeof(lx));
21     memset(ly, 0, sizeof(ly));
22     memset(lk, 0, sizeof(lk));
23     memset(w, 0, sizeof(w));
24     cin >> nl >> nr >> m;
25     /*二分图的大小*/
26     n = max(nl,nr);
27
28     /* 负权图初始化要加 hdu2813 */
29     for (int i = 1; i <= n; ++i) {
30        for (int j = 1; j <= n; ++j) w[i][j] = -INF;
31        ly[i] = -INF;
32     }
33     /* ----------------------*/
34
35     for (int i = 1; i <= m; ++i)
36     {
```

```
        cin >> x >> y >> z;
        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!
        lx[y]=max(lx[y],z); //负权可以不写？但我一直写的
    }
    /* -----------KM---------- */
    for (int i = 1; i <= n; ++i)
    {
        for (int j = 1; j <= n; ++j) slk[j]=INF,vy[j]=0;
        for(lk[py=0]=i;lk[py];py=p)
        {
            vy[py]=1; d=INF; x=lk[py];
            for (int y = 1; y <= n; ++y) {
             if(!vy[y])
             {
                if(lx[x]+ly[y]-w[x][y]<slk[y]) slk[y]=lx[x]+ly[y]-w[x][y],pre[y]=py;
                if(slk[y]<d) d=slk[y],p=y;
             }
            }
            for (int y = 0; y <= n; ++y) {
             if(vy[y]) lx[lk[y]]-=d,ly[y]+=d;
              else slk[y]-=d;
            }

        }
        for(;py;py=pre[py]) lk[py]=lk[pre[py]];
    }
    ll ans = 0;
    正权 : for (int i = 1; i <= n; ++i) ans+=lx[i]+ly[i];
    负权 : for (int i = 1; i <= n; ++i) if (w[lk[i]][i] != -INF) ans+=w[lk[i]][i];
    printf("%lld\n",ans);
    for (int i = 1; i <= nl; ++i) {
     if(w[lk[i]][i]) printf("%d ",lk[i]);
      else printf("0 ");
    }
    puts("");
        //输出方案
      /*---------------------------------*/
    return 0;
}
```

1.3.5 带花树

```
//时间复杂度O(n^3)
#include <bits/stdc++.h>

using namespace std;

const int maxn = 1100;
int n, m, x, y;
vector<int> G[maxn];
namespace Blossom {
    int mate[maxn], n, ret, nxt[maxn], f[maxn], mark[maxn], vis[maxn], t;
```

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

```
                }
            }
        }
    }
    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;
}
```

1.3.6 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。
```

1.3.7 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
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

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。

二分图的判断：染色法
```

```
假设dfs初始点a涂黑色，并与他相邻的点就涂白色。
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。

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
于同一个顶点，则称M是一个匹配。
最大匹配：包含边数最多的匹配。
完美匹配（完备匹配）：所有点都在匹配边上的匹配。
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。

匈牙利算法 O(V * E)
HK算法 O(sqrt(V) * E)
```

1.4 2-SAT

1.4.1 输出任意解

```cpp
//洛谷 P4782
//从0开始的偶数为false，从1开始的奇数为true
const int N = 2*(int)EDIT+100; //点数*2
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;
        }
    }
    cout<<"POSSIBLE"<<'\n';
    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
        if (belong[i] > belong[i^1]) { //false->true 也就是只能为真
            cout<<1<<' ';
        } else cout<<0<<' ';
    }
    cout<<'\n';
}
void addEdge(int a, int b) {
    G[a].emplace_back(b);
}
int n,m,a,ai,b,bi;
int main() {
    cin>>n>>m;
    init(n);
    for(int i=0; i<m; ++i){
        cin>>a>>ai>>b>>bi;
        --a;--b;
        a*=2;b*=2;
        if(ai==0 && bi==0){
            addEdge(a^1,b);
            addEdge(b^1,a);
        } else if(ai==0 && bi==1){
            addEdge(a^1,b^1);
            addEdge(b,a);
        } else if(ai==1 && bi==0){
            addEdge(a,b);
            addEdge(b^1,a^1);
        } else{
            addEdge(a,b^1);
            addEdge(b,a^1);
        }
    }
    twoSat(n);
    return 0;
}
```

1.4.2 输出字典序最小解

```
// hdu 1814
// O(N*(N+M))
```

```cpp
// 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。
const int N=2*EDIT+100;
vector< vector<int> > G(N);
int n,m;
bool vis[N]; //染色标记，true表示选择
int stk[N],top; //栈
void init(int n){
    for(int i=0; i<=2*n; ++i) G[i].clear();
}
void addEdge(int u,int v){
    G[u].emplace_back(v);
}
bool dfs(int now){
    if(vis[now^1]) return false;
    if(vis[now]) return true;
    vis[now]=true;
    stk[top++]=now;
    for(int to:G[now]){
        if(!dfs(to)) return false;
    }
    return true;
}
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';
        }
    }
```

```
55      return 0;
56  }
```

1.4.3 思路

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

1.5 强连通

1.5.1 有向可有环图

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

```
36          belong[temp] = scc;
37          vis[temp] = 0;
38      } while (temp != x);
39   }
40}
41
42int n, m;
43vi in, out;
44
45int main() {
46   while (cin >> n >> m) {
47      init(n); //初始化别忘了
48      forn(i, m) {
49         int u, v;
50         cin >> u >> v;
51         G[u].eb(v); //建图
52      }
53      for (int i = 1; i <= n; ++i) if (!dfn[i]) tarjan(i); //tarjan求强连通
54      /* solving */
55      if (scc == 1) {
56         cout << 0 << '\n';
57         continue;
58      }
59      in = vi(scc+1, 0);
60      out = vi(scc+1, 0);
61      for (int i = 1; i <= n; ++i) {
62         for (int j : G[i]) {
63            if (belong[i] == belong[j]) continue; //缩点，同一个强连通分量之间不用连边
64            ++out[belong[i]];
65            ++in[belong[j]];
66         }
67      }
68      int in0 = 0, out0 = 0;
69      for (int i = 1; i <= scc; ++i) {
70         if (!in[i]) ++in0;
71         if (!out[i]) ++out0;
72      }
73      cout << max(in0, out0) << '\n';
74      /* end of solving */
75   }
76   return 0;
77}
```

1.6 双连通

1.6.1 割点桥

```
1//时间复杂度O(V+E)
2//add_block[i]:割掉i以后会多产生的连通块数，即割掉后图中的连通块数为add_block[i]+1
3const int N = edit+100; //点数
4vector<int> E[N];
5struct BCC {
```

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

```
59          for (int i = 1; i <= n; ++i) {
60              int t = fa[i];
61              if (belong[i] != belong[t]) {
62                  E[belong[i]].push_back(belong[t]);
63                  E[belong[t]].push_back(belong[i]);
64              }
65          }
66      }
67  }bcc;
68  int n, m;
69
70  int main() {
71      cin >> n >> m;
72      for(int i=0; i<=n; ++i) E[i].clear(); //初始化
73      forn(i, m) {
74          int x, y;
75          cin >> x >> y; //顶点编号从1~n
76          E[x].eb(y);
77          E[y].eb(x);
78      }
79      bcc.solve(n, E);
80      vi res;
81      for1(i, n) if (bcc.cut[i]) res.eb(i);
82      cout << SZ(res) << '\n';
83      for (int x : res) cout << x << ' ';
84      cout << '\n';
85      return 0;
86  }
```

1.6.2 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
```

1.6.3 经典例题

```
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
题意:
最少加几条边使整张图边bcc。
思路:
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。

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。
```

1.6.4 POJ 2942

```cpp
#include <stdio.h>
#include <vector>
#include <algorithm>
#include <string.h>
#include <limits.h>
#include <string>
#include <iostream>
#include <queue>
#include <math.h>
#include <map>
```

```cpp
#include <stack>
#include <sstream>
#include <set>
#include <iterator>
#include <list>
#include <cstdio>
#include <iomanip>
#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){
```

```
64          int to=G[now][i];
65          if(belong[now]!=belong[to]) continue;
66          if(col[to]==-1){
67              if(findOddCircle(to,c^1)) return true;
68          }
69          if(col[to]==col[now]) return true;
70      }
71      return false;
72  }
73  void dfs(int x, int pre) {
74      stk[top++] = x;
75      low[x] = dfn[x] = ++tot;
76      fa[x] = pre;
77      int son = 0;
78      for (int i=0; i<int(G[x].size()); ++i) {
79          int to=int(G[x][i]);
80          if (to == pre) continue;
81          if (!dfn[to]) {
82              ++son;
83              dfs(to, x);
84              low[x] = min(low[x], low[to]);
85              if (low[to] >= dfn[x]) {
86                  cut[x] = 1;
87                  add_block[x]++;
88                  ++bcc;
89                  int temp;
90                  vector<int> p; p.clear();
91                  do {
92                      temp = stk[--top];
93                      p.pb(temp);
94                      belong[ temp] = bcc;
95                  } while (temp != to); //魔改bcc，得出所有的bcc(包括大bcc包含的小的bcc)
96                  belong[x]=bcc;
97                  p.pb(x);
98                  if(int(p.size())>=3){
99                      ms(col,-1);
100                     if(findOddCircle(temp,0)) for(int j=0; j<int(p.size()); ++j) vis
                            [p[j]]=1;
101                 }
102             }
103             if (low[to] > dfn[x]) bridge.push_back(make_pair(x, to));
104         }
105         else low[x] = min(low[x], dfn[to]);
106     }
107     if (x == pre && son > 1) {
108         cut[x] = 1;
109         add_block[x] = son-1;
110     }
111 }
112 void solve(int _n, vector<int> E[]) {
113     n = _n;
114     for (int i = 1; i <= n; ++i) {
115         belong[i]=0;
```

```
116             G[i] = E[i];
117             low[i] = dfn[i] = stk[i] = fa[i] = 0;
118             cut[i] = add_block[i] = 0;
119             vis[i]=0;
120         }
121         bcc = top = tot = 0;
122         bridge.clear();
123         for (int i = 1; i <= n; ++i) if (!dfn[i]) dfs(i, i);
124     }
125     void rebuild(vector<int> E[]) {
126         for (int i = 1; i <= n; ++i) E[i].clear();
127         for (int i = 1; i <= n; ++i) {
128             int t = fa[i];
129             if (belong[i] != belong[t]) {
130                 E[belong[i]].push_back(belong[t]);
131                 E[belong[t]].push_back(belong[i]);
132             }
133         }
134     }
135 }bcc;
136 signed main() {
137     ios::sync_with_stdio(false);
138     cin.tie(0);
139     cout.precision(10);
140     cout << fixed;
141 #ifdef LOCAL_DEFINE
142     freopen("input.txt", "r", stdin);
143 #endif
144     while(~scanf("%d%d",&n,&m)){
145         if(!n && !m) break;
146         for(int i=0; i<=n; ++i){
147             E[i].clear();
148             for(int j=0; j<=n; ++j) mep[i][j]=0;
149         }
150         forn(i, m){
151             int u,v;
152             scanf("%d%d",&u,&v);
153             mep[u][v]=mep[v][u]=1;
154         }
155         for(int i=1; i<=n; ++i){
156             for(int j=1; j<=n; ++j){
157                 if(i==j) continue; //别tm建自环
158                 if(!mep[i][j]){
159                     E[i].pb(j);
160                 }
161             }
162         }
163         bcc.solve(n,E);
164         int sum=n;
165         for(int i=1; i<=n; ++i) if(bcc.vis[i]) --sum;
166         printf("%d\n",sum);
167     }
168 #ifdef LOCAL_DEFINE
```

```
    cerr << "Time elapsed: " << 1.0 * clock() / CLOCKS_PER_SEC << " s.\n";
#endif
    return 0;
}
```

1.7 欧拉回路

1.7.1 模板

```
//下面O(n+m)求欧拉回路的代码中，n为点数，m为边数。
//若有解则一次输出经过的边的编号。
//若是无向图，则正数表示x到y，负数表示y到x。
const int N=点数;
const int M=边数;
namespace UndirectedGraph{
    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;
    int ans[M],cnt;
    void addEdge(int x,int y,int z){
        d[x]++;
        v[++ed]=y; w[ed]=z; nxt[ed]=g[x]; g[x]=ed;
    }
    void dfs(int x){
        for(int &i=g[x];i;){
            if(vis[i]){i=nxt[i];continue;}
            vis[i]=vis[i^1]=1;
            int j=w[i];
            dfs(v[i]);
            ans[++cnt]=j;
        }
    }
    void solve(){
        scanf("%d%d",&n,&m);
        ed=1;
        for(int i=0; i<=n; ++i) cnt=d[i]=g[i]=0;
        for(int i=m+1; i<=2*m+10; ++i) vis[i]=0;
        for(int i=0; i<=m; ++i) vis[i]=ans[i]=0;
        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);
        for(i=1; i<=n; ++i) if(d[i]&1) {puts("NO");return;}
        for(i=1; i<=n; ++i) if(g[i]) {dfs(i);break;}
        for(i=1; i<=n; ++i) if(g[i]) {puts("NO");return;}
        puts("YES");
        for(i=m; i; i--) printf("%d ",ans[i]);
    }
}
namespace DirectedGraph{
    int n,m,i,x,y,d[N],g[N],v[M],vis[M],nxt[M],ed;
    int ans[M],cnt;
    void addEdge(int x,int y){
        d[x]++; d[y]--;
        v[++ed]=y; nxt[ed]=g[x]; g[x]=ed;
    }
```

```
43      void dfs(int x){
44          for(int &i=g[x];i;){
45              if(vis[i]){i=nxt[i];continue;}
46              vis[i]=1;
47              int j=i;
48              dfs(v[i]);
49              ans[++cnt]=j;
50          }
51      }
52      void solve(){
53          scanf("%d%d",&n,&m);
54          ed=0;
55          for(int i=0; i<=n; ++i) cnt=d[i]=g[i]=0;
56          for(int i=0; i<=m+10; ++i) vis[i]=ans[i]=0;
57          for(i=1; i<=m; ++i) scanf("%d%d",&x,&y),addEdge(x,y);
58          for(i=1; i<=n; ++i) if(d[i]) {puts("NO");return;}
59          for(i=1; i<=n; ++i) if(g[i]) {dfs(i);break;}
60          for(i=1; i<=n; ++i) if(g[i]) {puts("NO");return;}
61          puts("YES");
62          for(i=m; i; i--) printf("%d ",ans[i]);
63      }
64  }
```

1.7.2 知识点

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

1.8 LCA

1.8.1 ST 表

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

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

```
68        for(int i = 0;i <= n; i++)
69        {
70            loc[i] = 0;
71            dist[i] = 0;
72            len[i] = 0;
73            G[i].clear();
74        }
75    }
76    void addEdge(int a,int b,type w=1)
77    {
78        G[a].emplace_back(node(b,w));
79        G[b].emplace_back(node(a,w));
80    }
81    void solve(int root)
82    {
83        dfs(root, 1);
84        initRMQ(all);
85    }
86    #undef type
87 }lca;
88
89 int main() {
90
91    n = read();
92    lca.init(n);
93    for (int i = 0; i < n-1; ++i) {
94        int a, b;
95        a = read(); b = read();
96        lca.addEdge(a, b, 1);
97    }
98    lca.solve(1);
99    q = read();
100   while (q--) {
101       int a, b;
102       a = read(); b = read();
103       printf("%d\n", lca.queryLCA(a, b));
104       printf("%d\n", lca.getLen(a, b)); //深度 1
105       printf("%d\n", lca.getDist(a, b)); //长度 w
106   }
107
108   return 0;
109 }
```

1.8.2 离线

```
1 //P3379
2 //时间复杂度O(V + E)
3 const int maxn = 500000+1000;
4 int n, m, s, ans[maxn], vis[maxn], fa[maxn];
5 vvi g(maxn);
6 vector< vector< pair<int, int> > > q(maxn);
7
```

```cpp
int find_root(int x) {
    if (x == fa[x]) return x;
    else {
        return (fa[x] = find_root(fa[x]));
    }
}
void uni_root(int a, int b) {
    int aa = find_root(a);
    int bb = find_root(b);
    if (aa != bb) fa[bb] = aa;
}
void LCA(int u, int par) {
    for (int to : g[u]) {
        if (to == par) continue;
        LCA(to, u);
        uni_root(u, to);
    }
    vis[u] = 1;
    for (auto it : q[u]) {
        if (vis[it.fi]) {
            ans[it.se] = find_root(it.fi);
        }
    }
}

int main() {

    cin >> n >> m >> s;
    forn(i, n-1) {
        int u, v;
        cin >> u >> v;
        g[u].eb(v);
        g[v].eb(u);
    }
    forn(i, m) {
        int u, v;
        cin >> u >> v;
        q[u].pb({v, i});
        q[v].pb({u, i});
    }
    ms(vis, 0);
    for1(i, n) fa[i] = i;
    LCA(s, 0);
    forn(i, m) cout << ans[i] << '\n';

    return 0;
}
```

1.9 最大团

1.9.1 Bron-Kerbosch

```cpp
//O(3^(n/3))
#include <bits/stdc++.h>
using namespace std;
int n;
const int maxn = 60;
int ma[maxn], g[maxn][maxn], f[maxn][maxn], ans;
int dfs(int cur, int tot) {
   if (!cur) {
      if (tot > ans) return ans = tot, 1;
      return 0;
   }
   for (int i = 0, j, u, nxt; i < cur; ++i) {
      if (cur - i + tot <= ans) return 0;
      u = f[tot][i], nxt = 0;
      if (ma[u] + tot <= ans) return 0;
      for (int j = i + 1; j < cur; ++j) if (g[u][f[tot][j]]) f[tot + 1][nxt++] = f[
          tot][j];
      if (dfs(nxt, tot + 1)) return 1;
   }
   return 0;
}
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;
}
```

1.9.2 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
```

1.10 拓扑排序

1.10.1 toposort

```cpp
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;
}
```

2 计算几何

2.1 点

```cpp
const double eps = 1e-8;
const double inf = 1e20;
const double pi = acos(-1.0);
```

```cpp
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);
    }
    //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;
    }
    //返回长度
    double len() {
        return hypot(x, y); //库函数
    }
    //返回长度的平方
    double len2() {
        return x * x + y * y;
    }
    //返回两点的距离
    double distance(Point p) {
        return hypot(x - p.x, y - p.y);
    }
    //返回两点距离的平方
    double distance2(Point p) {
```

```
57          return (x-p.x)*(x-p.x)+(y-p.y)*(y-p.y);
58      }
59      Point operator + (const Point &b) const {
60          return Point(x + b.x, y + b.y);
61      }
62      Point operator * (const double &k) const {
63          return Point(x * k, y * k);
64      }
65      Point operator / (const double &k) const {
66          return Point(x / k, y / k);
67      }
68      //计算 pa 和 pb 的夹角
69      //就是求这个点看a, b的夹角
70      //测试LightOJ 1203
71      double rad(Point a, Point b) {
72          Point p = *this;
73          return fabs(atan2(fabs((a - p) ^ (b - p)), (a - p) * (b - p)));
74      }
75      //化为长度为r的向量
76      Point trunc(double r) {
77          double l = len();
78          if (!sgn(l)) return *this;
79          r /= l;
80          return Point(x * r, y * r);
81      }
82      //逆时针旋转 90 度
83      Point rotleft() {
84          return Point(-y, x);
85      }
86      //顺时针旋转 90 度
87      Point rotright() {
88          return Point(y, -x);
89      }
90      //绕着 p 点逆时针旋转 angle
91      Point rotat(Point p, double angle) {
92          Point v = (*this) - p;
93          double c = cos(angle), s = sin(angle);
94          return Point(p.x + v.x * c - v.y * s, p.y + v.x * s + v.y * c);
95      }
96  };
97  //叉积
98  double cross(Point A,Point B,Point C){
99      return (B-A)^(C-A);
100 }
101 //点积
102 double dot(Point A,Point B,Point C){
103     return (B-A)*(C-A);
104 }
```

2.2 线

```
1   struct Line {
```

```
    Point s, e;
    Line() {}
    Line(Point _s, Point _e) {
        s = _s;
        e = _e;
    }
    bool operator == (Line v) {
        return (s == v.s) && (e == v.e);
    }
    //根据一个点和倾斜角 angle 确定直线, 0 <= angle < pi
    Line (Point p, double angle) {
        s = p;
        if (sgn(angle - pi / 2) == 0) {
            e = (s + Point(0, 1));
        } else {
            e = (s + Point(1, tan(angle)));
        }
    }
    //ax + by + c == 0
    Line(double a, double b, double c) {
        if (sgn(a) == 0) {
            s = Point(0, -c / b);
            e = Point(1, -c / b);
        } else if (sgn(b) == 0) {
            s = Point(-c / a, 0);
            e = Point(-c / a, 1);
        } else {
            s = Point(0, -c / b);
            e = Point(1, (-c - a) / b);
        }
    }
    void input() {
        s.input();
        e.input();
    }
    void adjust() {
        if (e < s) swap(s, e);
    }
    //求线段长度
    double length() {
        return s.distance(e);
    }
    //返回直线倾斜角
    double angle() {
        double k = atan2(e.y - s.y, e.x - s.x);
        if (sgn(k) < 0) k += pi;
        if (sgn(k - pi) == 0) k -= pi;
        return k;
    }
    //点和直线关系
    //1 线在点的左侧
    //2 线在点的右侧
    //3 点在直线上
```

```
55      int relation(Point p) {
56          int c = sgn((p - s) ^ (e - s));
57          if (c < 0) return 1;
58          else if (c > 0) return 2;
59          else return 3;
60      }
61      //点在线段上的判断
62      bool pointonseg(Point p) {
63          return sgn((p - s) ^ (e-s)) == 0 && sgn((p - s) * (p - e)) <= 0;
64      }
65      //两向量平行（对应直线平行或重合）
66      bool parallel(Line v) {
67          return sgn((e - s) ^ (v.e - v.s)) == 0;
68      }
69      //两线段相交判断
70      //2 规范相交
71      //1 非规范相交
72      //0 不相交
73      int segcrossseg(Line v) {
74          int d1 = sgn((e - s) ^ (v.s - s));
75          int d2 = sgn((e - s) ^ (v.e - s));
76          int d3 = sgn((v.e - v.s) ^ (s - v.s));
77          int d4 = sgn((v.e - v.s) ^ (e - v.s));
78          if ((d1 ^ d2) == -2 && (d3 ^ d4) == -2) return 2;
79          return (d1 == 0 && sgn((v.s - s) * (v.s - e)) <= 0) ||
80              (d2 == 0 && sgn((v.e - s) * (v.e - e)) <= 0) ||
81              (d3 == 0 && sgn((s - v.s) * (s - v.e)) <= 0) ||
82              (d4 == 0 && sgn((e - v.s) * (e - v.e)) <= 0);
83      }
84      //直线和线段相交判断
85      //-*this line -v seg
86      //2 规范相交
87      //1 非规范相交
88      //0 不相交
89      int linecrossseg(Line v) {
90          int d1 = sgn((e - s) ^ (v.s - s));
91          int d2 = sgn((e - s) ^ (v.e - s));
92          if ((d1 ^ d2) == -2) return 2;
93          return (d1 == 0 || d2 == 0);
94      }
95      //两直线关系
96      //0 平行
97      //1 重合
98      //2 相交
99      int linecrossline(Line v) {
100         if ((*this).parallel(v))
101             return v.relation(s) == 3;
102         return 2;
103     }
104     //求两直线的交点
105     //要保证两直线不平行或重合
106     Point crosspoint(Line v) {
107         double a1 = (v.e - v.s) ^ (s - v.s);
```

```
108         double a2 = (v.e - v.s) ^ (e - v.s);
109         return Point((s.x * a2 - e.x * a1) / (a2 - a1),
110                      (s.y * a2 - e.y * a1) / (a2 - a1));
111     }
112     //点到直线的距离
113     double dispointtoline(Point p) {
114         return fabs((p - s) ^ (e - s)) / length();
115     }
116     //点到线段的距离
117     double dispointtoseg(Point p) {
118         if (sgn((p - s) * (e - s)) < 0 || sgn((p - e) * (s - e)) < 0)
119             return min(p.distance(s), p.distance(e));
120         return dispointtoline(p);
121     }
122     //返回线段到线段的距离
123     //前提是两线段不相交，相交距离就是0了
124     double dissegtoseg(Line v) {
125         return min(min(dispointtoseg(v.s), dispointtoseg(v.e)),
126                    min(v.dispointtoseg(s), dispointtoseg(e)));
127     }
128     //返回p在直线上的投影
129     Point lineprog(Point p) {
130         return s + ( ((e - s) * ((e - s) * (p - s))) / ((e - s).len2()) );
131     }
132     //返回点p关于直线的对称点
133     Point symmetrypoint(Point p) {
134         Point q = lineprog(p);
135         return Point(2 * q.x - p.x, 2 * q.y - p.y);
136     }
137 };
```

2.3 圆

```
1  struct Circle{
2      Point p;
3      double r;
4      Circle(){}
5      Circle(Point _p,double _r){
6          p=_p;
7          r=_r;
8      }
9      Circle(double x,double y,double _r){
10         p=Point(x,y);
11         r=_r;
12     }
13     //面积
14     double area(){
15         return pi*r*r;
16     }
17     //周长
18     double circumference(){
19         return 2*pi*r;
```

```
    }
};
```

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();
    }
    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]);
    }
    struct cmp{
        Point p;
        cmp(const Point &p0){p=p0;}
        bool operator()(const Point &aa,const Point &bb){
            Point a=aa,b=bb;
            int d=sgn((a-p)^(b-p));
            if(d==0){
                return sgn(a.distance(p)-b.distance(p))<0;
            }
            return d>0;
        }
    };
    //进行极角排序
    //首先需要找到最左下角的点
    //需要重载号好Point的<操作符（min函数要用）
    void norm(){
        Point mi=p[0];
        for(int i=1;i<n;++i) mi=min(mi,p[i]);
        sort(p,p+n,cmp(mi));
    }
    //得到凸包
    //得到凸包里面的点编号是0~n-1的
    //两种凸包的方法
    //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
    //测试 LightOJ1203 Light1239
    void getconvex(polygon &convex){
        sort(p,p+n);
        convex.n=n;
        for(int i=0; i<min(n,2); ++i) convex.p[i]=p[i];
        if(convex.n==2 && (convex.p[0]==convex.p[1])) convex.n--; //特判
        if(n<=2) return;
        int &top=convex.n;
        top=1;
```

```
48        for(int i=2; i<n; ++i){
49            while(top && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i]))<=0) --top;
50            convex.p[++top]=p[i];
51        }
52        int temp=top;
53        convex.p[++top]=p[n-2];
54        for(int i=n-3; i>=0; --i){
55            while(top!=temp && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i]))<=0)
                  top--;
56            convex.p[++top]=p[i];
57        }
58        if(convex.n==2 && (convex.p[0]==convex.p[1])) convex.n--;//特判
59        convex.norm(); //原来的得到的是顺时针的点，排序后逆时针。
60    }
61    //得到凸包的另外一种方法
62    //测试 LightOJ1203 LightOJ1239
63    void Graham(polygon &convex){
64        norm();
65        int &top=convex.n;
66        top=0;
67        if(n==1){
68            top=1;
69            convex.p[0]=p[0];
70            return;
71        }
72        if(n==2){
73            top=2;
74            convex.p[0]=p[0];
75            convex.p[1]=p[1];
76            if(convex.p[0]==convex.p[1]) --top;
77            return;
78        }
79        convex.p[0]=p[0];
80        convex.p[1]=p[1];
81        top=2;
82        for(int i=2; i<n; ++i){
83            while(top>1 && sgn((convex.p[top-1]-convex.p[top-2])^
84                               (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
85            convex.p[top++]=p[i];
86        }
87        if(convex.n==2 && convex.p[0]==convex.p[1]) convex.n--; //特判
88    }
89    //判断是不是凸的
90    bool isconvex(){
91        bool s[3];
92        memset(s,false,sizeof(s));
93        for(int i=0; i<n; ++i){
94            int j=(i+1)%n;
95            int k=(j+1)%n;
96            s[sgn((p[j]-p[i])^(p[k]-p[i]))+1]=true;
97            if(s[0] && s[2]) return false;
98        }
```

```
99          return true;
100     }
101     //旋转卡壳
102     double rotate_calipers(){
103         double ans=0;
104         if(n==1) return ans;
105         else if(n==2){
106             ans=max(ans,p[0].distance(p[1]));
107             return ans;
108         }
109         else if(n==3){
110             ans=max(ans, p[0].distance(p[1]));
111             ans=max(ans, p[0].distance(p[2]));
112             ans=max(ans, p[1].distance(p[2]));
113             return ans;
114         } else{
115             int i,j=1;
116             p[n++]=p[0];
117             for(int i=0; i<n; ++i){
118                 for(;cross(p[i+1],p[j+1],p[i])>cross(p[i+1],p[j],p[i]);j=(j+1)%n);
119                 ans=max(ans, max(p[i].distance(p[j]),p[i+1].distance(p[j])));//最远点对
120             }
121             return ans;
122         }
123     }
124     //得到周长
125     //测试 LightOj1239
126     double getcircumference(){
127         double sum=0;
128         for(int i=0; i<n; ++i){
129             sum+=p[i].distance(p[(i+1)%n]);
130         }
131         return sum;
132     }
133     //得到面积
134     double getarea(){
135         double sum=0;
136         for(int i=0; i<n; ++i) sum+=(p[i]^p[(i+1)%n]);
137         return fabs(sum)/2;
138     }
139 };
```

2.5 function

```
1 //判断三点是否共线
2 bool sameLine(Point a, Point b, Point c){
3     double ax=c.x-a.x, ay=c.y-a.y;
4     double bx=c.x-b.x, by=c.y-b.y;
5     if(sgn(ax*by-ay*bx)==0)return true;
6     return false;
7 }
```

2.6 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！！
```

2.7 凸包

2.7.1 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;
}
```

2.8 旋转卡壳

2.8.1 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 {
```

```
36          return sgn(x - b.x) == 0 ? sgn(y - b.y) < 0 : x < b.x;
37      }
38      Point operator - (const Point &b) const {
39          return Point(x - b.x, y - b.y, idx - b.idx);
40      }
41      //点积
42      double operator * (const Point &b) const {
43          return x * b.x + y * b.y;
44      }
45      //叉积
46      double operator ^ (const Point &b) const {
47          return x * b.y - y * b.x;
48      }
49      //返回两点的距离
50      double distance(Point p) {
51          return hypot(x - p.x, y - p.y);
52      }
53  }fp;
54  struct polygon{
55      int n;
56      Point p[maxp];
57      void input(int _n){
58          n=_n;
59          for(int i=0; i<n; ++i){
60              p[i].input();
61              p[i].idx=i;
62          }
63      }
64      void add(Point q){
65          p[n++]=q;
66      }
67      struct cmp{
68          Point p;
69          cmp(const Point &p0){p=p0;}
70          bool operator()(const Point &aa,const Point &bb){
71              Point a=aa,b=bb;
72              int d=sgn((a-p)^(b-p));
73              if(d==0){
74                  return sgn(a.distance(p)-b.distance(p))<0;
75              }
76              return d>0;
77          }
78      };
79      //进行极角排序
80      //首先需要找到最左下角的点
81      //需要重载号好Point的<操作符（min函数要用）
82      void norm(){
83          Point mi=p[0];
84          for(int i=1;i<n;++i) mi=min(mi,p[i]);
85          sort(p,p+n,cmp(mi));
86      }
87      //得到凸包
88      //得到凸包里面的点编号是0~n-1的
```

```
89  //两种凸包的方法
90  //注意如果有影响，要特判下所有点共点，或者共线的特殊情况
91  //测试 LightOJ1203 Light1239
92  void getconvex(polygon &convex){
93      sort(p,p+n);
94      convex.n=n;
95      for(int i=0; i<min(n,2); ++i) convex.p[i]=p[i];
96      if(convex.n==2 && (convex.p[0]==convex.p[1])) convex.n--; //特判
97      if(n<=2) return;
98      int &top=convex.n;
99      top=1;
100     for(int i=2; i<n; ++i){
101         while(top && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i]))<=0) --top;
102         convex.p[++top]=p[i];
103     }
104     int temp=top;
105     convex.p[++top]=p[n-2];
106     for(int i=n-3; i>=0; --i){
107         while(top!=temp && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i]))<=0)
                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     {
```

```
            for (int i=0;i<n;i++)
            {
                int x=(i+1)%n,y=(i+2)%n;
                for (int j=(i+2)%n;j!=i;(++j)%=n)
                {
                    while (x!=j && OK(p[i],p[j],p[x],p[x+1])) (++x)%=n;
                    while (y!=i && OK(p[i],p[j],p[y],p[y+1])) (++y)%=n;
                    now=fabs((p[x]-p[i])^(p[j]-p[i]))+fabs((p[y]-p[i])^(p[j]-p[i]));
                    if (now>ans) ans=now;
                }
            }
        }
        printf("%.3lf\n",ans/2.0);
        return ;
    }
}plo;
int main() {
    int tc;
    scanf("%d",&tc);
    while(tc--){
        scanf("%d",&n);
        plo.input(n);
        polygon cv; plo.getconvex(cv); //得到凸包
        cv.Rotating_Calipers(plo); //旋转卡壳
    }
    return 0;
}


//n<=4000, x,y是整数, x,y<=1e9 5s
#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 {
    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);
```

```
193     }
194     void output() {
195         printf("%.8f %.8f\n", x, y);
196     }
197     bool operator == (Point b) const {
198         return sgn(x - b.x) == 0 && sgn(y - b.y) == 0;
199     }
200     bool operator < (Point b) const {
201         return sgn(x - b.x) == 0 ? sgn(y - b.y) < 0 : x < b.x;
202     }
203     Point operator - (const Point &b) const {
204         return Point(x - b.x, y - b.y, idx - b.idx);
205     }
206     //点积
207     //叉积
208     i64 operator ^ (const Point &b) const {
209         return x * b.y - y * b.x;
210     }
211     //返回两点的距离
212     double distance(Point p) {
213         return hypot(x - p.x, y - p.y);
214     }
215 }fp;
216 struct polygon{
217     int n;
218     Point p[maxp];
219     void input(int _n){
220         n=_n;
221         for(int i=0; i<n; ++i){
222             p[i].input();
223             p[i].idx=i;
224         }
225     }
226     void add(Point q){
227         p[n++]=q;
228     }
229     struct cmp{
230         Point p;
231         cmp(const Point &p0){p=p0;}
232         bool operator()(const Point &aa,const Point &bb){
233             Point a=aa,b=bb;
234             int d=sgn((a-p)^(b-p));
235             if(d==0){
236                 return sgn(a.distance(p)-b.distance(p))<0;
237             }
238             return d>0;
239         }
240     };
241     //进行极角排序
242     //首先需要找到最左下角的点
243     //需要重载号好Point的<操作符（min函数要用）
244     void norm(){
245         Point mi=p[0];
```

```
246         for(int i=1;i<n;++i) mi=min(mi,p[i]);
247         sort(p,p+n,cmp(mi));
248     }
249     //得到凸包
250     //得到凸包里面的点编号是0~n-1的
251     //两种凸包的方法
252     //注意如果有影响，要特判下所有点共点，或者共线的特殊情况
253     //测试 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 }
```

2.9 扫描线

2.9.1 矩形面积交 hdu1255

```
1  const int maxn = ; //两倍的矩形数量，最好开大点
2  int n;
3  double v[maxn];
4  struct Line {
5      double x_;
6      double y_, y__;
7      int state;
8      bool operator < (const Line &oth) const {return x_<oth.x_;}
9  }line[maxn];
10 struct s1 {
11     int tl, tr;
12     double l, r;
13     int cover;
```

```
14      double len, more;
15  }sgt[maxn<<3];
16
17  void pushUp(int node) {
18      int l = sgt[node].tl; int r = sgt[node].tr;
19      if (sgt[node].cover >= 2) {
20          sgt[node].more = sgt[node].len = sgt[node].r-sgt[node].l;
21      } else if (sgt[node].cover == 1) {
22          sgt[node].len = sgt[node].r-sgt[node].l;
23          sgt[node].more = sgt[node<<1].len+sgt[node<<1|1].len;
24      } else {
25          sgt[node].len = sgt[node<<1].len+sgt[node<<1|1].len;
26          sgt[node].more = sgt[node<<1].more+sgt[node<<1|1].more;
27      }
28  }
29  void build(int node, int l, int r) {
30      sgt[node].tl = l; sgt[node].tr = r;
31      sgt[node].l = v[l]; sgt[node].r = v[r];
32      sgt[node].cover = 0;
33      sgt[node].len = 0;
34      sgt[node].more = 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, double ul, double ur, int val) {
41      double 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);
49      pushUp(node);
50  }
51  int tc;
52
53  int main() {
54
55      scanf("%d", &tc);
56      while (tc--) {
57          scanf("%d", &n);
58          for1(i, n) {
59              double x_, y_, x__, y__;
60              scanf("%lf%lf%lf%lf", &x_, &y_, &x__, &y__);
61              //cin >> x_ >> y_ >> x__ >> y__;
62              v[i] = y_; v[i+n] = y__;
63              line[i] = (Line){x_, y_, y__, 1};
64              line[i+n] = (Line){x__, y_, y__, -1};
65          }
66          sort(v+1, v+1+(n<<1));
```

```
67        sort(line+1, line+1+(n<<1));
68        build(1, 1, n<<1);
69        double ans = 0;
70        for (int i = 1; i <= (n<<1); ++i) {
71            ans += sgt[1].more*(line[i].x_-line[i-1].x_);
72            update(1, line[i].y_, line[i].y__, line[i].state);
73            //cerr << i << ' ' << ans << endl;
74        }
75        //cout << ans << '\n';
76        printf("%.2f\n", ans);
77    }
78
79    return 0;
80 }
```

2.9.2 面积并坐标 double 版

```
1  const int maxn = 2*(int)1e6+1000;
2  int n;
3  double v[maxn];
4  struct Line {
5     double x_;
6     double y_, y__;
7     int state;
8     bool operator < (const Line &oth) const {return x_<oth.x_;}
9  }line[maxn];
10 struct s1 {
11    double l, r;
12    int cover;
13    double len;
14 }sgt[maxn<<3];
15
16 void pushUp(int node) {
17    if (sgt[node].cover) sgt[node].len = sgt[node].r-sgt[node].l;
18    else sgt[node].len = sgt[node<<1].len+sgt[node<<1|1].len;
19 }
20 void build(int node, int l, int r) {
21    sgt[node].l = v[l]; sgt[node].r = v[r];
22    sgt[node].cover = 0;
23    sgt[node].len = 0;
24    if (r-l <= 1) return;
25    int mid = (l+r)>>1;
26    build(node<<1, l, mid);
27    build(node<<1|1, mid, r);
28 }
29 void update(int node, double ul, double ur, int val) {
30    double l = sgt[node].l, r = sgt[node].r;
31    if (ul <= l && r <= ur) {
32       sgt[node].cover += val;
33       pushUp(node);
34       return;
35    }
```

```
36      if (ul < sgt[node<<1].r) update(node<<1, ul, ur, val);
37      if (ur > sgt[node<<1|1].l) update(node<<1|1, ul, ur, val);
38      pushUp(node);
39  }
40
41  int main() {
42
43      while (cin >> n) {
44          if (!n) break;
45          for1(i, n) {
46              double x_, y_, x__, y__;
47              cin >> x_ >> y_ >> x__ >> y__;
48              v[i] = y_; v[i+n] = y__;
49              line[i] = (Line){x_, y_, y__, 1};
50              line[i+n] = (Line){x__, y_, y__, -1};
51          }
52          sort(v+1, v+1+(n<<1));
53          sort(line+1, line+1+(n<<1));
54          build(1, 1, n<<1);
55          double ans = 0;
56          for (int i = 1; i <= (n<<1); ++i) {
57              ans += sgt[1].len*(line[i].x_-line[i-1].x_);
58              update(1, line[i].y_, line[i].y__, line[i].state);
59          }
60          cout << ans << '\n';
61      }
62
63      return 0;
64  }
```

2.9.3 面积并坐标 i64 版 O(nlgn)

```
1   const int maxn = 2*(int)1e5+1000; //边的个数=矩形的个数*2
2   int n;
3   int v[maxn];
4   struct Line {
5       int x_;
6       int y_, y__;
7       int state;
8       bool operator < (const Line &oth) const {return x_<oth.x_;}
9   }line[maxn];
10  struct s1 {
11      int l, r;
12      int cover;
13      i64 len;
14  }sgt[maxn<<3]; //注意这个大小
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) {
```

```
    sgt[node].l = v[l]; sgt[node].r = v[r];
    sgt[node].cover = 0;
    sgt[node].len = 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, int ul, int ur, int val) {
    int l = sgt[node].l, r = sgt[node].r;
    if (ul <= l && r <= ur) {
        sgt[node].cover += val;
        pushUp(node);
        return;
    }
    if (ul < sgt[node<<1].r) update(node<<1, ul, ur, val);
    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
    pushUp(node);
}

int main() {

    cin >> n;
    for1(i, n) {
        int x_, y_, x__, y__;
        cin >> x_ >> y_ >> x__ >> y__;
        v[i] = y_; v[i+n] = y__;
        line[i] = (Line){x_, y_, y__, 1};
        line[i+n] = (Line){x__, y_, y__, -1};
    }
    sort(v+1, v+1+(n<<1));
    sort(line+1, line+1+(n<<1));
    build(1, 1, n<<1);
    i64 ans = 0;
    for (int i = 1; i <= (n<<1); ++i) {
        ans += sgt[1].len*(line[i].x_-line[i-1].x_);
        update(1, line[i].y_, line[i].y__, line[i].state);
    }
    cout << ans << '\n';

    return 0;
}
```

2.9.4 周长并 O(nlgn)

```
const int maxn = 2*(int)1e5+1000;
int n;
int v[maxn];
struct Line {
    int x_;
    int y_, y__;
```

```
7     int state;
8     bool operator < (const Line &oth) const {return x_<oth.x_;}
9  }line[maxn];
10 struct s1 {
11    int l, r;
12    int cover;
13    i64 len;
14    int lcover, rcover;
15    int diff;
16 }sgt[maxn<<3]; //注意这个大小
17 void pushUp(int node) {
18    if (sgt[node].cover) {
19       sgt[node].len = sgt[node].r-sgt[node].l;
20       sgt[node].lcover = sgt[node].rcover = 1;
21       sgt[node].diff = 1;
22    } else {
23       sgt[node].len = sgt[node<<1].len+sgt[node<<1|1].len;
24       sgt[node].lcover = sgt[node<<1].lcover;
25       sgt[node].rcover = sgt[node<<1|1].rcover;
26       sgt[node].diff = sgt[node<<1].diff+sgt[node<<1|1].diff-(sgt[node<<1].rcover&
             sgt[node<<1|1].lcover);
27    }
28 }
29 void build(int node, int l, int r) {
30    sgt[node].l = v[l]; sgt[node].r = v[r];
31    sgt[node].cover = 0;
32    sgt[node].len = 0;
33    sgt[node].lcover = sgt[node].rcover = 0;
34    sgt[node].diff = 0;
35    if (r-l <= 1) return;
36    int mid = (l+r)>>1;
37    build(node<<1, l, mid);
38    build(node<<1|1, mid, r);
39 }
40 void update(int node, int ul, int ur, int val) {
41    int l = sgt[node].l, r = sgt[node].r;
42    if (ul <= l && r <= ur) {
43       sgt[node].cover += val;
44       pushUp(node);
45       return;
46    }
47    if (ul < sgt[node<<1].r) update(node<<1, ul, ur, val);
48    if (ur > sgt[node<<1|1].l) update(node<<1|1, ul, ur, val); //或者ur>sgt[node<<1].
          r 因为sgt[node<<1].r==sgt[node<<1|1].l
49    pushUp(node);
50 }
51 i64 myabs(i64 a) {return (a>0?a:-a);}
52
53 int main() {
54
55    cin >> n;
56    for1(i, n) {
57       int x_, y_, x__, y__;
```

```
58        cin >> x_ >> y_ >> x__ >> y__; //左下角坐标 右上角坐标
59        v[i] = y_; v[i+n] = y__;
60        line[i] = (Line){x_, y_, y__, 1};
61        line[i+n] = (Line){x__, y_, y__, -1};
62    }
63    sort(v+1, v+1+(n<<1));
64    sort(line+1, line+1+(n<<1));
65    build(1, 1, n<<1);
66    i64 ans = 0;
67    i64 lst = 0;
68    for (int i = 1; i <= (n<<1); ++i) {
69        ans += (line[i].x_-line[i-1].x_)*2*sgt[1].diff; //vertical
70        update(1, line[i].y_, line[i].y__, line[i].state);
71        ans += myabs(sgt[1].len-lst); //horizontal
72        lst = sgt[1].len;
73    }
74    cout << ans << '\n';
75
76    return 0;
77 }
```

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 {
```

```
    ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
    cin >> n >> m;
    build_tree(1, n, 1);
    while (m--) {
        cin >> op;
        if (op == 1) { //check in
            cin >> o;
            if (o > sum[1]) {
                cout << 0 << endl;
            } else {
                int pos = query_tree(1, n, 1, o);
                cout << pos << endl;
                update_tree(1, n, 1, 1, pos, pos + o - 1);
            }
        } else if (op == 2) { //check out
            cin >> o >> p;
            update_tree(1, n, 1, 0, o, o + p - 1);
        }
    }
    return 0;
}
```

3.2 树状数组

3.2.1 逆序对

```
// 兔子的逆序对 https://ac.nowcoder.com/acm/problem/20861
// 每交换一对数字，就会改变数组逆序对个数的奇偶性
const int maxn = (int)1e5 + 100;
const int cst = (int)1e5;
int n, m, a[maxn], b[maxn], ft[maxn];

void init() {
    memset(ft, 0, sizeof(ft));
}
int query(int x) {
    ll res = 0;
    for (int i = x; i >= 1; i -= lowbit(i)) res += ft[i];
    return res;
}
void update(int x, int val) {
    for (int i = x; i <= cst; i += lowbit(i)) ft[i] += val;
}

int main() {
    init();
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> a[i];
    ll sum = 0;
    for (int i = 1; i <= n; ++i) {
        sum += query(cst) - query(a[i]);
        update(a[i], 1);
```

```
    }
    bool odd = (sum & 1);
    cin >> m;
    for (int i = 0; i < m; ++i) {
        int l, r;
        cin >> l >> r;
        int len = r - l + 1;
        len /= 2;
        if (len & 1) {
            if (odd) odd = false;
            else odd = true;
        }
        cout << (odd ? "dislike" : "like") << '\n';
    }
    return 0;
}
```

3.3 主席树

3.3.1 静态查询区间第 k 大

```
//P 3834
const int maxn = 2*(int)1e5+1000;
int n, m, a[maxn], b[maxn], tVal[maxn*40], t[maxn*40], lt[maxn*40], rt[maxn*40], tot
    ;

int build(int l, int r) {
    int node = ++tot;
    tVal[node] = 0;
    int mid = (l + r) >> 1;
    if (l < r) {
        lt[node] = build(l, mid);
        rt[node] = build(mid + 1, r);
    }
    return node;
}
int update(int l, int r, int par, int p) {
    int node = ++tot;
    lt[node] = lt[par]; rt[node] = rt[par]; tVal[node] = tVal[par] + 1;
    int mid = (l + r) >> 1;
    if (l < r) {
        if (p <= mid) lt[node] = update(l, mid, lt[par], p); //lt[par]!!
        else rt[node] = update(mid + 1, r, rt[par], p); //rt[par]!!
    }
    return node; //node!!
}
int query(int n1, int n2, int l, int r, int k) {
    if (l == r) return l;
    int mid = (l + r) >> 1;
    int sum = tVal[lt[n2]] - tVal[lt[n1]];
    if (sum >= k) return query(lt[n1], lt[n2], l, mid, k);
    else return query(rt[n1], rt[n2], mid + 1, r, k - sum);
```

```
}
void init() {
    tot = 0;
}

int main() {

    init(); //初始化
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
        b[i] = a[i];
    }
    sort(b + 1, b + 1 + n);
    int len = unique(b + 1, b + 1 + n) - b - 1;
    t[0] = build(1, len);
    for (int i = 1; i <= n; ++i) {
        int p = lower_bound(b + 1, b + 1 + len, a[i]) - b;
        t[i] = update(1, len, t[i - 1], p);
    }
    for (int i = 0; i < m; ++i) {
        int l, r, k;
        cin >> l >> r >> k;
        int p = query(t[l - 1], t[r], 1, len, k);
        cout << b[p] << '\n';
    }
    return 0;
}
```

3.3.2 动态查询区间第 k 大

```
//p2617
#define lowbit(x) (x&(-x))
const int maxn = (int)1e5+100;
int n, m, a[maxn], b[maxn<<1], len; //len: 离散化后数组的长度
int t[maxn*60], lt[maxn*360], rt[maxn*360], tVal[maxn*360], tot; //tot: 动态开点
int n1, n2, t1[maxn], t2[maxn];
struct qq {
    int l, r, ra, val;
    char op;
}q[maxn];

void update(int &node, int l, int r, int p, int val) {
    if (!node) node = ++tot;
    tVal[node] += val;
    int mid = (l + r) >> 1;
    if (l == r) return;
    if (p <= mid) update(lt[node], l, mid, p, val);
    else update(rt[node], mid + 1, r, p, val);
}
void change(int idx, int val) {
    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  }
```

3.4 字典树

3.4.1 trie 树

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

```
36 void init() { //多组数据慎用，要么把字典树开小一点
37     memset(trie, 0, sizeof(trie));
38     memset(sum, 0, sizeof(sum));
39     memset(ed, 0, sizeof (ed));
40     tot = 0;
41     flag = false;
42 }
43
44 int main() {
45     int kase = 1;
46     init();
47     while (~scanf("%s", s)) {
48         if (strlen(s) == 1 && s[0] == '9') {
49             if (flag == false) printf("Set %d is immediately decodable\n", kase++);
50             else printf("Set %d is not immediately decodable\n", kase++);
51             init();
52         } else {
53             Insert(s);
54         }
55     }
56     return 0;
57 }
```

4 字符串

4.1 KMP

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

```cpp
//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);
        }
        dfs(1, 0);

        for (int i = 1; i <= n; ++i) {
            int maxx = n - sz[i];
            for (int j = 0; j < int(G[i].size()); ++j) {
                int to = G[i][j];
                if (to == fa[i]) continue; //!!
                maxx = max(maxx, sz[to]);
            }
            res[i] = maxx;
        }

```

```
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);
```

```
    }
}

int main() {
    while (cin >> n) {
        init(n);
        for (int i = 2; i <= n; ++i) {
            int u, val;
            cin >> u >> val;
            G[i].emplace_back(make_pair(u, val));
            G[u].emplace_back(make_pair(i, val));
        }
        dfs(1, 0);
        dfs2(1, 0);
        for (int i = 1; i <= n; ++i) {
            cout << max(f[i], g[i]) << '\n';
        }
    }
    return 0;
}
```

6 树上问题

6.1 树的直径

```
int n;
vvi g;
vi p;

pii dfs(int v, int par = -1, int dist = 0) {
    p[v] = par;
    pii res = mp(dist, v);
    for (auto to : g[v]) {
        if (to == par) continue;
        res = max(res, dfs(to, v, dist + 1));
    }
    return res;
}

int main() {

    cin >> n;
    g = vvi(n);
    p = vi(n);
    forn(i, n - 1) {
        int v1, v2;
        cin >> v1 >> v2;
        --v1; --v2;
        g[v1].eb(v2);
        g[v2].eb(v1);
    }
    pii da = dfs(0);
```

```
    pii db = dfs(da.se);
    vi diam;
    int v = db.se;
    while (v != -1) {
        diam.eb(v);
        v = p[v];
    }
    if (int(diam.size()) == n) { //直径就是整个图，也就是整个图就是一条链
        cout << n - 1 << '\n';
        cout << diam[0] + 1 << " " << diam[1] + 1 << " " << diam.back() + 1 << '\n';
        return 0;
    }
    queue<int> q;
    vi d(n, -1);
    for (int u : diam) {
        d[u] = 0;
        q.push(u);
    }
    while (!q.empty()) {
        int v = q.front();
        q.pop();
        for (auto to : g[v]) {
            if (d[to] == -1) {
                d[to] = d[v] + 1;
                q.push(to);
            }
        }
    }
    pii mx = mp(d[0], 0);
    forn(i, n) {
        mx = max(mx, mp(d[i], i));
    }
    cout << int(diam.size()) - 1 + mx.fi << '\n';
    cout << diam[0] + 1 << " " << diam.back() + 1 << " " << mx.se + 1 << '\n';
    return 0;
}
```

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 二分三分

8.2.1 二分三分

8.3 莫队

8.3.1 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
```

8.3.2 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;
}
```

8.3.3 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
#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;

```

```cpp
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);
    putchar('0' + x % 10);
}
void add(int x) {
    --cnt[times[x]];
    ++times[x];
    ++cnt[times[x]];
}
void cut(int x) {
    --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];
```

```
    }
    int tot = n;
    forn(i, m) {
        int op, l, r;
        op = read(); l = read(); r = read();
        if (op == 1)
            ++qdx, que[qdx].set(l, r, mdx, qdx);
        if (op == 2) {
            ++mdx;
            rl[mdx] = l;
            ++tot;
            rr[mdx] = a[tot] = b[tot] = r;
        }
    }
    sort(b + 1, b + 1 + tot);
    int dig = unique(b + 1, b + 1 + tot) - b - 1;
    for1(i, n) {
        d[i] = lower_bound(b + 1, b + 1 + dig, a[i]) - b;
        c[i] = d[i];
    }
    for1(i, mdx) {
        int tmp = lower_bound(b + 1, b + 1 + dig, rr[i]) - b;
        mod[i].set(rl[i], 0, tmp);
    }
    for1(i, mdx) {
        mod[i].x = c[mod[i].wz];
        c[mod[i].wz] = mod[i].y;
    }
    work();
    for1(i, qdx) {
        write(ans[i]);
        puts("");
    }


#ifdef LOCAL_DEFINE
    cerr << "Time elapsed: " << 1.0 * clock() / CLOCKS_PER_SEC << " s.\n";
#endif
    return 0;
}
```

8.4 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
```

8.5 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;
}
```

8.6 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;
}
```

8.7 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());
```

8.8 输入一行看有多少个数

```
//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();
}
```

8.9 最小（大）表示法

```
//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);
}
```

8.10 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;
}
```

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

```
#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

```
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

```
//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;
```

```
22 namespace linear_seq {
23    const i64 N=10010;
24    i64 res[N],base[N],_c[N],_md[N];
25
26    vector<i64> Md;
27    void mul(i64 *a,i64 *b,i64 k) {
28       rep(i,0,k+k) _c[i]=0;
29       rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
30       for (i64 i=k+k-1; i>=k; i--) if (_c[i])
31             rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
32       rep(i,0,k) a[i]=_c[i];
33    }
34    i64 solve(i64 n,vi64 a,vi64 b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]+...
35       i64 ans=0,pnt=0;
36       i64 k=SZ(a);
37       assert(SZ(a)==SZ(b));
38       rep(i,0,k) _md[k-1-i]=-a[i];
39       _md[k]=1;
40       Md.clear();
41       rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
42       rep(i,0,k) res[i]=base[i]=0;
43       res[0]=1;
44       while ((1ll<<pnt)<=n) pnt++;
45       for (i64 p=pnt; p>=0; p--) {
46          mul(res,res,k);
47          if ((n>>p)&1) {
48             for (i64 i=k-1; i>=0; i--) res[i+1]=res[i];
49             res[0]=0;
50             rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
51          }
52       }
53       rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
54       if (ans<0) ans+=mod;
55       return ans;
56    }
57    vi64 BM(vi64 s) {
58       vi64 C(1,1),B(1,1);
59       i64 L=0,m=1,b=1;
60       rep(n,0,SZ(s)) {
61          i64 d=0;
62          rep(i,0,L+1) d=(d+(i64)C[i]*s[n-i])%mod;
63          if (d==0) ++m;
64          else if (2*L<=n) {
65             vi64 T=C;
66             i64 c=mod-d*powmod(b,mod-2)%mod;
67             while (SZ(C)<SZ(B)+m) C.pb(0);
68             rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
69             L=n+1-L;
70             B=T;
71             b=d;
72             m=1;
73          } else {
74             i64 c=mod-d*powmod(b,mod-2)%mod;
```

```
            while (SZ(C)<SZ(B)+m) C.pb(0);
            rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
            ++m;
        }
    }
    return C;
  }
  i64 gao(vi64 a,i64 n) {
    vi64 c=BM(a);
    c.erase(c.begin());
    rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
    return solve(n,c,vi64(a.begin(),a.begin()+SZ(c)));
  }
};

i64 m, f[300];

int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  cin >> n >> m;
  for (int i = 0; i < m; ++i) f[i] = 1;
  for (int i = m; i <= 250; ++i) {
    f[i] = (f[i - 1] + f[i - m]) % mod;
  }
  vi64 v;
  for (int i = 1; i <= 250; ++i) {
    v.emplace_back(f[i]);
  }
  cout << linear_seq::gao(v,n-1)%mod << '\n';
  return 0;
}
```

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 坑点

```
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。
17ccpc 秦皇岛的java题，64M，t=20，每次for3000次就会爆。
```

10.1.2 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));
```

10.1.3 注意点

```
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。
17ccpc 秦皇岛的java题，64M，t=20，每次for3000次就会爆。
```

10.2 python

10.2.1 python

```
//https://ac.nowcoder.com/acm/contest/5670/E
def gcd(a, b):
   if b == 0 : return a
   else : return gcd(b, a % b)
n = int(input())
a = [1]
vis = [0] * (n + 1)
temp = input().split()
for x in temp :
   tx = int(x)
   a.append(tx)
ans = 1
for i in range(1, 1 + n) :
   u = i
   cnt = 0
   if vis[u] == 1 : continue
   while vis[u] == 0 :
      vis[u] = 1;
      cnt += 1
      u = a[u]
   ans = (ans * cnt) // gcd(ans, cnt)
ans = int(ans)
```

```
23 print(ans)
```