ACM_Template

FreeArcanite@Wuhan University

Last build at October 16, 2013

Contents

1 图论

1.1 2-sat

```
int n, m;
vector <int> mp[MAXN << 1];
bool mark[MAXN << 1];
int S[MAXN << 1], c;

bool dfs(int u) {
    if(mark[u ^ 1]) return false;
    if(mark[u])     return true;
    mark[u] = true;
    S[c ++] = u;
    for(int i = 0; i < SZ(mp[u]); ++ i)
        if(!dfs(mp[u][i]))    return false;
    return true;
}

void init() {
    for(int i = 0; i < (n << 1); ++ i) mp[i].clear();
    memset(mark, false, sizeof(mark));
}

void add_clause(int x, int xval, int y, int yval) {
    x = x * 2 + xval;
    y = y * 2 + yval;
    mp[x ^ 1].push_back(y);
    mp[y ^ 1].push_back(x);
}

bool solve() {
    for(int i = 0; i < (n << 1); i += 2)
        if(!mark[i] && !mark[i + 1]) {
            c = 0;
            if(!dfs(i)) {
                while(c > 0)    mark[S[-- c]] = false;
                if(!dfs(i + 1)) return false;
            }
        }
    return true;
}
```

1.2 构建双连通图 Tarjan

求出所有的桥，删除桥边，将连通块缩点，在将桥边连接回来，构成树，叶子节点为leaf
添加的边数为(leaf + 1) / 2
具体方法为，首先把两个最近公共祖先最远的两个叶节点之间连接一条边，这样可以把这两个点到祖先的路径上所有点收缩到一起，因为一个形成的环一定是双连通的。然后再找两个最近公共祖先最远的两个叶节点，这样一对一对找完，恰好是(leaf+1)/2次，把所有点收缩到了一起。

1.3 欧拉环和哈密尔顿路径

欧拉路径(一笔画问题)
连通的无向图 G 有欧拉路径的充要条件是：G中奇顶点（连接的边数量为奇数的顶点）的数目等于0或者2。
连通的无向图 G 是欧拉环（存在欧拉回路）的充要条件是：G中每个顶点的度都是偶数。
一个有向图是欧拉图当且仅当该图的基图（将所有有向边变为无向边后形成的无向图，这里同样不考虑度数为0的点）是连通的且所有点的入度等于出度
一个有向图是半欧拉图当且仅当该图的基图是连通的且有且只有一个点的入度比出度少1（作为欧拉路径的起点），有且只有一个点的入度比出度多1（作为终点），其余点的入度等于出度。

哈密尔顿图
设G=(V,E)是一个无向简单图，|V|=n. n⩾3.
若对于任意的两个顶点u，v∈V，d(u)+d(v) ⩾n，
那么, G是哈密尔顿图

以下为非递归求欧拉路径

```
int v = edge_num - 1;

void dfs() {
    int x = 0, y, tp = 1;
    stk[0] = 0;
    for(int i = 0; i < n; ++ i) {
        now[i] = head[i];
    }
    bool flag;
    while(tp) {
        flag = false;
        for(int p = now[x]; p != -1; p = edge[p].next)
            if(!vis[p]) {
                y = vis[p].v;
                vis[edge[p]] = true;
                vis[edge[p ^ 1]] = true; //有向图删除此句
                now[x] = p;
                stk[tp ++] = y;
                x = y;
                flag = true;
                break;
            }
        if(!flag) {
            res[v --] = x + 1;
            x = stk[--tp - 1];
        }
    }
}
```

1.4 求双连通分量 Tarjan

对于边双连通分支，直接删除所有的桥/割边即可

以下为点双连通分支求法

```
int cnt, top;
vector <int> adj[MAXN];
int dfn[MAXN], low[MAXN], sta[MAXN];

void get_bicon_com(int pre, int u) {
    sta[top ++] = u;
    dfn[u] = low[u] = ++ cnt;
    for(int i = 0; i < adj[u].size(); ++ i) {
        int v = adj[u][i];
        if(!dfn[v]) {
            get_bicon_com(u, v);
            low[u] = min(low[u], low[v]);
            if(dfn[u] <= low[v]) {
                vector <int> tmp;
                tmp.push_back(u);
                for(sta[top] = 0; sta[top] != v; tmp.push_back(sta
                    [-- top]));
                //中存储的为点双连通分支tmp
            }
        }
        else if(v != pre)   low[u] = min(low[u], dfn[v]);
    }
    return;
}

void init() {
    top = 0;
    cnt = 0;
    memset(dfn, 0, sizeof(dfn));
    memset(low, 0, sizeof(low));
}
```

1.5 求割边/桥 Tarjan

求割边/桥(不含重边) Tarjan

```
    vector <int> adj[MAXN];
bool invis[MAXN];
int dfn[MAXN], low[MAXN];

void Get_bridge(int pre, int u) {
    invis[u] = true;
    dfn[u] = low[u] = ++ cnt;
    for(int i = 0; i < adj[u].size(); ++ i) {
        int v = adj[u][i];
        if(!dfn[v]) {
            Get_bridge(u, v);
            if(dfn[u] < low[v]) {
                //u 即为割边v
            }
```

```
            low[u] = min(low[u], low[v]);
        }
        else if(v != pre && invis[v])
            low[u] = min(low[u], dfn[v]);
    }
    invis[u] = false;
    return;
}

void init() {
    cnt = 0;
    memset(invis, false, sizeof(invis));
    memset(dfn, 0, sizeof(dfn));
    memset(low, 0, sizeof(low));
}
```

求割边/桥(含重边) Tarjan

```
void tarjan(int k,int father)
{
  dfn[k]=low[k]=++NOW;
    for(int x=last[k];x;x=pre[x])
    {
        if(!dfn[son[x]])
        {
      tag[son[x]]=id[x];
            tarjan(son[x],k);
            low[k]=min(low[k],low[son[x]]);
        }
        else
            if(tag[k]!=id[x]) low[k]=min(low[k],dfn[son[x]]);
    }
    if(tag[k] && dfn[k]==low[k])
    flag[tag[k]]=1;
}
```

1.6 求割点 Tarjan

child为根节点(root)的孩子数，大于1则是割点。

```
int cnt, root;
vector <int> adj[MAXN];
int dfn[MAXN], low[MAXN];
bool invis[MAXN];

int get_cut_point(int pre, int u) {
    int child = 0;
    int vis[u] = true;
    dfn[u] = low[u] = ++ cnt;
    for(int i = 0; i < adj[u].size(); ++ i) {
        int v = adj[u][i];
        if(!dfn[v]) {
            ++ child;
```

```
            get_cut_point(u, v);
            if(u != root && dfn[u] <= low[v])
                //是割点u
            low[u] = min(low[u], dfn[v]);
        }
        else if(v != pre && invis[v])
            low[u] = min(low[u], dfn[v]);
    }
    invis[u] = false;
    return child;
}

void init() {
    cnt = 0;
    memset(invis, false, sizeof(invis));
    memset(dfn, 0, sizeof(dfn));
    memset(low, 0, sizeof(low));
}
```

1.7 第K短路

A* + Dijstra-第K短路问题
估值函数: f = g + h
对于估值函数中的h
对于无向图，我们只需对原图做一次在 终点 的 dijstra ，就可以得到所有点到终点的最短距离
对于有向图，我们可以吧原图反向建边，得到一个新图，再在新图中对 终点 做一次 dijstra ，这样就可以求出各点到终点的最短距离
From s to t的第k短路

```
typedef struct Status {
    int pos, len, h;
    bool operator < (const Status &cmp) const {
        if(cmp.len + cmp.h == len + h)  return cmp.len < len;
        else                            return cmp.len + cmp.h <
           len + h;
    }
} Status;

typedef struct Edge {
    int v, next, w;
} Edge;

Edge node[MAXM + MAXN], in_node[MAXM + MAXN];
int dis[MAXN];
bool vis[MAXN];
int n, m, ednum, inednum;

int kth_path(int s, int t, int k) {
    if(dis[s] == INF)   return -1;
    if(s == t)  ++ k;
```

```
21      priority_queue <Status> q;
22      Status now;
23      now.pos = s, now.len = 0, now.h = dis[s];
24      q.push(now);
25      int cnt = 0;
26      while(!q.empty()) {
27          Status top = q.top(); q.pop();
28          if(top.pos == t) ++ cnt;
29          if(cnt == k)    return top.len;
30          for(int i = node[top.pos].next; i != -1; i = node[i].next)
               {
31              Status tmp;
32              tmp.pos = node[i].v, tmp.len = node[i].w + top.len, tmp
                   .h = dis[node[i].v];
33              q.push(tmp);
34          }
35      }
36      return -1;
37  }
```

1.8 最小生成树 Kruskal

Kruskal MST经过并查集优化。

```
1   typedef struct Line {
2       int u, v, c;
3       bool operator < (const Line &cmp) const {
4           return c < cmp.c;
5       }
6   } Line;
7
8   Line l[MAXN * MAXN];
9   bool vis[MAXN * MAXN];
10  int parent[MAXN], rk[MAXN];
11  int n, m;
12
13  void makeset(int n) {
14      for(int i = 1; i <= n; ++ i) {
15          parent[i] = i;
16          rk[i] = 0;
17      }
18  }
19
20  int getparent(int x) {
21      if(parent[x] == x)  return x;
22      else {
23          parent[x] = getparent(parent[x]);
24          return parent[x];
25      }
26  }
27
28  void unionset(int x, int y) {
```

```
29      if(rk[x] > rk[y])
30          parent[y] = x;
31      else {
32          parent[x] = y;
33          if(rk[x] == rk[y])
34              ++ rk[y];
35      }
36  }
37
38  void MST() {
39      int cnt = 0;
40      memset(vis, false, sizeof(vis));
41      sort(l, l + m);
42      makeset(n);
43      for(int i = 0; i < m; ++ i) {
44          int x = getparent(l[i].u);
45          int y = getparent(l[i].v);
46          if(x != y) {
47              unionset(x, y);
48              ++ cnt;
49              vis[i] = true;
50              if(cnt == n - 1)    break;
51          }
52      }
53      return;
54  }
```

1.9 二分图匹配 HK算法

```
1   int nx, ny;
2   int dx[4000], dy[4000], cx[4000], cy[4000];
3   vector<int> mp[4000];
4
5   bool bfs() {
6       bool tag = false;
7       queue<int> q;
8       memset(dx, 0, sizeof(dx));
9       memset(dy, 0, sizeof(dy));
10      for(int i = 0; i < nx; i ++)
11          if(cx[i] == -1)
12              q.push(i);
13      while(!q.empty()) {
14          int u = q.front();
15          q.pop();
16          for(int i = 0; i < (int)mp[u].size(); i ++) {
17              int v = mp[u][i];
18              if(!dy[v]) {
19                  dy[v] = dx[u] + 1;
20                  if(cy[v] == -1)
21                      tag = true;
22                  else {
```

```
                    dx[cy[v]] = dy[v] + 1;
                    q.push(cy[v]);
                }
            }
        }
    }
    return tag;
}

bool dfs(int u) {
    for(int i = 0; i < (int)mp[u].size(); i ++) {
        int v = mp[u][i];
        if(dy[v] == dx[u] + 1) {
            dy[v] = 0;
            if(cy[v] == -1 || dfs(cy[v])) {
                cx[u] = v;
                cy[v] = u;
                return true;
            }
        }
    }
    return false;
}

int hk() {
    int match = 0;
    memset(cx, -1, sizeof(cx));
    memset(cy, -1, sizeof(cy));
    while(bfs()) {
        for(int i = 0; i < nx; i ++)
            if(cx[i] == -1 && dfs(i))
                match ++;
    }
    return match;
}
```

1.10 最大团 最大独立集

最大团与最大独立集求解
最大独立集为原图的补图求最大团
以下为标准的最大团求解

```
int n, m;
int mx;
int map[110][110], stk[110][110];
int tt[110], dp[110], ans[110];

void dfs(int ns, int dep) {
    if(ns == 0) {
        if(dep > mx) {
            mx = dep;
            for(int i = 1; i <= dep; ++ i)
                ans[i] = tt[i]; //ans begin from 1
```

```
        }
        return;
    }
    for(int i = 0; i < ns; ++ i) {
        int k = stk[dep][i], cnt = 0;
        if(dep + n - k <= mx)   return;
        if(dep + dp[k] <= mx)   return;
        for(int j = i + 1; j < ns; ++ j) {
            int p = stk[dep][j];
            if(map[k][p])
                stk[dep + 1][cnt ++] = p;
        }
        tt[dep + 1] = k;
        dfs(cnt, dep + 1);
    }
}

int clique() {
    mx = 0;
    for(int i = n; i >= 1; -- i) {
        int cnt = 0;
        for(int j = i + 1; j <= n; ++ j)
            if(map[i][j])
                stk[1][cnt ++] = j;
        tt[1] = i;
        dfs(cnt, 1);
        dp[i] = mx;
    }
    return mx;
}
```

1.11 次小生成树

每个节点i遍历整个最小生成树，定义F[j]为从i到j的路径上最大边的权值。
遍历图求出F[j]的值，然后对于添加每条不在最小生成树中的边(i,j)，新的生成树权值之和就是MinST + w(i,j) - F[j]，记录其最小值，则为次小生成树。

```
const int INF = 0x7FFFFFFF;

typedef struct edge {
    int nxt, w, v;
    int op;
    bool flag;
} edge;

int ans;
int t, n, m;
int head1[MAXN], head2[MAXN], MST[MAXN], f[MAXN], rec[MAXN];
edge e[MAXN * MAXN], ne[MAXN * MAXN];
int edgenum1, edgenum2, sum;

```

```
15 inline void addedge(int head[], edge ed[], int &edgenum, int u, int
       v, int w) {
16     ed[edgenum].nxt = head[u];
17     ed[edgenum].flag = false;
18     ed[edgenum].v = v;
19     ed[edgenum].w = w;
20     ed[edgenum].op = edgenum + 1;
21     head[u] = edgenum;
22     ++ edgenum;
23     ed[edgenum].nxt = head[v];
24     ed[edgenum].flag = false;
25     ed[edgenum].v = u;
26     ed[edgenum].w = w;
27     ed[edgenum].op = edgenum - 1;
28     head[v] = edgenum;
29     ++ edgenum;
30 }
31
32 void prim() {
33     for(int i = 1; i <= n; ++ i)    MST[i] = INF;
34     for(int i = 1; ; ) {
35         MST[i] = - INF;
36         for(int j = head1[i]; j != -1; j = e[j].nxt) {
37             int v = -1;
38             if(e[j].w < MST[v = e[j].v]) {
39                 MST[v] = e[j].w;
40                 rec[v] = e[j].op;
41             }
42         }
43         int Minn = INF;
44         i = 0;
45         for(int j = 1; j <= n; ++ j)
46             if(MST[j] != - INF && MST[j] < Minn) {
47                 Minn = MST[j];
48                 i = j;
49             }
50         if(i == 0)
51             break;
52     }
53     for(int i = 2; i <= n; ++ i) {
54         sum += e[rec[i]].w;
55         int v = e[rec[i]].v;
56         e[rec[i]].flag = e[e[rec[i]].op].flag = true;
57         addedge(head2, ne, edgenum2, i, v, e[rec[i]].w);
58     }
59 }
60
61 void dfs(int u) {
62     for(int i = head2[u]; i != -1; i = ne[i].nxt) {
63         int v = ne[i].v;
64         if(f[v] == - INF) {
```

```
65              f[v] = f[u];
66              if(ne[i].w > f[v])
67                  f[v] = ne[i].w;
68              dfs(v);
69          }
70      }
71  }
72
73  void smst() {
74      for(int i = 1; i <= n; ++ i) {
75          for(int j = 1; j <= n; ++ j)    f[j] = - INF;
76          ++ f[i];
77          dfs(i);
78          for(int j = head1[i]; j != -1; j = e[j].nxt) {
79              int v = e[j].v;
80              if(!e[j].flag) {
81                  int tmp = sum + e[j].w - f[v];
82                  if(tmp < ans)
83                      ans = tmp;
84              }
85          }
86      }
87  }
88
89  void init() {
90      memset(head1, -1, sizeof(head1));
91      memset(head2, -1, sizeof(head2));
92      memset(rec, 0, sizeof(rec));
93      edgenum1 = edgenum2 = 0;
94      ans = INF;
95      sum = 0;
96      scanf("%d%d", &n, &m);
97      for(int i = 0; i < m; ++ i) {
98          int u, v, w;
99          scanf("%d%d%d", &u, &v, &w);
100         addedge(head1, e, edgenum1, u, v, w);
101     }
102 }
```

1.12 最短路 SPFA

SPFA-求最短路径/求负环

```
1  const int INF = 0x1fffffff;
2
3  typedef struct Edge {
4      int v, next, w;
5  } Edge;
6
7  Edge node[MAXN * MAXN];
8  bool vis[MAXN];
9  int in[MAXN];
```

```
10
11 bool spfa(int root) {
12     queue <int> q;
13     memset(vis, 0, sizeof(vis));
14     memset(in, 0, sizeof(in));
15     for(int i = 0; i < n; ++ i) dist[i] = INF;
16     vis[root] = true;
17     in[root] = 1;
18     dist[root] = 0;
19     q.push(root);
20     while(!q.empty()) {
21         int u = q.front(); q.pop();
22         vis[u] = false;
23         for(int son = node[u].next; son != -1; son = node[son].next
               ) {
24             int v = node[son].v;
25             int w = node[son].w;
26             if(dist[v] > dist[u] + w) {
27                 dist[v] = dist[u] + w;
28                 if(!vis[v]) {
29                     ++ in[v];
30                     if(in[v] > n)   return false;
31                     vis[v] = true;
32                     q.push(v);
33                 }
34             }
35         }
36     }
37     return true;
38 }
```

1.13 稳定婚姻问题

稳定婚姻问题，采用Gale-Shaplley算法。
第i个男生和第match_b[i]个女生结婚最稳定。

```
1  int n;
2  const int MAXN = 510;
3  int maze_g[MAXN][MAXN]; //用来存每个女生喜欢第几个男生的程度 排位从开始1
4  int maze_b[MAXN][MAXN]; //用来存每个男生第几个喜欢的女生是谁 排位从开始1
5  int match_g[MAXN], match_b[MAXN];
6  int p[MAXN];
7
8  void solve() {
9      memset(match_g, -1, sizeof(match_g));
10     memset(match_b, -1, sizeof(match_b));
11     for(int i = 1; i <= n; ++ i) p[i] = 1; //p[i用来标记男生的表白次
           数]
12     bool flag = true;
13     while(flag) {
14         flag = false;
```

```
        for(int i = 1; i <= n; ++ i) {
            if(match_b[i] == -1 && p[i] <= n) {
                int a = maze_b[i][p[i] ++];
                if(match_g[a] == -1) {
                    match_g[a] = i;
                    match_b[i] = a;
                }
                else {
                    if(maze_g[a][i] < maze_g[a][match_g[a]]) { //如果本轮男生更优秀则选择本轮男生
                        match_b[i] = a;
                        match_b[match_g[a]] = -1;
                        match_g[a] = i;
                    }
                }
                flag = true;
            }
        }
    }
}
```

1.14 仙人掌图直径

```
int ans,son[500000],last[500000],pre[500000],tot,NOW,dfn[500000],low[500000],a[500000],f[500000],father[500000],que[500000];
void add(int a,int b){tot++;son[tot]=b;pre[tot]=last[a];last[a]=tot;}
void dp(int x,int y)
{
  int m,i,head,tail,p;
  for(m=0;y!=x;y=father[y]) a[++m]=f[y];
  for(a[++m]=f[x],i=1;i<m;i++) a[m+i]=a[i];
  head=tail=1,que[1]=1;p=(m>>1);
  for(int i=2;i<=m+p;i++)
  {
    while(head<=tail && i-que[head]>p) head++;
    ans=max(ans,a[que[head]]+a[i]+i-que[head]);
    while(head<=tail && a[que[tail]]+i-que[head]<=a[i]) tail--;
    que[++tail]=i;
  }
  for(i=1;i<m;i++) f[x]=max(f[x],a[i]+min(i,m-i));
}
void tarjan(int k)
{
  dfn[k]=low[k]=++NOW;
  for(int x=last[k];x!=0;x=pre[x])
  {
    if(son[x]!=father[k])
    {
      if(!dfn[son[x]])
      {
```

```
27          father[son[x]]=k;
28          tarjan(son[x]);
29          low[k]=min(low[k],low[son[x]]);
30          if(dfn[k]<low[son[x]])
31          {
32            ans=max(ans,f[k]+f[son[x]]+1);
33            f[k]=max(f[k],f[son[x]]+1);
34          }
35        }
36        else low[k]=min(low[k],dfn[son[x]]);
37      }
38    }
39    for(int x=last[k];x!=0;x=pre[x])
40    {
41      if(father[son[x]]!=k && dfn[k]<dfn[son[x]]) dp(k,son[x]);
42    }
43  }
44  int main()
45  {
46    int a,b,c,n,m;scanf("%d␣%d",&n,&m);
47    for(int i=1;i<=m;i++)
48    {
49      scanf("%d␣%d",&c,&a);
50      for(int j=1;j<c;j++) scanf("%d",&b),add(a,b),add(b,a),a=b;
51    }
52    tarjan(1);
53    printf("%d\n",ans);
54    return 0;
55  }
```

2 网络流

2.1 费用流

```
bool spfa()
{
    int head,tail;
  memset(dist,127,sizeof(dist));
  que[head=tail=1]=s; dist[s]=0;
  while(head<=tail)
  {
    int p=que[head++];
    for(int x=last[p];x!=0;x=pre[x])
    {
      if(len[x]>0 && dist[son[x]]>dist[p]+cost[x])
      {
        dist[son[x]]=dist[p]+cost[x];
        P[son[x]]=x;
        que[++tail]=son[x];
      }
    }
  }
  return dist[t]<dist[100000];
}
int main()
{
  while(spfa())
  {
    flow=914990825;
    for(int now=t;now!=s;now=from[P[now]])
      flow=min(flow,len[P[now]]);
    for(int now=t;now!=s;now=from[P[now]])
    {
      len[P[now]]-=flow;
      if(P[now]&1) len[P[now]+1]+=flow;
      else len[P[now]-1]+=flow;
    }
    ans1+=flow;
    ans2+=dist[t]*flow;
  }
}
```

2.2 最大流 Dinic

用邻间表实现。

```
struct Edge {
    int u, v, cap;
};
int dep[1100], ptr[1100], n, s, t;
vector<Edge> e;
vector<int> g[1100];

```

```
8  inline void init() {
9      for(int i = 0; i < n; i ++)
10         g[i].clear();
11     e.clear();
12 }
13
14 inline void add_edge(int u, int v, int cap) {
15     Edge e1 = {u, v, cap};
16     Edge e2 = {v, u, 0};
17     g[u].push_back(e.size());
18     e.push_back(e1);
19     g[v].push_back(e.size());
20     e.push_back(e2);
21 }
22
23 bool bfs() {
24     queue<int> q;
25     memset(dep, -1, sizeof(dep));
26     dep[s] = 0;
27     q.push(s);
28     while(!q.empty()) {
29         int u = q.front();
30         q.pop();
31         for(int i = 0; i < (int)g[u].size(); i ++) {
32             int id = g[u][i];
33             int v = e[id].v;
34             if(dep[v] == -1 && e[id].cap) {
35                 dep[v] = dep[u] + 1;
36                 q.push(v);
37             }
38         }
39     }
40     return dep[t] != -1;
41 }
42
43 int dfs(int u, int flow) {
44     if(!flow) return 0;
45     if(u == t) return flow;
46     for(; ptr[u] < (int)g[u].size(); ptr[u] ++) {
47         int id = g[u][ptr[u]];
48         int v = e[id].v;
49         if(dep[u] + 1 != dep[v])
50             continue;
51         int add = dfs(v, min(flow, e[id].cap));
52         if(add) {
53             e[id].cap -= add;
54             e[id^1].cap += add;
55             return add;
56         }
57     }
58     return 0;
```

```
}

int dinic() {
    int maxflow = 0;
    while(bfs()) {
        memset(ptr, 0, sizeof(ptr));
        while(true) {
            int minflow = dfs(s, inf);
            if(minflow)
                maxflow += minflow;
            else break;
        }
    }
    return maxflow;
}
```

用链表实现。

```
bool bfs()
{
  while(!que.empty()) que.pop();
  memset(dist,-1,sizeof(dist));
  dist[s]=0;
  que.push(s);
  while(!que.empty())
  {
    int p=que.front();
    que.pop();
    for(int x=last[p];x!=0;x=pre[x])
    {
      if(len[x]>0 && dist[son[x]]<0)
      {
        dist[son[x]]=dist[p]+1;
        que.push(son[x]);
      }
    }
  }
  return dist[t]>0;
}
int find(int k,int flow)
{
  if(k==t) return flow;
  int f=0,tmp;
  for(int x=last[k];x!=0 && f<flow;x=pre[x])
  {
    if(len[x]>0 && dist[son[x]]==dist[k]+1)
    {
      tmp=find(son[x],min(flow-f,len[x]));
      if(tmp==0) dist[son[x]]=-1;
      else
      {
        f+=tmp;
```

```
35          len[x]-=tmp;
36          if(x&1) len[x+1]+=tmp;
37          else len[x-1]+=tmp;
38        }
39      }
40    }
41    return f;
42  }
43  int main()
44  {
45    int tmp,ans=0;
46    while(bfs())
47      while(tmp=find(s,914990825))
48        ans+=tmp;
49  }
```

2.3 最小费用最大流

```
1  int n, s, t, dis[200], pre[200];
2  struct Edge {
3      int u, v, cap, cost;
4  };
5  vector<int> g[200];
6  vector<Edge> e;
7
8  inline void init() {
9      for(int i = 0; i < n; i ++)
10         g[i].clear();
11     e.clear();
12 }
13
14 inline void add_edge(int u, int v, int cap, int cost) {
15     Edge e1 = {u, v, cap, cost};
16     Edge e2 = {v, u, 0, -cost};
17     g[u].pb(e.size());
18     e.pb(e1);
19     g[v].pb(e.size());
20     e.pb(e2);
21 }
22
23 bool spfa() {
24     bool inq[200];
25     for(int i = 0; i < n; i ++) {
26         pre[i] = -1;
27         dis[i] = inf;
28         inq[i] = false;
29     }
30     queue<int> q;
31     dis[s] = 0;
32     inq[s] = true;
33     q.push(s);
```

```
    while(!q.empty()) {
        int u = q.front();
        q.pop();
        inq[u] = false;
        for(int i = 0; i < (int)g[u].size(); i ++) {
            int id = g[u][i];
            int v = e[id].v;
            if(e[id].cap && get_min(dis[v], dis[u] + e[id].cost)) {
                pre[v] = id;
                if(!inq[v]) {
                    inq[v] = true;
                    q.push(v);
                }
            }
        }
    }
    return dis[t] < inf;
}

pair<int, int> mcmf() {
    int maxflow = 0;
    int mincost = 0;
    while(spfa()) {
        int minflow = inf;
        for(int i = pre[t]; i != -1; i = pre[e[i].u])
            get_min(minflow, e[i].cap);
        for(int i = pre[t]; i != -1; i = pre[e[i].u]) {
            e[i].cap -= minflow;
            e[i^1].cap += minflow;
        }
        maxflow += minflow;
        mincost += minflow * dis[t];
    }
    return make_pair(mincost, maxflow);
}

int main() {
    return 0;
}
```

2.4 上下界流

建图

```
for(int i = 1; i <= m;i++)
{
  scanf("%d %d %d %d",&a,&b,&c,&d);
    add(a,b,d-c);
    add(b,a,0);
    du[a]-=c;
    du[b]+=c;
}
```

处理有源汇有上下界最大流问题是:
1.构造附加网络 （添加 t→s 容量inf 添加ss和tt ss→v(du[v]¿0) v→tt(du[v]¡0) c(u,v) = 上界-下界 du[i]为所有流入i的下界减去所有流出i的下界
2.对ss、tt求最大流(ss、tt满流则有解)
3.若有解，对s、t求最大流

而有源汇有上下界最小流问题则是:
1.构造附加网络(不添加[t,s]边)
2.对ss、tt求最大流

3.添加[t,s]边
4.对ss、tt求最大流
5.若ss、tt满流，则[t,s]的流量就是最小流

3 数据结构

3.1 树链剖分

```
1  struct node
2  {
3    int MAX,sum,lc,rc,l,r;
4  }tree[500005];
5  int segpos[500005],dep[500005],now[500005],val[500005],son[500005],
       pre[500005],last[500005],tot,belong[500005],father[500005],size
       [500005],ROOT[500005];
6  void add(int a,int b){son[++tot]=b;pre[tot]=last[a];last[a]=tot;}
7  void maketree(int l,int r)
8  {
9    tree[tot].l=l,tree[tot].r=r;
10   int k=tot,mid=(l+r)>>1;
11   if(l==r)
12   {
13     tree[tot].sum=tree[tot].MAX=val[now[mid]];
14     return ;
15   }
16   tree[k].lc=++tot;maketree(l,mid);
17   tree[k].rc=++tot;maketree(mid+1,r);
18   tree[k].MAX=max(tree[tree[k].lc].MAX,tree[tree[k].rc].MAX);
19   tree[k].sum=tree[tree[k].lc].sum+tree[tree[k].rc].sum;
20 }
21 void change(int root,int pos,int k)
22 {
23   if(tree[root].l==tree[root].r)
24   {
25     tree[root].MAX=tree[root].sum=k;
26     return ;
27   }
28   int mid=(tree[root].l+tree[root].r)>>1;
29   if(pos<=mid) change(tree[root].lc,pos,k);
30   else change(tree[root].rc,pos,k);
31   tree[root].MAX=max(tree[tree[root].lc].MAX,tree[tree[root].rc].
       MAX);
32   tree[root].sum=tree[tree[root].lc].sum+tree[tree[root].rc].sum;
33 }
34 int findsum(int root,int a,int b)
35 {
36   if(tree[root].l>=a && tree[root].r<=b)
37     return tree[root].sum;
38   int mid=(tree[root].l+tree[root].r)>>1,sum=0;
39   if(a<=mid) sum+=findsum(tree[root].lc,a,b);
40   if(b>mid) sum+=findsum(tree[root].rc,a,b);
41   return sum;
42 }
43 int findmax(int root,int a,int b)
44 {
```

```
45    if(tree[root].l>=a && tree[root].r<=b)
46      return tree[root].MAX;
47    int mid=(tree[root].l+tree[root].r)>>1,MAX=-914990825;
48    if(a<=mid) MAX=max(MAX,findmax(tree[root].lc,a,b));
49    if(b>mid) MAX=max(MAX,findmax(tree[root].rc,a,b));
50    return MAX;
51  }
52  void dfs(int k)
53  {
54    size[k]=1; int MAX=0,maxn=-1;
55    for(int x=last[k];x!=0;x=pre[x])
56    {
57      if(son[x]!=father[k])
58      {
59        father[son[x]]=k;
60        dep[son[x]]=dep[k]+1;
61        dfs(son[x]);
62        size[k]+=size[son[x]];
63        if(size[son[x]]>MAX)
64        {
65          MAX=size[son[x]];
66          maxn=son[x];
67        }
68      }
69    }
70    if(maxn!=-1)
71      belong[maxn]=1;
72  }
73  void build(int k)
74  {
75    if(belong[k]) belong[k]=belong[father[k]];
76    else belong[k]=k;
77    int flag=0;
78    for(int x=last[k];x!=0;x=pre[x])
79      if(son[x]!=father[k])
80      {
81        build(son[x]);
82        flag=1;
83      }
84    if(!flag)
85    {
86      int cnt=0,i;
87      for(i=k;i!=belong[i];i=father[i])
88      {
89        segpos[i]=++cnt;
90        now[cnt]=i;
91      }
92      segpos[i]=++cnt; now[cnt]=i;
93      ROOT[belong[i]]=++tot;
94      maketree(1,cnt);
95    }
```

```
96  }
97  void find(int a,int b,int &MAX,int &sum)
98  {
99    MAX=-914990825; sum=0;
100   while(belong[a]!=belong[b])
101   {
102     if(dep[belong[a]]<dep[belong[b]])
103       swap(a,b);
104     sum+=findsum(ROOT[belong[a]],segpos[a],segpos[belong[a]]);
105     MAX=max(MAX,findmax(ROOT[belong[a]],segpos[a],segpos[belong[a
          ]]));
106     a=father[belong[a]];
107   }
108   if(segpos[a]>segpos[b])
109     swap(a,b);
110   sum+=findsum(ROOT[belong[a]],segpos[a],segpos[b]);
111   MAX=max(MAX,findmax(ROOT[belong[a]],segpos[a],segpos[b]));
112 }
113 int main()
114 {
115   int n,m,a,b;
116   char op[10];
117   scanf("%d",&n);
118   for(int i=1;i<=n-1;i++)
119   {
120     scanf("%d %d",&a,&b);
121     add(a,b),add(b,a);
122   }
123   for(int i=1;i<=n;i++)
124     scanf("%d",&val[i]);
125   tot=0; dfs(1); build(1);
126   scanf("%d",&m);
127   for(int i=1;i<=m;i++)
128   {
129     scanf("%s",op);
130     if(op[0]=='C')
131     {
132       scanf("%d %d",&a,&b);
133       change(ROOT[belong[a]],segpos[a],b);
134     }
135     else
136     {
137       scanf("%d %d",&a,&b);
138       int sum,MAX;
139       find(a,b,MAX,sum);
140       if(op[1]=='S') printf("%d\n",sum);
141       else printf("%d\n",MAX);
142     }
143   }
144   return 0;
145 }
```

3.2 Splay维护数列

```
struct node
{
  int val,lmax,rmax,MAX,size,sum,rev;
  bool sameflag;
  node *a[2],*fa;
  void set(node *t,int p){a[p]=t;t->fa=this;}
  int dir(){return this==fa->a[1];}
  void pushdown()
  {
    if(size==0) return;
    if(rev)
    {
      a[0]->rev^=1;
      a[1]->rev^=1;
      swap(a[0],a[1]);
      swap(lmax,rmax);
      rev=0;
    }
    if(sameflag)
    {
      a[0]->sameflag=a[1]->sameflag=1;
      a[0]->val=a[1]->val=val;
      sum=val*size;
      if(val>0) lmax=rmax=MAX=sum;
      else lmax=rmax=MAX=val;
      sameflag=0;
    }
  }
  void update()
  {
    if(size==0) return;
    this->pushdown();a[0]->pushdown();a[1]->pushdown();
    size=a[0]->size+a[1]->size+1;
    sum=a[0]->sum+a[1]->sum+val;
    lmax=max(a[0]->lmax,a[0]->sum+val+max(0,a[1]->lmax));
    rmax=max(a[1]->rmax,a[1]->sum+val+max(0,a[0]->rmax));
    MAX=max(a[0]->MAX,a[1]->MAX);
    MAX=max(MAX,max(0,a[0]->rmax+a[1]->lmax)+val);
    MAX=max(MAX,max(a[0]->rmax,a[1]->lmax)+val);
  }
}tree[1000005],nulldata,*null=&nulldata,*stack[2000000];
int num[5000005],tot,top;
struct Splay
{
  node *root;
  node *newnode(int k)
  {
    node *t;
    if(top==0) t=&tree[++tot];
```

```
50      else t=stack[top--];
51      t->val=t->sum=k;
52      t->rev=t->sameflag=0;
53      t->lmax=t->rmax=t->MAX=k;
54      t->fa=t->a[0]=t->a[1]=null;
55      t->size=1;
56      return t;
57    }
58    node *maketree(int l,int r,node *fa)
59    {
60      if(l>r) return null;
61      int mid=(l+r)>>1;
62      node *t=newnode(num[mid]);
63      t->a[0]=maketree(l,mid-1,t);
64      t->a[1]=maketree(mid+1,r,t);
65      t->fa=fa;
66      t->update();
67      return t;
68    }
69    void rotate(node *t)
70    {
71      node *fa=t->fa;int p=t->dir();
72      fa->pushdown(),t->pushdown();
73      fa->fa->set(t,fa->dir());
74      fa->set(t->a[!p],p);
75      t->set(fa,!p);
76      fa->update();
77      if(fa==root) root=t;
78    }
79    void splay(node *t,node *fa=null)
80    {
81      if(t==null) return;
82      t->pushdown();
83      while(t->fa!=fa)
84      {
85        if(t->fa->fa==fa) rotate(t);
86        else t->dir()==t->fa->dir()?(rotate(t->fa),rotate(t)):(rotate
              (t),rotate(t));
87      }
88      t->update();
89    }
90    void select(int k,node *fa)
91    {
92      node *now=root;
93      while(1)
94      {
95        now->pushdown();
96        int tmp=now->a[0]->size;
97        if(k==tmp+1) break;
98        if(k<=tmp) now=now->a[0];
99        else k-=tmp+1,now=now->a[1];
```

```
100     }
101     splay(now,fa);
102   }
103 }*splay;
104 void recycle(node *t)
105 {
106   if(t==null) return;
107   stack[++top]=t;
108   recycle(t->a[0]),recycle(t->a[1]);
109 }
110 int main()
111 {
112   int n,m,a,b,c;char op[100]; splay=new Splay; int t=0;  null->MAX=
        null->val=null->lmax=null->rmax=-914990825;
113   scanf("%d␣%d",&n,&m);  num[1]=-914990825;num[n+2]=-914990825;
114   for(int i=1;i<=n;i++) scanf("%d",&num[i+1]);
115   splay->root=splay->maketree(1,n+2,null);
116   for(int i=1;i<=m;i++)
117   {
118     scanf("%s",op);
119     if(op[0]=='I')
120     {
121       scanf("%d␣%d",&a,&b);
122       for(int j=1;j<=b;j++) scanf("%d",&num[j]);
123       splay->select(id(a),null);
124       splay->select(id(a+1),splay->root);
125       splay->root->a[1]->a[0]=splay->maketree(1,b,splay->root->a
            [1]);
126       splay->splay(splay->root->a[1]->a[0]);
127     }
128     else if(op[0]=='D')
129     {
130       scanf("%d␣%d",&a,&b);
131       splay->select(id(a-1),null);
132       splay->select(id(a+b),splay->root);
133       recycle(splay->root->a[1]->a[0]);
134       splay->root->a[1]->a[0]=null;
135       splay->splay(splay->root->a[1]);
136     }
137     else if(op[2]=='K')
138     {
139       scanf("%d␣%d␣%d",&a,&b,&c);
140       splay->select(id(a-1),null);
141       splay->select(id(a+b),splay->root);
142       splay->root->a[1]->a[0]->sameflag=1;
143       splay->root->a[1]->a[0]->val=c;
144       splay->splay(splay->root->a[1]->a[0]);
145     }
146     else if(op[0]=='R')
147     {
148       scanf("%d␣%d",&a,&b);
```

```
149       splay->select(id(a-1),null);
150       splay->select(id(a+b),splay->root);
151       splay->root->a[1]->a[0]->revˆ=1;
152       splay->splay(splay->root->a[1]->a[0]);
153     }
154     else if(op[0]=='G')
155     {
156       scanf("%d␣%d",&a,&b);
157       splay->select(id(a-1),null);
158       splay->select(id(a+b),splay->root);
159       printf("%d\n",splay->root->a[1]->a[0]->sum);
160       splay->splay(splay->root->a[1]->a[0]);
161     }
162     else printf("%d\n",splay->root->MAX);
163   }
164 //  system("pause");
165   return 0;
166 }
```

3.3 第K值

```
1  struct SegT
2  {
3      int l,r,lc,rc,sum;
4  }tree[233333*9];
5  int val[233333],s[233333],tot,ROOT[233333];
6  int maketree(int l,int r)
7  {
8      int k=++tot;
9      tree[tot].l=l,tree[tot].r=r;
10     if(l==r)
11         return k;
12     int mid=(l+r)>>1;
13     tree[k].lc=maketree(l,mid);
14     tree[k].rc=maketree(mid+1,r);
15     return k;
16 }
17 int change(int pre,int pos)
18 {
19     int k=++tot;
20     tree[k]=tree[pre];
21     tree[k].sum++;
22     if(tree[k].l==tree[k].r)
23         return k;
24     int mid=(tree[k].l+tree[k].r)>>1;
25     if(pos<=mid) tree[k].lc=change(tree[pre].lc,pos),tree[k].rc=
           tree[pre].rc;
26     else tree[k].rc=change(tree[pre].rc,pos),tree[k].lc=tree[pre].
           lc;
27     return k;
28 }
29 int find(int now,int pre,int k)
```

```
{
    if(tree[now].l==tree[now].r)
        return tree[now].l;
    int tmp=tree[tree[now].lc].sum-tree[tree[pre].lc].sum;
    if(tmp>=k)
        return find(tree[now].lc,tree[pre].lc,k);
    return find(tree[now].rc,tree[pre].rc,k-tmp);
}
int main() {
    int n,m;
    scanf("%d %d",&n,&m);
    for(int i=1;i<=n;i++)
        scanf("%d",&val[i]);
    memcpy(s,val,sizeof(val));
    sort(s+1,s+1+n);
    ROOT[0]=maketree(1,n);
    for(int i=1;i<=n;i++)
        ROOT[i]=change(ROOT[i-1],lower_bound(s+1,s+1+n,val[i])-s);
    for(int i=1;i<=m;i++)
    {
        int a,b,c;
        scanf("%d %d %d",&a,&b,&c);
        printf("%d\n",s[find(ROOT[b],ROOT[a-1],c)]);
    }
    return 0;
}
```

3.4 树上第K值

```
struct SegT
{
    int l,r,sum,lc,rc;
}tree[2333333];
int last[233333],pre[233333],son[233333],tot,MAX;
int n,m,f[233333][21];
int root[233333],dist[233333],head,tail,que[233333],val[233333],s
   [233333];
void add(int a,int b)
{
    tot++;
    son[tot]=b;
    pre[tot]=last[a];
    last[a]=tot;
}
void bfs()
{
    dist[1]=1;
    que[tail++]=1;
    while(head<tail)
    {
        int p=que[head++];
        for(int x=last[p];x!=0;x=pre[x])
```

```
23          {
24              if(f[p][0]!=son[x])
25              {
26                  dist[son[x]]=dist[p]+1;
27                  f[son[x]][0]=p;
28                  que[tail++]=son[x];
29                  for(int j=1;j<18;j++)
30                      f[son[x]][j]=f[f[son[x]][j-1]][j-1];
31              }
32          }
33      }
34  }
35  int lca(int a,int b)
36  {
37      if(dist[a]>dist[b]) swap(a,b);
38      for(int i=17;i>=0;i--)
39      {
40          if(dist[f[b][i]]>=dist[a]) b=f[b][i];
41          if(a==b) return a;
42      }
43      for(int i=17;i>=0;i--)
44          if(f[a][i]!=f[b][i]) a=f[a][i],b=f[b][i];
45      return f[a][0];
46  }
47
48  int maketree(int l,int r)
49  {
50      int k=++tot;
51      tree[k].l=l,tree[k].r=r;
52      if(l==r)
53          return k;
54      int mid=(l+r)>>1;
55      tree[k].lc=maketree(l,mid);
56      tree[k].rc=maketree(mid+1,r);
57      return k;
58  }
59  int change(int pr,int pos)
60  {
61      int k=++tot;
62      tree[k]=tree[pr];
63      tree[k].sum++;
64      if(tree[k].l==tree[k].r)
65          return k;
66      int mid=(tree[k].l+tree[k].r)>>1;
67      if(pos<=mid)
68      {
69          tree[k].lc=change(tree[pr].lc,pos);
70          tree[k].rc=tree[pr].rc;
71      }
72      else
73      {
```

```
74          tree[k].lc=tree[pr].lc;
75          tree[k].rc=change(tree[pr].rc,pos);
76      }
77      return k;
78  }
79  int find(int a,int b,int p,int pf,int k)
80  {
81      int l=1,r=MAX;
82      while(l!=r)
83      {
84          int mid=(l+r)>>1;
85          int sum=tree[tree[a].lc].sum+tree[tree[b].lc].sum-tree[tree
                [p].lc].sum-tree[tree[pf].lc].sum;
86          if(sum>=k) a=tree[a].lc,b=tree[b].lc,p=tree[p].lc,pf=tree[
                pf].lc,r=mid;
87          else a=tree[a].rc,b=tree[b].rc,p=tree[p].rc,pf=tree[pf].rc,
                k-=sum,l=mid+1;
88      }
89      return l;
90  }
91  int main() {
92      scanf("%d %d",&n,&m);
93      for(int i=1;i<=n;i++)
94          scanf("%d",&val[i]),s[i]=val[i];
95      sort(s+1,s+1+n);
96      MAX=unique(s+1,s+1+n)-s-1;
97      for(int i=1;i<=n;i++)
98          val[i]=lower_bound(s+1,s+1+MAX,val[i])-s;
99      for(int i=1;i<=n-1;i++)
100     {
101         int a,b;
102         scanf("%d %d",&a,&b);
103         add(a,b);
104         add(b,a);
105     }
106     bfs();
107     tot=0;
108     root[0]=maketree(1,MAX);
109     for(int i=0;i<tail;i++)
110         root[que[i]]=change(root[f[que[i]][0]],val[que[i]]);
111     for(int i=1;i<=m;i++)
112     {
113         int a,b,c;
114         scanf("%d %d %d",&a,&b,&c);
115         int p=lca(a,b);
116         int pf=f[p][0];
117         printf("%d\n",s[find(root[a],root[b],root[p],root[pf],c)]);
118     }
119     return 0;
120 }
```

3.5 第K值-带修改

```
struct SegT
{
    int lc,rc,sum;
}tree[12633333];

int val[23333*3],tot,root[23333*3],tl[233],tr[233],n,m,MAX;
char op[13333][3];
int a[13333],b[13333],c[13333],s[23333*3];
int maketree(int l,int r)
{
    int k=++tot,mid=(l+r)>>1;
    if(l==r)
        return k;
    tree[k].lc=maketree(l,mid);
    tree[k].rc=maketree(mid+1,r);
    return k;
}
int change(int pre,int l,int r,int pos,int v)
{
    int k=++tot;
    tree[k].sum=tree[pre].sum+v;
    if(l==r)
        return k;
    int mid=(l+r)>>1;
    if(pos<=mid)
    {
        tree[k].lc=change(tree[pre].lc,l,mid,pos,v);
        tree[k].rc=tree[pre].rc;
    }
    else
    {
        tree[k].lc=tree[pre].lc;
        tree[k].rc=change(tree[pre].rc,mid+1,r,pos,v);
    }
    return k;
}
int get(int l,int r,int k)
{
    tl[0]=tr[0]=0;
    for(int x=l;x;x-=(x&(-x)))
        tl[++tl[0]]=root[x];
    for(int x=r;x;x-=(x&(-x)))
        tr[++tr[0]]=root[x];
    int ans1=1,ans2=MAX;
    for(int s1=0,s2=0;ans1!=ans2;s1=0,s2=0)
    {
        int mid=(ans1+ans2)>>1;
        for(int x=1;x<=tl[0];x++)
            s1+=tree[tree[tl[x]].lc].sum;
```

```
50          for(int x=1;x<=tr[0];x++)
51              s2+=tree[tree[tr[x]].lc].sum;
52          if(s2-s1>=k)
53          {
54              for(int x=1;x<=tl[0];x++)
55                  tl[x]=tree[tl[x]].lc;
56              for(int x=1;x<=tr[0];x++)
57                  tr[x]=tree[tr[x]].lc;
58              ans2=mid;
59          }
60          else
61          {
62              for(int x=1;x<=tl[0];x++)
63                  tl[x]=tree[tl[x]].rc;
64              for(int x=1;x<=tr[0];x++)
65                  tr[x]=tree[tr[x]].rc;
66              ans1=mid+1;
67              k-=s2-s1;
68          }
69      }
70      return ans1;
71  }
72  int main()
73  {
74      scanf("%d␣%d",&n,&m);
75      tot=0,MAX=n;
76      for(int i=1;i<=n;i++)
77          scanf("%d",&s[i]),val[i]=s[i];
78      for(int i=1;i<=m;i++)
79      {
80          scanf("%s",op[i]);
81          if(op[i][0]=='C') scanf("%d␣%d",&a[i],&b[i]),s[++MAX]=b[i];
82          else scanf("%d␣%d␣%d",&a[i],&b[i],&c[i]);
83      }
84      sort(s+1,s+1+MAX);
85      MAX=unique(s+1,s+1+MAX)-s-1;
86      root[1]=maketree(1,MAX);
87      for(int i=2;i<=n+1;i++)
88          root[i]=root[1];
89      for(int i=2;i<=n+1;i++)
90      {
91          for(int x=i;x<=n+1;x+=(x&(-x)))
92              root[x]=change(root[x],1,MAX,lower_bound(s+1,s+1+MAX,
                    val[i-1])-s,1);
93      }
94      for(int i=1;i<=m;i++)
95      {
96          if(op[i][0]=='Q') printf("%d\n",s[get(a[i],b[i]+1,c[i])]);
97          else
98          {
99              for(int x=a[i]+1;x<=n+1;x+=(x&(-x)))
```

```
100                 root[x]=change(root[x],1,MAX,lower_bound(s+1,s+1+
                        MAX,val[a[i]])-s,-1);
101             val[a[i]]=b[i];
102             for(int x=a[i]+1;x<=n+1;x+=(x&(-x)))
103                 root[x]=change(root[x],1,MAX,lower_bound(s+1,s+1+
                        MAX,b[i])-s,1);
104         }
105     }
106     return 0;
107 }
```

3.6 LCA-最近公共祖先

```
1  void bfs()
2  {
3      dep[1]=1;
4      que[tail++]=1;
5      while(head<tail)
6      {
7          int p=que[head++];
8          for(int x=last[p];x!=0;x=pre[x])
9          {
10             if(f[p][0]!=son[x])
11             {
12                 dep[son[x]]=dep[p]+1;
13                 f[son[x]][0]=p;
14                 que[tail++]=son[x];
15                 for(int j=1;j<=20;j++)
16                     f[son[x]][j]=f[f[son[x]][j-1]][j-1];
17             }
18         }
19     }
20 }
21 int lca(int a,int b)
22 {
23     if(dep[a]>dep[b]) swap(a,b);
24     for(int i=20;i>=0;i--)
25     {
26         if(dep[f[b][i]]>=dep[a]) b=f[b][i];
27         if(a==b) return a;
28     }
29     for(int i=20;i>=0;i--)
30         if(f[a][i]!=f[b][i]) a=f[a][i],b=f[b][i];
31     return f[a][0];
32 }
```

3.7 线段树: 扫描线 & 哈希(例子是面积合并)

```
1  #define lson l, m, rt << 1
2  #define rson m + 1, r, rt << 1 | 1
3
4  struct Seg {
```

```
5       double h, l, r;
6       int s;
7       Seg() {}
8       Seg(double a,double b,double c,int d) : l(a) , r(b) , h(c) , s(
            d) {}
9     bool operator < (const Seg &cmp) const {
10      return h < cmp.h;
11    }
12  } ss[maxn];
13
14  int col[maxn << 2];
15  double sum[maxn << 2], sum2[maxn << 2];
16  double X[maxn];
17
18  void PushUp(int rt, int l, int r) {
19      if(col[rt])
20          sum[rt] = X[r + 1] - X[l];
21      else if(l == r)
22          sum[rt] = 0;
23      else
24          sum[rt] = sum[rt << 1] + sum[rt << 1 | 1];
25  }
26
27  void update(int L, int R, int c, int l, int r, int rt) {
28      if(L <= l && r <= R) {
29          col[rt] += c;
30          PushUp(rt, l, r);
31          return;
32      }
33      int m = (l + r) >> 1;
34      if(L <= m)  update(L, R, c, lson);
35      if(m < R)   update(L, R, c, rson);
36      PushUp(rt, l, r);
37  }
38
39  int Bin(double key, int n, double X[]) {
40      int l = 0, r = n - 1;
41      while(l <= r) {
42          int m = (l + r) >> 1;
43          if(X[m] == key)
44              return m;
45          if(X[m] < key)
46              l = m + 1;
47          else
48              r = m - 1;
49      }
50      return -1;
51  }
52
53  int main() {
54      int n, caseno = 1;
```

```
    while(scanf("%d", &n) != EOF && n) {
        int m = 0;
        while(n --) {
            double a, b, c, d;
            scanf("%lf%lf%lf%lf", &a, &b, &c, &d);
            X[m] = a;
            ss[m ++] = Seg(a, c, b, 1);
            X[m] = c;
            ss[m ++] = Seg(a, c, d, -1);
        }
        sort(X, X + m);
        sort(ss, ss + m);
        int k = 1;
        for(int i = 1; i < m; i ++) {
            if(X[i] != X[i -1])
                X[k ++] = X[i];
        }
        memset(col, 0, sizeof(col));
        memset(sum, 0, sizeof(sum));
        double ret = 0;
        for(int i = 0; i < m - 1; i ++) {
            int l = Bin(ss[i].l, k, X);
            int r = Bin(ss[i].r, k, X) - 1;
            if(l <= r)
                update(l, r, ss[i].s, 0, k - 1, 1);
            ret += sum[1] * (ss[i + 1].h - ss[i].h);
        }
        printf("Test␣case␣#%d\nTotal␣explored␣area:␣%.2lf\n\n",
           caseno ++, ret);
    }
    return 0;
}
```

3.8 线段树: 区间更新

线段树: 区间更新，包含区间查询和单点查询。

```
int li[maxn], ri[maxn];
int rec[maxn << 3];
int sum[maxn << 4];
int col[maxn << 4];
int hash[maxn];

void PushUP(int rt) {
    sum[rt] = sum[rt << 1] + sum[rt << 1 | 1];
}

void PushDown(int rt, int m) {
    if(col[rt]) {
         col[rt << 1] = col[rt << 1 | 1] = col[rt];
         sum[rt << 1] = (m - (m >> 1)) * col[rt];
         sum[rt << 1 | 1] = (m >> 1) * col[rt];
         col[rt] = 0;
```

```
17      }
18  }
19
20  void update(int L, int R, int c, int l, int r, int rt) {
21      if(L <= l && r <= R) {
22          sum[rt] = c * (r - l + 1);
23          col[rt] = c;
24          return;
25      }
26      PushDown(rt, r - l + 1);
27      int m = (l + r) >> 1;
28      if(L <= m)  update(L, R, c, lson);
29      if(R > m)   update(L, R, c, rson);
30      sum[rt] = 0;
31  }
32
33  void build(int l, int r, int rt) {
34      col[rt] = 0;
35      if(l == r) {
36          sum[rt] = 0;
37          return;
38      }
39      int m = (l + r) >> 1;
40      build(lson);
41      build(rson);
42      PushUP(rt);
43  }
44
45  //单点查询
46  void query(int l, int r, int rt) {
47      if(sum[rt] > 0) {
48          LL pos = sum[rt] / (LL)(r - l + 1);
49          hash[pos] = 1;
50          return;
51      }
52      if(l == r)  return;
53      int m = (l + r) >> 1;
54      query(lson);
55      query(rson);
56      return;
57  }
58
59  //区间查询
60  int query(int L, int R, int l, int r, int rt) {
61      if(L <= l && r <= R) {
62          return sum[rt];
63      }
64      int m = (l + r) >> 1;
65      int ret = 0;
66      if(L <= m)  ret += query(L, R, lson);
67      if(R > m)   ret += query(L, R, rson);
```

```
68      return ret;
69  }
70
71  int scan(int key, int n, int x[]) {
72      int l = 0, r = n - 1;
73      while(l <= r) {
74          int m = (l + r) >> 1;
75          if(x[m] == key) return m;
76          if(x[m] < key)
77              l = m + 1;
78          else
79              r = m - 1;
80      }
81      return -1;
82  }
83
84  int main() {
85      scanf("%d", &t);
86      while(t --) {
87          memset(hash, 0, sizeof(hash));
88          scanf("%d", &n);
89          int tt = 0;
90          for(int i = 0; i < n; i ++) {
91              scanf("%d%d", &li[i], &ri[i]);
92              rec[tt ++] = li[i];
93              rec[tt ++] = ri[i];
94          }
95          sort(rec, rec + 2 * n);
96          int m = 1;
97          for(int i = 1; i < tt; i ++) {
98              if(rec[i] != rec[i - 1])
99                  rec[m ++] = rec[i];
100         }
101         tt = m;
102         for(int i = 1; i < tt; i ++) {
103             if(rec[i] - rec[i - 1] > 1) {
104                 rec[m ++] = rec[i] + 1;
105             }
106         }
107         sort(rec, rec + m);
108         build(0, m, 1);
109         for(int i = 0; i < n; i ++) {
110             int l = scan(li[i], m, rec);
111             int r = scan(ri[i], m, rec);
112             update(l, r, i + 1, 0, m, 1);
113         }
114         query(0, m, 1);
115         int res = 0;
116         for(int i = 1; i <= n; i ++) {
117             if(hash[i] == 1) {
118                 res ++;
```

```
119             }
120         }
121         printf("%d\n", res);
122     }
123 }
```

3.9 Splay Tree

```
1  struct node
2  {
3    int cnt,val,size;
4    node *fa,*a[2];
5    void set(node *t,int p){a[p]=t;t->fa=this;}
6    int dir(){return fa->a[1]==this;}
7    void update(){size=a[0]->size+a[1]->size+cnt;}
8  }tree[100000],nulldata,*null=&nulldata;
9  int tot;
10 struct Splay
11 {
12   node *root;
13   node *newnode(int k)
14   {
15     node *t=&tree[++tot];
16     t->size=t->cnt=1;
17     t->fa=t->a[0]=t->a[1]=null;
18     t->val=k;
19     return t;
20   }
21   void rotate(node *t)
22   {
23     node *fa=t->fa;int p=t->dir();
24     fa->fa->set(t,fa->dir());
25     fa->set(t->a[!p],p);
26     t->set(fa,!p);
27     fa->update();
28   }
29   void splay(node *t,node *fa=null)
30   {
31     while(t->fa!=fa)
32     {
33       if(t->fa->fa==null) rotate(t);
34       else t->dir()==t->fa->dir()?(rotate(t->fa),rotate(t)):(rotate
           (t),rotate(t));
35     }
36     if(fa==null) root=t;
37     t->update();
38   }
39   void insert(int k)
40   {
41     if(root==null) {root=newnode(k);return;}
42     node *t=root,*fa=null;
```

```
43      while(t!=null)
44      {
45        fa=t;
46        if(k<t->val) t=t->a[0];
47        else if(k>t->val) t=t->a[1];
48        else break;
49      }
50      if(t!=null) t->cnt++,splay(t);
51      else fa->set(t=newnode(k),k>fa->val),splay(t);
52    }
53    node *findnode(int k)
54    {
55      node *t=root;
56      while(t!=null)
57      {
58        if(k<t->val) t=t->a[0];
59        else if(k==t->val) break;
60        else t=t->a[1];
61      }
62      return t;
63    }
64
65    int find(int k)
66    {
67      node *t=findnode(k);
68      if(t==null) return -1;
69      return t->cnt;
70    }
71    int countmin(int k)
72    {
73      node *t=root; int ans=0;
74      while(t!=null)
75      {
76        if(k>t->val) ans+=t->a[0]->size+t->cnt,t=t->a[1];
77        else if(k==t->val) {ans+=t->a[0]->size;break;}
78        else t=t->a[0];
79      }
80      return ans;
81    }
82    node *findmaxnode(node *t){while(t->a[1]!=null) t=t->a[1];return
        t;}
83    void erase(int k)
84    {
85      node *t=findnode(k);
86      if(t==null) return ;
87      if(t->cnt>1) t->cnt--,splay(t);
88      else
89      {
90        splay(t);
91        if(t->a[0]==null) {root=t->a[1];root->fa=null;return ;}
92        if(t->a[1]==null) {root=t->a[0];root->fa=null;return ;}
```

```
93        node *x=findmaxnode(t->a[0]); splay(x,t);
94        x->a[1]=t->a[1]; t->a[1]->fa=x; root=x; x->fa=null; root->
            update();
95      }
96    }
97  };
98  Splay *splay;
99  int main()
100 {
101 //  freopen("a.in","r",stdin); freopen("a.out","w",stdout);
102   splay=new Splay; splay->root=null;
103   int n,m,op,x;
104   scanf("%d %d",&n,&m);
105   for(int i=1;i<=n;i++)
106     scanf("%d",&x),splay->insert(x);
107   for(int i=1;i<=m;i++)
108   {
109     scanf("%d",&op);
110     if(op==1) scanf("%d",&x),splay->insert(x);
111     else if(op==2) scanf("%d",&x),printf("%d\n",splay->find(x));
112     else if(op==3) scanf("%d",&x),printf("%d\n",splay->countmin(x))
          ;
113     else if(op==4) scanf("%d",&x),splay->erase(x);
114   }
115 //  system("pause");
116   return 0;
117 }
```

3.10 树套树

```
1   struct node
2   {
3     int val,size,father,a[2];
4   }tree[5000000];
5   int tot,n,Null,src[10005],pree[10005],NONE=-INT_MAX;
6   int newnode(int k)
7   {
8     tot++;
9     tree[tot].father=tree[tot].a[1]=tree[tot].a[0]=Null;
10    tree[tot].val=k;tree[tot].size=1;
11    return tot;
12  }
13  inline void update(const int& root)
14  {
15      tree[root].size=tree[tree[root].a[0]].size+tree[tree[root].a
          [1]].size+1;
16  }
17  struct splaytree
18  {
19    int head;
20    splaytree(){head=Null;}
21    void rotate(const int& root,const int& p)
```

```
22  {
23    int k=tree[root].father;
24    tree[k].a[!p]=tree[root].a[p];tree[root].father=tree[k].father;
25    if(tree[root].a[p]!=Null) tree[tree[root].a[p]].father=k;
26    if(tree[k].father!=Null) tree[tree[k].father].a[tree[tree[k].
        father].a[1]==k]=root;
27    tree[k].father=root;
28    tree[root].a[p]=k;
29    update(k);
30  }
31  void splay(const int& root,const int& f)
32  {
33    while(tree[root].father!=f)
34    {
35      int k=tree[root].father,p1,p2;
36      if(tree[k].a[0]==root) p1=0;else p1=1;
37      if(tree[tree[k].father].a[0]==k) p2=0;else p2=1;
38      if(tree[k].father==f) rotate(root,1-p1);
39      else if(p1==p2) rotate(k,1-p2),rotate(root,1-p1);
40      else rotate(root,1-p1),rotate(root,1-p2);
41    }
42    update(root);
43    if(f==Null) head=root;
44  }
45  void insert(int k)
46  {
47    int now=head;
48    if(head==0) head=newnode(k),tree[tot].father=Null;
49    else
50    {
51      while(1)
52      {
53        if(tree[now].a[k>tree[now].val]!=Null) now=tree[now].a[k>
            tree[now].val];
54        else break;
55      }
56      tree[now].a[k>tree[now].val]=newnode(k);
57      tree[tot].father=now;
58      splay(tot,Null);
59    }
60  }
61  int countmin(int k)
62  {
63    int now=head,ans=0;
64    while(now!=Null)
65    {
66      if(tree[now].val==k) ans+=tree[tree[now].a[0]].size,now=tree[
          now].a[1];
67      else if(tree[now].val<k) ans+=tree[tree[now].a[0]].size+1,now
          =tree[now].a[1];
68      else now=tree[now].a[0];
```

```
69      }
70      return ans;
71    }
72    int find(int k)
73    {
74      int now=head;
75      while(1)
76      {
77        if(tree[now].val==k) return now;
78        if(tree[now].val>k) now=tree[now].a[0];
79        else now=tree[now].a[1];
80      }
81      return Null;
82    }
83    int findpre(int now,int k)
84    {
85      if(now==Null) return NONE;
86      if(tree[now].val<k)
87      {
88        int p=findpre(tree[now].a[1],k);
89        if(p!=NONE) return p;
90        return tree[now].val;
91      }
92      return findpre(tree[now].a[0],k);
93    }
94    int findnext(int now,int k)
95    {
96      if(now==Null) return NONE;
97      if(tree[now].val>k)
98      {
99        int p=findnext(tree[now].a[0],k);
100       if(p!=NONE) return p;
101       return tree[now].val;
102     }
103     return findnext(tree[now].a[1],k);
104   }
105   int findpre(int k){return findpre(head,k);}
106   int findnext(int k){return findnext(head,k);}
107   int findmax(int root){while(tree[root].a[1]!=Null) root=tree[root
        ].a[1];return root;}
108   void del(int k)
109   {
110     int now=find(k);splay(now,Null);
111     if(tree[now].a[0]==Null) {head=tree[now].a[1],tree[head].father
          =Null;return;}
112     if(tree[now].a[1]==Null) {head=tree[now].a[0],tree[head].father
          =Null;return;}
113     int x=findmax(tree[now].a[0]);splay(x,now);
114     tree[x].a[1]=tree[now].a[1];tree[tree[now].a[1]].father=x;
115     head=x;update(head);tree[head].father=Null;
116   }
```

```
}c[10005],color[1000005];
int findsum(int x,int limit)
{
  int ans=0;
  for(;x>=1;x-=(x&(-x))) ans+=c[x].countmin(limit);
  return ans;
}
void del(int x,int k)
{
  for(;x<=n;x+=(x&(-x))) c[x].del(k);
}
void insert(int x,int k)
{
  for(;x<=n;x+=(x&(-x))) c[x].insert(k);
}
void makechange(int now,int next)
{
  del(now,pree[now]);
  pree[now]=next;
  insert(now,pree[now]);
}
```

3.11 二维树状数组

```
const int N = 2100, M = 2100;
int n,m;
struct BIT
{
    int dt[N][M];
    void modify (int x, int y, lint delt)
  {
        for (int i = x; i <= n; i += i&(-i))
            for (int j = y; j <= m; j += j & (-j))
                dt[i][j] += delt;
    }
    lint sum (int x,int y)
  {
        lint res = 0;
        for (int i = x; i; i -= i & (-i))
            for (int j = y; j; j -= j & (-j))
                res += dt[i][j];
        return res;
    }
}A,B,C,D;

void modify (int x, int y, lint delt)
{
    A.modify(x, y, delt);
    B.modify(x, y, x * delt);
    C.modify(x, y, y * delt);
    D.modify(x, y, x * y * delt);
}
```

```
29 lint sum (int x, int y)
30 {
31     return A.sum(x, y) * (x + 1) * (y + 1) - (y + 1) * B.sum(x, y)
          - (x + 1) * C.sum(x, y)+D.sum(x, y);
32 }
33 void modify (int x1, int y1, int x2, int y2, lint delt)
34 {
35     modify(x1, y1, delt); modify(x2 + 1, y2 + 1, delt);
36     modify(x2 + 1, y1, -delt); modify(x1, y2 + 1, -delt);
37 }
38 lint sum (int x1, int y1, int x2, int y2)
39 {
40     return sum(x2, y2) - sum(x2, y1 - 1) - sum(x1 - 1, y2) + sum(x1
          - 1, y1 - 1);
41 }
42 int main ()
43 {
44     char op;getchar();
45     scanf("%d␣%d", &n, &m);
46     while (1)
47     {
48         do op=getchar(); while (op!='L' && op!='k' && op!=-1);
49         if (op==-1) break;
50         if (op=='L')
51         {
52             int a,b,c,d,y;
53             scanf("%d␣%d␣%d␣%d␣%d\n",&a,&b,&c,&d,&y);
54             if(a>c) swap(a,c);
55             if(b>d) swap(b,d);
56             modify(a,b,c,d,y);
57         }
58         else
59         {
60             int a, b, c, d;
61             scanf("%d␣%d␣%d␣%d",&a,&b,&c,&d);
62             if (a>c) swap(a,c);
63             if (b>d) swap(b,d);
64             printf("%d\n", sum(a,b,c,d));
65         }
66     }
67     return 0;
68 }
```

4 计算几何

4.1 2D计算几何代码库

```
struct pnt {
    double x, y;
    pnt(){}
    pnt(double _x, double _y) : x(_x), y(_y) {}
}p[110];
struct vct{double x, y;};
struct seg{pnt s, t; double a, b, c;};
const pnt o = pnt(0.0, 0.0);

vct operator - (pnt t, pnt s) {
    vct ret = {t.x - s.x, t.y - s.y};
    return ret;
}

seg operator ^ (pnt s, pnt t) {
    double a = s.y - t.y;
    double b = t.x - s.x;
    double c = -(a * s.x + b * s.y);
    seg ret = {s, t, a, b, c};
    return ret;
}

double operator * (vct a, vct b) {
    return a.x * b.y - a.y * b.x;
}

double operator ^ (vct a, vct b) {
    return a.x * b.x + a.y * b.y;
}

vct _rotate(vct v, double sn) {
    double cs = sqrt(1.0 - sn * sn);
    vct ret = {v.x * cs - v.y * sn, v.x * sn + v.y * cs};
    return ret;
}

double dis(pnt a, pnt b) {
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.
       y));
}

double dis(pnt a, seg b) {
    double s = (b.t - b.s) * (a - b.s);
    if(((b.t - b.s) ^ (a - b.s)) > 0.0 && ((b.s - b.t) ^ (a - b.t))
        > 0.0)
        return _abs(s / dis(b.s, b.t));
    return min(dis(a, b.s), dis(a, b.t));
```

```
46 }
47
48 double getS(pnt *p, int n) {
49     double ret = 0.0;
50     for(int i = 0; i < n; i ++)
51         ret += (p[i] - o) * (p[(i+1)%n] - o);
52     return ret / 2.0;
53 }
54
55 bool isIntersect(seg a, seg b) {
56     if(max(a.s.x, a.t.x) < min(b.s.x, b.t.x) - eps ||
57        max(a.s.y, a.t.y) < min(b.s.y, b.t.y) - eps ||
58        max(b.s.x, b.t.x) < min(a.s.x, a.t.x) - eps ||
59        max(b.s.y, b.t.y) < min(a.s.y, a.t.y) - eps)
60         return false;
61     if(((a.t - a.s) * (b.t - a.s)) * ((a.t - a.s) * (b.s - a.s)) >
          eps ||
62        ((b.t - b.s) * (a.t - b.s)) * ((b.t - b.s) * (a.s - b.s)) >
            eps)
63         return false;
64     return true;
65 }
66
67 pnt intersect(seg a, seg b) {
68     pnt p1 = a.s;
69     pnt p2 = a.t;
70     pnt p3 = b.s;
71     pnt p4 = b.t;
72     double x = (p3.x - p4.x) * ((p1 - o) * (p2 - o)) - (p1.x - p2.x
          ) * ((p3 - o) * (p4 - o));
73     double y = (p3.y - p4.y) * ((p1 - o) * (p2 - o)) - (p1.y - p2.y
          ) * ((p3 - o) * (p4 - o));
74     double t = (p2 - p1) * (p4 - p3);
75     pnt ret(x / t, y / t);
76     return ret;
77 }
78
79 int cut(seg t, int cnt, pnt *p, pnt *q) {
80     int cur = 0;
81     for(int i = 1; i <= cnt; i ++)
82         if(t.a * p[i].x + t.b * p[i].y + t.c > -eps)
83             q[++cur] = p[i];
84         else {
85             if(t.a * p[i-1].x + t.b * p[i-1].y + t.c > eps)
86                 q[++cur] = intersect(t, p[i-1] ^ p[i]);
87             if(t.a * p[i+1].x + t.b * p[i+1].y + t.c > eps)
88                 q[++cur] = intersect(t, p[i] ^ p[i+1]);
89         }
90     cnt = cur;
91     for(int i = 1; i <= cnt; i ++)
92         p[i] = q[i];
```

```
93      p[0] = p[cnt];
94      p[cnt+1] = p[1];
95      return cnt;
96  }
97
98  void regularize(pnt *p, int n) {
99      if(getS(p, n) > 0)
100         return;
101     for(int i = 0; i < n / 2; i ++)
102         _swap(p[i], p[n-i-1]);
103 }
104
105 bool cmp(pnt a, pnt b) {
106     if(!sgn((a - p[0]) * (b - p[0])))
107         return dis(a, p[0]) < dis(b, p[0]);
108     return (a - p[0]) * (b - p[0]) > 0;
109 }
110
111 int graham(pnt *p, pnt *sk, int n) {
112     for(int i = 1; i < n; i ++)
113         if(sgn(p[0].y - p[i].y) > 0 || (!sgn(p[0].y - p[i].y) && p
               [0].x > p[i].x))
114             _swap(p[0], p[i]);
115     sort(p + 1, p + n, cmp);
116     for(int i = 0; i < 3; i ++)
117         sk[i] = p[i];
118     int tp = 2;
119     for(int i = 3; i < n; i ++) {
120         while((sk[tp] - sk[tp-1]) * (p[i] - sk[tp-1]) < -eps)
121             tp --;
122         sk[++tp] = p[i];
123     }
124     return tp + 1;
125 }
```

4.2 3D 两线段距离

```
1   const int inf = 1e9 + 7;
2   const double eps = 1e-7;
3   inline int sgn(const double x) {return (x > eps) - (x < -eps);}
4   template <class T> T _abs(T x) {if(x < 0) x = -x; return x;}
5   struct tp
6   {
7       double x,y,z;
8   }A,B,C,D;
9   struct Point3D
10  {
11      long double x,y,z;
12  Point3D(long double _x = 0,long double _y = 0,long double _z = 0)
13      {
14          x = _x;
15          y = _y;
```

```
16          z = _z;
17      }
18
19      Point3D operator -(const Point3D &b)const
20      {
21          return Point3D(x-b.x,y-b.y,z-b.z);
22      }
23      Point3D operator ^(const Point3D &b)const
24      {
25          return Point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
26      }
27      long double operator *(const Point3D &b)const
28      {
29          return x*b.x+y*b.y+z*b.z;
30      }
31  }a,b,c,d;
32
33  long double Norm(Point3D p)
34  {
35      return sqrt(p*p);
36  }
37  long double calc(Point3D a,Point3D k1,Point3D b,Point3D k2)
38  {
39      Point3D tmp = k1^k2;
40      return fabs(tmp*(a-b))/sqrt(tmp*tmp);
41  }
42
43  Point3D shift(Point3D x, Point3D y, double d) {
44      long double k = d / sqrt(y.x * y.x + y.y * y.y + y.z * y.z);
45      x.x += k * y.x;
46      x.y += k * y.y;
47      x.z += k * y.z;
48      return x;
49  }
50
51  Point3D intersection(Point3D u1, Point3D u2, Point3D v1, Point3D v2
        ){
52    Point3D ret=u1;
53    long double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
54        /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
55    ret.x+=(u2.x-u1.x)*t;
56    ret.y+=(u2.y-u1.y)*t;
57    ret.z+=(u2.z-u1.z)*t;
58    return ret;
59  }
60
61  long double getl(Point3D v) {
62      return v.x * v.x + v.y * v.y + v.z * v.z;
63  }
64  int main() {
65      int T;
```

```
66      scanf("%d",&T);
67      while(T --) {
68          scanf("%lf%lf%lf%lf%lf%lf",&A.x,&A.y,&A.z,&B.x,&B.y,&B.z);
69          scanf("%lf%lf%lf%lf%lf%lf",&C.x,&C.y,&C.z,&D.x,&D.y,&D.z);
70          a.x=(long double)A.x;
71          a.y=(long double)A.y;
72          a.z=(long double)A.z;
73          b.x=(long double)B.x;
74          b.y=(long double)B.y;
75          b.z=(long double)B.z;
76          c.x=(long double)C.x;
77          c.y=(long double)C.y;
78          c.z=(long double)C.z;
79          d.x=(long double)D.x;
80          d.y=(long double)D.y;
81          d.z=(long double)D.z;
82          long double ans = calc(a, b - a, c, d - c);
83          printf("%.6f\n",(double)ans);
84          Point3D d1 = b - a;
85          Point3D d2 = d - c;
86          Point3D p1 = a;
87          Point3D p2 = d;
88          Point3D tx = (d1 ^ d2);
89          //cout<<"fuck  "<<tx<<endl;
90          long double t1 = ((p2 - p1) ^ d2) * tx / getl(tx);
91          long double t2 = ((p2 - p1) ^ d1) * tx / getl(tx);
92          printf("%.6f␣%.6f␣%.6f␣%.6f␣%.6f␣%.6f\n",double(p1.x + t1 *
                d1.x), double(p1.y + t1 * d1.y), double(p1.z + t1 * d1.
               z), double(p2.x + t2 * d2.x), double(p2.y + t2 * d2.y),
               double(p2.z + t2 * d2.z));
93      }
94      return 0;
95  }
```

4.3 k-d tree第k远点

```
1   const int N=100100;
2   struct P
3   {
4     int x,y,id;
5   }cp[N],Q;
6   struct node
7   {
8     node *c[2];
9     int x,y,minx,miny,maxx,maxy,p;
10  }mem[N*3],*cur=mem,*root;
11  struct ANS
12  {
13    big d;
14    int id;
15  }ans[30];
16  int vis[N],cas,k;
```

```
17 bool cmpx(const P &a,const P &b){return a.x<b.x||(a.x==b.x&&a.y<b.y
   );}
18 bool cmpy(const P &a,const P &b){return a.y<b.y||(a.y==b.y&&a.x<b.x
   );}
19 big dis(int a,int b){return 1ll*a*a+1ll*b*b;}
20 node *build(int l,int r,bool t)
21 {
22   if(l>r)return NULL;
23   sort(cp+l,cp+r+1,t?cmpy:cmpx);
24   int mid=(l+r)>>1;
25   node *p=cur++;
26   p->c[0]=p->c[1]=NULL;
27   p->p=cp[mid].id;
28   p->x=p->minx=p->maxx=cp[mid].x;
29   p->y=p->miny=p->maxy=cp[mid].y;
30   if(l==r)return p;
31   p->c[0]=build(l,mid-1,!t);
32   p->c[1]=build(mid+1,r,!t);
33   for(int i=0;i<2;i++)
34   if(p->c[i])
35   {
36     p->minx=min(p->minx,p->c[i]->minx);
37     p->maxx=max(p->maxx,p->c[i]->maxx);
38     p->miny=min(p->miny,p->c[i]->miny);
39     p->maxy=max(p->maxy,p->c[i]->maxy);
40   }
41   return p;
42 }
43 void update(big d,int id)
44 {
45   if(d<ans[k].d)return ;
46   static int i,j;
47   for(i=1;i<=k;i++)
48   if(ans[i].d<d||(ans[i].d==d&&ans[i].id>id))
49   {
50     for(j=k;j>i;j--)ans[j]=ans[j-1];
51     ans[i].d=d;ans[i].id=id;
52     return ;
53   }
54 }
55 void query(node *tr,int l,int r,bool t)
56 {
57   if(tr==NULL)return ;
58   if(vis[tr->p]!=cas)vis[tr->p]=cas,update(dis(tr->x-Q.x,tr->y-Q.y)
      ,tr->p);
59   if(l==r||dis(max(Q.x-tr->minx,tr->maxx-Q.x),max(Q.y-tr->miny,tr->
      maxy-Q.y))<ans[k].d)return ;
60   int mid=(l+r)>>1;
61   (t?(Q.y<tr->y||(Q.x<tr->x&&Q.y==tr->y)):(Q.x<tr->x||(Q.y<tr->y&&Q
      .x==tr->x)))
62   ?(query(tr->c[1],mid+1,r,!t),query(tr->c[0],l,mid-1,!t)):
```

```
  (query(tr->c[0],l,mid-1,!t),query(tr->c[1],mid+1,r,!t));
}
int main()
{
  int i,n,m;
  cas=1;
  scanf("%d",&n);
  for(i=1;i<=n;i++)
  scanf("%d%d",&cp[i].x,&cp[i].y),cp[i].id=i;
  root=build(1,n,0);
  scanf("%d",&m);
  while(m--)
  {
    cas++;
    scanf("%d%d%d",&Q.x,&Q.y,&k);
    for(i=1;i<=k;i++)
    ans[i].id=n+1,ans[i].d=-1;
    query(root,1,n,0);
    printf("%d\n",ans[k].id);
  }
}
```

4.4 凸多边形面积并

```
const double eps=1e-9;
const int N=150;
int n,m=3;
inline int sgn(double x)
{
    if(fabs(x)<eps)return 0;
    return x>0?1:-1;
}
struct L
{
    double k,b;
    L(){}
    L(double _k,double _b):k(_k),b(_b){}
    bool operator<(const L &a)const
    {
        if(!sgn(k-a.k))return b<a.b;
        return sgn(k-a.k)<0;
    }
    bool operator==(const L &a)const
    {
        return !sgn(k-a.k)&&!sgn(b-a.b);
    }
};
struct P
{
    double x,y;
    inline void input()
    {
```

```
29          scanf("%lf%lf",&x,&y);
30      }
31      P(){}
32      P(double _x,double _y):x(_x),y(_y){}
33      P operator-(const P b)const
34      {
35          return P(x-b.x,y-b.y);
36      }
37      double operator*(const P b)const
38      {
39          return x*b.y-b.x*y;
40      }
41      bool operator<(const P &a)const
42      {
43          if(!sgn(x-a.x))return sgn(y-a.y)<0;
44          return sgn(x-a.x)<0;
45      }
46      bool operator==(const P &a)const
47      {
48          return !sgn(x-a.x)&&!sgn(y-a.y);
49      }
50  };
51  P cp;
52  struct SEG
53  {
54      P a,b;
55      SEG(){}
56      SEG(P _a,P _b):a(_a),b(_b){}
57  };
58  vector<SEG>seg1,seg2;
59  L line[10*N];
60  struct POLY
61  {
62      P p[4];
63      P&operator[](int x){return p[x];}
64      bool check()
65      {
66          p[m]=p[0];
67          double res=0;
68          int i;
69          for(i=0;i<m;i++)res+=p[i]*p[i+1];
70          if(!sgn(res))return false;
71          if(sgn(res)<0)reverse(p,p+m+1);
72          return true;
73      }
74  }tr[N];
75  void init()
76  {
77      int i,j,k=0;
78      scanf("%d",&n);
79      for(i=1;i<=n;i++)
```

```
80      {
81          for(j=0;j<m;j++)
82          tr[i][j].input();
83          if(tr[i].check())tr[++k]=tr[i];
84      }
85      n=k;
86  }
87  L get_L(const P a,const P b)
88  {
89      P c=b-a;
90      double k=c.y/c.x;
91      double bb=b.y-b.x*k;
92      return L(k,bb);
93  }
94  bool get_cut(const L f,const P a,const P b)
95  {
96      double tmp=f.k*(a.x-b.x)-(a.y-b.y);
97      if(!sgn(tmp))return false;
98      cp.x=(a*b-(a.x-b.x)*f.b)/tmp;
99      double sx=min(a.x,b.x),ex=max(a.x,b.x);
100     cp.y=cp.x*f.k+f.b;
101     double sy=min(a.y,b.y),ey=max(a.y,b.y);
102     return (sgn(cp.x-sx)>=0)&&(sgn(cp.x-ex)<=0)&&(sgn(cp.y-sy)>=0)
        &&(sgn(cp.y-ey)<=0);
103 }
104 double cal(const vector<SEG>seg)
105 {
106     int i,k=0,len=seg.size(),j;
107     bool touch;
108     double res=0;
109     for(i=0;i<len;i++)line[++k]=get_L(seg[i].a,seg[i].b);
110     sort(line+1,line+1+k);
111     len=unique(line+1,line+1+k)-line-1;
112     for(i=1;i<=len;i++)
113     {
114         vector<pair<double,int>>mark;
115         for(j=1;j<=n;j++)
116         {
117             vector<P>cut;
118             touch=false;
119             for(k=0;k<m;k++)
120             if(line[i]==get_L(tr[j][k],tr[j][k+1]))
121             {
122                 touch=true;
123                 break;
124             }
125             if(touch)continue;
126             for(k=0;k<m;k++)
127             if(get_cut(line[i],tr[j][k],tr[j][k+1]))
128             cut.pb(cp);
129             sort(cut.begin(),cut.end());
```

```
130             cut.resize(unique(cut.begin(),cut.end())-cut.begin());
131             if(cut.size()==2)
132             {
133                 mark.pb(mp(cut[0].x,0));
134                 mark.pb(mp(cut[1].x,1));
135             }
136         }
137         for(j=0;j<seg.size();j++)
138         if(line[i]==get_L(seg[j].a,seg[j].b))
139         {
140             double s=min(seg[j].a.x,seg[j].b.x);
141             double e=max(seg[j].a.x,seg[j].b.x);
142             mark.pb(mp(s,2));
143             mark.pb(mp(e,3));
144         }
145         sort(mark.begin(),mark.end());
146         int in=0,ct=0;
147         double last=mark[0].first;
148         for(j=0;j<mark.size();j++)
149         {
150             double y0=line[i].k*last+line[i].b;
151             double y1=line[i].k*mark[j].first+line[i].b;
152             if(!in&&ct)res+=(y0+y1)*(mark[j].first-last)/2.0;//,
                    printf("%.1lf %.1lf %.1lf\n",y0,y1,mark[j].first-
                    last);
153             last=mark[j].first;
154             if(mark[j].second==0)in++;
155             if(mark[j].second==1)in--;
156             if(mark[j].second==2)ct++;
157             if(mark[j].second==3)ct--;
158         }
159     }
160     return res;
161 }
162 double sol()
163 {
164     int i,j;
165     P A,B;
166     for(i=1;i<=n;i++)
167     {
168         for(j=0;j<m;j++)
169         {
170             A=tr[i][j];
171             B=tr[i][j+1];
172             if(sgn(A.x-B.x)>0)seg1.pb(SEG(A,B));
173             if(sgn(A.x-B.x)<0)seg2.pb(SEG(A,B));
174         }
175     }
176     return cal(seg1)-cal(seg2);
177 }
178 int main()
```

```
{
    init();
    printf("%.2lf\n",sol());
}
```

4.5 最小圆覆盖

```
const double eps=1e-8;
const int N=100010;
struct P
{
    double x,y;
};
P point[N],cir;
double R;
int n;
double dis(P a,P b)
{
    return(a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}
inline void makecir(P a,P b)
{
    cir.x=(a.x+b.x)/2.0;
    cir.y=(a.y+b.y)/2.0;
    R=dis(cir,b);
}
int main()
{
    int i,j,k;
    double a1,b1,c1,a2,b2,c2,s;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    scanf("%lf%lf",&point[i].x,&point[i].y);
    random_shuffle(point+1,point+1+n);
    makecir(point[1],point[2]);
    for(i=3;i<=n;i++)
    if(dis(cir,point[i])>R-eps)
    {
        makecir(point[1],point[i]);
        for(j=2;j<i;j++)
        if(dis(cir,point[j])>R-eps)
        {
            makecir(point[i],point[j]);
            for(k=1;k<j;k++)
            if(dis(cir,point[k])>R-eps)
            {
                a1=2*(point[j].x-point[i].x),b1=2*(point[j].y-point
                   [i].y),
                c1=-(point[j].x*point[j].x-point[i].x*point[i].x+
                   point[j].y*point[j].y-point[i].y*point[i].y);
                a2=2*(point[k].x-point[i].x),b2=2*(point[k].y-point
                   [i].y),
```

```
                c2=-(point[k].x*point[k].x-point[i].x*point[i].x+
                   point[k].y*point[k].y-point[i].y*point[i].y);
                s=a1*b2-a2*b1;
                if(fabs(s)>eps)
                {
                    cir.x=(b1*c2-b2*c1)/s;
                    cir.y=(a2*c1-a1*c2)/s;
                    R=dis(cir,point[i]);
                }
            }
        }
    }
    printf("%.10lf\n%.10lf␣%.10lf",sqrt(R),cir.x,cir.y);
}
```

4.6 Simpson圆并

```
const double eps=1e-13;
struct circle
{
  double lx,rx,r,r2,x,y;
}cir[1050],c[1050];
int n,del[1050];
pair<double,double>tmp[1099];
double f(int l,int r,double k)
{
  double ans=0.0;int num=0,j;
  for(int i=l;i<=r;i++)
  {
    if(k<cir[i].lx || k>cir[i].rx) continue;
    double len=sqrt(cir[i].r2-(cir[i].x-k)*(cir[i].x-k));
    tmp[++num].first=cir[i].y-len;
    tmp[num].second=cir[i].y+len;
  }
  sort(tmp+1,tmp+1+num);
  for(int i=1;i<=num;)
  {
    double s=tmp[i].first,e=tmp[i].second;
    for(j=i+1;j<=num && e>=tmp[j].first;e=max(e,tmp[j].second),j++)
       ;
    j--;
    ans+=e-s;i=j+1;
  }
  return ans;
}
double simpson(double fl,double fr,double fm,double l,double r)
{
    return (fm*4.0+fl+fr)*(r-l)/6.0;
}
double rsimpson(double l,double r,double mid,double fl,double fr,
   double fm,double s,int L,int R)
{
```

```
34    double ml=(l+mid)/2.0,mr=(mid+r)/2.0,fml=f(L,R,ml),fmr=f(L,R,mr),
        s1=simpson(fl,fm,fml,l,mid),s2=simpson(fm,fr,fmr,mid,r);
35    if(fabs(s-s1-s2)<eps)
36          return s1+s2;
37    return rsimpson(l,mid,ml,fl,fm,fml,s1,L,R)+rsimpson(mid,r,mr,fm,
        fr,fmr,s2,L,R);
38  }
39  bool com1(const circle& a,const circle& b)
40  {
41    return (a.lx==b.lx)?(a.rx<b.rx):(a.lx<b.lx);
42  }
43  bool com2(const circle& a,const circle& b)
44  {
45    return a.r>b.r;
46  }
47  inline double cendist(const circle& a,const circle& b)
48  {
49    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
50  }
51  int main()
52  {
53    int k=0;double ans=0.00;scanf("%d",&n);
54    for(int i=1;i<=n;i++)
55    {
56      scanf("%lf␣%lf␣%lf",&c[i].x,&c[i].y,&c[i].r);
57      c[i].r2=c[i].r*c[i].r,c[i].lx=c[i].x-c[i].r,c[i].rx=c[i].x+c[i
          ].r;
58    }
59    sort(c+1,c+1+n,com2);
60    for(int i=1;i<=n;i++)
61    {
62      if(del[i]) continue;
63      cir[++k]=c[i];
64      for(int j=i+1;j<=n;j++)
65      {
66        if(cendist(c[i],c[j])<=(c[i].r-c[j].r)*(c[i].r-c[j].r))
67                  del[j]=1;
68      }
69    }
70    n=k;
71    sort(cir+1,cir+1+n,com1);
72    for(int i=1;i<=n;)
73    {
74      int j;
75          double l=cir[i].lx,r=cir[i].rx,mid,fl,fr,fm,ss;
76      for(j=i+1;j<=n && r>cir[j].lx;r=max(r,cir[j].rx),j++);
77      j--;
78      mid=(l+r)/2.0;
79      fl=f(i,j,l);fr=f(i,j,r);fm=f(i,j,mid);
80      ss=simpson(fl,fr,fm,l,r);
81      ans+=rsimpson(l,r,mid,fl,fr,fm,ss,i,j);
```

```
82      i=j+1;
83    }
84    printf("%.3lf\n",ans);
85    return 0;
86  }
```

5 字符串

5.1 AC自动机

```
void insert(char *s)
{
  int now=1;
  for(;*s;s++)
  {
    if(tree[now].next[*s-'a']==0)
      tree[now].next[*s-'a']=++tot;
    now=tree[now].next[*s-'a'];
  }
  tree[now].flag++;
}
void buildac()
{
  que[tail++]=1;
  while(head<tail)
  {
    int p=que[head++];
    for(int i=0;i<26;i++)
    {
      int t=tree[p].next[i];
      tree[t].fail=p==1?1:tree[tree[p].fail].next[i];
      if(t==0) tree[p].next[i]=tree[t].fail;
      else que[tail++]=t;
    }
  }
}
int find(char *s)
{
  int now=1,ans=0;
  for(;*s;s++)
  {
    while(tree[now].next[*s-'a']==0 && now!=1) now=tree[now].fail;
    now=tree[now].next[*s-'a'];
    now=(now==0)?1:now;
    int tmp=now;
    while(tmp!=1)
    {
      ans+=tree[tmp].flag;
      tree[tmp].flag=0;
      tmp=tree[tmp].fail;
    }
  }
  return ans;
}
```

5.2 KMP

```
int next[MAXN];
```

```
2
3  void getNext(char t[]) {
4      int m = strlen(t);
5      int j = 0, k = -1;
6      next[0] = -1;
7      while(j < m) {
8          if(k == -1 || t[j] == t[k])
9              next[++j] = ++k;
10         else
11             k = next[k];
12     }
13 }
14
15 int kmpCount(char s[], char t[], int c[]) {
16     int ans = 0;
17     int j = 0;
18     int n = strlen(s);
19     int m = strlen(t);
20     if(n == 1 && m == 1) {
21         if(s[0] == t[0])
22             return 1;
23         else
24             return 0;
25     }
26     getNext(t);
27     for(int i = 0; i < n; i ++) {
28         while(j > 0 && s[i] != t[j])
29             j = next[j];
30         if(s[i] == t[j])
31             j ++;
32         if(j == m) {
33             c[ans++] = i - m + 1;
34             j = next[j];
35         }
36     }
37     return ans;
38 }
```

5.3 Manacher 子回文串算法

```
1 scanf("%s",str);
2 for(int i=0;i<m;i++) a[i*2]='*',a[2*i+1]=str[i];
3 n=m*2;a[n++]='*';p[0]=1; int id=0,MAX=0;
4 for(int i=1;i<n;i++)
5 {
6   p[i]=max(min(p[id*2-i],MAX-i),1);
7   while(i>=p[i] && i+p[i]<n && a[i+p[i]]==a[i-p[i]]) p[i]++;
8   if(p[i]+i-1>MAX) MAX=p[i]+i-1,id=i;
9 }
```

5.4 后缀数组

```
void radixsort()
{
  for(int y=1;y>=0;y--)
  {
    memset(c,0,sizeof(c));
    for(int i=1;i<=n;i++) c[re[i].k[y]]++;
    for(int i=1;i<=max(256,n);i++) c[i]+=c[i-1];
    for(int i=n;i>=1;i--) rt[c[re[i].k[y]]--]=re[i];
    for(int i=1;i<=n;i++) re[i]=rt[i];
  }
  for(int i=1;i<=n;i++)
  {
    rank[re[i].id]=rank[re[i-1].id];
    if(re[i].k[0]!=re[i-1].k[0] || re[i].k[1]!=re[i-1].k[1])
      rank[re[i].id]++;
  }
}
void calsa()
{
  re[0].k[0]=-1;
  for(int i=1;i<=n;i++)
    re[i].id=i,re[i].k[0]=str[i],re[i].k[1]=0;
  radixsort();
  for(int k=1;k<=n;k*=2)
  {
    for(int i=1;i<=n;i++)
      re[i].id=i,re[i].k[0]=rank[i],re[i].k[1]=i+k<=n?rank[i+k]:0;
    radixsort();
  }
  for(int i=1;i<=n;i++)
    sa[rank[i]]=i;
}
void calh()
{
  for(int i=1,h=0;i<=n;i++)
  {
    if(rank[i]==1) h=0;
    else
    {
      int k=sa[rank[i]-1];
      for(h=max(0,h-1);str[k+h]==str[i+h];h++);
    }
    height[rank[i]]=h;
  }
}
```

5.5 后缀自动机+拓扑

```
struct node
{
  node *c[11],*pre;
  int l,f,g;
```

```
}mem[N*2],*root,*last,*cur=mem,*q[N*2];
char str[N];
int w[N];
node *newnode()
{
  static int i;
  for(++cur,i=0;i<11;i++)cur->c[i]=0;
  cur->f=0;cur->l=0;cur->g=0;
  return cur;
}
void ins(int c,int l)
{
  static node *np,*p,*q,*r;
  np=newnode();np->l=l;
  for(p=last;p&&!p->c[c];p=p->pre)p->c[c]=np;
  last=np;
  if(!p)np->pre=root;
  else
  {
    if(p->c[c]->l==p->l+1)np->pre=p->c[c];
    else
    {
      q=p->c[c];r=newnode();
      *r=*q;
      r->l=p->l+1;
      q->pre=np->pre=r;
      for(;p&&p->c[c]==q;p=p->pre)p->c[c]=r;
    }
  }
}
    for(i=1;i<=cur-mem;i++)
      w[(mem+i)->l]++;
    for(i=1;i<=l;i++)
      w[i]+=w[i-1];
    for(i=1;i<=cur-mem;i++)
      q[w[(mem+i)->l]--]=mem+i;

//匹配过程
  int ans=0,l=0;
  for(i=1;i<=n;i++)
  {
    c=str[i]-'a';
    if(now->c[c])l++,now=now->c[c];
    else
    {
      while(now&&!now->c[c])now=now->pre;
      if(now)l=now->l+1,now=now->c[c];
      else now=root,l=0;
    }
    ans=max(ans,l);
  }
```

5.6 Tire Tree

```
const int N = 26;

struct node {
    bool tag;
    node *child[N];
    node() {
        tag = false;
        for(int i = 0; i < N; i ++)
            child[i] = NULL;
    }
}*rt;

void insert(char s[]) {
    node *p = rt;
    int k;
    for(int i = 0; s[i]; i ++, p = p->child[k]) {
        k = s[i] - 'a';
        if(!p->child[k])
            p->child[k] = new node();
    }
    p->tag = true;
}

bool search(char s[]) {
    node *p = rt;
    int k;
    for(int i = 0; s[i]; i ++, p = p->child[k]) {
        k = s[i] - 'a';
        if(!p->child[k])
            return false;
    }
    return p->tag;
}

void delet(node *p) {
    for(int i = 0; i < N; i ++)
        if(p->child[i])
            delet(p->child[i]);
    delete(p);
}

int main() {
    return 0;
}
```

可持续化版本

```
int next[3333333][2],size[3333333];
int father[223333],last[323333],pre[333333],son[333333],tot,tag;
int n,m,f[123333][21];
int root[333333],dist[333333],head,tail,que[333333],val[333333];
```

```
5  void add(int a,int b)
6  {
7      tot++;
8      son[tot]=b;
9      pre[tot]=last[a];
10     last[a]=tot;
11 }
12 void bfs()
13 {
14     dist[1]=1;
15     que[tail++]=1;
16     while(head<tail)
17     {
18         int p=que[head++];
19         for(int x=last[p];x!=0;x=pre[x])
20         {
21             if(f[p][0]!=son[x])
22             {
23                 dist[son[x]]=dist[p]+1;
24                 f[son[x]][0]=p;
25                 que[tail++]=son[x];
26                 for(int j=1;j<18;j++)
27                     f[son[x]][j]=f[f[son[x]][j-1]][j-1];
28             }
29         }
30     }
31 }
32 int lca(int a,int b)
33 {
34     if(dist[a]>dist[b]) swap(a,b);
35     for(int i=17;i>=0;i--)
36     {
37         if(dist[f[b][i]]>=dist[a]) b=f[b][i];
38         if(a==b) return a;
39     }
40     for(int i=17;i>=0;i--)
41         if(f[a][i]!=f[b][i]) a=f[a][i],b=f[b][i];
42     return f[a][0];
43 }
44 void ins(int k,int par,int pos,int c)
45 {
46     if(pos<0)
47         return ;
48     int tmp=(c>>pos)&1;
49     next[k][1-tmp]=next[par][1-tmp];
50     next[k][tmp]=++tag;
51     size[next[k][tmp]]=size[next[par][tmp]]+1;
52     ins(next[k][tmp],next[par][tmp],pos-1,c);
53 }
54 int find(int a,int b,int c)
55 {
```

```
56      int ans=0;
57      for(int i=15;i>=0;i--)
58      {
59          int tmp=((c>>i)&1);
60          if(size[next[a][1-tmp]]-size[next[b][1-tmp]]>0)
61          {
62              ans+=(1<<i);
63              a=next[a][1-tmp];
64              b=next[b][1-tmp];
65          }
66          else
67          {
68              a=next[a][tmp];
69              b=next[b][tmp];
70          }
71      }
72      return ans;
73  }
74  int main() {
75      while(scanf("%d␣%d",&n,&m)!=EOF)
76      {
77          head=tail=tag=0;
78          tot=0;
79          for(int i=1;i<=n;i++)
80              scanf("%d",&val[i]),last[i]=0,size[i]=0,dist[i]=0,f[i
                    ][0]=0;
81          for(int i=1;i<=n-1;i++)
82          {
83              int a,b;
84              scanf("%d␣%d",&a,&b);
85              add(a,b);
86              add(b,a);
87          }
88          bfs();
89          for(int i=0;i<tail;i++)
90          {
91              root[que[i]]=++tag;
92              ins(root[que[i]],root[f[que[i]][0]],15,val[que[i]]);
93          }
94          for(int i=1;i<=m;i++)
95          {
96              int a,b,c;
97              scanf("%d␣%d␣%d",&a,&b,&c);
98              int p=lca(a,b);
99              int ans=val[p]^c;
100             if(a!=p)
101                 ans=max(ans,find(root[a],root[p],c));
102             if(b!=p)
103                 ans=max(ans,find(root[b],root[p],c));
104             printf("%d\n",ans);
105         }
```

```
106         }
107         return 0;
108 }
```

6 数学

6.1 拓展GCD

求x, y满足gcd(a, b) = a * x + b * y

```
int extgcd(int a, int b, int & x, int & y) {
    if(b == 0) {
        x == 1, y == 1;
        return a;
    }
    int ret = extgcd(b, a % b, x, y);
    int tmp = x; x = y; y = tmp - a / b * y;
    return ret;
}
```

6.2 快速筛素数

```
bool is[M];
int prm[N];

int getprm(int n){
    int i, j, k = 0;
    int s, e = (int)(sqrt(0.0 + n) + 1);
    memset(is, 1, sizeof(is));
    prm[k++] = 2; is[0] = is[1] = 0;
    for (i = 4; i < n; i += 2) is[i] = 0;
    for (i = 3; i < e; i += 2) if (is[i]) {
        prm[k++] = i;
        for (s = i * 2, j = i * i; j < n; j += s) is[j] = 0;
    }
    for ( ; i < n; i += 2) if (is[i]) prm[k++] = i;
    return k;
}
```

6.3 欧拉函数

单独求欧拉函数phi(x)

```
int phi(int n) {
  int m = 1;
  for (int i = 2; n > 1; ++i) {
    if (n % i == 0) {
      m *= i - 1;
      n /= i;
      while (n % i == 0) {
        m *= i;
        n /= i;
      }
    }
  }
  return m;
}
```

预处理phi(x)

```
for (i = 1; i <= maxn; i++) phi[i] = i;
for (i = 2; i <= maxn; i += 2) phi[i] /= 2;
for (i = 3; i <= maxn; i += 2) if(phi[i] == i) {
    for (j = i; j <= maxn; j += i)
        phi[j] = phi[j] / i * (i - 1);
}
```

6.4 快速幂

```
lint quickpow(lint a, lint b, lint m) {
    lint ans = 1;
    while(b) {
        if(b & 1) {
            ans = (ans * a) % m;
            -- b;
        }
        b >>= 1;
        a = a * a % m;
    }
    return ans;
}
```

6.5 Polya定理

题意为：给你c种颜色的珠子，和一个长度为s(c,s<32)的项链，用这c种珠子串成这个项链，项链可以旋转和翻转，经过旋转和翻转所得的项链视为同一种项链，现在告诉你颜色总数c和项链的长度s，求共能组成几条不同的项链。

1.旋转置换

依次顺时针旋转1　n个，循环个数为gcd(i, n)

2.翻转置换

当n为偶数时,分两种情况,一种是中心轴在两个对称对象上,则循环个数为n/2+1,另一种是对称轴两边分别有n/2个对象,则循环个数为n/2;

当n为奇数时,对称轴就只能在一个对象上,则循环个数为n/2+1;

```
int c, s, ans;

int gcd(int a, int b) {
    if(b == 0)  return a;
    return gcd(b, a % b);
}

void solve() {
    int t = 0;
    ans = 0;
    for(int i = 1; i <= s; i ++) {
        ans += pow(c, gcd(i, s));
        t ++;
    }
    if(s % 2 == 0) {
        for(int i = 0; i < s / 2; i ++) {
```

```
            ans += pow(c, s / 2);
            t ++;
        }
        for(int i = 0; i < s / 2; i ++) {
            ans += pow(c, s / 2 + 1);
            t ++;
        }
    }
    else {
        for(int i = 0; i < s; i ++) {
            ans += pow(c, s / 2 + 1);
            t ++;
        }
    }
    ans /= t;
}

int main() {
    while(scanf("%d%d", &c, &s) != EOF) {
        if(c == 0 && s == 0) {
            return 0;
        }
        solve();
        printf("%d\n", ans);
    }
    return 0;
}
```

给出两个整数n和p，代表n个珠子，n种颜色，要求不同的项链数，并对结果mod（p）处理，置换只有旋转一种形式。
这道题代表了一类题目的优化
裸的算法是 Σn(gcd(n,i)) 1<i<=n
复杂度过高，进行优化。
置换群种循环的个数L = n / gcd(n, i)
因为如果L | n, 则有n / L | n
则环的长度L的范围是1 ∼ sqrt(L)
令a = gcd(n, i), 设i = at
则只有i与t互质的时候，gcd(i, t) = a
则最后可以优化为Σ(Φ(i) * n^i) % p

```
long long ans;
int t, num, n, p;
int isprime[50001];
int prime[8001];

void getprime() {
    num = 0;
    for(int i = 2; i <= 50000; i ++) {
        if(!isprime[i]) {
            prime[num ++] = i;
            for(int j = 1; j * i <= 50000; j ++) {
                isprime[i * j] = 1;
```

```
13              }
14          }
15      }
16  }
17
18  int euler(int x) {
19      int res = x;
20      for(int i = 0; i < num && prime[i] * prime[i] <= x; i++) {
21          if(x % prime[i] == 0) {
22              res = res / prime[i] * (prime[i] - 1);
23              while(x % prime[i] == 0) {
24                  x /= prime[i];
25              }
26          }
27      }
28      if(x > 1) res = res / x * (x - 1);
29      return res;
30  }
31
32  LL quickpow(LL m , LL n , LL k) {
33      LL tmp = 1;
34      m %= k;
35      while(n) {
36          if(n & 1)
37              tmp = (tmp * m) % k;
38          m = (m * m) % k;
39          n >>= 1;
40      }
41      return tmp;
42  }
43
44  int main() {
45      getprime();
46      scanf("%d", &t);
47      while(t --) {
48          ans = 0;
49          scanf("%d%d", &n, &p);
50          for(int i = 1; i < sqrt(n); i ++) {
51              if(n % i == 0) {
52                  ans = (ans + euler(i) % p * quickpow(n, n / i - 1,
                        p) + euler(n / i) % p * quickpow(n, i - 1, p)) %
                         p;
53                  //cout << quickpow(n,n-1,p) << " " << quickpow(n, n
                        / i - 1, p) << " " << euler(i) << " " << euler(
                        n / i) << endl;
54              }
55          }
56          if((int)sqrt(n) * (int)sqrt(n) == n) {
57              ans = (ans + quickpow(n, sqrt(n) - 1, p) * (euler(sqrt(
                    n)) % p)) % p;
58          }
```

```
59          ans %= p;
60          cout << ans << endl;
61      }
62      return 0;
63  }
```

6.6 高斯消元

```
1   const int N = 50;
2
3   int a[N][N];
4   int x[N];
5   bool fx[N];
6
7   int gcd(int a, int b) {
8       int t;
9       while(b) {
10          t = b;
11          b = a % b;
12          a = t;
13      }
14      return a;
15  }
16
17  inline int lcm(int a, int b) {
18      return a / gcd(a, b) * b;
19  }
20
21  int gauss(int e, int v) {
22      memset(x, 0, sizeof(x));
23      memset(fx, true, sizeof(fx));
24      int k;
25      int col = 0;
26      for(k = 0; k < e && col < v; k ++, col ++) {
27          int mxr = k;
28          for(int i = k + 1; i < e; i ++)
29              if(abs(a[i][col]) > abs(a[mxr][col]))
30                  mxr = i;
31          if(mxr != k)
32              for(int j = k; j <= v; j ++)
33                  _swap(a[k][j], a[mxr][j]);
34          if(!a[k][col]) {
35              k --;
36              continue;
37          }
38          for(int i = k + 1; i < e; i ++) {
39              if(a[i][col]) {
40                  int LCM = lcm(abs(a[i][col]), abs(a[k][col]));
41                  int ta = LCM / abs(a[i][col]);
42                  int tb = LCM / abs(a[k][col]);
43                  if(a[i][col] * a[k][col] < 0)
```

```
                    tb = -tb;
                for(int j = col; j <= v; j ++)
                    a[i][j] = a[i][j] * ta - a[k][j] * tb;
            }
        }
    }
    for(int i = k; i < e; i ++)
        if(a[i][col])
            return -1;
    if(k < v) {
        for(int i = k - 1; i >= 0; i --) {
            int fn = 0;
            int fid;
            for(int j = 0; j < v; j ++)
                if(a[i][j] && fx[j]) {
                    fn ++;
                    fid = j;
                }
            if(fn > 1)
                continue;
            int tmp = a[i][v];
            for(int j = 0; j < v; j ++)
                if(a[i][j] && j != fid)
                    tmp -= a[i][j] * x[j];
            x[fid] = tmp / a[i][fid];
            fx[fid] = 0;
        }
        return v - k;
    }
    for(int i = v - 1; i >= 0; i --) {
        int tmp = a[i][v];
        for(int j = i + 1; j < v; j ++)
            if(a[i][j])
                tmp -= a[i][j] * x[j];
        if(tmp % a[i][i])
            return -2;
        x[i] = tmp / a[i][i];
    }
    return 0;
}
```

6.7 组合数

```
for(int i = 0; i <= MAXN; i ++) {
    c[i][0] = c[i][i] = 1;
    for(int j = 1; j < i; j ++) {
        c[i][j] = c[i - 1][j] + c[i - 1][j - 1];
    }
}
```

6.8 数学基础知识

斯特林[striling]公式:
lnN!=NlnN－N+0.5ln(2N*/Pi)
所以base进制下N!的位数为[(lnN!/ln(base))]

费马小定理:
假如a是一个整数，p是一个质数，那么a^p - a 是p的倍数

四次方和公式:
$(6n^5 + 15n^4 + 10n^3$ - n) / 30

6.9 中国剩余定理、lucas定理

```
long long pri[5]={0,2,3,4679,35617},mod=999911659,jiecheng
   [5][40000],ans[5];
long long quickpower(long long k,long long p,long long M)
{
  if(p==0) return 1;
  if(p==1) return k;
  long long tmp=quickpower(k,p/2,M);
  tmp*=tmp,tmp%=M;if(p&1) tmp*=k;tmp%=M;
  return tmp;
}
long long C(long long n,long long m,long long M)
{
  return jiecheng[M][n]*quickpower(jiecheng[M][n-m]*jiecheng[M][m]%
     pri[M],pri[M]-2,pri[M])%pri[M];
}
long long cal(long long n,long long m,long long M)
{
  if(!m) return 1;
  if(n%pri[M]<m%pri[M]) return 0;
  return C(n%pri[M],m%pri[M],M)*cal(n/pri[M],m/pri[M],M)%pri[M];
}
void exgcd(long long a,long long b,long long &x,long long &y)
{
  if(b==0)
  {
    x=1,y=0;
    return;
  }
  exgcd(b,a%b,x,y);
  long long tmp=x;
  x=y;
  y=tmp-a/b*y;
}
int main()
{
//  freopen("ancient.in","r",stdin);freopen("ancient.out","w",
   stdout);
```

```
35    long long n,g,times=0;scanf("%lld␣%lld",&n,&g);g%=mod;
36    if(g==0) {printf("0\n");return 0;}
37    for(int i=1;i<=4;i++)
38    {
39      jiecheng[i][0]=1;
40      for(int j=1;j<=pri[i];j++) jiecheng[i][j]=jiecheng[i][j-1]*j,
          jiecheng[i][j]%=pri[i];
41    }
42    for(long long i=1;i*i<=n;i++)
43    {
44      if(n%i==0)
45      {
46        for(int j=1;j<=4;j++)
47        {
48          ans[j]+=cal(n,i,j),ans[j]%=pri[j];
49          if(i*i!=n) ans[j]+=cal(n,n/i,j),ans[j]%=pri[j];
50        }
51      }
52    }
53    for(int i=1;i<=4;i++)
54    {
55      long long Mi=(mod-1)/pri[i],x,y;
56      exgcd(Mi,pri[i],x,y);
57      times+=Mi*x*ans[i],times%=(mod-1);
58    }
59    if(times<0) times+=(mod-1);
60    printf("%lld\n",quickpower(g,times,mod));
61  //  system("pause");
62    return 0;
63  }
```

7 其他

7.1 爆栈define

c ++

```
#pragma comment(linker, "/STACK:1024000000,1024000000")
```

g ++

```
char* MyStack=new char[33554432];
int main() {
    char* SysStack=NULL;
    MyStack+=33554432-1048576;  //32M
    __asm__
    (
        "movl␣%%esp,%%eax\n\t"
        "movl␣%1,%%esp\n\t"
        :"=a"(SysStack)
        :"m"(MyStack)
    );
    //自己的东西
    __asm__
    (
        "movl␣%0,%%esp\n\t"
       ::"m"(SysStack)
    );
    return 0;
}
```

7.2 高精度计算

```
template <int SIZE,int YA_COUNT>
class int128
{
  private:
    int val[SIZE+2],ya;
    char form[10];
  public:
    int size;
    int128(int x=0)
    {
      memset(val,0,sizeof(val));
      ya=1;
      for(int i=1;i<=YA_COUNT;i++) ya*=10;
      sprintf(form,"%%0%d",YA_COUNT);
      int p=0;
      while(x!=0)
      {
        val[++p]=x%ya;
        x/=ya;
      }
      size=max(p,1);
```

```
22      }
23      int& operator [](int x)
24      {
25        return val[x];
26      }
27      friend bool operator >(int128& a,int128& b)
28      {
29        int i,la=a.size,lb=b.size;
30        if(la>lb) return 1;
31        else if(la<lb) return 0;
32        else
33        {
34          for(i=la;i>=1;i--)
35            if(a[i]>b[i]) return 1;
36            else if(a[i]<b[i]) return 0;
37        }
38        return 0;
39      }
40      friend bool operator ==(int128& a,int128& b)
41      {
42        int i,la=a.size,lb=b.size;
43        if(la!=lb) return 0;
44        else
45        {
46          for(i=la;i>=1;i--)
47            if(a[i]!=b[i]) return 0;
48        }
49        return 1;
50      }
51      friend int128 operator +(int128& a,int128& b)
52      {
53        int128 c;
54        c.size=max(a.size,b.size)+2;
55        for(int i=1;i<=c.size;i++)
56        {
57          c[i]=a[i]+b[i]+c[i-1]/a.ya;
58          c[i-1]%=a.ya;
59        }
60        while(!c[c.size] && c.size>1)
61          c.size--;
62        return c;
63      }
64      friend int128 operator -(int128& a,int128& b)
65      {
66        int128 c;
67        c.size=max(a.size,b.size);
68        for(int i=1;i<=c.size;i++)
69        {
70          c[i]+=a[i]-b[i];
71          if(c[i]<0)
72          {
```

```
73            c[i]+=a.ya;
74            c[i+1]--;
75          }
76        }
77        while(!c[c.size] && c.size>1)
78          c.size--;
79        return c;
80      }
81      friend int128 operator /(int128& a,int b)
82      {
83        int128 c=a;
84        for(int i=c.size;i>=1;i--)
85        {
86          c[i-1]+=c[i]%b*a.ya;
87          c[i]/=b;
88        }
89        c[0]/=a.ya;
90        while(!c[c.size] && c.size>1)
91          c.size--;
92        return c;
93      }
94      friend int128 operator *(int128& a,int128& b)
95      {
96        int128 c;
97        c.size=a.size+b.size;
98        for(int i=1;i<=a.size;i++)
99          for(int j=1;j<=b.size;j++)
100           c[i+j-1]+=a[i]*b[j];
101       for(int i=2;i<=c.size;i++)
102       {
103         c[i]+=c[i-1]/a.ya;
104         c[i-1]%=a.ya;
105       }
106       while(!c[c.size] && c.size>1)
107         c.size--;
108       return c;
109     }
110     friend int128 operator *(int128& a,int b)
111     {
112       int128 c=b;
113       int128 d=a*c;
114       return d;
115     }
116     friend istream& operator >>(istream& in,int128& a)
117     {
118       memset(a.val,0,sizeof(a.val));
119       char buf[20005];
120       scanf("%s",buf);
121       int len=strlen(buf);
122       for(int i=0;i<len;i++)
```

```
123         a[(len-i+YA_COUNT-1)/YA_COUNT]=a[(len-i+YA_COUNT-1)/
               YA_COUNT]*10+buf[i]-'0';
124       a.size=SIZE;
125       while(!a[a.size] && a.size>1)
126         a.size--;
127       return in;
128     }
129     friend ostream& operator <<(ostream& out,int128& a)
130     {
131       int p=a.size;
132       while(p>1 && a[p]==0) p--;
133       out<<a[p];
134       for(p--;p>=1;p--)
135         printf("%04d",a[p]);
136       return out;
137     }
138 };
139 #define bint int128<5125,4>
```

7.3 vim配置文件-vimrc

```
1  source $VIMRUNTIME/mswin.vim
2  behave mswin
3  imap <cr> <cr><left><right>
4  imap <c-]> {<cr>}<c-o>O<left><right>
5  imap <c-d> <c-o>dd
6  map <f6> =a{
7  map <c-t> :tabnew<cr>
8  syn on
9  colo ron
10 set gfn=Courier\ 10\ Pitch\ 12
11 set ru nu et sta nowrap ar acd ww=<,>,[,] sw=4 ts=4 cin noswf
12 syn on
13 hi LineNr guifg=LightBlue
14 set autochdir
15 set hlsearch
16 set selection=inclusive
17
18 map <f9> :call CR2()<cr><space>
19 func CR2()
20 exec "update"
21 exec "!xterm␣-u8␣-fn␣10*20␣-e␣\"g++␣%<.cpp␣-Wall␣-o␣%<␣&&␣time␣./%<
      ␣;␣read␣-n␣1\""
22 endfunc
23
24 map <f10> :call CR()<cr><space>
25 func CR()
26 exec "update"
27 exec "!xterm␣-u8␣-fn␣10*20␣-e␣\"g++␣%<.cpp␣-Wall␣-o␣%<␣&&␣time␣./%<
      ␣<␣%<.in␣;␣read␣-n␣1\""
28 endfunc
29
```

```
map <f2> :call CR3()<cr><space>
func CR3()
let l = 0
let l = l + 1 | call setline(l, '/*')
let l = l + 1 | call setline(l, ' * Author: FreeArcanite@WHU')
let l = l + 1 | call setline(l, ' * Created Time: '.strftime('%F %
   R'))
let l = l + 1 | call setline(l, ' * File Name: '.expand('%'))
let l = l + 1 | call setline(l, ' */')
let l = l + 1 | call setline(l, '#include <iostream>')
let l = l + 1 | call setline(l, '#include <cstdio>')
let l = l + 1 | call setline(l, '#include <algorithm>')
let l = l + 1 | call setline(l, '#include <cmath>')
let l = l + 1 | call setline(l, '#include <cstring>')
let l = l + 1 | call setline(l, '#include <string>')
let l = l + 1 | call setline(l, '#include <map>')
let l = l + 1 | call setline(l, '#include <vector>')
let l = l + 1 | call setline(l, '#include <queue>')
let l = l + 1 | call setline(l, '#include <stack>')
let l = l + 1 | call setline(l, '#include <set>')
let l = l + 1 | call setline(l, '#include <list>')
let l = l + 1 | call setline(l, '#include <iomanip>')
let l = l + 1 | call setline(l, '#define out(v) cout << #v << " " << v << endl')
let l = l + 1 | call setline(l, '#define filn freopen("in.txt", "r", stdin)')
let l = l + 1 | call setline(l, '#define flout freopen("out.txt", "w", stdout)')
let l = l + 1 | call setline(l, '#define pb push_back')
let l = l + 1 | call setline(l, '#define PI acos(-1.0)')
let l = l + 1 | call setline(l, 'using namespace std;')
let l = l + 1 | call setline(l, 'typedef long long lint;')
let l = l + 1 | call setline(l, 'const int inf = 1e9 + 7;')
let l = l + 1 | call setline(l, 'const double eps = 1e-8;')
let l = l + 1 | call setline(l, 'inline int sgn(const double x) { return (x > eps) - (x < -eps);}')
let l = l + 1 | call setline(l, 'template <class T> T __abs(T x) {if (x < 0) x = -x; return x;}')
let l = l + 1 | call setline(l, 'template <class T> bool get_max(T &a, const T b) {return a < b ? a = b, true : false;}')
let l = l + 1 | call setline(l, 'template <class T> bool get_min(T &a, const T b) {return a > b ? a = b, true : false;}')
let l = l + 1 | call setline(l, 'template <class T> void __swap(T &a, T &b) {T t = a; a = b; b = t;}')
let l = l + 1 | call setline(l, '')
let l = l + 1 | call setline(l, 'int main() {')
let l = l + 1 | call setline(l, '    return 0;')
let l = l + 1 | call setline(l, '}')
endfunc

map <f4> :call AddComment()<cr>
```

```
func AddComment()
  if (getline('.')[0] == '//')
    normal ^xx
  else
    normal 0i//
  endif
endfunc
```

7.4 头文件

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <cmath>
#include <cstring>
#include <string>
#include <map>
#include <vector>
#include <queue>
#include <stack>
#include <set>
#include <list>
#include <iomanip>
#define out(v) cout << #v << "␣" << v << endl
#define filn freopen("in.txt", "r", stdin)
#define flout freopen("out.txt", "w", stdout)
#define pb push_back
#define PI acos(-1.0)
using namespace std;
typedef long long lint;
const int inf = 1e9 + 7;
const double eps = 1e-8;
inline int sgn(const double x) {return (x > eps) - (x < -eps);}
template <class T> T _abs(T x) {if(x < 0) x = -x; return x;}
template <class T> bool get_max(T &a, const T b) {return a < b ? a
   = b, true : false;}
template <class T> bool get_min(T &a, const T b) {return a > b ? a
   = b, true : false;}
```