华南师范大学
ACM-ICPC 集训队
DeDeRong

2021 年 5 月 12 日

目录

1 图论

1.1 最短路径

1.1.1 Dijkstra

```
const int maxn = 1e4;
const int inf = 0x3f3f3f3f;

//d数组用来记录源点s到顶点i的最短距离
//v表示该顶点是否在顶点集S中
//g邻接矩阵存图，g[i][j]表示i到j的边的权值，无边时为inf
//n为顶点数量
int d[maxn], v[maxn];
int g[maxn][maxn];
int n;
void dij(int s)
{
    memset(v, 0, sizeof(v));
    for(int i=1;i<=n;i++)
    d[i] = g[s][i];
    v[s] = 1;
    for(int i=1;i<=n;i++)
    {
        int u = 0;
        for(int j=1;j<=n;j++)
        {
            if(!v[j] && (u==0 || d[j] < d[u]))
                u = j;
        }
        if(u==0)return ;
        v[u] = 1;
        for(int j=1;j<=n;j++)
        {
            d[j] = min(d[j], d[u]+g[u][j]);
        }
    }
}
```

1.1.2 Dijkstra 优化

```
const int maxn = 1e4;
const int inf = 0x3f3f3f3f;
typedef pair<int, int> P; //first表示最短距离，second表示顶点编号
//边：to表示这条边指向的顶点，权值为w
struct Edge
{
    int to, w;
};
//用vector实现邻接表
vector<Edge> g[maxn];
int d[maxn]; //记录源点到顶点i的最短距离
int n;

void dij(int s)
{
    priority_queue<P, vector<P>, greater<P> > q;
    memset(d, inf, sizeof(d));
    d[s] = 0;
    q.push(P(0, s));
    while(!q.empty())
    {
        P p = q.top();
        q.pop();
        int u = p.second;
```

```
25          if(d[u] < p.first) continue;
26          for(int i=0; i<g[u].size(); i++)
27          {
28              Edge e = g[u][i];
29              if(d[e.to] > d[u] + e.w)
30              {
31                  d[e.to] = d[u] + e.w;
32                  q.push(P(d[e.to], e.to));
33              }
34          }
35      }
36  }
```

1.1.3 Floyd

```
1  int g[maxn][maxn];
2  int n;
3  void floyd()
4  {
5      for(int k=1;k<=n;k++)
6          for(int i=1;i<=n;i++)
7              for(int j=1;j<=n;j++)
8                  g[i][j] = min(g[i][j], g[i][k] + g[k][j]);
9  }
```

1.1.4 Bellman-Ford

```
1  const int maxn = 1e4;
2  const int inf = 0x3f3f3f3f; //常用于表示无穷大
3
4  //边结构体，记录u->v的边，权值为w
5  struct Edge
6  {
7      int u, v, w;
8      Edge(int uu, int vv, int ww) { u=uu; v=vv; w=ww; }
9      Edge(){}
10 }e[maxn];
11 int edgecnt; // 边的数量
12 //加边操作
13 void addEdge(int u, int v, int w)
14 {
15     e[edgecnt++] = Edge(u, v, w);
16 }
17
18 int n; //顶点总数
19 int d[maxn]; //记录最短距离的数组
20
21 //存在负权回路则返回true，否则返回false
22 bool bellman_ford(int s)
23 {
24     memset(d, inf, sizeof(d));
25     d[s] = 0;
26     //进行n-1次松弛操作,第n次检查是否含有负权回路
27     for(int i=1;i<=n;i++)
28     {
29         int flag = 0;
30         for(int j=0; j<edgecnt; j++)
31         {
32             Edge t = e[j];
33             int u, v, w;
34             u = t.u; v = t.v; w = t.w;
35             if(d[v] > d[u] + w)
36             {
37                 d[v] = d[u] + w;
```

```
                flag = 1;
            }
        }
        if(!flag) return false;
        if(i==n && flag) return true;
    }
    return false;
}
```

1.1.5 SPFA

```
const int maxn = 1e4;
const int inf = 0x3f3f3f3f; //常用于表示无穷大

//边结构体，to表示边指向的顶点编号，权值为w
struct Edge
{
    int to, w;
    Edge(int tt, int ww) { to = tt; w = ww; }
    Edge(){}
};
//vector实现的邻接表
vector<Edge> g[maxn];
int n;//顶点数
//d表示最短距离， inq[i]表示结点是否在队列中，为1则在，cnt[i]记录i入队的次数
int d[maxn], inq[maxn], cnt[maxn];
//初始化
void init()
{
    memset(d, inf, sizeof(d));
    memset(inq, 0, sizeof(inq));
    memset(cnt, 0, sizeof(cnt));
}
//返回true表示存在负权回路
bool spfa(int s)
{
    init();
    d[s] = 0;
    inq[s] = 1;
    cnt[s] = 1;
    queue<int> q;
    q.push(s);
    while(!q.empty())
    {
        int u = q.front();
        inq[u] = 0;
        q.pop();
        for(int i=0;i < g[u].size(); i++)
        {
            Edge e = g[u][i];
            if(d[e.to] > d[u] + e.w)
            {
                d[e.to] = d[u] + e.w;
                if(inq[e.to] == 0)
                {
                    inq[e.to] = 1;
                    q.push(e.to);
                    cnt[e.to]++;
                    if(cnt[e.to] > n) return true;
                }
            }
        }
    }
    return true;
}
```

1.2 次短路

```
#include <bits/stdc++.h>
#define INF 1e16+100
#define ms(x,y) memset(x,y,sizeof(x))
using namespace std;

typedef long long ll;
typedef pair<ll,ll> P;

const double pi = acos(-1.0);
const int mod = 1e9 + 7;
const int maxn = 1e5 + 5;

struct Edge{
    ll to,cost;
};

ll n,m;
vector<Edge> a[maxn];
ll dist[maxn],dist2[maxn];

void addedge(ll u,ll v,ll w)
{
    a[u].push_back(Edge{v,w});
    a[v].push_back(Edge{u,w});
}

void solve()
{
    priority_queue<P, vector<P>, greater<P> >que;
    //ms(dist,INF);
    //ms(dist2,INF);
    fill(dist,dist+n,INF);
    fill(dist2,dist2+n,INF);
    dist[0]=0;
    que.push(P(0,0));
    while(que.size())
    {
        P u=que.top();que.pop();
        int v=u.second;
        ll d=u.first;
        if(dist2[v]<d) continue;    //不是次短距离则抛弃
        for(int i=0;i<a[v].size();i++)
        {
            Edge e=a[v][i];
            ll d2=d+e.cost;
            if(dist[e.to]>d2)   //更新最短
            {
                swap(dist[e.to],d2);
                que.push(P(dist[e.to],e.to));
            }
            if(dist2[e.to]>d2&&dist[e.to]<d2)   //更新次短
            {
                dist2[e.to]=d2;
                que.push(P(dist2[e.to],e.to));
            }
        }
    }
    printf("%lld\n",dist2[n-1]);
}

int main()
{
    //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
```

```
65      int t;
66      scanf("%d",&t);
67      while(t--)
68      {
69          scanf("%lld%lld",&n,&m);
70          for(int i=0;i<n;i++) a[i].clear();
71          for(int i=0;i<m;i++)
72          {
73              ll p,q,w;
74              scanf("%lld%lld%lld",&p,&q,&w);
75              addedge(p-1,q-1,w);
76          }
77          solve();
78      }
79      return 0;
80  }
```

1.3 LCA

1.3.1 倍增

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  #define inf 0x3f3f3f3f
5  typedef pair<int, int> P;
6  const int maxn = 5e5+5;
7  const ll mod = 1e9+7;
8
9  vector<int> son[maxn]; // 存储儿子顶点
10 // dep[i]表示顶点i的深度, n个顶点, m个询问, rt为树根, fa数组用来预处理顶点i向上跳2
     ^j步之后的顶点
11 int dep[maxn], n, m, rt, fa[maxn][20];
12 int v[maxn]={0}; // 是否访问标记
13
14 // pre是父顶点, rt是当前顶点
15 void dfs(int pre, int rt)
16 {
17     dep[rt] = dep[pre]+1; // 当前顶点的深度为父顶点加一
18     fa[rt][0] = pre; // 当前顶点向上跳一步为父顶点
19     v[rt] = 1; // 访问
20     // dp预处理
21     for(int i=1;i<=19;i++)
22     fa[rt][i] = fa[fa[rt][i-1]][i-1];
23     // 继续dfs
24     for(int i=0;i<son[rt].size();i++)
25     if(v[son[rt][i]]==0)
26     dfs(rt, son[rt][i]);
27 }
28
29 // 求解LCA(a, b)
30 int lca(int a, int b)
31 {
32     if(dep[a] < dep[b])
33         swap(a, b);
34     for(int i=19;i>=0;i--)
35     {
36         if(dep[a]-dep[b] >= (1<<i))
37         {
38             a = fa[a][i];
39         }
40     }
41     if(a==b)return a;
42     for(int i=19;i>=0;i--)
43     {
```

```
        if(fa[a][i] != fa[b][i])
        {
            a = fa[a][i];
            b = fa[b][i];
        }
    }
    return fa[a][0];
}

int main()
{
    scanf("%d%d%d", &n, &m, &rt);
    for(int i=1;i<n;i++)
    {
        int a, b;
        scanf("%d%d", &a, &b);
        son[a].push_back(b);
        son[b].push_back(a);
    }
    memset(fa, 0, sizeof(fa));
    memset(dep, inf, sizeof(dep));
    v[0]=1;
    dep[0] = 0;
    dfs(0, rt);
    for(int i=1;i<=m;i++)
    {
        int a, b;
        scanf("%d%d", &a, &b);
        printf("%d\n", lca(a, b));
    }
    return 0;
}
```

1.3.2 RMQ

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
#define inf 0x3f3f3f3f
typedef pair<int, int> P;
const int maxn = 5e5+5;
const ll mod = 1e9+7;

vector<int> g[maxn]; // 存图
// dep记录DFS序中每一个顶点的深度， vis记录DFS序， id记录顶点i第一次在DFS序中的位置， st表
int dep[maxn<<1]={0}, vis[maxn<<1]={0}, id[maxn]={0}, st[maxn<<1][25];
// dfs序计数用，看代码能理解
int dfs_c=1;

// 父顶点为pre， 当前顶点为now， 当前深度为d
void dfs(int pre, int now, int d)
{
    id[now] =  dfs_c; // now顶点在DFS序中第一次出现的位置是dfs_c
    dep[dfs_c] = d; // 记录now的深度
    vis[dfs_c++] = now; // DFS序中第dfs_c个顶点是now，同时将dfs_c加一
    for(int i=0;i<g[now].size();i++)
    {
        if(g[now][i]!=pre)
        {
            dfs(now, g[now][i], d+1);
            vis[dfs_c] = now;
            dep[dfs_c++] = d;
        }
    }
```

```
}

// 预处理st表
void getSt(int n)
{
    for(int i=1;i<=n;i++)
    st[i][0] = i;
    for(int j=1; (1<<j)<=n; j++)
    {
        for(int i=1;i+(1<<j)<=n; i++)
        {
            int a = st[i][j-1], b = st[i+(1<<(j-1))][j-1];
            if(dep[a] < dep[b])
                st[i][j] = a;
            else st[i][j] = b;
        }
    }
}

// 查询DFS序中区间[l, r]深度最小的顶点在DFS序中的位置
int query(int l, int r)
{
    int k = log2(r-l+1);
    int a = st[l][k];
    int b = st[r-(1<<k)+1][k];
    // 返回深度较小的那一个顶点在DFS序中的位置
    if(dep[a]<dep[b])return a;
    else return b;
}

// 求LCA(a, b)
int lca(int a, int b)
{
    int x, y;
    x = id[a], y = id[b];
    if(x>y)return vis[query(y, x)];
    else return vis[query(x, y)];
}

// 检查用的
void check(int n)
{
    for(int i=1;i<=dfs_c;i++)cout<<dep[i]<<" ";cout<<"\n\n";
    for(int i=1;i<=dfs_c;i++)cout<<vis[i]<<" ";cout<<"\n\n";
    for(int i=1;i<=n;i++)cout<<id[i]<<" ";cout<<"\n\n";
}

int main()
{
    int n, m, rt;
    scanf("%d%d%d", &n, &m, &rt);
    for(int i=1;i<n;i++)
    {
        int a, b;
        scanf("%d%d", &a, &b);
        g[a].push_back(b);
        g[b].push_back(a);
    }
    dfs(0, rt, 1);
    getSt(dfs_c);
    //check(n);
    for(int i=1;i<=m;i++)
    {
        int a, b;
        scanf("%d%d", &a, &b);
        printf("%d\n", lca(a, b));
```

```
96      }
97      return 0;
98  }
```

1.4 强连通分量

```
1   vector<int> g[maxn];
2   int low[maxn], dfn[maxn], sta[maxn], ins[maxn], belong[maxn];
3   int cnt, ind, tot; //cnt: 强连通分量的数量, ind: 时间戳, tot: sta的top
4
5   void init()
6   {
7       memset(ins, 0, sizeof(ins));
8       memset(belong, 0, sizeof(belong));
9       memset(dfn, 0, sizeof(dfn));
10      cnt = ind = tot = 0;
11  }
12
13  void Tarjan(int u)
14  {
15      low[u] = dfn[u] = ++ind;
16      ins[u] = 1;
17      sta[++tot] = u;
18      for(int i=0;i<g[u].size();i++)
19      {
20          int v = g[u][i];
21          if(!dfn[v])
22          {
23              Tarjan(v);
24              low[u] = min(low[u], low[v]);
25          }
26          else if(ins[v])
27          low[u] = min(low[u], dfn[v]);
28      }
29      int p;
30      if(low[u] == dfn[u])
31      {
32          ++cnt;
33          do
34          {
35              p = sta[tot--];
36              belong[p] = cnt;
37              ins[p] = 0;
38          }while(p != u);
39      }
40  }
```

1.5 割点

```
1   vector<int> g[maxn];
2   // iscut[i]: 若顶点i是割点, 则为1, 反之为0
3   int low[maxn], dfn[maxn], iscut[maxn];
4   int ind;
5
6   void init()
7   {
8       memset(dfn, 0, sizeof(dfn));
9       memset(iscut, 0, sizeof(iscut));
10      ind = 0;
11  }
12
13  // pa为u的父节点, 初始时Tarjan(i, i)
14  void Tarjan(int u, int pa)
```

```
{
    int cnt = 0; //用来记录子树的数量
    low[u] = dfn[u] = ++ind;
    for(int i=0;i<g[u].size();i++)
    {
        int v = g[u][i];
        if(!dfn[v])
        {
            Tarjan(v, u);
            low[u] = min(low[u], low[v]);
            // 若low[v]>=dfn[u], 并且u不是根节点, 则u是割点
            if(low[v] >= dfn[u] && pa!=u)
            iscut[u] = 1;
            // 若u是根节点, 则cnt++
            if(u == pa)
                cnt++;
        }
        else if(v != pa) //若v不等于父节点
        low[u] = min(low[u], dfn[v]);
    }
    if(cnt>=2 && u==pa) //根节点子树数量大于等于2, 则为割点
        iscut[u] = 1;
}
```

1.6 桥

```
// 用链式前向星来存储边
struct Edge
{
    // iscut表示是否为桥
    int to, next, iscut;
}e[maxn*maxn*2];

int head[maxn], low[maxn], dfn[maxn];
int ind, tot; // tot是边的数量

void init()
{
    memset(head, -1, sizeof(head));
    memset(dfn, 0, sizeof(dfn));
    ind = tot = 0;
}

void addedge(int u, int v)
{
    e[tot].to = v;
    e[tot].next = head[u];
    e[tot].iscut = 0;
    head[u] = tot++;
}

void Tarjan(int u, int pa)
{
    low[u] = dfn[u] = ++ind;
    for(int i=head[u]; ~i; i = e[i].next)
    {
        int v = e[i].to;
        if(v == pa) continue;
        if(!dfn[v])
        {
            Tarjan(v, u);
            low[u] = min(low[u], low[v]);
            // 是桥
            if(low[v] > dfn[u])
            {
```

```
                e[i].iscut = e[i^1].iscut = 1;
            }
        }
        else
        {
            low[u] = min(low[u], dfn[v]);
        }
    }
}
```

1.7 最大流

1.7.1 Dinic

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int, int> P;
const int maxn = 1e6+5;
const int inf = 0x3f3f3f3f;
const int mod = 1e9+7;

// 用链式前向星来存储图
struct ed
{
    int to, val, ne;
}edge[maxn<<1];
int head[maxn], dep[maxn];
// 顶点数n, 边数m, 源点s, 汇点e, 加边时的指针tot
int n, m, s, e, tot;

void init()
{
    tot = -1;
    memset(head, -1, sizeof(head));
}

void addEdge(int u, int v, int val)
{
    edge[++tot].to = v;
    edge[tot].val = val;
    edge[tot].ne = head[u];
    head[u] = tot;
}

// 就是最普通的bfs
int bfs()
{
    memset(dep, -1, sizeof(dep));
    dep[s] = 0;
    queue<int> q;
    q.push(s);
    while(!q.empty())
    {
        int u = q.front();
        q.pop();
        for(int i=head[u]; ~i; i=edge[i].ne)
        {
            int v = edge[i].to;
            if(dep[v]==-1 && edge[i].val>0)
            {
                dep[v] = dep[u]+1;
                q.push(v);
            }
        }
```

```
52      }
53      return (dep[e] != -1); //若dep[e]==-1则表示没有可以到达e的增广路了，算法结束。
54  }
55
56  // 当前顶点u，当前流量flow
57  // 初始时dfs(s, inf)
58  int dfs(int u, int flow)
59  {
60      if(u == e)return flow;
61      for(int i=head[u]; ~i; i=edge[i].ne)
62      {
63          int v = edge[i].to;
64          if(dep[v]==dep[u]+1 && edge[i].val)
65          {
66              int a = dfs(v, min(flow, edge[i].val));
67              if(a>0) //若找到增广路
68              {
69                  edge[i].val -= a;
70                  edge[i^1].val += a;
71                  return a;
72              }
73          }
74      }
75      return 0;
76  }
77
78  ll dinic()
79  {
80      ll ans = 0;
81      while(bfs())
82      {
83          int a = dfs(s, (1<<30));
84          ans += a;
85      }
86      return ans;
87  }
88
89  int main()
90  {
91      scanf("%d%d%d%d", &n, &m, &s, &e);
92      init();
93      for(int i=1;i<=m;i++)
94      {
95          int u, v, w;
96          scanf("%d%d%d", &u, &v, &w);
97          addEdge(u, v, w);
98          addEdge(v, u, 0); //反边
99      }
100     printf("%lld\n", dinic());
101     return 0;
102 }
```

1.7.2 Dinic 优化

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   typedef long long ll;
4   typedef pair<int, int> P;
5   const int maxn = 1e6+5;
6   const int inf = 0x3f3f3f3f;
7   const int mod = 1e9+7;
8
9   struct ed
10  {
11      int to, val, ne;
```

```
}edge[maxn<<1];
int head[maxn],dep[maxn], cur[maxn];
int n, m, s, e, tot;

void init()
{
    tot = -1;
    memset(head, -1, sizeof(head));
}

void addEdge(int u, int v, int val)
{
    edge[++tot].to = v;
    edge[tot].val = val;
    edge[tot].ne = head[u];
    head[u] = tot;
}

int bfs()
{
    memset(dep, -1, sizeof(dep));
    dep[s] = 0;
    queue<int> q;
    q.push(s);
    while(!q.empty())
    {
        int u = q.front();
        q.pop();
        for(int i=head[u]; ~i; i=edge[i].ne)
        {
            int v = edge[i].to;
            if(dep[v]==-1 && edge[i].val>0)
            {
                dep[v] = dep[u]+1;
                q.push(v);
            }
        }
    }
    return (dep[e] != -1);
}

int dfs(int u, int flow)
{
    if(u == e)return flow;
    // rflow用于多路增广, 表示流入到顶点u的剩余未流出的流量
    int rflow = flow;
    // 当前弧优化, 通过引用, 可以改变cur[i]的值, 使得下次遍历到顶点u时, 会直接从上次增广的边开始遍历
    for(int& i=cur[u]; ~i; i=edge[i].ne)
    {
        int v = edge[i].to;
        if(dep[v]==dep[u]+1 && edge[i].val)
        {
            int a = dfs(v, min(rflow, edge[i].val));
            edge[i].val -= a;
            edge[i^1].val += a;
            rflow -= a; // 剩余流量要减少
            if(rflow<=0)break; // 若没有剩余流量了, 就break
        }
    }
    // 若没有一丝流量流出, 则表示通过顶点u已经无法增广了, 于是炸点, dep可以设置为任何无意义值
    if(rflow == flow)
    dep[u] = -2;
    return flow - rflow; // 返回流出的流量
}
```

```
76
77 ll dinic()
78 {
79     ll ans = 0;
80     while(bfs())
81     {
82         // 新一轮dfs之前要对cur进行初始化
83         for(int i=1;i<=n;i++)cur[i] = head[i];
84         int a = dfs(s, (1<<30));
85         ans += a;
86     }
87     return ans;
88 }
89
90 int main()
91 {
92     scanf("%d%d%d%d", &n, &m, &s, &e);
93     init();
94     for(int i=1;i<=m;i++)
95     {
96         int u, v, w;
97         scanf("%d%d%d", &u, &v, &w);
98         addEdge(u, v, w);
99         addEdge(v, u, 0);
100    }
101    printf("%lld\n", dinic());
102    return 0;
103 }
```

1.8 费用流

1.8.1 最大费用最大流

```
1  const int maxn = 1e4 + 10;
2
3  struct Edge {
4      ll to, ne, lim, cos;
5  }e[maxn];
6  ll a[maxn], b[maxn], c[maxn];
7  ll head[maxn], dis[maxn], flow[maxn], pre[maxn], vis[maxn];
8  ll n, m, M, s, s1, t, tot;
9  void init() {
10     memset(head, -1, sizeof(head));
11     tot = 0;
12 }
13 // u->v, 流量, 花费
14 void add(int u, int v, int lim, int cos) {
15     e[tot].to = v;
16     e[tot].ne = head[u];
17     e[tot].lim = lim;
18     e[tot].cos = cos;
19     head[u] = tot++;
20
21     e[tot].to = u;
22     e[tot].ne = head[v];
23     e[tot].lim = 0;
24     e[tot].cos = -cos;
25     head[v] = tot++;
26 }
27 ll bfs() {
28     queue<ll> q;
29     q.push(s);
30     for (ll i = 0; i <= t + 10; i++)
31         dis[i] = -INF, flow[i] = INF, vis[i] = 0;
32     vis[s] = 1, dis[s] = 0;
```

```
    while (!q.empty()) {
        ll u = q.front();
        q.pop();
        for (ll i = head[u]; ~i; i = e[i].ne) {
            ll v = e[i].to, w = e[i].cos, lim = e[i].lim;
            if (lim && (dis[u] + w > dis[v])) {
                dis[v] = dis[u] + w;
                flow[v] = min(flow[v], lim);
                pre[v] = i;
                if (!vis[v]) {
                    vis[v] = 1;
                    q.push(v);
                }
            }
        }
        vis[u] = 0;
    }
    return flow[t] == INF ? 0 : 1;
}
void solve() {
    scanf("%lld%lld%lld", &n, &m, &M);
    /* 建图
    for (ll i = 1; i <= n; i++) scanf("%lld", &a[i]);
    for (ll i = 1; i <= n; i++) scanf("%lld", &b[i]);
    for (ll i = 1; i <= n; i++) scanf("%lld", &c[i]);
    s = 0, t = 4 * n + 4, s1 = 4 * n + 1;
    init();
    add(s, s1, m, 0);
    for (ll i = 1; i <= n; i++) add(s1, i, 1, 0);
    for (ll i = 1; i <= n; i++) {
        for (ll j = 1; j <= n; j++)
            add(i, n + j, 1, (b[i] + a[j]) * (b[i] ^ a[j]) % M);
    }
    for (ll i = 1; i <= n; i++) add(n + i, 2 * n + i, 1, 0);
    for (ll i = 1; i <= n; i++) {
        for (ll j = 1; j <= n; j++)
            add(2 * n + i, 3 * n + j, 1, (a[i] + c[j]) * (a[i] ^ c[j]) % M);
    }
    for (ll i = 1; i <= n; i++) add(3 * n + i, t, 1, 0);
    */
    ll ans = 0;
    while (bfs()) {
        ans += dis[t] * flow[t];
        ll v = t;
        while (1) {
            ll u = e[pre[v]^1].to;
            e[pre[v]].lim -= flow[t];
            e[pre[v]^1].lim += flow[t];
            v = u;
            if (v == s) break;
        }
    }
    printf("%lld\n", ans);
}
```

1.8.2 zkw 费用流

```
bool vis[200001];
int dist[200001];
//解释一下各数组的含义：vis两个用处：spfa里的访问标记，增广时候的访问标记，dist是每个点的距离标号
int n, m, s, t, ans = 0;
//s是起点，t是终点，ans是费用答案
int nedge = -1, p[200001], c[200001], cc[200001], nex[200001], head[200001];
```

```
7  //这里是边表，解释一下各数组的含义：p[i]表示以某一点出发的编号为i的边对应点，c表示编号为i的边的流量，cc表示编号为i的边的费用，nex和head不说了吧。。。
8  inline void addedge(int x, int y, int z, int zz) {
9      p[++nedge] = y;
10     c[nedge] = z;
11     cc[nedge] = zz;
12     nex[nedge] = head[x];
13     head[x] = nedge;
14 }
15 //建边（数组模拟边表倒挂）
16 inline bool spfa(int s, int t) {
17     memset(vis, 0, sizeof(vis));
18     for (int i = 0; i <= n; i++) dist[i]=1e9;
19     dist[t]=0; vis[t]=1;
20 //首先SPFA我们维护距离标号的时候要倒着跑，这样可以维护出到终点的最短路径
21     deque<int> q; q.push_back(t);
22 //使用了SPFA的SLF优化（SLF可以自行百度或Google）
23     while (!q.empty()) {
24         int now = q.front(); q.pop_front();
25         for (int k = head[now]; k > -1; k = nex[k])
26             if (c[k ^ 1] && dist[p[k]] > dist[now] - cc[k]) {
27 //首先c[k^1]是为什么呢，因为我们要保证正流，但是SPFA是倒着跑的，所以说我们要求c[k]的对应反向边是正的，这样保证走的方向是正确的
28                 dist[p[k]] = dist[now] - cc[k];
29 //因为已经是倒着的了，我们也可以很清楚明白地知道建边的时候反向边的边权是负的，所以减一下就对了（负负得正）
30                 if (!vis[p[k]]) {
31                     vis[p[k]] = 1;
32                     if (!q.empty() && dist[p[k]] < dist[q.front()])
33                         q.push_front(p[k]);
34                     else q.push_back(p[k]);
35 //SLF优化
36                 }
37             }
38         vis[now] = 0;
39     }
40     return dist[s] < 1e9;
41 //判断起点终点是否连通
42 }
43 inline int dfs(int x, int low) {
44 //这里就是进行增广了
45     if (x == t) { vis[t] = 1; return low; }
46     int used = 0, a; vis[x] = 1;
47 //这边是不是和dinic很像啊
48     for (int k = head[x]; k > -1; k = nex[k])
49         if (!vis[p[k]] && c[k] && dist[x] - cc[k] == dist[p[k]]) {
50 //这个条件就表示这条边可以进行增广
51             a = dfs(p[k], min(c[k], low - used));
52             if (a) ans += a * cc[k], c[k] -= a, c[k ^ 1] += a, used += a;
53 //累加答案，加流等操作都在这了
54             if (used == low) break;
55         }
56     return used;
57 }
58 inline int costflow(){
59     int flow=0;
60     while(spfa(s,t)){
61 //判断起点终点是否连通，不连通说明满流，做完了退出
62         vis[t]=1;
63         while(vis[t]){
64             memset(vis,0,sizeof vis);
65             flow+=dfs(s,1e9);
66 //一直增广直到走不到为止（这样也可以省时间哦）
67         }
68     }
69     return flow;//这里返回的是最大流，费用的答案在ans里
```

```
}
int main() {
    memset(nex, -1, sizeof(nex));
    memset(head, -1, sizeof(head));
    scanf("%d%d%d%d", &n, &m, &s, &t);
    for (int i = 1; i <= m; i++) {
        int x, y, z, zz;
        scanf("%d%d%d%d", &x, &y, &z, &zz);
        addedge(x, y, z, zz);
        addedge(y, x, 0, -zz);
    }
    printf("%d ",costflow());
    printf("%d",ans);
    return 0;
}
```

1.8.3 EK

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 123456789;
struct node {
    int to;//这条边的终点
    int cap;//这条边的流量（是指当前还能流的最大流量，而不是不变量）
    int coc;//coc存反向边。由于我使用了vector，所以coc记录的是data[x][i]中的i。具体怎么计算反向边下面说。
    int cost;//cost存一条边的费用
};//使用一个结构体存储所有的边以及反向边，所有边都是无向边。
vector<node> data[50005];//data用来存图
int dx[50005], pre1[50005], pre2[50005], incf[50005];
int n, m, s, t, ans, maxf;
bool inq[50005];
// dx，inq的意义和spfa中的dist，inq（或vis）相同，pre1，pre2，incf的用处下面会说。
// n,m,s,t如题目所示，ans是最小费用，maxf是最大流。
void add(int u, int v, int w, int f) {//加一条边的函数，由u到v连一条流为w，花费为f的边
    data[u].push_back((node){v, w, data[v].size(), f});
    data[v].push_back((node){u, 0, data[u].size() - 1, -f});//反向边的流要置为0。因为如果你设成正的，这条路就有可能由终点向起点流，在另一条路上达到最大流，更小费用。这显然不合法。
}
bool spfa() {
    fill(dx + 1, dx + n + 1, maxn);//初始化，一定别忘了。
    queue<int> q;
    memset(inq, 0, sizeof(inq));
    q.push(s);
    dx[s] = 0;
    inq[s] = 1;//以上这些变量，玩转spfa的你一定看起来很熟悉QAQ
    incf[s] = maxn;//那么incf呢？他记录的是一条增广路的最小流量。incf[i]代表当前增广路到i为止的最小流量，incf[t]为整条路最小流量。
    //如何理解呢，我们考虑短板效应，最大容量取决于最短木板。增广路亦然，它的最大流量取决于增广路上流量最小的边。
    while (!q.empty()) {
        int now = q.front();
        inq[now] = 0;
        q.pop();//是不是超熟悉的感觉QAQ（只说一下和spfa不一样的代码的意思）
        for (int i = 0; i < data[now].size(); i++) {
            node &tmp = data[now][i];
            if (tmp.cap > 0 && tmp.cost + dx[now] < dx[tmp.to]) {//有流量才能松弛~
                dx[tmp.to] = dx[now] + tmp.cost;//和spfa一样
                incf[tmp.to] = min(incf[now], tmp.cap);//更新增广路的“短板”。
                pre1[tmp.to] = now;//pre1[i]是这次增广路的i点是由哪个点流过来的。
                pre2[tmp.to] = i;//pre2[i]是这次增广路的i点是由pre1[i]的编号为多少的边流过来的。
                if(!inq[tmp.to]) {//QAQ spfa，spfa你还活着！
```

```
41                      inq[tmp.to] = 1;
42                      q.push(tmp.to);
43                  }
44              }
45          }
46      }
47      return dx[t] != maxn;//如果dx[t]被更过了，意味着一定不是最大流！返回1，找增广路去！
48  }
49  void update() {//更新答案和图程度的能力。
50      int x = t;//首先把当前点置为终点，我们沿着终点由增广路反向走到起点~
51      while (x != s) {//x到s的时候停止~
52          int y = pre1[x];//使用我们先前维护的数组来反向走增广路。
53          int i = pre2[x];
54          data[y][i].cap -= incf[t];//所有增广路上的边一律减去可扩最大容量！
55          data[x][data[y][i].coc].cap += incf[t];//反向边，一律增加这个容量。
56          //如果说为什么要弄反向边，我个人的理解是：增广一条增广路费用最优，然而最大流量未必。
57          //如果我们反向建边，可以由终点流向起点，这样其实意味着，当年这条边选择的流减少一点。
58          //不过我们未必要理性理解这个事，既然都建了反向边了，那就和正向边反着来呗~
59          x = y;//迭代一下
60      }
61      maxf += incf[t];//更新最大流
62      ans += dx[t] * incf[t];//更新费用
63  }
64  void EK() {
65      while (spfa()) {
66          update();
67      }
68  }
69  int main() {
70      cin >> n >> m >> s >> t;
71      for (int i = 1; i <= m; i++) {
72          int u, v, w, f;
73          cin >> u >> v >> w >> f;
74          add(u, v, w, f);//按照题意建图
75      }
76      EK();//最小费用流主体
77      cout << maxf << " " << ans << endl;
78      return 0;
79  }
```

1.9 二分图匹配

1.9.1 匈牙利算法

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef pair<int, int> P;
5  const int maxn = 1e3+5;
6  const int inf = 0x3f3f3f3f;
7  const int mod = 1e9+7;
8
9
10 int mp[maxn][maxn];
11 int use[maxn], link[maxn];
12 int n, m, e, ans;
13
14 int found(int u)
15 {
16     for(int i=1;i<=m;i++)
17     {
18         if(!use[i] && mp[u][i])
```

```
        {
            use[i] = 1;
            if(!link[i] || found(link[i]))
            {
                link[i] = u;
                return 1;
            }
        }
    }
    return 0;
}

int main()
{
    scanf("%d%d%d", &n, &m, &e);
    for(int i=1;i<=e;i++)
    {
        int u, v;
        scanf("%d%d", &u, &v);
        if(u<=n && v<=m)
        {
            mp[u][v] = 1;
        }
    }
    for(int i=1;i<=n;i++)
    {
        memset(use, 0, sizeof(use));
        if(found(i))ans++;
    }
    printf("%d\n", ans);
    return 0;
}
```

1.10 最小生成树

1.10.1 Prim

```
const int MAX=10000007;
int dis[5002],map[5002][5002],mark[5002];
int prim(int n)
{
    for(int i=1;i<=n;i++)  //初始化每个点到生成树中点的距离
    {
        dis[i]=map[1][i];
        mark[i]=0;
    }
    dis[1]=0;
    mark[1]=1; //1这个点加入生成树中。
    int sum=0;
    for(int i=1;i<n;i++) //枚举n-1条边
    {
        int sta=-1,Min=MAX;
        for(int j=1;j<=n;j++)  //找不在生成树中的点中距离生成树中的点长度最小的
        {
            if(!mark[j]&&dis[j]<Min)
            {
                Min=dis[j];
                sta=j;
            }
        }
        if(sta==-1) return -1; //没找到可以可以联通的路
        mark[sta]=1;   //新找到的点加入生成树
        sum+=Min;
        for(int j=1;j<=n;j++)  //更新树外的点到树中的点的距离
        {
```

```
29              if(!mark[j]&&dis[j]>map[sta][j])
30              dis[j]=map[sta][j];
31          }
32      }
33      return sum;
34  }
35
36  int main()
37  {
38      int n,m;
39      cin>>n>>m;
40      for(int i=1;i<=n;i++)
41      {
42          for(int j=1;j<=n;j++)
43          {
44              map[i][j]=MAX;
45          }
46      }
47      for(int i=1;i<=m;i++)
48      {
49          int a,b,c;
50          cin>>a>>b>>c;
51          if(c<map[a][b])
52          {
53              map[a][b]=c;
54              map[b][a]=c;
55          }
56      }
57      int ans = prim(n);
58      if(ans==-1)
59          cout<<"orz"<<endl;
60      else
61          cout<<ans<<endl;
62      return 0;
63  }
```

1.10.2 kruskal

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define N 5005
4  int father[N];
5
6  int find(int x)
7  {
8      int k = x;
9      while(father[k]!=k)
10     {
11         k = father[k];
12     }
13     while(father[x]!=x)
14     {
15         int temp = x;
16         x = father[x];
17         father[temp] = k;
18     }
19     return k;
20 }
21
22 void join(int a, int b)
23 {
24     int f1, f2;
25     f1 = find(a);
26     f2 = find(b);
27     father[f1] = f2;
```

```
}

struct edge
{
    int node1, node2;
    int cost;
};

vector<edge> edges;

bool cmp(edge a, edge b)
{
    return a.cost > b.cost;
}

int kruskal(int n)
{
    sort(edges.begin(), edges.end(), cmp);
    for(int i=1;i<=n;i++)
        father[i] = i;
    int sum=0;
    while(n!=1 && !edges.empty())
    {
        edge temp = edges[edges.size()-1];
        edges.pop_back();
        if(find(temp.node1)!=find(temp.node2))
        {
            sum += temp.cost;
            n--;
            join(temp.node1, temp.node2);
        }
    }
    if(n!=1 && edges.empty())
        sum = -1;
    return sum;
}

int main()
{
    int n,m;
    int result;
    cin>>n>>m;
    for(int i=1;i<=m;i++)
    {
        int a,b,c;
        cin>>a>>b>>c;
        edge t;
        t.node1=a;t.node2=b;t.cost=c;
        edges.push_back(t);
    }
    result = kruskal(n);
    if(result == -1)
        cout<<"orz"<<endl;
    else
        cout<<result<<endl;
    return 0;
}
```

1.11 次小生成树-POJ1679

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
#define inf 0x3f3f3f3f
typedef pair<int, int> P;
```

```
6  const int maxn = 110;
7  const ll mod = 1e9+7;
8
9  int n, m;
10 int g[maxn][maxn];
11 int d[maxn], v[maxn], maxd[maxn][maxn], pre[maxn], mst[maxn][maxn];
12 int ans = 0;
13
14 void prim()
15 {
16     for(int i=1;i<=n;i++)
17     {
18         v[i] = 0; d[i] = inf; pre[i] = 1;
19     }
20     memset(maxd, 0, sizeof(maxd));
21     memset(mst, 0, sizeof(mst));
22     ans = 0;
23     priority_queue<P, vector<P>, greater<P> > q;
24     d[1] = 0; q.push(P(0, 1));
25     while(!q.empty())
26     {
27         P p = q.top(); q.pop();
28         int u = p.second;
29         if(v[u]) continue;
30         v[u] = 1; ans += d[u];
31         mst[pre[u]][u] = mst[u][pre[u]] = 1;
32         for(int i=1; i<=n;i++)
33         {
34             if(v[i] && g[u][i] < inf)
35                 maxd[u][i] = maxd[i][u] = max(maxd[pre[u]][u], d[u]);
36             if(d[i] > g[u][i])
37             {
38                 d[i] = g[u][i];
39                 pre[i] = u;
40                 q.push(P(d[i], i));
41             }
42         }
43     }
44 }
45
46 int main()
47 {
48     int t;
49     cin>>t;
50     while(t--)
51     {
52         memset(g, inf, sizeof(g));
53         cin>>n>>m;
54         while(m--)
55         {
56             int a, b, c;
57             cin>>a>>b>>c;
58             g[a][b] = g[b][a] = c;
59         }
60         prim();
61         int flag = 0;
62         for(int i=1;i<=n&&!flag;i++)
63         {
64             for(int j=1;j<=n;j++)
65             {
66                 if(mst[i][j] || g[i][j]==inf)continue;
67                 if(g[i][j] == maxd[i][j])
68                 {
69                     flag = 1;
70                     break;
71                 }
```

```
72              }
73          }
74          if(flag) cout<<"Not Unique!"<<endl;
75          else cout<<ans<<endl;
76      }
77      return 0;
78  }
```

1.12 拓扑排序

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   const int maxn=30;
4
5   int head[maxn],ip,indegree[maxn];
6   int n,m,seq[maxn];
7
8   struct note
9   {
10      int v,next;
11  }edge[maxn*maxn];
12
13  void init()
14  {
15      memset(head,-1,sizeof(head));
16      ip=0;
17  }
18
19  void addedge(int u,int v)
20  {
21      edge[ip].v=v,edge[ip].next=head[u],head[u]=ip++;
22  }
23
24  int topo()
25  {
26      queue<int>q;
27      int indeg[maxn];
28      for(int i=0; i<n; i++)
29      {
30          indeg[i]=indegree[i];
31          if(indeg[i]==0)
32          q.push(i);
33      }
34      int k=0;
35      bool res=false;
36      while(!q.empty())
37      {
38          if(q.size()!=1)res=true;
39          int u=q.front();
40          q.pop();
41          seq[k++]=u;
42          for(int i=head[u]; i!=-1; i=edge[i].next)
43          {
44              int v=edge[i].v;
45              indeg[v]--;
46              if(indeg[v]==0)
47                  q.push(v);
48          }
49      }
50      if(k<n)return -1;// no
51      if(res)return 0;// more
52      return 1; // only
53  }
```

1.13 Floyd 找最小环

```
const int INF = 0x3f3f3f3f;
const int MAXN = 110;

int n, m;                //  n:节点个数, m:边的个数
int g[MAXN][MAXN];       //  无向图
int dist[MAXN][MAXN];    //  最短路径
int r[MAXN][MAXN];       //  r[i][j]: i到j的最短路径的第一步
int out[MAXN], ct;       //  记录最小环

int solve(int i, int j, int k)
{   //  记录最小环
    ct = 0;
    while (j != i)
    {
        out[ct++] = j;
        j = r[i][j];
    }
    out[ct++] = i;
    out[ct++] = k;
    return 0;
}

int main()
{
    while (scanf("%d%d", &n, &m) != EOF)
    {
        int i, j, k;
        for (i = 0; i < n; i++)
        {
            for (j = 0; j < n; j++)
            {
                g[i][j] = INF;
                r[i][j] = i;
            }
        }
        for (i = 0; i < m; i++)
        {
            int x, y, l;
            scanf("%d%d%d", &x, &y, &l);
            --x;
            --y;
            if (l < g[x][y])
            {
                g[x][y] = g[y][x] = l;
            }
        }
        memmove(dist, g, sizeof(dist));
        int Min = INF;              //  最小环
        for (k = 0; k < n; k++)
        {                           //  Floyd
            for (i = 0; i < k; i++) //  一个环中的最大结点为k(编号最大)
            {
                if (g[k][i] < INF)
                {
                    for (j = i + 1; j < k; j++)
                    {
                        if (dist[i][j] < INF && g[k][j] < INF && Min > dist[i][j]
                          + g[k][i] + g[k][j])
                        {
                            Min = dist[i][j] + g[k][i] + g[k][j];
                            solve(i, j, k);     //  记录最小环
                        }
                    }
                }
```

```
64              }
65              for (i = 0; i < n; i++)
66              {
67                  if (dist[i][k] < INF)
68                  {
69                      for (j = 0; j < n; j++)
70                      {
71                          if (dist[k][j] < INF && dist[i][j] > dist[i][k]+dist[k][j
                              ])
72                          {
73                              dist[i][j] = dist[i][k] + dist[k][j];
74                              r[i][j] = r[k][j];
75                          }
76                      }
77                  }
78              }
79          }
80          if (Min < INF)
81          {
82              for (ct--; ct >= 0; ct--)
83              {
84                  printf("%d", out[ct] + 1);
85                  if (ct)
86                  {
87                      printf(" ");
88                  }
89              }
90          }
91          else
92          {
93              printf("No solution.");
94          }
95          printf("\n");
96      }
97      return 0;
98  }
```

1.14 生成树计数 (基尔霍夫矩阵) BZOJ1002

```
1  // f[i] = 3*f[i-1] - f[i-2] + 2
2  // 需要高精度
3  #include<bits/stdc++.h>
4  using namespace std;
5  typedef long long ll;
6  typedef unsigned long long ull;
7  typedef pair<int, int> P;
8  const int maxn = 1e6+5;
9  const int inf = 0x3f3f3f3f;
10 const int mod = 1e9+7;
11 const double PI = acos(-1);
12
13
14 struct node
15 {
16     int a[101], len;
17 };
18 node f[101];
19 node mul(node a, int k)
20 {
21     for(int i=1;i<=a.len;i++)
22         a.a[i]*=k;
23     for(int i=1;i<=a.len;i++)
24     {
25         a.a[i+1]+=a.a[i]/10;
26         a.a[i]%=10;
```

```
    }
    while(a.a[a.len+1]!=0)a.len++;
    return a;
}

node sub(node a, node b)
{
    a.a[1] += 2;
    int j = 1;
    while(a.a[j]>=10)
    {
        a.a[j]%=10;
        a.a[j+1]++;
        j++;
    }
    for(int i=1;i<=a.len;i++)
    {
        a.a[i] -= b.a[i];
        if(a.a[i]<0)
        {
            a.a[i]+=10;
            a.a[i+1]--;
        }
    }
    while(a.a[a.len]==0)a.len--;
    return a;
}

int main()
{
    f[1].a[1] = 1; f[2].a[1] = 5;
    f[1].len = f[2].len = 1;
    int n;
    scanf("%d", &n);
    for(int i=3;i<=n;i++)
        f[i] = sub(mul(f[i-1], 3), f[i-2]);
    for(int i=f[n].len;i>0;i--)
        printf("%d", f[n].a[i]);
    printf("\n");
    return 0;
}
```

1.15 悬线法 BZOJ3039

```
// 悬线法求最大子矩阵
// 给定n*m的矩阵，含F和R，求最大的全F矩阵的面积×3

#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int, int> P;
const int maxn = 1e3+5;
const int inf = 0x3f3f3f3f;
const ll INF = 0x3f3f3f3f3f3f3f3f;
const int mod = 1e9+7;
const double PI = acos(-1);


char mp[maxn][maxn];
int l[maxn][maxn], r[maxn][maxn], h[maxn][maxn];
int n, m;

int main()
{
```

```
    ios::sync_with_stdio(0);
    cin>>n>>m;
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=m;j++)
        {
            cin>>mp[i][j];
        }
    }
    for(int i=1;i<=n;i++)
    {
        int tmp = 1;
        for(int j=1;j<=m;j++)
        {
            if(mp[i][j] == 'F')
            {
                l[i][j] = tmp;
                if(mp[i-1][j]=='F')
                    l[i][j] = max(l[i][j], l[i-1][j]);
            }
            else tmp = j+1;
        }
    }
    for(int i=1;i<=n;i++)
    {
        int tmp = m;
        for(int j=m;j>=1;j--)
        {
            if(mp[i][j]=='F')
            {
                r[i][j] = tmp;
                if(mp[i-1][j] == 'F')
                    r[i][j] = min(r[i][j], r[i-1][j]);
            }
            else tmp = j-1;
        }
    }
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=m;j++)
        {
            if(mp[i][j] == 'F')
            h[i][j] = h[i-1][j]+1;
        }
    }
    int ans = 0;
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=m;j++)
        {
            if(mp[i][j]=='F')
                ans = max(ans, h[i][j]*(r[i][j]-l[i][j]+1));
        }
    }
    cout<<3*ans<<endl;
    return 0;
}
```

1.16 欧拉回路

```
int head[N];
struct edgenode{
    int to;
    int next;
}tu[N];
```

```
int ans[maxn];///maxn是边的最大数量
bool vis[maxn];
int bj=0;
void dfs(int now)
{
    for(int i=head[now];i!=-1;i=tu[i].next)
    if(!vis[i])
    {
        vis[i]=1;
        vis[i^1]=1;///这里是求欧拉回路，这一题用不着写这句，因为可以走双向
        dfs(tu[i].to);
        ans[bj++]=i;///等于i是记录边，等于tu[i].to是记录点
    }
}

// 或者

struct Edge
{
    int v;
    bool vis;
    Edge(){}
    Edge(int v,bool vis):v(v),vis(vis){}
}edges[maxm*2];
vector<Edge>G[maxn];
vector<int> ans;
void init()
{
    for (int i = 0;i<=n;i++)
        G[i].clear();
}
void euler(int u)
{
    for (int i = 0;i<G[u].size();i++)
    {
        Edge &e = G[u][i];
        if (!e.vis)
        {
            e.vis=1;
            euler(e.v);
        }
    }
    ans.push_back(u);
    //printf("%d\n",u);
}
```

2 数据结构

2.1 并查集

```
int pre[maxn];

int Find(int x)
{
    int p,tmp;
    p=x;
    while(x!=pre[x])
        x=pre[x];
    while(p!=x)
    {
        tmp=pre[x];
        pre[x]=x;
        p=tmp;
    }
```

```
    return x;
}

void join(int x,int y)
{
    int fx=Find(x);
    int fy=Find(y);
    if(fx!=fy)
        pre[fx]=fy;
}
```

2.2 带权并查集

2.2.1 BZOJ1202

```
\\ 给出n个月份，每个月份都有营业额
\\ m个判断，表示区间[l, r]的和为w
\\ 判断是否有矛盾
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int, int> P;
const int maxn = 1e3+5;
const int inf = 0x3f3f3f3f;
const ll INF = 0x3f3f3f3f3f3f3f3f;
const int mod = 1e9+7;
const double PI = acos(-1);



int n, m;
int fa[105], d[105];
int find(int x)
{
    if(fa[x]==x)return x;
    int xx = find(fa[x]);
    d[x] += d[fa[x]];
    return fa[x] = xx;
}
int main()
{
    int t;
    scanf("%d", &t);
    while(t--)
    {
        scanf("%d%d", &n, &m);
        memset(d, 0, sizeof(d));
        for(int i=0;i<=n;i++) fa[i] = i;
        int f = 1;
        while(m--)
        {
            int s, t ,w;
            scanf("%d%d%d", &s, &t, &w);
            s--;
            int fas = find(s), fat = find(t);
            if(fas != fat)
            {
                fa[fat] = fas;
                d[fat] = d[s]+w-d[t];
            }
            else if(d[t]-d[s]!=w)f=0;
        }
        if(f)printf("true\n");
        else printf("false\n");
```

```
51      }
52      return 0;
53  }
```

2.3 ST 表

2.3.1 区间最大值

```
1  int st[maxn][20];
2  void st_init()
3  {
4      for(int i=1;i<=n;i++) st[i][0]=a[i]; // 长度为1的区间最小值党委就为自身
5      // 预处理从i开始，长度为2^j的区间
6      for(int j=1;(1<<j)<=n;j++)
7          for(int i=1;i+(1<<j)-1<=n;i++)
8              st[i][j]=max(st[i][j-1],st[i+(1<<(j-1))][j-1]);
9  }
10
11 int query(int l,int r)
12 {
13     int k=log2(r-l+1);
14     return min(st[l][k],st[r-(1<<k)+1][k]);
15 }
```

2.3.2 维护父节点

```
1  int fa[maxn][21];
2  void init() {
3      // 要初始化fa[i][0]，即父节点，这里未初始化
4      // fa[i][j]表示从i向上走2^j步所能到的的结点
5      for (int i = 1; i <= 20; i++)
6          for (int j = 1; j <= n; j++)
7              fa[j][i] = fa[fa[j][i - 1]][i - 1];
8  }
```

2.4 树状数组

```
1  /*
2  *  INIT: ar[]置为0;
3  *  CALL: add(i, v): 将i点的值加v; sum(i): 求[1, i]的和;
4  */
5  #define typev int   //  type of res
6  const int N = 1010;
7  typev ar[N];        //  index: 1 ~ N
8  int lowb(int t)
9  {
10     return t & (-t);
11 }
12
13 void add(int i, typev v)
14 {
15     for (; i < N; ar[i] += v, i += lowb(i));
16     return ;
17 }
18
19 typev sum(int i)
20 {
21     typev s = 0;
22     for (; i > 0; s += ar[i], i -= lowb(i));
23     return s;
24 }
```

2.5 线段树

2.5.1 单点修改 + 区间求和 HDU1166

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

int num[50005];
ll tree[4 * 50000 + 5];

void build(int p, int l, int r)
{
    if(l == r)
    {
        tree[p] = num[l];
        return ;
    }
    else
    {
        int mid = (r+l) >> 1;
        build(p<<1, l, mid);
        build(p<<1|1, mid+1, r);
        tree[p] = tree[p<<1] + tree[p<<1|1];
    }
}

void add(int p, int l, int r, int ind, int v)
{
    if(l == r)
    {
        tree[p] += v;
        return ;
    }
    else
    {
        int mid = (r+l) >> 1;
        if(ind <= mid) add(p<<1, l, mid, ind, v);
        else add(p<<1|1, mid+1, r, ind, v);
        tree[p] = tree[p<<1] + tree[p<<1|1];
    }
}

ll query(int p, int l, int r, int x, int y)
{
    if(x <= l && r <= y)
    {
        return tree[p];
    }
    else
    {
        int mid = (l+r) >> 1;
        ll ans = 0;
        if(x <= mid)
            ans += query(p<<1, l, mid, x, y);
        if(mid < y)
            ans += query(p<<1|1, mid+1, r, x, y);
        return ans;
    }
}

int main()
{
    int t;
    scanf("%d", &t);
    for(int i=1;i<=t;i++)
```

```
63      {
64          int n;
65          scanf("%d", &n);
66          for(int j=1;j<=n;j++)
67              scanf("%d", &num[j]);
68          build(1, 1, n);
69          string s;
70          printf("Case %d:\n", i);
71          while(cin>>s && s[0] != 'E')
72          {
73              if(s[0] == 'Q')
74              {
75                  int x, y;
76                  scanf("%d%d", &x, &y);
77                  printf("%lld\n", query(1, 1, n, x, y));
78              }
79              else if(s[0] == 'A')
80              {
81                  int x, y;
82                  scanf("%d%d", &x, &y);
83                  add(1, 1, n, x, y);
84              }
85              else
86              {
87                  int x, y;
88                  scanf("%d%d", &x, &y);
89                  add(1, 1, n, x, -y);
90              }
91          }
92      }
93      return 0;
94  }
```

2.5.2 区间修改 + 区间求和 POJ3465

```
1  #include<iostream>
2  using namespace std;
3  typedef long long ll;
4  ll num[100005];
5  ll tree[4 * 100000 + 5];
6  ll lazy[4 * 100000 + 5]={0};
7  int n,m;
8  void build(int p, int l, int r)
9  {
10     if(l == r)
11     tree[p] = num[l];
12     else
13     {
14         int mid = (l+r) >> 1;
15         build(p<<1, l, mid);
16         build(p<<1|1, mid+1, r);
17         tree[p]  = tree[p<<1] + tree[p<<1|1];
18     }
19 }
20
21 void pushdown(int p, int l, int r)
22 {
23     if(lazy[p])
24     {
25         lazy[p<<1] += lazy[p];
26         lazy[p<<1|1] += lazy[p];
27         tree[p<<1] += lazy[p] * (((l+r)>>1) - l + 1);
28         tree[p<<1|1] += lazy[p] * (r - ((l+r)>>1));
29         lazy[p] = 0;
30     }
```

```
}

void add(int p, int l, int r, int x, int y, ll v)
{
    if(x <= l && r <= y)
    {
        lazy[p] += v;
        tree[p] += v * (r-l+1);
        return ;
    }
    else
    {
        int mid = (l+r) >> 1;
        pushdown(p, l, r);
        if(x <= mid)
            add(p<<1, l, mid, x, y, v);
        if(y > mid)
            add(p<<1|1, mid+1, r, x, y, v);
        tree[p] = tree[p<<1] + tree[p<<1|1];
    }
}

ll query(int p, int l, int r, int x, int y)
{
    if(x <= l && r <= y)
    {
        return tree[p];
    }
    else
    {
        int mid = (l+r) >> 1;
        ll ans = 0;
        pushdown(p, l, r);
        if(x <= mid)
            ans += query(p<<1, l, mid, x, y);
        if(y > mid)
            ans += query(p<<1|1, mid+1, r, x, y);
        return ans;
    }
}

int main()
{
    cin>>n>>m;
    for(int i=1;i<=n;i++)
    cin>>num[i];
    build(1, 1, n);
    for(int i=1;i<=m;i++)
    {
        char a;
        cin>>a;
        if(a == 'Q')
        {
            int x, y;
            cin>>x>>y;
            cout<<query(1, 1, n, x, y)<<endl;
        }
        else
        {
            int x, y;
            ll c;
            cin>>x>>y>>c;
            add(1, 1, n, x, y, c);
        }
    }
    return 0;
```

```
}
```

2.5.3 单点修改 + 区间最值 HDU1754

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
int n,m;

int num[200005];
int tree[200000 * 4 + 5];

void build(int p, int l, int r)
{
    if(l == r)
    {
        tree[p] = num[l];
        return ;
    }
    else
    {
        int mid = (l+r) >> 1;
        build(p<<1, l, mid);
        build(p<<1|1, mid+1, r);
        tree[p] = max(tree[p<<1], tree[p<<1|1]);
        return ;
    }
}

void update(int p, int l, int r, int ind, int v)
{
    if(l == r)
    {
        tree[p] = v;
        return ;
    }
    else
    {
        int mid = (l+r) >> 1;
        if(ind <= mid)
            update(p<<1, l, mid, ind, v);
        else
            update(p<<1|1, mid+1, r, ind, v);
        tree[p] = max(tree[p<<1], tree[p<<1|1]);
    }
}

ll query(int p, int l, int r, int x, int y)
{
    if(x <= l && r <= y)
    {
        return tree[p];
    }
    else
    {
        int mid = (l+r) >> 1;
        ll ans = -1;
        if(x <= mid)
            ans = max(ans, query(p<<1, l, mid, x, y));
        if(y > mid)
            ans = max(ans, query(p<<1|1, mid+1, r, x, y));
        return ans;
    }
}

```

```
62 int main()
63 {
64     while(scanf("%d %d", &n, &m) != EOF)
65     {
66         for(int i=1;i<=n;i++)
67         {
68             scanf("%d", &num[i]);
69         }
70         build(1, 1, n);
71         for(int i=1;i<=m;i++)
72         {
73             char a;
74             int x, y;
75             scanf(" %c%d%d", &a, &x, &y);
76             if(a == 'Q')
77             {
78                 printf("%lld\n", query(1, 1, n, x, y));
79             }
80             else
81             {
82                 update(1, 1, n, x, y);
83             }
84         }
85     }
86     return 0;
87 }
```

2.5.4 区间染色 + 统计 + 离散化 POJ2528

```
1  #include<iostream>
2  #include<algorithm>
3  #include<string.h>
4  using namespace std;
5  typedef long long ll;
6
7  const int maxn = 20000 + 100;
8  int tree[maxn<<4];
9  int li[maxn],ri[maxn];
10 bool vis[maxn];
11 int lisan[maxn*3];
12 int ans = 0;
13
14 void init()
15 {
16     memset(tree, -1, sizeof(tree));
17     memset(vis, 0, sizeof(vis));
18     ans = 0;
19 }
20
21 void pushdown(int p)
22 {
23     tree[p<<1] = tree[p<<1|1] = tree[p];
24     tree[p] = -1;
25 }
26
27 void update(int p, int l, int r, int x, int y, int v)
28 {
29     if(x <= l && r <= y)
30     {
31         tree[p] = v;
32         return ;
33     }
34     if(tree[p]!=-1)
35         pushdown(p);
36     int mid = (l+r)>>1;
```

```
37      if(x <= mid)
38          update(p<<1, l, mid, x, y, v);
39      if(y > mid)
40          update(p<<1|1, mid+1, r, x, y, v);
41      tree[p] = -1;
42  }
43
44  void query(int p, int l, int r)
45  {
46      if(tree[p]!=-1)
47      {
48          if(vis[tree[p]]==0)
49          {
50              vis[tree[p]] = 1;
51              ans++;
52          }
53          return;
54      }
55      if(l==r)return;
56      int mid = (l+r)>>1;
57      query(p<<1, l, mid);
58      query(p<<1|1, mid+1, r);
59  }
60
61  int main()
62  {
63      int t;
64      cin>>t;
65      while(t--)
66      {
67          init();
68          int n;
69          cin>>n;
70          int tot = 0;
71          for(int i=0;i<n;i++)
72          {
73              cin>>li[i]>>ri[i];
74              lisan[tot++] = li[i];
75              lisan[tot++] = ri[i];
76          }
77          sort(lisan, lisan+tot);
78          int m = unique(lisan, lisan+tot) - lisan;
79          int t = m;
80          for(int i=1;i<t;i++)
81          {
82              if(lisan[i]-lisan[i-1]>1)
83              {
84                  lisan[m++] = lisan[i-1]+1;
85              }
86          }
87          sort(lisan, lisan+m);
88          for(int i=0;i<n;i++)
89          {
90              int x,y;
91              x = lower_bound(lisan, lisan+m, li[i]) - lisan;
92              y = lower_bound(lisan, lisan+m, ri[i]) - lisan;
93              update(1, 0, m-1, x, y, i);
94          }
95          query(1, 0, m-1);
96          cout<<ans<<endl;
97      }
98      return 0;
99  }
```

2.5.5 线段树 + 扫描线求矩阵覆盖周长 POJ1177

```
#include<iostream>
#include<vector>
#include<algorithm>
#include<cmath>
using namespace std;
typedef long long ll;

const int maxn = 10005;
vector<int> x;
int getID(int v)
{
    return lower_bound(x.begin(), x.end(), v) - x.begin();
}

struct Segment
{
    int l, r;
    int h;
    int flag;
}segment[maxn];
bool cmp(Segment a, Segment b)
{
    return a.h < b.h;
}

struct Node
{
    int l,r;
    int lr, rr;
    int len;
    int line;
    int s;
}node[maxn<<2];

void build(int p, int l, int r)
{
    node[p].l = l; node[p].r = r;
    node[p].line = node[p].len = node[p].s = 0;
    node[p].lr = node[p].rr = 0;
    if(l==r)return;
    int mid = (l+r)>>1;
    build(p<<1, l, mid);
    build(p<<1|1, mid+1, r);
}

void pushup(int p)
{
    if(node[p].s)
    {
        node[p].line = 1;
        node[p].rr = node[p].lr = 1;
        node[p].len = x[node[p].r+1] - x[node[p].l];
        return;
    }
    else if(node[p].l == node[p].r)
    {
        node[p].lr = node[p].rr = node[p].line = node[p].len = 0;
    }
    else
    {
        node[p].lr = node[p<<1].lr;
        node[p].rr = node[p<<1|1].rr;
        node[p].len = node[p<<1].len + node[p<<1|1].len;
        node[p].line = node[p<<1].line + node[p<<1|1].line - (node[p<<1].rr&&node[
```

```
            p<<1|1].lr);
    }
}

void update(int p, int l, int r, int v)
{
    if(node[p].r < l || node[p].l > r)return;
    if(l <= node[p].l && node[p].r <= r)
    {
        node[p].s += v;
        pushup(p);
        return;
    }
    update(p<<1, l, r, v);
    update(p<<1|1, l, r, v);
    pushup(p);
}

int main()
{
    int n;
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        int x1,x2,y1,y2;
        cin>>x1>>y1>>x2>>y2;
        Segment &s1 = segment[2*i-1];
        Segment &s2 = segment[i<<1];
        s1.l = s2.l = x1;
        s1.r = s2.r = x2;
        s1.h = y1; s2.h = y2;
        s1.flag = 1; s2.flag = -1;
        x.push_back(x1);
        x.push_back(x2);
    }
    sort(segment+1, segment+2*n+1, cmp);

    sort(x.begin(), x.end());
    x.erase(unique(x.begin(), x.end()), x.end());

    build(1, 0, x.size()-1);

    ll ans = 0;
    int last = 0;
    for(int i=1;i<=2*n;i++)
    {
        int l = getID(segment[i].l);
        int r = getID(segment[i].r);
        update(1, l, r-1, segment[i].flag);
        ans += abs(node[1].len - last);
        if(i!=2*n)
            ans += node[1].line * 2 * (segment[i+1].h - segment[i].h);
        last = node[1].len;
    }
    cout<<ans<<endl;
    return 0;
}
```

2.5.6 线段树 + 扫描线求矩阵面积并 HDU1542

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

const int maxn = 210;
```

```
6  int n;
7  vector<double> x;
8  inline int getID(double v)
9  {
10     return lower_bound(x.begin(), x.end(), v) - x.begin();
11 }
12
13 struct Segment
14 {
15     double l, r;
16     double h;
17     int flag;
18 }segment[maxn];
19 bool cmp(Segment a, Segment b)
20 {
21     return a.h < b.h;
22 }
23
24 struct Node
25 {
26     int l, r;
27     int s;
28     double len;
29 }node[maxn<<2];
30
31 void pushup(int p)
32 {
33     if(node[p].s)
34         node[p].len = x[node[p].r+1] - x[node[p].l];
35     else if(node[p].l == node[p].r)
36         node[p].len = 0;
37     else
38         node[p].len = node[p<<1].len + node[p<<1|1].len;
39 }
40
41 void build(int p, int l, int r)
42 {
43     if(l>r)return;
44     node[p].l = l; node[p].r = r;
45     node[p].s = 0; node[p].len = 0;
46     if(l==r) return;
47     int mid = (l+r)>>1;
48     build(p<<1, l, mid);
49     build(p<<1|1, mid+1, r);
50     pushup(p);
51 }
52
53 void update(int p, int l, int r, int v)
54 {
55     if(l>node[p].r || r<node[p].l) return;
56     if(l <= node[p].l && node[p].r <= r)
57     {
58         node[p].s += v;
59         pushup(p);
60         return;
61     }
62     update(p<<1, l, r, v);
63     update(p<<1|1, l, r, v);
64     pushup(p);
65 }
66
67 int main()
68 {
69     int cas = 0;
70     while(scanf("%d", &n) && n)
71     {
```

```
72          x.clear();
73          for(int i=1;i<=n;i++)
74          {
75              double x1,x2,y1,y2;
76              scanf("%lf%lf%lf%lf", &x1, &y1, &x2, &y2);
77              Segment &s1 = segment[2*i-1];
78              Segment &s2 = segment[i<<1];
79              s1.l=s2.l=x1;
80              s1.r=s2.r=x2;
81              s1.h=y1;
82              s2.h=y2;
83              s1.flag=1;
84              s2.flag=-1;
85              x.push_back(x1);
86              x.push_back(x2);
87          }
88          sort(segment+1, segment+2*n+1, cmp);
89
90          sort(x.begin(), x.end());
91          x.erase(unique(x.begin(), x.end()), x.end());
92
93          build(1, 0, x.size()-1);
94          double ans = 0;
95          for(int i=1;i<=2*n;i++)
96          {
97              int l=getID(segment[i].l);
98              int r=getID(segment[i].r);
99              update(1, l, r-1, segment[i].flag);
100             ans+=node[1].len*(segment[i+1].h - segment[i].h);
101         }
102         printf("Test case #%d\n", ++cas);
103         printf("Total explored area: %.2f\n\n", ans);
104     }
105     return 0;
106 }
```

2.5.7 状压 + 线段树维护区间 26 字母种类 CF1234D

```
1  /*
2  * 操作1: 将原pos位置的字母改为字母c
3  * 操作2: 查询区间[l, r]的字母种类数
4  */
5
6  #include<bits/stdc++.h>
7  using namespace std;
8  typedef long long ll;
9  typedef pair<int, int> P;
10 const int maxn = 1e5+5;
11 const int inf = 0x3f3f3f3f;
12 const int mod = 1e9+7;
13
14 char s[maxn];
15 const int N=1e5+5;
16 struct fun{
17     bitset<270>bit;
18 }tree[N<<2],v[N<<2];
19
20
21
22 void sett(int l,int r,int rt){  // 建树
23     if(l==r){
24         tree[rt].bit=0;
25         tree[rt].bit.set(s[l]-'a');
26         return ;
27     }
```

```
28      int mid=(l+r)>>1;
29      sett(l,mid,rt<<1);
30      sett(mid+1,r,rt<<1|1);
31      tree[rt].bit=(tree[rt<<1].bit|tree[rt<<1|1].bit);
32  }
33
34  void fun(int l,int r,int rt){     // 下推
35      if(v[rt].bit.count()){         // 统计 bit 中 1 的个数
36          v[rt<<1].bit=v[rt].bit;
37          v[rt<<1|1].bit=v[rt].bit;
38          tree[rt<<1].bit=tree[rt].bit;
39          tree[rt<<1|1].bit=tree[rt].bit;
40          v[rt].bit=0;              // 记得清零
41      }
42  }
43
44  void upset(int x,int y,int vel,int l,int r,int rt){
45      if(x<=l&&y>=r){
46          v[rt].bit.set(vel);
47          tree[rt].bit=0;
48          tree[rt].bit.set(vel);
49          return ;
50      }
51      fun(l,r,rt);
52      int mid=l+r>>1;
53      if(x<=mid) upset(x,y,vel,l,mid,rt<<1);
54      if(y>mid) upset(x,y,vel,mid+1,r,rt<<1|1);
55      tree[rt].bit=(tree[rt<<1].bit|tree[rt<<1|1].bit);
56  }
57  bitset<270>bb(0);
58  void findd(int x,int y,int l,int r,int rt){
59      if(x<=l&&y>=r){
60          bb|=tree[rt].bit;
61          return ;
62      }
63
64      fun(l,r,rt);
65      int mid=l+r>>1;
66      if(x<=mid)
67          findd(x,y,l,mid,rt<<1);
68      if(y>mid)
69          findd(x,y,mid+1,r,rt<<1|1);
70  }
71
72
73  int main()
74  {
75      int q;
76      int op;
77      int l, r;
78      char ch;
79      int ind;
80      int n;
81      scanf("%s", s+1);
82      n = strlen(s+1);
83      scanf("%d", &q);
84      sett(1, n, 1);
85      while(q--)
86      {
87          scanf("%d", &op);
88          if(op==1)
89          {
90              scanf("%d ", &ind);
91              ch = getchar();
92              upset(ind, ind, ch-'a', 1, n, 1);
93          }
```

```
94          else
95          {
96              scanf("%d%d", &l, &r);
97              bb = 0;
98              findd(l, r, 1, n, 1);
99              printf("%d\n", bb.count());
100         }
101     }
102     return 0;
103 }
```

2.6 主席树

2.6.1 区间第 K 大

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  #define inf 0x3f3f3f3f
5  typedef pair<int, int> P;
6  const int maxn = 1e6+5;
7  const ll mod = 1e9+7;
8
9  // 顶点, 代表区间[l, r]中有v个数字
10 struct node
11 {
12     int l, r, v;
13 }tree[maxn*20]; // 空间开大一点, 因为要动态开点
14
15 // edit[i]存的是第i颗权值线段树的根节点在tree数组中的位置
16 // a是存放原数据的数组, b是离散化后的数组, tot表示顶点的个数
17 int edit[maxn], a[maxn], b[maxn], tot=0;
18
19 // 建树
20 int build(int l, int r)
21 {
22     // 这里就是动态开辟新的结点, 就是将tot加一
23     int pos = ++tot;
24     tree[pos].v = 0; //初始化为0
25     if(l==r)return pos; //到根节点了, 返回
26     // 二分建树没什么好说的
27     int mid = (l+r)>>1;
28     tree[pos].l = build(l, mid);
29     tree[pos].r = build(mid+1, r);
30     //要返回pos的位置, 因为edit数组要存新根的位置
31     return pos;
32 }
33
34 // 就是插入操作, 插入新的元素
35 // ed是前一版本的结点在tree的位置, 更新区间[l, r], 位置为v
36 int update(int ed, int l, int r, int v)
37 {
38     // 动态开点
39     int pos = ++tot;
40     //先将新一版本的当前结点复制为上一个版本的对应结点
41     tree[pos] = tree[ed]; tree[pos].v++; //新一版本的结点的v要加一, 因为对应区间插
         入了一个数
42     if(l==r) return pos; //到叶子节点了, 返回
43     // 二分
44     int mid = (l+r)>>1;
45     // 如果更新位置v在左子树中, 递归更新即可, 在右子树中同理, 最后要返回pos
46     if(v<=mid) tree[pos].l = update(tree[ed].l, l, mid, v);
47     else tree[pos].r = update(tree[ed].r, mid+1, r, v);
48     return pos;
49 }
```

```
50
51 // 查询区间[l, r]第k大的数字, pre对应l-1版本的权值线段树的节点位置, ed代表r版本的
     权值线段树的节点位置
52 int query(int pre, int ed, int l, int r, int k)
53 {
54     // 到叶子节点就返回
55     if(l==r) return l;
56     // 二分
57     int mid = (l+r)>>1;
58     // 先计算左子树的数字个数
59     int x = tree[tree[ed].l].v - tree[tree[pre].l].v;
60     // 若左子树的数字个数大于等于k, 说明我们要找的数字在左子树中, 递归走到左子树继
       续寻找
61     if(x>=k) return query(tree[pre].l, tree[ed].l, l, mid, k);
62     // 否则就在右子树中, 我们要在右子树中寻找k-x大的数字, 递归寻找就好
63     else return query(tree[pre].r, tree[ed].r, mid+1, r, k-x);
64 }
65
66 int main()
67 {
68     int n, q; // n个数字, q次询问
69     scanf("%d%d", &n, &q); // 这一题cin/cout会被卡
70     // 输入数据, 并copy到b数组中
71     for(int i=1;i<=n;i++)
72     {
73         scanf("%d", &a[i]);
74         b[i] = a[i];
75     }
76     // 离散化
77     sort(b+1, b+1+n);
78     int m = unique(b+1, b+n+1) - b-1;
79     // 构建一颗空的权值线段树, edit[0]存放的就是这颗空树的根节点的位置
80     edit[0] = build(1, m);
81     // 插入n个数据
82     for(int i=1;i<=m;i++)
83     {
84         // 找到a[i]离散化后对应的位置
85         a[i] = lower_bound(b+1, b+m+1, a[i]) - b;
86         // edit[i]存放第i版本的权值线段树的根节点位置
87         edit[i] = update(edit[i-1], 1, m, a[i]);
88     }
89     // 处理q次询问
90     while(q--)
91     {
92         int x, y, k;
93         scanf("%d%d%d", &x, &y, &k);
94         // pos对应的是离散化后的位置, 所以最后输出b[pos]即可
95         int pos = query(edit[x-1], edit[y], 1, m, k);
96         printf("%d\n", b[pos]);
97     }
98     return 0;
99 }
```

2.6.2 动态区间第 K 大 (主席树套树状数组) ZOJ2112

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define inf 0x3f3f3f3f
5 typedef pair<int, int> P;
6 const int maxn = 60010;
7 const ll mod = 1e9+7;
8 const int M = 2500010;
9
10 int n, m, q, tot;
```

```
11 struct node
12 {
13     int l, r, v;
14 }tree[M];
15
16 // T是主席树，与上面代码的edit作用一样， S数组就是树状数组， use数组是树状数组求和时用的，记录的是树状数组中哪一些权值线段树要被用来求和
17 int T[maxn], S[maxn], use[maxn], a[maxn], b[maxn];
18
19 // 记录询问，因为要将修改后的值一起构建主席树，所以将在线转为离线
20 struct Q
21 {
22     // 对于查询区间[l, r]第k大的询问来说，flag为1
23     //若是修改操作，l记录修改的位置，r记录新值，flag为0
24     int l, r, k, flag;
25 }query[10010];
26
27 // 快速找出x在离散化后的位置
28 int HASH(int x)
29 {
30     return lower_bound(b+1, b+m+1, x) - b;
31 }
32
33 // 建静态主席树，和之前的一样
34 int build(int l, int r)
35 {
36     int pos = ++tot;
37     tree[pos].v = 0;
38     if(l==r) return pos;
39     int mid = (l+r)>>1;
40     tree[pos].l = build(l, mid);
41     tree[pos].r = build(mid+1, r);
42     return pos;
43 }
44
45 // 和之前的静态主席树update差不多，只不过不是直接将tree[pos].v+1,而是加参数v
46 //消除影响v就为-1， 添加影响v就为1， 其他没什么不同
47 int update(int ed, int l, int r, int p, int v)
48 {
49     int pos = ++tot;
50     tree[pos] = tree[ed];
51     tree[pos].v += v;
52     if(l==r) return pos;
53     int mid = (l+r)>>1;
54     if(p<=mid) tree[pos].l = update(tree[ed].l, l, mid, p, v);
55     else tree[pos].r = update(tree[ed].r, mid+1, r, p, v);
56     return pos;
57 }
58
59 // 树状数组的lowbit
60 int lowbit(int x) { return x&(-x); }
61
62 // 修改操作，修改位置x的影响
63 int add(int x, int v)
64 {
65     // 找出a[x]在离散化后的位置p
66     int p = HASH(a[x]);
67     while(x<=n)
68     {
69         // 修改操作，对树状数组中相应的权值线段树进行修改，消除影响：v=-1， 添加影响：v=1
70         // 因为树状数组中的权值线段树不需要可持久化，所以直接在原版本上修改就可以了
71         S[x] = update(S[x], 1, m, p, v);
72         x+=lowbit(x);
73     }
```

```
74 }
75
76 // 树状数组求和， 求左子树包含的数字个数，和静态主席树一样的思想，都是先求左子树
77 int sum(int x)
78 {
79     int ret = 0;
80     while(x)
81     {
82         // use[i]记录的就是树状数组中相应的权值线段树的结点位置
83         // 似乎这一句有一点点难以理解，结合整体代码多看几遍吧
84         ret += tree[tree[use[x]].l].v;
85         x-=lowbit(x);
86     }
87     return ret;
88 }
89
90 // 询问操作。树状数组求[pre, ed]的和， tpre和ted是静态主席树的区间左右顶点的位置，区间[l, r]第k大
91 int Query(int pre, int ed, int tpre, int ted, int l, int r, int k)
92 {
93     if(l==r)return l;
94     int mid = (l+r)>>1;
95     // sum就是树状数组的求和，相对于静态主席树，多了sum求修改操作的影响
96     // tmp为当前左子树的数字个数
97     int tmp = sum(ed) - sum(pre) + tree[tree[ted].l].v - tree[tree[tpre].l].v;
98     // 若左子树的数字个数大于等于k就往左子树继续走，这里和主席树没什么区别
99     if(tmp >= k)
100    {
101        // 两个for循环是更新左子树需要用到的树状数组中的权值线段树的位置，有那么一丢丢难以理解，多看几遍？
102        for(int i=ed; i; i-=lowbit(i)) use[i] = tree[use[i]].l;
103        for(int i=pre; i; i-=lowbit(i)) use[i] = tree[use[i]].l;
104        return Query(pre, ed, tree[tpre].l, tree[ted].l, l, mid, k);
105    }
106    else
107    {
108        // 走右子树同理
109        for(int i=ed; i; i-=lowbit(i)) use[i] = tree[use[i]].r;
110        for(int i=pre; i; i-=lowbit(i)) use[i] = tree[use[i]].r;
111        return Query(pre, ed, tree[tpre].r, tree[ted].r, mid+1, r, k-tmp);
112    }
113 }
114
115 int main()
116 {
117     int t; // t个case
118     scanf("%d", &t);
119     while(t--)
120     {
121         scanf("%d%d", &n, &q);
122         m = tot = 0; // 记得初始化
123         for(int i=1;i<=n;i++)
124         {
125             scanf("%d", &a[i]);
126             b[++m] = a[i];
127         }
128         char op[5];
129         for(int i=1;i<=q;i++)
130         {
131             scanf("%s", op);
132             // 查询操作
133             if(op[0]=='Q')
134             {
135                 scanf("%d%d%d", &query[i].l, &query[i].r, &query[i].k);
136                 query[i].flag = 1;
137             }
```

```
138             else
139             {
140                 scanf("%d%d", &query[i].l, &query[i].r);
141                 b[++m] = query[i].r; // 注意要将修改后的新值加入到待离散化的b数组
142                 query[i].flag = 0;
143             }
144         }
145         // 离散化
146         sort(b+1, b+m+1);
147         m = unique(b+1, b+m+1) - b-1;
148         // 构建主席树
149         T[0] = build(1, m);
150         for(int i=1;i<=n;i++)
151             T[i] = update(T[i-1], 1, m, HASH(a[i]), 1);
152         // 构建树状数组，每一个节点都是一颗空的权值线段树
153         for(int i=1;i<=n;i++)
154             S[i] = T[0];
155         // 离线处理q个询问
156         for(int i=1;i<=q;i++)
157         {
158             if(query[i].flag) // 查询
159             {
160                 // 两个for循环标记区间[l, r]要使用的树状数组中的权值线段树的位置
161                 for(int j=query[i].r; j; j-=lowbit(j)) use[j] = S[j];
162                 for(int j=query[i].l-1; j; j-=lowbit(j)) use[j] = S[j];
163                 printf("%d\n", b[Query(query[i].l-1, query[i].r, T[query[i].l-1],
                      T[query[i].r], 1, m, query[i].k)]);
164             }
165             else
166             {
167                 // 先消除影响
168                 add(query[i].l, -1);
169                 // 在原数组中更新值
170                 a[query[i].l] = query[i].r;
171                 //添加新值的影响
172                 add(query[i].l, 1);
173             }
174         }
175     }
176     return 0;
177 }
```

2.7 Splay

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef unsigned long long ull;
5  #define inf 0x3f3f3f3f
6  #define INF 0x3f3f3f3f3f3f3f3f
7  #define IO ios::sync_with_stdio(0)
8  #define DEBUG(x) cout<<"--->"<<(x)<<endl;
9  typedef pair<int, int> P;
10 const ll mod = 1e9+7;
11 const double eps = 1e-9;
12 const double PI = acos(-1);
13 const int maxn = 1e5+5;
14
15
16 struct node {
17     int ch[2];
18     int fa, val, cnt, size;
19 }tree[maxn];
20
21 int tot, root;
```

```
22
23 void update(int x) {
24     tree[x].size = tree[tree[x].ch[0]].size + tree[tree[x].ch[1]].size + tree[x].
         cnt;
25 }
26
27 int ident(int x, int f) {
28     return tree[f].ch[1] == x;
29 }
30
31 void connect(int x, int f, int s) {
32     tree[f].ch[s] = x;
33     tree[x].fa = f;
34 }
35
36 void rotate(int x) {
37     int f = tree[x].fa, ff = tree[f].fa, k = ident(x, f);
38     connect(tree[x].ch[k^1], f, k);
39     connect(x, ff, ident(f, ff));
40     connect(f, x, k^1);
41     update(f), update(x);
42 }
43
44 void splaying(int x, int top) {
45     if (!top) root = x;
46     while (tree[x].fa != top) {
47         int f = tree[x].fa, ff = tree[f].fa;
48         if (ff != top) (ident(f, ff) ^ ident(x, f)) ? rotate(x) : rotate(f);
49         rotate(x);
50     }
51 }
52
53 void newnode(int &now, int fa, int val) {
54     tree[now = ++tot].val = val;
55     tree[now].fa = fa;
56     tree[now].size = tree[now].cnt = 1;
57 }
58
59 void delnode(int x) {
60     splaying(x, 0);
61     if (tree[x].cnt > 1) tree[x].cnt--;
62     else if (tree[x].ch[1]) {
63         int p = tree[x].ch[1];
64         while (tree[p].ch[0]) p = tree[p].ch[0];
65         splaying(p, x);
66         connect(tree[x].ch[0], p, 0);
67         root = p;
68         tree[p].fa = 0;
69         update(root);
70     } else {
71         root = tree[x].ch[0], tree[root].fa = 0;
72     }
73 }
74
75 void insert(int val, int &now = root, int fa = 0) {
76     if (!now) newnode(now, fa, val), splaying(now, 0);
77     else if (val < tree[now].val) insert(val, tree[now].ch[0], now);
78     else if (val > tree[now].val) insert(val, tree[now].ch[1], now);
79     else tree[now].cnt++, splaying(now, 0);
80 }
81
82 void del(int val, int now = root) {
83     if (val == tree[now].val) delnode(now);
84     else if (val < tree[now].val) del(val, tree[now].ch[0]);
85     else del(val, tree[now].ch[1]);
86 }
```

```
87
88 int getrank(int val) {
89     int now = root, rank = 1;
90     while (now) {
91         if (tree[now].val == val) {
92             rank += tree[tree[now].ch[0]].size;
93             splaying(now, 0);
94             break;
95         }
96         if (val <= tree[now].val)
97             now = tree[now].ch[0];
98         else {
99             rank += tree[tree[now].ch[0]].size + tree[now].cnt;
100            now = tree[now].ch[1];
101        }
102    }
103    return rank;
104 }
105
106 int getnum(int rank) {
107    int now = root;
108    while (now) {
109        int lsize = tree[tree[now].ch[0]].size;
110        if (lsize+1 <= rank && rank <= lsize + tree[now].cnt) {
111            splaying(now, 0);
112            break;
113        } else if (lsize >= rank)
114            now = tree[now].ch[0];
115        else {
116            rank -= lsize + tree[now].cnt;
117            now = tree[now].ch[1];
118        }
119    }
120    return tree[now].val;
121 }
122
123 int getpre(int x) {
124    return getnum(getrank(x) - 1);
125 }
126
127 int getnxt(int x) {
128    return getnum(getrank(x + 1));
129 }
130
131 int main() {
132    int t;
133    scanf("%d", &t);
134    while (t--) {
135        int opt, x;
136        scanf("%d%d", &opt, &x);
137        if (opt == 1) insert(x);
138        else if (opt == 2) del(x);
139        else if (opt == 3) printf("%d\n", getrank(x));
140        else if (opt == 4) printf("%d\n", getnum(x));
141        else if (opt == 5) printf("%d\n", getpre(x));
142        else printf("%d\n", getnxt(x));
143    }
144    return 0;
145 }
```

2.8 Treap

2.8.1 普通 Treap

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

```
2  using namespace std;
3  typedef long long ll;
4  typedef unsigned long long ull;
5  #define inf 0x3f3f3f3f
6  #define INF 0x3f3f3f3f3f3f3f3f
7  #define IO ios::sync_with_stdio(0)
8  #define DEBUG(x) cout<<"--->"<<(x)<<endl;
9  typedef pair<int, int> P;
10 const ll mod = 1e9+7;
11 const double eps = 1e-9;
12 const double PI = acos(-1);
13 const int maxn = 2e5+5;
14
15
16 struct node {
17     int son[2];
18     int val, key;
19     int cnt, size;
20 } tree[maxn];
21
22 int tot, root;
23
24 mt19937 rnd(233);
25
26 void pushup(int p) {
27     tree[p].size = tree[tree[p].son[0]].size + tree[tree[p].son[1]].size + tree[p
       ].cnt;
28 }
29
30 void rotate(int &p, int d) {
31     int k = tree[p].son[d^1];
32     tree[p].son[d^1] = tree[k].son[d];
33     tree[k].son[d] = p;
34     pushup(p);
35     pushup(k);
36     p = k;
37 }
38
39 void insert(int &p, int x) {
40     if (!p) {
41         p = ++tot;
42         tree[p].size = tree[p].cnt = 1;
43         tree[p].val = x;
44         tree[p].key = rnd();
45         return;
46     }
47     if (tree[p].val == x) {
48         tree[p].size++;
49         tree[p].cnt++;
50         return;
51     }
52     int d = (x > tree[p].val);
53     insert(tree[p].son[d], x);
54     if (tree[p].key < tree[tree[p].son[d]].key) rotate(p, d^1);
55     pushup(p);
56 }
57
58 void del(int &p, int x) {
59     if (!p) return;
60     if (x < tree[p].val) del(tree[p].son[0], x);
61     else if (x > tree[p].val) del(tree[p].son[1], x);
62     else {
63         if (!tree[p].son[0] && !tree[p].son[1]) {
64             tree[p].cnt--;
65             tree[p].size--;
66             if (tree[p].cnt == 0) p = 0;
```

```
67          } else if (tree[p].son[0] && !tree[p].son[1]) {
68              rotate(p, 1);
69              del(tree[p].son[1], x);
70          } else if (!tree[p].son[0] && tree[p].son[1]) {
71              rotate(p, 0);
72              del(tree[p].son[0], x);
73          } else {
74              int d = (tree[tree[p].son[0]].key > tree[tree[p].son[1]].key);
75              rotate(p, d);
76              del(tree[p].son[d], x);
77          }
78      }
79      pushup(p);
80  }
81
82  int getrank(int p, int x) {
83      if (!p) return 0;
84      if (tree[p].val == x) return tree[tree[p].son[0]].size + 1;
85      else if (tree[p].val < x)
86          return tree[tree[p].son[0]].size + tree[p].cnt + getrank(tree[p].son[1], x
              );
87      else return getrank(tree[p].son[0], x);
88  }
89
90  int getnum(int p, int x) {
91      if (!p) return 0;
92      if (tree[tree[p].son[0]].size >= x) return getnum(tree[p].son[0], x);
93      else if (tree[tree[p].son[0]].size + tree[p].cnt < x)
94          return getnum(tree[p].son[1], x - tree[tree[p].son[0]].size - tree[p].cnt)
              ;
95      else return tree[p].val;
96  }
97
98  int getpre(int p, int x) {
99      if (!p) return -inf;
100     if (tree[p].val >= x) return getpre(tree[p].son[0], x);
101     else return max(tree[p].val, getpre(tree[p].son[1], x));
102 }
103
104 int getnxt(int p, int x) {
105     if (!p) return inf;
106     if (tree[p].val <= x) return getnxt(tree[p].son[1], x);
107     else return min(tree[p].val, getnxt(tree[p].son[0], x));
108 }
109
110 int main() {
111     int n;
112     scanf("%d", &n);
113     for (int i=1; i<=n; i++) {
114         int opt, x;
115         scanf("%d%d", &opt, &x);
116         if (opt == 1) insert(root, x);
117         else if (opt == 2) del(root, x);
118         else if (opt == 3) printf("%d\n", getrank(root, x));
119         else if (opt == 4) printf("%d\n", getnum(root, x));
120         else if (opt == 5) printf("%d\n", getpre(root, x));
121         else printf("%d\n", getnxt(root, x));
122     }
123     return 0;
124 }
```

2.8.2 无旋 Treap

```
1  #include<bits/stdc++.h>
2  using namespace std;
```

```
3  typedef long long ll;
4  typedef unsigned long long ull;
5  #define inf 0x3f3f3f3f
6  #define INF 0x3f3f3f3f3f3f3f3f
7  #define IO ios::sync_with_stdio(0)
8  #define DEBUG(x) cout<<"--->"<<(x)<<endl;
9  typedef pair<int, int> P;
10 const ll mod = 1e9+7;
11 const double eps = 1e-9;
12 const double PI = acos(-1);
13 const int maxn = 2e5+5;
14
15
16 struct node {
17     int l, r;
18     int val, key;
19     int size;
20 }tree[maxn];
21
22 int root, cnt;
23 mt19937 rnd(233);
24
25 int x, y, z;
26
27 int newnode(int val) {
28     ++cnt;
29     tree[cnt].val = val;
30     tree[cnt].size = 1;
31     tree[cnt].key = rnd();
32     return cnt;
33 }
34
35 void update(int now) {
36     tree[now].size = tree[tree[now].l].size + tree[tree[now].r].size + 1;
37 }
38
39 void split(int now, int val, int &x, int &y) {
40     if (!now) x = y = 0;
41     else {
42         if (tree[now].val <= val) {
43             x = now;
44             split(tree[now].r, val, tree[now].r, y);
45         } else {
46             y = now;
47             split(tree[now].l, val, x, tree[now].l);
48         }
49         update(now);
50     }
51 }
52
53 int merge(int x, int y) {
54     if (!x || !y) return x + y;
55     if (tree[x].key > tree[y].key) {
56         tree[x].r = merge(tree[x].r, y);
57         update(x);
58         return x;
59     } else {
60         tree[y].l = merge(x, tree[y].l);
61         update(y);
62         return y;
63     }
64 }
65
66 void insert(int val) {
67     split(root, val, x, y);
68     root = merge(merge(x, newnode(val)), y);
```

```
69 }
70
71 void del(int val) {
72     split(root, val, x, z);
73     split(x, val-1, x, y);
74     y = merge(tree[y].l, tree[y].r);
75     root = merge(merge(x, y), z);
76 }
77
78 int getrank(int val) {
79     int res;
80     split(root, val-1, x, y);
81     res = tree[x].size + 1;
82     root = merge(x, y);
83     return res;
84 }
85
86 int getnum(int rank) {
87     int now = root;
88     while (now) {
89         if (tree[tree[now].l].size + 1 == rank)
90             break;
91         else if (tree[tree[now].l].size >= rank)
92             now = tree[now].l;
93         else {
94             rank -= tree[tree[now].l].size + 1;
95             now = tree[now].r;
96         }
97     }
98     return tree[now].val;
99 }
100
101 int getpre(int val) {
102     split(root, val-1, x, y);
103     int now = x;
104     while (tree[now].r)
105         now = tree[now].r;
106     int res = tree[now].val;
107     root = merge(x, y);
108     return res;
109 }
110
111 int getnxt(int val) {
112     split(root, val, x, y);
113     int now = y;
114     while (tree[now].l)
115         now = tree[now].l;
116     int res = tree[now].val;
117     root = merge(x, y);
118     return res;
119 }
120
121 int main() {
122     int t;
123     scanf("%d", &t);
124     while (t--) {
125         int opt, x;
126         scanf("%d%d", &opt, &x);
127         if (opt == 1) insert(x);
128         else if (opt == 2) del(x);
129         else if (opt == 3) printf("%d\n", getrank(x));
130         else if (opt == 4) printf("%d\n", getnum(x));
131         else if (opt == 5) printf("%d\n", getpre(x));
132         else printf("%d\n", getnxt(x));
133     }
134     return 0;
```

```
}
```

2.8.3 无旋 Treap 维护区间

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
#define inf 0x3f3f3f3f
#define INF 0x3f3f3f3f3f3f3f3f
#define IO ios::sync_with_stdio(0)
#define DEBUG(x) cout<<"--->"<<(x)<<endl;
typedef pair<int, int> P;
const ll mod = 1e9+7;
const double eps = 1e-9;
const double PI = acos(-1);
const int maxn = 1e5+5;

struct node {
    int l, r;
    int val, key;
    int size;
    int reverse;
}tree[maxn];

int cnt, root;
mt19937 rnd(233);
int x, y, z;

int newnode(int val) {
    ++cnt;
    tree[cnt].val = val;
    tree[cnt].size = 1;
    tree[cnt].key = rnd();
    return cnt;
}

void update(int now) {
    tree[now].size = tree[tree[now].l].size + tree[tree[now].r].size + 1;
}

void pushdown(int now) {
    swap(tree[now].l, tree[now].r);
    tree[tree[now].l].reverse ^= 1;
    tree[tree[now].r].reverse ^= 1;
    tree[now].reverse = 0;
}

void split(int now, int sz, int &x, int &y) {
    if (!now) x = y = 0;
    else {
        if (tree[now].reverse) pushdown(now);
        if (tree[tree[now].l].size < sz) {
            x = now;
            split(tree[now].r, sz - tree[tree[now].l].size - 1, tree[now].r, y);
        } else {
            y = now;
            split(tree[now].l, sz, x, tree[now].l);
        }
        update(now);
    }
}

int merge(int x, int y) {
    if (!x || !y) return x + y;
```

```
62      if (tree[x].key < tree[y].key) {
63          if (tree[x].reverse) pushdown(x);
64          tree[x].r = merge(tree[x].r, y);
65          update(x);
66          return x;
67      } else {
68          if (tree[y].reverse) pushdown(y);
69          tree[y].l = merge(x, tree[y].l);
70          update(y);
71          return y;
72      }
73  }
74
75  void reverse(int l, int r) {
76      int x, y, z;
77      split(root, l-1, x, y);
78      split(y, r-l+1, y, z);
79      tree[y].reverse ^= 1;
80      root = merge(merge(x, y), z);
81  }
82
83  void print(int now) {
84      if (!now) return;
85      if (tree[now].reverse) pushdown(now);
86      print(tree[now].l);
87      printf("%d ", tree[now].val);
88      print(tree[now].r);
89  }
90
91  int main() {
92      int n, m;
93      scanf("%d%d", &n, &m);
94      for (int i=1; i<=n; i++)
95          root = merge(root, newnode(i));
96      while (m--) {
97          int l, r;
98          scanf("%d%d", &l, &r);
99          reverse(l, r);
100     }
101     print(root);
102     printf("\n");
103     return 0;
104 }
```

2.9 替罪羊树

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   typedef long long ll;
4   typedef unsigned long long ull;
5   #define inf 0x3f3f3f3f
6   #define INF 0x3f3f3f3f3f3f3f3f
7   #define IO ios::sync_with_stdio(0)
8   #define DEBUG(x) cout<<"--->"<<(x)<<endl;
9   typedef pair<int, int> P;
10  const ll mod = 1e9+7;
11  const double eps = 1e-9;
12  const double PI = acos(-1);
13  const int maxn = 2e5+5;
14
15
16  struct node {
17      int l, r, val;
18      int size, fact;
19      int exist;
```

```
}tree[maxn];

int cnt, root;
const double alpha = 0.75;  // 平衡因子
vector<int> v;  // 记录中序遍历的结果

// 新建结点
void newnode(int &now, int val) {
    now = ++cnt;
    tree[now].val = val;
    tree[now].size = tree[now].fact = 1;
    tree[now].exist = 1;
}

// 判断是否不平衡
bool imbalence(int now) {
    if (max(tree[tree[now].l].size, tree[tree[now].r].size) > tree[now].size *
      alpha
            || tree[now].size - tree[now].fact > tree[now].size * 0.3)
        return true;
    return false;
}

// 对now为根的子树做中序遍历
void inorder(int now) {
    if (!now) return;
    inorder(tree[now].l);
    if (tree[now].exist) v.push_back(now);
    inorder(tree[now].r);
}

// 对中序遍历的结果重构
void lift(int l, int r, int &now) {
    if (l == r) {
        now = v[l];
        tree[now].l = tree[now].r = 0;
        tree[now].size = tree[now].fact = 1;
        return;
    }
    int mid = (l + r) >> 1;
    while (l < mid && tree[v[mid]].val == tree[v[mid-1]].val) mid--;
    now = v[mid];
    if (l < mid) lift(l, mid-1, tree[now].l);
    else tree[now].l = 0;
    lift(mid+1, r, tree[now].r);
    tree[now].size = tree[tree[now].l].size + tree[tree[now].r].size + 1;
    tree[now].fact = tree[tree[now].l].fact + tree[tree[now].r].fact + 1;
}

// 对以now为根的子树重构
void rebuild(int &now) {
    v.clear();
    inorder(now);
    if (v.empty()) {
        now = 0;
        return;
    }
    lift(0, v.size()-1, now);
}

// 重构完后更新根到now链路上结点的信息
void update(int now, int end) {
    if (!now) return;
    if (tree[end].val < tree[now].val) update(tree[now].l, end);
    else update(tree[now].r, end);
    tree[now].size = tree[tree[now].l].size + tree[tree[now].r].size + 1;
```

```
}

// 检查是否需要重构
void check(int &now, int end) {
    if (now == end) return;
    if (imbalence(now)) {
        rebuild(now);
        update(root, now);
        return;
    }
    if (tree[end].val < tree[now].val) check(tree[now].l, end);
    else check(tree[now].r, end);
}

// 插入
void insert(int &now, int val) {
    if (!now) {
        newnode(now, val);
        check(root, now);
        return;
    }
    tree[now].size++;
    tree[now].fact++;
    if (val < tree[now].val) insert(tree[now].l, val);
    else insert(tree[now].r, val);
}

// 删除
void del(int now, int val) {
    if (tree[now].exist && tree[now].val == val) {
        tree[now].exist = 0;
        tree[now].fact--;
        check(root, now);
        return;
    }
    tree[now].fact--;
    if (val < tree[now].val) del(tree[now].l, val);
    else del(tree[now].r, val);
}

// 根据值获得排名，即比当前数小的个数+1
int getrank(int val) {
    int now = root, rank = 1;
    while(now) {
        if (val <= tree[now].val)
            now = tree[now].l;
        else {
            rank += tree[now].exist + tree[tree[now].l].fact;
            now = tree[now].r;
        }
    }
    return rank;
}

// 根据排名获得值
int getnum(int rank) {
    int now = root;
    while (now) {
        if (tree[now].exist && tree[tree[now].l].fact + tree[now].exist == rank)
            break;
        else if(tree[tree[now].l].fact >= rank)
            now = tree[now].l;
        else {
            rank -= tree[tree[now].l].fact + tree[now].exist;
            now = tree[now].r;
        }
```

```
151     }
152     return tree[now].val;
153 }
154
155 // 找x的前驱，即小于x且最大的数
156 int getpre(int x) {
157     return getnum(getrank(x) - 1);
158 }
159
160 // 找x的后继，即大于x且最小的数
161 int getnext(int x) {
162     return getnum(getrank(x + 1));
163 }
164
165 int main() {
166     int t;
167     scanf("%d", &t);
168     while (t--) {
169         int opt, x;
170         scanf("%d%d", &opt, &x);
171         if (opt == 1) insert(root, x);
172         else if(opt == 2) del(root, x);
173         else if(opt == 3) printf("%d\n", getrank(x));
174         else if(opt == 4) printf("%d\n", getnum(x));
175         else if(opt == 5) printf("%d\n", getpre(x));
176         else printf("%d\n", getnext(x));
177     }
178     return 0;
179 }
```

2.10 树链剖分

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef unsigned long long ull;
5  typedef pair<int, int> P;
6  const int maxn = 1e6+5;
7  const int inf = 0x3f3f3f3f;
8  const ll INF = 0x3f3f3f3f3f3f3f3f;
9  const int mod = 1e9+7;
10 const double PI = acos(-1);
11
12
13 #define ls k<<1
14 #define rs k<<1|1
15
16 ll root = 1;
17 struct node
18 {
19     ll u,v,w,nxt;
20 }edge[maxn];
21 ll head[maxn];
22 ll num;
23
24 inline void addedge(ll x, ll y)
25 {
26     edge[++num].u = x;
27     edge[num].v = y;
28     edge[num].nxt = head[x];
29     head[x] = num;
30 }
31
32 struct Tree
33 {
```

```
34      // f是lazy标记, siz是区间大小
35      ll l, r, f, w, siz;
36  }T[maxn<<2];
37  // a建线段树用, 即重新排序后的结点顺序
38  // b记录输入点权, tot[i]记录顶点i子树的大小, idx记录编号, deep深度
39  // son记录重儿子, top[i]记录i所在链的起点, fa记录父亲
40  ll a[maxn], b[maxn], tot[maxn], idx[maxn], deep[maxn];
41  ll son[maxn], top[maxn], fa[maxn], cnt;
42
43  void update(ll k)
44  {
45      T[k].w = T[ls].w + T[rs].w;
46  }
47  void pushdown(ll k)
48  {
49      if(!T[k].f)return ;
50      T[ls].w += T[k].f * T[ls].siz;
51      T[rs].w += T[k].f * T[rs].siz;
52      T[ls].f += T[k].f;
53      T[rs].f += T[k].f;
54      T[k].f = 0;
55  }
56  ll dfs1(ll now, ll f, ll dep)
57  {
58      deep[now] = dep;
59      tot[now] = 1;
60      fa[now] = f;
61      ll maxson = -1;
62      for(ll i=head[now]; ~i; i=edge[i].nxt)
63      {
64          if(edge[i].v==f)continue;
65          tot[now] += dfs1(edge[i].v, now, dep+1);
66          if(tot[edge[i].v]>maxson)
67              maxson=tot[edge[i].v], son[now]=edge[i].v;
68      }
69      return tot[now];
70  }
71  void dfs2(ll now, ll topf)
72  {
73      idx[now] = ++cnt;
74      a[cnt] = b[now];
75      top[now] = topf;
76      if(!son[now])return;
77      dfs2(son[now], topf);
78      for(ll i=head[now]; ~i; i=edge[i].nxt)
79          if(!idx[edge[i].v])
80              dfs2(edge[i].v, edge[i].v);
81  }
82  void build(ll k, ll l, ll r)
83  {
84      T[k].l=l; T[k].r=r; T[k].siz=r-l+1;
85      if(l==r)
86      {
87          T[k].w = a[l];
88          return;
89      }
90      ll mid = (l+r)>>1;
91      build(ls, l, mid);
92      build(rs, mid+1, r);
93      update(k);
94  }
95  // 单点修改
96  void pointadd(ll k, ll pos, ll val)
97  {
98      if(T[k].l == T[k].r)
99      {
```

```
100         T[k].w += val;return;
101     }
102     pushdown(k);
103     ll mid = (T[k].l+T[k].r)>>1;
104     if(pos<=mid)pointadd(ls, pos, val);
105     if(pos>mid)pointadd(rs, pos, val);
106     update(k);
107 }
108 // 区间修改
109 void intervaladd(ll k, ll l, ll r, ll val)
110 {
111     if(l<=T[k].l && T[k].r <= r)
112     {
113         T[k].w += T[k].siz*val;
114         T[k].f += val;
115         return ;
116     }
117     pushdown(k);
118     ll mid = (T[k].l + T[k].r)>>1;
119     if(l<=mid)intervaladd(ls, l, r, val);
120     if(r>mid)intervaladd(rs, l, r, val);
121     update(k);
122 }
123 // 区间查询
124 ll intervalask(ll k, ll l, ll r)
125 {
126     ll ans = 0;
127     if(l <= T[k].l && T[k].r <= r)
128     {
129         ans += T[k].w;
130         return ans;
131     }
132     pushdown(k);
133     ll mid = (T[k].l + T[k].r)>>1;
134     if(l<=mid)ans+=intervalask(ls, l, r);
135     if(r>mid)ans+=intervalask(rs, l, r);
136     return ans;
137 }
138 // 树上x到y的路径权值和
139 ll treesum(ll x, ll y)
140 {
141     ll ans = 0;
142     while(top[x]!=top[y])
143     {
144         if(deep[top[x]] < deep[top[y]])
145             swap(x, y);
146         ans += intervalask(1, idx[top[x]], idx[x]);
147         x=fa[top[x]];
148     }
149     if(deep[x]>deep[y])
150         swap(x, y);
151     ans += intervalask(1, idx[x], idx[y]);
152     return ans;
153 }
154
155
156 int main()
157 {
158     memset(head, -1, sizeof(head));
159     ll n, m;
160     scanf("%lld%lld", &n, &m);
161     for(int i=1;i<=n;i++)
162         scanf("%lld", &b[i]);
163     for(int i=1;i<n;i++)
164     {
165         ll x, y;
```

```
166         scanf("%lld%lld", &x, &y);
167         addedge(x, y); addedge(y, x);
168     }
169     dfs1(root, 0, 1);
170     dfs2(root, root);
171     build(1, 1, n);
172     while(m--)
173     {
174         ll opt, x, val;
175         scanf("%lld",&opt);
176         if(opt==1)
177         {
178             scanf("%lld%lld", &x, &val);
179             pointadd(1, idx[x], val);
180         }
181         else if(opt==2)
182         {
183             scanf("%lld%lld", &x, &val);
184             intervaladd(1, idx[x], idx[x]+tot[x]-1, val);
185         }
186         else
187         {
188             scanf("%lld", &x);
189             printf("%lld\n", treesum(root, x));
190         }
191     }
192     return 0;
193 }
```

2.11 可持久化数组

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef unsigned long long ull;
5  #define inf 0x3f3f3f3f
6  #define INF 0x3f3f3f3f3f3f3f3f
7  #define IO ios::sync_with_stdio(0)
8  #define DEBUG(x) cout<<"--->"<<(x)<<endl;
9  typedef pair<int, int> P;
10 const ll mod = 1e9+7;
11 const double eps = 1e-9;
12 const double PI = acos(-1);
13 const int maxn = 1e6+5;
14
15 struct node {
16     int l, r, val;
17 }tree[maxn * 40];
18
19 int root[maxn], a[maxn];
20 int cnt;
21
22 void build(int l, int r, int &now) {
23     now = ++cnt;
24     if (l == r) {
25         tree[now].val = a[l];
26         return;
27     }
28     int mid = (l + r) >> 1;
29     build(l, mid, tree[now].l);
30     build(mid+1, r, tree[now].r);
31 }
32
33 void modify(int l, int r, int pre, int &now, int pos, int val) {
34     now = ++cnt;
```

```
35      tree[now] = tree[pre];
36      if (l == r) {
37          tree[now].val = val;
38          return ;
39      }
40      int mid = (l + r) >> 1;
41      if (pos <= mid) modify(l, mid, tree[pre].l, tree[now].l, pos, val);
42      else modify(mid+1, r, tree[pre].r, tree[now].r, pos, val);
43  }
44
45  int query(int l, int r, int now, int pos) {
46      if (l == r) return tree[now].val;
47      int mid = (l + r) >> 1;
48      if (pos <= mid) return query(l, mid, tree[now].l, pos);
49      else return query(mid+1, r, tree[now].r, pos);
50  }
51
52  int main() {
53      int n, m;
54      scanf("%d%d", &n, &m);
55      for (int i=1; i<=n; i++) scanf("%d", &a[i]);
56      build(1, n, root[0]);
57      for (int i=1; i<=m; i++) {
58          int ver, opt;
59          scanf("%d%d", &ver, &opt);
60          if (opt == 1) {
61              int pos, val;
62              scanf("%d%d", &pos, &val);
63              modify(1, n, root[ver], root[i], pos, val);
64          } else {
65              int pos;
66              scanf("%d", &pos);
67              printf("%d\n", query(1, n, root[ver], pos));
68              root[i] = root[ver];
69          }
70      }
71      return 0;
72  }
```

2.12 可持久化并查集

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   typedef long long ll;
4   typedef unsigned long long ull;
5   #define inf 0x3f3f3f3f
6   #define INF 0x3f3f3f3f3f3f3f3f
7   #define IO ios::sync_with_stdio(0)
8   #define DEBUG(x) cout<<"--->"<<(x)<<endl;
9   typedef pair<int, int> P;
10  const ll mod = 1e9+7;
11  const double eps = 1e-9;
12  const double PI = acos(-1);
13  const int maxn = 2e5+5;
14
15  struct node {
16      int l, r, val;
17  }tree[maxn * 40 * 2];
18
19  int rootfa[maxn], rootdep[maxn];
20  int cnt, tot;
21  int n, m;
22
23  void build(int l, int r, int &now) {
24      now = ++cnt;
```

```
25      if (l == r) {
26          tree[now].val = ++tot;
27          return;
28      }
29      int mid = (l + r) >> 1;
30      build(l, mid, tree[now].l);
31      build(mid+1, r, tree[now].r);
32  }
33
34  void modify(int l, int r, int pre, int &now, int pos, int val) {
35      now = ++cnt;
36      tree[now] = tree[pre];
37      if (l == r) {
38          tree[now].val = val;
39          return;
40      }
41      int mid = (l + r) >> 1;
42      if (pos <= mid) modify(l, mid, tree[pre].l, tree[now].l, pos, val);
43      else modify(mid+1, r, tree[pre].r, tree[now].r, pos, val);
44  }
45
46  int query(int l, int r, int now, int pos) {
47      if (l == r) return tree[now].val;
48      int mid = (l + r) >> 1;
49      if (pos <= mid) return query(l, mid, tree[now].l, pos);
50      else return query(mid+1, r, tree[now].r, pos);
51  }
52
53  int find(int ver, int x) {
54      int fx = query(1, n, rootfa[ver], x);
55      return fx == x ? x : find(ver, fx);
56  }
57
58  void merge(int ver, int x, int y) {
59      x = find(ver-1, x); // 新版本还未操作，要从上一个版本找
60      y = find(ver-1, y);
61      if (x == y) {
62          rootfa[ver] = rootfa[ver-1];
63          rootdep[ver] = rootdep[ver-1];
64      } else {
65          int depx = query(1, n, rootdep[ver-1], x);
66          int depy = query(1, n, rootdep[ver-1], y);
67          if (depx < depy) {
68              modify(1, n, rootfa[ver-1], rootfa[ver], x, y);
69              rootdep[ver] = rootdep[ver-1];
70          } else if(depx > depy) {
71              modify(1, n, rootfa[ver-1], rootfa[ver], y, x);
72              rootdep[ver] = rootdep[ver-1];
73          } else {
74              modify(1, n, rootfa[ver-1], rootfa[ver], x, y);
75              modify(1, n, rootdep[ver-1], rootdep[ver], y, depy+1);
76          }
77      }
78  }
79
80  int main() {
81      scanf("%d%d", &n, &m);
82      build(1, n, rootfa[0]);
83      for (int i=1; i<=m; i++) {
84          int opt;
85          scanf("%d", &opt);
86          if (opt == 1) {
87              int a, b;
88              scanf("%d%d", &a, &b);
89              merge(i, a, b);
90          } else if(opt == 2) {
```

```
            int k;
            scanf("%d", &k);
            rootfa[i] = rootfa[k];
            rootdep[i] = rootdep[k];
        } else {
            int a, b;
            scanf("%d%d", &a, &b);
            rootfa[i] = rootfa[i-1];
            rootdep[i] = rootdep[i-1];
            int fa = find(i, a);
            int fb = find(i, b);
            printf("%d\n", (fa == fb ? 1 : 0));
        }
    }
    return 0;
}
```

2.13 莫队

2.13.1 普通莫队

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
#define inf 0x3f3f3f3f
#define INF 0x3f3f3f3f3f3f3f3f
#define IO ios::sync_with_stdio(0)
#define DEBUG(x) cout<<"--->"<<(x)<<endl;
typedef pair<int, int> P;
const ll mod = 1e9+7;
const double eps = 1e-9;
const double PI = acos(-1);
const int maxn = 2e5+5;


struct Q {
    int l, r, k;
}q[maxn];

int a[maxn], pos[maxn], cnt[maxn];
ll ans[maxn];
ll res;

bool cmp(Q x, Q y) {
    return pos[x.l] == pos[y.l] ? x.r < y.r : pos[x.l] < pos[y.l];
}

void add(int x) {
    cnt[a[x]]++;
    res += 1ll * cnt[a[x]] * cnt[a[x]] - 1ll * (cnt[a[x]] - 1) * (cnt[a[x]] - 1);
}

void sub(int x) {
    cnt[a[x]]--;
    res -= 1ll * (cnt[a[x]] + 1) * (cnt[a[x]] + 1) - 1ll * cnt[a[x]] * cnt[a[x]];
}

int main() {
    int n, m, k;
    cin >> n >> m >> k;
    int size = sqrt(n);
    for (int i=1; i<=n; i++) {
        cin >> a[i];
        pos[i] = i / size;
```

```
45      }
46      for (int i=0; i<m; i++) {
47          cin >> q[i].l >> q[i].r;
48          q[i].k = i;
49      }
50      sort(q, q+m, cmp);
51      int l = 1, r = 0;
52      for (int i=0 ;i<m; i++) {
53          while (q[i].l < l) add(--l);
54          while (q[i].r > r) add(++r);
55          while (q[i].l > l) sub(l++);
56          while (q[i].r < r) sub(r--);
57          ans[q[i].k] = res;
58      }
59      for (int i=0; i<m; i++) cout << ans[i] << '\n';
60      return 0;
61  }
```

2.13.2 带修莫队

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   typedef long long ll;
4   typedef unsigned long long ull;
5   #define inf 0x3f3f3f3f
6   #define INF 0x3f3f3f3f3f3f3f3f
7   #define IO ios::sync_with_stdio(0)
8   #define DEBUG(x) cout<<"--->"<<(x)<<endl;
9   typedef pair<int, int> P;
10  const ll mod = 1e9+7;
11  const double eps = 1e-9;
12  const double PI = acos(-1);
13  const int maxn = 1e6+5;
14
15  // 查询
16  struct Q {
17      int l, r, k, t;
18  }q[maxn];
19  // 修改
20  struct M {
21      int p, col;
22  }c[maxn];
23  int a[maxn], cnt[maxn], ans[maxn], pos[maxn];
24  int cntq, cntc, n, m, size;
25
26  bool cmp(Q a, Q b) {
27      return (pos[a.l] ^ pos[b.l]) ? pos[a.l] < pos[b.l] : ((pos[a.r] ^ pos[b.r]) ?
          pos[a.r] < pos[b.r] : a.t < b.t);
28  }
29
30  int main() {
31      scanf("%d", &n);
32      scanf("%d", &m);
33      // 块大小与普通莫队不同
34      size = pow(n, 2.0 / 3.0);
35      for (int i=1; i<=n; i++) {
36          pos[i] = i / size;
37      scanf("%d", &a[i]);
38      }
39      for (int i=1; i<=m; i++) {
40          char opt[100];
41          scanf("%s", opt);
42          if (opt[0] == 'Q') {
43              ++cntq;
44              scanf("%d%d", &q[cntq].l, &q[cntq].r);
```

```
45              q[cntq].k = cntq;
46              q[cntq].t = cntc;   // 查询询问的时间戳
47          } else {
48              ++cntc;
49              scanf("%d%d", &c[cntc].p, &c[cntc].col);
50          }
51      }
52      sort(q + 1, q + cntq + 1, cmp);
53      int l = 1, r = 0, time = 0, res = 0;
54      for (int i=1; i<=cntq; i++) {
55          int ql = q[i].l, qr = q[i].r, qt = q[i].t;
56          while (ql < l) {
57              l--;
58              cnt[a[l]]++;
59              res += cnt[a[l]] == 1;
60          }
61          while (ql > l) {
62              cnt[a[l]]--;
63              res -= cnt[a[l]] == 0;
64              l++;
65          }
66          while (qr < r) {
67              cnt[a[r]]--;
68              res -= cnt[a[r]] == 0;
69              r--;
70          }
71          while (qr > r) {
72              r++;
73              cnt[a[r]]++;
74              res += cnt[a[r]] == 1;
75          }
76          // 下面两个是针对修改的移动
77          while (time < qt) {
78              ++time;
79              if (ql <= c[time].p && c[time].p <= qr) {
80                  cnt[a[c[time].p]]--;
81                  cnt[c[time].col]++;
82                  res -= cnt[a[c[time].p]] == 0;
83                  res += cnt[c[time].col] == 1;
84              }
85              swap(a[c[time].p], c[time].col);
86          }
87          while (time > qt) {
88              if (ql <= c[time].p && c[time].p <= qr) {
89                  cnt[a[c[time].p]]--;
90                  cnt[c[time].col]++;
91                  res -= cnt[a[c[time].p]] == 0;
92                  res += cnt[c[time].col] == 1;
93              }
94              swap(a[c[time].p], c[time].col);
95              --time;
96          }
97          ans[q[i].k] = res;
98      }
99      for (int i=1; i<=cntq; i++)
100         printf("%d\n", ans[i]);
101     return 0;
102 }
```

2.14 树上启发式合并

```
1 // 给出一个树，求出每个节点的子树中出现次数最多的颜色的编号和
2 const int maxn = 1e5 + 5;
3
4 int n, tot;
```

```
5  int a[maxn], sz[maxn], hson[maxn], head[maxn], cnt[maxn];
6  ll ans[maxn];
7  ll sum, mx, skip;
8
9  struct Edge {
10     int to, nxt;
11 }edge[maxn << 1];
12
13 void addedge(int u, int v) {
14     edge[++tot].to = v;
15     edge[tot].nxt = head[u];
16     head[u] = tot;
17 }
18
19 void dfs1(int u, int fa) {
20     sz[u] = 1;
21     for (int i = head[u]; i; i = edge[i].nxt) {
22         int v = edge[i].to;
23         if (v == fa) continue;
24         dfs1(v, u);
25         sz[u] += sz[v];
26         if (hson[u] == 0 || sz[v] > sz[hson[u]]) hson[u] = v;
27     }
28 }
29
30 void add(int u, int fa, int val) {
31     cnt[a[u]] += val;
32     if (cnt[a[u]] > mx) {
33         mx = cnt[a[u]];
34         sum = a[u];
35     } else if (cnt[a[u]] == mx) {
36         sum += a[u];
37     }
38     for (int i = head[u]; i; i = edge[i].nxt) {
39         int v = edge[i].to;
40         if (v == fa || v == skip) continue;
41         add(v, u, val);
42     }
43 }
44
45 void dfs2(int u, int fa, int keep) {
46     for (int i = head[u]; i; i = edge[i].nxt) {
47         int v = edge[i].to;
48         if (v == fa || v == hson[u]) continue;
49         dfs2(v, u, 0);
50     }
51     if (hson[u]) {
52         dfs2(hson[u], u, 1);
53         skip = hson[u];
54     }
55     add(u, fa, 1);
56     skip = 0;
57     ans[u] = sum;
58     if (!keep) {
59         add(u, fa, -1);
60         sum = 0;
61         mx = 0;
62     }
63 }
64
65 int main() {
66     // freopen("in.txt", "r", stdin);
67     scanf("%d", &n);
68     for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
69     for (int i = 1; i < n; i++) {
70         int u, v;
```

```
71          scanf("%d%d", &u, &v);
72          addedge(u, v);
73          addedge(v, u);
74      }
75      dfs1(1, 0);
76      dfs2(1, 0, 1);
77      for (int i = 1; i <= n; i++) printf("%lld ", ans[i]);
78      printf("\n");
79      return 0;
80  }
```

3 DP

3.1 背包

```
1   const int MAXN = 10000;
2   const int SIZE = 100000;
3
4   int dp[SIZE];
5   int volume[MAXN], value[MAXN], c[MAXN];
6   int n, v;           //  总物品数, 背包容量
7
8   //  01背包
9   void ZeroOnepark(int val, int vol)
10  {
11      for (int j = v ; j >= vol; j--)
12      {
13          dp[j] = max(dp[j], dp[j - vol] + val);
14      }
15  }
16
17  //  完全背包
18  void Completepark(int val, int vol)
19  {
20      for (int j = vol; j <= v; j++)
21      {
22          dp[j] = max(dp[j], dp[j - vol] + val);
23      }
24  }
25
26  //  多重背包
27  void Multiplepark(int val, int vol, int amount)
28  {
29      if (vol * amount >= v)
30      {
31          Completepark(val, vol);
32      }
33      else
34      {
35          int k = 1;
36          while (k < amount)
37          {
38              ZeroOnepark(k * val, k * vol);
39              amount -= k;
40              k <<= 1;
41          }
42          if (amount > 0)
43          {
44              ZeroOnepark(amount * val, amount * vol);
45          }
46      }
47  }
48
49  int main()
```

```
{
    while (cin >> n >> v)
    {
        for (int i = 1 ; i <= n ; i++)
        {
            cin >> volume[i] >> value[i] >> c[i];       //    费用，价值，数量
        }
        memset(dp, 0, sizeof(dp));
        for (int i = 1; i <= n; i++)
        {
            Multiplepark(value[i], volume[i], c[i]);
        }
        cout << dp[v] << endl;
    }
    return 0;
}
```

3.2 树型 DP

没有上司的舞会
每个结点有权值，只有其父节点不在子集里自己才算贡献

```
const int maxn = 6e3 + 5;

vector<int> g[maxn];
int a[maxn], vis[maxn];
int rt;
int dp[maxn][2];
void fun(int u) {
    dp[u][0] = 0;
    dp[u][1] = a[u];
    for (auto v:g[u]) {
        fun(v);
        dp[u][0] += max(dp[v][0], dp[v][1]);
        dp[u][1] += dp[v][0];
    }
}
int main() {
    // freopen("in.txt", "r", stdin);
    int n;
    scanf("%d", &n);
    for (int i=1; i<=n; i++) scanf("%d", &a[i]);
    for (int i=1; i<n; i++) {
        int u, v;
        scanf("%d%d", &v, &u);
        g[u].push_back(v);
        vis[v] = 1;
    }
    for (int i=1; i<=n; i++) {
        if (!vis[i]) {
            rt = i;
            break;
        }
    }
    fun(rt);
    printf("%d\n", max(dp[rt][0], dp[rt][1]));
    return 0;
}
```

3.3 状压 DP

$n*n$ 的棋盘，放 k 个国王，使得两两不攻击的方案数, 国王可以攻击八个方向

```cpp
const int maxn = 2e5 + 5;

int cnt = 0;
int sta[1500], x[1500];
ll dp[15][1500][150];
int main() {
    // freopen("in.txt", "r", stdin);
    int n, K;
    cin >> n >> K;
    for (int i = 0; i < (1 << n); ++i) {
        if (i & (i << 1)) continue;
        int sum = 0;
        for (int j = 0; j < n; j++) {
            if (i & (1 << j)) sum++;
        }
        sta[++cnt] = i;
        x[cnt] = sum;
    }
    for (int i = 1; i <= cnt; i++) dp[1][i][x[i]] = 1;
    for (int i = 2; i <= n; i++) {
        for (int j = 1; j <= cnt; j++) {
            for (int k = 1; k <= cnt; k++) {
                if (sta[j] & sta[k]) continue;
                if (sta[j] & (sta[k] << 1)) continue;
                if ((sta[j] << 1) & sta[k]) continue;
                for (int s = K; s >= x[j]; s--) dp[i][j][s] += dp[i - 1][k][s - x[
                  j]];
            }
        }
    }
    ll ans = 0;
    for (int i = 1; i <= cnt; i++) ans += dp[n][i][K];
    cout << ans << '\n';
    return 0;
}
```

3.4 数位 DP

3.4.1 HDU 2089 不要 62

```cpp
// 求区间[n, m]中不含4和62的数字个数
// dp[pos][sta]:在第pos位中，前一位是否为6的满足条件的数字个数。
// sta: 0表示前一位不为6, 1:表示前一位为6

#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int, int> P;
const int maxn = 1e6+5;
const int inf = 0x3f3f3f3f;
const int mod = 1e9+7;
const double PI = acos(-1);


int n, m;
int dp[10][2];
int num[10];

int dfs(int pos, int pre, int sta, bool limit)
{
    if(pos==-1)return 1;
    if(!limit && dp[pos][sta]!=-1)return dp[pos][sta];
    int up = limit?num[pos]:9;
```

```
    int tmp = 0;
    for(int i=0;i<=up;i++)
    {
        if(pre==6 && i==2)continue;
        if(i==4)continue;
        tmp += dfs(pos-1, i, i==6, limit&&i==num[pos]);
    }
    if(!limit) dp[pos][sta] = tmp;
    return tmp;
}

int slove(int x)
{
    int pos = 0;
    while(x)
    {
        num[pos++] = x%10;
        x/=10;
    }
    return dfs(pos-1, -1, 0, true);
}

int main()
{
    while(~scanf("%d%d", &n, &m) && n && m)
    {
        memset(dp, -1, sizeof(dp));
        printf("%d\n", slove(m) - slove(n-1));
    }
    return 0;
}
```

3.4.2 SCOI 2009 Windy 数

```
// 求区间[l, r]中不含前导零且相邻位之差的绝对值大于等于2的数的个数

#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int, int> P;
const int maxn = 1e6+5;
const int inf = 0x3f3f3f3f;
const int mod = 1e9+7;
const double PI = acos(-1);


ll l, r;
ll a[20], dp[20][20];

ll dfs(int pos, int pre, int sta, bool limit)
{
    if(pos==-1)return 1;
    if(!limit && dp[pos][pre]!=-1)return dp[pos][pre];
    int up = limit ? a[pos] : 9;
    ll ans= 0;
    for(int i=0;i<=up;i++)
    {
        if(sta!=0)
        {
            ans += dfs(pos-1, i, sta&&(i==0), limit&&(i==up));
        }
        else
        {
            if(abs(pre-i)>=2)
```

```
32              {
33                  ans += dfs(pos-1, i, sta&&(i==0), limit&&(i==up));
34              }
35          }
36      }
37      if(!limit&&pre!=0) dp[pos][pre] = ans;
38      return ans;
39  }
40
41  ll slove(int x)
42  {
43      int pos = 0;
44      while(x)
45      {
46          a[pos++] = x%10;
47          x/=10;
48      }
49      return dfs(pos-1, 0, 1, true);
50  }
51
52  int main()
53  {
54      memset(dp, -1, sizeof(dp));
55      scanf("%lld%lld", &l, &r);
56      printf("%lld\n", slove(r) - slove(l-1));
57      return 0;
58  }
```

3.5 区间 DP

3.5.1 51Nod 1021 石子合并

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef pair<int, int> P;
5  const int maxn = 1e5+5;
6  const int inf = 0x3f3f3f3f;
7  const int mod = 1e9+7;
8  const double PI = acos(-1);
9
10 int n;
11 int num[105];
12 int sum[105];
13 int dp[105][105];
14 int main()
15 {
16     memset(dp, inf, sizeof(dp));
17     cin>>n;
18     for(int i=1;i<=n;i++)
19     {
20         cin>>num[i];
21         sum[i] = sum[i-1]+num[i];
22         dp[i][i] = 0;
23     }
24
25     for(int len=1;len<=n;len++)
26     {
27         for(int i=1;i<=n-len+1;i++)
28         {
29             int end = i+len-1;
30             for(int j=i;j<end;j++)
31             {
32                 dp[i][end] = min(dp[i][end], dp[i][j] + dp[j+1][end]+sum[end]-sum[
                    i-1]);
```

```
33              }
34          }
35      }
36      cout<<dp[1][n]<<endl;
37      return 0;
38  }
```

3.6 最长公共递增子序列

```
1  /*
2  *  最长公共递增子序列 O(n^2)
3  *  f记录路径,DP记录长度, 用a对b扫描,逐步最优化。
4  */
5  const int N = 1010;
6
7  int f[N][N], dp[N];
8
9  int gcis(int a[], int la, int b[], int lb, int ans[])
10 {   //  a[1...la], b[1...lb]
11     int i, j, k, mx;
12     memset(f, 0, sizeof(f));
13     memset(dp, 0, sizeof(dp));
14     for (i = 1; i <= la; i++)
15     {
16         memcpy(f[i], f[i-1], sizeof(f[0]));
17         for (k = 0, j = 1; j <= lb; j++)
18         {
19             if (b[j - 1] < a[i - 1] && dp[j] > dp[k])
20             {
21                 k = j;
22             }
23             if (b[j - 1] == a[i - 1] && dp[k] + 1 > dp[j])
24             {
25                 dp[j] = dp[k] + 1,
26                 f[i][j] = i * (lb + 1) + k;
27             }
28         }
29     }
30     for (mx = 0, i = 1; i <= lb; i++)
31     {
32         if (dp[i] > dp[mx])
33         {
34             mx = i;
35         }
36     }
37     for (i = la * lb + la + mx, j = dp[mx]; j; i = f[i / (lb + 1)][i % (lb + 1)],
         j--)
38     {
39         ans[j - 1] = b[i % (lb + 1) - 1];
40     }
41     return dp[mx];
42 }
```

3.7 最长公共子序列

```
1  const int N = 1010;
2
3  int a[N][N];
4
5  int LCS(const char *s1, const char *s2)
6  {   //  s1:0...m, s2:0...n
7      int m = (int)strlen(s1), n = (int)strlen(s2);
8      int i, j;
```

```
    a[0][0] = 0;
    for (i = 1; i <= m; ++i)
    {
        a[i][0] = 0;
    }
    for (i = 1; i <= n; ++i)
    {
        a[0][i] = 0;
    }
    for (i = 1; i <= m; ++i)
    {
        for (j = 1; j <= n; ++j)
        {
            if (s1[i - 1] == s2[j - 1])
            {
                a[i][j] = a[i - 1][j - 1] + 1;
            }
            else if (a[i - 1][j] > a[i][j - 1])
            {
                a[i][j]= a[i - 1][j];
            }
            else
            {
                a[i][j] = a[i][j - 1];
            }
        }
    }
    return a[m][n];
}
```

3.8 最长有序子序列

```
int n;
int a[maxn]; // 原数组
int s[maxn]; // 记录LIS
int LIS()
{
    int top = 0;
    for(int i=0;i<n;i++)
    {
        int pos = upper_bound(s, s+top, a[i])-s;
        s[pos] = a[i];
        top = max(top, pos+1);
    }
    return top;
}
```

```
/*
*  递增（默认）
*  递减
*  非递增
*  非递减 (1)>= && <  (2)<  (3)>=
*/
const int MAXN = 1001;

int a[MAXN], f[MAXN], d[MAXN];   //  d[i] 用于记录 a[0...i] 以 a[i] 结尾的最大长度

int bsearch(const int *f, int size, const int &a)
{
    int l = 0, r = size - 1;
    while (l <= r)
    {
        int mid = (l + r) / 2;
        if (a > f[mid - 1] && a <= f[mid])  //  (1)
```

```
18          {
19              return mid;
20          }
21          else if (a < f[mid])
22          {
23              r = mid - 1;
24          }
25          else
26          {
27              l = mid + 1;
28          }
29      }
30      return -1;
31  }
32
33  int LIS(const int *a, const int &n)
34  {
35      int i, j, size = 1;
36      f[0] = a[0];
37      d[0] = 1;
38      for (i = 1; i < n; ++i)
39      {
40          if (a[i] <= f[0])               //  (2)
41          {
42              j = 0;
43          }
44          else if (a[i] > f[size - 1])    //  (3)
45          {
46              j = size++;
47          }
48          else
49          {
50              j = bsearch(f, size, a[i]);
51          }
52          f[j] = a[i];
53          d[i] = j + 1;
54      }
55      return size;
56  }
57
58  int main()
59  {
60      int i, n;
61      while (scanf("%d", &n) != EOF)
62      {
63          for (i = 0; i < n; ++i)
64          {
65              scanf("%d", &a[i]);
66          }
67          printf("%d\n", LIS(a, n));      // 求最大递增 / 上升子序列(如果为最大非降
              子序列,只需把上面的注释部分给与替换)
68      }
69      return 0;
70  }
```

3.9 优化

3.9.1 四边形不等式优化

对于任意的 $l_1 \le l_2 \le r_1 \le r_2$，均有 $w(l_1,r_1)+w(l_2,r_2) \le w(l_1,r_2)+w(l_2,r_1)$ 成立，则满足四边形不等式

```
1  const int maxn = 1e4 + 5;
2
3  int a[205], pre[205];
4  int dp2[205][205], m[205][205];
```

```
5
6  int main() {
7      // freopen("in.txt", "r", stdin);
8      int n;
9      cin >> n;
10     for (int i = 1; i <= n; i++) {
11         cin >> a[i];
12         a[i + n] = a[i];
13     }
14     for (int i = 1; i <= (n<<1); i++) {
15         pre[i] = pre[i - 1] + a[i];
16         m[i][i] = i;
17     }
18     for (int i = (n<<1); i >= 1; i--) {
19         for (int j = i + 1; j <= (n<<1); j++) {
20             dp2[i][j] = inf;
21             for (int k = m[i][j - 1]; k <= m[i + 1][j]; k++) {
22                 if (dp2[i][j] > dp2[i][k] + dp2[k + 1][j] + pre[j] - pre[i - 1]) {
23                     dp2[i][j] = dp2[i][k] + dp2[k + 1][j] + pre[j] - pre[i - 1];
24                     m[i][j] = k;
25                 }
26             }
27         }
28     }
29     int mi = inf;
30     for (int i = 1, j = n; j <= (n<<1); i++, j++) {
31         mi = min(mi, dp2[i][j]);
32     }
33     cout << mi << '\n';
34     return 0;
35 }
```

4 字符串

4.1 KMP

```
1  // 能够获得t在s中出现的所有位置
2
3  #include<bits/stdc++.h>
4  using namespace std;
5  typedef long long ll;
6  typedef unsigned long long ull;
7  typedef pair<int, int> P;
8  const int maxn = 1e6+5;
9  const int inf = 0x3f3f3f3f;
10 const int mod = 1e9+7;
11 const double PI = acos(-1);
12
13
14 char s[maxn], t[maxn];
15 int ne[maxn];
16 int ls, lt;
17
18 void init()
19 {
20     scanf("%s", s+1);
21     scanf("%s", t+1);
22     ls = strlen(s+1);
23     lt = strlen(t+1);
24 }
25
26 void get_ne()
27 {
28     int j = 0;
```

```
    for(int i=2;i<=lt;i++)
    {
        while(j && t[i]!=t[j+1])
            j = ne[j];
        if(t[i]==t[j+1])j++;
        ne[i] = j;
    }
}

void kmp()
{
    int j=0;
    vector<int> v;
    for(int i=1;i<=ls;i++)
    {
        while(j>0 && s[i]!=t[j+1])j = ne[j];
        if(s[i]==t[j+1])j++;
        if(j==lt)
        {
            v.push_back(i-lt+1);
            j = ne[j];
        }
    }
    for(auto x : v)printf("%d\n", x);
}

int main()
{
    init();
    get_ne();
    kmp();
    return 0;
}
```

4.2 EXKMP

```
int nxt[maxn], extend[maxn];
char s[maxn], t[maxn];

void getNext()
{
    int n = strlen(t);
    nxt[0] = n;
    int a=0, p=0;
    for(int i=1;i<n;i++)
    {
        if(i>=p || i+nxt[i-a]>=p)
        {
            if(i>=p)p=i;
            while(p<n && t[p]==t[p-i])p++;
            nxt[i] = p-i;
            a=i;
        }
        else
            nxt[i] = nxt[i-a];
    }
}

void getExtend()
{
    int n = strlen(s), m = strlen(t);
    int a=0, p=0;
    getNext();
    for(int i=0;i<n;i++)
    {
```

```
30          if(i>=p || i+nxt[i-a]>=p)
31          {
32              if(i>=p)p=i;
33              while(p<n && p-i<m && s[p]==t[p-i])p++;
34              extend[i] = p-i;
35              a=i;
36          }
37          else
38              extend[i] = nxt[i-a];
39      }
40  }
41
42  int main()
43  {
44      scanf("%s", s);
45      scanf("%s", t);
46      getExtend();
47      int lens = strlen(s), lent = strlen(t);
48      for(int i=0;i<lent;i++)
49          printf("%d ", nxt[i]);
50      printf("\n");
51      for(int i=0;i<lens;i++)
52          printf("%d ", extend[i]);
53      printf("\n");
54      return 0;
55  }
```

4.3 字符串 Hash

4.3.1 自然溢出

```
1  typedef unsigned long long ull;
2  const int maxn = 1e5 + 10;
3  const ull base = 131;
4  ull p[maxn], has[maxn];
5  // 获取s[l-r]的哈希值
6  ull get(int l, int r) {
7      return has[r] - has[l - 1] * p[r - l + 1];
8  }
9  void gethash(string s) {
10     len = s.length();
11     p[0] = 1; has[0] = 0;
12     for (int i = 1; i <= len; i++) {
13         p[i] = p[i - 1] * base;
14         has[i] = has[i - 1] * base + (ull)(s[i - 1]);
15     }
16 }
```

4.4 字典树 Trie

```
1  int tree[maxn][30]; //字典树，有maxn个结点，30表示字符集的大小，比如小写字母就26个
2  int isend[maxn]; //表示结点i是否为某个串的结尾
3  int tot; //节点数
4
5  void insert(char *s)
6  {
7      // 根节点为0号结点
8      int root=0, len=strlen(s);
9      for(int i=0;i<len;i++)
10     {
11         // pos是指当前字符应该为root的哪一个儿子
12         int pos = s[i]-'a';
13         // 如果不存在，则新创建
```

```
14          if(!tree[root][pos])
15              tree[root][pos] = ++tot;
16          // root往下走
17          root = tree[root][pos];
18      }
19      // 标记当前编号的结点是某个字符串的结尾
20      isend[root] = 1;
21  }
22
23  int find(char *s)
24  {
25      int root=0, len=strlen(s);
26      for(int i=0;i<len;i++)
27      {
28          int pos = s[i]-'a';
29          root = tree[root][pos];
30          // 不存在则返回0
31          if(!root)return 0;
32      }
33      return 1;
34  }
```

4.5 Manacher

```
1   char s[maxn];
2   char t[2*maxn];
3   int p[2*maxn];
4   int manacher()
5   {
6       int len = strlen(s);
7       // 预处理字符串
8       int l=0;
9       t[l++] = '$';
10      t[l++] = '#';
11      for(int i=0;i<len;i++)
12      {
13          t[l++] = s[i];
14          t[l++] = '#';
15      }
16      t[l] = '@';
17      int id = 0, mx = 0;
18      int maxlen = 1;
19      for(int i=0;i<l;i++)
20      {
21          // 预设p[i]的值
22          p[i] = mx>i ? min(p[2*id-i], mx-i) : 1;
23          // 朴素算法向左右拓展，因为处理后的字符串头和尾都是特殊字符，所以不会越界
24          while(t[i+p[i]] == t[i-p[i]])p[i]++;
25          // 能否更新mx
26          if(mx < p[i]+i)
27          {
28              mx = p[i]+i;
29              id = i;
30          }
31          // 更新最长回文子串的长度
32          if(maxlen < p[i]-1)
33              maxlen = p[i]-1;
34      }
35      return maxlen;
36  }
```

4.6 最小表示法

```
int getmin()
{
    int i=0, j=1, k=0, n=strlen(s);
    while(i<n && j<n && k<n)
    {
        int t = s[(i+k)%n]-s[(j+k)%n];
        if(!t)
            k++;
        else
        {
            if(t>0)
                i += k+1;
            else
                j += k+1;
            if(i==j)
                j++;
            k=0;
        }
    }
    return min(i, j);
}
```

4.7 最大表示法

```
int getmax()
{
    int i=0, j=1, k=0, n=strlen(s);
    while(i<n && j<n && k<n)
    {
        int t = s[(i+k)%n]-s[(j+k)%n];
        if(!t)
            k++;
        else
        {
            if(t<0)
                i += k+1;
            else
                j += k+1;
            if(i==j)
                j++;
            k=0;
        }
    }
    return min(i, j);
}
```

5 数学

5.1 欧拉函数

定义：欧拉函数 phi(x) 代表小于等于 x 的数中和 x 互质的数的个数 (小于显然只对 1 成立)，比如说小于等于 9 的数中与 9 互质的有 1,2,4,5,7,8, 则 phi(9)=6. 以 phi(x) 表示小于等于 x 的数中与 x 互质的数的个数可以这样得到：$phi(x) = x * (1 - 1/p_1) * (1 - 1/p_2) * (1 - 1/p_3) \cdots * (1 - 1/p_n)$ 其中 p1,p2,p3…pn 是 x 的所有质因数，每个质因数只使用一次，使用上述公式，则 phi(9)=9*(1-1/3)=6.

特性: 1. 若 a 为质数,$phi[a] = a - 1$;
2. 若 a 为质数,$b\%a = 0, phi[a * b] = phi[b] * a$
3. 若 a,b 互质,$phi[a * b] = phi[a] * phi[b]$(当 a 为质数时,$if b mod a != 0, phi[a * b] = phi[a] * phi[b]$)
4. 若 p 为质数，$phi(p^k) == p^k - p^{k-1} == (p - 1) * p^{k-1}$
5. 当 n>2 时，phi(n) 是偶数
6.phi(1)=1; （具体情况看题目，有时题目会要求为 0)

欧拉筛法：欧拉筛法的原理如下，观察 phi(x) 的求解式子，首先令 phi(x)=x, 然后仿照素数筛法，如果 x 能被 2,3,5,7…. 这些素数筛到 (即素数 prime 是 x 的因子)，则执行 phi(x)=phi(x)*(1-1/prime)

```
// 埃拉托斯特尼筛求欧拉函数值
const int maxn = 1000000;
int phi[maxn+1];
bool isPrime[maxn+1];
void Eular()
{
    for(int i=1;i<=maxn;i++) phi[i]=i;
    memset(isPrime,true,sizeof(isPrime));
    isPrime[0]=isPrime[1]=false;
    phi[1]=1;
    for(int i=2;i<=maxn;i++)
    {
        // i是质数
        if(isPrime[i])
        {
            for(int j=i;j<=maxn;j+=i)
            {
                // i与i的倍数都更新一遍
                isPrime[j]=false;
                phi[j] -= phi[j]/i;   // phi(x)=phi(x)*(1-1/prime) = phi(x) - phi(x)/prime
            }
        }
    }
}


// 欧拉筛求欧拉函数，时间复杂度接近O(n)
const int maxn = 1100000;
int num = 0;
int phi[maxn+1];
int prime[maxn+1];
int flag[maxn+1];

void eular()
{
    memset(flag, 0, sizeof(flag));
    phi[1]=1;//1要特判
    for (int i=2;i<=maxn;i++)
    {
        if (flag[i]==0)//这代表i是质数
        {
            prime[++num]=i;
            phi[i]=i-1;
        }
        for (int j=1;j<=num&&prime[j]*i<=maxn;j++)//经典的欧拉筛写法
        {
            flag[i*prime[j]]=1;//先把这个合数标记掉
            if(i%prime[j]==0)
            {
                phi[i*prime[j]]=phi[i]*prime[j];//若prime[j]是i的质因子，则根据计算公式，i已经包括i*prime[j]的所有质因子
                break;//经典欧拉筛的核心语句，这样能保证每个数只会被自己最小的因子筛掉一次
            }
            else phi[i*prime[j]]=phi[i]*phi[prime[j]];//利用了欧拉函数是个积性函数的性质
        }
    }
}
```

5.2 枚举约数

```
// 枚举约数(普通)
vector<int> factor(int n)
{
    vector<int> a;
    for(int i = 2; i*i <= n; i++){
        if((n%i)==0){
            a.push_back(i);
            if((n/i)!=i)a.push_back(n/i);//根号n的情况不要重复添加
        }
    }
    return a;
}

特殊例题：f(n)为n所有约数的和，给你一个数n,让你求从1到n中f(n)为偶数的数有多少个
#include <bits/stdc++.h>
#define ll long long
#define INF 0x3f3f3f3f

using namespace std;

int main()
{
    int t, cas = 1;
    long long n, sum;
    cin >> t;
    while(t--)
    {
        scanf("%lld", &n);
        sum = n;
        sum -= (int)sqrt(n);
        sum -= (int)sqrt(n/2);
        printf("Case %d: %lld\n", cas++, sum);
    }
    return 0;
}
```

5.3 全错排

```
问题：n封信给n个人，问全都送错的可能性有多少种
#include<iostream>
#include<stdio.h>
using namespace std;
int main()
{
    long long int arr[21];
    int num,i;
    arr[1]=0; arr[2]=1;
    for(i=3;i<21;i++)
        arr[i]=(i-1)*(arr[i-1]+arr[i-2]);    // d[n]= (n-1)*( d[n-1] + d[n-2])
    while(scanf("%d",&num)!=EOF)
    {
        cout<<arr[num]<<endl;
    }
    return 0;
}
```

5.4 唯一分解定理

算术基本定理可表述为：任何一个大于 1 的自然数 N, 如果 N 不为质数，那么 N 可以唯一分解成有限个质数的乘积 $N = P_1^{a_1} * P_2^{a_2} * P_3^{a_3} \cdots P_n^{a_n}$，这里 P1<P2<P3.....<Pn 均为质数，其中指数 ai 是正整数。

(1) 一个大于 1 的正整数 N，如果它的标准分解式为: $N = P_1^{a_1} P_2^{a_2} \cdots P_n^{a_n}$ ，那么它的正因数个数为 $\sigma_0(N) = (1+a_1)(1+a_2)\cdots(1+a_n)$

(2) 它的全体正因数之和为

$$\sigma_1(N) = (1+p_1+p_1^2+\cdots+p_1^{a_1})(1+p_1+p_2^2+\cdots+p_2^{a_2})\cdots(1+p_n+p_n^2+\cdots+p_n^{a_n})$$

当 $\sigma_1(N) = 2N$ 时就称 N 为完全数

5.4.1 例题一

求出有几种边长为整数，面积等于 a 的矩形，且矩形的短边不小于 b

```
1  //考点: 素数筛选 + 唯一分解定理
2  #include <bits/stdc++.h>
3  #define ll long long
4  #define INF 0x3f3f3f3f
5
6  using namespace std;
7
8  const int MAX = 1e6 + 10; // 边长
9  int prime[MAX], k; // k为全局变量, 指向最后一个素数的后一个单元
10 bool isPrime[MAX];
11
12 // 素数筛选
13 void Prime()
14 {
15     k = 0;
16     memset(isPrime, true, sizeof(isPrime));
17     isPrime[1] = false; // 1不是素数
18     for(int i = 2; i < MAX; i++)
19     {
20         if(isPrime[i])
21         {
22             prime[k++] = i;
23             for(int j = 2 ; i * j < MAX ; j++)
24             {
25                 isPrime[i * j] = false;
26             }
27         }
28     }
29 }
30
31 // 唯一分解定理找n的因子个数(N的因子个数 M = (1 + a1)*(1 + a2)*(1 + a3)*...*(1 +
     an))
32 long long solve(long long n)
33 {
34     long long ans = 0, sum = 1;
35     for(long long i = 0; i < k && prime[i]*prime[i] <= n; i++)
36     {
37         if(n % prime[i] == 0) // 从小到大寻找素数因子
38         {
39             ans = 0;
40             while(n % prime[i] == 0) // 计算素数因子次数
41             {
42                 ans++;
43                 n /= prime[i];
44             }
45             sum *= (1 + ans);
46         }
47     }
48     if(n > 1) sum *= 2; // 如果处理到最后n还不是1, 则最后剩下的一定是个素数
49     return sum;
50 }
51
52 int main()
53 {
```

```
54      Prime();
55      int t, x = 0;
56      long long ab, a, num;
57      scanf("%d", &t);
58      while(t--)
59      {
60          x++;
61          scanf("%lld%lld", &ab, &a);
62          if(ab < a * a)
63          {
64              printf("Case %d: 0\n", x);
65              continue;
66          }
67          num = solve(ab); // 获得因子数(一个数的因子包括他本身)
68          num /= 2;        // 因子对数
69          for(long long i = 1 ; i < a ; i++)
70          {
71              if(ab % i == 0) num--; //将边小于a的情况减去
72          }
73          printf("Case %d: %lld\n", x, num);
74      }
75      return 0;
76  }
```

5.4.2 例题二

给你一个数 $x = b^p$, 求 p 的最大值

$$x = p_1^{x_1} * P_2^{x_2} * \cdots * P_s^{x_s}$$

题目要求 $x = b^p$,x 只有一个因子的 p 次幂构成如果 $x = 12 = 2^2 * 3^1$, 要让 $x = b^p$, 即 12 应该是 $12 = 12^1$ 所以 $p = gcd(x_1, x_2, x_3, \cdots, x_s)$; 比如：$24 = 2^3 * 3^1$,p 应该是 gcd(3, 1)=1, 即 $24 = 24^1$ $324 = 3^4 * 2^2$,p 应该是 gcd(4, 2)=2, 即 $324 = 18^2$ 本题有一个坑，就是 x 可能为负数，如果 x 为负数的话，$x = b^q$, q 必须是奇数，所以将 x 转化为正数求得的解如果是偶数的话必须将其循环除 2 直到出现奇数

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   const int maxn=1e6+5;   // 卡内存
5   bool isprime[maxn];
6   int prime[maxn], psize=0;
7
8   // 素数筛选
9   void getPrimes()
10  {
11      memset(isprime, true, sizeof(isprime));
12      isprime[0] = isprime[1] = false;
13      for (int i = 2; i < maxn; i++) if (isprime[i])
14      {
15          prime[psize++] = i;
16          for(int j = 2; i * j < maxn; j++)
17              isprime[i*j] = false;
18      }
19  }
20
21  // 辗转相除
22  int gcd(int a, int b)
23  {
24      return b == 0 ? a : gcd(b, a%b);
25  }
26
27  int main()
28  {
29      int t, cas = 1;
30      long long n;
31      getPrimes();
32      scanf("%d", &t);
```

```
33      while (t--)
34      {
35          scanf("%lld", &n);
36          int f = 0;
37          if(n < 0)
38          {
39              n = - n;
40              f = 1;
41          }
42          int res, ans = 0;
43          for(int i = 0 ; i < psize && prime[i] * prime[i] <= n ; i++)
44          {
45              if(n % prime[i] == 0)
46              {
47                  res = 0;
48                  while(n % prime[i] == 0)
49                  {
50                      res++;
51                      n /= prime[i];
52                  }
53                  if(ans == 0)
54                      ans = res;
55                  else
56                      ans = gcd(ans, res);
57              }
58          }
59          if(n > 1) ans = gcd(ans, 1); // 如果分解完后剩下的n大于1，则剩下的一定是一
              个素数
60          if(f == 1)
61          {
62              while(ans % 2 == 0)
63                  ans /= 2; // 如果n为负数且最后算出来指数为偶数，则循环除2直到变为
                      奇数
64          }
65          printf("Case %d: %d\n", cas++, ans);
66      }
67      return 0;
68  }
```

5.5 快速幂

```
1  typedef long long ll;
2  ll pow(ll x,ll n,ll mod)
3  {
4      ll res=1;
5      while(n>0)
6      {
7          if(n&1)
8          {
9              res=res*x;
10             res=res%mod;
11         }
12         x=x*x;
13         x=x%mod;
14         n>>=1;
15     }
16     return res;
17 }
18
19 例题：给你一个n，一个k，求n^k的结果的前三位与后三位
20 //考点：快速幂取模(可获得后三位) + 数论运算(取前三位)
21
22 /**
23 * 设10^w=n^k，两边同时取log10，那么w=k*log10(n)。
```

```
* 再设x=(int)w（w的整数部分），y=w-x（w的小数部分），那么10^w=10^(x+y)=10^x*10^y=n^k;
* 由于10^x是10的倍数，那么10^y=n^k/10^x，因为0<y<1,所以10^y为一位数，即最前边那一位，所以求出y来
*/

#include <bits/stdc++.h>
#define ll long long
#define INF 0x3f3f3f3f

using namespace std;

// 快速幂取模
ll pow_mod(ll a, ll b, ll mod)
{
    int ans = 1;
    int base = a % mod;
    while(b){
        if(b & 1) ans = (ans*base) % mod;
        base = (base*base) % mod;
        b >>= 1;
    }
    return ans;
}

int main()
{
    ll n, k;
    ll ans1, ans2;
    int t, cas = 1;
    scanf("%d", &t);
    while(t--)
    {
        scanf("%lld%lld", &n, &k);
        ans2 = pow_mod(n, k, 1000);   // 取后三位
        double w = k*log10(n);
        w = w-(ll)w;
        ll ans1=(ll)(pow(10,w)*100);  // 取前三位
        printf("Case %d: %lld %03lld\n", cas++, ans1, ans2);   // 后三位要补前导零
    }
    return 0;
}
```

5.6 矩阵快速幂

```
const int N=9;
const int mod = 1e9 + 7;
struct Matrix{///矩阵结构体
    ll matrix[N][N];
};

void init(Matrix &res)///初始化为单位矩阵
{
    memset(res.matrix, 0, sizeof(res.matrix));
    for(int i = 0; i < N; i++)
        res.matrix[i][i] = 1;
}
Matrix multiplicative(Matrix a,Matrix b)///矩阵乘法
{
    Matrix res;
    memset(res.matrix, 0, sizeof(res.matrix));
    for(int i = 0 ; i < N ; i++){
        for(int j = 0 ; j < N ; j++){
            for(int k = 0 ; k < N ; k++){
                res.matrix[i][j] += a.matrix[i][k]*b.matrix[k][j];
```

```
                res.matrix[i][j] %= mod;
            }
        }
    }
    return res;
}
Matrix Pow(Matrix mx, ll m)///矩阵快速幂
{
    Matrix res, base=mx;
    init(res);
    while(m) {
        if (m & 1)
            res = multiplicative(res,base);
        base = multiplicative(base,base);
        m >>= 1;
    }
    return res;
}
```

5.7 欧拉降幂

定理：
欧拉定理：若 gcd(a, p) = 1, 则 $a^p \equiv a^{b\%\varphi(P)}(mod \quad P)$，其中 gcd(a, p) = 1。
拓展欧拉定理：假设 a 为任意数，b 和 m 为正整数，且 $b > \varphi(m)$，a 和 m 不一定要互质，那么有如下公式：

$$a^b \equiv \begin{cases} a^{bMod\phi(m)} & gcd(a,m) = 1 \\ a^b & gcd(a,m) \neq 1 \bigwedge b < \phi(m) \\ a^{bMod\phi(m)+\phi(m)} & gcd(a,m) \neq 1 \bigwedge b \geq \phi(m) \end{cases} (Mod \quad m)$$

5.7.1 例题一

```
/*
Given A,B,C, You should quickly calculate the result of A^B mod C. (1<=A,C
  <=1000000000,1<=B<=10^1000000).（求(a^b)%c,其中b非常大,是个大数）
*/
#include <cstdio>
#include <iostream>
#include <cstring>
#include <cmath>
using namespace std;
typedef long long ll;
const int MAX=1000100;
ll  fastPow(ll  a,ll b,ll mod)
{
    ll ans=1;
    a %= mod;
    while(b)
    {
        if(b&1)
        {
            ans = (ans*a)%mod;
        }
        b >>= 1;
        a = (a*a)%mod;
    }
    return ans;
}
ll  eulerFunction(ll x)
{
    ll eulerNumbers = x;
    for(ll i = 2; i*i <= x; i++)
    {
        if(x % i == 0)
        {
```

```
33              eulerNumbers = eulerNumbers / i * (i-1);
34              while(x % i == 0)
35              {
36                  x /= i;
37              }
38          }
39      }
40      if(x > 1)
41      {
42          eulerNumbers = eulerNumbers / x * (x-1);
43      }
44      return eulerNumbers;
45  }
46  ll eulerDropPow(ll a,char b[],ll c)
47  {
48      ll eulerNumbers = eulerFunction(c);
49      ll descendingPower=0;
50      for(ll i=0,len = strlen(b); i<len; ++i)
51      {
52          descendingPower=(descendingPower*10+b[i]-'0') % eulerNumbers;
53      }
54      descendingPower += eulerNumbers;
55      return fastPow(a,descendingPower,c);
56  }
57  int main()
58  {
59      ll a,c;
60      char b[MAX];
61      while(~scanf("%lld%s%lld",&a,b,&c))
62      {
63          printf("%lld\n",eulerDropPow(a,b,c));
64      }
65      return 0;
66  }
```

5.8 广义欧拉降幂

5.8.1 例题一

给定 a 和 K 以及 m，试求出

$$a^{a^{a^{\cdots}}} \ mod \ m$$

其中共 K 层幂塔。其中 a，K 和 m 小于等于 1e6
1. 首先，当 a = 1 或者 b = 0 时特判，得出答案为 1%m
2. 本题中 b = f(a, k-1,INF)，如果 $a \geq phi(m)$，那么显然 b 一定大于 phi(m)，即满足拓展欧拉定理中第 3 种情况
3. 如果 k = 1，则 b = f(a,k-1,INF) = 1，此时只需判断 phi(m) 是否大于 1 即可判断当然情况符合拓展欧拉定理第 3 还是第 2 种情况
4. 剩下的情况我们就可以通过递归来判断 b 是否大于 phi(m)，因为若 $b \geq phi(m)$，那么 $\log_a b \geq log_a phi(m)$

```
1   #include<cstdio>
2   #include<cmath>
3   typedef long long ll;
4   const int N = 1e6+10;
5   ll qpow(ll a,ll b,ll m){
6       ll res = 1;
7       while(b){
8           if(b&1) res = res*a%m;
9           a = a*a%m;
10          b >>= 1;
11      }
12      return res;
13  }
14  int v[N],primes[N],phi[N];
```

```
int init(){
    int cnt = 0;
    for(int i = 2;i < N;i++){
        if(!v[i]){
            primes[cnt++] = i;
            v[i] = i;
            phi[i] = i-1;
        }
        for(int j = 0;j < cnt;j++){
            if(primes[j] > v[i] || primes[j]*i >= N)
                break;
            v[i*primes[j]] = primes[j];
            phi[i*primes[j]] = phi[i]*(i%primes[j] ? primes[j]-1 : primes[j]);
        }
    }
}
ll gcd(ll a,ll b){
    if(b == 0) return a;
    return gcd(b,a%b);
}
bool check(ll a,ll b,ll p){
    if(b == 0) return p <= 1;   //f(a,0,p) = 1
    if(a >= p) return true;//f(a,b,INF) > p
    return check(a,b-1,log(p)/log(a));
}
ll f(ll a,ll b,ll m){
    if(m == 1) return 0;        //递归终止条件1,此时之后答案恒为0
    if(b <= 1) return qpow(a,b,m);//递归终止条件2
    ll ph = phi[m];
    //  printf("%lld\n",ph);
    if(gcd(a,m) == 1) return qpow(a,f(a,b-1,ph),m); //欧拉定理
    if(check(a,b-1,ph)) return qpow(a,f(a,b-1,ph)+ph,m);//拓展欧拉定理情况1
    return qpow(a,f(a,b-1,ph),m);       //拓展欧拉定理情况2
}
ll a,b,m;
int t;
int main(){
    scanf("%d",&t);
    init();
    while(t--){
        scanf("%lld%lld%lld",&a,&b,&m);
        printf("%lld\n",f(a,b,m));
    }
    return 0;
}
```

5.8.2 例题二

给定长度为 n 的正整数序列和模数 m，q 次询问区间 [l,r] 累乘幂%m 的答案。$n,q \neq 10^5, m, a_i \neq 10^9$。

$$w_1^{(w_2^{(w_3^{(\dots w_m)})})}$$

```
#include <bits/stdc++.h>
#define aaa cout<<233<<endl;
#define endl '\n'
#define pb push_back
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;
// mt19937 rnd(time(0));
const int inf = 0x3f3f3f3f;//1061109567 > 1e9
const ll linf = 0x3f3f3f3f3f3f3f3f;
const double eps = 1e-7;
```

```
13 const double pi = 3.14159265358979;
14 const int maxn = 1e5 + 5;
15 const int maxm = 1e3 + 5;
16 const int mod = 1e9 + 7;
17
18 ll a[maxn];
19
20 inline ll modulo(ll x, ll mod){return x < mod ? x : x % mod + mod;}
21 inline ll pow_(ll a, ll b, ll p)
22 {
23     ll ret = 1;
24     while(b)
25     {
26         if(b & 1)ret = modulo(ret * a, p);
27         a = modulo(a * a, p);
28         b >>= 1;
29     }
30     return ret;
31 }
32 unordered_map<ll, ll> phi_;
33 inline ll phi(ll x)
34 {
35     if(phi_[x])return phi_[x];
36     ll ans = x;
37     ll t = x;
38     for(ll i = 2; i * i <= x; ++i)
39     {
40         if(x % i == 0)
41         {
42             ans = ans / i * (i - 1);
43             while(x % i == 0)x /= i;
44         }
45     }
46     if(x > 1)ans = ans / x * (x - 1);
47     phi_[t] = ans;
48     return ans;
49 }
50 //这里根据题意来更改，k表示共有k个指数
51 ll f(ll a, ll b, ll k, ll p)
52 {
53     if(p == 1)return 1;
54     if(k == 0)return 1;
55     return pow_(a, f(a, a, k - 1, phi(p)), p);
56 }
57 ll f(ll l, ll r, ll p)
58 {
59     if(p == 1)return 1;
60     if(l == r + 1)return 1;
61     return pow_(a[l], f(l + 1, r, phi(p)), p);
62 }
63
64 int main()
65 {
66     // double pp = clock();
67     // freopen("233.in", "r", stdin);
68     // freopen("233.out", "w", stdout);
69     ios_base::sync_with_stdio(0);
70     cin.tie(0);cout.tie(0);
71
72     ll n, m;
73     cin >> n >> m;
74     for(int i = 1; i <= n; ++i)cin >> a[i];
75     int q; cin >> q;
76     while(q--)
77     {
78         ll l, r; cin >> l >> r;
```

```
79          cout << f(l, r, m) % m << endl;
80      }
81
82      // cout << endl << (clock() - pp) / CLOCKS_PER_SEC << endl;
83      return 0;
84  }
```

5.8.3 例题三

求

$$n^{(n-1)^{(n-2)^{\cdots^1}}}$$

```
1   #include <bits/stdc++.h>
2   #define aaa cout<<233<<endl;
3   #define endl '\n'
4   #define pb push_back
5   using namespace std;
6   typedef long long ll;
7   typedef unsigned long long ull;
8   typedef long double ld;
9   // mt19937 rnd(time(0));
10  const int inf = 0x3f3f3f3f;//1061109567 > 1e9
11  const ll linf = 0x3f3f3f3f3f3f3f3f;
12  const double eps = 1e-7;
13  const double pi = 3.14159265358979;
14  const int maxn = 1e5 + 5;
15  const int maxm = 1e3 + 5;
16  const int mod = 1e9 + 7;
17
18
19  inline ll modulo(ll x, ll mod){return x < mod ? x : x % mod + mod;}
20  inline ll pow_(ll a, ll b, ll p)
21  {
22      ll ret = 1;
23      while(b)
24      {
25          if(b & 1)ret = modulo(ret * a, p);
26          a = modulo(a * a, p);
27          b >>= 1;
28      }
29      return ret;
30  }
31  unordered_map<ll, ll> phi_;
32  inline ll phi(ll x)
33  {
34      if(phi_[x])return phi_[x];
35      ll ans = x;
36      ll t = x;
37      for(ll i = 2; i * i <= x; ++i)
38      {
39          if(x % i == 0)
40          {
41              ans = ans / i * (i - 1);
42              while(x % i == 0)x /= i;
43          }
44      }
45      if(x > 1)ans = ans / x * (x - 1);
46      phi_[t] = ans;
47      return ans;
48  }
49
50  // ll f(ll l, ll r, ll p)
51  // {
52  //      if(p == 1)return 1;
53  //      if(l == r + 1)return 1;
```

```
//     return pow_(a[l], f(l + 1, r, phi(p)), p);
// }

ll f(ll a, ll p)
{
    if(p == 1)return 1;
    if(a == 1)return 1;
    return pow_(a, f(a - 1, phi(p)), p);
}

int main()
{
    // double pp = clock();
    // freopen("233.in", "r", stdin);
    // freopen("233.out", "w", stdout);
    ios_base::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);

    ll n, m;
    while(cin >> n >> m)cout << f(n, m) % m << endl;

    // cout << endl << (clock() - pp) / CLOCKS_PER_SEC << endl;
    return 0;
}
```

5.9 逆序数

```
/*
*  也可以用树状数组做
*  a[0...n-1] cnt=0; call: MergeSort(0, n)
*/
const int N = 1010;
int a[N];
int c[N];
int cnt = 0;

void MergeSort(int l, int r)
{
    int mid, i, j, tmp;
    if (r > l + 1)
    {
        mid = (l + r) / 2;
        MergeSort(l, mid);
        MergeSort(mid, r);
        tmp = l;
        for (i = l, j = mid; i < mid && j < r;)
        {
            if (a[i] > a[j])
            {
                c[tmp++] = a[j++];
                cnt += mid - i;
            }
            else
            {
                c[tmp++] = a[i++];
            }
        }
        if (j < r)
        {
            for (; j < r; ++j)
            {
                c[tmp++] = a[j];
            }
        }
        else
```

```
        {
            for (; i < mid; ++i)
            {
                c[tmp++]=a[i];
            }
        }
        for (i = l; i < r; ++i)
        {
            a[i] = c[i];
        }
    }
    return ;
}
```

5.10 无序序列变有序的最少交换次数

5.10.1 相邻元素交换

```
\\ 等于逆序数个数
```

5.10.2 任意元素交换

```
*
*  交换任意两数的本质是改变了元素位置,
*  故建立元素与其目标状态应放置位置的映射关系
*/
int getMinSwaps(vector<int> &A)
{
    //  排序
    vector<int> B(A);
    sort(B.begin(), B.end());
    map<int, int> m;
    int len = (int)A.size();
    for (int i = 0; i < len; i++)
    {
        m[B[i]] = i;    //  建立每个元素与其应放位置的映射关系
    }

    int loops = 0;      //  循环节个数
    vector<bool> flag(len, false);
    //  找出循环节的个数
    for (int i = 0; i < len; i++)
    {
        if (!flag[i])
        {
            int j = i;
            while (!flag[j])
            {
                flag[j] = true;
                j = m[A[j]];    //  原序列中j位置的元素在有序序列中的位置
            }
            loops++;
        }
    }
    return len - loops;
}
```

5.11 GCD

5.11.1 非递归

```
int gcd(int x, int y)
{
    if (!x || !y)
    {
        return x > y ? x : y;
    }

    for (int t; t = x % y, t; x = y, y = t) ;

    return y;
}
```

5.11.2 递归

```
int gcd(int a, int b)
{
    return b==0 ? a : gcd(b, a%b);
}
```

5.12 EXGCD

欧几里得是用来求 a,b 的最大公约数，那么扩展欧几里得不仅能求出 a,b 的最大公约数，还能求出满足 ax+by=gcd(a,b) 的一组可行解。

```
/*
*  求x，y使得gcd(a, b) = a * x + b * y;
*/
int extgcd(int a, int b, int &x, int &y)
{
    if (b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }
    int d = extgcd(b, a % b, x, y);
    int t = x;
    x = y;
    y = t - a / b * y;
    return d;
}
```

5.12.1 例题

例题：给出 a,b,c,x1,x2,y1,y2，求满足 ax+by+c=0，且 $x \in [x1, x2], y \in [y1, y2]$ 的整数解个数

先处理无解情况：
1、当 a=0 并且 b=0，而 $c \neq 0$ 时，显然无解；当 a=0,b=0，而 c=0 时，[x1,x2],[y1,y2] 都为可行解，根据乘法原理，可行解的个数为 (x2-x1+1)*(y2-y1+1);

2、当 a=0 $b \neq 0$ 时：此时即为求解 by=c，则 y=c/b，如果 c/b 不是整数或 c/b 不在 [y1,y2] 的范围内，无解否则 [x1,x2] 内全部整数都为可行解。

3、当 b=0,$a \neq 0$ 时，同上。

4、若 c 不是 gcd(a,b) 的倍数，方程显然无解。

```
#include <iostream>
#include <cmath>
#include <cstdio>
typedef ll longlong
using namespace std;
```

```
6
7  ll a,b,c,x1,x2,yy1,y2,x0,yy0; // 全局变量
8  inline ll cmin(const ll &x,const ll &y) {return x<y?x:y;}
9  inline ll cmax(const ll &x,const ll &y) {return x>y?x:y;}
10
11 ll gcd(ll a,ll b)
12 {
13     if (b==0) return a;
14     return gcd(b,a % b);
15 }
16
17 void exgcd(ll a,ll b)
18 {
19     if (b==0){x0=1;yy0=0;return;}
20     exgcd(b,a%b);
21     ll t=x0;x0=yy0;yy0=t-a/b*yy0;
22     return;
23 }
24
25 int main()
26 {
27     cin >> a >> b >> c;
28     cin >> x1 >> x2;
29     cin >> yy1 >> y2;
30     c=-c;
31     if (c<0) {a=-a;b=-b;c=-c;}
32     if (a<0) {a=-a;ll t=x1;x1=-x2;x2=-t;}
33     if (b<0) {b=-b;ll t=yy1;yy1=-y2;y2=-t;}
34     if (a==0 && b==0)
35     {
36         if (c==0)
37         {
38             cout << (x2-x1+1)*(y2-yy1+1) << endl;
39             return 0;
40         }
41         cout << "0" << endl; return 0;
42     }
43     else if (a==0)
44     {
45         if (c % b ==0)
46             if (c/b<=y2 && c/b>=yy1) {cout << x2-x1+1 << endl; return 0;}
47         cout << "0" << endl; return 0;
48     }
49     else if (b==0)
50     {
51     if (c%a==0)
52         if (c/a<=x2 && c/a>=x1) {cout << y2-yy1+1 << endl; return 0;}
53     cout << "0" << endl; return 0;
54     }
55
56     ll d=gcd(a,b);
57     if (c%d!=0){cout << "0" << endl; return 0;}
58
59     a=a/d;b=b/d;c=c/d;
60     exgcd(a,b);
61     x0=x0*c;yy0=yy0*c;
62
63     double tx2=x2,tx1=x1,tx0=x0,ta=a,tb=b,tc=c,ty1=yy1,ty2=y2,ty0=yy0;
64     ll down1=floor(((tx2-tx0)/tb)),down2=floor(((ty0-ty1)/ta));
65     ll r=cmin(down1,down2);
66     ll up1=ceil(((tx1-tx0)/tb)),up2=ceil(((ty0-ty2)/ta));
67     ll l=cmax(up1,up2);
68     if (r<l) cout << "0" << endl;
69     else cout << r-l+1 << endl;
70     return 0;
71 }
```

5.13 逆元

对于正整数 a 和 m，如果有 $ax \equiv 1(mod \quad m)$, 那么把这个同余方程中 x 的最小正整数解叫做 a 模 m 的逆元。

5.13.1 拓展欧几里得法

```
// 扩展欧几里得(求a对于mod的逆元，要求a与mod互素)
ll exgcd(ll a, ll b, ll &x, ll &y)
{
    //求2对于1e9+7的逆元就是 exgcd(2, 1e9+7, x, y),其中x的值就是inv2,
    if (b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }
    ll r = exgcd(b, a % b, x, y);
    ll t = x % mod;
    x = y % mod;
    y = ((t - a / b * y) % mod + mod) % mod;
    return r;
}
```

5.13.2 费马小定理法

```
// 费马小定理(求a对于mod的逆元，要求mod为素数)
ll power_mod(ll a, ll b, ll mod)
{
    ll ans = 1;
    while (b)
    {
        if (b & 1) ans = ans * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ans;
}
inv2 = power_mod(a, mod - 2, mod);
```

5.14 中国剩余定理

中国剩余定理给出了以下的一元线性同余方程:

$$(S): \begin{cases} x \equiv a_1 & (mod \quad m_1) \\ x \equiv a_2 & (mod \quad m_2) \\ \vdots & \\ x \equiv a_n & (mod \quad m_n) \end{cases}$$

中国剩余定理说明：假设整数 $m_1, m_2, \cdots, m_n$ 两两互质，则对任意的整数：$a_1, a_2, \cdots, a_n$ 方程组 (S) 有解，并且通解可以通过如下方式构造得到：

```
//n个方程: x=a[i](mod m[i]) (0<=i<n)
ll china(int n, ll *a, ll *m){
    ll M = 1, ret = 0;
    for(int i = 0; i < n; i ++) M *= m[i];
    for(int i = 0; i < n; i ++){
        ll w = M / m[i];
        ret = (ret + w * inv(w, m[i]) * a[i]) % M;
    }
    return (ret + M) % M;
}
```

5.14.1 例题

人自出生起就有体力，情感和智力三个生理周期，分别为 23，28 和 33 天。一个周期内有一天为峰值，在这一天，人在对应的方面（体力，情感或智力）表现最好。通常这三个周期的峰值不会是同一天。现在给出三个日期，分别对应于体力，情感，智力出现峰值的日期。然后再给出一个起始日期，要求从这一天开始，算出最少再过多少天后三个峰值同时出现。

分析：假设 x 为这个天数，n1 为第一个周期出现的日期，p1 为其周期，以此类推，则如果同时出现峰值，则：

$x = n_1 + k_1 * p_1$
$x = n_2 + k_2 * p_2$
$x = n_3 + k_3 * p_3$

两侧同时取余 p 有：
$x\%p_1 = n_1\%p_1$ 即 $x\%p_1 = a_1$
$x\%p_2 = n_2\%p_2$ 即 $x\%p_2 = a_2$
$x\%p_3 = n_3\%p_3$ 即 $x\%p_3 = a_3$

$p_1 = 23, p_2 = 27, p_3 = 33$ 两两互质

```
#include<cstdio>
typedef long long ll;
const int N = 100000 + 5;
void ex_gcd(ll a, ll b, ll &x, ll &y, ll &d){
    if (!b) {d = a, x = 1, y = 0;}
    else{
        ex_gcd(b, a % b, y, x, d);
        y -= x * (a / b);
    }
}

ll inv(ll t, ll p){//如果不存在，返回-1
    ll d, x, y;
    ex_gcd(t, p, x, y, d);
    return d == 1 ? (x % p + p) % p : -1;
}

ll china(int n, ll *a, ll *m){//中国剩余定理
    ll M = 1, ret = 0;
    for(int i = 0; i < n; i ++) M *= m[i];
    for(int i = 0; i < n; i ++){
        ll w = M / m[i];
        ret = (ret + w * inv(w, m[i]) * a[i]) % M;
    }
    return (ret + M) % M;
}

int main(){
    ll p[3], r[3], d, ans, MOD = 21252;
    int cas = 0;
    p[0] = 23; p[1] = 28; p[2] = 33;
    while(~scanf("%lld%lld%lld%lld", &r[0], &r[1], &r[2], &d) && (~r[0] || ~r[1]
      || ~r[2] || ~d)){
        ans = ((china(3, r, p) - d) % MOD + MOD) % MOD;
        printf("Case %d: the next triple peak occurs in %I64d days.\n", ++cas, ans
           ? ans : 21252);
    }
    return 0;
}
```

5.15 拓展中国剩余定理

```
//m1,m2, ... ,mn两两不保证互质
```

```
#include<cstdio>
#include<algorithm>
using namespace std;
typedef long long ll;
typedef pair<ll, ll> PLL;
PLL linear(ll A[], ll B[], ll M[], int n) {//求解A[i]x = B[i] (mod M[i]),总共n个线性方程组
    ll x = 0, m = 1;
    for(int i = 0; i < n; i ++) {
        ll a = A[i] * m, b = B[i] - A[i]*x, d = gcd(M[i], a);
        if(b % d != 0)  return PLL(0, -1);//答案不存在，返回-1
        ll t = b/d * inv(a/d, M[i]/d)%(M[i]/d);
        x = x + m*t;
        m *= M[i]/d;
    }
    x = (x % m + m ) % m;
    return PLL(x, m);//返回的x就是答案，m是最后的lcm值
}
```

5.16 素数

5.16.1 判断小于 MAXN 的数是否为素数

```
/*
*  素数筛选，判断小于MAXN的数是不是素数
*  notprime是一张表，false表示是素数，true表示不是
*/
const int MAXN = 1000010;
bool notprime[MAXN];

void init()
{
    memset(notprime, false, sizeof(notprime));
    notprime[0] = notprime[1] = true;
    for (int i = 2; i < MAXN; i++)
    {
        if (!notprime[i])
        {
            if (i > MAXN / i)   //  阻止后边i * i溢出（或者i,j用long long)
            {
                continue;
            }
            //  直接从i * i开始就可以，小于i倍的已经筛选过了
            for (int j = i * i; j < MAXN; j += i)
            {
                notprime[j] = true;
            }
        }
    }
}
```

5.16.2 埃式筛法

```
const int MAX = 1e6 + 10;
int prime[MAX], k; // k为全局变量，指向最后一个素数的后一个单元
bool isPrime[MAX];
void Prime()
{
    k = 0;
    memset(isPrime, true, sizeof(isPrime));
    isPrime[1] = false; // 1不是素数
    for(int i = 2; i < MAX; i++)
    {
        if(isPrime[i])
```

```
12          {
13              prime[k++] = i;
14              for(int j = 2 ; i * j < MAX ; j++)
15              {
16                  isPrime[i * j] = false;
17              }
18          }
19      }
20  }
```

5.16.3 线性筛法

```
1  /*
2  *  素数筛选，查找出小于等于MAXN的素数
3  *  prime[0]存素数的个数
4  */
5
6  const int MAXN = 100000;
7  int prime[MAXN + 1];
8
9  void getPrime()
10 {
11     memset(prime, 0, sizeof(prime));
12     for (int i = 2; i <= MAXN; i++)
13     {
14         if (!prime[i])
15         {
16             prime[++prime[0]] = i;
17         }
18         for (int j = 1; j <= prime[0] && prime[j] <= MAXN / i; j++)
19         {
20             prime[prime[j] * i] = 1;
21             if (i % prime[j] == 0)
22             {
23                 break;
24             }
25         }
26     }
27 }
```

5.16.4 Miller Rabin

```
1  /*
2  *  随机素数测试（伪素数原理）
3  *  CALL: bool res = miller(n);
4  *  快速测试n是否满足素数的“必要”条件，出错概率极低
5  *  对于任意奇数n > 2和正整数s，算法出错概率 2^(-s)
6  */
7
8  int witness(int a, int n)
9  {
10     int x, d = 1;
11     int i = ceil(log(n - 1.0) / log(2.0)) - 1;
12     for (; i >= 0; i--)
13     {
14         x = d;
15         d = (d * d) % n;
16         if (d == 1 && x != 1 && x != n - 1)
17         {
18             return 1;
19         }
20         if (((n - 1) & (1 << i)) > 0)
21         {
22             d = (d * a) % n;
```

```
        }
    }
    return (d == 1 ? 0 : 1);
}

int miller(int n, int s = 50)
{
    if (n == 2)     //  质数返回1
        return 1;
    if (n % 2 == 0) //  偶数返回0
        return 0;
    int j, a;
    for (j = 0; j < a; j++)
    {
        a = rand() * (n - 2) / RAND_MAX + 1;
        //  rand()只能随机产生[0, RAND_MAX)内的整数
        //  而且这个RAND_MAX只有32768直接%n的话是永远
        //  也产生不了[RAND_MAX, n)之间的数
        if (witness(a, n))
        {
            return 0;
        }
    }
    return 1;
}
```

5.16.5 求 1e11 内的素数个数

```
// 方法一
// O(n^(3/4))
ll f[340000],g[340000],n;
void init()
{
    ll i,j,m;
    for(m=1;m*m<=n;++m)f[m]=n/m-1;
    for(i=1;i<=m;++i)g[i]=i-1;
    for(i=2;i<=m;++i)
    {
        if(g[i]==g[i-1])continue;
        for(j=1;j<=min(m-1,n/i/i);++j)
        {
            if(i*j<m)f[j]-=f[i*j]-g[i-1];
            else f[j]-=g[n/i/j]-g[i-1];
        }
        for(j=m;j>=i*i;--j)g[j]-=g[j/i]-g[i-1];
    }
}
int main()
{
    while(scanf("%I64d",&n)!=EOF)
    {
        init();
        cout<<f[1]<<endl;
    }
    return 0;
}
```

```
// 方法二
// O(n^(2/3))
const int N = 5e6 + 2;
bool np[N];
int prime[N], pi[N];
int getprime()
{
    int cnt = 0;
```

```
9     np[0] = np[1] = true;
10    pi[0] = pi[1] = 0;
11    for(int i = 2; i < N; ++i)
12    {
13        if(!np[i]) prime[++cnt] = i;
14        pi[i] = cnt;
15        for(int j = 1; j <= cnt && i * prime[j] < N; ++j)
16        {
17            np[i * prime[j]] = true;
18            if(i % prime[j] == 0)   break;
19        }
20    }
21    return cnt;
22 }
23 const int M = 7;
24 const int PM = 2 * 3 * 5 * 7 * 11 * 13 * 17;
25 int phi[PM + 1][M + 1], sz[M + 1];
26 void init()
27 {
28    getprime();
29    sz[0] = 1;
30    for(int i = 0; i <= PM; ++i)  phi[i][0] = i;
31    for(int i = 1; i <= M; ++i)
32    {
33        sz[i] = prime[i] * sz[i - 1];
34        for(int j = 1; j <= PM; ++j)
35            phi[j][i] = phi[j][i - 1] - phi[j / prime[i]][i - 1];
36    }
37 }
38 int sqrt2(LL x)
39 {
40    LL r = (LL)sqrt(x - 0.1);
41    while(r * r <= x)   ++r;
42    return int(r - 1);
43 }
44 int sqrt3(LL x)
45 {
46    LL r = (LL)cbrt(x - 0.1);
47    while(r * r * r <= x)   ++r;
48    return int(r - 1);
49 }
50 LL getphi(LL x, int s)
51 {
52    if(s == 0)  return x;
53    if(s <= M)  return phi[x % sz[s]][s] + (x / sz[s]) * phi[sz[s]][s];
54    if(x <= prime[s]*prime[s])   return pi[x] - s + 1;
55    if(x <= prime[s]*prime[s]*prime[s] && x < N)
56    {
57        int s2x = pi[sqrt2(x)];
58        LL ans = pi[x] - (s2x + s - 2) * (s2x - s + 1) / 2;
59        for(int i = s + 1; i <= s2x; ++i) ans += pi[x / prime[i]];
60        return ans;
61    }
62    return getphi(x, s - 1) - getphi(x / prime[s], s - 1);
63 }
64 LL getpi(LL x)
65 {
66    if(x < N)   return pi[x];
67    LL ans = getphi(x, pi[sqrt3(x)]) + pi[sqrt3(x)] - 1;
68    for(int i = pi[sqrt3(x)] + 1, ed = pi[sqrt2(x)]; i <= ed; ++i)
69        ans -= getpi(x / prime[i]) - i + 1;
70    return ans;
71 }
72 LL lehmer_pi(LL x)
73 {
74    if(x < N)   return pi[x];
```

```
    int a = (int)lehmer_pi(sqrt2(sqrt2(x)));
    int b = (int)lehmer_pi(sqrt2(x));
    int c = (int)lehmer_pi(sqrt3(x));
    LL sum = getphi(x, a) +(LL)(b + a - 2) * (b - a + 1) / 2;
    for (int i = a + 1; i <= b; i++)
    {
        LL w = x / prime[i];
        sum -= lehmer_pi(w);
        if (i > c) continue;
        LL lim = lehmer_pi(sqrt2(w));
        for (int j = i; j <= lim; j++) sum -= lehmer_pi(w / prime[j]) - (j - 1);
    }
    return sum;
}
int main()
{
    init();
    LL n;
    while(~scanf("%lld",&n))
    {
        printf("%lld\n",lehmer_pi(n));
    }
    return 0;
}
```

5.17 组合数

5.17.1 求单次

```
int com(int n, int r)   //  return C(n, r)
{
    if (n - r > r)
    {
        r = n - r;      //  C(n, r) = C(n, n - r)
    }
    int i, j, s = 1;
    for (i = 0, j = 1; i < r; ++i)
    {
        s *= (n - i);
        for (; j <= r && s % j == 0; ++j)
        {
            s /= j;
        }
    }
    return s;
}
```

5.17.2 线性打表

```
typedef long long ll;
const ll mod = 1e9 + 7;
const ll M = 2e5 + 10;
//fact[i]是i的阶乘，ifact[i]是阶乘的除法逆元，两者用于求组合数
ll fact[M],ifact[M];
//快速幂求n^k余m的结果
ll pow_mod(ll n,ll k,ll mod) {
    ll res = 1;
    n = n % mod;
    while (k > 0) {
        if (k & 1)
            res = res * n % mod;
        n = n * n % mod;
        k >>= 1;
    }
```

```
16      return res;
17  }
18  void init() {
19      fact[0] = ifact[0] = 1;
20      for(int i = 1; i < M; ++i) {
21          fact[i] = (fact[i-1] * i) % mod;
22      }
23      ifact[M-1] = pow_mod(fact[M-1], mod - 2, mod);
24      for(int i = M - 1; i > 0; i--)
25          ifact[i-1] = ifact[i] * i % mod;
26  }
27  ll C(ll n,ll m) {
28      if(n < m)
29          return 0;
30      return  fact[n] * ifact[m] % mod * ifact[n - m] % mod;
31  }
```

5.17.3 少量 C(n,m) 打表

```
1  ll c[maxn][maxn];
2  void init() {
3      c[0][0] = 1;
4      for (int i = 1; i < maxn; i++) {
5          for (int j = 0; j <= i; j++) {
6              if (!j) c[i][j] = 1;
7              else c[i][j] = c[i - 1][j - 1] + c[i - 1][j];
8          }
9      }
10 }
```

5.18 阶乘长度

```
1  #define PI 3.1415926
2
3  int main()
4  {
5      int n, a;
6      while (~scanf( “%d”, &n))
7      {
8          a = (int)((0.5 * log(2 * PI * n) + n * log(n) - n) / log(10));
9          printf("%d\n", a + 1);
10     }
11     return 0;
12 }
```

5.19 全排列

```
1  #define MAX_N 10
2  int n;                          //  共n个数
3  int rcd[MAX_N];                 //  记录每个位置填的数
4  int used[MAX_N];                //  标记数是否用过
5  int num[MAX_N];                 //  存放输入的n个数
6
7  void full_permutation(int l)
8  {
9      int i;
10     if (l == n)
11     {
12         for (i = 0; i < n; i++)
13         {
14             printf("%d", rcd[i]);
15             if (i < n-1)
```

```
16              {
17                  printf(" ");
18              }
19          }
20          printf("\n");
21          return ;
22      }
23      for (i = 0; i < n; i++)          //  枚举所有的数(n个),循环从开始
24      if (!used[i])
25      {                                //  若num[i]没有使用过, 则标记为已使用
26          used[i] = 1;
27          rcd[l] = num[i];             //  在l位置放上该数
28          full_permutation(l+1);       //  填下一个位置
29          used[i] = 0;                 //  清标记
30      }
31  }
```

5.20 求斐波那契第 N 项

```
1   /*
2   *  求斐波那契数列第N项, 模MOD
3   */
4   #define mod(a, m) ((a) % (m) + (m)) % (m)
5   const int MOD = 1e9 + 9;
6   struct MATRIX
7   {
8       long long a[2][2];
9   };
10
11  MATRIX a;
12  long long f[2];
13
14  void ANS_Cf(MATRIX a)
15  {
16      f[0] = mod(a.a[0][0] + a.a[1][0], MOD);
17      f[1] = mod(a.a[0][1] + a.a[1][1], MOD);
18      return ;
19  }
20
21  MATRIX MATRIX_Cf(MATRIX a, MATRIX b)
22  {
23      MATRIX ans;
24      int k;
25      for (int i = 0; i < 2; i++)
26      {
27          for (int j = 0; j < 2; j++)
28          {
29              ans.a[i][j] = 0;
30              k = 0;
31              while (k < 2)
32              {
33                  ans.a[i][j] += a.a[k][i] * b.a[j][k];
34                  ans.a[i][j] = mod(ans.a[i][j], MOD);
35                  ++k;
36              }
37          }
38      }
39      return ans;
40  }
41
42  MATRIX MATRIX_Pow(MATRIX a, long long n)
43  {
44      MATRIX ans;
45      ans.a[0][0] = 1;
46      ans.a[1][1] = 1;
```

```
47      ans.a[0][1] = 0;
48      ans.a[1][0] = 0;
49      while (n)
50      {
51          if (n & 1)
52          {
53              ans = MATRIX_Cf(ans, a);
54          }
55          n = n >> 1;
56          a = MATRIX_Cf(a, a);
57      }
58      return ans;
59  }
60
61  int main()
62  {
63      long long n;
64      while (cin >> n)
65      {
66          if (n == 1)
67          {
68              cout << '1' << '\n';
69              continue;
70          }
71          a.a[0][0] = a.a[0][1] = a.a[1][0] = 1;
72          a.a[1][1] = 0;
73          a = MATRIX_Pow(a, n - 2);
74          ANS_Cf(a);
75          cout << f[0] << '\n';
76      }
77      return 0;
78  }
```

5.21 质因子

```
1  vector<int> primeFactorList;
2
3  void getPrimeFactor(int num)
4  {
5      for(int i=2;i*i<=num;i++)
6      {
7          if(num%i==0)
8          {
9              primeFactorList.push_back(i);
10             while(num%i==0) num=num/i;
11         }
12     }
13     if(num>1) primeFactorList.push_back(num);
14 }
```

5.22 大数质因子分解

5.22.1 pollard_rho

```
1  inline LL add(LL x, LL y, LL mod) {
2      return (x += y) < mod ? x : x - mod;
3  }
4
5  inline LL mul(LL x, LL y, LL mod) {
6      const int BLEN = __builtin_clzll(mod) - 1;
7      const LL BMSK = (1LL << BLEN) - 1;
8      LL ret = 0;
9      if(x < y)
```

```
10          swap(x, y);
11      while(y > 0) {
12          ret += x * (y & BMSK);
13          ret = ret < mod ? ret : ret % mod;
14          y >>= BLEN;
15          x <<= BLEN;
16          x = x < mod ? x : x % mod;
17      }
18      return ret;
19  }
20
21  inline LL Pow(LL x, LL k, LL mod) {
22      LL ret = mod > 1 ? 1 : 0;
23      for( ; k > 0; k >>= 1, x = mul(x, x, mod))
24          if(k & 1)
25              ret = mul(ret, x, mod);
26      return ret;
27  }
28
29  inline bool miller_rabin(LL n) {
30      if(n == 2)
31          return 1;
32      if(n < 2 || !(n & 1))
33          return 0;
34      LL s, r;
35      for(s = 0, r = n - 1; !(r & 1); r >>= 1, ++s);
36      static const int ptot = 12, pr[ptot] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
          31, 37};
37      for(int p : pr) {
38          if(p >= n)
39              break;
40          LL cur = 1, nxt = p;
41          for(LL k = r; k > 0; k >>= 1, nxt = mul(nxt, nxt, n))
42              if(k & 1)
43                  cur = mul(cur, nxt, n);
44          for(int i = 0; i < s; ++i) {
45              nxt = mul(cur, cur, n);
46              if(nxt == 1 && cur != 1 && cur != n - 1)
47                  return 0;
48              cur = nxt;
49          }
50          if(cur != 1)
51          return 0;
52      }
53      return 1;
54  }
55
56  inline LL pollard_rho(LL n) {
57      static mt19937_64 rnd(996);
58      while(1) {
59          LL x = rnd() % (n - 3) + 2, y = x, c = rnd() % n, tim = 0, prd = 1;
60          for(LL i = 1, stp = 2; ; ++i) {
61              if(i == stp) {
62                  y = x;
63                  stp <<= 1;
64              }
65              if((x = add(mul(x, x, n), c, n)) == y)
66                  break;
67              LL tmp = prd;
68              if((prd = mul(prd, abs(y - x), n)) == 0)
69                  return __gcd(tmp, n);
70              static const int maxt = 100;
71              if((++tim) < maxt)
72                  continue;
73              if((prd = __gcd(prd, n)) > 1)
74                  return prd;
```

```
            tim = 0;
        }
        if(tim > 0 && (prd = __gcd(prd, n)) > 1)
            return prd;
    }
    assert(0);
}

inline void factorize(LL n, vector<pair<LL, int> > &ret) {
    vector<LL> vec;
    queue<LL> que;
    que.push(n);
    while(!que.empty()) {
        LL x = que.front();
        que.pop();
        for(LL y : vec)
            for( ; x % y == 0; x /= y, vec.push_back(y));
        if(x == 1)
            continue;
        if(miller_rabin(x)) {
            vec.push_back(x);
        } else {
            LL y = pollard_rho(x);
            que.push(y);
            que.push(x / y);
        }
    }
    sort(vec.begin(), vec.end());
    ret.clear();
    for(auto x : vec)
        if(!ret.empty() && ret.back().first == x) {
            ++ret.back().second;
        } else {
            ret.push_back(make_pair(x, 1LL));
        }
}

// 用法
vector<pair<LL, int> >d;
factorize(x,d);
```

5.23 公共因子数

```
//一串数字的公共因子数
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    ll n,gc;
    cin>>n;
    cin>>gc;
    n--;
    while(n--)
    {
        ll tmp;
        cin>>tmp;
        gc = __gcd(gc,tmp);
        if(gc==1)
        {
            cout<<"1"<<endl;
            return 0;
```

```
        }
    }
    ll ans =0;
    for(ll i=1;i*i<=gc;i++)
    {
        if(gc%i==0)
        {
            ans++;
            if(i*i!=gc) ans++;
        }
    }
    cout<<ans<<endl;
}
```

5.24 除法分块

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
#define inf 0x3f3f3f3f
#define INF 0x3f3f3f3f3f3f3f3f
#define IO ios::sync_with_stdio(0)
typedef pair<int, int> P;
const int maxn = 2e5+5;
const ll mod = 1e9+7;
using namespace std;


ull cal(ull n)
{
    ull ans = 0;
    for(ull l=1, r; l<=n; l=r+1)
    {
        r = n/(n/l);
        ans += (n/l) * (l+r)*(r-l+1)/2;
    }
    return ans;
}
int main()
{
    IO;
    ull a, b;
    cin>>a>>b;
    cout<<cal(b) - cal(a-1)<<"\n";
    return 0;
}
```

5.25 高斯消元

5.25.1 求逆矩阵

```
\\ 普通矩阵
#include<bits/stdc++.h>
using namespace std;
const int N = 505;
const int mod = 1e9+7;
int a[N][N << 1];
int n;
int ppow(int a,int b,int mod) {
    int ans = 1 % mod; a %= mod;
    while(b) {
        if (b & 1) ans = 1ll * ans * a % mod;
```

```
        a = 1ll * a * a % mod;
        b >>= 1;
    }
    return ans;
}
int Gauss_rev(int n) {
    for (int i = 1; i <= n; i++) {
        for (int j = i; j <= n; j++) {//找第i列非零的行换上来
            if (a[j][i]) {
                swap(a[i], a[j]);
                break;
            }
        }
        if (!a[i][i]) return 0;//无解
        int kk = ppow(a[i][i], mod - 2, mod);//逆元
        for (int j = i; j <= n * 2; j++) {//当前行每一列都除以a[i][i]
            a[i][j] = 1ll * a[i][j] * kk % mod;
        }
        for(int j = 1; j <= n; j++) {//其他行
            if (j != i) {
                kk = a[j][i];
                for (int k = i; k <= n * 2; k++){
                    a[j][k] = (a[j][k] - 1ll * kk * a[i][k] % mod + mod) % mod;
                }
            }
        }
    }
    return 1;
}
int main() {
    scanf("%d",&n);
    for (int i = 1;i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            scanf("%d", &a[i][j]);
        }
        a[i][i + n] = 1;
    }
    if (!Gauss_rev(n)) {
        puts("No Solution");
    } else {
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                printf("%d ", a[i][j + n]);
            }
        puts("");
        }
    }
    return 0;
}

// 01矩阵，开关问题，结果异或
#include<bits/stdc++.h>
using namespace std;
const int N=505;
bitset<N << 1> a[N];
int n;
int Gauss_rev(int n) {
    for(int i = 1; i <= n; i++) {
        for (int j = i; j <= n; j++) {//找第i列非零的行换上来
            if (a[j][i]) {
                swap(a[i], a[j]);
                break;
            }
        }
        if (!a[i][i]) return 0;//无解
        for (int j = 1; j <= n; j++) {//其他行
```

```
78              if (a[j][i] && j != i) {
79                  a[j] ^= a[i];
80              }
81          }
82      }
83      return 1;
84  }
85  int main() {
86      scanf("%d", &n);
87      for (int i = 1;i <= n; i++) {
88          for (int j = 1; j <= n; j++) {
89              scanf("%d", &a[i][j]);
90          }
91          a[i][i + n] = 1;
92      }
93      if (!Gauss_rev(n)) {
94          puts("No Solution");
95      } else {
96          for (int i = 1; i <= n; i++) {
97              for (int j = 1; j <= n; j++) {
98                  printf("%d ", a[i][j + n]);
99              }
100             puts("");
101         }
102     }
103     return 0;
104 }
```

5.25.2 求异或矩阵的解

```
1  int n;
2  int a[maxn][maxn];
3  vector<int> g[maxn];
4  int deg[maxn];
5  int free_cnt;
6  int free_x[maxn];
7  // 无解 -1
8  // 唯一解 0
9  // 多解，自由变元数量，存于free_x[]中 1
10 int gauss() {
11     int r, c;
12     for (r = 0, c = 0; c < n; c++) {
13         int t = r;
14         for (int i = r; i < n; i++) {
15             if (a[i][c]) {
16                 t = i;
17                 break;
18             }
19         }
20         if (!a[t][c]) {
21             free_x[free_cnt++] = c;
22             continue;
23         }
24         for (int i = c; i < n + 1; i++) swap(a[t][i], a[r][i]);
25         for (int i = r + 1; i < n; i++) {
26             if (a[i][c]) {
27                 for (int j = c; j < n + 1; j++) a[i][j] ^= a[r][j];
28             }
29         }
30         r++;
31     }
32     if (r < n) {
33         for (int i = r; i < n; i++) {
34             if (a[i][n]) {
35                 return -1;
```

```
            }
        }
        return 1;
    }
    for (int i = n - 1; i >= 0; i--) {
        for (int j = i + 1; j < n; j++) {
            a[i][n] ^= (a[i][j] & a[j][n]);
        }
    }
    // 唯一解是a[i][n]
    return 0;
}
```

5.25.3 求方程组解

```
const int maxn=110;
int n;
double f[maxn][maxn],ans[maxn];
void gauss() {
    for(int i = 1; i <= n; i++) {
        int l = i;
        for(int j = l + 1; j <= n; j++)
            if (fabs(f[l][i]) < fabs(f[j][i]))
                l = j;
        if(l != i)
        for(int j = i; j<= n + 1; j++)
            swap(f[l][j], f[i][j]);
        for(int j = i + 1; j <= n; j++) {
            double tmp = f[j][i] / f[i][i];
            for(int k = i; k <= n + 1; k++)
                f[j][k] = f[j][k] - f[i][k] * tmp;
        }
    }
    for(int i = n; i>= 1; i--) {
        double tmp = f[i][n + 1];
        for(int j = n; j > i; j--)
            tmp -= ans[j] * f[i][j];
        ans[i] = tmp / f[i][i];
    }
}

int main() {
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n+1;j++)
            scanf("%lf",&f[i][j]);
    gauss();
    for(int i=1;i<=n;i++)
        printf("%d ",int(ans[i]+0.5));

    return 0;
}
```

5.26 FFT

求 $x_1 * x_3 = 2 * x_2$ 的数量

```
typedef long long ll;
const double pi = acos(-1.0);
const int maxn = 1e6 + 10;

int n, m, k;
struct complexx {
    double x, y;
```

```
8       complexx(double x = 0.0, double y = 0.0) : x(x), y(y) {}
9       complexx operator + (complexx &a) { return complexx(x + a.x, y + a.y); }
10      complexx operator - (complexx &a) { return complexx(x - a.x, y - a.y); }
11      complexx operator * (complexx &a) { return complexx(x * a.x - y * a.y, y * a.x
            + x * a.y); }
12  };
13  complexx a[maxn], c[maxn];
14  int rev[maxn];
15  int b[maxn], sum[maxn];
16  void fft(complexx y[], int len, int on) {
17      for (int i = 0; i < len; i++) {
18          if (rev[i] > i) swap(y[i], y[rev[i]]);
19      }
20      for (int h = 2; h <= len; h <<= 1) {
21          complexx wn(cos(-on * 2 * pi / h), sin(-on * 2 * pi / h));
22          for (int j = 0; j < len; j += h) {
23              complexx w(1, 0);
24              for (int k = j; k < j + h / 2; k++) {
25                  complexx u = y[k];
26                  complexx t = w * y[k + h / 2];
27                  y[k] = u + t;
28                  y[k + h / 2] = u - t;
29                  w = w * wn;
30              }
31          }
32      }
33      if (on == -1) {
34          for (int i = 0; i < len; i++) {
35              y[i].x /= len;
36          }
37      }
38  }
39  int main() {
40      // freopen("in.txt", "r", stdin);
41      int m1 = 0, m3 = 0;
42      scanf("%d", &n);
43      for (int i = 1; i <= n; i++) {
44          int x;
45          scanf("%d", &x);
46          x += 3e4;
47          a[x].x += 1;
48          m1 = max(m1, x);
49      }
50      scanf("%d", &k);
51      for (int i = 1; i <= k; i++) {
52          scanf("%d", &b[i]);
53          b[i] += 3e4;
54      }
55      scanf("%d", &m);
56      for (int i = 1; i <= m; i++) {
57          int x;
58          scanf("%d", &x);
59          x += 3e4;
60          c[x].x += 1;
61          m3 = max(m3, x);
62      }
63      int len = 1, l = 0;
64      for (; len <= m1 + m3; len <<= 1, l++);
65      for (int i = 0; i < len; i++) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (l -
            1));
66      fft(a, len, 1);
67      fft(c, len, 1);
68      for (int i = 0; i < len; i++) {
69          a[i] = a[i] * c[i];
70      }
71      fft(a, len, -1);
```

```
72      for (int i = 0; i <= len; i++) {
73          sum[i] = (int)(a[i].x + 0.5);
74      }
75      ll ans = 0;
76      for (int i = 1; i <= k; i++) ans += sum[2 * b[i]];
77      printf("%lld\n", ans);
78      return 0;
79  }
```

5.27 NTT

```
1   // P = 998244353, G为P的原根
2   const int G = 3, P = (119 << 23) + 1;
3   int n, m, L, R[maxn];
4   int A[maxn], B[maxn];
5   int qpow(int a, int b) {
6       int ans = 1;
7       for (; b; b >>= 1,a = 1ll * a * a % P)
8           if (b & 1) ans = 1ll * ans * a % P;
9       return ans;
10  }
11  void NTT(int* a, int f) {
12      for (int i = 0; i < n; i++) if (i < R[i]) swap(a[i], a[R[i]]);
13      for (int i = 1; i < n; i <<= 1) {
14          int gn = qpow(G, (P - 1) / (i << 1));
15          for (int j = 0; j < n; j += (i << 1)) {
16              int g = 1;
17              for (int k = 0; k < i; k++, g = 1ll * g * gn % P) {
18                  int x = a[j + k],y = 1ll * g * a[j + k + i] % P;
19                  a[j + k] = (x + y) % P; a[j + k + i] = (x - y + P) % P;
20              }
21          }
22      }
23      if (f == 1) return;
24      int nv = qpow(n,P - 2); reverse(a + 1,a + n);
25      for (int i = 0; i < n; i++) a[i] = 1ll * a[i] * nv % P;
26  }
27  int main() {
28      n = read(); m = read();
29      for (int i = 0; i <= n; i++) A[i] = read();
30      for (int i = 0; i <= m; i++) B[i] = read();
31      m = n + m; for (n = 1; n <= m; n <<= 1) L++;
32      for (int i = 0; i < n; i++) R[i] = (R[i >> 1] >> 1) | ((i & 1) << (L - 1));
33      NTT(A,1); NTT(B,1);
34      for (int i = 0; i < n; i++) A[i] = 1ll * A[i] * B[i] % P;
35      NTT(A,-1);
36      for (int i = 0; i <= m; i++) printf("%d ",A[i]);
37      return 0;
38  }
```

5.28 原根

```
1   ll m;
2   ll fac[maxn], cnt;
3
4   ll quickPower(ll a, ll b, ll M) {
5       ll ans = 1ll;
6       ll base = a;
7       while (b) {
8           if (b & 1) {
9               ans *= base;
10              ans %= M;
11          }
```

```
        base *= base;
        base %= M;
        b >>= 1;
    }
    return ans;
}

void get_fac(ll x) {
    x--;
    ll m = sqrt(x) + 0.5;
    for (int i = 2; i <= m; i++) {
        if (x % i == 0) fac[cnt++] = i;
        while (x % i == 0) x /= i;
    }
    if (x > 1) fac[cnt++] = x;
}

int main() {
    scanf("%lld", &m);
    get_fac(m);
    for (ll g = 2; g < m; g++) {
        int f = 1;
        for (int j = 0; j < cnt; j++) {
            if (quickPower(g, (m - 1) / fac[j], m) == 1ll) {
                f = 0;
                break;
            }
        }
        if (f) {
            printf("%d", g);
            break;
        }
    }
    return 0;
}
```

5.29 1-n 的 x 次方和

5.29.1 一次方

$n * (n + 1) / 2$

5.29.2 二次方

$n * (n + 1) * (2 * n + 1) / 6$

5.29.3 三次方

$(n * (n + 1))^2 / 4$

6 STL

6.1 vector

```
vector<int> s;
//  定义一个空的vector对象，存储的是int类型的元素
vector<int> s(n);
//  定义一个含有n个int元素的vector对象
vector<int> s(first, last);
//  定义一个vector对象，并从由迭代器first和last定义的序列[first, last)中复制初值

s[i]                //  直接以下标方式访问容器中的元素
s.front()           //  返回首元素
s.back()            //  返回尾元素
```

```
s.push_back(x)       //  向表尾插入元素x
s.size()             //  返回表长
s.empty()            //  表为空时，返回真，否则返回假
s.pop_back()         //  删除表尾元素
s.begin()            //  返回指向首元素的随机存取迭代器
s.end()              //  返回指向尾元素的下一个位置的随机存取迭代器
s.insert(it, val)    //  向迭代器it指向的元素前插入新元素val
s.insert(it, n, val)//  向迭代器it指向的元素前插入n个新元素val
s.insert(it, first, last)
//  将由迭代器first和last所指定的序列[first, last)插入到迭代器it指向的元素前面
s.erase(it)          //  删除由迭代器it所指向的元素
s.erase(first, last)//  删除由迭代器first和last所指定的序列[first, last)
s.reserve(n)         //  预分配缓冲空间，使存储空间至少可容纳n个元素
s.resize(n)          //  改变序列长度，超出的元素将会全部被删除，如果序列需要扩展（原空间小于n），元素默认值将填满扩展出的空间
s.resize(n, val)     //  改变序列长度，超出的元素将会全部被删除，如果序列需要扩展（原空间小于n），val将填满扩展出的空间
s.clear()            //  删除容器中的所有元素
s.swap(v)            //  将s与另一个vector对象进行交换
s.assign(first, last)   //  将序列替换成由迭代器first和last所指定的序列[first, last)，[first, last)不能是原序列中的一部分

//  要注意的是，resize操作和clear操作都是对表的有效元素进行的操作，但并不一定会改变缓冲空间的大小
//  vector上还定义了序列之间的比较操作运算符（>、<、>=、<=、==、!=），可以按照字典序比较两个序列。
```

6.2 set

```
// 有序
set<int> s;
s.begin()        //  返回指向第一个元素的迭代器
s.clear()        //  清除所有元素
s.count()        //  返回某个值元素的个数
s.empty()        //  如果集合为空，返回true(真)
s.end()          //  返回指向最后一个元素之后的迭代器，不是最后一个元素
s.equal_range() //  返回集合中与给定值相等的上下限的两个迭代器
s.erase()        //  删除集合中的元素
s.find()         //  返回一个指向被查找到元素的迭代器
s.get_allocator()   //  返回集合的分配器
s.insert()       //  在集合中插入元素
s.lower_bound() //  返回指向大于（或等于）某值的第一个元素的迭代器
s.key_comp()     //  返回一个用于元素间值比较的函数
s.max_size()     //  返回集合能容纳的元素的最大限值
s.rbegin()       //  返回指向集合中最后一个元素的反向迭代器
s.rend()         //  返回指向集合中第一个元素的反向迭代器
s.size()         //  集合中元素的数目
s.swap()         //  交换两个集合变量
s.upper_bound() //  返回大于某个值元素的迭代器
s.value_comp()  //  返回一个用于比较元素间的值的函数

// 多重集合
multiset<int> s;
// 操作类似
```

6.3 pair

```
pair<T1, T2> p1;
pair<T1, T2> p1(v1, v2);
p1.first;
p1.second;
p1 = make_pair(v1, v2);

```

```
vector<pair<int, int> > v;
sort(v.begin(), v.end()); //根据pair的first排序，从小到大
```

6.4 stack

```
stack<int> s;
s.push(x);  //  入栈
s.pop();    //  出栈
s.top();    //  访问栈顶
s.empty();  //  当栈空时，返回true
s.size();   //  访问栈中元素个数
```

6.5 queue

```
queue<int> q;
q.push(x);  //  入队列
q.pop();    //  出队列
q.front();  //  访问队首元素
q.back();   //  访问队尾元素
q.empty();  //  判断队列是否为空
q.size();   //  访问队列中的元素个数

//priority_queue（优先队列）。优先队列与队列的差别在于优先队列不是按照入队的顺序出队，而是按照队列中元素的优先权出队列（默认为大者优先，也可以通过指定算子来指定自己的优先顺序）。

priority_queue模版类有三个模版参数，第一个是元素类型，第二个是容器类型，第三个是比较算子。其中后两者都可以忽略，默认容器为vector，默认算子为less，即小的往前排，大的往后排（出队列时列尾元素先出队）。

priority_queue<int> q;
priority_queue<pair<int, int> > qq; //  注意在两个尖括号之间一定要留空格，防止误判
priority_queue<int, vector<int>, greater<int> > qqq;//  定义小的先出队列

q.empty()      //  如果队列为空，则返回true，否则返回false
q.size()       //  返回队列中元素的个数
q.pop()        //  删除队首元素，但不返回其值
q.top()        //  返回具有最高优先级的元素值，但不删除该元素
q.push(item)   //  在基于优先级的适当位置插入新元素

//deque双端队列
#include<deque>
deque<int> dep;

deq.push_front(const T& x); //头插
deq.push_back(const T& x);  //尾插
deq.insert(iterator it, const T& x);     //任意位置

deq.pop_front();     //删除头部
deq.pop_back();      //删除尾部
deq.erase(iterator it);      //删除任意

deq[1]; // 并不会检查是否越界
deq.at(1); // 以上两者的区别就是 at 会检查是否越界，是则抛出 out of range 异常
deq.front();     //访问头部
deq.back();      //访问尾部
```

6.6 map

```
map<T1, T2> mp;
mp[key] = value;
T2 value = mp[key];
```

```
mp.size();        //  返回元素个数
mp.empty();       //  判断是否为空
mp.clear();       //  清空所有元素
```

6.7 bitset

```
const int MAXN = 32;

bitset<MAXN> bt;             //  bt 包括 MAXN 位，下标 0 ~ MAXN - 1，默认初始化为 0
bitset<MAXN> bt1(0xf);       //  0xf 表示十六进制数 f，对应二进制 1111，将 bt1 低 4 位初始化为 1
bitset<MAXN> bt2(012);       //  012 表示八进制数 12，对应二进制 1010，即将 bt2 低 4 位初始化为 1010
bitset<MAXN> bt3("1010");    //  将 bt3 低 4 位初始化为 1010

bt.any()         //  bt 中是否存在置为 1 的二进制位？
bt.none()        //  bt 中不存在置为 1 的二进制位吗？
bt.count()       //  bt 中置为 1 的二进制位的个数
bt.size()        //  bt 中二进制位的个数
bt[pos]          //  访问 bt 中在 pos 处的二进制位
bt.test(pos)     //  bt 中在 pos 处的二进制位是否为 1
bt.set()         //  把 bt 中所有二进制位都置为 1
bt.set(pos)      //  把 bt 中在 pos 处的二进制位置为 1
bt.reset()       //  把 bt 中所有二进制位都置为 0
bt.reset(pos)    //  把 bt 中在pos处的二进制位置为0
bt.flip()        //  把 bt 中所有二进制位逐位取反
bt.flip(pos)     //  把 bt 中在 pos 处的二进制位取反
bt[pos].flip()   //  同上
bt.to_ulong()    //  用 bt 中同样的二进制位返回一个 unsigned long 值
```

6.8 algorithm

```
reverse(begin, end) // 反转

unique(begin, end)  // 需排序，去除重复的相邻元素，常用于求不同元素个数
int n=unique(a, a+10)-a;

lower_bound(begin, end, value)  //返回指向第一个不小于给定值的元素的迭代器

upper_bound(begin, end, value)  //返回指向第一个大于给定值的元素的迭代器

next_permutation(array) //一种排列的下一种排列
```

7 计算几何

7.1 点（向量）模板

```
const double eps = 1e-8;
const double inf = 1e20;
const double pi = acos(-1.0);
const int maxp = 1010;
// 精度三态函数
int sgn(double x) {
    if(fabs(x) < eps) return 0;
    if(x < 0) return -1;
    else return 1;
}
typedef  struct Point vec; // 向量
struct Point {
    double x, y;
    double poe;//与原点斜率
```

```
15      Point() {}
16      Point(double _x,double _y) {
17          x = _x;
18          y = _y;
19      }
20      void input() {
21          scanf("%lf%lf", &x, &y);
22      }
23      void output() {
24          printf("%.2f %.2f\n", x, y);
25      }
26      vec chuizhi() { //返回垂直向量
27          return vec(-y, x);
28      }
29      bool operator == (Point b)const {
30          return sgn(x-b.x) == 0 && sgn(y-b.y) == 0;
31      }
32      bool operator < (Point b)const {
33          return sgn(x-b.x)== 0?sgn(y-b.y)<0:x<b.x;
34      }
35      Point operator -(const Point &b)const {
36          return Point(x-b.x,y-b.y);
37      }
38      // 向量叉积  a^b>0,a在b的顺时针, <0,逆时针, =0, 共线（平行）, 同向或反向
39      double operator ^(const Point &b)const {
40          return x*b.y - y*b.x;
41      }
42      // 向量点积  a • b = |a| |b|  cos ,   为a, b夹角, a*b<0,说明夹角是钝角, =0垂
          直, 大于0是锐角
43      double operator *(const Point &b)const {
44          return x*b.x + y*b.y;
45      }
46      //到原点的距离
47      double len() {
48          return hypot(x,y);
49      }
50      double len2() {
51          return x*x + y*y;
52      }
53
54      // 当前点到点 p 的距离
55      double distance(Point p) {
56          return hypot(x-p.x,y-p.y);
57      }
58      Point operator +(const Point &b)const{
59          return Point(x+b.x,y+b.y);
60      }
61      Point operator *(const double &k)const{
62          return Point(x*k,y*k);
63      }
64      Point operator /(const double &k)const{
65          return Point(x/k,y/k);
66      }
67      //返回的是度数
68      double atn2() {
69          poe = atan2(y, x);
70          return poe;
71      }
72  };
```

7.2 直线模板

```
1  struct Line{
2      Point s,e;
3      double poe;
```

```
4       Line(){}
5       Line(Point _s,Point _e){
6           s = _s;
7           e = _e;
8       }
9       bool operator ==(Line v){
10          return (s == v.s)&&(e == v.e);
11      }
12      void input(){
13          s.input();
14          e.input();
15      }
16      // 线段长度
17      double length(){
18          return s.distance(e);
19      }
20      //与x轴的交点
21      double x_len() {
22          return e.x - (s.x - e.x) * e.y / (s.y - e.y);
23      }
24      //与y轴的交点
25      double y_len() {
26          return e.y - (s.x - e.x) * e.x / (s.y - e.y);
27      }
28      // 点 p 是否在当前线段上
29      bool pointonseg(Point p){
30          //第一个是判断直线上（叉积），第二个是线段内（点积）(钝角小于0)
31          return sgn((p-s)^(e-s)) == 0 && sgn((p-s)*(p-e)) <= 0;
32      }
33      // 两条直线的交点  （如果要线段的交点，则在前面判定两线段是否相交inter函数 ，
          如果要直线和线段的交点，则在前面判定Seg_inter_line）
34      pair<Point,int> operator &(const Line &b) const{
35          Point res = s;
36          if(sgn((s-e)^(b.s-b.e))==0)//共线或平行
37          {
38              if(sgn((b.s -s)^(b.e-s)) == 0) return make_pair(res,0);//重合
39              else return make_pair(res,1);//平行
40          }
41          double k = ((s-b.s)^(b.s - b.e))/((s-e)^(b.s - b.e));
42          //res.x += (e.x - s.x) *k;
43          //res.y +=(e.y - s.y) *k;
44          res =  res+(e-s)*k;
45          return make_pair(res,2);//相交
46      }
47      double atn2() {
48          poe = atan2(e.y - s.y, e.x - s.x);
49          return poe;
50      }
51  };
```

7.3 圆模板

```
1   truct circle
2   {
3       Point o;
4       double r;
5       void input(){
6           o.input();
7           scanf("%lf", &r);
8       }
9       // 点 p 是否在当前圆内或上
10      bool pointincir(Point p){
11          return o.distance(p) <= r;
12      }
13      Point PointOncirle(double t) //圆上任意一点，t是从最顶上的点开始的角度
```

```
14      {
15          return Point(o.x + r * cos(t), o.y + r * sin(t));
16      }
17  };
```

7.4 两点距离

```
1  double dist(Point a,Point b){
2      return sqrt((b-a)*(b-a));
3  }
```

7.5 叉积

```
1  //叉积，判断p相对于a->b向量的位置，>0,p在ab的顺时针，<0,逆时针，=0在ab直线上
2  //也可理解为>0时，a->p 在a->b的顺时针方向，其几何意义就是以两个向量为边的平行四边形的面积
3  double xmult(Point p ,Point a,Point b) {
4      return (p-a)^(b-a);
5  }
```

7.6 点积

```
1  //点积  大于0， bpa是锐角，小于0是钝角，=0垂直
2  double dot(Point p ,Point a,Point b) {
3      return (a-p)*(b-p);
4  }
```

7.7 两线段是否相交

```
1  //这是判断两线段是否相交，判断直线是否相交们只需判断是否平行就行（线里面的函数）
2  int inter(Line l1,Line l2){
3      return
4      max(l1.s.x,l1.e.x) >= min(l2.s.x,l2.e.x)
5      && max(l2.s.x,l2.e.x) >= min(l1.s.x,l1.e.x)
6      && max(l1.s.y,l1.e.y) >= min(l2.s.y,l2.e.y)
7      && max(l2.s.y,l2.e.y) >= min(l1.s.y,l1.e.y)
8      && sgn((l2.s-l1.s)^(l1.e-l1.s))*sgn((l2.e-l1.s)^(l1.e-l1.s))<=0
9      && sgn((l1.s-l2.s)^(l2.e-l2.s))*sgn((l1.e-l2.s)^(l2.e-l2.s))<=0;
10 }
```

7.8 判断直线和线段相交

```
1  //判断直线和线段相交
2  bool Seg_inter_line(Line l1,Line l2) //判断直线l1和线段l2是否相交
3  {
4      //就是判断l2的两个端点是否在l1的两边，用叉积来计算l2线段的两个端点在l1直线同侧还是异侧
5      //return sgn((l2.s-l1.e)^(l1.s-l1.e))*sgn((l2.e-l1.e)^(l1.s-l1.e)) <= 0;
6      return sgn(xmult(l2.s,l1.s,l1.e)*sgn(xmult(l2.e,l1.s,l1.e))<=0);//判断l1 和l2是否相交，用叉积来计算l2线段的两个端点在l1直线同侧还是异侧
7  }
```

7.9 判断点 p 是否在线段 l 上

```
int onseg(Point p,Line l)//判断点p是否在线段l上
{
    return
        sgn(xmult(p,l.s,l.e))==0 &&//叉积,或者 sgn((l.s-p)^(l.e-p))==0 &&
        sgn(p.x - l.s.x) * (p.x-l.e.x)<=0 &&
        sgn(p.y - l.s.y) * (p.y - l.e.y)<=0;
}
```

7.10 判断凸多边形

```
//允许共线边
//点可以是顺时针给出也可以是逆时针给出
//点的编号1~n-1
bool isconvex(Point poly[],int n) {
    bool s[3];
    memset(s,false,sizeof(s));
    for(int i=0;i<n;i++) {
        //如果是凸多边形，那么poly[(i+2)]一直在poly[(i+1)]的逆时针，一直都是s[2]
          true，一旦在顺时针了，s[0]也true就return false;
        s[sgn((poly[(i+1)%n]-poly[i])^(poly[(i+2)%n]-poly[i]))+1] =true;
        if(s[0]&&s[2]) return false;
    }
    return true;
}
```

7.11 判断点在任意多边形内

```
int inPoly(Point p,Point poly[],int n) {
    int cnt;
    Line ray,side;
    cnt = 0;
    ray.s = p;
    ray.e.y = p.y;
    ray.e.x = -100000000000.0;//-INF,注意取值防止越界

    for(int i = 0;i < n;i++) {
        side.s = poly[i];
        side.e = poly[(i+1)%n];

        if(onseg(p,side))return 0;

        //如果平行轴则不考虑
        if(sgn(side.s.y - side.e.y) == 0)
            continue;

    if(onseg(side.s,ray)) {
        if(sgn(side.s.y - side.e.y) > 0)cnt++;
    } else if(onseg(side.e,ray)) {
        if(sgn(side.e.y - side.s.y) > 0)cnt++;
    } else if(inter(ray,side))
        cnt++;
    }
    if(cnt % 2 == 1)return 1;
    else return -1;
}
```

7.12 点到线段最近的点

```
Point NearestPointToLineSeg(Point P,Line L) {
    Point result;
    double t = ((P-L.s)*(L.e-L.s))/((L.e-L.s)*(L.e-L.s));
    //点到直线的距离，即点在线段的上方
    if(t >= 0 && t <= 1) {
        result.x = L.s.x + (L.e.x - L.s.x)*t;
        result.y = L.s.y + (L.e.y - L.s.y)*t;
        //cout<<1;
    }
    else//点不在线段上方
    {
        if(dist(P,L.s) < dist(P,L.e))
            result = L.s;
        else result = L.e;
        //cout<<2;
    }
    return result;
}
```

7.13 点到线段的距离

```
double dian_dao_xianduan(Line L,Point P) {
    Point result;
    double t = ((P-L.s)*(L.e-L.s))/((L.e-L.s)*(L.e-L.s));
    //点到直线的距离，即点在线段的上方
    if(t >= 0 && t <= 1) {
        result.x = L.s.x + (L.e.x - L.s.x)*t;
        result.y = L.s.y + (L.e.y - L.s.y)*t;
        //cout<<1;
    }
    else//点不在线段上方
    {
        if(dist(P,L.s) < dist(P,L.e))
            result = L.s;
        else result = L.e;
        //cout<<2;
    }
    return result.distance(P);
}
```

7.14 多边形模板

```
//求一个多边形的面积，用叉积求，叉积是两个向量合成一个四边形的面积
struct polygon{
    int num;
    Point po[20];
    void add(Point q){po[++num]=q;}
    double area() //多边形面积
    {
        double ans = 0;
        po[0] = po[num];
        for (int i = 0; i < num; i++)
            ans += (po[i] ^ po[i + 1]);
        ans = fabs(ans) * 0.5;
        return ans;
    }
    double zhouchang() {
        po[0] = po[num];
        double sum = 0;
        for (int i = 0; i < num; i++)
            sum += (po[i] - po[i + 1]).len();
        return sum;
```

```
    }
};
```

7.15 三角形 abc 的外接圆圆心

```
Point circle_center( Point a,  Point b,  Point c)
{
    Point center;
    double a1 = b.x - a.x, b1 = b.y - a.y, c1 = (a1 * a1 + b1 * b1) / 2;
    double a2 = c.x - a.x, b2 = c.y - a.y, c2 = (a2 * a2 + b2 * b2) / 2;
    double d = a1 * b2 - a2 * b1;
    center.x = a.x + (c1 * b2 - c2 * b1) / d;
    center.y = a.y + (a1 * c2 - a2 * c1) / d;
    return center;
}
```

7.16 最小圆覆盖

```
// 返回能覆盖所有点的半径最小圆
circle min_cover_circle(Point p[] ,int n) {
    random_shuffle(p, p + n);              ///打乱所有点
    circle cc;
    Point c = p[0];
    double r = 0.0;

    for(int i =1;i<n;i++) {
        if (sgn(dist(p[i], c) - r) > 0) ///pi在圆外部
        {
            c = p[i];
            r = 0;                  ///将圆心设为pi半径为0
            for (int j = 0; j < i; ++j) ///重新检查前面的点
            {
                if (sgn(dist(p[j], c) - r) > 0)///两点定圆
                {
                    c.x = (p[i].x + p[j].x) / 2;
                    c.y = (p[i].y + p[j].y) / 2;
                    r = dist(p[j], c);
                    for (int k = 0; k < j; ++k) {
                        if (sgn(dist(p[k], c) - r) > 0) {
                            c = circle_center(p[i], p[j], p[k]);
                            r = dist(p[i], c);
                        }
                    }
                }
            }
        }
    }
    cc.o = c;
    cc.r = r;
    return cc;
}
```

7.17 浮点型 gcd

```
double gcd(double x, double y) //浮点型gcd
{
    while (fabs(x) > eps && fabs(y) > eps) {
        if (x > y)
            x -= floor(x / y) * y;
        else
            y -= floor(y / x) * x;
    }
```

```
9     return x + y;
10 }
```

7.18 极角排序

```
1  bool cmp(vec a, vec b) //极角排序,通过叉积进行极角排序
2  {
3      vec c(0, 0);
4      if (sgn((a - c) ^ (b - c)) == 0)
5          return a.x < b.x;
6      return sgn((a - c) ^ (b - c)) > 0;
7  }
8  bool cmp2(vec a, vec b) {
9      return a.y < b.y;
10 }
11 bool cmp1( Point A, Point B)
12 {
13     if(A.x==B.x)
14         return A.y<B.y;
15     else
16         return A.x<B.x;
17 }
18 Point p[maxp];
19 Point temp[maxp];
20 int pos;
21 ///使用atan2会丢失精度，尽量不用这种
22 bool cmp3(Point a,Point b)
23 {
24
25     //double tmp  = (a-p[pos])^(b-p[pos]);
26     Point aa = a - p[pos];
27     Point bb = b - p[pos];
28     double tmp = atan2(aa.y,aa.x) - atan2(bb.y,bb.x);
29     if(sgn(tmp)==0) return dist(a,p[pos])<dist(b,p[pos]);
30     else if(sgn(tmp)<0) return true;//说明pa在pb右边，需更靠近
31     else return false;
32 }
33 bool cmp4(Point a,Point b) //这种就是上面cmp那种，用叉积的，只不过等于的时候，距离
     p[pos]近的排前面
34 {
35     double tmp  = (a-p[pos])^(b-p[pos]);
36     if(sgn(tmp)==0) return dist(a,p[pos])<dist(b,p[pos]);
37     else if(sgn(tmp)>0) return true;//说明pa在pb右边，需更靠近
38     else return false;
39 }
40
41 int qua(Point a)
42 {
43     if(a.x>0&&a.y>=0)  return 1;
44     if(a.x<=0&&a.y>0)  return 2;
45     if(a.x<0&&a.y<=0)  return 3;
46     if(a.x>=0&&a.y<0)  return 4;
47
48 }
49
50
51 bool cmp5(Point a,Point b)  //先按象限从小到大排序 再按极角从小到大排序
52 {
53     if(qua(a)==qua(b))//返回值就是象限
54         return cmp(a,b);
55     else return qua(a)<qua(b);
56 }
```

7.19 平面最近点对

```
// 返回距离
double mindist(int l, int r) //平面最近点对
{
    if (l == r)
        return 2<<20;
    if(l+1==r)
        return dist(p[l],p[r]);
    int mid = (l + r) >> 1;

    double ans = mindist(l, mid);
    ans = min(ans, mindist(mid + 1, r));

    int tot = 0;
    for (int i = l; i <= r; i++)
        if (fabs(p[mid].x - p[i].x) <= ans)
            temp[tot++] = p[i];

    sort(temp, temp + tot, cmp2);

    for (int i = 0; i < tot; i++)
        for (int j = i + 1; j < tot; j++) {
            if (temp[j].y - temp[i].y >=ans)
                break;
            ans = min(ans, dist(temp[j],temp[i]));
        }
    return ans;
}
void mindistsolve() {
    int n;
    scanf("%d", &n);
    for(int i = 0; i < n; ++i)
        p[i].input();
    sort(p,p+n,cmp1);
    double ans = mindist(0,n-1);
    printf("%.4f\n", ans);
}
```

7.20 求多边形的重心

```
double sanjiaoxingarea(Point p0,Point p1,Point p2) {
    double area=0;
    area=p0.x*p1.y+p1.x*p2.y+p2.x*p0.y-p1.x*p0.y-p2.x*p1.y-p0.x*p2.y;
    return area/2;
}
///上面的快一点，下面的慢一点
double sanjiaoxingarea2(Point p0,Point p1,Point p2) {
    return(p1-p0)^(p2-p0)/2;
}

Point duo_zhongxin(Point po[], int n) //多边形重心
{
    Point p0,p1,p2;
    double sumarea=0,sumx=0,sumy=0;
    p0 = po[0];p1 = po[1];
    for(int i=2;i<n;i++) {
        p2 = po[i];
        double area=sanjiaoxingarea2(p0,p1,p2);//求新添三角形的面积
        sumarea+=area;
        sumx+=(p0.x+p1.x+p2.x)*area;
        sumy+=(p0.y+p1.y+p2.y)*area;
        p1=p2;//求总面积
    }
    p2.x = sumx/sumarea/3;
```

```
25      p2.y = sumy/sumarea/3;
26      return p2;
27  }
28
29  //另外一种
30  Point Gravity(Point p[], int n){//返回多边形的重心
31      double area = 0;
32      Point center;
33      center.x = 0;
34      center.y = 0;
35      for (int i = 0; i < n-1; i++) {
36          area += (p[i].x*p[i+1].y - p[i+1].x*p[i].y)/2;
37          center.x += (p[i].x*p[i+1].y - p[i+1].x*p[i].y) * (p[i].x + p[i+1].x);
38          center.y += (p[i].x*p[i+1].y - p[i+1].x*p[i].y) * (p[i].y + p[i+1].y);
39      }
40      area += (p[n-1].x*p[0].y - p[0].x*p[n-1].y)/2;
41      center.x += (p[n-1].x*p[0].y - p[0].x*p[n-1].y) * (p[n-1].x + p[0].x);
42      center.y += (p[n-1].x*p[0].y - p[0].x*p[n-1].y) * (p[n-1].y + p[0].y);
43      center.x /= 6*area;
44      center.y /= 6*area;
45      return center;
46  }
```

7.21 将多边形的点逆时针排序

```
1   // 未用例题证明
2   Point center;
3   bool PointCmp(const Point &a,const Point &b) {
4       if (a.x >= 0 && b.x < 0)
5           return true;
6       if (a.x == 0 && b.x == 0)
7           return a.y > b.y;
8       //向量OA和向量OB的叉积
9       int det = (a.x - center.x) * (b.y - center.y) - (b.x - center.x) * (a.y -
          center.y);
10      if (det < 0)
11          return true;
12      if (det > 0)
13          return false;
14      //向量OA和向量OB共线，以距离判断大小
15      int d1 = (a.x - center.x) * (a.x - center.x) + (a.y - center.y) * (a.y -
          center.y);
16      int d2 = (b.x - center.x) * (b.x - center.y) + (b.y - center.y) * (b.y -
          center.y);
17      return d1 > d2;
18  }
19  bool Cmp(const Point &a,const Point &b){
20      return !PointCmp(a,b);
21  }
22  void ClockwiseSortPoints(Point* vPoints, int n) {
23      //计算重心
24      center=Gravity(vPoints,n);
25      sort(vPoints,vPoints+n,Cmp);
26  }
```

7.22 凸包

```
1   // 栈s就是凸包，具体使用看后面例题
2   //原始点
3   Point p[MAXN];
4   //凸包的栈
5   Point s[MAXN];
6   int n,top;
```

```
bool cmp(Point p1,Point p2) {
    int tmp = xmult(p1,p[0],p2);
    if(sgn(tmp)>0) return true;
    else if(sgn(tmp)==0 && dist(p[0],p1)<dist(p[0],p2)) return true;
    else return false;
}

void graham() {
    Point p0 = p[0];
    int k = 0;
    //找到最左下角的点
    for(int i=1;i<n;i++) {
        if( (p0.y>p[i].y) || ((p0.y==p[i].y)&&(p0.x>p[i].x)) ) {
            p0 = p[i];
            k=i;
        }
    }
    p[k] = p[0];
    p[0] = p0;
    //其他点极角排序
    sort(p+1,p+n,cmp);
    if(n==1) {
        top = 0;s[0] = p[0];
    }
    if(n==2) {
        top =1;
        s[0] =p[0];
        s[1] = p[1];
    }
    if(n>2) {
        top =1;
        s[0] =p[0];
        s[1] = p[1];
        for(int i=2;i<n;i++) {
            while(top>0 && sgn(xmult(s[top],s[top-1],p[i]))<=0) top--; //注意这里要<=0，要把共线的也去掉，但poj1228就要<0,要把共线的也要进栈
            top++;
            s[top] = p[i];
        }
    }
    //    //底下这个玩意用来输出凸包上点的坐标
    //   for(int i=0;i<=top;++i)
    //       printf("(%d,%d)\n",s[i].x,s[i].y);
}
```

7.23 凸包直径，旋转卡壳

```
//选择卡壳得到凸包的直径，凸包上最远的两个点的距离（先graham求得凸包，p是原始点，s是凸包的点），要特判只有两个点的情况
double rotating_caliper(){
    double ans = 0;
    if(n==3){
        ans=max(dist(p[0],p[1]),dist(p[0],p[2]));
        ans=max(ans,dist(p[1],p[2]));
    } else {
        int j =2;
        s[top+1] = s[0];//这里好像做不做都行
        for(int i=0;i<=top;i++) {
            while(abs(xmult(s[i],s[i+1],s[j]))<abs(xmult(s[i],s[i+1],s[j+1])))
            //while(abs((s[i]-s[i+1])^(s[j]-s[i+1]))<abs((s[i]-s[i+1])^(s[j+1]-s[i+1])))
                j = (j+1)%(top+1);//上面不做top，这里应该是top，不是top+1
            ans = max(ans,dist(s[i],s[j]));
        }
    }
```

```
17      return ans;
18  }
```

7.24 半平面交

```
1   // 求多边形的核是否存在，注意点是逆时针给的，如果是顺时针要reverse一下，具体使用看
      下面例题
2   //上面Line模板里面换下相交的重载符&
3   Point operator &(const Line &b) const{
4       Point res = s;
5       double k = ((s-b.s)^(b.s - b.e))/((s-e)^(b.s - b.e));
6       res =  res+(e-s)*k;
7       return res;
8   }
9
10  Line Q[MAXN];  //直线的双端队列
11  Point p[MAXN]; //记录最初给的点集
12  Line line[MAXN]; //由最初的点集生成直线的集合
13  Point pp[MAXN]; //记录半平面交的结果的点集
14
15  //半平面交，直线的左边代表有效区域
16  bool HPIcmp(Line a,Line b) { //直线排序函数
17      if(fabs(a.k - b.k) > eps)return a.k < b.k; //斜率排序
18      //斜率相同我也不知道怎么办
19      return ((a.s - b.s)^(b.e - b.s)) < 0;
20  }
21
22  void HPI(Line line[], int n, Point res[], int &resn)
23  { //line是半平面交的直线的集合 n是直线的条数 res是结果
24      //的点集 resn是点集里面点的个数,res是一个多边形的区域
25      int tot = n;
26      sort(line,line+n,HPIcmp);
27      tot = 1;
28      for(int i = 1;i < n;i++)
29          if(fabs(line[i].k - line[i-1].k) > eps) //去掉斜率重复的
30              line[tot++] = line[i];
31      int head = 0, tail = 1;
32      Q[0] = line[0];
33      Q[1] = line[1];
34      resn = 0;
35      for(int i = 2; i < tot; i++)
36      { //如果双端队列底端或顶端两个半平面的交点在当前半平面之外，则删除
37          if(fabs((Q[tail].e-Q[tail].s)^(Q[tail-1].e-Q[tail-1].s)) < eps || fabs((Q[
              head].e-Q[head].s)^(Q[head+1].e-Q[head+1].s)) < eps)
38              return;
39          //判断顶端的两个半平面的交点是否在加入线右边
40          while(head < tail && (xmult(Q[tail]&Q[tail-1],line[i].s,line[i].e)) > eps)
41              tail--;
42          //判断底端的两个半平面的交点是否在加入线右边
43          while(head < tail && (xmult(Q[head]&Q[head+1],line[i].s,line[i].e)) > eps)
44              head++;
45          Q[++tail] = line[i];
46      }
47      //最后判断最先加入的直线和最后的直线的影响
48      while(head < tail && (xmult(Q[tail]&Q[tail-1],Q[head].s,Q[head].e)) > eps)
49          tail--;
50      while(head < tail && (xmult(Q[head]&Q[head+1],Q[tail].s,Q[tail].e)) > eps)
51          head++;
52      if(tail <= head + 1) return;
53      //保存点到 res数组
54      for(int i = head; i < tail; i++)
55          res[resn++] = Q[i]&Q[i+1];
56      if(head < tail - 1)
57          res[resn++] = Q[head]&Q[tail];
58  }
```

7.25 半平面交的 cut 模板

```
// poj3335解，cut用起来比较麻烦，但有时必须要用cut，比如求不等式方程是否有可行解（这里是默认点是顺时针的）
//有时精度要开到1e-18
const double eps = 1e-18;
//通过两点，确定直线方程
void Get_equation(Point p1,Point p2,double &a,double &b,double &c) {
    a = p2.y - p1.y;
    b = p1.x - p2.x;
    c = p2.x*p1.y - p1.x*p2.y;
}
//求交点  直线p1p2和直线系数为abc的交点
Point Intersection(Point p1,Point p2,double a,double b,double c) {
    double u = fabs(a*p1.x + b*p1.y + c);
    double v = fabs(a*p2.x + b*p2.y + c);
    Point t;
    t.x = (p1.x*v + p2.x*u)/(u+v);
    t.y = (p1.y*v + p2.y*u)/(u+v);
    return t;
}
//计算多边形面积
double CalcArea(Point p[],int n) {
    double res = 0;
    for(int i = 0;i < n;i++)
        res += (p[i]^p[(i+1)%n]);
    return fabs(res/2);
}
//每次cut的临时点数组
Point tp[110];
//最后的核的点
Point p[110];
//初始点
Point pp[110];
int n;//初始点的个数
void Cut(double a, double b, double c, Point p[], int &cnt)//用直线ax+by+c==0切割多边形,返回核点和点的数量,第一个传进来的cnt必须要是原始点的数量
{
    int tmp = 0;
    for(int i = 1; i <= cnt; ++i) {
        if(sgn(a*p[i].x + b*p[i].y + c) >=0)//题目是顺时钟给出点的，所以一个点在直线右边的话，那么带入值就会大于等于0，说明这个点还在切割后的多边形内，将其保留
            tp[++tmp] = p[i];
        else {
            if(sgn(a*p[i-1].x + b*p[i-1].y + c) >0)//判断这个点前后的两个点，它和它相邻的点构成直线与，ax+by+c==0所构成的交点可能在新切割出的多边形内，
                tp[++tmp] = Intersection(p[i-1], p[i], a, b, c);
            if(sgn(a*p[i+1].x + b*p[i+1].y + c) >0)
                tp[++tmp] = Intersection(p[i], p[i+1], a, b, c);
        }
    }
    for(int i = 1; i <= tmp; ++i)//更新切割后的多边形
        p[i] = tp[i];
    p[0] = p[tmp];
    p[tmp+1] = p[1];
    cnt = tmp;
}


void solve() {
    //初始化，pp一直都是初始点，而p每cut一次都可能变
```

```
    for(int i=1;i<=n;i++) {
        p[i]=pp[i];
    }
    pp[n+1]=pp[1];
    p[n+1]=p[1];
    p[0]=p[n];
    int resn=n;

    for(int i=1;i<=n;i++) {
        double a,b,c;
        Get_equation(pp[i],pp[i+1],a,b,c);
        Cut(a,b,c,p,resn);
    }
    if(resn) printf("YES\n"); //如果最终点集中点的个数不为 0
    else printf("NO\n");
}

int main() {
    int t;
    scanf("%d",&t);
    while(t--) {
        scanf("%d",&n);
        for(int i=1;i<=n;i++) {
            pp[i].input();
        }
    solve();
    }
    return 0;
}
```

7.26 将线段平移

```
// 有些cut里面会用到, 看后面例题
void change(Point a,Point b,Point &c,Point &d,double p)
{ //将线段ab往左移动距离p,修改得到线段cd
    double len=dist(a,b);
    /*三角形相似推出下面公式*/
    double dx=(a.y-b.y)*p/len;
    double dy=(b.x-a.x)*p/len;
    c.x=a.x+dx; c.y=a.y+dy;
    d.x=b.x+dx; d.y=b.y+dy;
}
void change2(Point a,Point b,Point &c,Point &d,double p)
{ //将线段ab往右移动距离p,修改得到线段cd
    double len=dist(a,b);
    /*三角形相似推出下面公式*/
    double dx=(b.y-a.y)*p/len;
    double dy=(a.x-b.x)*p/len;
    c.x=a.x+dx; c.y=a.y+dy;
    d.x=b.x+dx; d.y=b.y+dy;
}
```

7.27 例题

7.27.1 凸包周长

```
#define MAXN 110
using namespace std;
const double eps = 1e-8;
const double inf = 1e20;
const double pi = acos(-1.0);
const int maxp = 1010;
//求凸包的周长
```

```
8  // 精度三态函数
9  int sgn(double x) {
10     if(fabs(x) < eps) return 0;
11     if(x < 0) return -1;
12     else return 1;
13 }
14 typedef  struct Point vec; // 向量
15 //省略点模板
16 struct Point {
17     double x, y;
18 };
19 double dist(Point a,Point b){
20     return sqrt((b-a)*(b-a));
21 }
22
23 //叉积，判断p相对于a->b向量的位置，>0,p在ab的顺时针，<0,逆时针，=0在ab直线上
24 double xmult(Point p ,Point a,Point b) {
25     return (p-a)^(b-a);
26 }
27 //点积  大于0， bpa是锐角，小于0是钝角，=0垂直
28 double dot(Point p ,Point a,Point b) {
29     return (a-p)*(b-p);
30 }
31
32 //原始点
33 Point p[MAXN];
34 //凸包的栈
35 Point s[MAXN];
36 int n,top;
37 bool cmp(Point p1,Point p2) {
38     int tmp = xmult(p1,p[0],p2);
39     if(sgn(tmp)>0) return true;
40     else if(sgn(tmp)==0 && dist(p[0],p1)<dist(p[0],p2)) return true;
41     else return false;
42 }
43
44 void graham() {
45     Point p0 = p[0];
46     int k = 0;
47     //找到最左下角的点
48     for(int i=1;i<n;i++) {
49         if( (p0.y>p[i].y) || ((p0.y==p[i].y)&&(p0.x>p[i].x)) ) {
50             p0 = p[i];
51             k=i;
52         }
53     }
54     p[k] = p[0];
55     p[0] = p0;
56     //其他点极角排序
57     sort(p+1,p+n,cmp);
58     if(n==1) {
59         top = 0;s[0] = p[0];
60     }
61     if(n==2) {
62         top =1;
63         s[0] =p[0];
64         s[1] = p[1];
65     }
66     if(n>2) {
67         top =1;
68         s[0] =p[0];
69         s[1] = p[1];
70         for(int i=2;i<n;i++) {
71             while(top>0 && sgn(xmult(s[top],s[top-1],p[i]))<=0) top--;
72             top++;
73             s[top] = p[i];
```

```
        }
    }
    //    //底下这个玩意用来输出凸包上点的坐标
    //   for(int i=0;i<=top;++i)
    //       printf("(%d,%d)\n",s[i].x,s[i].y);
}
void getans() {
    double ans=0.0;
    if(top==0) ans=0.0;
    else if(top==1)
    ans = dist(s[0],s[1]);
    else{
        for(int i=0;i<=top;i++)
            ans+=dist(s[i],s[(i+1)%(top+1)]);
    }
    printf("%.2f\n",ans);
}
int main() {
    while(scanf("%d",&n)&&n) {
        for(int i=0;i<n;i++)
            p[i].input();
        graham();
        getans();
    }
    return 0;
}
```

7.27.2 半平面交 cut

```
// poj3384
//思路：这个题是在 POJ 3525 题基础上才能做的。我们已经知道了求解多边形中可以放入的一个最大的圆的半径，现在是放入两个相同的圆。我们知道，两个圆相离越近重叠面积越大，覆盖的总面积越小，也就是说找一个圆心距最大的两点就可以了。//基于前一个题，我们知道可以放入的最大圆的圆心（不是圆本身）一定在多边形的核内，放入两个圆时也是这样。假设两个圆的半径为 r ，我们就可以让多边形的每条边向内推进 r 的距离，这时候可以求出新的多边形的核，而这个核内距离最大的两点就是最大的圆心距。

using namespace std;
const double eps = 1e-18;
// 精度三态函数
int sgn(double x) {
    if(fabs(x) < eps) return 0;
    if(x < 0) return -1;
    else return 1;
}
typedef  struct Point vec; // 向量
struct Point {
    double x, y;
};

//通过两点，确定直线方程
void Get_equation(Point p1,Point p2,double &a,double &b,double &c) {
    a = p2.y - p1.y;
    b = p1.x - p2.x;
    c = p2.x*p1.y - p1.x*p2.y;
}
//求交点  直线p1p2和直线系数为abc的交点
Point Intersection(Point p1,Point p2,double a,double b,double c) {
    double u = fabs(a*p1.x + b*p1.y + c);
    double v = fabs(a*p2.x + b*p2.y + c);
    Point t;
    t.x = (p1.x*v + p2.x*u)/(u+v);
    t.y = (p1.y*v + p2.y*u)/(u+v);
    return t;
}
```

```
32 //计算多边形面积
33 double CalcArea(Point p[],int n) {
34     double res = 0;
35     for(int i = 0;i < n;i++)
36         res += (p[i]^p[(i+1)%n]);
37     return fabs(res/2);
38 }
39 //每次cut的临时点数组
40 Point tp[110];
41 //最后的核的点
42 Point p[110];
43 //初始点
44 Point pp[110];
45 //向内推进r后的点
46 Point ppp[110];
47 int n;//初始点的个数
48 double r;//半径
49 double dist(Point a,Point b) {
50     return sqrt((a-b)*(a-b));
51 }
52 void change(Point a,Point b,Point &c,Point &d,double p)
53 { //将线段ab往左移动距离p,修改得到线段cd
54     double len=dist(a,b);
55     /*三角形相似推出下面公式*/
56     double dx=(a.y-b.y)*p/len;
57     double dy=(b.x-a.x)*p/len;
58     c.x=a.x+dx; c.y=a.y+dy;
59     d.x=b.x+dx; d.y=b.y+dy;
60 }
61 void change2(Point a,Point b,Point &c,Point &d,double p)
62 { //将线段ab往右移动距离p,修改得到线段cd
63     double len=dist(a,b);
64     /*三角形相似推出下面公式*/
65     double dx=(b.y-a.y)*p/len;
66     double dy=(a.x-b.x)*p/len;
67     c.x=a.x+dx; c.y=a.y+dy;
68     d.x=b.x+dx; d.y=b.y+dy;
69 }
70
71 void Cut(double a, double b, double c, Point p[], int &cnt)//用直线ax+by+c==0切割
    多边形,返回核点和点的数量,第一个传进来的cnt必须要是原始点的数量
72 {
73     int tmp = 0;
74     for(int i = 1; i <= cnt; ++i) {
75         if(sgn(a*p[i].x + b*p[i].y + c) >=0)//题目是顺时钟给出点的, 所以一个点在直
              线右边的话, 那么带入值就会大于等于0, 说明这个点还在切割后的多边形内, 将
              其保留
76             tp[++tmp] = p[i];
77         else {
78             if(sgn(a*p[i-1].x + b*p[i-1].y + c) >0)//判断这个点前后的两个点, 它和
                  它相邻的点构成直线与, ax+by+c==0所构成的交点可能在新切割出的多边形
                  内,
79                 tp[++tmp] = Intersection(p[i-1], p[i], a, b, c);
80             if(sgn(a*p[i+1].x + b*p[i+1].y + c) >0)
81                 tp[++tmp] = Intersection(p[i], p[i+1], a, b, c);
82         }
83     }
84     for(int i = 1; i <= tmp; ++i)//更新切割后的多边形
85         p[i] = tp[i];
86     p[0] = p[tmp];
87     p[tmp+1] = p[1];
88     cnt = tmp;
89 }
90 void init() {
91     for(int i = 1;i<=n;i++) p[i] =pp[i];
92     p[n+1] = p[1];
```

```
93      p[0] = p[n];
94  }
95
96  void solve() {
97      //初始化p，pp一直都是初始点，而p每cut一次都可能变
98      pp[n+1]=pp[1];
99      init();
100     int resn=n;
101     for(int i=1;i <=n;i++)
102     { //将多边形的每条边向内移动 r 的距离,（顺时针）a，b向右移动
103         Point ta,tb;
104         change2(pp[i],pp[i+1],ta,tb,r);
105         double a,b,c;
106         Get_equation(ta,tb,a,b,c);
107         Cut(a,b,c,p,resn);
108     }
109     int x,y;
110     double dis;
111     x=0;
112     y=0;
113     dis=0;
114     //双重循环寻找距离最大的两个点
115     for(int i=1;i<=resn;i++)
116         for(int j=i+1;j<=resn;j++)
117             if(dist(p[i],p[j])>dis) {
118                 x=i;
119                 y=j;
120                 dis=dist(p[i],p[j]);
121             }
122     printf("%.4f %.4f %.4f %.4f\n",p[x].x,p[x].y,p[y].x,p[y].y);
123 }
124
125 int main() {
126     while(~scanf("%d%lf",&n,&r)) {
127         for(int i = 1;i <=n;i++)
128             pp[i].input();
129         solve();
130     }
131     return 0;
132 }
```

7.27.3 半平面交求不等式解 cut

```
1  const double eps = 1e-18;
2  // 精度三态函数
3  int sgn(double x) {
4      if(fabs(x) < eps) return 0;
5      if(x < 0) return -1;
6      else return 1;
7  }
8  typedef  struct Point vec; // 向量
9  //此次省略point的模板
10 struct Point {
11     double x, y;
12 };
13
14 //通过两点，确定直线方程
15 void Get_equation(Point p1,Point p2,double &a,double &b,double &c) {
16     a = p2.y - p1.y;
17     b = p1.x - p2.x;
18     c = p2.x*p1.y - p1.x*p2.y;
19 }
20 //求交点  直线p1p2和直线系数为abc的交点
21 Point Intersection(Point p1,Point p2,double a,double b,double c) {
22     double u = fabs(a*p1.x + b*p1.y + c);
```

```
23      double v = fabs(a*p2.x + b*p2.y + c);
24      Point t;
25      t.x = (p1.x*v + p2.x*u)/(u+v);
26      t.y = (p1.y*v + p2.y*u)/(u+v);
27      return t;
28  }
29  //计算多边形面积
30  double CalcArea(Point p[],int n) {
31      double res = 0;
32      for(int i = 0;i < n;i++)
33          res += (p[i]^p[(i+1)%n]);
34      return fabs(res/2);
35  }
36  //每次cut的临时点数组
37  Point tp[110];
38  double V[110],U[110],W[110];
39  int n;
40  const double INF = 100000000000.0;
41  //最后的核的点
42  Point p[110];
43  void Cut(double a, double b, double c, Point p[], int &cnt)//用直线ax+by+c==0切割
      多边形
44  {
45      int tmp = 0;
46      for(int i = 1; i <= cnt; ++i) {
47          if(a*p[i].x + b*p[i].y + c > eps)
48              tp[++tmp] = p[i];
49          else {
50              if(a*p[i-1].x + b*p[i-1].y + c > eps)//判断这个点前后的两个点
51                  tp[++tmp] = Intersection(p[i-1], p[i], a, b, c);
52              if(a*p[i+1].x + b*p[i+1].y + c > eps)
53                  tp[++tmp] = Intersection(p[i], p[i+1], a, b, c);
54          }
55      }
56      for(int i = 1; i <= tmp; ++i)//更新切割后的多边形
57          p[i] = tp[i];
58      p[0] = p[tmp];
59      p[tmp+1] = p[1];
60      cnt = tmp;
61  }
62
63  bool solve(int id) {
64      p[1] = Point(0,0);
65      p[2] = Point(INF,0);
66      p[3] = Point(INF,INF);
67      p[4] = Point(0,INF);
68      p[0] = p[4];
69      p[5] = p[1];
70      int cnt = 4;
71      for(int i = 0;i < n;i++)
72          if(i != id)
73          {   //化简，x/u1 + y/v1 + z/w1 < x/u2 + y/v2 + z/w2，然后是大于，所以cut里
              面是顺时针，大于
74              double a = (V[id] - V[i])/(V[i]*V[id]);
75              double b = (U[id] - U[i])/(U[i]*U[id]);
76              double c = (W[id] - W[i])/(W[i]*W[id]);
77              if(sgn(a) == 0 && sgn(b) == 0 && sgn(c)<=eps) {
78                  return false;
79              }
80              Cut(a,b,c,p,cnt);
81          }
82      if(sgn(CalcArea(p,cnt)) == 0)return false;
83      else return true;
84  }
85  int main() {
86      //freopen("in.txt","r",stdin);
```

```
87      //freopen("out.txt","w",stdout);
88      while(scanf("%d",&n) == 1) {
89          for(int i = 0;i < n;i++)
90              scanf("%lf%lf%lf",&V[i],&U[i],&W[i]);
91          for(int i = 0;i < n;i++) {
92              if(solve(i))printf("Yes\n");
93              else printf("No\n");
94          }
95      }
96      return 0;
97  }
```

7.27.4 半平面交求多边形能放得下的最大的圆

```
1   // poj3525
2   ////题意：求多边形内到边上距离最远的距离。可以转化为求多边形内可以放入的最大的圆的
      半径。思路：可以放入的最大圆的圆心（不是圆本身）一定在多边形的核内，假设，最大圆
      半径为 r 。如果我们可以将多边形的每条边向内推进 r 长度的距离，这时候可以想见多边
      形里面放不开这个圆了，并且圆心所在的多边形的核也一定是变成了一个点。否则就多边形
      的边就还可以向内推进，这个圆就不是最大的了。当我们不知道最大圆半径的时候，我们可
      以用二分的方法求半径的值，如果对于当前假设的圆半径 r ，每条边向内推进 r 距离后，
      发现多边形的核不存在了，则说明当前圆半径太大了，反之，就是圆半径小了，用二分找到
      最大的圆半径即可。
3
4   #define MAXN 110
5   using namespace std;
6   const double eps = 1e-8;
7   const double inf = 1e20;
8   const double pi = acos(-1.0);
9   const int maxp = 1010;
10  // 精度三态函数
11  int sgn(double x) {
12      if(fabs(x) < eps) return 0;
13      if(x < 0) return -1;
14      else return 1;
15  }
16  typedef  struct Point vec; // 向量
17  //省略Point的模板
18  struct Point {
19      double x, y;
20  };
21  struct Line{
22      Point s,e;
23      double poe;//也是斜率，不过懒得改这个罢了
24      double k;//斜率
25      Line(){}
26      Line(Point _s,Point _e){
27          s = _s;
28          e = _e;
29          k = atan2(e.y - s.y,e.x - s.x);
30      }
31      bool operator ==(Line v){
32          return (s == v.s)&&(e == v.e);
33      }
34      void input(){
35          s.input();
36          e.input();
37      }
38      // 两条直线的交点  （如果要线段的交点，则在前面判定两线段是否相交inter函数 ，
          如果要直线和线段的交点，则在前面判定Seg_inter_line）
39      Point operator &(const Line &b) const{
40          Point res = s;
41          double k = ((s-b.s)^(b.s - b.e))/((s-e)^(b.s - b.e));
42          //res.x += (e.x - s.x) *k;
43          //res.y +=(e.y - s.y) *k;
```

```
44          res =  res+(e-s)*k;
45          return res;
46      }
47  };
48  double dist(Point a,Point b){
49      return sqrt((b-a)*(b-a));
50  }
51
52  ///这个要改改，看看要不要这样(p-a)^(b-a);更好对应>0,顺，<0逆
53  //叉积，判断p相对于a->b向量的位置，>0,p在ab的顺时针，<0,逆时针，=0在ab直线上
54  double xmult(Point p ,Point a,Point b) {
55      return (p-a)^(b-a);
56  }
57  //点积  大于0，  bpa是锐角，小于0是钝角，=0垂直
58  double dot(Point p ,Point a,Point b) {
59      return (a-p)*(b-p);
60  }
61  Line Q[MAXN];
62  Point p[MAXN]; //记录最初给的点集
63  Line line[MAXN]; //由最初的点集生成直线的集合
64  Point pp[MAXN]; //记录半平面交的结果的点集
65
66  //半平面交，直线的左边代表有效区域
67  bool HPIcmp(Line a,Line b)
68  { //直线排序函数
69      if(fabs(a.k - b.k) > eps)return a.k < b.k; //斜率排序
70      //斜率相同我也不知道怎么办
71      return ((a.s - b.s)^(b.e - b.s)) < 0;
72  }
73
74  void HPI(Line line[], int n, Point res[], int &resn)
75  { //line是半平面交的直线的集合 n是直线的条数 res是结果
76      //的点集 resn是点集里面点的个数,res是一个多边形的区域
77      int tot = n;
78      sort(line,line+n,HPIcmp);
79      tot = 1;
80      for(int i = 1;i < n;i++)
81          if(fabs(line[i].k - line[i-1].k) > eps) //去掉斜率重复的
82              line[tot++] = line[i];
83      int head = 0, tail = 1;
84      Q[0] = line[0];
85      Q[1] = line[1];
86      resn = 0;
87      for(int i = 2; i < tot; i++)
88      { //如果双端队列底端或顶端两个半平面的交点在当前半平面之外，则删除
89          if(fabs((Q[tail].e-Q[tail].s)^(Q[tail-1].e-Q[tail-1].s)) < eps || fabs((Q[
              head].e-Q[head].s)^(Q[head+1].e-Q[head+1].s)) < eps)
90              return;
91          //while(head < tail && (((Q[tail]&Q[tail-1]) - line[i].s)^(line[i].e-line[
              i].s)) > eps)  //判断顶端的两个半平面的交点是否在加入线右边
92          while(head < tail && (xmult(Q[tail]&Q[tail-1],line[i].s,line[i].e)) > eps)
93              tail--;
94          //while(head < tail && (((Q[head]&Q[head+1]) - line[i].s)^(line[i].e-line[
              i].s)) > eps)  //判断底端的两个半平面的交点是否在加入线右边
95          while(head < tail && (xmult(Q[head]&Q[head+1],line[i].s,line[i].e)) > eps)
96              head++;
97          Q[++tail] = line[i];
98      }
99      //最后判断最先加入的直线和最后的直线的影响
100     //while(head < tail && (((Q[tail]&Q[tail-1]) - Q[head].s)^(Q[head].e-Q[head].s
          )) > eps)
101     while(head < tail && (xmult(Q[tail]&Q[tail-1],Q[head].s,Q[head].e)) > eps)
102         tail--;
103     //while(head < tail && (((Q[head]&Q[head-1]) - Q[tail].s)^(Q[tail].e-Q[tail].e
          )) > eps)
104     while(head < tail && (xmult(Q[head]&Q[head+1],Q[tail].s,Q[tail].e)) > eps)
```

```
105         head++;
106     if(tail <= head + 1) return;
107     //保存点到 res数组
108     for(int i = head; i < tail; i++)
109         res[resn++] = Q[i]&Q[i+1];
110     if(head < tail - 1)
111         res[resn++] = Q[head]&Q[tail];
112 }
113 void change(Point a,Point b,Point &c,Point &d,double p)
114 { //将线段ab往左移动距离p,修改得到线段cd
115     double len=dist(a,b);
116     /*三角形相似推出下面公式*/
117     double dx=(a.y-b.y)*p/len;
118     double dy=(b.x-a.x)*p/len;
119     c.x=a.x+dx; c.y=a.y+dy;
120     d.x=b.x+dx; d.y=b.y+dy;
121 }
122
123 int n;
124 double BSearch() {
125     double l=0,r=100000;
126     while(r-l>=eps){
127         double mid = (l+r)/2;
128         for(int i=0;i<n;i++){
129             Point t1,t2;
130             change(p[i],p[(i+1)%n],t1,t2,mid);
131             line[i] = Line(t1,t2);
132         }
133         int resn;
134         HPI(line,n,pp,resn);
135         //等于0说明移多了
136         if(resn==0) r = mid-eps;
137         else l = mid+eps;
138     }
139     return l;
140 }
141
142 int main() {
143     while(~scanf("%d",&n)&&n) {
144         for(int i = 0;i < n;i++)
145             p[i].input();
146         printf("%.6f\n",BSearch());
147     }
148     return 0;
149 }
```

7.27.5 自适应辛普森法

计算

$$\int_L^R \frac{cx+d}{ax+b}dx$$

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  double a,b,c,d,l,r;
4  double f(double x) {
5      return (c*x+d)/(a*x+b);        //原函数
6  }
7
8  double simpson(double l,double r) {       //Simpson公式,都是这个公式
9      double mid=(l+r)/2;
10     return (f(l)+4*f(mid)+f(r))*(r-l)/6;
11 }
12
13 double asr(double l,double r,double eps,double ans) {
```

```
14      double mid=(l+r)/2;
15      double ll=simpson(l,mid),rr=simpson(mid,r);
16      if(fabs(ll+rr-ans)<=15*eps) return ll+rr+(ll+rr-ans)/15;      //确认精度
17      return asr(l,mid,eps/2,ll)+asr(mid,r,eps/2,rr);      //精度不够则递归调用
18  }
19  double asr(double l,double r,double eps) {
20      return asr(l,r,eps,simpson(l,r));
21  }
22  int main() {
23      scanf("%lf%lf%lf%lf%lf%lf",&a,&b,&c,&d,&l,&r);
24      printf("%.6f",asr(l,r,1e-6));
25      return 0;
26  }
```

7.28 其他模板

```
1   int bijiao(double x, double y) { //判断x>y
2       if (fabs(x - y) < eps)
3           return 0;
4       if (x > y)
5           return 1;
6       return -1;
7   }
8   //返回x>=y
9   int dayu_dengyu(double x, double y) {
10      if (fabs(x - y) < eps || x > y)
11          return 1;
12      return 0;
13  }
14  double zhixian_dian_juli(Line l, Point a) //直线到点的距离
15  {
16      return fabs(xmult(a,l.s,l.e)) / l.length();
17  }
18  double zhixian_dian_juli2(Line l, Point a) //直线到点的距离
19  {
20      return fabs((a - l.s) ^ (l.e - l.s)) / (l.e - l.s).len();
21  }
22  Line zhongchuixian(Line l) //求线段中垂线原理很简单,求两点的中点a,然后求个线段的向
      量v,再求垂直向量n,第二个点b=a+n
23  {
24      Point a = (l.s + l.e) / 2.0;
25      vec tp = l.e - l.s;
26      return Line(a, a + Point(-tp.y, tp.x));
27  }
28  Point zhongxin_duichen(Point a, Point b) //a为点, b为对称中心
29  {
30      return Point(2 * b.x - a.x, 2 * b.y - a.y);
31  }
32  double jiajiao(Point a, Point b, Point c) //a为角的顶点
33  {
34      vec A = b - c, B = a - c, C = b - a;
35      return acos((C * C + B * B - A * A) / (B.len() * C.len() * 2));
36  }
37  double jiajiao(vec u, vec v) //向量夹角
38  {
39      return acos((u * v) / (u.len() * v.len()));
40  }
41
42  Point touying(Line l1, Point p3) //c投影在直线ab上的位置
43  {
44      Point p1 = l1.s;
45      Point p2 = l1.e;
46      Point foot;
47      double dx = p1.x -p2.x;
48      double dy = p1.y -p2.y;
```

```
49      double u = ((p3.x - p1.x) * dx + (p3.y - p1.y) * dy) / (dx * dx + dy * dy);
50      foot.x = p1.x + u * dx;
51      foot.y = p1.y + u * dy;
52      return foot;
53  }
54  Point fanshe(Line l, Point c) //求c关于直线ab的对称点c'
55  {
56      Point A = touying(l, c);
57      return (A - c) * 2.0 + c;
58  }
59
60  double mianji(Point a, Point b, Point c) //三点面积
61  {
62      return fabs((b - a) ^ (c - a)) * 0.5;
63  }
64  double xianduanjuli(Line l1, Line l2) //两线段距离
65  {
66      if (inter(l1, l2))
67          return 0.0;
68      double minn = inf;
69      double l = dian_dao_xianduan(l1, l2.s);
70      minn = dayu_dengyu(minn, l) ? l : minn;
71      l = dian_dao_xianduan(l1, l2.e);
72      minn = dayu_dengyu(minn, l) ? l : minn;
73      l = dian_dao_xianduan(l2, l1.s);
74      minn = dayu_dengyu(minn, l) ? l : minn;
75      l = dian_dao_xianduan(l2, l1.e);
76      minn = dayu_dengyu(minn, l) ? l : minn;
77      return minn;
78  }
79  Point zhixian_zhixian_jiaodian(Line l1, Line l2) //两直线交点
80  {
81      //double t = ((l1.s.x - l2.s.x) * (l2.s.y - l2.e.y) - (l1.s.y - l2.s.y) * (l2.
          s.x - l2.e.x)) / ((l1.s.x - l1.e.x) * (l2.s.y - l2.e.y) - (l1.s.y - l1.e.y)
          * (l2.s.x - l2.e.x));
82      double t = ((l1.s-l2.s)^(l2.s - l2.e))/((l1.s-l1.e)^(l2.s - l2.e));
83      return l1.s + (l1.e - l1.s) * t;
84  }
85  int zhixian_yuan_jiaodian(circle c, Line l, Point &p1, Point &p2) { //直线与圆交点
      以及个数
86      if (zhixian_dian_juli(l, c.o) > c.r)
87          return 0;
88      Point pi = c.o;
89      pi.x += l.s.y - l.e.y;
90      pi.y += l.e.x - l.s.x;
91      pi = zhixian_zhixian_jiaodian(Line(pi, c.o), l);
92      double t = sqrt(c.r * c.r - (pi - c.o).len() * (pi - c.o).len()) / (l.s - l.e)
          .len();
93      p1 = pi + (l.e - l.s) * t;
94      p2 = pi - (l.e - l.s) * t;
95      return 1 + !(p1 == p2);
96  }
97  //线段是否和圆相交
98  bool xianduan_yuan_xiangjiao(Line seg,circle cir)
99  {
100     if(cir.pointincir(seg.s)||cir.pointincir(seg.e)) return true;
101     else{
102         Point d =touying(seg,cir.o);
103         double X1 = min(seg.s.x, seg.e.x), X2 = max(seg.s.x, seg.e.x);
104         double Y1 = min(seg.s.y, seg.e.y), Y2 = max(seg.s.y, seg.e.y);
105         double len1 = d.distance(cir.o);
106         //d在线段上且d和圆点的距离小于r
107         if((d.x >= X1 && d.x <= X2) && (d.y >= Y1 && d.y <= Y2) && (len1 <= cir.r)
              )
108             return true;
109     }
```

```
110     return false;
111 }
112 int xianduan_yuan_jiaodian(circle a, Line l, Point &p1, Point &p2) { //线段与圆交点以及个数
113     int tmp = zhixian_yuan_jiaodian(a, l, p1, p2);
114     if (tmp == 0)
115         return 0;
116     if (l.e < l.s)
117         swap(l.s, l.e);
118     return l.pointonseg(p1) + l.pointonseg(p2) - (p1 == p2);
119 }
120 double yuanxin_liangdian_jiajiao(Point a, Point b, circle c) //圆心与两点形成的夹角
121 {
122     Point _a = touying(Line(c.o, b), a);
123     double l = (a - _a).len();
124     return asin(l / c.r);
125 }
126 pair<Point, Point> yuan_yuan_jiaodian(circle a, circle b) //求圆和圆的交点
127 {
128     Point tmp =  a.o - b.o;
129     double l = tmp.len();
130     double x = acos((a.r * a.r + l * l - b.r * b.r) / (2.0 * a.r * l));
131     double t = atan2((b.o - a.o).y, (b.o - a.o).x);
132     return make_pair(a.o + vec(cos(t - x) * a.r, sin(t - x) * a.r), a.o + vec(cos(x + t) * a.r, sin(t + x) * a.r));
133 }
134 vec xiangliang_xuanzhuan(vec a, double pi) //将向量按照起点旋转逆时针pi
135 {
136     double x, y;
137     x = cos(pi) * a.x - sin(pi) * a.y;
138     y = sin(pi) * a.x + cos(pi) * a.y;
139     return vec(x, y);
140 }
141 pair<Point, Point> guodian_yuan_qiedian(circle a, Point p) //过一点做圆的切线求切点
142 {
143     Point tmp = a.o -p;
144     double l = tmp.len();
145     double t = asin(a.r / l);
146     double lb = l * cos(t);
147     vec x = a.o - p;
148     x = x / l * lb;
149     return make_pair(p + xiangliang_xuanzhuan(x, t), p + xiangliang_xuanzhuan(x, -t));
150 }
151 double yuan_yuan_xiangjiao(circle a, circle b) { //圆与圆相交面积,整个围住的时候会出错，前面要加个判断是否园内才行
152     double l = (a.o - b.o).len();
153     double ang1 = 2.0 * acos((a.r * a.r + l * l - b.r * b.r) / (2.0 * a.r * l));
154     double ang2 = 2.0 * acos((b.r * b.r + l * l - a.r * a.r) / (2.0 * b.r * l));
155     double s1 = 0.5 * a.r * a.r * sin(ang1), s2 = 0.5 * b.r * b.r * sin(ang2);
156     double h1 = 0.5 * ang1 * a.r * a.r, h2 = 0.5 * ang2 * b.r * b.r;
157     return h1 + h2 - s1 - s2;
158 }
159 int yuanyuan_waigongqiexian(Point p, circle a, circle b, Line *l) //求两圆的外公切线以及个数
160 {
161     int cnt = 0;
162     Point _o = b.o - a.o;
163     _o.atn2();
164     double base = _o.poe;
165     double d = (a.o - b.o).len(), ang = acos((a.r - b.r) / d);
166     l[cnt] = Line(a.PointOncirle(base - ang), b.PointOncirle(base - ang));
167     if (sgn((a.o - l[cnt].s) ^ (l[cnt].e - l[cnt].s)) == sgn((p - l[cnt].s) ^ (l[cnt].e - l[cnt].s))) //p和圆心在切线的同侧
```

```
    cnt++;
    l[cnt] = Line(a.PointOncirle(base + ang), b.PointOncirle(base + ang));
    if (sgn((a.o - l[cnt].s) ^ (l[cnt].e - l[cnt].s)) == sgn((p - l[cnt].s) ^ (l[
      cnt].e - l[cnt].s)))
    cnt++;
    return cnt;
}
int yuanyuan_gongqiexian(circle a, circle b, Point *u, Point *v) //求圆和圆公切线
  以及切线个数
{
    int cnt = 0;
    if (a.r < b.r) {
        swap(a, b);
        swap(u, v);
    }
    Point tmp = a.o - b.o;
    double l = tmp.len();
    double rdiff = a.r - b.r;
    double rsum = a.r + b.r;
    if (sgn(l - rdiff) < 0)
        return 0;
    double base = atan2((b.o - a.o).y, (b.o - a.o).x);
    if (sgn(l) == 0)
        return -1;
    if (sgn(l - rdiff) == 0) {
        u[cnt] = v[cnt] = a.PointOncirle(base);
        cnt++;
        return 1;
    }
    double ang = acos((a.r - b.r) / l);
    u[cnt] = a.PointOncirle(base + ang);
    v[cnt] = b.PointOncirle(base + ang);
    cnt++;
    u[cnt] = a.PointOncirle(base - ang);
    v[cnt] = b.PointOncirle(base - ang);
    cnt++;
    if (sgn(l - rsum) == 0) {
        u[cnt] = v[cnt] = a.PointOncirle(base);
        cnt++;
    } else if (sgn(l - rsum) > 0) {
        double ang = acos((a.r + b.r) / l);
        u[cnt] = a.PointOncirle(base + ang);
        v[cnt] = b.PointOncirle(pi + base + ang);
        cnt++;
        u[cnt] = a.PointOncirle(base - ang);
        v[cnt] = b.PointOncirle(pi + base - ang);
        cnt++;
    }
    return cnt;
}
```

7.29 三角形面积

```
\\ 海伦公式
int p = (a+b+c)/2;
int s = sqrt(p * (p - a) * (p - b) * (p - c));

\\ 两边和夹角
\\ a, b为边, x为a和b的夹角
s = 0.5 * a * b * sin(x/90.0*acos(0));
```

7.30 两圆面积交

```
const double PI = acos(-1);

struct circle
{
    double x, y, r;
};

// 计算圆心距
double dist(circle a, circle b)
{
    return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
}

double area(circle a, circle b)
{
    if((dist(a, b) + min(a.r, b.r)) <= max(a.r, b.r)) // 内含或重合
    {
        if(a.r < b.r)
            return PI*a.r*a.r;
        else
            return PI*b.r*b.r;
    }
    else if(dist(a, b) >= (a.r + b.r)) // 相切
    {
        return 0.0;
    }
    else
    {
        double length = dist(a, b);
        // 利用三角形余弦定理求圆心角
        double d1 = 2*acos((a.r*a.r+length*length-b.r*b.r)/(2*a.r*length));
        double d2 = 2*acos((b.r*b.r+length*length-a.r*a.r)/(2*b.r*length));
        // 利用圆心角求得扇形面积再减去三角形面积后两部分相加就是相交面积
        double area1 = a.r*a.r*d1/2 - a.r*a.r*sin(d1)/2;
        double area2 = b.r*b.r*d2/2 - b.r*b.r*sin(d2)/2;
        return area1 + area2;
    }
}
```

7.31 矩形相交

```
//一.相交面积==白色
//这做法看起来思路简单但是很多坑

//应该是两个黑白相交面积之和-黑黑白相交面积，而不是-黑黑相交面积！(当时就wa在这)

//求相交面积先判断是否相交（即x1,x2大小与y1,y2大小）
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

struct CC{
    ll x1,x2,y1,y2;
};

CC jiao(CC a,CC b)//相交矩形的坐标，可能不存在
{
    CC c1;
    c1.x1 = max(a.x1,b.x1);
    c1.y1 = max(a.y1,b.y1);
    c1.x2 = min(a.x2,b.x2);
    c1.y2 = min(a.y2,b.y2);
    return c1;
}
```

```
24
25 int main()
26 {
27     CC w,b1,b2;
28     cin>>w.x1>>w.y1>>w.x2>>w.y2>>b1.x1>>b1.y1>>b1.x2>>b1.y2>>b2.x1>>b2.y1>>b2.x2>>
         b2.y2;
29     CC j1 = jiao(w,b1);
30     CC j2 = jiao(w,b2);
31     CC j3 = jiao(j1,j2);
32     ll s,s1,s2,s3;
33     s= (w.x2-w.x1) * (w.y2-w.y1);//白色
34     if(j1.x1>j1.x2 || j1.y1>j1.y2) s1 = 0;//不想交则为0
35     else s1 = (j1.x2-j1.x1)*(j1.y2-j1.y1);
36     if(j2.x1>j2.x2 || j2.y1>j2.y2) s2 = 0;//不想交则为0
37     else s2 = (j2.x2-j2.x1)*(j2.y2-j2.y1);
38     if(j3.x1>j3.x2 || j3.y1>j3.y2) s3 = 0;//不想交则为0
39     else s3 = (j3.x2-j3.x1)*(j3.y2-j3.y1);
40
41     if(s1+s2-s3 == s) cout<<"NO"<<endl;
42     else cout<<"YES"<<endl;
43     return 0;
44 }
```

8 博弈论

8.1 巴什博奕

问题描述

只有一堆 n 个物品，两个人轮流从中取物，规定每次最少取一个，最多取 m 个，最后取光者为胜

博弈定理

若 $n\%(m+1)==0$ 后手必胜，否则先手必胜

代码模板

```
1 #include <iostream>
2 #include <cstdio>
3 using namespace std;
4 int t,n,m;
5 int main() {
6     scanf("%d", &t);
7     while(t--) {
8         scanf("%d%d", &n, &m);
9         if(n%(m+1)) printf("First Win\n");
10        else printf("Second Win\n");
11    }
12    return 0;
13 }
```

例题变式

Public Sale HDU-2149

题意

一块地在拍卖，刚开始底价为 0，两个人轮流开始加价，不过每次加价的幅度要在 1～N 之间，当价格大于或等于田地的成本价 M 时，主办方就把这块田地卖给这次叫价的人。A 每次都先开始叫价，问他第一次要出多少，才能保证一定能买到这块地，如果有多种可能，按小到大顺序输出。

思路

巴什博弈，直接枚举第一次可能取的数字，如果减去这个数字后，剩余的数满足巴什博弈推论，则满足要求

```
1 #include <iostream>
2 #include <cstdio>
3 using namespace std;
4 int m,n;
5 bool flag = 0;
6 int main() {
```

```
7     while(scanf("%d%d", &m, &n) != EOF) {
8         if(m <= n) {
9             for(int i = m; i <= n; ++i) {
10                if(i == m) printf("%d", i);
11                else printf(" %d", i);
12            }
13            printf("\n");
14            continue;
15        }
16        if(m%(n+1)) {
17            flag = 0;
18            for(int i = 1; i <= n; ++i) {
19                if((m-i)%(n+1) == 0) {
20                    if(flag == 0) printf("%d", i);
21                    else printf(" %d", i);
22                    flag++;
23                }
24            }
25            printf("\n");
26        } else {
27            printf("none\n");
28        }
29    }
30    return 0;
31 }
```

Buttons POJ-2368

题意

题意为一堆 K 粒纽扣，两个玩家轮流从堆中取纽扣，一次可以取 1 到 L 个纽扣，取得最后一个纽扣的人是赢家。让你找到数字最小的数字 L(2<L<K) 确保第二个人为赢家，如果不存在则输出 0

思路

很基础的巴什博弈，输入纽扣数 K，要求使后手能赢的最小的 L 值，其实就是找 $K\%(L+1) == 0$ 时 L 的最小值，只需找到 K 的大于 3 的最小因子 x , 那么 $L = x - 1$

```
1  /* 给定k，问取石子的范围L最小是多少，能让第二个人赢 */
2  #include <iostream>
3  #include <cstdio>
4  #include <cmath>
5  using namespace std;
6  typedef long long ll;
7
8  int k;
9
10 int main() {
11     scanf("%d", &k);
12     int fac = k;
13     for(int i = 2; i*i <= k; ++i) {
14         if(k%i == 0) {
15             if(i > 2) fac = min(fac, i);
16             if(k/i > 2) fac = min(fac, k/i);
17         }
18     }
19     if(fac > 2) printf("%d\n", fac-1);
20     else printf("0\n");
21     return 0;
22 }
```

8.2 斐波那契博弈

问题描述

有一堆个数为 n 的石子，游戏双方轮流取石子，满足：

1) 先手不能在第一次把所有的石子取完；

2) 之后每次可以取的石子数介于 1 到对手刚取的石子数的 2 倍之间（包含 1 和对手刚取的石子数的 2 倍）。

约定取走最后一个石子的人为赢家，求先手必败态。

博弈定理

这个游戏叫做 Fibonacci Nim，肯定和 Fibonacci 数列：f[n]：1,2,3,5,8,13,21,34,55,89,…有密切的关系。先手胜当且仅当 n 不是 Fibonacci 数。换句话说，先手必败态构成 Fibonacci 数列。

```
#include <iostream>
#include <cstdio>
using namespace std;
const int maxn = 46; // 第45个斐波那契数已经超过1e9
int f[maxn];
int t, n, flag;

void init() {
    f[1] = f[2] = 1;
    for(int i = 3; i < maxn; ++i) {
        f[i] = f[i-1]+f[i-2];
    }
}
int main() {
    init();
    scanf("%d", &t);
    while(t--) {
        scanf("%d", &n);
        flag = 1;
        for(int i = 1; i < maxn; ++i) {
            if(f[i] == n) {
                flag = 0;
                break;
            }
        }
        if(flag) printf("A\n");
        else printf("B\n");
    }
    return 0;
}
```

8.3 威佐夫博弈

问题描述

有两堆石子，每次每个人可以从任意一堆石子中取任意多的石子或者从两堆石子中取同样多的石子，不能取得人输，分析谁会获得胜利

博弈定理

假设两堆石子为 (x,y) ，其中 x<y，那么先手必败，当且仅当 $(y-x)*\frac{1+\sqrt{5}}{2}=x$ ，公式后者的数字实际就是 1.618，有些题目要求精度较高，我们可以用下述式子来表示这个值：(1.0+sqrt(5.0))/2.0

```
while(scanf("%d%d", &a, &b) != EOF) {
    if(a>b) swap(a,b);
    int tmpa = int((b-a) * (1.0+sqrt(5.0))/2.0);
    if(a == tmpa) printf("后手赢\n");
    else printf("先手赢\n");
}
```

例题变式

取石子游戏 HDU2177

题意

有两堆石子，数量任意，可以不同。游戏开始由两个人轮流取石子。游戏规定，每次有两种不同的取法，一是可以在任意的一堆中取走任意多的石子；二是可以在两堆中同时取走相同数量的石子。最后把石子全部取完者为胜者。如果轮到你先取，假设双方都采取最好的策略，问最后你是胜者还是败者。如果你胜，你第 1 次可以

怎样取子

思路

威佐夫博弈的变式，除了判断谁赢，还要判断第一步该怎么拿，直接分两种情况枚举，第一个分支，从两堆物品中同时取出相同数量的物品，第二个分支，只从一堆物品中取物品，用 map 记录已经考虑过的取法

```
using namespace std;
typedef long long ll;

int x,y;
map<int, int> mp;

int main() {
    while(1) {
        scanf("%d%d", &x, &y);
        if(x == 0 && y == 0) break;
        if(x>y) swap(x,y);
        double eqa = (1.0+sqrt(5.0))/2.0;
        int tmpx = int((y-x)*eqa);
        if(x == tmpx) {
            printf("0\n");
        } else {
            printf("1\n");
            for(int i = 1; i <= x; ++i) {
                int n = x-i;
                int m = y-i;
                if(n>m) swap(n,m);
                if(int((m-n)*eqa) == n) {
                    if(mp[n] == 0 && mp[m] == 0) {
                        printf("%d %d\n", n, m);
                        mp[n] = mp[m] = 1;
                    }
                }
            }
            for(int i = 1; i <= y; ++i) {
                int n = x;
                int m = y-i;
                if(n>m) swap(n,m);
                if(int((m-n)*eqa) == n) {
                    if(mp[n] == 0 && mp[m] == 0) {
                        printf("%d %d\n", n, m);
                        mp[n] = mp[m] = 1;
                    }
                }
            }
            // 只从小堆取也得考虑，情况不会完全重复(不考虑的话可用数据5 7或19 20
              去hack，5 7会输出3 5 3 5 4 7，19 20会少了12 20)
            for(int i = 1; i <= x; ++i) {
                int n = x-i;
                int m = y;
                if(n>m) swap(n,m);
                if(int((m-n)*eqa) == n) {
                    if(mp[n] == 0 && mp[m] == 0) {
                        printf("%d %d\n", n, m);
                        mp[n] = mp[m] = 1;
                    }
                }
            }
        }
    }
    return 0;
}
```

威佐夫游戏 V2 51Nod - 1185

题意

有 2 堆石子。A B 两个人轮流拿，A 先拿。每次可以从一堆中取任意个或从 2 堆中取相同数量的石子，但不可

不取。拿到最后 1 颗石子的人获胜。假设 A B 都非常聪明，拿石子的过程中不会出现失误。给出 2 堆石子的数量，问最后谁能赢得比赛

思路

威佐夫博弈模板，不过要注意这题石子的数据范围是 1e18，直接乘 $\frac{1+\sqrt{5}}{2}$ 的话会有精度问题，所以我们就减少精度问题，就将 0.6180339887498948482045868 34...0.6180339887498948482045868 34... 拆成整数放进数组里，拆成三部分就行了，然后通过乘法来减少精度的损失

```
/* 大整数乘法还是会出现精度问题，依然要控制精度 */
#include <iostream>
#include <cstdio>
#include <cmath>
#include <map>
using namespace std;
typedef long long ll;

int t;
ll a,b;
// 1.6180339887498948482045868 34，我们可以先不计算1，先算(b-a)*0.6180339...，然后再加上一个(b-a)即可
ll epa[3] = {618033988,749894848,204586834};
// 把long long数字拆成两个部分再来乘法
ll MOD = 1000000000;

int main() {
    scanf("%d", &t);
    while(t--) {
        scanf("%lld%lld", &a, &b);
        if(a>b) swap(a,b);
        ll cnt = b-a;
        ll tmpa = cnt/MOD, tmpb = cnt%MOD; // 把long long数字拆分成两部分
        // 模拟一个两位数乘三位数
        ll ans = tmpb*epa[2];
        ans = tmpa*epa[2] + tmpb*epa[1] + ans/MOD;
        ans = tmpa*epa[1] + tmpb*epa[0] + ans/MOD;
        ans = tmpa*epa[0] + ans/MOD + cnt;
        if(ans == a) printf("B\n");
        else printf("A\n");
    }
    return 0;
}
```

8.4 对称博弈

问题描述

一般对称博弈用于排列成环的情况，比如把一堆硬币排成一圈，然后只能取连续的，问最后谁赢

博弈定理

说是定理其实是个很直观的东西，假设有 N 个硬币，每次可以取 K 个，如果 $N < K$ ，则先手必赢，如果 $K = 1$ ，则根据 N 的奇偶性判断，否则如果是 $n > k\&\&k! = 1$ 的话，无论你第一个人拿什么，怎么拿，后手的人完全可以通过拿走一定数量的硬币把剩下的分成两段相同的连续部分，然后一直模仿第一个人的拿法，这样后手就一定会取得胜利，因为最后一步是后手走的

Coin Game HDU3951

题意

两个玩家用一圈 n 个硬币开始游戏。他们依次从圈子中拿走硬币，每次他们可以拿 1　K 连续硬币。（想象一下，十个硬币编号从 1 到 10，K 等于 3，因为 1 和 10 是连续的，你可以拿走连续的 10,1,2，但是如果 2 被拿走，你不能拿 1，3 ，4，因为 1 和 3 不连续）拿最后一枚硬币的玩家赢得比赛

思路

断环然后对称取（与之有个类似的问题就是平面放圆盘，此处就不放例题，大概思路是把第一个放中间，然后其他的对称放）

```
#include <iostream>
```

```
#include <cstdio>
using namespace std;
int t, n, k, cas = 0;
int main() {
    scanf("%d", &t);
    while(t--) {
        scanf("%d%d", &n, &k);
        if(k == 1) {
            if(n&1) printf("Case %d: first\n", ++cas);
            else printf("Case %d: second\n", ++cas);
        } else if(n <= k) {
            printf("Case %d: first\n", ++cas);
        } else { // 断环对称取
            printf("Case %d: second\n", ++cas);
        }
    }
    return 0;
}
```

8.5 nim 博弈

问题描述

有任意堆物品，每堆物品的个数是任意的，双方轮流从中取物品，每一次只能从一堆物品中取部分或全部物品，最少取一件，取到最后一件物品的人获胜。

博弈定理

把每堆物品数全部异或起来，如果得到的值为 0，那么先手必败，否则先手必胜。

```
while(cin>>n) {
    temp=0;
    for(int i=0;i<n;i++) {
        cin>>ans;
        temp^=ans;
    }
    if(temp==0)  cout<<"后手必胜"<<endl
    else cout<<"先手必胜"<<endl;
}
```

例题变式

Being a Good Boy in Spring Festival HDU1850

题意

桌子上有 M 堆扑克牌；每堆牌的数量分别为 Ni(i=1…M)；两人轮流进行；每走一步可以任意选择一堆并取走其中的任意张牌；桌子上的扑克全部取光，则游戏结束；最后一次取牌的人为胜者。现在我们不想研究到底先手为胜还是为负，我只想问大家：——“先手的人如果想赢，第一步有几种选择呢？

思路

这题是 nim 博弈的一个简单变式，问的是如果先手能赢，则第一步该如何取，假如要从 a1 中取走的 h 个剩 a1 ‘个，使 $a1' \oplus a2 \oplus a3 \oplus ..... = 0$，则有 $a1' = a1 \oplus k = a2 \oplus a3 \oplus a4 \oplus a5 \oplus .....$;(因为 $a \oplus a = 0; 0 \oplus a = a;$)，因此我们只需要判断 $a1 \oplus k$ 是否比原本的小，即 $ai \oplus k < ai$，如果是，则说明这一堆可以取。

```
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;
int main() {
    int a[110];
    int t;
    while(cin >> t&&t) {
        int k = 0;
        for(int i = 0; i < t; i++) {
            cin >> a[i];
            k = k^a[i];
        }
        int cnt = 0;
```

```
        for(int i = 0; i < t; i++)
            if((k^a[i]) < a[i])
                cnt++;
        cout << cnt << endl;
    }
    return 0;
}
```

取 m 堆石子游戏 HDU2176

题意

m 堆石子, 两人轮流取. 只能在 1 堆中取. 取完者胜. 先取者负输出 No. 先取者胜输出 Yes, 然后输出怎样取子. 如果从有 a 个石子的堆中取若干个后剩下 b 个后会胜就输出 a b

思路

nim 博弈变式，如果 $(k \oplus a[i]) < a[i]$ ，说明能取，取后的数量是 $k \oplus a[i]$ ，要注意一个坑点，异或计算的优先级比四则运算要低

```
#include <iostream>
#include <cstdio>
using namespace std;
const int maxn = 2e5+5;

int m, a[maxn];

int main() {
    while(cin >> m && m) {
        int k = 0;
        for(int i = 1; i <= m; ++i) {
            cin>>a[i];
            k ^= a[i];
        }
        if(k == 0) cout << "NO\n";
        else {
            cout << "YES\n";
            for(int i = 1; i <= m; ++i) {
                if((a[i]^k) < a[i]) {
                cout << a[i] << " " << (a[i]^k) << "\n";
                }
            }
        }
    }
    return 0;
}
```

8.6 SG 函数

```
#define MAX 1005
/* 计算从1—n范围内的SG值。
Array(存储可以走的步数, Array[0]表示可以有多少种走法)
Array[]需要从小到大排序 */
/*HDU1847博弈SG函数
1.可选步数为1—m的连续整数, 直接取模即可, SG(x) = x % (m+1);
2.可选步数为任意步, SG(x) = x;
3.可选步数为一系列不连续的数, 用GetSG(计算) */

int SG[MAX], hash[MAX];
void GetSG(int Array[], int n = MAX—1) {
    memset(SG, 0, sizeof(SG));
    for(int i = 0; i <= n; ++i) {
        memset(hash, 0, sizeof(hash));
        for(int j = 1; j <= Array[0]; ++j) {
            if(i < Array[j]) break;
            hash[SG[i — Array[j]]] = 1;
        }
        for(int j = 0; j <= n; ++j)
```

```
20              if(!hash[j]) { SG[i] = j; break; }
21      }
22  }
```

9 其它

9.1 数据类型范围

数据类型	范围
char	-128 - 127
int	2147483648 - 2147483647(十位)
long long	-9223372036854775808 - 9223372036854775807(大约十九位)
double	1.7 * 10308

9.2 头文件

```
1  //#include<stdio.h>
2  //#include<iostream>
3  //#include<queue>
4  //#include<algorithm>
5  //#include<cstring>
6  //#include<vector>
7  //#include<cmath>
8  //#include<string>
9  //#include<map>
10 //#include<set>
11 #include<bits/stdc++.h>
12 using namespace std;
13 typedef long long ll;
14 #define inf 0x3f3f3f3f
15 typedef pair<int, int> P;
16 const int maxn = 5e4+5;
17 const ll mod = 1e9+7;
```

9.3 Vim 配置

```
1  // open ~/.vimrc
2  syntax on
3  set nu ts=4 sw=4 mouse=a cin
4  colo desert
5
6  inoremap ' ''<ESC>i
7  inoremap " ""<ESC>i
8  inoremap ( ()<ESC>i
9  inoremap [ []<ESC>i
10 inoremap { {<CR>}<ESC>O
11 map <C-A> ggVG"+y
12 map <F5> :call CR()<CR>
13 func! CR()
14 exec "w"
15 exec "!g++ -O2 -g -std=c++11 -Wall % -o %<"
16 exec "! ./%<"
17 endfunc
```

9.4 输入挂

9.4.1 关闭同步

```
1  #define endl '\n'
2
```

```
ios::sync_whit_stdio(0);
cin.tie(0);
```

9.4.2 IO

```
#include<cstdio>

inline void read(int &x)    //看情况可去掉负数部分
{
    int t = 1;
    char ch = getchar();
    while(ch < '0' || ch > '9') { if(ch == '-') t = -1; ch = getchar();}
    x = 0;
    while(ch >= '0' && ch <= '9'){ x=x*10+ch-'0'; ch = getchar(); }
    x*=t;
}

void print(int i){
    if(i<10){
        putchar('0'+i);
        return ;
    }
    print(i/10);
    putchar('0'+i%10);
}
```

9.5 C++ 大数

9.5.1 大数加法

```
string add(string a,string b)
{   //两数相加
    string res="";
    int i=1;
    string first="0";
    while(true) {
        int tai=a.size()-i;
        int tbi=b.size()-i;
        if(tai<0 && tbi<0)
            break;          //从两数最右边开始模拟加法运算直到两数都遍历完
        int ta,tb;
        if(tai<0)
            ta=0;   //如果没数则至为0
        else
            ta=a[tai]-'0';
        if(tbi<0)
            tb=0;   //如果没数则至为0
        else
            tb=b[tbi]-'0';
        int temp=ta+tb+first[0]-'0';    //相加 first保存上一个的进位信息
        first[0]=temp%10+'0';   //当前位是对10取余
        res=first+res;
        first[0]=temp/10+'0';   //进位是除10
        i++;
    }
    if(first!="0")
        res=first+res;  //如果进位还有则添加
    if(res[0]=='0' && res.size()>1) //去除前导0
        res.erase(res.begin());
    return res;
}
```

9.5.2 大数乘法

```
// 还需要配合大数加法
string mul(string a,string b) {
    if(b.size()==1) {   //如果b只有一位则使其分别相乘
        string res="";
        string first="0";
        int mb=b[0]-'0';
        for(int i=a.size()-1; i>=0; i--) {  //从最后开始依次计算
            int temp=(a[i]-'0') * mb + (first[0]-'0');
            first[0]=temp%10+'0';   //当前位
            res=first+res;
            first[0]=temp/10+'0';   //进位
        }
        if(first!="0") {
            res=first+res;  //处理进位
        }
        if(res[0]=='0' && res.size()>1)
            res.erase(res.begin()); //除去前导0
        return res;
        }
    //否则则把b拆分为一位
    string res="0";
    string zero="";
    for(int i=b.size()-1; i>=0; i--) {  //从b的最后一位开始
        string temp=mul(a,b.substr(i,1));   //计算当前为与a相乘
        res=add(res,temp+zero); //在其后添加适当的0再与结果相加
        zero=zero+"0";
    }
    return res;
}
```

9.5.3 整数转 string

```
string inttostring(int m) { //将整数转为string
    string res="";
    string temp="0";
    while(m) {
        temp[0]=m%10+'0';
        res=temp+res;
        m/=10;
    }
    if(res=="")
        res="0";
    return res;
}
```

9.6 Java

```
valueOf(parament); 将参数转换为制定的类型

比如 int a=3;

BigInteger b=BigInteger.valueOf(a);

则b=3;

String s=”12345”;

BigInteger c=BigInteger.valueOf(s);

则c=12345;

```

```
// 常用函数
1.赋值:
BigInteger a=new BigInteger("1");
BigInteger b=BigInteger.valueOf(1);

2.运算:
  add(); 大整数相加
BigInteger a=new BigInteger(“23”);
BigInteger b=new BigInteger(“34”);
a.add(b);

 subtract(); 相减
 multiply(); 相乘
 divide(); 相除取整
 remainder(); 取余
 pow(); a.pow(b)=a^b
 gcd(); 最大公约数
 abs(); 绝对值
 negate(); 取反数
 mod(); a.mod(b)=a%b=a.remainder(b);

3.BigInteger构造函数:
一般用到以下两种:
BigInteger(String val);
将指定字符串转换为十进制表示形式;
BigInteger(String val,int radix);
将指定基数的 BigInteger 的字符串表示形式转换为 BigInteger

4.基本常量:
A=BigInteger.ONE 1
B=BigInteger.TEN 10
C=BigInteger.ZERO 0

5.n.compareTo(BigInteger.ZERO)==0  //相当于n==0

6.if(a[i].compareTo(n)>=0 && a[i].compareTo(m)<=0)   // a[i]>=n && a[i]<=m

// 模板
import java.math.BigInteger;
import java.math.BigDecimal;
import java.util.Scanner;
import java.util.*;
import java.io.*;
public class Main {
    public static void main(String [] args){
        Scanner cin = new Scanner(System.in);
        BigInteger a, b;
        while(cin.hasNext())//相当于c语言中的scanf("%d", &n) != EOF
        {
            a = cin.nextBigInteger();
            b = cin.nextBigInteger();
            System.out.println(a.add(b));//大整数加法
            System.out.println(a.subtract(b));//大整数减法
            System.out.println(a.multiply(b));//大整数乘法
            System.out.println(a.divide(b));//大整数除法, 取整
            System.out.println(a.remainder(b));//大整数取模
            System.out.println(a.abs());//对大整数a取绝对值
            int x = 0;
            System.out.println(a.pow(x));//大整数a的x次幂
            int y = 8;
            System.out.println(a.toString(y));//返回大整数a的p进制用字符串表现的形
              式
            System.out.println(a.toString());//返回大整数a的十进制用字符串表现的形
              式
            //大整数之间的比较
            if( a.compareTo(b) == 0 ) System.out.println("a == b"); //大整数a==b
```

```
79              else if( a.compareTo(b) > 0 ) System.out.println("a > b"); //大整数a>b
80              else if( a.compareTo(b) < 0 ) System.out.println("a < b"); //大整数a<b
                  )
81
82              BigDecimal c, d;
83              c = cin.nextBigDecimal();
84              d = cin.nextBigDecimal();
85              System.out.println(c.add(d));//浮点数相加
86              System.out.println(c.subtract(d));//浮点数相减
87              System.out.println(c.multiply(d));//浮点数相乘
88          }
89      }
90  }
91
92  // 输入方式
93  int a = cin.nextInt();
94  String s = cin.nextLine();
95  double b = cin.nextDouble();
```

9.7 Python

9.7.1 正常输入输出

```
1  a = int(input())
2  #输入一个int型数
3  arr = input("");
4  #输入一个一维数组，每个数之间使空格隔开
5  num = [int(n) for n in arr.split()]
6  #将输入每个数以空格键隔开做成数组，忽略前导空格和多个空格
7
8  print(num, end='') #end默认是'\n'
9  # 正常输出
```

9.7.2 多组输入直到文件尾结束

```
1  while True:
2      try:
3          a, b = map(int, raw_input().strip().split())
4          print(a+b)
5      except EOFError:
6          break
```

9.7.3 知道数据组数

```
1  tcase = int(raw_input().strip())
2  for case in range(tcase):
3      a, b = map(int, raw_input().strip().split())
4      print(a+b)
```

9.7.4 多组输入

遇到 0 0 结束

```
1  while True:
2      a, b = map(int, raw_input().strip().split())
3      if a == 0 and b == 0:
4          break
5      print(a+b)
```

输入有多组，并且题目告诉你每组输入遇见什么结束，与第三种不同之处在于，每组输入都有相应的细化

```
tcase = int(raw_input().strip())
for case in range(tcase):
    a, b = map(int, raw_input().strip().split())
    if a == 0 and b == 0:
        break
    print(a+b)
```

这次的输入实现输入一个整数，告诉我们有多少行，在输入每一行。对于每一行的输入，有划分为第一个数和其他的数，第一个数代表那一组数据一共有多少输入

```
tcase = int(raw_input().strip())
for case in range(tcase):
    data = map(int, raw_input().strip().split())
    n, array = data[0], data[1:]

    sum = 0
    for i in range(n):
        sum += array[i]
    print(sum)
```

有多种输入数据，对于每组输入数据的第一个数代表该组数据接下来要输入数据量

```
while True:
    try:
        data = map(int, raw_input().strip().split())
        n, array = data[0], data[1:]

        sum = 0
        for i in range(n):
            sum += array[i]
        print(sum)
    except EOFError:
        raise
```

9.7.5 进制转换

```
def main():
    code = ['0','1','2','3','4','5','6','7','8','9',
'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U
  ','V','W','X','Y','Z',
'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u
  ','v','w','x','y','z']
    x,y,z = input().split()
    x = int(x)
    y = int(y)
    # 转为10进制
    num = 0
    for ch in z:
        if ord('0') <= ord(ch) <= ord('9'):  # 0-9
            num = num * x + ord(ch) - ord('0')
        elif ord('A') <= ord(ch) <= ord('Z'):  # A-Z
            num = num * x + ord(ch) - ord('A') + 10
        else:  # a-z
            num = num * x + ord(ch) - ord('a') + 36

    if num == 0:
        print('0')
        return
    res = ""
    while num > 0:
        tmp = num % y
        res = res + code[tmp]
        num = num // y
```

```
    res = res[::-1]
    print(res)

main()
```

9.8 二分答案

```
#include<bits/stdc++.h>
using namespace std;
int n,m,A;
int  l,r,mid,a[100005];
int  total ,cnt;
bool judge(int mid)
{
    int num = 0;
    int l = a[0];
    for(int i =1;i<n;i++)
    {
        if(a[i]-l<mid) num++;
        else  l = a[i];
        if(num>A) return false;
    }
    return true;
}

int main()
{
    while(cin>>n>>m)
    {
        A = n-m;//最大剩余牛栏数
        l =r = 0;
        int  ans;
        for(int i = 0;i<n;i++)
        {
            cin>>a[i];
            r = max(r,a[i]);
        }
        sort(a,a+n);
        while(l<=r)
        {
            mid = (l+r)/2;
            total = cnt = 0;
            if(judge(mid))
            {
                l = mid+1;
                ans = mid;
            }//若此答案可行，从mid+1 ~ r区间继续查找（更大答案），即修改左界l=mid
              +1
            else
            {
                r = mid -1;  //反之，若此答案不可行，从l ~ mid-1区间查找（合理答
                  案），即修改左界l=mid -1
                //ans = mid;
            }
        }
        cout<<ans<<endl;
    }
    return 0;
}
```

9.9 三分

$$\frac{n*k+m}{1-(1-p^k)}$$

```
#include <bits/stdc++.h>
using namespace std;
#define eps 1e-9
typedef long long ll;

int t;
double n,m,p;

double qpow(double a, ll k) {
    double ans = 1;
    double tmp = a;
    while(k) {
        if(k&1) ans *= tmp;
        tmp *= tmp;
        k >>= 1;
    }
    return ans;
}

double f(double n, double m, double p, ll k) {
    return (n*k+m) / (1.0-qpow(1-p,k));
}

int main() {
    scanf("%d", &t);
    while(t--) {
        scanf("%lf%lf%lf", &n, &m, &p);
        p /= 10000.0;
        ll l = 1, r = 1e9;
        while(r - l > 2) {
            ll k1 = l+(r-l)/3;
            ll k2 = r-(r-l)/3;
            double v1 = f(n,m,p,k1);
            double v2 = f(n,m,p,k2);
            if(v1+eps > v2) l = k1;
            else r = k2;
            //  cout << l << " " << r << endl;
        }
        double ans = min(f(n,m,p,l-1),min(f(n,m,p,l),min(f(n,m,p,l+1),min(f(n,m,p,
          r),f(n,m,p,r+1)))));
        printf("%.10f\n", ans);
    }
    return 0;
}
```