THE BLACK AKER

2019.10.11

Powered by Xiaojun Dong, Yichao Li, Yubing Luo

目录

Data_Structure

cdq

```
/************三维偏序问题**************/
const int maxn = 1e5+100;
int t,n,bit[maxn],ans[maxn];
struct que{
    int a,b,c,idx;
} q[maxn], tmp[maxn];
int cmp(que a, que b){
    if(a.a != b.a) return a.a < b.a;
    else if(a.b != b.b) return a.b < b.b;
    else return a.c < b.c;
}
int cmp1(que a, que b){
    if(a.b != b.b) return a.b < b.b;
    else return a.c < b.c;
}
int inline lowbit(int x){
    return x&-x;
}
void add(int x, int k){
    for(int i = x; i < maxn; i += lowbit(i)){
        bit[i] += k;
    }
}
void reset(int p){
    while(p<=100000){
        bit[p]=0;
        p+=lowbit(p);
    }
}
int sum(int x){
    int res = 0;
    for(int i = x; i > 0; i -= lowbit(i)){
        res += bit[i];
    }
    return res;
}
void cdq(int l, int r){
    /*l,r 为第一维,分治中对第二维进行排序，第三维 bit 维护*/
    if(l == r)  return;
    int m = (l+r)/2;
    cdq(l,m);
    cdq(m+1,r);
    for(int i = l; i <= r; i++) tmp[i] = q[i];
    sort(tmp+l,tmp+m+1,cmp1);
    sort(tmp+m+1,tmp+r+1,cmp1);
    int j = l;
    /*处理[l,m]之中的修改对[m+1,r]中的询问的影响*/
    for(int i=m+1;i<=r;i++){
        while(j<=m&&tmp[j].b<=tmp[i].b){
            add(tmp[j++].c,1);
        }
        ans[tmp[i].idx]+=sum(tmp[i].c);
    }
    /*reset bit*/
    for(int i = l; i <= m; i++){
        reset(tmp[i].c);
    }
}
int main(){
    sc(t);
    while(t--){
        sc(n);
        for(int i = 0; i < n; i++){
            scanf("%d%d%d", &q[i].a, &q[i].b, &q[i].c);
            q[i].idx = i;
        }
        sort(q, q+n,cmp);
        memset(ans, 0, sizeof(ans));
        cdq(0, n-1);
        for(int i = n-2; i >= 0; i--){
            if(q[i].a == q[i+1].a && q[i].b == q[i+1].b
&& q[i].c == q[i+1].c)
                ans[q[i].idx] = ans[q[i+1].idx];
        }
        for(int i = 0; i < n; i++)
            printf("%d\n", ans[i]);
    }
```

```
    return 0;
}
```

LeftTree

```
/*******左偏树模板,pop,push 等功能自行实现**********/
struct Heap{
    int l, r, v, d;
} heap[maxn];
int Merge(int x, int y){
    if(!x || !y) return x+y;
    ///大根堆
    ///if(heap[x].v < heap[y].v) swap(x,y);
    ///小根堆
    ///if(headp[x].v > heap[y].v) swap(x,y);
    heap[x].r = Merge(heap[x].r, y);
    ///保持左端深度更大
    if(heap[heap[x].l].d < heap[heap[x].r].d)
        swap(heap[x].l, heap[x].r);
    heap[x].d = heap[heap[x].r].d + 1;
    return x;
}
```

Chain_div

```
///暂时只更新(点权修改,链上点权值和/最大值询问),例题 bzoj1036
const int maxn = 1e5+10;
///树链剖分模板
/*********************************************/
int val_p[maxn], val_e[maxn*2], ///点权与边权
        to[maxn*2], nxt[maxn*2],head[maxn*2], tot;
void tree_init(){
    tot = 0;
    memset(head, 0, sizeof(head));
}
void add_edg(int u, int v, int w = 0){
    tot++; val_e[tot] = w; to[tot] = v;
    nxt[tot] = head[u]; head[u] = tot;
}
void read_edg(int op = 1){///op=1 --> 双向边; op=0 --> 单向边
    int u, v, w;
    scanf("%d%d%d", &u, &v, &w);
    add_edg(u, v, w);
    if(op) add_edg(v, u, w);
}
int tid[maxn],    ///dfs 序/tid[i]-->节点 i 在线段树中的位置
    rk[maxn],     ///rk[i]-->dfs 序第 i 个在树中的节点编号
    top[maxn],    ///所在链顶点
    sz[maxn],     ///子树 size
    son[maxn],    ///重儿子
    dep[maxn],    ///深度
    fa[maxn],     ///父节点
    idx;          ///dfs 序计数器
void dfs1(int u, int f){
    ///更新 dep[u], fa[u], sz[u], son[u]
    dep[u] = dep[f]+1;
    fa[u] = f;
    sz[u] = 1;
    for(int i = head[u]; i; i = nxt[i]){
        int v = to[i];
        if(v == f) continue;
        dfs1(v, u);
        sz[u] += sz[v];
        if(son[u] < 0 || sz[v] > sz[son[u]])
            son[u] = v;
    }
}
void dfs2(int u, int tp){
    ///更新 top[u], tid[u], rk[u]
    top[u] = tp;
    tid[u] = idx;
    rk[idx++] = u;
    if(son[u] >= 0) dfs2(son[u], tp);
    for(int i = head[u]; i; i = nxt[i]){
        int v = to[i];
        if(v == fa[u] || v == son[u]) continue;
        dfs2(v, v);
    }
}
void chain_div(int _n, int rt = 1){///rt 取决于题目是否为有根树
```

```
    memset(son, -1, sizeof(int)*(_n+5));
    dep[rt] = idx = 0;
    dfs1(rt, rt);
    dfs2(rt, rt);
}
/************************************************/
///线段树和调用部分仅作参考，赛场上自行发挥
int mx[maxn*3], val[maxn], n, sum[maxn*3];
void pick_up(int rt){
    mx[rt] = max(mx[lson], mx[rson]);
    sum[rt] = sum[lson] + sum[rson];
}
void build(int rt = 0, int l = 0, int r = n-1){
    if(l == r){
        mx[rt] = sum[rt] = val[l];
        return;
    }
    build(lson, l, mid);
    build(rson, mid+1, r);
    pick_up(rt);
}
void update(int x, int v, int rt = 0, int l = 0, int r =
n-1){
    if(l > x || r < x) return;
    if(l == x && r == x){
        mx[rt] = sum[rt] = v;
        return;
    }
    update(x, v, lson, l, mid);
    update(x, v, rson, mid+1, r);
    pick_up(rt);
}
inline pii add(pii a, pii b){
    return pii(max(a.fi,b.fi), a.se+b.se);
}
pii query(int x, int y, int rt = 0, int l = 0, int r =
n-1){
    if(l > y || r < x) return mk(-INF, 0);
    if(l >= x && r <= y){
        return mk(mx[rt], sum[rt]);
    }
    pii a = query(x, y, lson, l, mid),
        b = query(x, y, rson, mid+1, r);
    return add(a,b);
}
pii solve(int u, int v){
    pii ans = mk(-INF, 0);
    while(top[u] != top[v]){
        if(dep[top[u]] < dep[top[v]]) swap(u,v);
        ans = add(ans, query(tid[top[u]], tid[u]));
        u = fa[top[u]];
    }
    if(dep[u] > dep[v]) swap(u, v);
    ans = add(ans, query(tid[u], tid[v]));
    return ans;
}
int main(){
    tree_init();
    sc(n);
    for(int i = 1; i < n; i++){
        int u, v; sc(u); sc(v);
        add_edg(u,v);
        add_edg(v,u);
    }
    for(int i = 1; i <= n; i++)
        sc(val_p[i]);
    chain_div(n);
    for(int i = 1; i <= n; i++)
        val[tid[i]] = val_p[i];
    build();
    int q; sc(q);
    while(q--){
        char s[5];
        int x,y;
        scanf("%s%d%d", s, &x, &y);
        if(s[1] == 'M'){
            printf("%d\n", solve(x,y).fi);
        }else if(s[1] == 'H'){
            update(tid[x], y);
        }else{
            printf("%d\n", solve(x,y).se);
        }
    }
    return 0;
```

```
}
```

LinkCutTree(Splay_Tree_included)

```
#define rev(x) (swap(l[x],r[x]),b[x]^=1)
#define judge(x) (l[f[x]]==x||r[f[x]]==x)
#define root(x) (access(x),modify(x),rev(x))
//LCA 换根

int
l[maxn],r[maxn],f[maxn],b[maxn],mx[maxn],v[maxn];

bool find(int x,int y)     //判断是否在同一棵树
{
    while(f[x])x=f[x];
    while(f[y])y=f[y];
    return x==y;
}

void update(int t)         //更新 splay 上维护的信息
{
    mx[t]=max(mx[l[t]],mx[r[t]]);
    mx[t]=max(mx[t],v[t]);
}

void rotate(int x,int l[],int r[]) //splay 旋转
{
    int t=f[x];
    f[l[t]=r[x]]=t;f[x]=f[t];
    if(judge(t))l[f[t]]==t?l[f[t]]=x:r[f[t]]=x;
    f[r[x]=t]=x;update(r[x]);
}

void deal(int x)             //splay 标记下传
{
    if(judge(x))deal(f[x]);
    if(b[x])b[x]^=1,rev(l[x]),rev(r[x]);
}

void modify(int x)               //splay 换根
{
    deal(x);
    while(judge(x))if(l[f[x]]==x)
    {
        if(judge(f[x])&&l[f[f[x]]]==f[x])
            rotate(f[x],l,r);
        rotate(x,l,r);
    }
    else
    {
        if(judge(f[x])&&r[f[f[x]]]==f[x])
            rotate(f[x],r,l);
        rotate(x,r,l);
    }
    update(x);
}

void access(int x)                //路径变重边
{
    for(int y=0;x;y=x,x=f[x])
        modify(x),l[x]=y,update(x);
}

void link(int x,int y)             //加边
{
    root(x);
    f[x]=y;
}

void cut(int x,int y)             //删边
{
    root(x);access(y);
    modify(y);
    f[x]=r[y]=0;
}
```

```
int query(int z,int y)              //查询两点之间权最大的
点
{
    if(!find(z,y))return 2e9;
    root(z);
    access(y);modify(y);
    while(v[y]!=mx[y])
        y=(mx[l[y]]==mx[y]?l[y]:r[y]);
    return y;
}
```

Mergingtree

```
#include <cstdio>
#include <algorithm>
using namespace std;
const int maxn = 1e5+10;
const int maxd = 20;
typedef struct
{
    int num[maxn];
    int cnt[maxn];
}partition_tree;
partition_tree tree[maxd];
int sorted[maxn];
void build(int d, int l, int r)
{
    if(l == r)return;
    int m = (l+r)>>1;
    int need = m-l+1;
    for(int i = l; i <= r; i++)if(tree[d].num[i] <
sorted[m])need--;
    int p = l, q = m+1;
    for(int i = l, cnt = 0; i <= r; i++){
        int num = tree[d].num[i];
        if(num < sorted[m] || (num == sorted[m] &&
need)){
            if(num == sorted[m])need--;
            cnt++;
            tree[d+1].num[p++] = num;
        }
        else tree[d+1].num[q++] = num;
        tree[d].cnt[i] = cnt;
    }
    build(d+1, l, m);
    build(d+1, m+1, r);
}
int query(int d, int l, int r, int ql, int qr, int k)
{
    if(l == r)return tree[d].num[l];
    int ly;
    if(l == ql)ly = 0;
    else ly = tree[d].cnt[ql-1];
    int tot = tree[d].cnt[qr]-ly;
    int newl, newr, mid = (l+r) / 2;
    if(tot >= k){
        newl = l+ly;
        newr = newl+tot-1;
        return query(d+1, l, mid, newl, newr, k);
    }
    else{
        newl = mid+1+(ql-l-ly);
        newr = newl+((qr-ql+1)-tot)-1;
        return query(d+1, mid+1, r,  newl, newr, k-tot);
    }
}
void solve()
{
    int n, m;
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i++){
        scanf("%d", &tree[0].num[i]);
        sorted[i] = tree[0].num[i];
    }
    sort(sorted+1, sorted+n+1);
    build(0, 1, n);
    for(int i = 0; i < m; i++){
        int l, r, k;
        scanf("%d%d%d", &l, &r, &k);
        int ans = query(0, 1, n, l, r, k);
        printf("%d\n", ans);
    }
}
```

Partitiontree

```
#include <cstdio>
#include <algorithm>
using namespace std;
const int maxn = 1e5+10;
const int maxd = 20;
typedef struct
{
    int num[maxn];
    int cnt[maxn];
}partition_tree;
partition_tree tree[maxd];
int sorted[maxn];
void build(int d, int l, int r)
{
    if(l == r)return;
    int m = (l+r)>>1;
    int need = m-l+1;
    for(int i = l; i <= r; i++)if(tree[d].num[i] <
sorted[m])need--;
    int p = l, q = m+1;
    for(int i = l, cnt = 0; i <= r; i++){
        int num = tree[d].num[i];
        if(num < sorted[m] || (num == sorted[m] &&
need)){
            if(num == sorted[m])need--;
            cnt++;
            tree[d+1].num[p++] = num;
        }
        else tree[d+1].num[q++] = num;
        tree[d].cnt[i] = cnt;
    }
    build(d+1, l, m);
    build(d+1, m+1, r);
}
int query(int d, int l, int r, int ql, int qr, int k)
{
    if(l == r)return tree[d].num[l];
    int ly;
    if(l == ql)ly = 0;
    else ly = tree[d].cnt[ql-1];
    int tot = tree[d].cnt[qr]-ly;
    int newl, newr, mid = (l+r) / 2;
    if(tot >= k){
        newl = l+ly;
        newr = newl+tot-1;
        return query(d+1, l, mid, newl, newr, k);
    }
    else{
        newl = mid+1+(ql-l-ly);
        newr = newl+((qr-ql+1)-tot)-1;
        return query(d+1, mid+1, r,  newl, newr, k-tot);
    }
}
void solve()
{
    int n, m;
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i++){
        scanf("%d", &tree[0].num[i]);
        sorted[i] = tree[0].num[i];
    }
    sort(sorted+1, sorted+n+1);
    build(0, 1, n);
    for(int i = 0; i < m; i++){
        int l, r, k;
        scanf("%d%d%d", &l, &r, &k);
        int ans = query(0, 1, n, l, r, k);
        printf("%d\n", ans);
    }
}
```

PersistentSegmentTree

```
#include <cstdio>
using namespace std;
const int maxn = 1e5+10;
const int logq = 20;
int root[maxn], ls[maxn*logq], rs[maxn*logq],
sum[maxn*logq];
int n, cnt;
void build(int &rt, int l, int r)
{
```

```
    rt = cnt++;
    sum[rt] = 0;
    if(l == r-1)return;
    int m = (l+r)/2;
    build(ls[rt], l, m); build(rs[rt], m, r);
}

void update(int &rt, int l, int r, int last, int val)
{
    rt = ++cnt;
    ls[rt] = ls[last]; rs[rt] = rs[last]; sum[rt] =
sum[last]+1;
    if(l == r-1)return;
    int m = (l+r)/2;
    if(val < m)update(ls[rt], l, m, ls[last], val);
    else update(rs[rt], m, r, rs[last], val);
}

int query(int ss, int tt, int l, int r, int k)//查询区间第
k 大
{
    if(l == r-1)return l;
    int m = (l+r)/2;
    int cnt = sum[ls[tt]] - sum[ls[ss]];
    if(k <= cnt)return query(ls[ss], ls[tt], l, m, k);
    else return query(rs[ss], rs[tt], m, r, k-cnt);
}
```

undo_dsu

```
/***四种操作:1.连 x,y; 2.删掉边 x,y; 3.询问 x 所在连通块的 size;
4.询问 x,y 是否连通***/
/***大致思路:记录每个连边的持续时间
****************************************/
/************solve(l,r)碰到持续时间和 l,r 一样的操作就添加,回溯
时撤销************/
#include <bits/stdc++.h>
using namespace std;
const int N = 2e5+5;
typedef pair<int,int> pii;
class jobs{
    public:
    int type;
    int x,y,l,r;
    jobs() {}
    jobs(int a,int b,int c,int d,int e) :
type(a),x(b),y(c),l(d),r(e) {}
};
int n,m,x,y,op,ans[N],siz[N],f[N];
unordered_map <long long,int> pre;
vector<jobs> job;
int find(int x,vector<pii> &mem){
    if(x == f[x]) return x;
    mem.push_back(make_pair(x,f[x]));
    f[x] = find(f[x],mem);
    return f[x];
}
void deal(int l,int r,vector<jobs> &q){
    vector<pii> mem,mem2;
    for(jobs &p : q)
     if(p.l == l && p.r == r && p.type == 1){
        int x = p.x,y = p.y;
        if(find(x,mem) != find(y,mem)){
            mem.push_back(make_pair(f[x],f[f[x]]));
            mem2.push_back(make_pair(f[y],siz[f[y]]));
            siz[f[y]] += siz[f[x]];
            f[f[x]] = f[y];
        }
     }
    for(jobs &p : q)
     if(p.l == l && p.r == r && p.type != 1){
        int x = p.x,y = p.y;
        if(p.type == 2) ans[l] = siz[find(x,mem)];
        else ans[l] = N + (find(x,mem) == find(y,mem));
     }
    if(l != r){
        vector<jobs> q1,q2;
        int mid = (l+r)>>1;
        for(jobs &p : q)
         if(p.r <= mid) q1.push_back(p);
         else
          if(p.l <= mid){
              if(p.l == l && p.r == r) continue;
              q1.push_back(jobs(1,p.x,p.y,p.l,mid));
```

```
            q2.push_back(jobs(1,p.x,p.y,mid+1,p.r));
        }
        else q2.push_back(p);
        q.clear();
        deal(l,mid,q1);
        deal(mid+1,r,q2);
    }
    for(int i = mem.size()-1;i >= 0;i--) f[mem[i].first]
= mem[i].second;
    for(int i = mem2.size()-1;i >= 0;i--)
siz[mem2[i].first] = mem2[i].second;
}
int main(){
    scanf("%d%d",&n,&m);
    for(int i = 1;i <= m;i++){
        scanf("%d",&op);
        if(op == 1){
            scanf("%d%d",&x,&y);
            if(x < y) swap(x,y);
            pre[1ll*x*N+y] = i;
        }
        if(op == 2){
            scanf("%d%d",&x,&y);
            if(x < y) swap(x,y);
            job.push_back(jobs(1,x,y,pre[1ll*x*N+y],i-
1));
            pre.erase(pre.find(1ll*x*N+y));
        }
        if(op == 3){
            scanf("%d",&x);
            job.push_back(jobs(2,x,0,i,i));
        }
        if(op == 4){
            scanf("%d%d",&x,&y);
            job.push_back(jobs(3,x,y,i,i));
        }
    }
    for(pair<long long,int> v : pre)
        job.push_back(jobs(1,v.first/N,v.first %
N,v.second,m));
    for(int i = 1;i <= n;i++)
        f[i] = i,siz[i] = 1;
    deal(1,m,job);
    for(int i = 1;i <= m;i++){
        if(!ans[i]) continue;
        if(ans[i] < N) cout<<ans[i]<<endl;
        else{
            if(ans[i] == N) cout<<"No"<<endl;
            else cout<<"Yes,cap"<<endl;
        }
    }
}
```

Graph

KM

```
#include <cstring>
#include <cstdio>
#include <algorithm>
using namespace std;
const int INF = 0x3f3f3f3f;
const int maxn = 3e2+10;
int dx[maxn], dy[maxn], mat[maxn], pre[maxn], sl[maxn];
int G[maxn][maxn];
bool vis[maxn];
int n;
void bfs(int x)
{
    memset(vis, 0, sizeof(vis));
    memset(pre, 0, sizeof(pre));
    memset(sl, 0x3f, sizeof(sl));
    int nx = 0, ny = 0;
    mat[ny] = x;
    do{
        nx = mat[ny]; vis[ny] = 1;
        int d = INF, t = 0;
        for(int i = 1;i <= n; i++)
            if(!vis[i]){
                int cost = dx[nx]+dy[i]-G[nx][i];
                if(cost < sl[i]){
                    sl[i] = cost; pre[i] = ny;
                }
                if(sl[i] < d){
                    d = sl[i]; t = i;
```

```
                }
            }
        for(int i = 0; i <= n; i++){
            if(vis[i])dx[mat[i]] -= d, dy[i] += d;
            else sl[i] -= d;
        }
        ny = t;
    }while(mat[ny]);
    while(ny)mat[ny] = mat[pre[ny]], ny = pre[ny];
}
int KM() {
    memset(mat, 0, sizeof(mat));
    memset(dy, 0, sizeof(dy));
    for(int i = 1; i <= n; i++){
        dx[i] = -INF;
        for(int j = 1; j <= n; j++)dx[i] = max(dx[i],
G[i][j]);
    }
    for(int i = 1; i <= n; i++)bfs(i);
    int ret = 0;
    for(int i = 1; i <= n; i++)if(mat[i])ret +=
G[mat[i]][i];
    return ret;
}
```

CostFlow_dijkstra

```
#include <vector>
#include <cstring>
#include <queue>
using namespace std;
typedef pair<int, int> pii;
#define fi first
#define se second
#define pb push_back
#define mk make_pair
const int INF = 0x3f3f3f3f;
const int maxv = 1e4+10;
int n, m, ans;
struct edge
{
    int to, cap, rev, cost;
    edge(int to, int cap, int rev, int
cost):to(to),cap(cap),rev(rev),cost(cost){}
};
vector<edge> G[maxv];
bool vis[maxv];
int dist[maxv];
void add_edge(int from, int to, int cap, int cost)
{
    G[from].pb(edge(to, cap, (int)G[to].size(), cost));
    G[to].pb(edge(from, 0, (int)G[from].size()-1, -
cost));
}

bool dijkstra(int s, int t)
{
    memset(dist, 0x3f, sizeof(dist));
    dist[t] = 0;
    priority_queue<pii, vector<pii>, greater<pii> > pq;
    pq.push(mk(0, t));
    while(!pq.empty()){
        pii u = pq.top(); pq.pop();
        if(dist[u.se] < u.fi)continue;
        for(auto &e : G[u.se]){
            if(G[e.to][e.rev].cap && dist[e.to] >
dist[u.se]-e.cost){
                dist[e.to] = dist[u.se]-e.cost;
                pq.push(mk(dist[e.to], e.to));
            }
        }
    }
    return dist[s] < INF;
}
int dfs(int v, int t, int f)
{
    vis[v] = 1;
    if(v == t)return f;
    int ret = 0;
    for(auto &e : G[v]){
        if(!vis[e.to] && e.cap > 0 && dist[v]-e.cost ==
dist[e.to]){
            int d = dfs(e.to, t, min(e.cap, f-ret));
            if(d){
```

```
                ans += e.cost*d;
                e.cap -= d;
                G[e.to][e.rev].cap += d;
                ret += d;
                if(ret == f)break;
            }
        }
    }
    return ret;
}
int costflow(int s, int t)
{
    int flow = 0;
    while(dijkstra(s, t)){
        vis[t] = 1;
        while(vis[t]){
            memset(vis, 0, sizeof(vis));
            flow += dfs(s, t, INF);
        }
    }
    return flow;
}
```

Dijkstra_heap

```
#include <queue>
#include <cstring>
using namespace std;
typedef pair<int, int> pii;
#define mk make_pair
#define fi first
#define se second
const int maxn = 1e5+10;
vector<pii> G[maxn];
int cost[maxn];
void dijkstra(int s)
{
    memset(cost, 0x3f, sizeof(cost));
    cost[s] = 0;
    priority_queue<pii, vector<pii>, greater<pii> >pq;
    pq.push(mk(0, s));
    while(!pq.empty()){
        pii p = pq.top(); pq.pop();
        if(cost[p.se] < p.fi)continue;
        for(auto v : G[p.se]){
            if(cost[v.fi] > cost[p.se]+v.se){
                cost[v.fi] = cost[p.se]+v.se;
                pq.push(mk(cost[v.fi], v.fi));
            }
        }
    }
}
```

ISAP

```
#include<bits/stdc++.h>
using namespace std;
struct edge{int st,ed,fl,ne;}x[maxm];
queue<int>q;
int d[maxn],g[maxn],bj[maxn],now[maxn];
//ISAP
int dfs(int t,int pre){
    if(t==n)return pre;
    int mn=n-1,res=pre,k;
    for(int i=bj[t];i!=-1;i=x[i].ne){
        if(!x[i].fl)continue;
        if(d[t]==d[x[i].ed]+1){
            k=dfs(x[i].ed,min(res,x[i].fl));
            x[i].fl-=k;
            x[i^1].fl+=k;
            res-=k;
            if(d[1]>=n)return pre-res;
            if(!res)break;
        }
        mn=min(mn,d[x[i].ed]);
    }
    if(res==pre){
        if(!(--g[d[t]]))d[1]=n;
        g[d[t]=mn+1]++;
    }
    return pre-res;
}
int main(){
    ...
```

```
    g[0]=n;
        while(d[1]<n)ans+=dfs(1,inf);
    return 0;
}
```

Dinic

```
#include <cstdio>
#include <cstring>
#include <queue>
#include <vector>
using namespace std;
#define pb push_back
#define SZ(x) ((int)x.size())
const int INF = 0x3f3f3f3f;
const int maxn = 1e5+10;
struct edge
{
    int to, cap, rev;
    edge(){}
    edge(int to, int cap, int rev):to(to), cap(cap),
rev(rev){}
};
vector<edge> G[maxn];
int lev[maxn];
int n, m;
void init()
{
    for(int i = 0; i <= n; i++)G[i].clear();
}
void add_edge(int u, int v, int w)
{
    G[u].pb(edge(v, w, SZ(G[v])));
    G[v].pb(edge(u, w, SZ(G[u])-1));
}
bool bfs(int s, int t)
{
    memset(lev, -1, sizeof(lev));
    lev[s] = 0;
    queue<int> q;
    q.push(s);
    while(!q.empty()){
        int u = q.front(); q.pop();
        for(auto &e : G[u]){
            if(e.cap > 0 && lev[e.to] < 0){
                lev[e.to] = lev[u]+1;
                if(e.to == t)return 1;
                q.push(e.to);
            }
        }
    }
    return 0;
}
int dfs(int v, int t, int f)
{
    if(v == t)return f;
    int ret = 0;
    for(auto &e : G[v]){
        if(e.cap > 0 && lev[e.to] == lev[v]+1){
            int d = dfs(e.to, t, min(e.cap, f-ret));
            if(d > 0){
                e.cap -= d;
                G[e.to][e.rev].cap += d;
                ret += d;
                if(ret == f)break;
            }

        }
    }
    if(!ret)lev[v] = 0;
    return ret;
}
int max_flow(int s, int t)
{
    int flow = 0;
    while(bfs(s, t))flow += dfs(s, t, INF);
    return flow;
}
```

DominatorTree

```
#include <cstdio>
#include <cstring>
#include <vector>
using namespace std;
```

```cpp
typedef long long ll;
#define pb push_back
const int maxn = 1e6+10;
vector<int> G[maxn], G2[maxn], dom[maxn];
int dfn[maxn], id[maxn], fa[maxn];
int semi[maxn], best[maxn], par[maxn], idom[maxn];
ll ans[maxn];
int n, m, idx;
void init()
{
    for(int i = 0; i <= n; i++)G[i].clear(),
G2[i].clear(), dom[i].clear();
    memset(dfn, 0, sizeof(dfn));
    memset(ans, 0, sizeof(ans));
    idx = 0;
}
void dfs(int u)
{
    dfn[u] = ++idx;
    id[idx] = u;
    for(auto v : G[u]){
        if(!dfn[v]){
            dfs(v);
            fa[v] = u;
        }
    }
}
int find(int x)
{
    if(par[x] == x)return x;
    int y = find(par[x]);
    if(dfn[semi[best[par[x]]]] <
dfn[semi[best[x]]])best[x] = best[par[x]];
    return par[x] = y;
}
void tarjan()
{
    for(int i = idx; i >= 2; i--){
        int u = id[i];
        for(auto v : G2[u]){
            if(!dfn[v])continue;
            find(v);
            if(dfn[semi[best[v]]] < dfn[semi[u]])semi[u]
= semi[best[v]];
        }
        dom[semi[u]].pb(u);
        int x = par[u] = fa[u];
        for(auto v : dom[x]){
            find(v);
            if(dfn[semi[best[v]]] < dfn[fa[u]])idom[v] =
best[v];
            else idom[v] = fa[u];
        }
        dom[x].clear();
    }
    for(int i = 2; i <= idx; i++){
        int u = id[i];
        if(idom[u] != semi[u])idom[u] = idom[idom[u]];
        dom[idom[u]].pb(u);
    }
}
void solve(int s)
{
    for(int i = 1; i <= n; i++)par[i] = best[i] =
semi[i] = i;
    dfs(s);
    tarjan();
}
```

Euler

```cpp
#include <cstdio>
#include <vector>
#include <cstring>
using namespace std;
#define pb push_back
const int maxn = 1e5+10;
const int maxm = 1e6+10;
struct edge
{
    int to, nxt;
    edge(){}
    edge(int to, int nxt):to(to), nxt(nxt){}
}es[maxm];
int head[maxn], deg[maxn];
```

```
bool used[maxm], vis[maxn];
vector<int> seq;
int n, m, cnt;
void init()
{
    memset(deg, 0, sizeof(deg));
    memset(vis, 0, sizeof(vis));
    memset(used, 0, sizeof(used));
}
void add_edge(int u, int v)
{
    es[cnt] = edge(v, head[u]);
    head[u] = cnt++;
    es[cnt] = edge(u, head[v]);
    head[v] = cnt++;
}
void dfs(int u)
{
    vis[u] = 1;
    for(int i = head[u]; ~i; i = head[u]){
        if(used[i>>1])continue;
        used[i>>1] = 1;
        int v = es[i].to;
        dfs(v);
        seq.pb(i);
    }
}
```

FlowerTree

```
#include <queue>
#include <cstdio>
#include <vector>
#include <cstring>
using namespace std;
const int maxn = 5e2+10;
vector<int> G[maxn];
int fa[maxn], vis[maxn], mat[maxn], pre[maxn],
typ[maxn];
int n, m, tim;
queue<int> q;
void init()
{
    tim = 0;
    memset(vis, 0, sizeof(vis));
    memset(mat, 0, sizeof(mat));
    memset(pre, 0, sizeof(pre));
}
int lca(int x, int y)
{
    tim++;
    x = fa[x], y = fa[y];
    while(vis[x] != tim){
        if(x){
            vis[x] = tim;
            x = fa[pre[mat[x]]];
        }
        swap(x, y);
    }
    return x;
}
void blossom(int x, int y, int f)
{
    while(fa[x] != f){
        pre[x] = y;
        y = mat[x];
        if(typ[y] == 1){
            typ[y] = 0;
            q.push(y);
        }
        fa[x] = fa[y] = fa[f];
        x = pre[y];
    }
}
bool augument(int s)
{
    for(int i = 1; i <= n; i++)fa[i] = i;
    memset(typ, -1, sizeof(typ));
    while(!q.empty())q.pop();
    typ[s] = 0;
    q.push(s);
    while(!q.empty()) {
        int u = q.front(); q.pop();
        for(auto v : G[u]){
            if(typ[v] == -1){
```

```
                pre[v] = u;
                typ[v] = 1;
                if(!mat[v]){
                    for(int to = v, from = u; to; from =
pre[to]){
                        mat[to] = from;
                        swap(mat[from], to);
                    }
                    return 1;
                }
                typ[mat[v]] = 0;
                q.push(mat[v]);
            }
            else if(typ[v] == 0 && fa[u] != fa[v]){
                int f = lca(u, v);
                blossom(u, v, f);
                blossom(v, u, f);
            }
        }
    }
    return 0;
}
int solve()
{
    init();
    int ret = 0;
    for(int i = 1; i <= n; i++){
        if(!mat[i])ret += augument(i);
    }
    return ret;
}
```

Floyd

```
#include <algorithm>
using namespace std;
const int maxn = 5e2+10;
const int INF = 0x3f3f3f3f;
int G[maxn][maxn];//原始距离
int dis[maxn][maxn];//最短路
int road[maxn][maxn];
int path[maxn];//记录最小环路径(无法计算二元环)
int n, m, cnt;
void record(int s, int t)
{
    if(road[s][t]){
        record(s, road[s][t]);
        record(road[s][t], t);
    }
    else path[cnt++] = t;
}
int floyd()
{
    int ret = INF;
    for(int k = 1; k <= n; k++){
        for(int i = 1; i < k; i++){
            for(int j = i+1; j < k; j++){
                if(dis[i][j] != INF && G[i][k] != INF &&
G[k][j] != INF && ret > dis[i][j]+G[i][k]+G[k][j]){ //注
意溢出
                    ret = dis[i][j]+G[i][k]+G[k][j];
                    cnt = 0;
                    path[cnt++] = i;
                    record(i, j);
                    path[cnt++] = k;
                }
            }
        }
        for(int i = 1; i <= n; i++){
            for(int j = 1; j <= n; j++){
                if(dis[i][j] > dis[i][k]+dis[k][j]){
                    dis[i][j] = dis[i][k]+dis[k][j];
                    road[i][j] = k;
                }
                dis[i][j] = min(dis[i][j],
dis[i][k]+dis[k][j]);
            }
        }
    }
    if(ret == INF)return -1; //无环
    return ret;
}
```

HopcroftKarp

```cpp
#include <queue>
#include <cstring>
using namespace std;
const int INF = 0x3f3f3f3f;
const int maxn = 2e3+10;
vector<int> G[maxn];
int mtx[maxn], mty[maxn], dx[maxn], dy[maxn];
bool vis[maxn];
int dis, uN, vN;
bool bfs()
{
    dis = INF;
    memset(dx, -1, sizeof(dx));
    memset(dy, -1, sizeof(dy));
    queue<int> q;
    for(int i = 1; i <= uN; i++){
        if(!mtx[i])q.push(i), dx[i] = 0;
    }
    while(!q.empty()){
        int u = q.front(); q.pop();
        if(dx[u] > dis)break;
        for(auto v : G[u]){
            if(dy[v] == -1){
                dy[v] = dx[u]+1;
                if(!mty[v])dis = dy[v];
                else{
                    dx[mty[v]] = dy[v]+1;
                    q.push(mty[v]);
                }
            }
        }
    }
    return dis < INF;
}
bool dfs(int u)
{
    for(auto v : G[u]){
        if(!vis[v] && dy[v] == dx[u]+1){
            vis[v] = 1;
            if(dy[v] == dis && mty[v])continue;
            else if(!mty[v] || dfs(mty[v])){
                mty[v] = u, mtx[u] = v;
                return 1;
            }
        }
    }
    return 0;
}
int max_match()
{
    int ret = 0;
    while(bfs()){
        memset(vis, 0, sizeof(vis));
        for(int i = 1; i <= uN; i++){
            if(!mtx[i] && dfs(i))ret++;
        }
    }
    return ret;
}
```

Hungarian

```cpp
#include <vector>
#include <cstring>
using namespace std;
const int maxn = 5e2+10;
vector<int> G[maxn];
int mat[maxn];
bool vis[maxn];
int n;
bool dfs(int u){
    for(auto v : G[u]){
        if(!vis[v]){
            vis[v] = 1;
            if (mat[v] == 0 || dfs(mat[v])){
                mat[v] = u;
                return true;
            }
        }
    }
    return false;
}
```

```
int solve()
{
    memset(mat, 0, sizeof(mat));
    int ans = 0;
    for(int i = 1; i <= n; i++){
        memset(vis, 0, sizeof(vis));
        if(dfs(i))ans++;
    }
    return ans;
}
```

Johnson

```
#include <cstdio>
#include <vector>
#include <queue>
using namespace std;
typedef pair<int, int> pii;
#define pb push_back
#define mk make_pair
#define fi first
#define se second
const int INF = 0x3f3f3f3f;
const int maxn = 1e3+10;
vector<pii> G[maxn];
int dist[maxn][maxn], h[maxn];
int n;
bool bellmanFord()
{
    for(int i = 1; i <= n; i++)G[0].pb(mk(i, 0));
    for(int i = 1; i <= n; i++)h[i] = INF;
    for(int i = 0; i <= n; i++){
        for(int j = 0; j <= n; j++){
            for(auto p : G[j]){
                if(h[p.fi] > h[j]+p.se){
                    h[p.fi] = h[j]+p.se;
                    if(i == n)return 1;
                }
            }
        }
    }
    return 0;
}
void dijkstra(int s)
{
    for(int i = 1; i <= n; i++)dist[s][i] = INF;
    dist[s][s] = 0;
    priority_queue<pii, vector<pii>, greater<pii> > pq;
    pq.push(mk(0, s));
    while(!pq.empty()){
        pii p = pq.top(); pq.pop();
        if(dist[s][p.se] < p.fi)continue;
        for(auto v : G[p.se]){
            if(dist[s][v.fi] > dist[s][p.se]+v.se){
                dist[s][v.fi] = dist[s][p.se]+v.se;
                pq.push(mk(dist[s][v.fi], v.fi));
            }
        }
    }
}
bool johnson()
{
    if(bellmanFord())return 0;//存在负圈
    for(int i = 1; i <= n; i++){
        for(auto &v : G[i])v.se += h[i]-h[v.fi];
    }
    for(int i = 1; i <= n; i++)dijkstra(i);
    return 1;//不存在负圈
}
```

Kosaraju

```
#include <vector>
using namespace std;
const int maxn = 1e5+10;
vector<int> G[maxn], G2[maxn];
vector<int> S;
int scc[maxn], cnt, n, m;
bool vis[maxn];
void dfs1(int u)
{
    if(vis[u])return;
    else vis[u] = 1;
```

```
    for(auto v : G[u])dfs1(v);
    S.push_back(u);
}
void dfs2(int u)
{
    if(scc[u])return;
    scc[u] = cnt;
    for(auto v : G2[u])dfs2(v);
}
void korasaju()
{
    for(int i = 1; i <= n; i++)dfs1(i);
    for(int i = n-1; i >= 0; i--){
        if(!scc[S[i]]){
            cnt++;
            dfs2(S[i]);
        }
    }
}
```

Lca_ST

```
#include <cstdio>
#include <vector>
#include <cmath>
using namespace std;
const int maxn = 1e5+10;
const int maxd = 20;
vector<int> G[maxn];
int beg[maxn], id[maxn*2], dep[maxn*2], LOG[maxn*2];
int st[maxn][maxd];
int n, tot;
void init_rmq()
{
    for(int i = 0; i < tot; i++)st[i][0] = i;
    for(int i = 0, j = 0; i < tot; i++){
        while(i >= (1<<(j+1))) j++;
        LOG[i] = j;
    }
    for(int j = 1; (1<<j) <= tot; j++){
        for(int i = 0; i+(1<<j)-1 < tot; i++){
            int u = st[i][j-1], v = st[i+(1<<(j-1))][j-
1];
            st[i][j] = dep[u]<dep[v]?u:v;
        }
    }
}
int query(int l, int r)
{
    int k = LOG[r-l+1];
    int u = st[l][k], v = st[r-(1<<k)+1][k];
    return dep[u]<dep[v]?u:v;
}
int lca(int x, int y)
{
    int l = beg[x], r = beg[y];
    if(l > r)swap(l, r);
    return id[query(l, r)];
}
void dfs(int u, int f, int d)
{
    beg[u] = tot; id[tot] = u; dep[tot] = d; tot++;
    for(auto v : G[u]){
        if(v == f)continue;
        dfs(v, u, d+1);
        id[tot] = u; dep[tot] = d; tot++;
    }
}
```

Lca_doubling

```
#include <algorithm>
using namespace std;
const int maxn = 1e6+10;
const int maxd = 21;
int anc[maxn][maxd];//anc[i][0]->par[i]
int dep[maxn];
int n;//1-index
void init_lca()
{
    for(int i = 1; i < maxd; i++){
        for(int j = 1; j <= n; j++){
            anc[j][i] = anc[anc[j][i-1]][i-1];
        }
    }
```

```
}
int lca(int u, int v)
{
    if(dep[u] < dep[v])swap(u, v);
    for(int i = maxd-1; i >= 0; i--){
        if(dep[anc[u][i]] >= dep[v])u = anc[u][i];
    }
    if(u == v)return u;
    for(int i = maxd-1; i >= 0; i--){
        if(anc[u][i] != anc[v][i])u = anc[u][i], v =
anc[v][i];
    }
    return anc[u][0];
}
```

Lca_tarjan

```
#include <cstdio>
#include <vector>
using namespace std;
#define pb push_back
#define SZ(x) ((int)x.size())
const int maxn = 1e5+10;
struct data
{
    int v, rev, f;
    data(){}
    data(int v, int rev):v(v), rev(rev){}
};
vector<data> q[maxn];
vector<int> G[maxn];
int par[maxn], ans[maxn];
bool vis[maxn];
void add_query(int u, int v)
{
    q[u].pb(data(v, SZ(q[v])));
    q[v].pb(data(u, SZ(q[u])-1));
}
int find(int x)
{
    return par[x] == x ? x : par[x] = find(par[x]);
}
void unite(int u, int v)
{
    u = find(u); v = find(v);
    if(u != v)par[u] = v;
}
void dfs(int u)
{
    vis[u] = 1;
    for(auto v : G[u]){
        if(!vis[v]){
            dfs(v);
            unite(v, u);
        }
    }
    for(auto &d : q[u]){
        if(vis[d.v])d.f = q[d.v][d.rev].f = find(d.v);
    }
}
```

MatrixTree

```
#include <vector>
using namespace std;
typedef long long ll;
#define ABS(x) ((x)>=0?(x):(-(x)))
const int maxn = 5e2+10;
vector<int> G[maxn];
int n; //0-index
ll det()
{
    ll ret = 1;
    for(int i = 1; i < n; i++){
        for(int j = i+1; j < n; j++){
            while(G[j][i]){
                ll t = G[i][i]/G[j][i];
                for(int k = i; k < n; k++){
                    G[i][k] -= G[j][k]*t;
                }
                for(int k = i; k < n; k++){
                    swap(G[i][k], G[j][k]);
                }
                ret = -ret;
```

```
            }
        }
        if(!G[i][i])return 0;
        ret *= G[i][i];
    }
    return ABS(ret);
}
```

MaximumDensitySubgraph

```
#include <cstring>
#include <queue>
#include <vector>
#include <cstdio>
using namespace std;
typedef pair<int, int> pii;
#define mk make_pair
#define pb push_back
#define SZ(x) ((int)x.size())
#define fi first
#define se second
const double eps = 1e-10;
const int INF = 0x3f3f3f3f;
const int maxn = 1e4+10;
const int maxm = 1e4+10;
struct edge
{
    int to, rev;
    long double cap;
    edge(){}
    edge(int to, int rev, long double cap):to(to),
rev(rev), cap(cap){}
};
vector<edge> G[maxn];
int lev[maxn];
pii in[maxm];
bool vis[maxn];
int n, m, cnt;
void init()
{
    for(int i = 0; i < maxn; i++)G[i].clear();
}
void add_edge(int u, int v, long double cap)
{
    G[u].pb(edge(v, SZ(G[v]), cap));
    G[v].pb(edge(u, SZ(G[u])-1, 0));
}
bool solve(long double mid)
{
    init();
    int src = n+m+1, dst = src+1;
    long double sum = 0;
    for(int i = 1; i <= n; i++)add_edge(i, dst, mid);
    for(int i = 1; i <= m; i++){
        add_edge(src, i+n, 1), sum += 1;
        add_edge(i+n, in[i].fi, INF);
        add_edge(i+n, in[i].se, INF);
    }
    long double ret = sum-max_flow(src, dst);
    return ret > eps;
}
void dfs(int u)
{
    vis[u] = 1;
    for(auto e : G[u]){
        if(!vis[e.to] && e.cap > eps)dfs(e.to);
    }
}
vector<int> maximumDensitySubgraph()
{
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= m; i++){
        int u, v;
        scanf("%d%d", &u, &v);
        in[i] = mk(u, v);
    }
    long double l = 0, r = m, ans = 0;
    while(l+1.0/(n*n) <= r){
        long double mid = (l+r)/2;
        if(solve(mid)){
            ans = mid;
            l = mid;
        }
        else r = mid;
```

```
    }
    solve(ans);
    int src = n+m+1;
    dfs(src);
    vector<int> vec;
    for(int i = 1; i <= n; i++)if(vis[i])vec.pb(i);
    if(vec.empty())vec.pb(1);
    return vec;
}
```

MaximumWeightClosureGraph

```
#include <cstdio>
#include <vector>
using namespace std;
#define pb push_back
#define SZ(x) ((int)x.size())
typedef long long ll;
const int maxn = 5e3+10;
const int INF = 0x3f3f3f3f;
struct edge
{
    int to, cap, rev;
    edge(){}
    edge(int to, int cap, int rev):to(to), cap(cap),
rev(rev){}
};
vector<edge> G[maxn];
bool vis[maxn];
int cnt;
void add_edge(int u, int v, int cap)
{
    G[u].pb(edge(v, cap, SZ(G[v])));
    G[v].pb(edge(u, cap, SZ(G[u])-1));
}
void dfs(int u)
{
    vis[u] = 1;
    for(auto e : G[u]){
        if(e.cap && !vis[e.to])dfs(e.to);
    }
}
ll maximumWeightClosureGraph()
{
    int n, m;
    scanf("%d%d", &n, &m);
    int src = n+1, dst = src+1;
    ll sum = 0;
    for(int i = 1; i <= n; i++){
        int w; scanf("%d", &w);
        if(w > 0)add_edge(src, i, w), sum += w;
        else if(w < 0)add_edge(i, dst, -w);
    }
    for(int i = 1; i <= m; i++){
        int u, v;
        scanf("%d%d", &u, &v);
        add_edge(u, v, INF);
    }
    ll ans = sum-max_flow(src, dst);
    dfs(src);
    vector<int> vec;
    for(int i = 1; i <= n; i++){
        if(vis[i])vec.pb(i);//所选点集
    }
    return ans;
}
```

MixedEuler

```
#include <cstdio>
#include <stack>
#include <vector>
using namespace std;
typedef pair<int, int> pii;
#define pb push_back
#define mk make_pair
#define SZ(x) ((int)x.size())
#define fi first
#define se second
const int maxn = 1e3+10;
const int maxm = 2e4+10;
struct edge
{
    int to, cap, rev;
```

```
    edge(){}
    edge(int to, int cap, int rev):to(to), cap(cap),
rev(rev){}
};
vector<edge> G[maxn];
vector<int> G2[maxn];
int lev[maxn], deg[maxn];
stack<int> S;
pii p[maxm];
int n, m;
void add_edge(int u, int v, int cap)
{
    G[u].pb(edge(v, cap, SZ(G[v])));
    G[v].pb(edge(u, 0, SZ(G[u])-1));
}
bool mixedEuler()
{
    int num = 0;
    for(int i = 0; i < m; i++){
        int u, v, type;
        scanf("%d%d%d", &u, &v, &type);
        deg[u]++, deg[v]--;
        if(type == 1)p[num++] = mk(u, v);//undirected
        else G2[u].pb(v);//directed
    }
    bool flg = 1;
    int src = n+1, dst = src+1;
    for(int i = 1; i <= n; i++){
        if(deg[i]%2 != 0){
            flg = 0;
            break;
        }
        if(deg[i] > 0)add_edge(src, i, deg[i]/2);
        else if(deg[i] < 0)add_edge(i, dst, -deg[i]/2);
    }
    if(!flg)return 0;//没有合法欧拉路
    for(int i = 0; i < num; i++)add_edge(p[i].fi,
p[i].se, 1);
    max_flow(src, dst);
    for(auto e : G[src]){
        if(e.cap){
            flg = 0;
            break;
        }
    }
    if(!flg)return 0;
    else{
        for(int i = 1; i <= n; i++){
            for(auto e : G[i]){
                if(e.to <= n && e.cap)G2[i].pb(e.to);
            }
        }
        euler();
    }
    return 1;
}
```

MstCounting

```
#include <vector>
using namespace std;
typedef long long ll;
#define pb push_back
const int maxn = 110;
const int maxe = 1010;
struct edge
{
    int u, v, w;
    bool operator < (const edge &rbs)const
    {
        return w<rbs.w;
    }
}es[maxe];
ll mat[maxn][maxn];
int link[maxn][maxn];
int fa[maxn];
int par[maxn];
bool vis[maxn];
vector<int> vec[maxn];
int n, m;
ll mod;
void init()
{
    memset(link, 0, sizeof(link));
```

```
    memset(vis, 0, sizeof(vis));
    for(int i = 0; i <= n; i++)fa[i] = i, par[i] = i;
}
int find(int x, int arr[])
{
    return x==arr[x]?x:arr[x]=find(arr[x], arr);
}
ll det(int sz)
{
    ll ret = 1;
    for(int i = 0; i < sz; i++){
        for(int j = 0; j < sz; j++)mat[i][j] =
(mat[i][j]%mod+mod)%mod;
    }
    for(int i = 1; i < sz; i++){
        for(int j = i+1; j < sz; j++){
            while(mat[j][i]){
                ll t = mat[i][i]/mat[j][i];
                for(int k = i; k < sz; k++){
                    mat[i][k] = (mat[i][k]-
mat[j][k]*t)%mod;
                    swap(mat[i][k], mat[j][k]);
                }
                ret = (-ret+mod)%mod;
            }
        }
        if(!mat[i][i])return 0;
        ret = ret*mat[i][i]%mod;
    }
    return (ret+mod)%mod;
}
ll solve()
{
    sort(es, es+m);
    ll ret = 1;
    int pre = -1;
    for(int i = 0; i <= m; i++){
        if(es[i].w != pre || i == m){
            for(int j = 0; j < n; j++){
                if(vis[j]){
                    int fj = find(j, par);
                    vec[fj].pb(j);
                    fa[j] = fj;
                    vis[j] = 0;
                }
            }
            for(int j = 0; j < n; j++){
                int sz = vec[j].size();
                if(sz <= 1)continue;
                memset(mat, 0, sizeof(mat));
                for(int k = 0; k < sz; k++){
                    for(int h = k+1; h < sz; h++){
                        int u = vec[j][k], v =
vec[j][h];
                        mat[k][h] = mat[h][k] = -
link[u][v];
                        mat[k][k] += link[u][v];
                        mat[h][h] += link[v][u];
                    }
                }
                ret = ret*det(sz)%mod;
            }
            for(int j = 0; j < n; j++){
                vec[j].clear();
            }
            if(i == m)break;
        }
        int u = es[i].u, v = es[i].v;
        u = find(u, fa); v = find(v, fa);
        if(u == v)continue;
        vis[u] = vis[v] = 1;
        par[find(v, par)] = find(u, par);
        link[u][v]++; link[v][u]++;
        pre = es[i].w;
    }
    int flg = find(0, fa);
    for(int i = 1; i < n; i++){
        if(find(i, fa) != flg)return 0;
    }
    return (ret+mod)%mod;
}
```

PointDivideAndConquer

```
#include <cstdio>
#include <algorithm>
#include <vector>
#include <cstring>
using namespace std;
typedef pair<int, int> pii;
#define fi first
#define se second
#define pb push_back
#define mk make_pair
const int maxn = 1e5+10;
int sz[maxn], mx[maxn];
bool vis[maxn];
vector<pii> G[maxn];
vector<int> dis;
int n, k;
int ans, ms, rt;

void init()
{
    for(int i = 1; i <= n; i++)vis[i] = 0, G[i].clear();
    ans = 0;
}

void dfs_size(int u, int f)
{
    sz[u] = 1;
    mx[u] = 0;
    for(auto p : G[u]){
        if(p.fi != f && !vis[p.fi]){
            dfs_size(p.fi, u);
            sz[u] += sz[p.fi];
            if(sz[p.fi] > mx[u])mx[u] = sz[p.fi];
        }
    }
}

void dfs_root(int r, int u, int f)
{
    if(sz[r]-sz[u] > mx[u])mx[u] = sz[r]-sz[u];
    if(mx[u] < ms)ms = mx[u], rt = u;
    for(auto p : G[u]){
        if(p.fi != f && !vis[p.fi])dfs_root(r, p.fi, u);
    }
}

void dfs_dis(int u, int d, int f)
{
    dis.pb(d);
    for(auto p : G[u]){
        if(p.fi != f && !vis[p.fi])dfs_dis(p.fi, d+p.se,
u);
    }
}

int calc(int u, int d)
{
    int res = 0;
    dis.clear();
    dfs_dis(u, d, 0);
    sort(dis.begin(), dis.end());
    int i = 0, j = (int)dis.size()-1;
    while(i < j){
        while(dis[i]+dis[j]> k && i < j)j--;
        res += j-i;
        i++;
    }
    return res;
}

void dfs(int u)
{
    ms = n;
    dfs_size(u, 0);
    dfs_root(u, u, 0);
    ans += calc(rt, 0);
    vis[rt] = 1;
    int pnt = rt;
    for(auto p : G[pnt]){
        if(!vis[p.fi]){
            ans -= calc(p.fi, p.se);
            dfs(p.fi);
```

```
        }
    }
}

void pointDivideAndConquer()
{
    dfs(1);
    printf("%d\n", ans);
}
```

Prim_heap

```
#include <cstring>
#include <queue>
using namespace std;
typedef pair<int, int> pii;
#define fi first
#define se second
#define mk make_pair
const int maxn = 1e6+10;
vector<pii> G[maxn]; //需要去重边
int cost[maxn];
bool vis[maxn];
int prim()
{
    priority_queue<pii, vector<pii>, greater<pii> > pq;
    memset(cost, 0x3f, sizeof(cost));
    cost[1] = 0;
    pq.push(mk(0, 1));
    int ret = 0;
    while(!pq.empty()){
        pii p = pq.top(); pq.pop();
        if(vis[p.se])continue;
        vis[p.se] = 1;
        ret += cost[p.se];
        for(auto v : G[p.se]){
            if(cost[v.fi] > v.se){
                cost[v.fi] = v.se;
                pq.push(mk(cost[v.fi], v.fi));
            }
        }
    }
    return ret;
}
```

Prufer

给一棵树，它的 prufer 序列由一下步骤得到：

（1）选择度数为 1 的编号最小的点，把它删掉并把和它相连的点加入序列
（2）重复第一步直到剩下两个点

prufer 序列的性质：

长度为 n-2,记每个点的度数为 di,那么每个点都会在 prufer 序列中出现 di-1 次

StoerWagner

```
#include <cstdio>
#include <algorithm>
#include <cstring>
using namespace std;
const int INF = 0x3f3f3f3f;
const int maxn = 5e2+10;
int G[maxn][maxn];
int dis[maxn], id[maxn];
int n, m;

int StoerWagner(int n)
{
    int ret = INF;
    for(int i = 0; i < n; i++)id[i] = i;
    while(n > 1){
        memset(dis, 0, sizeof(dis));
        int k = 0;
        for(int i = 1; i < n; i++){
            swap(id[k], id[i-1]);
            for(int j = k = i; j < n; j++){
                dis[id[j]] += G[id[i-1]][id[j]];
                if(dis[id[j]] > dis[id[k]])k = j;
            }
        }
        ret = min(ret, dis[id[k]]);
```

```
        int s = id[n-2], t = id[n-1];
        for(int i = 0; i < n-2; i++){
            int u = id[i];
            G[u][s] = G[s][u] += G[u][t];
        }
        id[k] = id[n--];
    }
    return ret;
}
```

Tarjan_cutEdge

```
#include <cstdio>
#include <vector>
using namespace std;
typedef pair<int, int> pii;
#define pb push_back
#define mk make_pair
const int maxn = 1e5+10;
int dfn[maxn], low[maxn];
vector<int> G[maxn];
vector<pii> cut;
int n, idx;
void dfs(int u, int f)
{
    dfn[u] = low[u] = idx++;
    for(auto v : G[u]){
        if(v == f)continue;
        if(!dfn[v]){
            dfs(v, u);
            low[u] = min(low[u], low[v]);
            if(low[v] > dfn[u]){
                cut.pb(mk(min(u, v), max(u, v)));
            }
        }
        else if(dfn[v] < dfn[u])low[u] = min(low[u],
dfn[v]);
    }
}
void tarjan()
{
    for(int i = 1; i <= n; i++){
        if(!dfn[i])dfs(i, 0);
    }
}
```

Tarjan_cutPoint

```
#include <cstdio>
#include <vector>
using namespace std;
const int maxn = 1e5+10;
vector<int> G[maxn];
int dfn[maxn], low[maxn], cut[maxn];
int n, idx;
void dfs(int u, int f)
{
    dfn[u] = low[u] = ++idx;
    int cnt = 0;
    for(auto v : G[u]){
        if(v == f)continue;
        if(!dfn[v]){
            cnt++;
            dfs(v, u);
            low[u] = min(low[u], low[v]);
            if(low[v] >= dfn[u])cut[u] = 1;
        }
        if(dfn[v] < dfn[u])low[u] = min(low[u], dfn[v]);
    }
    if(!f && cnt <= 1)cut[u] = 0;
}
int tarjan()
{
    int ret = 0;
    for(int i = 1; i <= n; i++){
        if(!dfn[i])dfs(i, 0);
    }
    for(int i = 1; i <= n; i++){
        if(cut[i])ret++;
    }
    return ret;
}
```

Tarjan_scc

```
#include <algorithm>
#include <vector>
#include <stack>
using namespace std;
const int maxn = 1e5+10;
vector<int> G[maxn];
stack<int> sk;
int low[maxn], dfn[maxn], scc[maxn], num[maxn];
int n, cnt, idx;
void dfs(int u, int f)
{
    low[u] = dfn[u] = ++idx;
    sk.push(u);
    for(auto v : G[u]){
        if(v == f)continue;
        if(!dfn[v]){
            dfs(v, u);
            low[u] = min(low[u], low[v]);
        }
        else if(!scc[v])low[u] = min(low[u], dfn[v]);
    }
    if(low[u] == dfn[u]){
        cnt++;
        while(1){
            int x = sk.top(); sk.pop();
            scc[x] = cnt; num[cnt]++;
            if(x == u)break;
        }
    }
}
void tarjan()
{
    for(int i = 1; i <= n; i++){
        if(!dfn[i])dfs(i, 0);
    }
}
```

ThreeMemberedRing

```
#include <vector>
using namespace std;
#define pb push_back
const int maxn = 2e5+10;
struct edge
{
    int u, v;
}es[maxn];
vector<int> G[maxn];
int deg[maxn], vis[maxn];
int n, m;
int Counting()
{
    int ans = 0;
    for(int i = 1; i <= m; i++){
        int u = es[i].u, v = es[i].v;
        if(deg[u] < deg[v] || (deg[u] == deg[v] && u <
v))G[u].pb(v);
        else G[v].pb(u);
    }
    for(int i = 1; i <= m; i++){
        int u = es[i].u, v = es[i].v;
        for(auto w : G[u])vis[w] = i;
        for(auto w : G[v])if(vis[w] == i)ans++;
    }
    return ans;
}
```

Tarjan_bcc_edge

```
#include <cstdio>
#include <algorithm>
using namespace std;
const int maxn = 1e5+10;
const int maxm = 1e6+10;
struct edge
{
    int to, nxt;
}es[maxm];
int dfn[maxn], low[maxn], bcc[maxn], head[maxn];
bool cut[maxm];
int n, m, idx, cnt;
void dfs(int u, int f)
{
```

```
    dfn[u] = low[u] = ++idx;
    for(int i = head[u]; ~i; i = es[i].nxt){
        int v = es[i].to;
        if(v == f)continue;
        if(!dfn[v]){
            dfs(v, u);
            low[u] = min(low[u], low[v]);
            if(low[v] >= dfn[u])cut[i>>1] = 1;
        }
        else if(dfn[v] < dfn[u])low[u] =
min(low[u], dfn[v]);
    }
}

void dfs2(int u)
{
    bcc[u] = cnt;
    for(int i = head[u]; ~i; i = es[i].nxt){
        if(!cut[i>>1])dfs2(es[i].to);
    }
}

void tarjan()
{
    for(int i = 1; i <= n; i++){
        if(!dfn[i])dfs(i, 0);
    }
    for(int i = 1; i <= n; i++){
        if(!bcc[i]){
            cnt++;
            dfs2(i);
        }
    }
}
```

Tarjan_bcc_point

```
#include <cstdio>
#include <stack>
#include <vector>
using namespace std;
typedef pair<int, int> pii;
#define fi first
#define se second
#define mk make_pair
const int maxn = 1e5+10;
int dfn[maxn], low[maxn], bcc[maxn];
stack<pii> sk;
vector<int> G[maxn];
int n, m, idx, cnt;

void dfs(int u, int f)
{
    dfn[u] = low[u] = ++idx;
    for(auto v : G[u]){
        if(!dfn[v]){
            sk.push(mk(u, v));
            dfs(v, u);
            low[u] = min(low[u], low[v]);
            if(low[v] >= dfn[u]){
                cnt++;
                while(1){
                    pii p = sk.top(); sk.pop();
                    bcc[p.fi] = bcc[p.se] = cnt;
                    if(p.fi == u && p.se ==
v)break;
                }
            }
        }
        else if(dfn[v] < dfn[u] && v != f){
            sk.push(mk(u, v));
            low[u] = min(low[u], dfn[v]);
        }
    }
}

void tarjan()
{
```

```
    for(int i = 1; i <= n; i++){
        if(!dfn[i])dfs(i, 0);
    }
}
```

Zhuliu

```
//O(nm)
#include <cstring>
const int INF = 0x3f3f3f3f;
const int maxn = 1010;
const int maxe = 40100;
struct edge
{
    int u, v, w;
}es[maxe];
int vis[maxn], id[maxn], pre[maxn], in[maxn];
int n, m;
//如果是无根的最小树形图，那么建一个超级根向所有点连边，权值为
sum(w)+1，如果 ret-sum>=sum，无解，否则有解
int zhuliu(int root)
{
    int ret = 0;
    while(1){
        memset(id, -1, sizeof(id));
        memset(vis, -1, sizeof(vis));
        memset(in, 0x3f, sizeof(in));
        for(int i = 0; i < m; i++)if(es[i].u != es[i].v
&& in[es[i].v] > es[i].w){
            in[es[i].v] = es[i].w;
            pre[es[i].v] = es[i].u;
        }
        for(int i = 0; i < n; i++)if(i != root && in[i]
== INF)return INF;//不联通
        in[root] = 0;
        int loop = 0;
        for(int i = 0; i < n; i++){
            ret += in[i];
            int v = i;
            while(v != root && id[v] == -1 && vis[v] !=
i){
                vis[v] = i;
                v = pre[v];
            }
            if(vis[v] == i){
                for(int j = pre[v]; j != v; j =
pre[j])id[j] = loop;
                id[v] = loop++;
            }
        }
        if(!loop)break;
        for(int i = 0; i < n; i++)if(id[i] == -1)id[i] =
loop++;
        for(int i = 0; i < m; i++){
            int v = es[i].v;
            es[i].u = id[es[i].u];
            es[i].v = id[es[i].v];
            if(es[i].u != es[i].v)es[i].w -= in[v];
        }
        root = id[root];
        n = loop;
    }
    return ret;
}
```

Geometry

2DCommon

```
#include <cstdio>
#include <cmath>
using namespace std;
const double eps = 1e-10;
int dcmp(double x){return x<-eps?-1:x>eps;}
struct Point
{
    double x, y;
    Point(){}
    Point(double x, double y):x(x), y(y){}
    Point operator + (const Point &rhs)const{return
Point(x+rhs.x, y+rhs.y);}
    Point operator - (const Point &rhs)const{return
Point(x-rhs.x, y-rhs.y);}
```

```
    Point operator * (const double p)const{return
Point(x*p, y*p);}
    Point operator / (const double p)const{return
Point(x/p, y/p);}
    bool operator < (const Point &rhs)const{return x <
rhs.x || (x == rhs.x && y < rhs.y);}
    bool operator == (const Point
&rhs)const{return !dcmp(x-rhs.x) && !dcmp(y-rhs.y);}
};
typedef Point Vector;
double Dot(Vector A, Vector B){return A.x*B.x+A.y*B.y;}
double Cross(Vector A, Vector B){return A.x*B.y-
A.y*B.x;}
double Length(Vector A){return sqrt(Dot(A, A));}
double Angle(Vector A, Vector B){return acos(Dot(A,
B)/Length(A)/Length(B));}
double Area(Point A, Point B, Point C){return Cross(B-A,
C-A);}
Vector Rotate(Vector A, double rad){return
Vector(A.x*cos(rad)-A.y*sin(rad),
A.x*sin(rad)+A.y*cos(rad));}
Vector Normal(Vector A)
{
    double L = Length(A);
    return Vector(-A.y/L, A.x/L);
}
Point GetLineIntersection(Point P, Vector v, Point Q,
Vector w)
{
    Vector u = P-Q;
    double t = Cross(w, u)/Cross(v, w);
    return P+v*t;
}//求两直线交点(需保证 Cross(v, w) != 0)
double DistanceToLine(Point P, Point A, Point B)
{
    Vector v1 = B-A, v2 = P-A;
    return fabs(Cross(v1, v2))/Length(v1);
}//求点 P 到直线 AB 的距离
double DistanceToSegment(Point P, Point A, Point B)
{
    if(A == B)return Length(P-A);
    Vector v1 = B-A, v2 = P-A, v3 = P-B;
    if(dcmp(Dot(v1, v2)) < 0)return Length(v2);
    else if(dcmp(Dot(v1, v3)) > 0)return Length(v3);
    else return fabs(Cross(v1, v2))/Length(v1);
}//求点 P 到线段 AB 的距离
Point GetLineProjection(Point P, Point A, Point B)
{
    Vector v = B-A;
    return A+v*(Dot(v, P-A)/Dot(v, v));
}//求点 P 在直线 AB 上的投影
bool SegmentProperIntersection(Point a1, Point a2, Point
b1, Point b2)
{

    double c1 = Cross(a2-a1, b1-a1), c2 = Cross(a2-a1,
b2-a1),
        c3 = Cross(b2-b1, a1-b1), c4 = Cross(b2-b1, a2-
b1);
    return dcmp(c1)*dcmp(c2) < 0 && dcmp(c3)*dcmp(c4) <
0;
}//线段相交判定(不含端点)
bool OnSegment(Point p, Point a1, Point a2)
{
    return dcmp(Cross(a1-p, a2-p)) == 0 && dcmp(Dot(a1-
p, a2-p)) < 0;
}
double PolygonArea(Point*p, int n)
{
    double area = 0;
    for(int i = 1; i < n-1; i++)area += Cross(p[i]-p[0],
p[i+1]-p[0]);
    return area/2;
}//多边形(凹凸皆可)有向面积，逆时针为正，
bool isPointInPolygon(Point p, int n, Point* poly)
{
    int wn = 0;
    for(int i = 0; i < n; i++){
        if(OnSegment(p, poly[i], poly[(i+1)%n]))return -
1;//在边界上
        int k = dcmp(Cross(poly[(i+1)%n]-poly[i], p-
poly[i]));
```

```
        int d1 = dcmp(poly[i].y-p.y);
        int d2 = dcmp(poly[(i+1)%n].y-p.y);
        if(k > 0 && d1 <= 0 && d2 > 0)wn++;
        if(k < 0 && d2 <= 0 && d1 > 0)wn--;
    }
    return wn;
}//判断点 p 是否在多边形内
```

ConvexHull

```
#include <cstdio>
#include <algorithm>
#include "2DCommon.h"
int ConverHull(Point* p, int n, Point* ch)
{
    sort(p, p+n);
    int m = 0;
    for(int i = 0; i < n; i++){
        while(m > 1 && Cross(ch[m-1]-ch[m-2], p[i]-ch[m-
2]) <= 0)m--;
        ch[m++] = p[i];
    }
    int k = m;
    for(int i = n-2; i >= 0; i--){
        while(m > k && Cross(ch[m-1]-ch[m-2], p[i]-ch[m-
2]) <= 0)m--;
        ch[m++] = p[i];
    }
    if(n > 1)m--;
    return m;
}
```

HalfplaneIntersection

```
#include <cstdio>
#include <algorithm>
#include "2DCommon.h"
struct Line
{
    Point P;
    Vector v;
    double ang;
    Line(){}
    Line(Point P, Vector v):P(P), v(v){}
    bool operator < (const Line& L)const{return ang <
L.ang;}
};
bool OnLeft(Line L, Point p)
{
    return Cross(L.v, p-L.P);
}
Point GetIntersection(Line a, Line b)
{
    Vector u = a.P-b.P;
    double t = Cross(b.v, u)/Cross(a.v, b.v);
    return a.P+a.v*t;
}
int HalfplaneIntersection(Line* L, int n, Point* poly)
{
    sort(L, L+n);
    int first, last;
    Point *p = new Point[n];
    Line *q = new Line[n];
    q[first=last=0] = L[0];
    for(int i = 1; i < n; i++){
        while(first < last && !OnLeft(L[i], p[last-
1]))last--;
        while(first < last && !OnLeft(L[i],
p[first]))first++;
        q[last++] = L[i];
        if(fabs(Cross(q[last].v, q[last-1].v)) < eps){
            last--;
            if(OnLeft(q[last], L[i].P))q[last] = L[i];
        }
        if(first < last)p[last-1] =
GetIntersection(q[last-1], q[last]);
    }
    while(first < last && !OnLeft(q[first], p[last-
1]))last--;
    if(last-first <= 1)return 0;
    p[last] = GetIntersection(q[last], q[first]);
    int m = 0;
    for(int i = first; i <= last; i++)poly[m++] = p[i];
    return m;
```

```
}
```

RotateCalip

```
/*旋转卡壳求凸包中最远点对*/
double Dis(Point a, Point b){return Length(a-b);}
double Area(Point a, Point b, Point c){return Cross(b-
a,c-a);}
double rot_calip(Point* p, int n){
    double res = 0;
    p[n] = p[0];
    for(int i = 0, j = 1; i < n; i++){
        while(Area(p[i],p[i+1],p[j+1]) >
Area(p[i],p[i+1],p[j])) j=(j+1)%n;
        res = max(res, Dis(p[i], p[j]));
        res = max(res, Dis(p[i+1], p[j]));
    }
    return res;
}
```

RotatingCalipers

```
#include <cstdio>
#include <algorithm>
#include "2DCommon.h"
using namespace std;
double Dis(Point a, Point b){return Length(a-b);}
double Diameter(Point* p, int n)
{
    p[n] = p[0];
    double ret = 0;
    for(int i = 0, j = 1; i < n; i++){
        while(Area(p[i], p[i+1], p[j+1]) > Area(p[i],
p[i+1], p[j]))j = (j+1)%n;
        ret = max(ret, Dis(p[i], p[j]));
        ret = max(ret, Dis(p[i+1], p[j]));
    }
    return ret;
}
double MaximumTriangle(Point *p, int n)
{
    if(n <= 2)return 0;
    p[n] = p[0];
    double ret = 0;
    for(int i = 0, u = 1, v = 2; i < n; i++){
        while(Area(p[i], p[u], p[(v+1)%n]) > Area(p[i],
p[u], p[v]))v = (v+1)%n;
        while(Area(p[i], p[(u+1)%n], p[v]) > Area(p[i],
p[u], p[v]))u = (u+1)%n;
        ret = max(ret, Area(p[i], p[u], p[v]));
    }
    return ret;
}
```

Math

FFT

```
struct Complex{
    double a,b;
    Complex(double aa = 0, double bb = 0){a = aa, b =
bb;}
    Complex operator + (const Complex& rbs) const{
        return Complex(a+rbs.a, b+rbs.b);
    }
    Complex operator - (const Complex& rbs) const{
        return Complex(a-rbs.a, b-rbs.b);
    }
    Complex operator * (const Complex& rbs) const{
        return Complex(a*rbs.a-b*rbs.b,
a*rbs.b+b*rbs.a);
    }
};
void change(Complex y[], int len){
    for(int i = 1, j = len/2; i < len-1; i++){
        if(i < j) swap(y[i], y[j]);
        int k = len>>1;
        while(1){
            j ^= k;
            if(j&k) break;
            k >>= 1;
        }
    }
}
```

```
void fft(Complex y[], int len, int o){
    change(y, len);
    for(int h = 2; h <= len; h <<= 1){
        Complex wn(cos(-o*2*PI/h), sin(-o*2*PI/h));
        for(int i = 0; i < len; i += h){
            Complex w(1,0);
            for(int j = i; j < i+h/2; j++){
                Complex l = y[j], r = w*y[j+h/2];
                y[j] = l+r; y[j+h/2] = l-r;
                w = w*wn;
            }
        }
    }
    if(o == -1) for(int i = 0; i < len; i++) y[i].a /=
len;
}
```

FWT

```
const int MOD = 1e9+7;
const int inv2 = (MOD+1) >> 1;
///fwt opt = 1, ufwt opt = -1
void FWT_or(int *a, int N, int opt=1)
{
    for(int i=1;i<N;i<<=1)
        for(int p=i<<1,j=0;j<N;j+=p)
            for(int k=0;k<i;++k)

if(opt==1)a[i+j+k]=(a[j+k]+a[i+j+k])%MOD;
                else a[i+j+k]=(a[i+j+k]+MOD-a[j+k])%MOD;
}
void FWT_and(int *a, int N, int opt=1)
{
    for(int i=1;i<N;i<<=1)
        for(int p=i<<1,j=0;j<N;j+=p)
            for(int k=0;k<i;++k)
                if(opt==1)a[j+k]=(a[j+k]+a[i+j+k])%MOD;
                else a[j+k]=(a[j+k]+MOD-a[i+j+k])%MOD;
}
void FWT_xor(int *a, int N, int opt=1)
{
    for(int i=1;i<N;i<<=1)
        for(int p=i<<1,j=0;j<N;j+=p)
            for(int k=0;k<i;++k)
            {
                int X=a[j+k],Y=a[i+j+k];
                a[j+k]=(X+Y)%MOD;a[i+j+k]=(X+MOD-Y)%MOD;
                if(opt==-
1)a[j+k]=111*a[j+k]*inv2%MOD,a[i+j+k]=111*a[i+j+k]*inv2%
MOD;
            }
}
```

Linear_sieve

```
//d[i]为 i 的最小质因子的次幂
const int MAX = 1e6+10;
int
pnum,p[MAX],mob[MAX],noprime[MAX],facnum[MAX],d[MAX],phi
[MAX];
void get_all()
{
    pnum = 0;
    phi[1] = 1;
    mob[1] = 1;
    facnum[1] = 1;
    for(int i = 2; i < MAX; i++)
    {
        if(!noprime[i])
        {
            phi[i] = i - 1;
            mob[i] = -1;
            p[pnum ++] = i;
            facnum[i] = 2;
            d[i] = 1;
        }
        for(int j = 0; j < pnum && i * p[j] < MAX; j++)
        {
            noprime[i * p[j]] = true;
            if(i % p[j] == 0)
            {
                phi[i * p[j]] = phi[i] * p[j];
                mob[i * p[j]] = 0;
```

```
                facnum[i * p[j]] = facnum[i] / (d[i] +
1) * (d[i] + 2);
                d[i * p[j]] = d[i] + 1;
                break;
            }
            phi[i * p[j]] = phi[i] * (p[j] - 1);
            mob[i * p[j]] = -mob[i];
            facnum[i * p[j]] = facnum[i] * 2;
            d[i * p[j]] = 1;
        }
    }
}
```

NTT

```
/******
 * NTT Algorithm
 * Time Complexity: O(nlog n)
 * available data range rely on the value of g and
modn
 * default range:
 * * 0 <= ai < 1004535809 (a number slightly bigger
than 1e9)
 * * n<= 2^21
 ******/
typedef long long LL;
const int mod = 119 << 23 | 1;
const int G = 3;
LL wn[30];
void getwn(){ //  千万不要忘记
    for (int i = 0; i <= 20; i++)
     wn[i] = qpow(G, (mod - 1) / (1 << i));
}
void ntt(LL y[], int len, int on=1){
    for (int i = 1, j = (len>>1); i < len - 1;
i++){
        if (i < j) swap(y[i], y[j]);
        int k = len / 2;
        while (j >= k) j -= k, k >>= 1;
        if (j < k) j += k;
    }
    for (int h = 2, id = 1; h <= len; h <<= 1,
id++){
        for (int j = 0; j < len; j += h){
            LL w = 1;
            for (int k = j; k < j + h / 2; k++){//记
得对原数组取模
                LL u = y[k], t = w * y[k + h / 2] %
mod;
                y[k] = (u + t) % mod, y[k + h / 2]
= ((u - t) + 2*mod) % mod;
                w = w * wn[id] % mod;
            }
        }
    }
    if (on == -1){
        //  原本的除法要用逆元
        LL inv = qpow(len, mod - 2);
        for (int i = 1; i < len / 2; i++)
swap(y[i], y[len - i]);
        for (int i = 0; i < len; i++) y[i] = y[i] *
inv % mod;
    }
}
/**
prime number:
  g   r   k              modn
  3   1  16  1<<16|1   = 65537
  3   7  26  7<<26|1   = 469762049
  3 119  23  119<<23|1 = 998244353
  3 479  21  479<<21|1 = 1004535809
 31  15  27  15<<27|1  = 2013265921
  3  17  27  17<<27|1  = 2281701377
  5  27  56  27<<56|1  = 1945555039024054273
  **/
```

```
/*
素数  rr  kk  gg
3   1   1   2
5   1   2   2
17  1   4   3
97  3   5   5
193 3   6   5
257 1   8   3
7681    15  9   17
12289   3   12  11
40961   5   13  3
65537   1   16  3
786433  3   18  10
5767169 11  19  3
7340033 7   20  3
23068673    11  21  3
104857601   25  22  3
167772161   5   25  3
469762049   7   26  3
1004535809  479 21  3
2013265921  15  27  31
2281701377  17  27  3
3221225473  3   30  5
75161927681 35  31  3
77309411329 9   33  7
206158430209    3   36  22
2061584302081   15  37  7
2748779069441   5   39  3
6597069766657   3   41  5
39582418599937  9   42  5
79164837199873  9   43  5
263882790666241 15  44  7
1231453023109121    35  45  3
1337006139375617    19  46  3
3799912185593857    27  47  5
4222124650659841    15  48  19
7881299347898369    7   50  6
31525197391593473   7   52  3
180143985094819841  5   55  6
1945555039024054273 27  56  5
4179340454199820289 29  57  3
*/
```

Combination_pre

```
///简单递推
int c[maxn][maxn];
void init(){
    memset(c,0,sizeof(c));
    c[0][0] = c[1][0] = c[1][1] = 1;
    for(int i = 2; i < maxn; i++){
        c[i][0] = 1;
        for(int j = 1; j <= i; j++)
            c[i][j] = (c[i-1][j] + c[i-1][j-1])%mod;
    }
}

///O(nlogn)预处理阶乘+逆元
const int maxc = 2e6+10;
LL fac[maxc], iv[maxc];
void extgcd(LL aa,LL bb,LL& dd,LL& xx,LL& yy){
    if(!bb){
        dd = aa;
        xx = 1;
        yy = 0;
    }else{
        extgcd(bb, aa%bb, dd, yy, xx);
        yy -= xx*(aa/bb);
    }
}
LL inv(LL aa,LL mm){
    LL dd, xx, yy;
    extgcd(aa, mm, dd, xx, yy);
    return dd==1?(xx+mm)%mm:-1;
}
void init(){
    fac[0] = 1;
    for(int i = 1; i < maxc; i++) fac[i] = (i*fac[i-
1])%mod;
```

```
    iv[maxc-1] = inv(fac[maxc-1], mod);
    for(int i = maxc-2; i >= 0; i--) iv[i] =
iv[i+1]*(i+1)%mod;
}
LL comb(int y, int x){
    if(x < y) swap(x,y);
    return fac[x]*iv[x-y]%mod*iv[y]%mod;
}


//O(n)预处理,mod必须是质数
const int maxc = 1e6+10;
int FAC[maxc], IVF[maxc], IV[maxc];
void init(){
    IV[1] = FAC[0] = FAC[1] = IVF[0] = IVF[1] = 1;
    for(int i = 2; i < maxc; i++){
        FAC[i] = 1LL*FAC[i-1]*i%mod;
        IV[i] = 1LL*(mod-mod/i)*IV[mod%i]%mod;
        IVF[i] = 1LL*IVF[i-1]*IV[i]%mod;
    }
}
LL comb(int x, int y){
    if(x < y) swap(x,y);
    return 1LL*FAC[x]*IVF[x-y]%mod*IVF[y]%mod;
}
```

LinearBasis

```
typedef long long ll;
const int maxl = 64;
ll b[maxl];
void insert(ll x)
{
    for(int i = maxl-1; i >= 0; i--){
        if((x>>i)&1){
            if(b[i])x ^= b[i];
            else{
                b[i] = x;
                for(int j = i-1; j >= 0; j--)if(b[j] &&
((b[i]>>j)&1))b[i] ^= b[j];
                for(int j = i+1; j < maxl;
j++)if((b[j]>>i)&1)b[j] ^= b[i];
                break;
            }
        }
    }
}

ll get_max()
{
    ll ret = 0;
    for(int i = 0; i < maxl; i++)ret ^= b[i];
    return ret;
}
```

Likegcd

```
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
typedef long long LL;
const int Mo = int(1e9) + 7;
const LL Inv2 = 500000004;
const LL Inv6 = 166666668;
/// f(n)=sigma{[0,n], (a*i+b)/c}
/// g(n)=sigma{[0,n], i*((a*i+b)/c)}
/// h(n)=sigma{[0,n], ((a*i+b)/c)^2}
struct Node{
    int a,b,c;

    Node(void){}
    Node(int a,int b,int c) : a(a),b(b),c(c){}
};
LL sqr(LL a){
    a %= Mo;
    return a * a % Mo;
}
LL Calc(int sig,LL R){
    R %= Mo;
    if (sig == 0) return (R + 1) % Mo;
    if (sig == 1) return R * (R + 1) % Mo * Inv2 %
Mo;
```

```
    if (sig == 2) return R * (R + 1) % Mo * (2 *
R % Mo + 1) % Mo * Inv6 % Mo;
}
Node Solve(LL R,LL a,LL b,LL c){
    R %= Mo;
    if (!R) return Node((b / c) % Mo,sqr(b / c),0);
    if (!a)
    {
        LL tmp = (b / c) % Mo;
        return Node(tmp * Calc(0,R) % Mo,tmp *
tmp % Mo * Calc(0,R) % Mo,tmp * Calc(1,R) % Mo);
    }
    LL MAX = (a * R + b) / c - 1;
    if ((a / c) || (b / c))
    {
        Node cr;
        Node tmp = Solve(R,a % c,b % c,c);
        cr.a = (((a / c) % Mo * Calc(1,R) % Mo + (b
/ c) % Mo * Calc(0,R) % Mo) % Mo + tmp.a) % Mo;
        cr.b = (sql(a / c) * Calc(2,R) % Mo + sqr(b
/ c) * Calc(0,R) % Mo) % Mo;
        cr.b = ((cr.b + tmp.b) % Mo + (2 * ((a /
c) % Mo) % Mo * ((b / c) % Mo) % Mo * Calc(1,R))) %
Mo;
        cr.b = ((cr.b + 2 * ((a / c) % Mo) % Mo *
tmp.c % Mo) % Mo + 2 * ((b / c) % Mo) % Mo *
tmp.a % Mo) % Mo;
        cr.c = (((a / c) % Mo * Calc(2,R) % Mo +
((b / c) % Mo) * Calc(1,R) % Mo) % Mo + tmp.c) %
Mo;
        return cr;
    }
    Node cr;
    Node tmp = Solve(MAX,c,c - b - 1,a);
    MAX %= Mo;
    cr.a = ((R) * (MAX + 1) % Mo - tmp.a + Mo) %
Mo;
    cr.b = (2 * (R % Mo * Calc(1,MAX) % Mo - tmp.c
+ Mo) % Mo + cr.a) % Mo;
    cr.c = Inv2 * (((R % Mo * (R + 1) % Mo * (MAX +
1) % Mo - tmp.b + Mo) % Mo - tmp.a + Mo) % Mo) %
Mo;
    return cr;
}
int main(){

freopen("task.in","r",stdin),freopen("task.out","w"
,stdout);
    int A,B,C,L,R;
    scanf("%d%d%d%d%d", &A, &B, &C, &L, &R);
    printf("%d\n", (Solve(R,A,C,B).c - Solve(L -
1,A,C,B).c + Mo) % Mo);
    return 0;
}

/******************只有 f 的******/
LL f(LL a, LL b, LL c, LL n){
    if(!a) return (b/c)*(n+1);
    if(a>=c||b>=c) return
f(a%c,b%c,c,n)+(a/c)*n*(n+1)/2+(b/c)*(n+1);
    LL m=(a*n+b)/c, v=f(c,c-b-1,a,m-1);
    return n*m-v;
}
```

Min_25_sieve

设质数集合$P=\{p_1,p_2,\ldots,p_t\}$，$p_t^2\le n$，$minp(x)$表示x的最小素因子。
设$g(m,j)=\sum_{i=1}^{m}[i\in P\vee minp(i)>p_j]f'(i)$，显然对于$p_j^2>m$的$j$，$g(m,j)$都是相同的，只关心$p_j^2\le m$的情况。
我们现在知道$g(m,0)$，要求$g(m,t)$。不难推出转移：

$$g(m,j)=g(m,j-1)-f'(p_j)\cdot(g(\lfloor\frac{m}{p_j}\rfloor,j-1)-g(p_{j-1},j-1))$$

筛所有数的函数值

类似的我们设$s(m,j)=\sum_{i=1}^{m}[i\in P\vee minp(i)\geq p_j]f(i)$。我们已知$s(m,t+1)=g(m,t)$要求$s(m,1)$。转移是:

$$s(m,j)=s(m,j+1)+\sum_{e\geq 1,p_j^{e+1}\leq m} f(p_j^e)\cdot(s(\lfloor\frac{m}{p_j^e}\rfloor,j+1)-s(p_j,j+1))+f(p_j^{e+1})$$

一种更快的做法

假设我们只需要$s(n,1)$而不需要第一维其它取值的答案，我们有一种常数更小的做法。
我们新设$s(m,j)=\sum_{i=1}^{m}[minp(i)\geq p_j]f(i)$

$$s(m,j)=g(m,t)-g(p_{j-1},t)+\sum_{k=j}^{t}\sum_{e\geq 1,p_k^{e+1}\leq m} f(p_k^e)\cdot s(\lfloor\frac{m}{p_k^e}\rfloor,j+1)+f(p_k^{e+1})$$

```
const LL mod = 1e9+7;
const int maxn = 1e6+10;
LL p[maxn], pnum, nop[maxn];
void prime_sieve(){
    for(int i = 2; i < maxn; i++){
        if(!nop[i]) p[++pnum] = i;
        for(int j = 1; p[j]*i < maxn; j++){
            nop[p[j]*i] = 1;
            if(i%p[j] == 0) break;
        }
    }
}
LL qpow(LL x, LL k){
    LL res = 1;
    while(k > 0){
        if(k&1) res = res*x%mod;
        x = x*x%mod;
        k >>= 1;
    }
    return res;
}
LL g[2][maxn], h[2][maxn], nn, n, iv2;
inline LL get(LL func[2][maxn], LL idx){
    if(idx <= nn) return func[0][idx];
    return func[1][n/idx];
}
///把所有数看成质数的函数值，根据函数修改
inline LL ff1(LL x){ return x;}
inline LL ff2(LL x){ return 1;}

inline LL sum1(LL x){ x %= mod; return
x*(x+1)%mod*iv2%mod;}
inline LL sum2(LL x){ return x%mod;}
inline void _add(LL& a, LL b){a = (a+b)>=mod?a+b-
mod:a+b;}
inline void _sub(LL& a, LL b){a = (a-b)<0?a-
b+mod:a-b;}
void calc(){
    ///把所有数看成质数的函数前缀和，根据函数修改
    for(int i = 1; i <= nn; i++){
        g[0][i] = sum1(i)-1, g[1][i] = sum1(n/i)-1;
        h[0][i] = sum2(i)-1, h[1][i] = sum2(n/i)-1;
    }

    for(int j = 1; j <= pnum; j++){
        for(int i = 1; i <= nn && n >=
1LL*i*p[j]*p[j]; i++){
            _sub(g[1][i], ff1(p[j])*(get(g,
n/i/p[j])-get(g,p[j-1]))%mod);
            _sub(h[1][i], ff2(p[j])*(get(h,
n/i/p[j])-get(h,p[j-1]))%mod);
        }
        for(int i = nn; i >= 1 && i >=
1LL*p[j]*p[j]; i--){
            _sub(g[0][i], ff1(p[j])*(get(g,
i/p[j])-get(g,p[j-1]))%mod);
            _sub(h[0][i], ff2(p[j])*(get(h,
i/p[j])-get(h,p[j-1]))%mod);
        }
    }
```

```
    ///如果分成了多个函数分别求，在这里合并
    for(int i = 1; i <= nn; i++){
        _sub(g[0][i], h[0][i]); _add(g[0][i],2);
        _sub(g[1][i], h[1][i]); _add(g[1][i],2);
        if(i < 2) _add(g[0][i], -2);
    }
}

///f(p^c)的值，根据函数修改
inline LL f(LL prime, LL k){ return prime ^ k;}
LL ask(LL a, int b){
    if(a < p[b]) return 0;
    LL ret = get(g,a)-get(g,p[b]-1);
    if(n < 1LL*p[b]*p[b]) return ret;
    for(int j = b; j<=pnum && 1LL*p[j]*p[j]<=a;
j++){
        for(LL pp = p[j], k=1; p[j]*pp <= a; k++,
pp *= p[j]){
            _add(ret, ask(a/pp,j+1)*f(p[j],k)%mod);
            _add(ret, f(p[j],k+1));
        }
    }
    return ret;
}

int main(){
    prime_sieve();
    cin >> n;
    nn = (LL)sqrt(n);
    iv2 = (mod+1)/2;
    calc();
    cout << (ask(n, 1)+1+mod)%mod << endl;
    return 0;
}
```

Euler's Power

```
int bj;
//bj 用于标记乘法是否取过模
inline int work(int a,int b,int m)
{
    if(1ll*a*b>=m)bj=1;
    return 1ll*a*b%m;
}

int qpow(int a,int b,int m)
{
    int re;
    for(re=1;b;b>>=1,a=work(a,a,m))
        if(b&1)re=work(re,a,m);
    return re;
}

int euler(int v[],int l,int r)
{
    bj=0;
    int mx=min(r-l+1,(int)M.size());
    int now=work(v[l+mx-1],1,M[mx-1]);
    for(int i=mx-2;i>=0;i--)
    {
        if(bj)now+=M[i+1];
        bj=0;
        now=qpow(v[l+i],now,M[i]);
    }
    return now;
}

int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
        scanf("%d",&v[i]);
    //将降幂过程模数塞入 M
    int tmp=m;
```

```
    for(M.push_back(tmp);tmp!=1;tmp=getphi(tmp),M.push_back(tmp));

    scanf("%d",&q); //每次询问[l,r]之间的顺序幂乘：a[l]^a[l+1)^a[l+2)…^a[r)
    for(int i=1;i<=q;i++)
    {
        int l,r;
        scanf("%d%d",&l,&r);
        printf("%d\n",euler(v,l,r));
    }

    return 0;
}
```

CRT (with ExGCD)

```
//X % a[i] = r[i] (1<=i<=n)
int exgcd(int a,int b,int &x,int &y)
{
    if(!b)return x=1,y=0,a;
    int d=exgcd(b,a%b,x,y),t=x;
    return x=y,y=t-a/b*y,d;
}

pair<int,int> CRT(int a[],int r[],int n)
{
    int R=0,M=1;
    for(int i=1;i<=n;i++)M*=a[i];
    for(int i=1;i<=n;i++)
    {
        int t=M/a[i],x,y;
        exgcd(t,a[i],x,y);
        R=(r[i]*t%M*x%M+R)%M;
    }
    return make_pair(M,(R+M)%M);
    //解的形式：ans = R (mod M)
    //return (M, R)
}
```

ExCRT

```
//X % a[i] = r[i] (1<=i<=n)
pair<int,int> solve(int a[],int r[],int n)
{
    int M=a[1],R=r[1];
    for(int i=2;i<=n;i++)
    {
        int x,y;
        int g=exgcd(M,a[i],x,y);
        if((r[i]-R)%g)return make_pair(-1,-1);
        int t=a[i]/g;
        x=(((r[i]-R)/g*x)%t+t)%t;
        R+=x*M;
        M=M*a[i]/g;
    }
    return make_pair(M,R);
    //解的形式：ans = R (mod M)
    //return (M, R)
}
```

Search

Alpha-beta

```
// player = 1 表示轮到己方， player = 0 表示轮到对方，己方希望结果最大
// cur_node 表示当前局面(结点)
maxmin(player, cur_node, alpha, beta)
{
    if 达到终结局面
        return 该局面结点的估价值 f
    end
    if player == 1 // 轮到己方走
```

```
        for 每一种走法 do
            new_node = get_next(cur_node) // 遍历当前局面
cur_node 的所有子局面
            val = maxmin(player^1, new_node, alpha,
beta); // 把新产生的局面交给对方，对方返回一个新局面的估价值
            if val > alpha
                alpha = val;
            end
            if alpha > beta
                return alpha;
            end
        end
        return alpha;
    else // 轮到对方走
        for 每一种走法 do
            new_node = get_next(cur_node) // 遍历当前局面
cur_node 的所有子局面
            val = maxmin(player^1, new_node, alpha,
beta); // 把新产生的局面交给对方，对方返回一个新局面的估价值
            if val < beta
                beta = val;
            end
            if alpha > beta
                return beta;
            end
        end
        return beta;
    end
}
```

ExactCover

```
const int maxnode = 1e5+10;
const int maxn = 1e3+10;
struct DLX
{
    int n, m, size;
    int U[maxnode], D[maxnode], L[maxnode], R[maxnode],
col[maxnode], row[maxnode];
    int H[maxn], S[maxn];
    int ansd, ans[maxn];
    void init(int _n, int _m)
    {
        n = _n, m = _m;
        for(int i = 0; i <= m; i++)
            S[i] = 0, U[i] = D[i] = i, L[i] = i - 1,
R[i] = i + 1;
        L[0] = m, R[m] = 0;
        size = m;
        for(int i = 1; i <= n; i++)H[i] = -1;
    }
    void push(int r, int c)
    {
        ++S[col[++size] = c];
        row[size] = r;
        D[size] = D[c];
        U[D[c]] = size;
        U[size] = c;
        D[c] = size;
        if(H[r] < 0)H[r] = L[size] = R[size] = size;
        else{
            R[size] = R[H[r]];
            L[R[H[r]]] = size;
            L[size] = H[r];
            R[H[r]] = size;
        }
    }
    void del(int c)
    {
        L[R[c]] = L[c];
        R[L[c]] = R[c];
        for(int i = D[c]; i != c; i = D[i]){
            for(int j = R[i]; j != i; j = R[j]){
                U[D[j]] = U[j];
                D[U[j]] = D[j];
                --S[col[j]];
            }
        }
    }
    void reback(int c)
    {
        for(int i = U[c]; i != c; i = U[i])
            for(int j = L[i]; j != i; j = L[j])
```

```
                ++S[col[U[D[j]] = D[U[j]] = j]];
        L[R[c]] = R[L[c]] = c;
    }
    bool dancing(int dep)
    {
        if(R[0] == 0){
            ansd = dep;
            return true;
        }
        int c = R[0];
        for(int i = R[0]; i != 0; i = R[i]){
            if(S[i] < S[c])c = i;
        }
        del(c);
        for(int i = D[c]; i != c; i = D[i]){
            ans[dep] = row[i];
            for(int j = R[i]; j != i; j =
R[j])del(col[j]);
            if(dancing(dep + 1))return true;
            for(int j = L[i]; j != i; j =
L[j])reback(col[j]);
        }
        reback(c);
        return false;
    }
}dlx;
```

MultiCover

```
const int maxn = 1e3+10;
const int maxnode = 1e6+10;
const int INF = 0x3f3f3f3f;
struct DLX
{
    int n,m,size;
    int
U[maxnode],D[maxnode],R[maxnode],L[maxnode],Row[maxnode]
,Col[maxnode];
    int H[maxn],S[maxn];
    int ansd;
    void init(int _n,int _m)
    {
        ansd = INF;
        n = _n, m = _m;
        for(int i = 0;i <= m;i++)
        {
            S[i] = 0;
            U[i] = D[i] = i;
            L[i] = i-1;
            R[i] = i+1;
        }
        R[m] = 0; L[0] = m;
        size = m;
        for(int i = 1;i <= n;i++)H[i] = -1;
    }
    void push(int r,int c)
    {
        ++S[Col[++size]=c];
        Row[size] = r;
        D[size] = D[c];
        U[D[c]] = size;
        U[size] = c;
        D[c] = size;
        if(H[r] < 0)H[r] = L[size] = R[size] = size;
        else
        {
            R[size] = R[H[r]];
            L[R[H[r]]] = size;
            L[size] = H[r];
            R[H[r]] = size;
        }
    }
    void del(int c)
    {
        for(int i = D[c];i != c;i = D[i])
            L[R[i]] = L[i], R[L[i]] = R[i];
    }
    void reback(int c)
    {
        for(int i = U[c];i != c;i = U[i])
            L[R[i]] = R[L[i]] = i;
    }
    bool v[maxn];
    int f()
```

```
    {
        int ret = 0;
        for(int c = R[0]; c != 0;c = R[c])v[c] = true;
        for(int c = R[0]; c != 0;c = R[c]){
            if(v[c])
            {
                ret++;
                v[c] = false;
                for(int i = D[c];i != c;i = D[i])
                    for(int j = R[i];j != i;j = R[j])
                        v[Col[j]] = false;
            }
        }
        return ret;
    }
    void dancing(int d)
    {
        if(d + f() >= ansd)return;
        if(R[0] == 0)
        {
            if(d < ansd)ansd = d;
            return;
        }
        int c = R[0];
        for(int i = R[0];i != 0;i = R[i])
            if(S[i] < S[c])
                c = i;
        for(int i = D[c];i != c;i = D[i])
        {
            del(i);
            for(int j = R[i];j != i;j = R[j])del(j), --
S[Col[j]];
            dancing(d+1);
            for(int j = L[i];j != i;j = L[j])reback(j),
++S[Col[j]];
            reback(i);
        }
    }
}dlx;
```

Sudoku

```
#include <cstdio>
#include <cstring>
const int maxn = 1e3+10;
const int maxnode = 5e5+10;
char s[100];
struct DLX
{
    int n, m, size;
    int U[maxnode], D[maxnode], L[maxnode], R[maxnode],
col[maxnode], row[maxnode];
    int H[maxn], S[maxn];
    int ansd, ans[maxn];
    void init(int _n, int _m)
    {
        n = _n;
        m = _m;
        for(int i = 0; i <= m; i++)S[i] = 0, U[i] = D[i]
= i, L[i] = i - 1, R[i] = i + 1;
        L[0] = m, R[m] = 0;
        size = m;
        for(int i = 1; i <= n; i++)H[i] = -1;
    }
    void push(int r, int c)
    {
        ++S[col[++size] = c];
        row[size] = r;
        D[size] = D[c];
        U[D[c]] = size;
        U[size] = c;
        D[c] = size;
        if(H[r] < 0)H[r] = L[size] = R[size] = size;
        else{
            R[size] = R[H[r]];
            L[R[H[r]]] = size;
            L[size] = H[r];
            R[H[r]] = size;
        }
    }
    void del(int c)
    {
```

```
        L[R[c]] = L[c];
        R[L[c]] = R[c];
        for(int i = D[c]; i != c; i = D[i]){
            for(int j = R[i]; j != i; j = R[j]){
                U[D[j]] = U[j];
                D[U[j]] = D[j];
                --S[col[j]];
            }
        }
    }
    void reback(int c)
    {
        for(int i = U[c]; i != c; i = U[i]){
            for(int j = L[i]; j != i; j = L[j]){
                ++S[col[U[D[j]] = D[U[j]] = j]];
            }
        }
        L[R[c]] = R[L[c]] = c;
    }
    bool dancing(int dep)
    {
        if(R[0] == 0){
            for(int i = 0; i < dep; i++)s[(ans[i]-1)/9]
= (ans[i]-1)%9+'1';
            for(int i = 0; i < 81; i++)printf("%c",
s[i]);
            puts("");
            return true;
        }
        int c = R[0];
        for(int i = R[0]; i != 0; i = R[i])if(S[i] <
S[c])c = i;
        del(c);
        for(int i = D[c]; i != c; i = D[i]){
            ans[dep] = row[i];
            for(int j = R[i]; j != i; j =
R[j])del(col[j]);
            if(dancing(dep + 1))return true;
            for(int j = L[i]; j != i; j =
L[j])reback(col[j]);
        }
        reback(c);
        return false;
    }
}dlx;
void place(int &r, int &c1, int &c2, int &c3, int &c4,
int i, int j, int k)
{
    r = (i*9+j)*9+k;
    c1 = i*9+j+1;
    c2 = 9*9+i*9+k;
    c3 = 9*9*2+j*9+k;
    c4 = 9*9*3+((i/3)*3+(j/3))*9+k;
}
void solve()
{
    dlx.init(9*9*9, 9*9*4);
    int r, c1, c2, c3, c4;
    for(int i = 0; i < 9; i++){
        for(int j = 0; j < 9; j++){
            for(int k = 1; k <= 9; k++){
                if(s[i*9+j] == '.' || s[i*9+j]-'0' ==
k){
                    place(r, c1, c2, c3, c4, i, j, k);
                    dlx.push(r, c1);
                    dlx.push(r, c2);
                    dlx.push(r, c3);
                    dlx.push(r, c4);
                }
            }
        }
    }
    dlx.dancing(0);
}
```

String

KMP

```
#include <cstring>
const int maxn = 1e6+10;
char s[maxn];//模式串
char t[maxn];//匹配串
```

```
int nxt[maxn];
int n, m;
void get_next()
{
    int k = -1;
    n = strlen(s);
    nxt[0] = -1;
    for(int i = 1; i < n; i++){
        while(k >= 0 && s[i] != s[k+1])k = nxt[k];
        if(s[i] == s[k+1])k++;
        nxt[i] = k;
    }
    //int cyc = n-1-nxt[n-1];
    //cyc = n%cyc?1:cyc;
}

bool match()
{
    int k = -1;
    m = strlen(t);
    for(int i = 0; i < n; i++){
        while(k >= 0 && t[i] != s[k+1])k = nxt[k];
        if(t[i] == s[k+1])k++;
        if(k == n-1)return 1;
    }
    return 0;
}
```

AhoCorasickAutomaton

```
#include <cstring>
#include <queue>
using namespace std;
typedef long long ll;
const int maxn = 1e6+10;
const int sigma = 26;
char s[maxn];
struct Node
{
    int nxt[maxn][sigma], fail[maxn], last[maxn],
num[maxn], idx;
    int newNode()
{
    int x = ++idx;
    memset(nxt[x], -1, sizeof(nxt[x]));
    fail[x] = last[x] = 0;
    return x;
}
void init()
{
    idx = 0;
    memset(nxt[0], -1, sizeof(nxt[0]));
    fail[0] = last[0] = 0;
}
void insert()
{
    int rt = 0, n = strlen(s);
    for(int i = 0; i < n; i++){
        int u = s[i]-'a';
        if(nxt[rt][u] == -1)nxt[rt][u] = newNode();
        rt = nxt[rt][u];
    }
    last[rt]++; num[rt]++;
}
void get_fail()
{
    queue<int> q;
    int rt = 0;
    fail[rt] = rt;
    for(int i = 0; i < sigma; i++){
        int &u = nxt[rt][i];
        if(u == -1)u = rt;
        else{
            fail[u] = rt;
            q.push(u);
        }
    }
    while(!q.empty()){
        int u = q.front(); q.pop();
        num[u] += num[fail[u]];
        for(int i = 0; i < sigma; i++){
            int &v = nxt[u][i];
            int w = fail[u];
            if(v == -1)v = nxt[w][i];
```

```
                else{
                    fail[v] = nxt[w][i];
                    q.push(v);
                }
            }
        }
    }
    ll get_ans()
    {
        int rt = 0, n = strlen(s);
        ll ret = 0;
        for(int i = 0; i < n; i++){
            int u = s[i]-'a';
            rt = nxt[rt][u];
            //ret += num[rt];//多重匹配计数
            //int tmp = rt;//单次匹配计数
            //while(tmp){
                //if(last[tmp])ret += last[tmp],
last[tmp] = 0;
                //tmp = fail[tmp];
            //}
        }
        return ret;
    }
}ACAM;
```

ExtKMP

```
#include <algorithm>
using namespace std;
const int maxn = 2e6+10;
char s[maxn], t[maxn];
int nxt[maxn], ext[maxn];
void get_next()
{
    int k = 0, n = strlen(t);
    while(k+1 < n && t[k]==t[k+1])k++;
    nxt[1] = k;
    int pos = 1;
    for(int i = 2; i < n; i++){
        if(i+nxt[i-pos] < pos+nxt[pos])nxt[i] = nxt[i-
pos];
        else{
            k = max(0, pos+nxt[pos]-i);
            while(i+k < n && t[k] == t[i+k])k++;
            nxt[i] = k;
            pos = i;
        }
    }
}
void exKMP()
{
    get_next();
    int k = 0, n = strlen(s);
    while(k < n && s[k] == t[k])k++;
    ext[0] = k;
    int pos = 0;
    for(int i = 1; i < n; i++){
        if(i+nxt[i-pos] < pos+ext[pos])ext[i] = nxt[i-
pos];
        else{
            k = max(0, pos+ext[pos]-i);
            while(i+k < n && s[i+k] == t[k])k++;
            ext[i] = k;
            pos = i;
        }
    }
}
```

Manacher

```
#include <cstring>
#include <algorithm>
using namespace std;
const int maxn = 1e6+10;
char s[maxn];//原串
char t[maxn];//新串
int ma[maxn];
int n;//串长
int manacher()
{
    t[0] = '$', t[1] = '#';
    n = strlen(s);
```

```
    for(int i = 0; i < n; i++)t[i*2+2] = s[i], t[i*2+3]
= '#';
    int mx = -1, ctr = -1, ret = 1;
    for(int i = 1; i < 2*n+2; i++){
        if(mx > i)ma[i] = min(ma[2*ctr-i], mx-i);
        else ma[i] = 1;
        while(t[i-ma[i]] == t[i+ma[i]])ma[i]++;
        if(i+ma[i]-1 > mx){
            mx = i+ma[i]-1;
            ctr = i;
        }
        ret = max(ret, ma[i]-1);
    }
    return ret;
}
```

PalindromeAutomaton

```
#include <algorithm>
#include <cstring>
using namespace std;
typedef long long ll;
const int maxn = 1e6+10;
const int sigma = 26;
char s[maxn];
struct Node
{
    int nxt[maxn][sigma];//后继节点，表示存在 c+s+c 回文
    int len[maxn];//该状态对应的串的长度
    int sz[maxn];//该状态对应的串的出现次数
    int fail[maxn];//该状态的失配指针
    int idx, state;
    void init()
    {
        idx = 1, state = 0;
        fail[0] = fail[1] = 1; len[1] = -1;
        memset(nxt[0], 0, sizeof(nxt[0]));
        memset(nxt[1], 0, sizeof(nxt[1]));
    }
    int newNode()
    {
        int p = ++idx;
        memset(nxt[p], 0, sizeof(nxt[p]));
        len[p] = sz[p] = fail[p] = 0;
        return p;
    }
    void insert(int x, int pos)
    {
        int rt = state;
        while(s[pos-len[rt]-1] != s[pos])rt = fail[rt];
        if(!nxt[rt][x]){
            int v = newNode(), q = fail[rt];
            len[v] = len[rt]+2;
            while(s[pos-len[q]-1] != s[pos])q = fail[q];
            fail[v] = nxt[q][x];
            nxt[rt][x] = v;
        }
        state = nxt[rt][x], sz[state]++;
    }
    ll count()
    {
        ll ans = 0;
        for(int i = idx; i > 0; i--){
            sz[fail[i]] += sz[i];
            ans = max(ans, 1LL*len[i]*sz[i]);
        }
        return ans;
    }
}PAM;
```

ShiftAnd

```
#include <cstdio>
#include <bitset>
#include <cstring>
using namespace std;
const int sigma = 26;//字符集大小
const int maxn = 1e6+10;
bitset<maxn> bs[sigma], ans;
char s[sigma], t[maxn];
int n;//匹配串长度
void init()
{
    for(int i = 0; i < sigma; i++)bs[i].reset();
```

```
    for(int i = 0; i < n; i++){
        scanf("%s", s);//第 i 个位置可以匹配的字符集
        int l = strlen(s);
        for(int j = 0; j < l; j++)bs[s[j]-'a'].set(i);
    }
}
bool match()
{
    scanf("%s", t);
    int l = strlen(t);
    bool flg = 0;
    for(int i = 0; i < l; i++){
        ans <<= 1; ans.set(0);
        ans &= bs[t[i]-'a'];
        if(ans[n-1])flg = 1;
    }
    return flg;
}
```

SmallestRepresentation

```
#include <cstring>
#include <algorithm>
using namespace std;
const int maxn = 1e6+10;
char s[maxn];
int n;

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

SuffixAutomaton

```
#include <cstring>
#include <algorithm>
#include <vector>
using namespace std;
#define pb push_back
const int maxn = 1e6+10;
const int sigma = 26;
int ans[maxn];
char s[maxn];
struct Trie
{
    int nxt[maxn][sigma];//后继节点
    int par[maxn];//parent 指针
    int len[maxn];//该状态对应的最长子串的长度
    int right[maxn];//该状态的 right 集合大小
    vector<int> G[maxn];//parent 树
    int lst, idx;
    int newNode()
    {
        int u = ++idx;
        memset(nxt[u], 0, sizeof(nxt[u]));
        par[u] = len[u] = right[u] = 0;
        return u;
    }
    void init()
    {
        for(int i = 0; i <= idx; i++)G[i].clear();
        idx = 0;
        lst = newNode();
    }
    void extend(int c)
    {
        int rt = lst, np = newNode();
        right[np] = 1;
        len[np] = len[rt]+1;
        for(; rt && !nxt[rt][c]; rt = par[rt])nxt[rt][c]
= np;
```

```
        if(!rt)par[np] = 1;
        else{
            int q = nxt[rt][c];
            if(len[q] == len[rt]+1)par[np] = q;
            else{
                int nq = ++idx;
                right[nq] = 0;
                par[nq] = par[q];
                len[nq] = len[rt]+1;
                memcpy(nxt[nq], nxt[q], sizeof(nxt[q]));
                par[np] = par[q] = nq;
                for(; rt && nxt[rt][c] == q; rt =
par[rt])nxt[rt][c] = nq;
            }
        }
        lst = np;
    }
    void dfs(int u)
    {
        for(auto v : G[u]){
            dfs(v);
            right[u] += right[v];
        }
    }
    void solve()
    {
        for(int i = 2; i <= idx; i++)G[par[i]].pb(i);
        dfs(1);
        for(int i = 2; i <= idx; i++)ans[len[i]] =
max(ans[len[i]], right[i]);
    }
}SAM;
```

SuffixArray

```
struct SA{
    int s[maxn], sa[maxn], t[maxn], t2[maxn], c[maxn],
        rk[maxn], height[maxn];
    void build_sa(int m, int n){
        ///s[i] in {1,m-1}, length is n
        ///add 0 at end of string
        s[n++]=0;
        int *x=t, *y=t2;
        for(int i=0;i<m;i++) c[i]=0;
        for(int i=0;i<n;i++) c[x[i]=s[i]]++;
        for(int i=1;i<m;i++) c[i]+=c[i-1];
        for(int i=n-1;~i;i--) sa[--c[x[i]]]=i;
        for(int k=1;k<=n;k<<=1){
            int p=0;
            for(int i=n-k;i<n;i++) y[p++]=i;
            for(int i=0;i<n;i++) if(sa[i]>=k)
                y[p++] = sa[i]-k;
            for(int i=0;i<m;i++) c[i]=0;
            for(int i=0;i<n;i++) c[x[y[i]]]++;
            for(int i=1;i<m;i++) c[i]+=c[i-1];
            for(int i=n-1;~i;i--) sa[--c[x[y[i]]]]=y[i];
            swap(x,y);
            p=1; x[sa[0]]=0;
            for(int i=1;i<n;i++)
                x[sa[i]] = y[sa[i-1]]==y[sa[i]] &&
                    y[sa[i-1]+k]==y[sa[i]+k] ? p-1 :
p++;
            if(p>=n) break;
            m = p;
        }
        int k=0;
        for(int i=0;i<n;i++) rk[sa[i]]=i;
        for(int i=0;i<n;i++){
            if(k) k--;
            if(rk[i]==0) continue;
            int j=sa[rk[i]-1];
            while(s[i+k]==s[j+k]) k++;
            height[rk[i]]=k;
        }
    }
    int dp[maxn][30];
    void initrmq(int n){
        for(int i=1;i<=n;i++) dp[i][0]=height[i];
        for(int j=1;(1<<j)<=n;j++)
            for(int i=1;i+(1<<j)-1<=n;i++)
                dp[i][j]=min(dp[i][j-1],dp[i+(1<<(j-
1))][j-1]);
    }
    int rmq(int l, int r){
```

```
        int k=31-__builtin_clz(r-l+1);
        return min(dp[l][k], dp[r-(1<<k)+1][k]);
    }
    int lcp(int a, int b){
        a = rk[a], b = rk[b];
        if(a>b) swap(a,b);
        return rmq(a+1,b);
    }
};
```

Others

Vimrc

```
syntax on
set sw=4
set ts=4
set softtabstop=4
set smartindent
set expandtab
set nu

map <F8>  :call Debug()<CR>
map <F9>  :call Compile()<CR>
map <F10> :call Run()<CR>

func! Compile()
      exec "w"
      exec "!g++ % -o %< -g -Wall -std=c++11"
endfunc

func! Run()
      exec "!time ./%<"
endfunc

func! Gdb()
      exec "!gdb %<"
endfunc
```

BM

```
//要求给出*线性递推式*的前几项，初项数量要大于递推式项数
#include <cstdio>
#include <vector>
#include <cassert>
using namespace std;
#define sc(x) scanf("%d", &x)
#define rep(i,a,n) for (int i=a;i<n;i++)
#define pb push_back
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
const ll mod=1e9+7;
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0);
for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return
res;}
namespace linear_seq {
    const int maxn=1e3+10;
    ll res[maxn],base[maxn],_c[maxn],_md[maxn];
    vector<int> Md;
    void mul(ll *a,ll *b,int k) {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if (a[i]) rep(j,0,k)
_c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for (int i=k+k-1;i>=k;i--) if (_c[i])
            rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-
k+Md[j]]-_c[i]*_md[Md[j]])%mod;
        rep(i,0,k) a[i]=_c[i];
    }
    int solve(ll n,VI a,VI b) { // a 系数 b 初值
b[n+1]=a[0]*b[n]+...
        ll ans=0,pnt=0;
        int k=SZ(a);
        assert(SZ(a)==SZ(b));
        rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
        Md.clear();
        rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
        rep(i,0,k) res[i]=base[i]=0;
        res[0]=1;
        while ((1ll<<pnt)<=n) pnt++;
        for (int p=pnt;p>=0;p--) {
```

```
            mul(res,res,k);
            if ((n>>p)&1) {
                for (int i=k-1;i>=0;i--)
res[i+1]=res[i];res[0]=0;
                rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-
res[k]*_md[Md[j]])%mod;
            }
        }
        rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
    VI BM(VI s) {
        VI C(1,1),B(1,1);
        int L=0,m=1,b=1;
        rep(n,0,SZ(s)) {
            ll d=0;
            rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
            if (d==0) ++m;
            else if (2*L<=n) {
                VI T=C;
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B))
C[i+m]=(C[i+m]+c*B[i])%mod;
                L=n+1-L; B=T; b=d; m=1;
            } else {
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B))
C[i+m]=(C[i+m]+c*B[i])%mod;
                ++m;
            }
        }
        return C;
    }
    int gao(VI a,ll n) {
        VI c=BM(a);
        c.erase(c.begin());
        rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
        return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
    }
};

int main() {
    int T;
    sc(T);
    while(T--){
        ll n;
        scanf("%lld",&n);

printf("%d\n",linear_seq::gao(VI{2,3,4,6,9,13},n-1));
    }
}
```

BigDecimal.java

```
import java.math.BigDecimal;

public class Main {
    public static void main(String[] args) {
        BigDecimal num1=new BigDecimal("123.45");
        BigDecimal num2=new BigDecimal("4.5");
        //加法
        System.out.println(num1.add(num2));
        //减法
        System.out.println(num1.subtract(num2));
        //乘法
        System.out.println(num1.multiply(num2));
        //除法（在 divide 的时候就设置好要精确的小数位数和舍入模
式）

System.out.println(num1.divide(num2,10,BigDecimal.ROUND_
HALF_DOWN));
        //取绝对值
        System.out.println(num1.abs());
        //取反
        System.out.println(num1.negate());
        //取最大值
        System.out.println(num1.max(num2));
        //取最小值
        System.out.println(num1.min(num2));
        //是否相等
```

```
        System.out.println(num1.equals(num2));
        //判断大小( > 返回 1, < 返回-1)
        System.out.println(num2.compareTo(num1));
    }
}
```

```
public class <b>BigDecimal</b>
extends Number
implements Comparable<BigDecimal>
```

不可变的、任意精度的有符号十进制数。BigDecimal 由任意精度的整数非标度值 和 32 位的整数标度 (scale) 组成。如果为零或正数，则标度是小数点后的位数。如果为负数，则将该数的非标度值乘以 10 的负 scale 次幂。因此，BigDecimal 表示的数值是 (unscaledValue × 10-scale)。

BigDecimal 类提供以下操作：算术、标度操作、舍入、比较、哈希算法和格式转换。toString() 方法提供 BigDecimal 的规范表示形式。

BigDecimal 类使用户能完全控制舍入行为。如果未指定舍入模式，并且无法表示准确结果，则抛出一个异常；否则，通过向该操作提供适当的 MathContext 对象，可以对已选择的精度和舍入模式执行计算。在任何情况下，可以为舍入控制提供八种舍入模式。使用此类（例如，ROUND_HALF_UP）中的整数字段来表示舍入模式已过时；应改为使用 RoundingMode enum（例如，RoundingMode.HALF_UP）的枚举值。

当为 MathContext 对象提供 0 的精度设置（例如，MathContext.UNLIMITED）时，算术运算是准确的，它们是不采用任何 MathContext 对象的算术方法。（这是第 5 版之前的版本支持的唯一行为。）为了计算准确结果，不使用附带 0 精度设置的 MathContext 对象的舍入模式设置，因此与该对象无关。在除法中，准确的商可能是一个无限长的十进制扩展；例如，1 除以 3 所得的商。如果商具有无穷的十进制扩展，但是指定了该操作返回准确结果，则抛出 ArithmeticException。否则，像其他操作那样，返回除法运算的准确结果。

当精度设置不为 0 时，BigDecimal 算法的规则完全符合 ANSI X3.274-1996 和 ANSI X3.274-1996/AM 1-2000（ 7.4 节）中定义的算法的可选操作模式。与上述标准不同，BigDecimal 包括多种舍入模式，它们对于版本 5 以前的 BigDecimal 版本中的除法是强制性的。这些 ANSI 标准和 BigDecimal 规范之间的任何冲突都按照有利于 BigDecimal 的方式进行解决。

由于同一数值可以有不同的表示形式（具有不同的标度），因此运算和舍入的规则必须同时指定数值结果和结果表示形式中所用的标度。

一般情况下，当准确结果（在除法中，可能有无限多位）比返回的数值具有更多位数时，舍入模式和精度设置确定操作如何返回具有有限位数的结果。 首先，MathContext 的 precision 设置指定要返回的总位数；这确定了结果的精度。位数计数从准确结果的最左边的非零数字开始。舍入模式确定丢弃的尾部位数如何影响返回的结果。

对于所有算术运算符，运算的执行方式是，首先计算准确的中间结果，然后，使用选择的舍入模式将其舍入为精度设置（如有必要）指定的位数。如果不返回准确结果，则将丢弃准确结果的某些数位。当舍入增加了返回结果的大小时，前导数字“9”的进位传播可能会创建新的数位。例如，将值 999.9 舍入为三位数字，则在数值上等于一千，表示为 100×101。在这种情况下，新的 "1" 是返回结果的前导数位。

除了逻辑的准确结果外，每种算术运算都有一个表示结果的首选标度。下表列出了每个运算的首选标度。

算术运算结果的首选标度
运算　结果的首选标度
加 max(addend.scale(), augend.scale())
减 max(minuend.scale(), subtrahend.scale())
乘 multiplier.scale() + multiplicand.scale()
除 dividend.scale() - divisor.scale()
这些标度是返回准确算术结果的方法使用的标度；准确相除可能必须使用较大的标度除外，因为准确的结果可能有较多的位数。例如，1/32 得到 0.03125。
舍入之前，逻辑的准确中间结果的标度是该运算的首选标度。如果用 precision 位数无法表示准确的数值结果，则舍入会选择要返回的一组数字，并将该结果的标度从中间结果的标度减小到可以表示实际返回的 precision 位数的最小标度。如果准确结果可以使用最多 precision 个数字表示，则返回具有最接近首选标度的标度的结果表示形式。尤其是，通过

移除结尾零并减少标度，可以用少于 precision 个数字来表示准确的可表示的商。例如，使用 floor 舍入模式将结果舍入为三个数字，
19/100 = 0.19 // integer=19, scale=2
但是
21/110 = 0.190 // integer=190, scale=3

注意，对于加、减和乘，标度的缩减量将等于丢弃的准确结果的数字位置数。如果舍入导致进位传播创建一个新的高位，则当未创建新的数位时，会丢弃该结果的附加数字。

其他方法可能与舍入语义稍微不同。例如，使用指定的算法的 pow 方法得到的结果可能偶尔不同于舍入得到的算术结果，如最后一位有多个单位（ulp）。

可以通过两种类型的操作来处理 BigDecimal 的标度：标度/舍入操作和小数点移动操作。标度/舍入操作（setScale 和 round）返回 BigDecimal，其值近似地（或精确地）等于操作数的值，但是其标度或精度是指定的值；即：它们会增加或减少对其值具有最小影响的存储数的精度。小数点移动操作（movePointLeft 和 movePointRight）返回从操作数创建的 BigDecimal，创建的方法是按指定方向将小数点移动一个指定距离。

为了简洁明了起见，整个 BigDecimal 方法的描述中都使用了伪代码。伪代码表达式 (i + j) 是"其值为 BigDecimal i 加 BigDecimal j 的 BigDecimal"的简写。伪代码表达式 (i == j) 是"当且仅当 BigDecimal i 表示与 BigDecimal j 相同的值时，则为 true"的简写。可以类似地解释其他伪代码表达式。方括号用于表示特定的 BigInteger 和定义 BigDecimal 值的标度对；例如，[19, 2] 表示 BigDecimal 在数值上等于 0.19，标度是 2。

注：如果 BigDecimal 对象用作 SortedMap 中的键或 SortedSet 中的元素，则应特别小心，因为 BigDecimal 的自然排序与 equals 方法不一致。有关更多信息，请参见 Comparable、SortedMap 或 SortedSet。

当为任何输入参数传递 null 对象引用时，此类的所有方法和构造方法都将抛出 NullPointerException。

另请参见：
BigInteger, MathContext, RoundingMode, SortedMap, SortedSet, 序列化表格

字段摘要
static BigDecimal ONE
值为 1，标度为 0。
static int ROUND_CEILING
接近正无穷大的舍入模式。
static int ROUND_DOWN
接近零的舍入模式。
static int ROUND_FLOOR
接近负无穷大的舍入模式。
static int ROUND_HALF_DOWN
向"最接近的"数字舍入，如果与两个相邻数字的距离相等，则为上舍入的舍入模式。
static int ROUND_HALF_EVEN
向"最接近的"数字舍入，如果与两个相邻数字的距离相等，则向相邻的偶数舍入。
static int ROUND_HALF_UP
向"最接近的"数字舍入，如果与两个相邻数字的距离相等，则为向上舍入的舍入模式。
static int ROUND_UNNECESSARY
断言请求的操作具有精确的结果，因此不需要舍入。
static int ROUND_UP
舍入远离零的舍入模式。
static BigDecimal TEN
值为 10，标度为 0。
static BigDecimal ZERO
值为 0，标度为 0。

构造方法摘要
BigDecimal(BigInteger val)
将 BigInteger 转换为 BigDecimal。
BigDecimal(BigInteger unscaledVal, int scale)
将 BigInteger 非标度值和 int 标度转换为 BigDecimal。
BigDecimal(BigInteger unscaledVal, int scale, MathContext mc)

将 BigInteger 非标度值和 int 标度转换为 BigDecimal（根据上下文设置进行舍入）。

BigDecimal(BigInteger val, MathContext mc)

将 BigInteger 转换为 BigDecimal（根据上下文设置进行舍入）。

BigDecimal(char[] in)

将 BigDecimal 的字符数组表示形式转换为 BigDecimal，接受与 BigDecimal(String) 构造方法相同的字符序列。

BigDecimal(char[] in, int offset, int len)

将 BigDecimal 的字符数组表示形式转换为 BigDecimal，接受与 BigDecimal(String) 构造方法相同的字符序列，同时允许指定子数组。

BigDecimal(char[] in, int offset, int len, MathContext mc)

将 BigDecimal 的字符数组表示形式转换为 BigDecimal，接受与 BigDecimal(String) 构造方法相同的字符序列，同时允许指定子数组，并根据上下文设置进行舍入。

BigDecimal(char[] in, MathContext mc)

将 BigDecimal 的字符数组表示形式转换为 BigDecimal，接受与 BigDecimal(String) 构造方法相同的字符序列（根据上下文设置进行舍入）。

BigDecimal(double val)

将 double 转换为 BigDecimal，后者是 double 的二进制浮点值准确的十进制表示形式。

BigDecimal(double val, MathContext mc)

将 double 转换为 BigDecimal（根据上下文设置进行舍入）。

BigDecimal(int val)

将 int 转换为 BigDecimal。

BigDecimal(int val, MathContext mc)

将 int 转换为 BigDecimal（根据上下文设置进行舍入）。

BigDecimal(long val)

将 long 转换为 BigDecimal。

BigDecimal(long val, MathContext mc)

将 long 转换为 BigDecimal（根据上下文设置进行舍入）。

BigDecimal(String val)

将 BigDecimal 的字符串表示形式转换为 BigDecimal。

BigDecimal(String val, MathContext mc)

将 BigDecimal 的字符串表示形式转换为 BigDecimal，接受与 BigDecimal(String) 构造方法相同的字符串（按照上下文设置进行舍入）。

方法摘要

BigDecimal abs()

返回 BigDecimal，其值为此 BigDecimal 的绝对值，其标度为 this.scale()。

BigDecimal abs(MathContext mc)

返回其值为此 BigDecimal 绝对值的 BigDecimal（根据上下文设置进行舍入）。

BigDecimal add(BigDecimal augend)

返回一个 BigDecimal，其值为 (this + augend)，其标度为 max(this.scale(), augend.scale())。

BigDecimal add(BigDecimal augend, MathContext mc)

返回其值为 (this + augend) 的 BigDecimal（根据上下文设置进行舍入）。

byte byteValueExact()

将此 BigDecimal 转换为 byte，以检查丢失的信息。

int compareTo(BigDecimal val)

将此 BigDecimal 与指定的 BigDecimal 比较。

BigDecimal divide(BigDecimal divisor)

返回一个 BigDecimal，其值为 (this / divisor)，其首选标度为 (this.scale() - divisor.scale())；如果无法表示准确的商值（因为它有无穷的十进制扩展），则抛出 ArithmeticException。

BigDecimal divide(BigDecimal divisor, int roundingMode)

返回一个 BigDecimal，其值为 (this / divisor)，其标度为 this.scale()。

BigDecimal divide(BigDecimal divisor, int scale, int roundingMode)

返回一个 BigDecimal，其值为 (this / divisor)，其标度为指定标度。

BigDecimal divide(BigDecimal divisor, int scale, RoundingMode roundingMode)

返回一个 BigDecimal，其值为 (this / divisor)，其标度为指定标度。

BigDecimal divide(BigDecimal divisor, MathContext mc)

返回其值为 (this / divisor) 的 BigDecimal（根据上下文设置进行舍入）。

BigDecimal divide(BigDecimal divisor, RoundingMode roundingMode)

返回一个 BigDecimal，其值为 (this / divisor)，其标度为 this.scale()。

BigDecimal[] divideAndRemainder(BigDecimal divisor)

返回由两个元素组成的 BigDecimal 数组，该数组包含 divideToIntegralValue 的结果，后跟对两个操作数计算所得到的 remainder。

BigDecimal[] divideAndRemainder(BigDecimal divisor, MathContext mc)

返回由两个元素组成的 BigDecimal 数组，该数组包含 divideToIntegralValue 的结果，后跟根据上下文设置对两个操作数进行舍入计算所得到的 remainder 的结果。

BigDecimal divideToIntegralValue(BigDecimal divisor)

返回 BigDecimal，其值为向下舍入所得商值 (this / divisor) 的整数部分。

BigDecimal divideToIntegralValue(BigDecimal divisor, MathContext mc)

返回 BigDecimal，其值为 (this / divisor) 的整数部分。

double doubleValue()

将此 BigDecimal 转换为 double。

boolean equals(Object x)

比较此 BigDecimal 与指定的 Object 的相等性。

float floatValue()

将此 BigDecimal 转换为 float。

int hashCode()

返回此 BigDecimal 的哈希码。

int intValue()

将此 BigDecimal 转换为 int。

int intValueExact()

将此 BigDecimal 转换为 int，以检查丢失的信息。

long longValue()

将此 BigDecimal 转换为 long。

long longValueExact()

将此 BigDecimal 转换为 long，以检查丢失的信息。

BigDecimal max(BigDecimal val)

返回此 BigDecimal 和 val 的最大值。

BigDecimal min(BigDecimal val)

返回此 BigDecimal 和 val 的最小值。

BigDecimal movePointLeft(int n)

返回一个 BigDecimal，它等效于将该值的小数点向左移动 n 位。

BigDecimal movePointRight(int n)

返回一个 BigDecimal，它等效于将该值的小数点向右移动 n 位。

BigDecimal multiply(BigDecimal multiplicand)

返回一个 BigDecimal，其值为 (this × multiplicand)，其标度为 (this.scale() + multiplicand.scale())。

BigDecimal multiply(BigDecimal multiplicand, MathContext mc)

返回其值为 (this × multiplicand) 的 BigDecimal（根据上下文设置进行舍入）。

BigDecimal negate()

返回 BigDecimal，其值为 (-this)，其标度为 this.scale()。

BigDecimal negate(MathContext mc)

返回其值为 (-this) 的 BigDecimal（根据上下文设置进行舍入）。

BigDecimal plus()

返回 BigDecimal，其值为 (+this)，其标度为 this.scale()。

BigDecimal plus(MathContext mc)

返回其值为 (+this) 的 BigDecimal（根据上下文设置进行舍入）。

BigDecimal pow(int n)

返回其值为 (thisn) 的 BigDecimal，准确计算该幂，使其具有无限精度。

BigDecimal pow(int n, MathContext mc)

返回其值为 (thisn) 的 BigDecimal。

int precision()

返回此 BigDecimal 的精度。

BigDecimal remainder(BigDecimal divisor)

返回其值为 (this % divisor) 的 BigDecimal。

BigDecimal remainder(BigDecimal divisor, MathContext mc)

返回其值为 (this % divisor) 的 BigDecimal（根据上下文设置进行舍入）。

BigDecimal round(MathContext mc)

返回根据 MathContext 设置进行舍入后的 BigDecimal。

int scale()

返回此 BigDecimal 的标度。

BigDecimal scaleByPowerOfTen(int n)

返回其数值等于 (this * 10n) 的 BigDecimal。

BigDecimal setScale(int newScale)

返回一个 BigDecimal，其标度为指定值，其值在数值上等于此 BigDecimal 的值。

BigDecimal setScale(int newScale, int roundingMode)

返回一个 BigDecimal，其标度为指定值，其非标度值通过此 BigDecimal 的非标度值乘以或除以十的适当次幂来确定，以维护其总值。

BigDecimal setScale(int newScale, RoundingMode roundingMode)

返回 BigDecimal，其标度为指定值，其非标度值通过此 BigDecimal 的非标度值乘以或除以十的适当次幂来确定，以维护其总值。

short shortValueExact()

将此 BigDecimal 转换为 short，以检查丢失的信息。

int signum()

返回此 BigDecimal 的正负号函数。

BigDecimal stripTrailingZeros()

返回数值上等于此小数，但从该表示形式移除所有尾部零的 BigDecimal。

BigDecimal subtract(BigDecimal subtrahend)

返回一个 BigDecimal，其值为 (this - subtrahend)，其标度为 max(this.scale(), subtrahend.scale())。

BigDecimal subtract(BigDecimal subtrahend, MathContext mc)

返回其值为 (this - subtrahend) 的 BigDecimal（根据上下文设置进行舍入）。

BigInteger toBigInteger()

将此 BigDecimal 转换为 BigInteger。

BigInteger toBigIntegerExact()

将此 BigDecimal 转换为 BigInteger，以检查丢失的信息。

String toEngineeringString()

返回此 BigDecimal 的字符串表示形式，需要指数时，则使用工程计数法。

String toPlainString()

返回不带指数字段的此 BigDecimal 的字符串表示形式。

String toString()

返回此 BigDecimal 的字符串表示形式，如果需要指数，则使用科学记数法。

BigDecimal ulp()

返回此 BigDecimal 的 ulp（最后一位的单位）的大小。

BigInteger unscaledValue()

返回其值为此 BigDecimal 的非标度值 的 BigInteger。

static BigDecimal valueOf(double val)

使用 Double.toString(double) 方法提供的 double 规范的字符串表示形式将 double 转换为 BigDecimal。

static BigDecimal valueOf(long val)

将 long 值转换为具有零标度的 BigDecimal。

static BigDecimal valueOf(long unscaledVal, int scale)

将 long 非标度值和 int 标度转换为 BigDecimal。

BigInteger.java

```
import java.io.*;
import java.math.BigInteger;
import java.util.*;
public class Main {

    public static void main(String[] args) {
//声明和赋值
        Scanner cin=new Scanner(System.in);//在 import java.util.*;包中，
        //实现了大整数输入的方法    cin.nextBigInteger()

        BigInteger big1=BigInteger.ZERO;//0 声明一个大整数并常数值赋值 0
        BigInteger big2=BigInteger.ONE;//1
        BigInteger big3=BigInteger.TEN;//10
```

```
        BigInteger big4 =new BigInteger("1000");//赋除了
0，1，10 之外的常数值的方法一
        BigInteger big5= BigInteger.valueOf(1000);//没有
new 赋除了之外的常数值的方法二
        //BigInteger bi1 =  new BigInteger(55,new
Random());  生成一个个随机的大整数

        BigInteger [] a=new BigInteger [1005];//声明大整数
数组，并赋值
        a[1]=BigInteger.ZERO;
        a[2]=BigInteger.ONE;
        a[3]=a[2];
        a[4]=
BigInteger.valueOf(3);//a[4]=a[2].add(a[2].add(a[2]));
        int n = cin.nextInt();
        System.out.println(a[n]);

//手动输入
        int n1=cin.nextInt();//输入一个 int 型的 n1
        long n2=cin.nextLong();//输入一个 long 的 n2

        BigInteger n3 = cin.nextBigInteger();//声明一个大
数，并手动输入这个值
        BigInteger n4 = cin.nextBigInteger();

//case 处理
        while(cin.hasNext())
        {
            //cin.hasNext()输入结束同 c++里的
while(scanf("%d",&n)!=EOF)
        }
        while(cin.hasNextLong())
        {
            //同上，这个输入的是 long 类型时
        }
        int test = cin.nextInt();
        while(test-- >0)
        {
            //同 c++里的 while(scanf(test--))
        }
        for(int tt=4;tt<=test;tt++)
        {
            //需要记录 case++时候
            System.out.println("Case "+tt+":");//这里不断
开，用\n 会报错？？
            System.out.println(big1+" + "+big2+" =
"+big1.add(big2));
        }
//四则运算及其他
        System.out.println(n3.add(n4));//n3+n4
        System.out.println(n3.subtract(n4));//n3-n4
        System.out.println(n3.multiply(n4));//n3*n4
        System.out.println(n3.divide(n4));//n3/n4（两个大
整数，整除的商
        System.out.println(n3.remainder(n4));//n3%n4（两
个大整数，取余数
        System.out.println(n3.mod(n4));//同上
        System.out.println(n3.gcd(n4));//n3 和 n4 的最大公约
数
        System.out.println(n3.abs());//n3 取绝对值
        System.out.println(n3.negate());//n3 取相反数
        System.out.println(n3.min(n4));//n3
        System.out.println(n3.max(n4));//n3
        System.out.println(n3.abs());//n3 取绝对值
        System.out.println(n3.pow(2));//n3 的 2 次方（指数
        System.out.println(n3.toString(2)); // 转化为 x 的
n 进制;
        System.out.println(n3.compareTo(n4)==0); //x 和 y
进行比较
          if(big1.compareTo(big2) > 0)

              System.out.println("bd1 is greater than
bd2");

          else if(big1.compareTo(big2) == 0)

              System.out.println("bd1 is equal to bd2");
```

```
        else if(big1.compareTo(big2) < 0)

            System.out.println("bd1 is lower than
bd2");

    }
}
```

```
public class <b>BigInteger</b>
extends Number
implements Comparable<BigInteger>
```

不可变的任意精度的整数。所有操作中，都以二进制补码形式表示 BigInteger（如 Java 的基本整数类型）。BigInteger 提供所有 Java 的基本整数操作符的对应物，并提供 java.lang.Math 的所有相关方法。另外，BigInteger 还提供以下运算：模算术、GCD 计算、质数测试、素数生成、位操作以及一些其他操作。

算术运算的语义完全模仿 Java 整数算术运算符的语义，如 The Java Language Specification 中所定义的。例如，以零作为除数的除法抛出 ArithmeticException，而负数除以正数的除法则产生一个负（或零）的余数。Spec 中关于溢出的细节都被忽略了，因为 BigIntegers 所设置的实际大小能适应操作结果的需要。

位移操作的语义扩展了 Java 的位移操作符的语义以允许产生负位移距离。带有负位移距离的右移操作会导致左移操作，反之亦然。忽略无符号的右位移运算符（>>>），因为该操作与由此类提供的"无穷大的词大小"抽象结合使用时毫无意义。

逐位逻辑运算的语义完全模仿 Java 的逐位整数运算符的语义。在执行操作之前，二进制运算符（and、or、xor）对两个操作数中的较短操作数隐式执行符号扩展。

比较操作执行有符号的整数比较，类似于 Java 的关系运算符和相等性运算符执行的比较。

提供的模算术操作用来计算余数、求幂和乘法可逆元。这些方法始终返回非负结果，范围在 0 和 (modulus - 1)（包括）之间。

位操作对其操作数的二进制补码表示形式的单个位进行操作。如有必要，操作数会通过扩展符号来包含指定的位。单一位操作不能产生与正在被操作的 BigInteger 符号不同的 BigInteger，因为它们仅仅影响单个位，并且此类提供的"无穷大词大小"抽象可保证在每个 BigInteger 前存在无穷多的"虚拟符号位"数。

为了简洁明了，在整个 BigInteger 方法的描述中都使用了伪代码。伪代码表达式 (i + j) 是"其值为 BigInteger i 加 BigInteger j 的 BigInteger"的简写。伪代码表达式 (i == j) 是"当且仅当 BigInteger i 表示与 BigInteger j 相同的值时，才为 true"的简写。可以类似地解释其他伪代码表达式。

当为任何输入参数传递 null 对象引用时，此类中的所有方法和构造方法都将抛出 NullPointerException。

从以下版本开始：
JDK1.1
另请参见：
BigDecimal, 序列化表格

字段摘要
static BigInteger ONE
BigInteger 的常量 1。
static BigInteger TEN
BigInteger 的常量 10。
static BigInteger ZERO
BigInteger 的常量 0。

构造方法摘要
BigInteger(byte[] val)
将包含 BigInteger 的二进制补码表示形式的 byte 数组转换为 BigInteger。
BigInteger(int signum, byte[] magnitude)
将 BigInteger 的符号-数量表示形式转换为 BigInteger。
BigInteger(int bitLength, int certainty, Random rnd)
构造一个随机生成的正 BigInteger，它可能是一个具有指定 bitLength 的素数。

BigInteger(int numBits, Random rnd)
构造一个随机生成的 BigInteger，它是在 0 到 (2numBits - 1)（包括）范围内均匀分布的值。
BigInteger(String val)
将 BigInteger 的十进制字符串表示形式转换为 BigInteger。
BigInteger(String val, int radix)
将指定基数的 BigInteger 的字符串表示形式转换为 BigInteger。

方法摘要
BigInteger abs()
返回其值是此 BigInteger 的绝对值的 BigInteger。
BigInteger add(BigInteger val)
返回其值为 (this + val) 的 BigInteger。
BigInteger and(BigInteger val)
返回其值为 (this & val) 的 BigInteger。
BigInteger andNot(BigInteger val)
返回其值为 (this & ~val) 的 BigInteger。
int bitCount()
返回此 BigInteger 的二进制补码表示形式中与符号不同的位的数量。
int bitLength()
返回此 BigInteger 的最小的二进制补码表示形式的位数，不包括 符号位。
BigInteger clearBit(int n)
返回其值与清除了指定位的此 BigInteger 等效的 BigInteger。
int compareTo(BigInteger val)
将此 BigInteger 与指定的 BigInteger 进行比较。
BigInteger divide(BigInteger val)
返回其值为 (this / val) 的 BigInteger。
BigInteger[] divideAndRemainder(BigInteger val)
返回包含 (this / val) 后跟 (this % val) 的两个 BigInteger 的数组。
double doubleValue()
将此 BigInteger 转换为 double。
boolean equals(Object x)
比较此 BigInteger 与指定的 Object 的相等性。
BigInteger flipBit(int n)
返回其值与对此 BigInteger 进行指定位翻转后的值等效的 BigInteger。
float floatValue()
将此 BigInteger 转换为 float。
BigInteger gcd(BigInteger val)
返回一个 BigInteger，其值是 abs(this) 和 abs(val) 的最大公约数。
int getLowestSetBit()
返回此 BigInteger 最右端（最低位）1 比特的索引（即从此字节的右端开始到本字节中最右端 1 比特之间的 0 比特的位数）。
int hashCode()
返回此 BigInteger 的哈希码。
int intValue()
将此 BigInteger 转换为 int。
boolean isProbablePrime(int certainty)
如果此 BigInteger 可能为素数，则返回 true，如果它一定为合数，则返回 false。
long longValue()
将此 BigInteger 转换为 long。
BigInteger max(BigInteger val)
返回此 BigInteger 和 val 的最大值。
BigInteger min(BigInteger val)
返回此 BigInteger 和 val 的最小值。
BigInteger mod(BigInteger m)
返回其值为 (this mod m) 的 BigInteger。
BigInteger modInverse(BigInteger m)
返回其值为 (this-1 mod m) 的 BigInteger。
BigInteger modPow(BigInteger exponent, BigInteger m)
返回其值为 (thisexponent mod m) 的 BigInteger。
BigInteger multiply(BigInteger val)
返回其值为 (this * val) 的 BigInteger。
BigInteger negate()
返回其值是 (-this) 的 BigInteger。
BigInteger nextProbablePrime()
返回大于此 BigInteger 的可能为素数的第一个整数。
BigInteger not()

返回其值为 (~this) 的 BigInteger。

BigInteger or(BigInteger val)

返回其值为 (this | val) 的 BigInteger。

BigInteger pow(int exponent)

返回其值为 (thisexponent) 的 BigInteger。

static BigInteger probablePrime(int bitLength, Random rnd)

返回有可能是素数的、具有指定长度的正 BigInteger。

BigInteger remainder(BigInteger val)

返回其值为 (this % val) 的 BigInteger。

BigInteger setBit(int n)

返回其值与设置了指定位的此 BigInteger 等效的 BigInteger。

BigInteger shiftLeft(int n)

返回其值为 (this << n) 的 BigInteger。

BigInteger shiftRight(int n)

返回其值为 (this >> n) 的 BigInteger。

int signum()

返回此 BigInteger 的正负号函数。

BigInteger subtract(BigInteger val)

返回其值为 (this - val) 的 BigInteger。

boolean testBit(int n)

当且仅当设置了指定的位时，返回 true。

byte[] toByteArray()

返回一个 byte 数组，该数组包含此 BigInteger 的二进制补码表示形式。

String toString()

返回此 BigInteger 的十进制字符串表示形式。

String toString(int radix)

返回此 BigInteger 的给定基数的字符串表示形式。

static BigInteger valueOf(long val)

返回其值等于指定 long 的值的 BigInteger。

BigInteger xor(BigInteger val)

返回其值为 (this ^ val) 的 BigInteger。

Main.java

1. 输入输出

格式为：Scanner cin = new Scanner (new BufferedInputStream(System.in));

或者、Scanner cin = new Scanner (System.in);

函数：System.out.print(); System.out.println();

System.out.printf();

System.out.print(); // cout << …;

System.out.println(); // cout << … << endl;

System.out.printf(); // 与 C 中的 printf 用法类似.

例程：

```
import java.io.*;
import java.math.*;
import java.util.*;
import java.text.*;
public class Main{
    public static void main(String[] args)
    {
        Scanner cin = new Scanner (new
        BufferedInputStream(System.in));
        int a; double b; BigInteger c; String st;
        a = cin.nextInt(); b = cin.nextDouble(); c
=cin.nextBigInteger(); d = cin.nextLine();
        // 每种类型都有相应的输入函数.
        a = 12345; b = 1.234567;
        System.out.println(a + " " + b);
        System.out.printf("%d %10.5f\n", a, b);
        // 输入 b 为字宽为 10，右对齐，保留小数点后 5 位，四舍五入.
    }
}
```

2. 字符串处理

java 中字符串 String 是不可以修改的，要修改只能转换为字符数组.

例程：

```
{
    int i;
    Scanner cin = new Scanner (new
    BufferedInputStream(System.in));
    String st = "abcdefg";
    System.out.println(st.charAt(0)); // st.charAt(i)就相当于 st[i].
    char [] ch = st.toCharArray(); // 字符串转换为字符数组.
```

```
    for (i = 0; i < ch.length; i++) ch[i] += 1;
    System.out.println(ch); // 输入为"bcdefgh".
    if (st.startsWith("a")) // 如果字符串以'0'开头.
    {
        st = st.substring(1); // 则从第 1 位开始 copy(开头为
第 0 位).
    }
}
```

3. 排序

函数：Arrays.sort();

例程：

```
Scanner cin = new Scanner (new
BufferedInputStream(System.in));
int n = cin.nextInt();
int a[] = new int [n];
for (int i = 0; i < n; i++) a[i] = cin.nextInt();
Arrays.sort(a);
for (int i = 0; i < n; i++) System.out.print(a[i] + "
");
```

4. 结构体排序

例子：一个结构体有两个元素 String x，int y，排序，如果 x 相等 y 升序，否者 x 升序。

Comparator

强行对某个对象 collection 进行整体排序的比较函数，可以将 Comparator 传递给 Collections.sort 或 Arrays.sort。

接口方法：这里也给出了两种方法。

```
import java.util.*;
class structSort{
    String x;
    int y;
}
class cmp implements Comparator<structSort>{
    public int compare(structSort o1, structSort o2) {
        if(o1.x.compareTo(o2.x) == 0){//这个相当于 c/c++中
strcmp (o1.x , o2,x)
            return o1.y - o2.y;
        }
        return o1.x.compareTo(o2.x);
    }
}
public class Main {
    public static void main(String[] args) {
        Comparator<structSort> comparator = new
Comparator<structSort>(){
            public int compare(structSort o1, structSort
o2) {
                if(o1.x.compareTo(o2.x) == 0){
                    return o1.y - o2.y;
                }
                return o1.x.compareTo(o2.x);
            }
        };
        Scanner cin = new Scanner(System.in);
        int n = cin.nextInt();
        structSort a[] = new structSort[10];
        for (int i = 0; i < n; i++) {
            a[i] = new structSort();
            a[i].x = cin.next();
            a[i].y = cin.nextInt();
        }
        Arrays.sort(a,0,n,comparator);//这个直接使用
Comparator
        Arrays.sort(a,0,n,new cmp());//这个实现
Comparator，就就跟 c++中的 sort 函数调用就差不多了
        for (int i = 0; i < n; i++) {
            System.out.println(a[i].x+" "+a[i].y);
        }
    }
}
```

Comparable 强行对实现它的每个类的对象进行整体排序，实现此接口的对象列表（和数组）可以通过 Collections.sort 或 Arrays.sort 进行自动排序。就是输入完了直接就默认排序了，

接口方法：

```
import java.util.*;
class structSort implements Comparable<structSort>{
    String x;
    int y;
    public int compareTo(structSort o1) {
```

```
        if(this.x.compareTo(o1.x) == 0){
            return this.y - o1.y;
        }
        return this.x.compareTo(o1.x);
    }
}
public class Main {
    public static void main(String[] args) {
        Scanner cin = new Scanner(System.in);
        int n = cin.nextInt();
        structSort a[] = new structSort[10];
        for (int i = 0; i < n; i++) {
            a[i] = new structSort();
            a[i].x = cin.next();
            a[i].y = cin.nextInt();
        }
        Arrays.sort(a,0,n);
        for (int i = 0; i < n; i++) {
            System.out.println(a[i].x+" "+a[i].y);
        }
    }
}
```

Contor

```
#include <cstring>
const int maxn = 13;
int fac[maxn];
int s[maxn];
bool tmp[maxn];
int n;
void init()
{
    fac[0] = 1;
    for(int i = 1; i < maxn; i++)fac[i] = fac[i-1]*i;
}
int cantor()
{
    int num=0;
    for(int i = 0; i < n-1; i++)
    {
        int tmp = 0;
        for(int j = i+1; j<n; j++)
            if(s[j] < s[i])tmp++;
        num += fac[n-i-1]*tmp;
    }
    return num;
}
void _cantor(int x)
{
    memset(tmp, 0, sizeof(tmp));
    for(int i=n-1;i>=0;i--)
    {
        int k = x/fac[i]; x %= fac[i];
        int j = 1;
        for(int sum = 0; sum < k || tmp[j]; j++)
            if(!tmp[j])sum++;
        s[n-1-i]=j;
        tmp[j]=1;
    }
}
```

Input_plug

```
//fread 版本
namespace IO {
    const int MT = 27 * 1024 * 1024;  /// 10MB 请注意输入
数据的大小！！！
    char IO_BUF[MT];
    int IO_PTR, IO_SZ;
    /// 要记得把这一行添加到 main 函数第一行！！！
    void begin() {
        IO_PTR = 0;
        IO_SZ = fread (IO_BUF, 1, MT, stdin);
    }
    template<typename T>
    inline bool scan_d (T & t) {
        while (IO_PTR < IO_SZ && IO_BUF[IO_PTR] != '-'
&& (IO_BUF[IO_PTR] < '0' || IO_BUF[IO_PTR] > '9'))
            IO_PTR ++;
        if (IO_PTR >= IO_SZ) return false;
        bool sgn = false;
        if (IO_BUF[IO_PTR] == '-') sgn = true, IO_PTR
++;
```

```
        for (t = 0; IO_PTR < IO_SZ && '0' <=
IO_BUF[IO_PTR] && IO_BUF[IO_PTR] <= '9'; IO_PTR ++)
            t = t * 10 + IO_BUF[IO_PTR] - '0';
        if (sgn) t = -t;
        return true;
    }
    inline bool scan_s (char s[]) {
        while (IO_PTR < IO_SZ && (IO_BUF[IO_PTR] == ' '
|| IO_BUF[IO_PTR] == '\n') ) IO_PTR ++;
        if (IO_PTR >= IO_SZ) return false;
        int len = 0;
        while (IO_PTR < IO_SZ && IO_BUF[IO_PTR] != ' '
&& IO_BUF[IO_PTR] != '\n')
            s[len ++] = IO_BUF[IO_PTR], IO_PTR ++;
        s[len] = '\0';
        return true;
    }
    template<typename T>
    void print(T x) {
        static char s[33], *s1; s1 = s;
        if (!x) *s1++ = '0';
        if (x < 0) putchar('-'), x = -x;
        while(x) *s1++ = (x % 10 + '0'), x /= 10;
        while(s1-- != s) putchar(*s1);
    }
    template<typename T>
    void println(T x) {
        print(x); putchar('\n');
    }
};
//getchar 版本
inline bool scan_d(int &num){
        char in;bool IsN=false;
        in=getchar();
        if(in==EOF) return false;
        while(in!='-'&&(in<'0'||in>'9')) in=getchar();
        if(in=='-'){ IsN=true;num=0;}
        else num=in-'0';
        while(in=getchar(),in>='0'&&in<='9'){
                num*=10,num+=in-'0';
        }
        if(IsN) num=-num;
        return true;
}
inline bool scan_lf(double &num){
        char in;double Dec=0.1;
        bool IsN=false,IsD=false;
        in=getchar();
        if(in==EOF) return false;
        while(in!='-'&&in!='.'&&(in<'0'||in>'9'))
                in=getchar();
        if(in=='-'){IsN=true;num=0;}
        else if(in=='.'){IsD=true;num=0;}
        else num=in-'0';
        if(!IsD){
                while(in=getchar(),in>='0'&&in<='9'){
                        num*=10;num+=in-'0';}
        }
        if(in!='.'){
                if(IsN) num=-num;
                return true;
        }else{
                while(in=getchar(),in>='0'&&in<='9'){
                        num+=Dec*(in-'0');Dec*=0.1;
                }
        }
        if(IsN) num=-num;
        return true;
}
```

Formulas

海伦公式

$S = \sqrt{p(p-a)(p-b)(b-c)}$

$p = (a+b+c)/2$

a, b, c 为三角形三边长

基姆拉尔森公式：

```
if (m == 1 || m == 2) m += 12, y --;
w = (d + 2*m + 3*(m+1)/5 + y + y/4 - y/100 + y/400) % 7
ans(w) = {星期一，星期二，星期三，星期四，星期五，星期六，星期天}
y/m/d 为年月日
```

威尔逊定理：

(p-1)! ≡ p-1 (mod p)，p 为质数

欧拉定理：

```
a^x % p = {
    if gcd(a, p) == 1:
        a ^ (x%phi(p))
    if gcd(a, p) != 1:
        if x < phi(p):
            a ^ x
        if x >= phi(p):
            a ^ (x%phi(p) + phi(p))
}
```

平面图的欧拉定理：

V − E + F = 2

v 是 G 的顶点数，e 是 G 的边数，f 是 G 的面数