South China University of Technology

SCUT_gugugu

TEMPLATE

Last build at October 31, 2019

Contents

1 Graph Theory

1.1 Shortest Path

1.1.1 Dijkstra

```
typedef long long LL;
const int MAXN = ;
const int MAXM = ;
const LL DINF = ;
typedef pair<LL, int> P;
struct Edge {
    int to, nxt;
    LL w;
}e[MAXM];
int head[MAXN], ecnt;
LL d[MAXN];
priority_queue<P, vector<P>, greater<P> > q;
inline void addEdge(int x, int y, LL w) {
    e[++ecnt] = (Edge) {y, head[x], w}; head[x] = ecnt;
}
void dijkstra(int st, int n) {
    for(int i = 0; i <= n; i++) d[i] = DINF;
    d[st] = 0;
    q.push(make_pair(0, st));
    while(!q.empty()) {
        P x = q.top(); q.pop();
        int u = x.second;
        if(d[u] != x.first) continue;
        for(int i = head[u], v; i; i = e[i].nxt) {
            v = e[i].to;
            if(d[v] > d[u] + e[i].w) {
                d[v] = d[u] + e[i].w;
                q.push(make_pair(d[v], v));
            }
        }
    }
}
```

1.1.2 SPFA

```
struct Edge {
    int to, nxt;
    LL w;
}e[MAXE];
int head[MAXN], ecnt;
LL d[MAXN];
bool exist[MAXN];
queue<int> q;
inline void addEdge(int x, int y, LL w) {
   e[++ecnt] = (Edge) {y, head[x], w}; head[x] = ecnt;
}
void SPFA(int st) {
    memset(d,0x3f,sizeof(d));
    d[st] = 0;
    q.push(st);
    exist[st] = 1;
    while(!q.empty()) {
```

```
        int u = q.front(); q.pop();
        exist[u] = 0;
        for(int i = head[u], v; i; i = e[i].nxt) {
            v = e[i].to;
            if(d[v] > d[u] + e[i].w) {
                d[v] = d[u] + e[i].w;
                //pre[v] = u;
                if(!exist[v]) {
                    q.push(v);
                    exist[v] = 1;
                }
            }
        }
    }
}
```

1.2 Johnson

```
void johnson() {
    //建图中,Edge需要from,w1,w2,去掉w;
    spfa(1);
    for(int u = 1; u <= n; u++)
        for(int i = head[u]; i; i = e[i].nxt)
            e[i].w2 = e[i].w1 + d[e[i].from] - d[e[i].to];
    dijkstra(s,n);
}
```

1.2.1 K Shortest Path (A*)

```
//可重复走同一条边 利用了反向边表示,ecnt初始化为1
//调用dijkstra(ed,n)跑反向图,注意if(i& 1){松弛}
int shrt[MAXN];
LL A_star(int st, int ed, int k, int n) {
    if(d[st] == d[0]) return -1;
    if(st == ed) k++;
    q.push(make_pair(d[st], st));
    while(!q.empty()) {
        P x = q.top(); q.pop();
        int u = x.second;
        LL xd = x.first;
        ++shrt[u];
        if(u == ed) {
            if(shrt[ed] == k) return xd;
        }
        for(int i = head[u], v; i; i = e[i].nxt)
        if(!(i & 1)) {
            v = e[i].to;
            if(shrt[v] < k)q.push(make_pair(xd-d[u]+e[i].w+d[v], v));
        }
    }
    return -1;
}
```

1.2.2 K Shortest Path (Protractable Heap)

```cpp
//可重复走同一条边
typedef double LD;
const int MAXN = ;
const int MAXM = ;
const int MAXLT = MAXM * 20;
const LD DINF = ;
const LD eps = ;

namespace LT{
    int tcnt;
    int ls[MAXLT], rs[MAXLT], dis[MAXLT];
    int to[MAXLT];
    LD val[MAXLT];

    inline int newnode(LD w, int _to) {
        ++tcnt;
        ls[tcnt] = rs[tcnt] = 0; dis[tcnt] = 1;
        val[tcnt] = w; to[tcnt] = _to;
        return tcnt;
    }
    inline int copynode(int id) {
        ++tcnt;
        ls[tcnt] = ls[id]; rs[tcnt] = rs[id]; dis[tcnt] = dis[id];
        val[tcnt] = val[id]; to[tcnt] = to[id];
        return tcnt;
    }
    void push_up(int x) {
        if(dis[ls[x]] < dis[rs[x]]) swap(ls[x], rs[x]);
        dis[x] = dis[rs[x]] + 1;
    }
    int merge(int x, int y) {
        if(!x || !y) return x^y;
        if(val[x] - val[y] > eps) swap(x, y);
        int p = copynode(x);
        rs[p] = merge(rs[p],y);
        push_up(p);
        return p;
    }
}
int rt[MAXN];

typedef pair<LD, int> P;
struct Edge {
    int to, nxt;
    LD w;
}e[MAXM];
int head[MAXN], ecnt;
int stan, sta[MAXN], fa[MAXN];
int vis[MAXN], cov[MAXM];
LD d[MAXN];
priority_queue<P, vector<P>, greater<P> > q;
inline void addEdge(int x, int y, LD w) {
    e[++ecnt] = (Edge) {y, head[x], w}; head[x] = ecnt;
}
void init(int n, int m) {
    ecnt = 1; stan = 0;
    for(int i = 1; i <= n; i++)
        head[i] = cov[i] = fa[i] = 0;
    for(int i = 1; i <= m; i++) {
        int u, v; LD w;
        scanf("%d%d%lf", &u,&v,&w);
```

```
62          addEdge(u, v, w);
63          addEdge(v, u, w);
64      }
65  }
66  void dijkstra(int st, int n) {
67      for(int i = 0; i <= n; i++) {d[i] = DINF; vis[i] = 0;}
68      while(!q.empty()) q.pop();
69      d[st] = 0;q.push(make_pair(0, st));
70      while(!q.empty()) {
71          P x = q.top(); q.pop();
72          int u = x.second;
73          if(vis[u]) continue;
74          vis[u] = 1;
75          for(int i = head[u], v; i; i = e[i].nxt)
76              if(i & 1) {
77                  v = e[i].to;
78                  if(d[v]- (d[u] + e[i].w) > eps) {
79                      d[v] = d[u] + e[i].w;
80                      q.push(make_pair(d[v], v));
81                  }
82              }
83      }
84  }
85  void buildT(int u) {
86      sta[++stan] = u;
87      vis[u] = 1;
88      for(int i = head[u], v; i; i = e[i].nxt)
89      if(i & 1) {
90          v = e[i].to;
91          if(fabs(d[v] - (d[u] + e[i].w)) < eps && !vis[v]) {
92              fa[v] = u; cov[i^1] = 1; buildT(v);
93          }
94      }
95  }
96  void buildH(int st, int n) {
97      buildT(st);
98      for(int i = 2, u, v; i <= ecnt; i += 2) {
99          if(!cov[i]) {
100             u = e[i^1].to; v = e[i].to;
101             if(fabs(d[u] - d[0]) < eps || fabs(d[v] - d[0]) < eps) continue;
102             rt[u] = LT::merge(rt[u], LT::newnode(d[v]+e[i].w-d[u], v));
103         }
104     }
105     for(int i = 2, u; i <= n; i++)
106         if(fa[u = sta[i]]) rt[u] = LT::merge(rt[u], rt[fa[u]]);
107 }
108 //求前k短路径,其和不超过W,问最大k int ans = 0;
109 //求第k短路径,LD ans = 0;
110 void getKth(int st, int ed, LD W) {
111     while(!q.empty()) q.pop();
112     //最短路要记入答案,注意d[st]-W<eps与W-d[st]<eps
113     //!if(d[st] - W > eps) return;
114     //!else{W -= d[st]; ++ans;}
115     //注意st和ed相同
116     //#if(st == ed) k++;
117     //#if(d[st] == d[0]) {ans = -1; return;}
118     //#if(--k == 0) {ans = d[st]; return;}
119
120     int u = rt[st], v;
121     if(u) q.push(make_pair(LT::val[u], u));
122     while (!q.empty()) {
```

```
        u = q.top().second; LD cur = q.top().first;
        q.pop();
        //!if (cur + d[st] - W > eps) break;
        //!else {W -= cur + d[st]; ++ans;}
        //#if (--k == 0) {ans = cur + d[st]; break;}
        v = LT::ls[u];
        if (v) q.push(make_pair(cur - LT::val[u] + LT::val[v], v));
        v = LT::rs[u];
        if (v) q.push(make_pair(cur - LT::val[u] + LT::val[v], v));
        v = rt[LT::to[u]];
        if (v) q.push(make_pair(cur + LT::val[v], v));
    }
}
void sol() {
    int n, m, st = , ed = ;
    //LD W; int k;
    init(n, m);
    dijkstra(ed, n);
    for(int i = 0; i <= n; i++) vis[i] = 0;
    buildH(ed, n);
    getKth(st, ed, W);
    printf("%d\n", ans);
}
```

1.3 Network Flow

1.3.1 ISAP

```
namespace NWF {
    struct Edge{
        int to, nxt;LL f;
    }e[MAXM << 1];
    int S, T, tot;
    int ecnt, head[MAXN], cur[MAXN], pre[MAXN], num[MAXN], dis[MAXN];
    queue<int> q;
    void init(int _S, int _T, int _tot){
        ecnt = 1; S = _S; T = _T; tot = _tot;
        memset(num,  0, (tot + 1) * sizeof(int));
        memset(head, 0, (tot + 1) * sizeof(int));
    }
    inline void addEdge(int u, int v, LL f) {
        e[++ecnt] = (Edge) {v, head[u], f}; head[u] = ecnt;
        e[++ecnt] = (Edge) {u, head[v], 0}; head[v] = ecnt;
    }
    void bfs() {
        memset(dis, 0, (tot + 1) * sizeof(int));
        q.push(T);
        dis[T] = 1;
        while(!q.empty()) {
            int u = q.front(), v; q.pop();
            num[dis[u]]++;
            for(int i = cur[u] = head[u]; i; i = e[i].nxt) {
                if(!dis[v = e[i].to]) {
                    dis[v] = dis[u] + 1;
                    q.push(v);
                }
            }
        }
    }
```

```
    LL augment() {
        LL flow = INF;
        for(int i = S; i != T; i = e[cur[i]].to)
            flow = min(flow, e[cur[i]].f);
        for(int i = S; i != T; i = e[cur[i]].to) {
            e[cur[i]].f -= flow;
            e[cur[i] ^ 1].f += flow;
        }
        return flow;
    }
    LL isap() {
        bfs();
        int u = S, v;
        LL flow = 0;
        while(dis[S] <= tot) {
            if(u == T) {
                flow += augment();
                u = S;
            }
            bool fg = 0;
            for(int i = cur[u]; i; i = e[i].nxt) {
                if(e[i].f && dis[u] > dis[v = e[i].to]) {
                    pre[v] = u;
                    cur[u] = i;
                    u = v;
                    fg = 1;
                    break;
                }
            }
            if(fg) continue;
            if(!--num[dis[u]]) break;
            int maxDis = tot;
            for(int i = head[u]; i; i = e[i].nxt) {
                if(e[i].f && maxDis > dis[v = e[i].to]) {
                    maxDis = dis[v];
                    cur[u] = i;
                }
            }
            num[dis[u] = maxDis + 1]++;
            if(u != S) u = pre[u];
        }
        return flow;
    }
}
```

1.3.2 HLPP

```
namespace NWF{
    struct Edge{
        int to,nxt;LL f;
    }e[MAXM << 1];
    int S, T, tot;
    int ecnt, head[MAXN], dis[MAXN], num[MAXN];
    LL sumf[MAXN];
    queue<int> q;
    list<int> dep[MAXN];
    void init(int _S,int _T,int _tot){
        ecnt = 1;S = _S;T = _T;tot = _tot;
        memset(num,  0, (tot + 1) * sizeof(int));
        memset(head, 0, (tot + 1) * sizeof(int));
```

```
        memset(sumf, 0, (tot + 1) * sizeof(LL));
    }
    void addEdge(int u,int v,LL f){
        e[++ecnt] = (Edge) {v, head[u], f};head[u] = ecnt;
        e[++ecnt] = (Edge) {u, head[v], 0};head[v] = ecnt;
    }
    void bfs(){
        memset(dis, 0, (tot + 1) * sizeof(int));
        q.push(T); dis[T] = 1;
        while(!q.empty()){
            int u=q.front(), v; q.pop();
            for(int i = head[u]; i; i = e[i].nxt)
            if(!dis[v = e[i].to]){
                dis[v] = dis[u] + 1;
                q.push(v);
            }
        }
    }
    LL hlpp(){
        bfs();
        dis[S] = tot + 1;
        for(int i = 1;i <= tot; ++i)num[dis[i]]++;
        for(int i = tot + 1; ~i; --i)dep[i].clear();
        int maxd = dis[S];LL f;
        dep[maxd].push_back(S);sumf[S] = INF;
        for(;;){
            while(maxd && dep[maxd].empty())maxd--;
            if(!maxd)break;
            int u = dep[maxd].back(), v;dep[maxd].pop_back();
            int minDis = tot + 1;
            for(int i = head[u]; i;i = e[i].nxt)
            if(e[i].f){
                if(dis[u] > dis[v = e[i].to]){
                    f = min(sumf[u], e[i].f);
                    e[i].f -= f;e[i^1].f += f;
                    if(sumf[u] != INF) sumf[u] -= f;
                    if(sumf[v] != INF) sumf[v] += f;
                    if(v!=S && v!=T && sumf[v] == f){
                        maxd = max(maxd, dis[v]);
                        dep[dis[v]].push_back(v);
                    }
                    if(!sumf[u])break;
                }else minDis=min(minDis, dis[v] + 1);
            }
            if(sumf[u]){
                if(!--num[dis[u]]){
                    for(int i = dis[u];i <= maxd;++i){
                        while(!dep[i].empty()){
                            --num[i];
                            dis[dep[i].back()] = tot + 1;
                            dep[i].pop_back();
                        }
                    }
                    maxd = dis[u] - 1;dis[u] = tot + 1;
                }else{
                    dis[u] = minDis;
                    if(minDis > tot)continue;
                    num[minDis]++;
                    maxd = max(maxd, minDis);
                    dep[minDis].push_back(u);
                }
```

```
            }
        }
        return sumf[T];
    }
}
```

1.3.3 Dinic

```
namespace NWF {
    struct Edge {
        int to, nxt;LL f;
    } e[MAXM << 1];
    int S, T, tot;
    int ecnt, head[MAXN], cur[MAXN], dis[MAXN];
    queue<int> q;
    void init(int _S, int _T, int _tot){
        ecnt = 1; S = _S; T = _T; tot = _tot;
        memset(head, 0, (tot + 1) * sizeof(int));
    }
    void addEdge(int u, int v, LL f) {
        e[++ecnt] = (Edge) {v, head[u], f}; head[u] = ecnt;
        e[++ecnt] = (Edge) {u, head[v], 0}; head[v] = ecnt;
    }
    bool bfs() {
        memset(dis, 0, (tot + 1) * sizeof(int));
        q.push(S); dis[S] = 1;
        while (!q.empty()) {
            int u = q.front(), v; q.pop();
            for (int i = cur[u] = head[u]; i ; i = e[i].nxt) {
                if (e[i].f && !dis[v = e[i].to]) {
                    q.push(v);
                    dis[v] = dis[u] + 1;
                }
            }
        }
        return dis[T];
    }
    LL dfs(int u, LL maxf) {
        if (u == T) return maxf;
        LL sumf = maxf;
        for (int &i = cur[u]; i; i = e[i].nxt) {
            if (e[i].f && dis[e[i].to] > dis[u]) {
                LL tmpf = dfs(e[i].to, min(sumf, e[i].f));
                e[i].f -= tmpf; e[i ^ 1].f += tmpf;
                sumf -= tmpf;
                if (!sumf) return maxf;
            }
        }
        return maxf - sumf;
    }
    LL dinic() {
        LL ret = 0;
        while (bfs()) ret += dfs(S, INF);
        return ret;
    }
}
```

1.3.4 MCMF

```cpp
namespace NWF{
    struct Edge {
        int to, nxt;LL f, c;
    } e[MAXM << 1];
    int S, T, tot;
    int ecnt, head[MAXN], cur[MAXN];LL dis[MAXN];
    bool exist[MAXN];
    queue<int> q;
    void init(int _S, int _T, int _tot){
        ecnt = 1; S = _S; T = _T; tot = _tot;
        memset(head, 0, (tot + 1) * sizeof(int));
    }
    void addEdge(int u, int v, LL f, LL c) {
        e[++ecnt] = (Edge) {v, head[u], f, c}; head[u] = ecnt;
        e[++ecnt] = (Edge) {u, head[v], 0,-c}; head[v] = ecnt;
    }
    bool spfa() {
        for(int i = 0;i <= tot; ++i){
            dis[i] = INF;exist[i] = cur[i] = 0;
        }
        q.push(S);dis[S] = 0;exist[S] = 1;
        while(!q.empty()) {
            int u = q.front(), v; q.pop();exist[u] = 0;
            for(int i = head[u]; i; i = e[i].nxt) {
                if(e[i].f && dis[v = e[i].to] > dis[u] + e[i].c) {
                    dis[v] = dis[u] + e[i].c;
                    cur[v] = i;
                    if(!exist[v]) {
                        q.push(v);
                        exist[v] = 1;
                    }
                }
            }
        }
        return dis[T] != INF;
    }
    LL mcmf() {
        LL cost = 0;
        while(spfa()) {
            LL flow = INF;
            for(int i = T; i != S; i = e[cur[i] ^ 1].to)
                flow = min(flow, e[cur[i]].f);
            for(int i = T; i != S; i = e[cur[i] ^ 1].to) {
                e[cur[i]].f -= flow;
                e[cur[i] ^ 1].f += flow;
            }
            cost += flow * dis[T];
        }
        return cost;
    }
}
```

1.4 Tree Related

1.4.1 Union Set

```cpp
int fa[MAXN], rnk[MAXN];
int Find(int x) { return x == fa[x] ? x : fa[x] = Find(fa[x]); }
bool same(int x, int y){ return Find(x) == Find(y); }
```

```
void unite(int x, int y)
{
    x = Find(x);
    y = Find(y);
    if(x == y) return;
    if(rnk[x] < rnk[y]) {
        fa[x] = y;
    }
    else {
        fa[y] = x;
        if(rnk[x] == rnk[y]) rnk[x]++;
    }
}
```

1.4.2 Kruskal

```
namespace MST{
    struct Edge{
        int u,v; LL w;
        bool operator < (const Edge& x) const { return w < x.w; }
    }e[MAXM];
    int ecnt, fa[MAXN];
    void addEdge(int u, int v, LL w) {
        e[++ecnt] = (Edge){v, u, w}; //headp[u] = ecnt;
    }
    int Find(int x) { return x == fa[x] ? x : fa[x] = Find(fa[x]); }
    LL kruskal(int n) {
        sort(e + 1, e + ecnt + 1);
        for(int i = 1; i <= n; i++) fa[i] = i;
        LL sum = 0;
        for (int i = 1; i <= ecnt; i++){
            int fu = Find(e[i].u), fv = Find(e[i].v);
            if(fu != fv){
                fa[fu] = fv;
                sum += e[i].w;
            }
        }
        return sum;
    }
}
```

1.4.3 Prim

```
namespace MST {
    struct Edge{
        int to,nxt; LL w;
    }e[MAXM];
    int ecnt, head[MAXN], vis[MAXN]; // pre[MAXN];
    LL dis[MAXN];
    void addEdge(int u, int v, LL w){
        e[++ecnt] = (Edge){v, head[u], w}; head[u] = ecnt;
        e[++ecnt] = (Edge){u, head[v], w}; head[v] = ecnt;
    }
    LL Prim(int n){
        for (int i = 1; i <= n; i++){
            //pre[i] = 0;
            vis[i] = 0;
            dis[i] = INF;
```

```
        }
        vis[1] = 1;
        LL sum = 0;
        for (int i = head[1]; i; i = e[i].nxt)
            dis[e[i].to] = min(dis[e[i].to],e[i].w);
        for (int j = 1; j < n; j++){
            int u; LL minDis = INF;
            for (int i = 1; i <= n; ++i)
                if (!vis[i] && dis[i] < minDis){
                    minDis = dis[i];
                    u = i;
                }
            if (minDis == INF) return -1;
            vis[u] = 1;
            sum += minDis;
            for (int i = head[u], v; i; i = e[i].nxt)
            if (!vis[v = e[i].to] && e[i].w < dis[v]){
                //pre[u] = v;
                dis[v] = e[i].w;
            }
        }
        return sum;
    }
}
```

1.4.4 Spanning Tree Calculation

关联矩阵 B:n*m 的矩阵, 其中 ek=(vi,vj),Bik 和 Bjk 一个为 1 一个为-1, 第 k 列其他元素为 0 度数矩阵 D:n*n 的矩阵, 其中 i!=j 时,D[i][j]=0;i==j 时,D[i][j]=vi 的度邻接矩阵 A:n*n 的矩阵,vi,vj 有边相连, 为 1, 否则为 0 Kirchhoff 矩阵: B*BT = D - A 生成树个数:Kirchhoff 矩阵 n-1 阶主子式的行列式值构造 Kirchhoff 矩阵, 调用 det(n)

```
LL a[MAXN][MAXN];
void getSTC(int n, int m) {
    for(int i = 1; i <= n; i++) {
        for(int j = 1; j <= n; j++)
            a[i][j] = a[j][i] = 0;
    }
    for(int i = 1,u, v; i <= m; i++) {
        scanf("%d%d", &u,&v);
        if(u == v) continue;
        a[u][v] = --a[v][u];
    }
    for(int i = 1; i <= n; i++) {
        int t = 0;
        for(int j = 1; j <= n; j++)
            t += a[i][j];
        a[i][i] = - t;
    }
    LL ans = det(n);
}
```

1.4.5 Minimum Spanning Tree Calculation

```
typedef long long LL;
const int MAXN = ;
const int MAXM = ;
int sum,ans1,ans2=1, Mod=;
```

```
int fa1[MAXN],fa2[MAXN];
bool vis[MAXN];
struct Edge{int v,u,val;}e[MAXM];
bool cmp(Edge A,Edge B){return A.val<B.val;}
int getfa1(int *fa,int x){return fa[x]=fa[x]==x?x:getfa1(fa,fa[x]);}
int getfa2(int *fa,int x){return fa[x]==x?x:getfa2(fa,fa[x]);}
void dfs(int tot,int l,int r){
    if(tot==0){++sum;return;}
    for(int i=l,fx,fy;i<r;++i)
        if(!vis[i]){
            vis[i]=true;
            fx=getfa2(fa2,e[i].u);fy=getfa2(fa2,e[i].v);
            if(fx!=fy){
                fa2[fx]=fy;
                dfs(tot-1,i+1,r);
                fa2[fx]=fx;
            }
            vis[i]=false;
        }
}
void sol(){
    int n,m;
    scanf("%d %d",&n,&m);
    for(int i=1;i<=n;++i)fa1[i]=fa2[i]=i;
    for(int i=1;i<=m;++i)
        scanf("%d %d %d",&e[i].v,&e[i].u,&e[i].val);
    std::sort(e+1,e+m+1,cmp);e[m+1].val=-1;
    for(int i=1,j=1,fx,fy,tot;i<=m;++i){
        for(;e[i].val==e[j].val;++j);
        tot=0;
        for(int k=i;k<j;++k){
            fx=getfa1(fa1,e[k].u);fy=getfa1(fa1,e[k].v);
            if(fx!=fy){++tot;ans1++;fa1[fx]=fy;}
        }
        if(!tot)continue;
        sum=0;dfs(tot,i,j);
        (ans2*=sum)%=Mod;
        for(int k=i;k<j;++k){
            fx=getfa1(fa2,e[k].u);fy=getfa1(fa2,e[k].v);
            if(fx!=fy)fa2[fx]=fy;
        }
    }
    if(ans1!=n-1)puts("0");else printf("%d",ans2);
}
```

1.4.6 Tree Divide and Conquer

```
struct Edge {
    int to, nxt, w;
}e[MAXM];
int head[MAXN], ecnt;
int sz[MAXN];
int d[MAXN], t[5], ans;
bool vis[MAXN];
inline void add_edge(int u, int v, int w) {
    e[++ecnt] = (Edge) {v, head[u], w}; head[u] = ecnt;
    e[++ecnt] = (Edge) {u, head[v], w}; head[v] = ecnt;
}
int getsz(int x, int fa) {
    sz[x] = 1;
```

```
    for(int i = head[x]; i; i = e[i].nxt) {
        int y = e[i].to;
        if(vis[y] || y == fa) continue;
        sz[x] += getsz(y, x);
    }
    return sz[x];
}
int getrt(int x) {
    int tot = getsz(x, 0) >> 1;
    while(1) {
        int u = -1;
        for(int i = head[x]; i; i = e[i].nxt) {
            int y = e[i].to;
            if(vis[y] || sz[y] > sz[x]) continue;
            if(u == -1 || sz[y] > sz[u]) u = y;
        }
        if(~u && sz[u] > tot) x = u;
        else break;
    }
    return x;
}
void getdep(int x, int fa) {
    t[d[x]]++;
    for(int i = head[x]; i; i = e[i].nxt) {
        int y = e[i].to;
        if(vis[y] || y == fa) continue;
        d[y] = (d[x] + e[i].w) % 3;
        getdep(y, x);
    }
}
int cal(int x, int v) {
    t[0] = t[1] = t[2] = 0;
    d[x] = v % 3;
    getdep(x, 0);
    return t[0] * t[0] + t[1] * t[2] * 2;
}
void solve(int x) {
    vis[x] = 1;
    ans += cal(x, 0);
    for(int i = head[x]; i; i = e[i].nxt) {
        int y = e[i].to;
        if(vis[y]) continue;
        ans -= cal(y, e[i].w);
        solve(getrt(y));
    }
}
int main() {
    solve(getrt(1));
}
```

1.5 LCA

1.5.1 Tree Decomposition LCA

```
int sz[MAXN], dep[MAXN], top[MAXN], fa[MAXN], son[MAXN], num[MAXN], totw;
struct Edge {
    int to, nxt;
}e[MAXN << 1];
int head[MAXN], ecnt;
```

```
inline void add_edge(int x, int y) {
    e[++ecnt] = (Edge) {y, head[x]}; head[x] = ecnt;
}
void dfs1(int x) {
    sz[x] = 1; son[x] = 0;
    for(int i = head[x]; i; i = e[i].nxt) {
        int v = e[i].to;
        if(v == fa[x]) continue;
        fa[v] = x;
        dep[v] = dep[x] + 1;
        dfs1(v);
        sz[x] += sz[v];
        if(sz[v] > sz[son[x]]) son[x] = v;
    }
}
void dfs2(int x) {
    B[num[x]] = A[x];
    if(son[x]) {
        top[son[x]] = top[x];
        num[son[x]] = ++totw;
        dfs2(son[x]);
    }
    for(int i = head[x]; i; i = e[i].nxt) {
        int v = e[i].to;
        if(v == fa[x] || v == son[x]) continue;
        top[v] = v;
        num[v] = ++totw;
        dfs2(v);
    }
}
int lca(int u, int v) {
    if(u == v) return u;
    while(top[u] != top[v]) {
        if(dep[top[u]] > dep[top[v]]) swap(u, v);
        v = fa[top[v]];
    }
    if(dep[u] > dep[v]) swap(u, v);
    return u;
}
inline void init() {
    memset(head, 0, sizeof(head)); ecnt = 0;
    fa[1] = 0; dep[1] = 1; top[1] = 1; num[1] = 1; totw = 1;
}
inline void pre() {
    dfs1(1); dfs2(1);
}
```

1.5.2 Tarjan LCA

```
vector< pair<int,int> > G[MAXN],ask[MAXN];
int fa[MAXN], ans[MAXN], vis[MAXN] ,dis[MAXN];
int Find(int x){
    return x == fa[x] ? x : fa[x] = Find(fa[x]);
}
void init(int n){
    memset(ans, 0,sizeof ans);
    memset(vis, 0,sizeof vis);
    for(int i = 0; i <= n; i++){
        G[i].clear();
        ask[i].clear();
```

```
    }
}
void LCA(int u){
    int v;
    fa[u] = u;
    vis[u] = true;
    for(auto it : ask[u])
        if(vis[v = it.first])
            ans[it.second] = dis[u] + dis[v] - 2 * dis[Find(it.first)];
    for(auto it : G[u])
    if(!vis[v = it.first]){
        dis[v] = dis[u] + it.second;
        LCA(v);
        fa[v] = u;
    }
}
```

1.6 Tarjan

1.6.1 SCC

```
namespace SCC{
    vector<int> G[MAXN];
    int dfs_clock, scc_cn, dfn[MAXN], low[MAXN], sccno[MAXN];
    stack<int> S;
    void addEdge(int u, int v) {
        G[u].push_back(v);
    }
    void tarjan(int u) {
        dfn[u] = low[u] = ++dfs_clock;
        S.push(u);
        for(auto v : G[u]) {
            if(!dfn[v]) {
                tarjan(v);
                low[u] = min(low[u], low[v]);
            }else if(!sccno[v]) {
                low[u] = min(low[u], dfn[v]);
            }
        }
        if(dfn[u] == low[u]) {
            scc_cnt++;
            for(;;) {
                int v = S.top(); S.pop();
                sccno[v] = scc_cnt;
                if(v == u) break;
            }
        }
    }
    void findSCC(int n) {
        for(int i = 1; i <= n; i++)
            if(!dfn[i]) tarjan(i);
    }
    void init(int n){
        dfs_clock = scc_cnt = 0;
        for(int i = 0;i <= n;++i){
            dfn[i] = low[i] = sccno[i] = 0;
            G[i].clear();
        }
    }
```

```
}
```

1.6.2 BCC

```
namespace BCC{
    struct Edge {
        int to, nxt;
    }e[MAXM << 1];
    int ecnt, head[MAXN];
    int dfs_clock, dfn[MAXN], low[MAXN];

    int is_vertex[MAXN], vbcc_cnt, vbccno[MAXN];
    vector<int> vbcc[MAXN];
    stack<int> vS;

    int ebcc_cnt, ebccno[MAXN];
    stack<int> eS;

    inline void addEdge(int u, int v) {
        e[++ecnt] = (Edge) {v, head[u]}; head[u] = ecnt;
        e[++ecnt] = (Edge) {u, head[v]}; head[v] = ecnt;
    }
    inline void init(int n) {
        ecnt = 1;
        dfs_clock = 0;
        vbcc_cnt = 0;
        ebcc_cnt = 0;
        for(int i = 1; i <= n; ++i){
            head[i] = dfn[i] = low[i] = 0;
            is_vertex[i] = 0;
            vbccno[i] = 0;
            ebccno[i] = 0;
        }
        while(!vS.empty()) vS.pop();
    }
    //root's edge = -1;
    void tarjan(int u, int edge) {
        dfn[u] = low[u] = ++dfs_clock;
        int ch = 0;
        vS.push(u);
        eS.push(u);
        for(int i = head[u], v; i; i = e[i].nxt) {
            if(!dfn[v = e[i].to]) {
                tarjan(v, i ^ 1);
                low[u] = min(low[u], low[v]);
                if(low[v] >= dfn[u]) {
                    ++ch;
                    if(edge > 0 || ch > 1) is_vertex[u] = 1;
                    vbcc[++vbcc_cnt].clear();
                    vbcc[vbcc_cnt].push_back(u);
                    for(int x;;){
                        x = vS.top();vS.pop();
                        vbcc[vbcc_cnt].push_back(x);
                        vbccno[x] = vbcc_cnt;
                        if(x == v)break;
                    }
                }
                if(low[v] > dfn[u]) {
                // i && i ^ 1 is bridge
                }
```

```
            }
            else if(dfn[v] < dfn[u] && i != edge)
                low[u] = min(low[u], dfn[v]);
        }
        if(dfn[u] == low[u]) {
            ebcc_cnt++;
            for(int v;;) {
                v = eS.top(); eS.pop();
                ebccno[v] = ebcc_cnt;
                if(v == u) break;
            }
        }
    }
    void findBCC(int n){
        for(int i = 1; i <= n; i++)
            if(!dfn[i]) tarjan(i, -1);

        //findBridge
        for(int u = 1; u <= n; u++) {
            for(int i = head[u], v; i; i = e[i].nxt)
            if(ebccno[u] != ebccno[v = e[i].to]) {
                //is bridge
            }
        }
    }
}
```

1.7 Cactus

1.7.1 Circle-Square Tree

```
#include <bits/stdc++.h>
using namespace std;
typedef pair<int, int> P;
const int MAXN = 2e4 + 5;
const int S = 15;
namespace Tree {
    struct Edge {
        int to, nxt, w;
    }e[MAXN << 1];
    int ecnt, head[MAXN];
    int rt, isrt[MAXN], fa[MAXN][S + 3];
    int sz[MAXN];
    inline void addEdge(int u, int v, int w) {
        e[++ecnt] = (Edge) {v, head[u], w}; head[u] = ecnt;
        fa[v][0] = u;
    }
}
int n, m, Q;
namespace BCC {
    struct Edge {
        int to, nxt, w;
    }e[MAXN << 1];
    int ecnt, head[MAXN];
    int dfs_clock, dfn[MAXN], low[MAXN];
    int is_vertex[MAXN], vbcc_cnt, vbccno[MAXN];
    vector<P> vbcc[MAXN];
    stack<P> vs;
    int tag[MAXN];
```

```
29      inline void addEdge(int u, int v, int w) {
30          e[++ecnt] = (Edge) {v, head[u], w}; head[u] = ecnt;
31          e[++ecnt] = (Edge) {u, head[v], w}; head[v] = ecnt;
32      }
33      inline void init(int n) {
34          ecnt = 1;
35          dfs_clock = 0;
36          vbcc_cnt = 0;
37          for(int i = 0; i <= 2 * n; i++){
38              head[i] = dfn[i] = low[i] = 0;
39              vbccno[i] = 0;
40              tag[i] = 0;
41          }
42          while(!vs.empty()) vs.pop();
43      }
44      //root's edge = -1;
45      void tarjan(int u, int edge) {
46          dfn[u] = low[u] = ++dfs_clock;
47          vs.push(P(u, e[edge ^ 1].w));
48          for(int i = head[u], v; i; i = e[i].nxt) {
49              if(!dfn[v = e[i].to]) {
50                  tarjan(v, i ^ 1);
51                  low[u] = min(low[u], low[v]);
52                  if(low[v] >= dfn[u]) {
53                      if(vs.top().first == v) {
54                          Tree::addEdge(u, v, vs.top().second);
55                          vs.pop();
56                          continue;
57                      }
58                      vbcc[++vbcc_cnt].clear();
59                      vbcc[vbcc_cnt].push_back(P(u, 0));
60                      Tree::isrt[u] = 1;
61                      int &sz = Tree::sz[n + vbcc_cnt];
62                      tag[vs.top().first] = n + vbcc_cnt;
63                      //Tree::addEdge(u, rt, 0);
64                      for(P x;;) {
65                          x = vs.top(); vs.pop();
66                          sz += x.second;
67                          //Tree::addEdge(rt, x.first, sz);
68                          vbcc[vbcc_cnt].push_back(x);
69                          vbccno[x.first] = vbcc_cnt;
70                          if(x.first == v) break;
71                      }
72                  }
73              }
74              else if(dfn[v] < dfn[u] && i != edge)
75                  low[u] = min(low[u], dfn[v]);
76          }
77          for(int i = head[u], v; i; i = e[i].nxt) {
78              if(tag[v = e[i].to]) {
79                  int r = tag[v]; Tree::sz[r] += e[i].w;
80                  tag[v] = 0;
81              }
82          }
83      }
84      void findBCC(int n) {
85          for(int i = 1; i <= n; i++)
86              if(!dfn[i]) tarjan(i, -1);
87      }
88  }
89  namespace Tree {
```

```
90      int dis[MAXN], dep[MAXN], len[MAXN];
91      inline void init(int n) {
92          BCC::init(n);
93          rt = n;
94          ecnt = 1;
95          for(int i = 0; i <= 2 * n; i++) {
96              head[i] = 0;
97              fa[i][0] = isrt[i] = dis[i] = dep[i]  = len[i] = 0;
98          }
99      }
100     void dfs(int x) {
101         for(int i = head[x], y; i; i = e[i].nxt) {
102             if(!dep[y = e[i].to]) {
103                 dep[y] = dep[x] + 1;
104                 dis[y] = dis[x] + e[i].w;
105                 dfs(y);
106             }
107         }
108     }
109     void pre() {
110         for(int k = 1; k <= BCC::vbcc_cnt; k++) {
111             rt++;
112             vector<P> &E = BCC::vbcc[k];
113             addEdge(E[0].first, rt, 0);
114             int cnt = 0;
115             for(int i = E.size() - 1; i >= 1; i--) {
116                 cnt += E[i].second;
117                 len[E[i].first] = cnt;
118                 addEdge(rt, E[i].first, min(cnt, sz[rt] - cnt));
119             }
120         }
121         for(int k = 1; k <= S; k++) {
122             for(int i = 1; i <= rt; i++) {
123                 fa[i][k] = fa[fa[i][k - 1]][k - 1];
124             }
125         }
126         dep[1] = 1;
127         dfs(1);
128     }
129     int up(int x, int d) {
130         for(int i = S; i >= 0; i--) {
131             if(dep[fa[x][i]] >= d) x = fa[x][i];
132         }
133         return x;
134     }
135     int lca(int u, int v) {
136         if(dep[u] > dep[v]) swap(u, v);
137         v = up(v, dep[u]);
138         if(u == v) return u;
139         for(int i = S; i >= 0; i--) {
140             if(fa[u][i] != fa[v][i]) {
141                 u = fa[u][i], v = fa[v][i];
142             }
143         }
144         return fa[u][0];
145     }
146     int query(int u, int v) {
147         int l = lca(u, v);
148         if(l <= n) return dis[u] + dis[v] - 2 * dis[l];
149         int x = up(u, dep[l] + 1), y = up(v, dep[l] + 1);
150         int res = dis[u] - dis[x] + dis[v] - dis[y];
```

```
        int tmp = abs(len[x] - len[y]);
        return res + min(tmp, sz[l] - tmp);
    }
}

int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
    using namespace Tree;
    cin >> n >> m >> Q;
    init(n);
    for(int i = 1, u, v, w; i <= m; i++) {
        cin >> u >> v >> w;
        BCC::addEdge(u, v, w);
    }
    BCC::findBCC(n);
    pre();
    int u, v;
    while(Q--) {
        cin >> u >> v;
        cout << query(u, v) << endl;
    }
    return 0;
}
```

2 Data Structures

2.1 Basic Structures

2.1.1 RMQ

```
struct RMQ {
    int d[MAXN][S + 2];
    inline void init(int *a, int n) {
        for(int i = 1; i <= n; i++) d[i][0] = a[i];
        for(int k = 1; (1 << k) <= n; k++)
            for(int i = 1; i + (1 << k) - 1 <= n; i++)
                d[i][k] = min(d[i][k - 1], d[i + (1 << (k - 1))][k - 1]);
    }
    inline int query(int l, int r) {
        if(l > r) swap(l, r);
        int k = 0;
        while((1 << (k + 1)) <= r - l + 1) k++;
        return min(d[l][k], d[r - (1 << k) + 1][k]);
    }
}rmq;
const int MAXM = 2e5 + 5, MAXN = 3e6 + 5, S = 22;
const LL INF = 1e18;
#define belong(x) (x / S + 1)
#define pos(x) (x % S + 1)
int Log[MAXN], sz;
struct RMQ {
    LL a[MAXN];
    LL d[MAXM][S + 2];
    LL pre[MAXM][S + 2], aft[MAXM][S + 2];
    inline void init(int n) {
        sz = n / S + 1;
        Log[0] = -1; for(int i = 1; i <= n; i++) Log[i] = Log[i / 2] + 1;
        for(int i = 1; i <= sz; i++) {
            pre[i][0] = aft[i][S + 1] = INF;
        }
        for(int i = 1; i <= n; i++) {
            pre[belong(i)][pos(i)] = min(pre[belong(i)][pos(i) - 1], a[i]);
        }
        for(int i = n; i >= 1; i--) {
            aft[belong(i)][pos(i)] = min(aft[belong(i)][pos(i) + 1], a[i]);
        }
        for(int i = 1; i <= sz; i++) {
            d[i][0] = aft[i][1];
        }
        for(int k = 1; k <= S; k++)
            for(int i = 1; i + (1 << k) <= sz; i++)
                d[i][k] = min(d[i][k - 1], d[i + (1 << (k - 1))][k - 1]);
    }
    inline LL ask(int l, int r) {
        assert(l <= r);
        LL res = INF;
        if(belong(l) == belong(r)) {
            for(int i = l; i <= r; i++) res = min(res, a[i]);
            return res;
        }
        res = min(aft[belong(l)][pos(l)], pre[belong(r)][pos(r)]);
        int k = Log[belong(r) - belong(l) - 1];
        if(~k) {
```

```
            res = min(res, d[belong(l) + 1][k]);
            res = min(res, d[belong(r) - (1 << k)][k]);
        }
        return res;
    }
}rmq;
```

2.1.2 Divide Blocks

```
int belong[MAXN], l[MAXN], r[MAXN];
int sz, num;
void build(int n) {
    sz = sqrt(n);
    num = n / sz; if(n % sz) num++;
    for(int i = 1; i <= num; i++) {
        l[i] = (i - 1) * sz + 1;
        r[i] = i * sz;
    }
    r[num] = n;
    for(int i = 1; i <= n; i++) {
        belong[i] = (i - 1) / sz + 1;
    }
}
```

2.2 Heap Structures

2.2.1 Leftist Tree

```
const int MAXN = ;
namespace LeftistTree{
    int ls[MAXN], rs[MAXN];
    int dis[MAXN];
    int fg[MAXN], sfa[MAXN], rt[MAXN];//利用rt得到堆根节点
    int val[MAXN];

    void push_down(int x) {};
    void push_up(int x) {
        if(dis[ls[x]] < dis[rs[x]]) swap(ls[x], rs[x]);
        dis[x] = dis[rs[x]] + 1;
    }
    int merge(int x, int y) {
        if(!x || !y) return x^y;
        if(val[x] > val[y] || (val[x] == val[y] && x > y)) swap(x, y);
        push_down(x);
        rs[x]  = merge(rs[x], y);
        push_up(x);
        return x;
    }
    int getSfa(int x) {return sfa[x] == x ? x : sfa[x] = getSfa(sfa[x]);}
    int uni(int x, int y) {//返回合并后的根
        if(!x || !y) return x^y;
        if(fg[x] || fg[y]) return;
        x = getSfa(x); y = getSfa(y);
        if(x == y) return;
        int z = merge(x, y);
        return sfa[x] = sfa[y] = sfa[z] = z;
    }
    void uni2(int a, int b) {
```

```
        //val[a] >>= 1;
        int c = merge(ls[a], rs[a]);
        ls[a] = rs[a] = dis[a] = 0;
        int a1 = merge(c, a);
        //val[b] >>= 1;
        c = merge(ls[b], rs[b]);
        ls[b] = rs[b] = dis[b] = 0;
        int b1 = merge(c, b);
        c = merge(a1, b1);
        sfa[a] = sfa[b] = sfa[c] = c;
        printf("%d\n", val[c])
    }
    int pop(int x) {//返回堆顶值,也可以用于返回根
        if(!x || fg[x]) return -1;
        x = getSfa(x); fg[x] = 1;
        push_down(x);//在删除堆顶时要下传标记
        int y = merge(ls[x], rs[x]);
        sfa[x] = sfa[y] = y;
        return val[x];
    }
    void init(int n) {
        for(int i = 1; i <= n; i++) {
            sfa[i] = i;
            ls[i] = rs[i] = dis[i] = fg[i] = 0;
        }
    }
}
//可持久化版本见k短路
```

2.3 Sequence Structures

2.3.1 Cartesian Tree

```
struct CartesianTree{
    int rt, fa[MAXN], ls[MAXN], rs[MAXN];
    int top, st[MAXN];
    int cnt[MAXN];
    void build(LL *a,int n) {
        top = rt = 0;
        for(int i = 1; i <= n; i++) {
            ls[i] = rs[i] = fa[i] = 0;
            while(top && a[st[top]] > a[i]) ls[i] = st[top--];
            fa[i] = st[top];
            if(ls[i]) fa[ls[i]] = i;
            if(fa[i]) rs[fa[i]] = i; else rt = i;
            st[++top] = i;
        }
    }
    void dfs(int x) {
        cnt[x] = 1;
        if(ls[x]) {dfs(ls[x]); cnt[x] += cnt[ls[x]];}
        if(rs[x]) {dfs(rs[x]); cnt[x] += cnt[rs[x]];}
    }
    LL getAns(LL *a, int n) {
        //dfs(rt);
        //————————
        return res;
    }
}T;
```

2.3.2 TreeArray

```
//树状数组上二分
int BS(int x) {
    int res = 0;
    for (int i = 1 << 18; i; i >>= 1)
        if ((res | i) <= Tn && T[res | i] <= x)
            x -= T[res |= i];
    return res;
}
```

2.3.3 Segment Tree

```
#define Ls(x) (x << 1)

#define Rs(x) (x << 1 | 1)
struct Tree {
    int l, r, lazy;
    LL sum, mx;
}tree[MAXN << 2];
int A[MAXN];
void push_up(int x) {
    tree[x].sum = tree[Ls(x)].sum + tree[Rs(x)].sum;
    tree[x].mx = max(tree[Ls(x)].mx, tree[Rs(x)].mx);
}
void push_down(int x) {
    if(tree[x].lazy) {
        tree[Ls(x)].sum += tree[x].lazy * (tree[Ls(x)].r - tree[Ls(x)].l + 1);
        tree[Rs(x)].sum += tree[x].lazy * (tree[Rs(x)].r - tree[Rs(x)].l + 1);
        tree[Ls(x)].mx += tree[x].lazy;
        tree[Rs(x)].mx += tree[x].lazy;
        tree[Ls(x)].lazy += tree[x].lazy;
        tree[Rs(x)].lazy += tree[x].lazy;
        tree[x].lazy = 0;
    }
}
void build(int x, int L, int R) {
    tree[x].lazy = 0;
    tree[x].l = L; tree[x].r = R;
    if(L == R) {
        tree[x].sum = A[L];
        tree[x].mx = A[L];

        return;
    }
    int mid = (L + R) >> 1;
    build(Ls(x), L, mid);
    build(Rs(x), mid + 1, R);
    push_up(x);
}
void update(int x, int L, int R, LL val) {
    if(tree[x].l >= L && tree[x].r <= R) {
        tree[x].lazy += val;
        tree[x].sum += val * (tree[x].r - tree[x].l + 1);
        tree[x].mx += val;
        return;
    }
    push_down(x);
    int mid = (tree[x].l + tree[x].r) >> 1;
```

```
    if(L <= mid) update(Ls(x), L, R, val);
    if(R > mid) update(Rs(x), L, R, val);
    push_up(x);
}
LL query(int x, int L, int R) {
    if(tree[x].l >= L && tree[x].r <= R)
        return tree[x].sum;
    push_down(x);
    int mid = (tree[x].l + tree[x].r) >> 1;
    LL res = 0;
    if(L <= mid) res += query(Ls(x), L, R);
    if(R > mid) res += query(Rs(x), L, R);

    return res;
}
LL query2(int x, int L, int R) {
    if(tree[x].l >= L && tree[x].r <= R)
        return tree[x].mx;
    push_down(x);
    int mid = (tree[x].l + tree[x].r) >> 1;
    LL res = -INF;
    if(L <= mid) res = max(res, query2(Ls(x), L, R));
    if(R > mid) res = max(res, query2(Rs(x), L, R));
    return res;
}
```

2.3.4 LiChao Tree

```
const double eps = 1e-12;
namespace LiT{
    const int MLIMIT = 40000;
    typedef double LD;
    struct line{LD k,b;int l,r,id;} T[MAXN << 2];
    //inline LD calc(line &a,int pos) {return a.k*vec[pos]+a.b;}
    inline LD calc(line &a,int pos) {return a.k*pos+a.b;}
    inline double cross(line &a,line &b) {
        if(b.k == a.k) return -1e9;
        return (double)(a.b-b.b)/(b.k-a.k);
    }
    void build(int v, int l, int r) {
        T[v].k = 0;T[v].b = -1e18;
        T[v].l = 0;T[v].r = MLIMIT;
        T[v].id = 0;
        if(l == r)return;
        int mid = (l+r)>>1;
        build(v<<1,l,mid);
        build(v<<1|1,mid+1,r);
    }
    void ins(int v,int l,int r, line k) {
        if(k.l <= l && r <= k.r) {
            LD fl = calc(k, l), fr = calc(k, r);
            LD gl = calc(T[v], l), gr = calc(T[v], r);
            if(fl - gl > eps && fr - gr > eps) T[v] = k;
            else if(fl - gl > eps || fr - gr > eps) {
                int mid = (l+r)>>1;
                if(calc(k, mid) - calc(T[v], mid) > eps) swap(k, T[v]);
                //if(vec[mid] - cross(k, T[v]) > eps)
                if(mid - cross(k, T[v]) > eps)
                    ins(v<<1, l, mid, k);else ins(v<<1|1, mid+1, r, k);
            }
```

```
            return;
        }
        int mid=(l+r)>>1;
        if(k.l <= mid) ins(v<<1, l, mid, k);
        if(mid < k.r)  ins(v<<1|1, mid+1, r, k);
    }
    LD ans;int ansid;
    void que(int v, int l, int r, int x) {
        LD tmp = calc(T[v], x);
        if(tmp > ans || (tmp == ans && T[v].id < ansid)) {
            ans = tmp;
            ansid = T[v].id;
        }
        if(l == r) return;
        int mid = (l+r)>>1;
        if(x <= mid) que(v<<1,l,mid,x);else que(v<<1|1,mid+1,r,x);
    }
};
//左闭右闭
```

2.3.5 Splay Tree

```
namespace splay{
    int n, m, sz, rt;
    int val[MAXN], id[MAXN];
    int tr[MAXN][2], size[MAXN], fa[MAXN], rev[MAXN], s[MAXN], lazy[MAXN];
    void push_up(int x) {
        int l = tr[x][0], r = tr[x][1];
        s[x] = max(val[x], max(s[l], s[r]));
        size[x] = size[l] + size[r] + 1;
    }
    void push_down(int x) {
        int l = tr[x][0], r = tr[x][1];
        if(lazy[x]) {
            if(l) {
                lazy[l] += lazy[x];
                s[l] += lazy[x];
                val[l] += lazy[x];
            }
            if(r) {
                lazy[r] += lazy[x];
                s[r] += lazy[x];
                val[r] += lazy[x];
            }
            lazy[x] = 0;
        }
        if(rev[x]) {
            rev[x] = 0;
            rev[l] ^= 1; rev[r] ^= 1;
            swap(tr[x][0], tr[x][1]);
        }
    }
    void rotate(int x, int &k) {
        int y = fa[x];
        int z = fa[y];
        int l, r;
        if(tr[y][0] == x) l = 0;
        else l = 1;
        r = l ^ 1;
        if(y == k) k = x;
```

```
        else {
            if(tr[z][0] == y) tr[z][0] = x;
            else tr[z][1] = x;
        }
        fa[x] = z; fa[y] = x; fa[tr[x][r]] = y;
        tr[y][l] = tr[x][r]; tr[x][r] = y;
        push_up(y); push_up(x);
    }
    void splay(int x, int &k) {
        int y, z;
        while(x != k) {
            y = fa[x];
            z = fa[y];
            if(y != k) {
                if((tr[y][0] == x) ^ (tr[z][0] == y)) rotate(x, k);

                else rotate(y, k);
            }
            rotate(x, k);
        }
    }
    int find(int x, int rank) {
        push_down(x);

        int l = tr[x][0], r = tr[x][1];
        if(size[l] + 1 == rank) return x;
        else if(size[l] >= rank) return find(l, rank);
        else return find(r, rank - size[l] - 1);
    }
    void update(int l, int r, int v) {
        int x = find(rt, l), y = find(rt, r + 2);
        splay(x, rt); splay(y, tr[x][1]);
        int z = tr[y][0];
        lazy[z] += v;
        val[z] += v;
        s[z] += v;
    }
    void reverse(int l, int r) {
        int x = find(rt, l), y = find(rt, r + 2);
        splay(x, rt); splay(y, tr[x][1]);
        int z = tr[y][0];
        rev[z] ^= 1;
    }
    void query(int l, int r) {
        int x = find(rt, l), y = find(rt, r + 2);
        splay(x, rt); splay(y, tr[x][1]);
        int z = tr[y][0];
        printf("%d\n", s[z]);
    }
    void build(int l, int r, int f) {
        if(l > r) return;
        int now = id[l], last = id[f];
        if(l == r) {
            fa[now] = last; size[now] = 1;
            if(l < f) tr[last][0] = now;
            else tr[last][1] = now;
            return;
        }
        int mid = (l + r) >> 1; now = id[mid];
        build(l, mid - 1, mid); build(mid + 1, r, mid);
        fa[now] = last;
```

```
        push_up(now);
        if(mid < f) tr[last][0] = now;
        else tr[last][1] = now;
    }
    void init() {
        s[0] = -INF;
        scanf("%d%d", &n, &m);
        for(int i = 1; i <= n + 2; i++) id[i] = ++sz;
        build(1, n + 2, 0); rt = (n + 3) >> 1;
    }
}
namespace splay{
    //内存回收池见fhq_treap
    int tcnt, root;
    int sz[MAXN];
    int tr[MAXN][2], fa[MAXN];
    int val[MAXN];
    int newnode(int w) {
        ++tcnt;
        sz[tcnt] = 1;
        fa[tcnt] = tr[tcnt][0] = tr[tcnt][1] = 0;
        //val[tcnt] = w;
        return tcnt;
    }
    void push_up(int v) {
        int l = tr[v][0], r = tr[v][1];
        sz[v] = sz[l] + 1 + sz[r];
    }
    void push_down(int v) {
        if(!v) return;
    }
    void init() {
        tcnt = 2;
        tr[root = fa[1] = 2][0] = 1;
        sz[1] = 1; sz[2] = 2;
        //val[1] = -INF; val[2] = INF;//权值平衡树
        //val[1] = val[2] = 0;//位置平衡树
        //1,2为哨兵节点,根据题意也可设置为n+1,n+2或1,n+1
    }
    void rotate(int x) {
        int y = fa[x], z = fa[y];
        push_down(y);push_down(x);
        int lr = tr[y][1] == x;
        if(z) tr[z][tr[z][1]==y] = x;
        fa[x] = z;
        fa[tr[y][lr] = tr[x][lr^1]] = y;
        fa[tr[x][lr^1] = y] = x;
        push_up(y); push_up(x);
    }
    void splay(int x, int k) {
        for(int y, z; (y = fa[x]) != k; rotate(x)) {
            if((z = fa[y]) != k) {
                if((tr[y][0] == x) ^ (tr[z][0] == y))
                    rotate(x); else rotate(y);
            }
        }
        if(!k) root = x;
    }
    int find(int x, int rank) {
        push_down(x);
        int l = tr[x][0], r = tr[x][1];
```

```
161         if(sz[l] + 1 == rank) return x;
162         if(sz[l] >= rank) return find(l, rank);
163         return find(r, rank - sz[l] - 1);
164     }
165     int build(int l, int r) {
166         if(l > r) return 0;
167         if(l == r) {
168             int num;scanf("%d", &num);
169             return newnode(num);
170         }
171         int mid = (l + r) >> 1;
172         int ls = build(l, mid-1);
173         int num; scanf("%d", &num);
174         int v = newnode(num);
175         int rs = build(mid+1, r);
176         if(ls) fa[ls] = v;
177         tr[v][0] = ls;
178         if(rs) fa[rs] = v;
179         tr[v][1] = rs;
180         push_up(v);
181         return v;
182     }
183     void insert(int pos, ...) {
184         int x = find(root, pos+1), y = find(root, pos+2);
185         splay(x, 0); splay(y, x);
186         //int z = newnode(w);//插入一个节点
187         //int z = build(1, n);//插入n个节点
188         fa[tr[y][0] = z] = y;
189         splay(z,0);
190     }
191     void modifyOrQuery(int l, int r, int v) {
192         int x = find(root, l), y = find(root, r + 2);
193         splay(x, 0); splay(y, x);
194         int z = tr[y][0];
195         if(!z) return;
196         //标记对本身无效,处理时将z点重新计算
197         splay(z,0);
198     }
199     void display(int v) {
200         if(!v) return;
201         push_down(v);
202         display(tr[v][0]);
203         if(val[v]) printf("%d ", val[v]);
204         display(tr[v][1]);
205     }
206     /*int findValue(int v) {
207         int res = root;
208         for(int cur = root;cur; res = cur, cur = tr[cur][val[cur] <= v]);
209         return res;
210     }
211     void insert(int w) {
212         int y = findValue(w);
213         int z = newnode(w);
214         fa[tr[y][val[y] <= w] = z] = y;
215         splay(z,0);
216     }*/
217     /*void split(int v) {//splay维护区间[l,r],区间分裂为[l,k-1],[k,k],[k+1,r];
218         //ump查看标号是否出现,mp维护子区间左端点
219         if(ump.find(v) == ump.end()) {
220             auto it = mp.upper_bound(v); --it;
221             int z = it->second;
```

```
            splay(z, 0);
            int pos = sz[tr[z][0]];
            int x = find(root,pos), y = find(root,pos+rc[z]-lc[z]+2);
            splay(x, 0); splay(y, x);
            z = tr[y][0];
            if(lc[z] != v) {
                tr[z][0] = newnode(lc[z],v-1);
                fa[tr[z][0]] = z;
                mp[lc[z]] = tr[z][0];
            }
            if(rc[z] != v) {
                tr[z][1] = newnode(v+1,rc[z]);
                fa[tr[z][1]] = z;
                mp[v+1] = tr[z][1];
            }
            lc[z] = rc[z] = v;
            splay(z,0);
            mp[v] = z;
            ump[v] = z;
        }
    }*/

}
```

2.3.6 Scapegoat Tree

```
struct ScapegoatTree{
    int Tsn; queue<int> q;
    int val[MAXM], ext[MAXM];
    int sz[MAXM], tsz[MAXM];
    int fa[MAXM], tr[MAXM][2];
    int root;
    double alp;
    void init() {
        root = 0;
        alp = 0.7;
    }
    int newnode(int x) {
        if(q.empty()) q.push(++Tsn);
        int tcnt = q.front(); q.pop();
        val[tcnt] = x; ext[tcnt] = 1;
        fa[tcnt] = tr[tcnt][0] = tr[tcnt][1] = 0;
        sz[tcnt] = tsz[tcnt] = 0;
        return tcnt;
    }
    void push_up(int v) {
        sz[v] = ext[v];tsz[v] = 1;
        if(tr[v][0]) {
            sz[v] += sz[tr[v][0]];
            tsz[v] += tsz[tr[v][0]];
        }
        if(tr[v][1]) {
            sz[v] += sz[tr[v][1]];
            tsz[v] += tsz[tr[v][1]];
        }
    }
    bool isBad(int v) {
        return (double(tsz[ tr[v][0] ]) > double(tsz[v]) * alp) ||
            (double(tsz[ tr[v][1] ]) > double(tsz[v]) * alp) ||
            (sz[v] * 2 < tsz[v]);
```

```
    }
    vector<int> vec;
    void rRecycle(int v) {
        if(tr[v][0]) rRecycle(tr[v][0]);
        if(ext[v]) vec.push_back(v); else q.push(v);
        if(tr[v][1]) rRecycle(tr[v][1]);
    }
    int rBuild(int l, int r) {
        int mid = (l + r) >> 1, v = vec[mid];
        tr[v][0] = (l <= mid-1) ? rBuild(l, mid - 1) : 0;
        if(tr[v][0]) fa[tr[v][0]] = v;
        tr[v][1] = (mid+1 <= r) ? rBuild(mid + 1, r) : 0;
        if(tr[v][1]) fa[tr[v][1]] = v;
        push_up(v);
        return v;
    }
    void rebuild(int x) {
        int v = 0;
        for(;x; x= fa[x]) {
            push_up(x);
            if(isBad(x)) v = x;
        }
        if(v && isBad(v)){
            vec.clear();
            int u = fa[v], lr = tr[u][1] == v;
            rRecycle(v);
            if(vec.size()) v = rBuild(0, vec.size() - 1); else v = 0;
            if(u == 0) fa[root = v] = 0;
            else{
                tr[u][lr] = v;
                if(v) fa[v] = u;
            }
        }
    }
    void ins(int x) {
        int p = root, q = root;
        for(;p && val[p] != x; q = p, p = tr[p][ val[p] < x]) ;
        if(!q) {
            p = root = newnode(x);
        }else if(p) {
            ext[p]++;
        }else{
            fa[p = tr[q][val[q] < x] = newnode(x) ] = q;
        }
        rebuild(p);
    }
    void del(int x) {
        int p = root;
        for(;p && val[p] != x; p = tr[p][ val[p] < x]);
        if(p && ext[p]){
            --ext[p];
            rebuild(p);
        }
    }
    int get_rank(int x) {
        int ret = 0;
        for(int p = root;p;) {
            if(val[p] < x) {
                ret += sz[tr[p][0]] + ext[p];
                p = tr[p][1];
            }else p = tr[p][0];
```

```
        }
        return ret + 1;
    }
    int get_Kth(int p, int k) {
        if(sz[tr[p][0]] >= k) return get_Kth(tr[p][0] ,k);
        k -= sz[tr[p][0]];
        if(ext[p] >= k) return val[p];
        k -= ext[p];
        return get_Kth(tr[p][1], k);
    }
    int pre(int x) {
        int id = get_rank(x);
        return get_Kth(root, id - 1);
    }
    int nxt(int x) {
        int id = get_rank(x + 1);
        return get_Kth(root ,id);
    }
    void display(int v) {
        if(tr[v][0]) display(tr[v][0]);
        cerr<<val[v]<<" ";
        if(tr[v][1]) display(tr[v][1]);
    }
}T;
```

2.3.7 FHQ Treap

```
namespace fhq_treap{
    int Tsz; queue<int> q; //内存回收池
    int tcnt, root;
    //int rt[MAXN]; //可持久化时使用rt,维护版本号(int &root), 空间开大
    int sz[MAXN], rnd[MAXN];
    int tr[MAXN][2];
    //int fa[MAXN];//维护fa时除了在pushup更新v节点左右孩子父节点信息还要在split和merge结束时维护root的fa信息,fa[root]=0;
    int val[MAXN], rev[MAXN];
    void init() {
        srand(time(0));
        Tsz = tcnt = root = 0;
    }
    int newnode(int v) {
        if(q.empty()) q.push(++Tsz);
        tcnt = q.front(); q.pop();
        sz[tcnt] = 1;
        rnd[tcnt] = rand();
        tr[tcnt][0] = tr[tcnt][1] = 0;
        //val[tcnt] = v;
        return tcnt;
    }
    /*int copynode(int id) {
        //++tcnt;//获取一个新的节点编号
        sz[tcnt] = sz[id];
        rnd[tcnt] = rnd[id];
        tr[tcnt][0] = tr[id][0];
        tr[tcnt][1] = tr[id][1];
        //val[tcnt] = val[id];
        return tcnt;
    }*/
    void push_up(int v) {
        int l = tr[v][0], r = tr[v][1];
```

```
33          sz[v] = sz[l] + 1 + sz[r];
34      }
35      void push_down(int v) {
36          if(!v) return;
37          int l = tr[v][0], r = tr[v][1];
38          //if(l) ;
39          //if(r) ;
40          //swap时候注意交换tr[v][0]和tr[v][1],而不是l和r;
41      }
42      /* //可持久化在push_down是要新建节点,否者历史版本有可能被多次下传,以rev为例
43      void push_down(int v) {
44          if(!v || !rev[v]) return;
45          int &l = tr[v][0], &r = tr[v][1];
46          if(l) {
47              l = copynode(l);
48              rev[l]^=1;
49              swap(tr[l][0],tr[l][1]);
50          };
51          if(r) {
52              r = copynode(r);
53              rev[r]^=1;
54              swap(tr[r][0],tr[r][1]);
55          };
56          rev[v] = 0;
57      }*/
58      void split(int v,int k,int &x,int &y) {
59          if(!v) {x=y=0;return;}
60          push_down(v);
61          //v = copynode(v);//可持久化时复制节点
62          /*if(k > sz[tr[v][0]]) {
63              x = v;
64              split(tr[v][1], k-sz[tr[v][0]]-1, tr[v][1], y);
65          }else{
66              y = v;
67              split(tr[v][0], k , x ,tr[v][0]);
68          }*/
69          if(val[v] <= k) {
70              x = v;
71              split(tr[v][1], k, tr[v][1], y);
72          }else{
73              y = v;
74              split(tr[v][0], k, x, tr[v][0]);
75          }
76          push_up(v);
77      }
78      int merge(int x,int y) {//x堆所有值均小于y堆
79          if(!x || !y) return x|y;
80          push_down(x); push_down(y);
81          if(rnd[x]<rnd[y]){
82              //x = copynode(x);//可持久化时复制节点,可不写
83              tr[x][1] = merge(tr[x][1],y);
84              push_up(x);
85              return x;
86          }else{
87              //y = copynode(y);//可持久化时复制节点,可不写
88              tr[y][0] = merge(x,tr[y][0]);
89              push_up(y);
90              return y;
91          }
92      }
93      void insert(int k) {
```

```
94          int x,y;
95          split(root,k,x,y);
96          root = merge(merge(x,newnode(k)),y);
97      }
98      void recycle(int v) {//回收一颗treap上所有节点
99          if(!v) return;
100         q.push(v);
101         recycle(tr[v][0]); recycle(tr[v][1]);
102     }
103     void erase(int k) {
104         int x,y,z;
105         split(root,k,x,y);
106         split(x,k-1,x,z);
107         z = merge(tr[z][0],tr[z][1]);
108         root = merge(x,merge(z,y));
109     }
110     void krank(int k) {
111         int x,y;
112         split(root,k-1,x,y);
113         printf("%d\n",sz[x]+1);
114         root = merge(x,y);
115     }
116     int find(int v,int k) {
117         if(sz[tr[v][0]]==k-1) return val[v];
118         if(sz[tr[v][0]]>=k) return find(tr[v][0],k);
119         return find(tr[v][1],k-sz[tr[v][0]]-1);
120     }
121     void pre(int k) {
122         int x,y;
123         split(root,k-1,x,y);
124         printf("%d\n",find(x,sz[x]));
125         root=merge(x,y);
126     }
127     void nxt(int k){
128         int x,y;
129         split(root,k,x,y);
130         printf("%d\n",find(y,1));
131         root=merge(x,y);
132     }
133     void reverse(int l,int r){
134         int x,y,z;
135         split(root, r, x, y);
136         split(x, l-1, x, z);
137         //rev[z] ^= 1;标记对本身无效,处理时将z点重新计算
138         root = merge(merge(x,z),y);
139     }
140     /*int getRank(int S) {
141         int res = sz[tr[S][0]]+1;
142         for(;fa[S];S = fa[S])
143             if(tr[fa[S]][1] == S) res += sz[tr[fa[S]][0]] + 1;
144        return res;
145     }*/
146     void display(int v) {
147         if(!v) return;
148         push_down(v);
149         display(tr[v][0]);
150         printf("%d ",val[v]);
151         display(tr[v][1]);
152     }
153 }
```

```
154 //一种可持久化平衡树的替代(非强制在线),由历史版本向当前版本连边,在dfs遍历中利用权值树状数组,普通平衡树等获取答案
```

2.4 Persistent Data Structures

2.4.1 Chairman Tree

```
1  struct Node {
2      int l, r;
3      LL sum;
4  }t[MAXN * 40];
5  int cnt, n;
6  int rt[MAXN];
7  void update(int pre, int &x, int l, int r, int v) {
8      x = ++cnt; t[x] = t[pre]; t[x].sum++;
9      if(l == r) return;
10     int mid = (l + r) >> 1;
11     if(v <= mid) update(t[pre].l, t[x].l, l, mid, v);
12     else update(t[pre].r, t[x].r, mid + 1, r, v);
13 }
14 int query(int x, int y, int l, int r, int v) {
15     if(l == r) return l;
16     int mid = (l + r) >> 1;
17     int sum = t[t[y].l].sum - t[t[x].l].sum;
18     if(sum >= v) return query(t[x].l, t[y].l, l, mid, v);
19     else return query(t[x].r, t[y].r, mid + 1, r, v - sum);
20 }
```

2.4.2 Unite Chairman Tree

```
1  //Q x到y路径第k大
2  //L link(x, y)
3  #include <bits/stdc++.h>
4  using namespace std;
5  typedef int LL;
6  const int MAXN = 8e4 + 5;
7  const int S = 18;
8  struct Node {
9      int l, r;
10     LL sum;
11 }t[MAXN * 800]; //2 * log^2(n)
12 int n, m, Q;
13 int cnt;
14 int rt[MAXN], sz[MAXN];
15 void update(int pre, int &x, int l, int r, int v) {
16     x = ++cnt; t[x] = t[pre]; t[x].sum++;
17     if(l == r) return;
18     int mid = (l + r) >> 1;
19     if(v <= mid) update(t[pre].l, t[x].l, l, mid, v);
20     else update(t[pre].r, t[x].r, mid + 1, r, v);
21 }
22 int query(int x, int y, int z, int w, int l, int r, int v) {
23     if(l == r) return l;
24     int mid = (l + r) >> 1;
25     int sum = t[t[x].l].sum + t[t[y].l].sum - t[t[z].l].sum - t[t[w].l].sum;
26     if(sum >= v) return query(t[x].l, t[y].l, t[z].l, t[w].l, l, mid, v);
27     return query(t[x].r, t[y].r, t[z].r, t[w].r, mid + 1, r, v - sum);
28 }
```

```cpp
int fa[MAXN][S + 3], dep[MAXN];
int val[MAXN];
vector<int> G[MAXN];
inline void addEdge(int x, int y) {
    G[x].push_back(y);
    G[y].push_back(x);
}
inline void upd(int x) {
    update(rt[fa[x][0]], rt[x], 1, n, val[x]);
    for(int i = 1; i <= S; i++) fa[x][i] = fa[fa[x][i - 1]][i - 1];
}
inline void Go(int &x, int step) {
    for(int i = S; i >= 0; i--) if(step >> i & 1) x = fa[x][i];
}
int lca(int x, int y) {
    if(dep[x] < dep[y]) swap(x, y);
    Go(x, dep[x] - dep[y]);
    if(x == y) return x;
    for(int i = S; i >= 0; i--) if(fa[x][i] != fa[y][i]) {
        x = fa[x][i], y = fa[y][i];
    }
    return fa[x][0];
}
int get_rt(int x) {
    for(int i = S; i >= 0; i--) if(fa[x][i]) x = fa[x][i];
    return x;
}
void dfs(int x, int f) {
    fa[x][0] = f;
    dep[x] = dep[f] + 1;
    upd(x);
    for(auto y : G[x]) {
        if(y == f) continue;
        dfs(y, x);
    }
}
void unite(int x, int y) {
    int rx = get_rt(x), ry = get_rt(y);
    if(sz[rx] > sz[ry]) swap(x, y), swap(rx, ry);
    addEdge(x, y);
    dfs(x, y);
    sz[ry] += cnt - rt[x] + 1;
}
void init() {
    cnt = 0;
    for(int i = 1; i <= n; i++) {
        rt[i] = sz[i] = 0;
        G[i].clear();
        dep[i] = 0;
    }
}
int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
    int T; cin >> T;
    while(T--) {
        cin >> n >> m >> Q;
        init();
        vector<int> b;
        map<int, int> mp;
        for(int i = 1; i <= n; i++) {
            cin >> val[i];
```

```
            b.push_back(val[i]);
        }
        sort(b.begin(), b.end());
        b.erase(unique(b.begin(), b.end()), b.end());
        for(int i = 1, tmp; i <= n; i++) {
            tmp = val[i];
            val[i] = lower_bound(b.begin(), b.end(), val[i]) - b.begin() + 1;
            mp[val[i]] = tmp;
        }
        for(int i = 1, u, v; i <= m; i++) {
            cin >> u >> v;
            addEdge(u, v);
        }
        for(int i = 1; i <= n; i++) if(!dep[i]) {
            dep[i] = 1;
            dfs(i, 0);
            sz[i] = cnt - rt[i] + 1;
        }
        char s[3]; int x, y, z, k, ans = 0;
        while(Q--) {
            cin >> s >> x >> y;
            x ^= ans; y ^= ans;
            if(s[0] == 'Q') {
                cin >> k; k ^= ans;
                z = lca(x, y);
                ans = query(rt[x], rt[y], rt[z], rt[fa[z][0]], 1, n, k);
                ans = mp[ans];
                cout << ans << endl;
            }
            else {
                unite(x, y);
            }
        }
    }
    return 0;
}
```

2.4.3 Persistent Trie

```
//区间异或最值查询
const int N=5e4+10;
int t[N];
int ch[N*32][2],val[N*32];
int cnt;
void init(){
    mem(ch,0);
    mem(val,0);
    cnt=1;
}
int add(int root,int x){
    int newroot=cnt++,ret=newroot;
    for(int i=30;i>=0;i--){
        ch[newroot][0]=ch[root][0];
        ch[newroot][1]=ch[root][1];
        int now=(x>>i)&1;
        root=ch[root][now];

        ch[newroot][now]=cnt++;
        newroot=ch[newroot][now];
        val[newroot]=val[root]+1;
```

```
    }

    return ret;
}
int query(int lt,int rt,int x){
    int ans=0;
    for(int i=30;i>=0;i--){
        int now=(x>>i)&1;
        if(val[ch[rt][now^1]]-val[ch[lt][now^1]]){
            ans|=(1<<i);
            rt=ch[rt][now^1];
            lt=ch[lt][now^1];
            } else{
            rt=ch[rt][now];
            lt=ch[lt][now];
        }
    }
    return ans;
}
```

2.4.4 SGT in BBST

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 1e5;
const int MAXM = 2e7;
const int LM = 0;
const int RM = 70005;
namespace T{
    int Tsz; queue<int> q;
    int ls[MAXM], rs[MAXM], val[MAXM];
    int Tan, Ta[105], Tbn, Tb[105];
    int newnode() {
        if(q.empty()) q.push(++Tsz);
        int x = q.front(); q.pop();
        ls[x] = rs[x] = val[x] = 0;
        return x;
    }
    void insert(int &x, int l, int r, int k, int f) {
        if(!x) x = newnode();
        val[x] += f;
        if(l == r) return;
        int mid = (l + r) >> 1;
        if(k <= mid) {
            insert(ls[x], l, mid, k,f);
        }else{
            insert(rs[x], mid+1, r, k,f);
        }
    }
    int query(int l, int r, int y) {
        if(l == r) return l;
        int mid = (l + r) >> 1;
        int sum = 0;
        for(int i = 0; i <= Tan; i++) sum -= val[ls[Ta[i]]];
        for(int i = 0; i <= Tbn; i++) sum += val[ls[Tb[i]]];

        if(y <= sum) {
            for(int i = 0; i <= Tan; i++) Ta[i] = ls[Ta[i]];
            for(int i = 0; i <= Tbn; i++) Tb[i] = ls[Tb[i]];
            return query(l, mid, y);
```

```
        }
        for(int i = 0; i <= Tan; i++) Ta[i] = rs[Ta[i]];
        for(int i = 0; i <= Tbn; i++) Tb[i] = rs[Tb[i]];
        return query(mid+1, r, y - sum);
    }
    void recycle(int v) {
        if(!v) return;
        q.push(v);
        recycle(ls[v]);
        recycle(rs[v]);
    }
};
namespace TT{
    int Tsn;
    int val[MAXN], Trt[MAXN];
    int sz[MAXN];
    int fa[MAXN], tr[MAXN][2];

    int root;
    double alp;

    int newnode(int x) {
        int tcnt = ++Tsn;
        val[tcnt] = x; Trt[tcnt] = 0;
        fa[tcnt] = tr[tcnt][0] = tr[tcnt][1] = 0;
        sz[tcnt] = 0;
        return tcnt;
    }
    bool isBad(int v) {
        return (double(sz[ tr[v][0] ]) > double(sz[v]) * alp) ||
            (double(sz[ tr[v][1] ]) > double(sz[v]) * alp);
    }
    vector<int> vec;
    void rRecycle(int v) {
        if(tr[v][0]) rRecycle(tr[v][0]);
        vec.push_back(v);
        T::recycle(Trt[v]);
        if(tr[v][1]) rRecycle(tr[v][1]);
    }
    int rBuild(int l, int r) {
        int mid = (l + r) >> 1, v = vec[mid];
        Trt[v] = 0; sz[v] = r - l + 1;
        for(int i = l; i <= mid; i++) {
            T::insert(Trt[v], LM, RM, val[vec[i]], 1);
        }
        tr[v][0] = (l <= mid-1) ? rBuild(l, mid - 1) : 0;
        if(tr[v][0]) fa[tr[v][0]] = v;
        tr[v][1] = (mid+1 <= r) ? rBuild(mid + 1, r) : 0;
        if(tr[v][1]) fa[tr[v][1]] = v;
        return v;
    }
    void rebuild(int v) {
        if(isBad(v)) {
            vec.clear();
            int u = fa[v], lr = tr[u][1] == v;
            rRecycle(v);
            if(vec.size()) v = rBuild(0, vec.size() - 1); else v = 0;
            if(u == 0) fa[root = v] = 0;
            else{
                tr[u][lr] = v;
                if(v) fa[v] = u;
```

```
            }
        }
    }
    int find(int x, int k) {
        int l = tr[x][0], r = tr[x][1];
        if(sz[l] + 1 == k) return x;
        if(sz[l] >= k) return find(l, k);
        return find(r, k - sz[l] - 1);
    }
    void ins(int x, int y) {
        int v = find(root, x);
        int p = tr[v][1], q = v;
        if(p) {
            for(;p; q = p, p = tr[p][0]);
            fa[p = tr[q][0] = newnode(y)] = q;
        }else {
            fa[p = tr[q][1] = newnode(y)] = q;
        }
        int fg = 0;
        T::insert(Trt[p], LM, RM, y, 1);
        sz[p] = 1;
        for(;fa[p]; p = fa[p]) {
            if(tr[fa[p]][0] == p)
                T::insert(Trt[fa[p]], LM, RM, y, 1);
            sz[fa[p]]++;
            if(isBad(fa[p])) fg = fa[p];
        }
        rebuild(fg);
    }
    void upd(int x, int y) {
        int p = find(root, x+1);
        int ty = val[p]; val[p] = y;
        T::insert(Trt[p], LM, RM, ty, -1);
        T::insert(Trt[p], LM, RM, y, 1);
        for(;fa[p]; p = fa[p]) {
            if(tr[fa[p]][0] == p) {
                T::insert(Trt[fa[p]], LM, RM, ty, -1);
                T::insert(Trt[fa[p]], LM, RM, y, 1);
            }
        }
    }
    int que(int x, int y,int z) {
        x = find(root, x);
        T::Ta[T::Tan = 0] = Trt[x];
        for(;fa[x]; x = fa[x])
            if(tr[fa[x]][1] == x) {
                T::Ta[++T::Tan] = Trt[fa[x]];
            }
        y = find(root, y+1);
        T::Tb[T::Tbn = 0] = Trt[y];
        for(;fa[y]; y = fa[y]) {
            if(tr[fa[y]][1] == y) {
                T::Tb[++T::Tbn] = Trt[fa[y]];
            }
        }
        return T::query(LM, RM, z);
    }
    void init(int n) {
        alp = 0.7;
        vec.clear();
        vec.push_back(newnode(RM));
```

```
        for(int i = 1, a; i <= n; i++) {
            scanf("%d", &a);
            vec.push_back(newnode(a));
        }
        root = rBuild(0,vec.size() - 1);
        fa[root] = 0;
    }
    void display(int v) {
        if(tr[v][0]) display(tr[v][0]);
        cerr<<val[v]<<" ";
        if(tr[v][1]) display(tr[v][1]);
    }
};
```

2.5 Tree Structures

2.5.1 dsu on tree

```
const int MAXN = 1e5 + 7;
vector<int> G[MAXN];
int bgison, dfs_clock, sz[MAXN], st[MAXN], bt[MAXN], et[MAXN];
int fg[MAXN], col[MAXN];
long long ans[MAXN];
void dfs1(int u, int fa) {
    sz[u] = 1;
    st[bt[u] = ++dfs_clock] = u;
    for(auto v : G[u])
    if(v != fa) {
        dfs1(v, u);
        sz[u] += sz[v];
    }
    et[u] = dfs_clock;
}
int maxx = 0;
void dfs2(int u, int fa, int keep) {
    int mx = -1, bigson = -1;
    for(auto &v : G[u])
    if(v != fa) {
        if(sz[v] > mx)
            mx = sz[v], bigson = v;
    }
    for(auto &v : G[u])
    if(v != fa && v != bigson)
        dfs2(v,u,0);
    if(bigson != -1) {
        dfs2(bigson, u, 1);
        ans[u] = ans[bigson];
        for(int &v : G[u])
            if(v != fa && v != bigson)
            for(int i = bt[v]; i <= et[v]; i++) {
                ++fg[col[st[i]]];
                if(fg[ col[st[i]] ] > maxx) maxx=fg[col[st[i]]], ans[u] = 0;
                if(fg[ col[st[i]] ] == maxx) ans[u] += col[st[i]];
            }
    }
    ++fg[col[u]];
    if(fg[col[u]] > maxx) maxx = fg[col[u]], ans[u] = 0;
    if(fg[col[u]] == maxx) ans[u] += col[u];
    if(keep == 0) {
```

```
        maxx = 0;
        for(int i = bt[u]; i <= et[u]; i++)
            fg[col[st[i]]] = 0;
    }
}
```

2.5.2 Vitural Tree

```
const int MAXN = ;
const int MAXM = ;
const LL INF = ;
const int S = 19;
int ecnt, head[MAXN];
struct Edge{int to, nxt; LL w;} e[MAXM], ve[MAXM];
inline void addEdgeT(int x, int y, LL w) {
    e[++ecnt] = (Edge) {y, head[x], w}; head[x] = ecnt;
}
int dep[MAXN], dfn_time, dfn[MAXN], fa[MAXN][S+1];
LL dis[MAXN][S+1];
void dfs(int v,int _fa) {
    dfn[v] = ++dfn_time;
    dep[v] = dep[_fa] + 1;
    fa[v][0] = _fa;
    for(int i = 1; i <= S; i++) {
        fa[v][i] = fa[fa[v][i-1]][i-1];
        dis[v][i] = min(dis[v][i-1], dis[fa[v][i-1]][i-1]);
    }
    for(int i = head[v], u; i; i = e[i].nxt)
    if((u = e[i].to) != _fa) {
        dis[u][0] = e[i].w;
        dfs(u, v);
    }
}
int getLca(int u, int v) {
    if(dep[u] < dep[v]) swap(u, v);
    for(int i = S; i >= 0; i--)
        if(dep[fa[u][i]] >= dep[v]) u = fa[u][i];
    if(u == v) return u;
    for(int i = S; i >= 0; i--)
        if(fa[u][i] != fa[v][i])
            u = fa[u][i], v = fa[v][i];
    return fa[u][0];
}
LL getDis(int u, int v) {
    if(dep[u] < dep[v]) swap(u, v);
    LL res = INF;
    for(int i = S; i >= 0; i--)
        if(dep[fa[u][i]] >= dep[v]) {
            res = min(res, dis[u][i]);
            u = fa[u][i];
        }
    return res;
}
namespace VituralTree{
    int hn, h[MAXN];
    int vecnt, vhead[MAXN];
    Edge ve[MAXM];
    int top, st[MAXN], cln, cl[MAXN];
    int fgn, fg[MAXN];//利用fg[i]==fgn判断是否为当前有效点
    void addEdgeVT(int x, int y, LL w) {
```

```
        ve[++vecnt] = (Edge) {y, vhead[x], w}; vhead[x] = vecnt;
    }
    inline void link(int u, int v) {
        LL w = getDis(u, v);
        addEdgeVT(u, v, w);
        addEdgeVT(v, u, w);
    }
    inline bool cmp(int a, int b) {return dfn[a] < dfn[b];}
    void build() {
        ++fgn;
        for(int i = 1; i <= hn; i++) fg[h[i]] = fgn;
        sort(h + 1, h + hn + 1, cmp);
        cl[cln = 1] = st[top = 1] = 1;
        for(int i = 1; i <= hn; i++) {
            int rem = getLca(st[top], h[i]);
            if(rem == st[top]) {
                if(rem != st[top]) cl[++cln] = st[++top] = h[i];
                continue;
            }
            while(top > 1 && dep[st[top - 1]] >= dep[rem]) {
                link(st[top - 1], st[top]); top--;
            }
            if(dep[st[top]] > dep[rem]) {
                link(rem, st[top]), top --;
            }
            if(rem != st[top]) cl[++cln] = st[++top] = rem;
            if(h[i] != st[top]) cl[++cln] = st[++ top] = h[i];
        }
        while(top > 1) {
            link(st[top - 1], st[top]), top --;
        }
    }
    void clear() {
        vecnt = 0;
        for(;cln; --cln) vhead[cl[cln]] = 0;
    }
    void sol() {
        build();
        //注意一号节点可能在虚树外面
        clear();
    }
}
```

2.5.3 Tree Decomposition

```
int sz[MAXN], dep[MAXN], top[MAXN], fa[MAXN], son[MAXN], num[MAXN], totw;
struct Edge {
    int to, nxt;
}e[MAXN << 1];
int head[MAXN], ecnt;
int n, m, Q;
#define Ls(x) (x << 1)
#define Rs(x) (x << 1 | 1)
struct Tree {
    int l, r, lazy;
    LL sum, mx;
}tree[MAXN << 2];
int A[MAXN], B[MAXN];
void push_up(int x) {
    tree[x].sum = tree[Ls(x)].sum + tree[Rs(x)].sum;
```

```
    tree[x].mx = max(tree[Ls(x)].mx, tree[Rs(x)].mx);
}
void push_down(int x) {
    if(tree[x].lazy) {
        tree[Ls(x)].sum += tree[x].lazy * (tree[Ls(x)].r - tree[Ls(x)].l + 1);
        tree[Rs(x)].sum += tree[x].lazy * (tree[Rs(x)].r - tree[Rs(x)].l + 1);
        tree[Ls(x)].mx += tree[x].lazy;
        tree[Rs(x)].mx += tree[x].lazy;
        tree[Ls(x)].lazy += tree[x].lazy;
        tree[Rs(x)].lazy += tree[x].lazy;
        tree[x].lazy = 0;
    }
}
void build(int x, int L, int R) {
    tree[x].lazy = 0;
    tree[x].l = L; tree[x].r = R;
    if(L == R) {
        tree[x].sum = B[L];
        tree[x].mx = B[L];
        return;
    }
    int mid = (L + R) >> 1;
    build(Ls(x), L, mid);
    build(Rs(x), mid + 1, R);
    push_up(x);
}
void update(int x, int L, int R, LL val) {
    if(tree[x].l >= L && tree[x].r <= R) {
        tree[x].lazy += val;
        tree[x].sum += val * (tree[x].r - tree[x].l + 1);
        tree[x].mx += val;
        return;
    }
    push_down(x);
    int mid = (tree[x].l + tree[x].r) >> 1;
    if(L <= mid) update(Ls(x), L, R, val);
    if(R > mid) update(Rs(x), L, R, val);
    push_up(x);
}
LL query(int x, int L, int R) {
    if(tree[x].l >= L && tree[x].r <= R)
        return tree[x].sum;
    push_down(x);
    int mid = (tree[x].l + tree[x].r) >> 1;
    LL res = 0;
    if(L <= mid) res += query(Ls(x), L, R);
    if(R > mid) res += query(Rs(x), L, R);
    return res;
}
LL query2(int x, int L, int R) {
    if(tree[x].l >= L && tree[x].r <= R)
        return tree[x].mx;
    push_down(x);
    int mid = (tree[x].l + tree[x].r) >> 1;
    LL res = -INF;
    if(L <= mid) res = max(res, query2(Ls(x), L, R));
    if(R > mid) res = max(res, query2(Rs(x), L, R));
    return res;
}
inline void add_edge(int x, int y) {
    e[++ecnt] = (Edge) {y, head[x]}; head[x] = ecnt;
```

```
}
void dfs1(int x) {
    sz[x] = 1; son[x] = 0;
    for(int i = head[x]; i; i = e[i].nxt) {
        int v = e[i].to;
        if(v == fa[x]) continue;
        fa[v] = x;
        dep[v] = dep[x] + 1;
        dfs1(v);
        sz[x] += sz[v];
        if(sz[v] > sz[son[x]]) son[x] = v;
    }
}
void dfs2(int x) {
    B[num[x]] = A[x];
    if(son[x]) {
        top[son[x]] = top[x];
        num[son[x]] = ++totw;
        dfs2(son[x]);
    }
    for(int i = head[x]; i; i = e[i].nxt) {
        int v = e[i].to;
        if(v == fa[x] || v == son[x]) continue;
        top[v] = v;
        num[v] = ++totw;
        dfs2(v);
    }
}
void up(int a, int b, int c) {
    int f1 = top[a], f2 = top[b];
    while(f1 != f2) {
        if(dep[f1] < dep[f2]) { swap(a, b); swap(f1, f2); }
        update(1, num[f1], num[a], c);
        a = fa[f1];
        f1 = top[a];
    }
    if(dep[a] > dep[b]) swap(a, b);
    update(1, num[a], num[b], c);
}
int qsum(int a, int b) {
    if(a == b) return query(1, num[a], num[a]);
    int f1 = top[a], f2 = top[b];
    int res = 0;
    while(f1 != f2) {
        if(dep[f1] < dep[f2]) { swap(a, b); swap(f1, f2); }
        res += query(1, num[f1], num[a]);
        a = fa[f1];
        f1 = top[a];
    }
    if(dep[a] > dep[b]) swap(a, b);
    res += query(1, num[a], num[b]);
    return res;
}
int qmax(int a, int b) {
    if(a == b) return query2(1, num[a], num[a]);
    int f1 = top[a], f2 = top[b];
    int res = -1000000000;
    while(f1 != f2) {
        if(dep[f1] < dep[f2]) { swap(a, b); swap(f1, f2); }
        res = max(res, query2(1, num[f1], num[a]));
        a = fa[f1];
```

```
        f1 = top[a];
    }
    if(dep[a] > dep[b]) swap(a, b);
    res = max(res, query2(1, num[a], num[b]));
    return res;
}
inline void init() {
    memset(head, 0, sizeof(head)); ecnt = 0;
    fa[1] = 0; dep[1] = 1; top[1] = 1; num[1] = 1; totw = 1;
}
inline void pre() {
    dfs1(1); dfs2(1); build(1, 1, totw);
}
```

2.5.4 Link-Cut Tree

```
namespace LCT {
    int fa[MAXN], rev[MAXN], tr[MAXN][2];
    int s[MAXN], val[MAXN];
    void push_up(int x) {
        int l = tr[x][0], r = tr[x][1];
        s[x] = s[l] + s[r] + val[x];
    }
    void Rev(int x) {
        rev[x] ^= 1; swap(tr[x][0], tr[x][1]);
    }
    void push_down(int x) {
        if(!rev[x]) return;
        int l = tr[x][0], r = tr[x][1];
        rev[x] = 0;
        if(l) Rev(l); if(r) Rev(r);
    }
    bool isroot(int x) {
        return tr[fa[x]][0] != x && tr[fa[x]][1] != x;
    }
    void pre(int x) {
        if(!isroot(x)) pre(fa[x]);
        push_down(x);
    }
    void rotate(int x) {
        int y = fa[x]; int z = fa[y];
        int l = tr[y][1] == x;
        int r = l ^ 1;
        if(!isroot(y)) tr[z][tr[z][1] == y] = x;
        fa[x] = z; fa[y] = x; fa[tr[x][r]] = y;
        tr[y][l] = tr[x][r]; tr[x][r] = y;
        push_up(y);
    }
    void splay(int x) {
        pre(x);
        int y, z;
        while(!isroot(x)) {
            y = fa[x]; z = fa[y];
            if(!isroot(y)) {
                if((tr[z][0] == y) == (tr[y][0] == x))rotate(y);
                else rotate(x);
            }
            rotate(x);
        }
        push_up(x);
```

```
45      }
46      void access(int x) {
47          int y = 0;
48          while(x) {
49              splay(x); tr[x][1] = y;
50              push_up(x);
51              y = x; x = fa[x];
52          }
53      }
54      void makeroot(int x) {
55          access(x); splay(x); Rev(x);
56      }
57      void lnk(int x, int y) {
58          makeroot(x); fa[x] = y;
59      }
60      void cut(int x, int y) {
61          makeroot(x); access(y); splay(y);
62          tr[y][0] = fa[x] = 0; push_up(y);
63      }
64      void update(int x, int y) {
65          makeroot(x); val[x] = y; push_up(x);
66      }
67      int query(int x, int y) {
68          makeroot(x); access(y); splay(y);
69          return s[y];
70      }
71      bool check(int x, int y) {
72          int tmp = y;
73          makeroot(x); access(y); splay(x);
74          while(!isroot(y)) y = fa[y];
75          splay(tmp);
76          return x == y;
77      }
78  }
```

3 String

3.1 Basics

3.1.1 Hash

```
const LL p1 = 201, p2 = 301, mod1 = 1200000319, mod2 = 2147483647;
struct Hash {
    LL a, b;
    void append(Hash pre, int v) {
        a = (pre.a * p1 + v) % mod1;
        b = (pre.b * p2 + v) % mod2;
    }
    void init(string S) {
        a = b = 0;
        for(int i = 0; i < S.size(); i++) append(*this, S[i]);
    }
    bool operator == (const Hash &x) const {
        return a == x.a && b == x.b;
    }
    bool operator < (const Hash &x) const {
        return a < x.a || (a == x.a && b < x.b);
    }
};
```

3.1.2 Minimum String

```
namespace minstring{
    int getmin(char *s, int sn) {
        int i = 0, j = 1, k = 0, t;
        while(i < sn && j < sn && k < sn) {
            t = s[(i + k) % sn] - s[(j + k) % sn];
            if(!t) k++;
            else {
                if(t > 0) i += k + 1; else j += k + 1;
                if(i == j) j++;
                k = 0;
            }
        }
        return i < j ? i : j;
    }
}
```

3.2 String Matching

3.2.1 Bitset Match

```
namespace BitMatch{
    const int S = 26;
    bitset<MAXN> bs[S], ret;
    char s[MAXN];
    inline int idx(char c) { return c - 'a'; }
    inline void init() {
        for(int i = 0; i < 26; i++) bs[i].reset();
        scanf("%s", s);
        int sn = strlen(s);
```

```
        for(int i = 0; i < sn; i++) bs[idx(s[i])].set(i);
    }
    void modify(int p, char ch) {
        bs[idx(s[p])].reset(p);
        s[p] = ch;
        bs[idx(s[p])].set(p);
    }
    int match(char *t, int tn) {//返回t串在s串中出现的次数
        ret = bs[idx(t[0])];
        for(int i = 1; i < tn; i++) {
            ret <<= 1;
            ret &= bs[idx(t[i])];
        }
        return ret.count();
    }
}
```

3.2.2 KMP && exKMP

```
namespace KMP {
    int fa[MAXN];
    void get_fail(char* t, int tn) {
        fa[0] = -1;
        int i = 0, j = -1;
        while(i < tn) {
            if (j == -1 || t[i] == t[j]) {
                ++i; ++j;
                fa[i] = t[i] != t[j] ? j : fa[j];
            }else{
                j = fa[j];
            }
        }
    }
    void kmp(char* s, int sn, char* t, int tn) {
        int i = 0, j = 0;
        while(i < sn) {
            if (j == -1 || s[i] == t[j]) {
                i++;j++;
                if(j == tn) {
                }
            }else j = fa[j];
        }
    }
}
namespace exKMP {
    int nxt[MAXN], ext[MAXN];
    void get_nxt(char* t, int tn) {
        int j = 0, mx = 0;
        nxt[0] = tn;
        for(int i = 1; i < tn; i++) {
            if(i >= mx || i + nxt[i - j] >= mx) {
                if(i > mx) mx = i;
                while(mx < tn && t[mx] == t[mx - i]) mx++;
                nxt[i] = mx - i;
                j = i;
            }else nxt[i] = nxt[i - j];
        }
    }
    void exkmp(char *s, int sn, char *t, int tn) {
        int j = 0, mx = 0;
```

```
        for(int i = 0; i < sn; i++) {
            if(i >= mx || i + nxt[i - j] >= mx) {
                if(i > mx) mx = i;
                while(mx < sn && mx - i < tn && s[mx] == t[mx - i]) mx++;
                ext[i] = mx - i;
                j = i;
            }else ext[i] = nxt[i - j];
        }
    }
}
```

3.2.3 AC Automaton

```
namespace AC {
    int ch[MAXN][sigma_size], last[MAXN];
    int val[MAXN], f[MAXN], sz;
    inline void init() { sz = 1; memset(ch[0], 0, sizeof(ch[0])); }
    inline int idx(char c) { return c - 'a'; }
    void insert(string s, int v) {
        int u = 0;
        for(int i = 0; i < s.size(); i++) {
            int c = idx(s[i]);
            if(!ch[u][c]) {
                memset(ch[sz], 0, sizeof(ch[sz]));
                val[sz] = 0;
                ch[u][c] = sz++;
            }
            u = ch[u][c];
        }
        val[u] = v;
    }
    void get_fail() {
        queue<int> q;
        f[0] = 0;
        for(int c = 0; c < sigma_size; c++) {
            int u = ch[0][c];
            if(u) { f[u] = 0; q.push(u); last[u] = 0; }
        }
        while(!q.empty()) {
            int r = q.front(); q.pop();
            for(int c = 0; c < sigma_size; c++) {
                int u = ch[r][c];
                if(!u) { ch[r][c] = ch[f[r]][c]; continue; }
                q.push(u);
                int v = f[r];
                while(v && !ch[v][c]) v = f[v];
                f[u] = ch[v][c];
                last[u] = val[f[u]] ? f[u] : last[f[u]];
            }
        }
    }
    inline void solve(int j) {
        if(j) {
            ans += val[j];
            solve(last[j]);
        }
    }
    void find(string T) {
        int j = 0;
        for(int i = 0; i < T.size(); i++) {
```

```
48              int c = idx(T[i]);
49              j = ch[j][c];
50              if(val[j]) solve(j);
51              else if(last[j]) solve(last[j]);
52          }
53      }
54  }
55  namespace AC {
56      int root, tcnt;
57      int ch[MAXN][sigma_size], fa[MAXN];
58      inline int newnode() {
59          fa[++tcnt] = 0;
60          for(int i = 0; i < sigma_size; ++i) ch[tcnt][i] = 0;
61          return tcnt;
62      }
63      inline void init() {
64          tcnt = -1;
65          root = newnode();
66      }
67      inline int idx(char c) { return c - 'a'; }
68      void extend(char *s, int sn) {
69          int cur = root;
70          for(int i = 0, c; i < sn; i++) {
71              if(!ch[cur][c = idx(s[i])])
72                  ch[cur][c] = newnode();
73              cur = ch[cur][c];
74          }
75      }
76      int q[MAXN], qh, qt;
77      void get_fail() {
78          qh = 1; qt = 0;
79          fa[root] = 0;
80          for(int c = 0, now; c < sigma_size; c++)
81              if((now = ch[root][c]) != 0)
82                  q[++qt] = now;
83          while(qh <= qt) {
84              int cur = q[qh++];
85              for(int c = 0, now; c < sigma_size; c++)
86                  if((now = ch[cur][c]) != 0) {
87                      fa[now] = ch[fa[cur]][c];
88                      q[++qt] = now;
89                  }else
90                      ch[cur][c] = ch[fa[cur]][c];
91          }
92      }
93  //统计模板串出现次数，每个模板串只计算一次
94  //      int cur = root, ans = 0;
95  //      for(int i = 0; i < sn; ++i) {
96  //          cur = ch[cur][idx(s[i])];
97  //          for(int j = cur; j && cnt[j] != -1; j = fa[j]) {
98  //              ans += cnt[j];
99  //              cnt[j] = -1;
100 //          }
101 //      }
102
103 }
```

3.3 Suffix Related

3.3.1 Suffix Array

```
namespace SA {
    char s[MAXN];
    int sa[MAXN], rank[MAXN], height[MAXN];
    int t[MAXN], t2[MAXN], c[MAXN], n;
    void clear() { n = 0; memset(sa, 0, sizeof(sa)); }
    void build(int m) {
        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 >= 0; 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 >= 0; 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;
        }
    }
    void buildHeight() {
        int k = 0;
        for(int i = 0; i < n; i++) rank[sa[i]] = i;
        for(int i = 0; i < n; i++) {
            if(k) k--;
            int j = sa[rank[i] - 1];
            while(s[i + k] == s[j + k]) k++;
            height[rank[i]] = k;
        }
    }
    void init() {
        n = strlen(s) + 1;
        build('z' + 1);
        buildHeight();
    }
}
```

3.3.2 Suffix Automaton

```
namespace SAM{
    int scnt, root, last;
    int fa[MAXN<<1], len[MAXN<<1], ch[MAXN<<1][26];
    int sc[MAXN<<1], tmpl[MAXN<<1], minl[MAXN<<1];

    int newnode(int _len, int q = 0) {
        fa[++scnt] = fa[q]; len[scnt] = _len;
        sc[scnt] = 0;tmpl[scnt] = 0; minl[scnt] = INF;
```

```
        for(int i = 0; i < 26; i++) ch[scnt][i] = ch[q][i];
        return scnt;
    }
    void init() {
        scnt = 0;
        root = last = newnode(0);
    }
    void extend(int c) {
        int p = last, np = newnode(len[p] + 1);
        for(;p && ch[p][c] == 0; p = fa[p]) ch[p][c] = np;
        if(!p) fa[np] = root;
        else{
            int q = ch[p][c];
            if(len[p] + 1 == len[q]) fa[np] = q;
            else{
                int nq = newnode(len[p] + 1, q);
                fa[np] = fa[q] = nq;
                for(; p && ch[p][c] == q; p = fa[p]) ch[p][c] = nq;
            }
        }
        last = np;
    }
    int c[MAXN], rs[MAXN << 1];
    void radix_sort(int n){
        for(int i = 0; i <= n; i++) c[i] = 0;
        for(int i = 1; i <= scnt; i++) c[len[i]]++;
        for(int i = 1; i <= n; i++) c[i] += c[i-1];
        for(int i = scnt; i >= 1; i--) rs[c[len[i]]--] = i;
    }
    void go(){
        scanf("%s",s);
        int n = strlen(s);
        for(int i = 0; i < n; ++i)
            extend(s[i] - 'a');
        radix_sort(n);
        //以下sc集合意义不同
        {//每个节点对应的位置之后有多少个不同子串
            for(int i = scnt; i >= 1; i--) {
                int S = 0;
                for(int j = 0; j < 26; j++)
                    S += sc[ ch[rs[i]][j] ];
                sc[rs[i]] = S + 1;
            }
        }
        {//right集合大小
            int cur = root;
            for(int i = 0; i < n; ++i) {
                cur = ch[cur][s[i]-'a'];
                sc[cur]++;
            }
            for(int i = scnt; i >= 1; --i) {
                sc[ fa[rs[i]] ] += sc[rs[i]];
            }
        }
        //公共子串
        //tmpl,当前字符串:在状态cur, 与模板串的最长公共后缀
        //minl, 多个字符串:在状态cur,与模板串的最长公共后缀
        //注意:在状态cur匹配成功时, cur的祖先状态与字符串的最长公共后缀
        for(; ~scanf("%s",s);) {
            int cur = root, Blen = 0;
            for(int i = 0; i <= scnt; i++)
```

```
                tmpl[i] = 0;
            n = strlen(s);
            for(int i = 0, x; i < n; i++) {
                x = s[i] - 'a';
                if(ch[cur][x]) {
                    ++Blen;
                    cur = ch[cur][x];
                }else{
                    for(;cur && ch[cur][x] == 0; cur = fa[cur]);
                    if(cur) {
                        Blen = len[cur] + 1;
                        cur = ch[cur][x];
                    }else{
                        cur = root; Blen = 0;
                    }
                }
                tmpl[cur] = max(tmpl[cur], Blen);
            }
            for(int i = scnt; i ; --i) {
                if( tmpl[ fa[rs[i]] ] < tmpl[ rs[i] ])
                    tmpl[ fa[rs[i]] ] = len[ fa[rs[i]] ];
                minl[ rs[i] ] = min(minl[ rs[i] ], tmpl[ rs[i] ]);
            }
        }
    }
}
namespace exSAM{
    int scnt, root;
    int fa[MAXN<<1], len[MAXN<<1], ch[MAXN<<1][26];
    int sc[MAXN<<1], tmpl[MAXN<<1], minl[MAXN<<1];

    int newnode(int _len, int q = 0) {
        fa[++scnt] = fa[q]; len[scnt] = _len;
        sc[scnt] = 0;tmpl[scnt] = 0; minl[scnt] = INF;
        for(int i = 0; i < 26; i++) ch[scnt][i] = ch[q][i];
        return scnt;
    }
    void init() {
        scnt = 0;
        root = newnode(0);
    }
    int work(int p,int c){
        int q = ch[p][c];
        int nq = newnode(len[p] + 1, q);
        fa[q] = nq;
        for(; p && ch[p][c] == q; p = fa[p]) ch[p][c] = nq;
        return nq;
    }
    int extend(int p, int c) {
        if (ch[p][c]){
            int q = ch[p][c];
            if (len[p] + 1 == len[q]) return q;
            return work(p, c);
        }
        int np = newnode(len[p] + 1);
        for(;p && ch[p][c] == 0; p = fa[p]) ch[p][c] = np;
        if (!p) fa[np] = root;
        else{
            int q = ch[p][c];
            if (len[p] + 1 == len[q]) fa[np] = q;
            else fa[np] = work(p, c);
```

```
        }
        return np;
    }
    void solve() {
        int n; scanf("%d",&n);
        for(int i = 1; i <= n; i++) {
            scanf("%s", s);
            int sn = strlen(s);
            int last = root;
            for(int j = 0; j < sn; ++j)
                last = extend(last, s[j] - 'a');
        }
    }
}
```

3.4 Palindrome Related

3.4.1 Manacher

```
namespace Manachar {
    char S[MAXN << 1];
    int scnt, ans;
    int p[MAXN << 1]; //p[i] - 1
    void init(char *s0, int sn0) {
        S[0] = '$'; S[1] = '#';
        for(int i = 0; i < sn0; i++) {
            S[2 * i + 2] = s0[i];
            S[2 * i + 3] = '#';
        }
        scnt = sn0 * 2 + 2;
        S[scnt] = '&';
    }
    void manachar() {
        int id = 0, mx = 0;
        for(int i = 1; i < scnt; i++) {
            p[i] = mx > i ? min(p[2 * id - i], mx - i) : 1;
            while(S[i + p[i]] == S[i - p[i]]) p[i]++;
            if(i + p[i] > mx) {
                mx = i + p[i];
                id = i;
            }
        }
    }
}
```

3.4.2 Palindromic Automaton

```
namespace PAM {
    int scnt, S[MAXN];
    int pcnt, last, len[MAXN], fail[MAXN], ch[MAXN][26];
    int cnt[MAXN]; //节点i表示的本质不同的串的个数(调用count())
    int num[MAXN]; //以节点i表示的最长回文串的最右端点为回文串结尾的回文串个数
    int newnode(int _len) {
        len[pcnt] = _len;
        cnt[pcnt] = num[pcnt] = 0;
        for(int i = 0; i < 26; i++) ch[pcnt][i] = 0;
        return pcnt++;
    }
```

```
    inline void init() {
        S[scnt = 0] = -1;
        pcnt = 0;newnode(0);newnode(-1);
        fail[0] = 1;last = 0;
    }
    int getfail(int x) {
        while(S[scnt - len[x] - 1] != S[scnt]) x = fail[x];
        return x;
    }
    void extend(int c) {
        S[++scnt] = c;
        int cur = getfail(last);
        if(!ch[cur][c]) {
            int now = newnode(len[cur] + 2);
            fail[now] = ch[getfail(fail[cur])][c];
            ch[cur][c] = now;
            num[now] = num[fail[now]] + 1;
        }
        last = ch[cur][c];
        cnt[last]++;
    }
    void count() {
        for(int i = pcnt - 1; i >= 0; i--) cnt[fail[i]] += cnt[i];
    }
};
```

3.5 Substring Automaton

```
for(int j = 0; j < 26; j++)
    ch[n][j] = ch[n+1][j] = n + 1;
for(int i = n; i >= 1; i--) {
    for(int j = 0; j < 26; j++)
        ch[i-1][j] = ch[i][j];
    ch[i-1][s[i]-'a'] = i;
}
```

4 Math

4.1 Algebra

4.1.1 FFT

```
//不预处理精度
const double pi = acos(-1.0);
const int MAXN = 300003;
struct comp {
    double x, y;
    comp operator + (const comp& a) const { return (comp) {x + a.x, y + a.y}; }
    comp operator - (const comp& a) const { return (comp) {x - a.x, y - a.y}; }
    comp operator * (const comp& a) const { return (comp) {x * a.x - y * a.y, x * a.y + y * a.x}; }
};
int rev[MAXN], T;
comp tmp;
void fft(comp *a, int r) {
    if(r == -1) for(int i = 0; i < T; i++) a[i] = a[i] * a[i];
    for(int i = 0; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
    for(int i = 2, mid = 1; i <= T; mid = i, i <<= 1) {
        comp step = (comp) {cos(pi / mid), r * sin(pi / mid)};
        for(int j = 0; j < T; j += i) {
            comp cur = (comp) {1, 0};
            for(int k = j; k < j + mid; k++, cur = cur * step) {
                tmp = a[k + mid] * cur;
                a[k + mid] = a[k] - tmp;
                a[k] = a[k] + tmp;
            }
        }
    }
    if(r == -1) for(int i = 0; i < T; i++) a[i].y = (int)(a[i].y / T / 2 + 0.5);
}
comp A[MAXN];
void init(int n) {
    for(T = 1; T <= n; T <<= 1);
    for(int i = 1; i < T; i++) {
        if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
        else rev[i] = rev[i >> 1] >> 1;
        //A[i] = (comp) {0, 0};
    }
}
//预处理精度
int rev[MAXN], T;
comp Sin[MAXN], tmp;
void fft(comp *a, int r) {
    if(r == -1) {
        for(int i = 0; i < (T >> 1); i++) Sin[i].y = -Sin[i].y;
        for(int i = 0; i < T; i++) a[i] = a[i] * a[i];
    }
    for(int i = 1; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
    for(int i = 2, mid = 1, s = (T >> 1); i <= T; mid = i, i <<= 1, s >>= 1) {
        for(int j = 0; j < T; j += i) {
            for(int k = j, cur = 0; k < j + mid; k++, cur += s) {
                tmp = a[k + mid] * Sin[cur];
                a[k + mid] = a[k] - tmp;
                a[k] = a[k] + tmp;
            }
```

```
        }
    }
    if(r == -1) for(int i = 0; i < T; i++) a[i].y = (int)(a[i].y / T / 2 + 0.5);
}
comp A[MAXN];
void init(int n) {
    for(T = 1; T <= n; T <<= 1);
    for(int i = 0; i < T; i++) {
        if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
        else rev[i] = rev[i >> 1] >> 1;
        //A[i] = (comp) {0, 0};
    }
    for(int i = 0; i < (T >> 1); i++) {
        Sin[i] = (comp) {cos(2 * pi * i / T), sin(2 * pi * i / T)};
    }
}
int main() {
    scanf("%d%d", &n, &m);
    init(n + m);
    for(int i = 0; i <= n; i++) scanf("%lf", &A[i].x);
    for(int i = 0; i <= m; i++) scanf("%lf", &A[i].y);
    fft(A, 1);
    fft(A, -1);
    for(int i = 0; i <= n + m; i++) printf("%d%c", (int)(A[i].y), i == n + m ? '\n' : '
    ');
    return 0;
}
```

4.1.2 NTT

4.常用NTT模数：

以下模数的共同$g = 3189$

$p = r \times 2^k + 1$	k	g
104857601	22	3
167772161	25	3
469762049	26	3
950009857	21	7
998244353	23	3
1004535809	21	3
2013265921	27	31
2281701377	27	3
3221225473	30	5

```
const int MAXN = 300005, G = 3, mod = 998244353; //or (479LL<<21) + 1
int rev[MAXN], T;
LL qpow(LL x, LL y) {
    LL res = 1;
    while(y) {
        if(y & 1) res = res * x % mod;
        x = x * x % mod;
        y >>= 1;
    }
    return res;
}
LL A[MAXN], B[MAXN];
void ntt(LL *a, int r) {
```

```
    if(r == -1) for(int i = 0; i < T; i++) A[i] = A[i] * B[i] % mod;
    for(int i = 0; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
    for(int i = 2, mid = 1; i <= T; mid = i, i <<= 1) {
        LL gn = qpow(G, (mod - 1) / i);
        if(r == -1) gn = qpow(gn, mod - 2);
        for(int j = 0; j < T; j += i) {
            LL cur = 1, tmp;
            for(int k = j; k < j + mid; k++, cur = cur * gn % mod) {
                tmp = a[k + mid] * cur % mod;
                a[k + mid] = ((a[k] - tmp) % mod + mod) % mod;
                a[k] = (a[k] + tmp) % mod;
            }
        }
    }
    if(r == -1) {
        LL inv = qpow(T, mod - 2);
        for(int i = 0; i < T; i++) a[i] = a[i] * inv % mod;
    }
}
void init(int n) {
    for(T = 1; T <= n; T <<= 1);
    for(int i = 0; i < T; i++) {
        if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
        else rev[i] = rev[i >> 1] >> 1;
    }
}
```

4.1.3 MTT

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const double pi = acos(-1.0);
const int MAXN = 300003;
struct comp {
    double x, y;
    comp operator + (const comp& a) const { return (comp) {x + a.x, y + a.y}; }
    comp operator - (const comp& a) const { return (comp) {x - a.x, y - a.y}; }
    comp operator * (const comp& a) const { return (comp) {x * a.x - y * a.y, x * a.y +
    y * a.x}; }
};
#define conj(a) ((comp){a.x, -a.y})
int rev[MAXN], T;
comp Sin[MAXN], tmp;
void fft(comp *a, int r) {
    for(int i = 1; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
    for(int i = 2, mid = 1, s = (T >> 1); i <= T; mid = i, i <<= 1, s >>= 1) {
        for(int j = 0; j < T; j += i) {
            for(int k = j, cur = 0; k < j + mid; k++, cur += s) {
                tmp = a[k + mid] * Sin[cur];
                a[k + mid] = a[k] - tmp;
                a[k] = a[k] + tmp;
            }
        }
    }
}
void init(int n) {
    for(T = 1; T <= n; T <<= 1);
    for(int i = 0; i < T; i++) {
        if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
```

```
        else rev[i] = rev[i >> 1] >> 1;
    }
    for(int i = 0; i < (T >> 1); i++) {
        Sin[i] = (comp) {cos(2 * pi * i / T), sin(2 * pi * i / T)};
    }
}
int n, m, mod;
void mtt(int *x, int *y) {
    for(int i = 0; i < T; i++) (x[i] += mod) %= mod, (y[i] += mod) %= mod;
    static comp a[MAXN], b[MAXN];
    static comp dfta[MAXN], dftb[MAXN], dftc[MAXN], dftd[MAXN];
    for(int i = 0; i < T; i++) {
        a[i] = {x[i] & 0x7fff, x[i] >> 15};
        b[i] = {y[i] & 0x7fff, y[i] >> 15};
    }
    fft(a, 1); fft(b, 1);
    for(int i = 0; i < T; i++) {
        int j = (T - i) & (T - 1);
        static comp da, db, dc, dd;
        da = (a[i] + conj(a[j])) * (comp){0.5, 0};
        db = (a[i] - conj(a[j])) * (comp){0, -0.5};
        dc = (b[i] + conj(b[j])) * (comp){0.5, 0};
        dd = (b[i] - conj(b[j])) * (comp){0, -0.5};
        dfta[j] = da * dc;
        dftb[j] = da * dd;
        dftc[j] = db * dc;
        dftd[j] = db * dd;
    }
    for(int i = 0; i < T; i++) {
        a[i] = dfta[i] + dftb[i] * (comp) {0, 1};
        b[i] = dftc[i] + dftd[i] * (comp) {0, 1};
    }
    //for(int i = 0; i < (T >> 1); i++) Sin[i].y = -Sin[i].y;
    fft(a, -1); fft(b, -1);
    for(int i = 0; i < T; i++) {
        static int da, db, dc, dd;
        da = (LL)(a[i].x / T + 0.5) % mod;
        db = (LL)(a[i].y / T + 0.5) % mod;
        dc = (LL)(b[i].x / T + 0.5) % mod;
        dd = (LL)(b[i].y / T + 0.5) % mod;
        x[i] = ((da + ((LL)(db + dc) << 15) + ((LL)dd << 30)) % mod + mod) % mod;
    }
}
int main() {
    static int a[MAXN], b[MAXN];
    scanf("%d%d%d", &n, &m, &mod);
    for(int i = 0; i <= n; i++) scanf("%d", a + i);
    for(int i = 0; i <= m; i++) scanf("%d", b + i);
    init(n + m);
    mtt(a, b);
    for(int i = 0; i <= n + m; i++) printf("%d%c", a[i], i == n + m ? '\n' : ' ');
    return 0;
}
```

4.1.4 FWT

```
void FWT(LL *a,int n) {
    for(int i = 2;i <= n; i <<= 1) {
        for(int j = 0; j < n; j += i) {
            for(int d = 0, w = i >> 1; d < w; d++){
```

```
                LL u = a[j + d], v = a[j + d + w];
                //xor: a[j + d] = u + v, a[j + d + w] = u - v;
                //and: a[j + d] = u + v;
                //or : a[j + d + w] = u + v;
            }
        }
    }
}
void UFWT(LL *a, int n) {
    for(int i = 2; i <= n; i <<= 1) {
        for(int j = 0; j < n; j += i) {
            for(int d = 0, w = i >> 1; d < w; d++) {
                LL u = a[j + d], v = a[j + d + w];
                //xor: a[j + d] = (u + v) / 2, a[j + d + w] = (u - v) / 2;
                //and: a[j + d] = u - v;
                //or : a[j + d + w] = v - u;
            }
        }
    }
}
void solve(int n) {
    FWT(a, n); FWT(b, n);
    for(int i = 0; i < n; i++) a[i] = a[i] * b[i];
    UFWT(a, n);
}
```

4.1.5 FFT Divide and Conquer

$$f_i = \sum_{j=1}^{i-1} f_j \cdot g_{i-j}$$

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int MAXN = 300005, G = 3, mod = 998244353;
namespace NTT {
    LL A[MAXN], B[MAXN]
    int rev[MAXN], T;
    LL qpow(LL x, LL y) {
        LL res = 1;
        while(y) {
            if(y & 1) res = res * x % mod;
            x = x * x % mod;
            y >>= 1;
        }
        return res;
    }
    void ntt(LL *a, int r) {
        for(int i = 0; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
        for(int i = 2, mid = 1; i <= T; mid = i, i <<= 1) {
            LL gn = qpow(G, (mod - 1) / i);
            if(r == -1) gn = qpow(gn, mod - 2);
            for(int j = 0; j < T; j += i) {
                LL cur = 1, tmp;
                for(int k = j; k < j + mid; k++, cur = cur * gn % mod) {
                    tmp = a[k + mid] * cur % mod;
                    a[k + mid] = ((a[k] - tmp) % mod + mod) % mod;
                    a[k] = (a[k] + tmp) % mod;
```

```
                }
            }
        }
        if(r == -1) {
            LL inv = qpow(T, mod - 2);
            for(int i = 0; i < T; i++) a[i] = a[i] * inv % mod;
        }
    }
    void init(int n) {
        for(T = 1; T <= n; T <<= 1);
        for(int i = 0; i < T; i++) {
            if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
            else rev[i] = rev[i >> 1] >> 1;
        }
    }
}
LL f[MAXN], g[MAXN];
using namespace NTT;
void solve(int l, int r) {
    if(l == r) return;
    int mid = (l + r) >> 1;
    solve(l, mid);
    init(r - l);
    for(int i = 0; i < T; i++) A[i] = B[i] = 0;
    for(int i = 0; i <= mid - l; i++) A[i] = f[i + l];
    for(int i = 0; i <= r - l; i++) B[i] = g[i];
    ntt(A, 1); ntt(B, 1);
    for(int i = 0; i < T; i++) A[i] = A[i] * B[i] % mod;
    ntt(A, -1);
    for(int i = mid + 1; i <= r; i++) f[i] =(f[i] + A[i - l]) % mod;
    solve(mid + 1, r);
}
int main() {
    int n; scanf("%d", &n);
    for(int i = 1; i < n; i++) scanf("%lld", g + i);
    f[0] = 1;
    solve(0, n - 1);
    for(int i = 0; i < n; i++) printf("%lld%c", f[i], i == n - 1 ? '\n' : ' ');
    return 0;
}
```

4.1.6 Linear Basis

```
//dynamic
const int D = 60;
struct Basis {
    vector<int> ind;
    vector<LL> base;
    Basis() {
        ind.resize(D, -1);
        base.resize(D);
    }
    bool update(LL x, int id) {
        for(int i = 0; i < D; i++) if(~ind[i] && x >> i & 1) {
            x ^= base[i];
        }
        if(!x) return 1;
        int pos = __builtin_ctzll(x);
        ind[pos] = id;
        base[pos] = x;
```

```
        return 0;
    }
};
//array
int Gauss(int n, int m) {
    int num = 1;
    for(int x = 1; x <= n && x <= m; x++) {
        int t = 0;
        for(int j = x; j <= m; j++) if(g[j][x]) { t = j; break; }
        if(t) {
            swap(g[x], g[t]);
            for(int i = x + 1; i <= n; i++) {
                if(g[i][x]) {
                    for(int k = 1; k <= m; k++) g[i][k] ^= g[x][k];
                }
            }
            num++;
        }
    }
    return --num;
}
//long long
int Gauss() {
    int num = 1;
    for(int k = 61; k >= 0; k--) {
        int t = 0;
        for(int j = num; j <= cnt; j++) if((A[j] >> k) & 1) { t = j; break; }
        if(t) {
            swap(A[t], A[num]);
            for(int j = num + 1; j <= cnt; j++) if((A[j] >> k) & 1) A[j] ^= A[num];
            num++;
        }
    }
    return --num;
}
//det
LL det(int n){
    LL ret = 1;
    for(int i = 1; i < n; i++){
        for(int j = i + 1; j < n; j++)
            while(a[j][i]){
                LL t = (LL)a[i][i] / a[j][i];
                for(int k = i; k < n; k++) {
                    a[i][k] = (a[i][k]-a[j][k] * t);
                    swap(a[i][k], a[j][k]);
                }
                ret = -ret;
            }
        if(a[i][i] == 0)return 0;
        ret = ret * a[i][i];
    }
    if(ret<0)ret = -ret;
    return ret;
}
```

4.1.7 Polynomial

Inverse:

$$\begin{aligned} A_x * B'_x &\equiv 1 && (\bmod\ x^{\frac{n}{2}}) \\ A_x * B_x &\equiv 1 && (\bmod\ x^n) \\ B_x &\equiv 2 \cdot B'_x - A_x \cdot {B'}_x^2 && (\bmod\ x^n) \end{aligned}$$

Division:

$$A_r(x) = x^n A(\frac{1}{x}) \Rightarrow A_r(x)[i] = A(x)[n-i]$$

$$A(x) = B(x) * Q(x) + R(x)$$

$$Q_r(x) = A_r(x) * B_r^{-1}(x)$$

$$R(x) = A(x) - B(x) * Q(x)$$

```cpp
//NTT模数
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int MAXN = 300005, G = 3, mod = 998244353;
namespace NTT {
    int rev[MAXN], T;
    LL qpow(LL x, LL y) {
        LL res = 1;
        while(y) {
            if(y & 1) res = res * x % mod;
            x = x * x % mod;
            y >>= 1;
        }
        return res;
    }
    void ntt(LL *a, int r) {
        for(int i = 0; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
        for(int i = 2, mid = 1; i <= T; mid = i, i <<= 1) {
            LL gn = qpow(G, (mod - 1) / i);
            if(r == -1) gn = qpow(gn, mod - 2);
            for(int j = 0; j < T; j += i) {
                LL cur = 1, tmp;
                for(int k = j; k < j + mid; k++, cur = cur * gn % mod) {
                    tmp = a[k + mid] * cur % mod;
                    a[k + mid] = ((a[k] - tmp) % mod + mod) % mod;
                    a[k] = (a[k] + tmp) % mod;
                }
            }
        }
        if(r == -1) {
            LL inv = qpow(T, mod - 2);
            for(int i = 0; i < T; i++) a[i] = a[i] * inv % mod;
        }
    }
    void init(int n) {
        for(T = 1; T <= n; T <<= 1);
        for(int i = 0; i < T; i++) {
            if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
            else rev[i] = rev[i >> 1] >> 1;
```

```
        }
    }
}
namespace poly {
using namespace NTT;
    void mul(LL *a, LL *b, LL *c, int n, int m) {
        init(n + m);
        static LL x[MAXN], y[MAXN];
        for(int i = 0; i < T; i++) {
            x[i] = i <= n ? a[i] : 0;
            y[i] = i <= m ? b[i] : 0;
        }
        ntt(x, 1); ntt(y, 1);
        for(int i = 0; i < T; i++) c[i] = x[i] * y[i] % mod;
        ntt(c, -1);
    }
    void poly_inv(LL *a, LL *b, int n) {
        if(n == 1) {
            b[0] = qpow(a[0], mod - 2);
            return;
        }
        static LL c[MAXN], d[MAXN];
        memset(c, 0, n * sizeof(LL));
        poly_inv(a, c, n >> 1);
        for(int i = 0; i < n; i++) {
            d[i] = a[i];
        }
        init(n);
        ntt(c, 1); ntt(d, 1);
        for(int i = 0; i < T; i++) b[i] = c[i] * (2 + mod - d[i] * c[i] % mod) % mod;
        ntt(b, -1);
        for(int i = n; i < T; i++) b[i] = 0;
    }
    void inv(LL *a, LL *b, int n) {//A must be different from B
        init(n);
        poly_inv(a, b, T);
    }
    //A_x = B_x * Q_x + R_x;
    void div(LL *A, LL *B, LL *Q, LL *R, int n, int m) {
        static LL f[MAXN], g[MAXN], inv_g[MAXN];
        for(int i = 0; i <= n; i++) f[n - i] = A[i];
        for(int i = 0; i <= m; i++) g[m - i] = B[i];
        //inv(G_r)
        for(int i = n - m + 1; i <= m; i++) g[i] = 0;
        inv(g, inv_g, n - m);
        //Q
        mul(f, inv_g, f, n, n - m);
        for(int i = 0; i <= n - m; i++) Q[i] = f[n - m - i];
        //R
        mul(Q, B, f, n - m, m);
        for(int i = 0; i < m; i++) R[i] = (A[i] + mod - f[i]) % mod;
    }
}
LL A[MAXN], B[MAXN];
LL Q[MAXN], R[MAXN];
int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
    int n, m; cin >> n >> m;
    for(int i = 0; i <= n; i++) cin >> A[i];
    for(int i = 0; i <= m; i++) cin >> B[i];
    poly::div(A, B, Q, R, n, m);
```

```
    for(int i = 0; i <= n - m; i++) cout << Q[i] << " "; cout << endl;
    for(int i = 0; i < m; i++) cout << R[i] << " "; cout << endl;
    return 0;
}
//非NTT模数求逆
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const double pi = acos(-1.0);
const int MAXN = 300003;
struct comp {
    double x, y;
    comp operator + (const comp& a) const { return (comp) {x + a.x, y + a.y}; }
    comp operator - (const comp& a) const { return (comp) {x - a.x, y - a.y}; }
    comp operator * (const comp& a) const { return (comp) {x * a.x - y * a.y, x * a.y + y * a.x}; }
};
#define conj(a) ((comp){a.x, -a.y})
int rev[MAXN], T;
comp Sin[MAXN], tmp;
void fft(comp *a, int r) {
    for(int i = 1; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
    for(int i = 2, mid = 1, s = (T >> 1); i <= T; mid = i, i <<= 1, s >>= 1) {
        for(int j = 0; j < T; j += i) {
            for(int k = j, cur = 0; k < j + mid; k++, cur += s) {
                tmp = a[k + mid] * Sin[cur];
                a[k + mid] = a[k] - tmp;
                a[k] = a[k] + tmp;
            }
        }
    }
}
void init(int n) {
    //for(T = 1; T <= n; T <<= 1);
    T = n << 1;
    for(int i = 0; i < T; i++) {
        if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
        else rev[i] = rev[i >> 1] >> 1;
    }
    for(int i = 0; i < (T >> 1); i++) {
        Sin[i] = (comp) {cos(2 * pi * i / T), sin(2 * pi * i / T)};
    }
}
const int mod = 1e9 + 7;
void mtt(int *x, int *y) {
    for(int i = 0; i < T; i++) (x[i] += mod) %= mod, (y[i] += mod) %= mod;
    static comp a[MAXN], b[MAXN];
    static comp dfta[MAXN], dftb[MAXN], dftc[MAXN], dftd[MAXN];
    for(int i = 0; i < T; i++) {
        a[i] = {x[i] & 0x7fff, x[i] >> 15};
        b[i] = {y[i] & 0x7fff, y[i] >> 15};
    }
    fft(a, 1); fft(b, 1);
    for(int i = 0; i < T; i++) {
        int j = (T - i) & (T - 1);
        static comp da, db, dc, dd;
        da = (a[i] + conj(a[j])) * (comp){0.5, 0};
        db = (a[i] - conj(a[j])) * (comp){0, -0.5};
        dc = (b[i] + conj(b[j])) * (comp){0.5, 0};
        dd = (b[i] - conj(b[j])) * (comp){0, -0.5};
        dfta[j] = da * dc;
```

```
        dftb[j] = da * dd;
        dftc[j] = db * dc;
        dftd[j] = db * dd;
    }
    for(int i = 0; i < T; i++) {
        a[i] = dfta[i] + dftb[i] * (comp) {0, 1};
        b[i] = dftc[i] + dftd[i] * (comp) {0, 1};
    }
    //for(int i = 0; i < (T >> 1); i++) Sin[i].y = -Sin[i].y;
    fft(a, -1); fft(b, -1);
    for(int i = 0; i < T; i++) {
        static int da, db, dc, dd;
        da = (LL)(a[i].x / T + 0.5) % mod;
        db = (LL)(a[i].y / T + 0.5) % mod;
        dc = (LL)(b[i].x / T + 0.5) % mod;
        dd = (LL)(b[i].y / T + 0.5) % mod;
        x[i] = ((da + ((LL)(db + dc) << 15) + ((LL)dd << 30)) % mod + mod) % mod;
    }
}
LL qpow(LL x, LL y) {
    LL res = 1;
    while(y) {
        if(y & 1) res = res * x % mod;
        x = x * x % mod;
        y >>= 1;
    }
    return res;
}
void poly_inv(int *a, int *b, int n) {
    if(n == 1) {
        b[0] = qpow(a[0], mod - 2);
        return;
    }
    static int c[MAXN], cc[MAXN], d[MAXN];
    memset(c, 0, n * sizeof(int)); memset(cc, 0, n * sizeof(int)); memset(d, 0, n *
    sizeof(int));
    poly_inv(a, c, n >> 1);
    for(int i = 0; i < n; i++) cc[i] = c[i], d[i] = a[i];
    init(n);
    mtt(cc, c);
    mtt(d, cc);
    for(int i = 0; i < T; i++) b[i] = (2 * c[i] % mod + mod - d[i]) % mod;
    for(int i = 0; i < n; i++) b[n + i] = 0;
}
int A[MAXN], B[MAXN];
int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
    int n; cin >> n; n--;
    for(T = 1; T <= n; T <<= 1);
    for(int i = 0; i <= n; i++) cin >> A[i];
    poly_inv(A, B, T);
    for(int i = 0; i <= n; i++) cout << B[i] << " ";
    cout << endl;
    return 0;
}

//LOJ挑战多项式(备用)
#include <bits/stdc++.h>
#define rep(i, a, b) for (int i = (a); i <= (b); i++)
#define per(i, a, b) for (int i = (a); i >= (b); i--)
#define REP(i, n) for (int i = (0); i < (n); i++)
```

```
#define fi first
#define se second
#define mp make_pair
#define pb push_back
using namespace std;
typedef unsigned long long ull;
typedef pair<int, int> pii;
typedef vector<int> poly;
typedef long long ll;
struct ano {
    char a[1 << 25], *s;
    char b[1 << 25], *t;
    ano() : s(a), t(b) { a[fread(a, 1, sizeof a, stdin)] = 0; }
    ~ano() { fwrite(b, 1, t - b, stdout); }
    operator int() {
        int x = 0;
        while (*s < 48) ++s;
        while (*s > 32) x = x * 10 + *s++ - 48;
        return x;
    }
    ll in() {
        ll x = 0;
        while (*s < 48) ++s;
        while (*s > 32) x = x * 10 + *s++ - 48;
        return x;
    }
    void out(int x, char e = ' ') {
        static char c[12];
        char *i = c;
        if (!x)
            *t++ = 48;
        else {
            while (x) {
                int y = x / 10;
                *i++ = x - y * 10 + 48, x = y;
            }
            while (i != c) *t++ = *--i;
        }
        *t++ = e;
    }
} buf;
const int mod = 998244353;
namespace Poly {
const int N = (1 << 20) + 5, g = 3;
inline int power(int x, int p) {
    int res = 1;
    for (; p; p >>= 1, x = (ll)x * x % mod)
        if (p & 1)
            res = (ll)res * x % mod;
    return res;
}
inline int fix(const int x) { return x >= mod ? x - mod : x; }
void dft(poly &A, int n) {
    static ull W[N << 1], *H[30], *las = W, mx = 0;
    for (; mx < n; mx++) {
        H[mx] = las;
        ull w = 1, wn = power(g, (mod - 1) >> (mx + 1));
        REP(i, 1 << n) *las++ = w, w = w * wn % mod;
    }
    if (A.size() != (1 << n))
        A.resize(1 << n);
```

```
    static ull a[N];
    for (int i = 0, j = 0; i < (1 << n); ++i) {
        a[i] = A[j];
        for (int k = 1 << (n - 1); (j ^= k) < k; k >>= 1)
            ;
    }
    for (int k = 0, d = 1; k < n; k++, d <<= 1)
        for (int i = 0; i < (1 << n); i += (d << 1)) {
            ull *l = a + i, *r = a + i + d, *w = H[k], t;
            for (int j = 0; j < d; j++, l++, r++) {
                t = (*r) * (*w++) % mod;
                *r = *l + mod - t, *l += t;
            }
        }
    REP(i, 1 << n) A[i] = a[i] % mod;
}
void idft(poly &a, int n) {
    a.resize(1 << n), reverse(a.begin() + 1, a.end());
    dft(a, n);
    int inv = power(1 << n, mod - 2);
    REP(i, 1 << n) a[i] = (ll)a[i] * inv % mod;
}
poly FIX(poly a) {
    while (!a.empty() && !a.back()) a.pop_back();
    return a;
}
poly add(poly a, poly b, int op = 0) {
    a.resize(max(a.size(), b.size()));
    REP(i, b.size()) a[i] = fix(op ? a[i] + mod - b[i] : a[i] + b[i]);
    return FIX(a);
}
poly mul(poly a, poly b, int t = 1) {
    if (t == 1 && a.size() + b.size() <= 24) {
        poly c(a.size() + b.size(), 0);
        REP(i, a.size()) REP(j, b.size()) c[i + j] = (c[i + j] + (ll)a[i] * b[j]) % mod;
        return FIX(c);
    }
    int n = 1, aim = a.size() * t + b.size();
    while ((1 << n) <= aim) n++;
    dft(a, n), dft(b, n);
    if (t == 1)
        REP(i, 1 << n) a[i] = (ll)a[i] * b[i] % mod;
    else
        REP(i, 1 << n) a[i] = (ll)a[i] * a[i] % mod * b[i] % mod;
    idft(a, n), a.resize(aim);
    return FIX(a);
}
poly mul(poly a, int b) {
    REP(i, a.size()) a[i] = (ll)a[i] * b % mod;
    return FIX(a);
}

poly inv(poly a, int n) {  // a[0] != 0
    a.resize(n);
    poly b;
    if (n == 1) {
        b.pb(power(a[0], mod - 2));
        return b;
    }
    b = inv(a, n + 1 >> 1);
    b = add(mul(b, 2), mul(b, a, 2), 1);
```

```
344     return b.resize(n), b;
345 }
346
347 poly Der(poly a) {
348     REP(i, a.size() - 1) a[i] = (ll)(i + 1) * a[i + 1] % mod;
349     return a.pop_back(), a;
350 }
351 poly Int(poly a) {
352     static int inv[N];
353     inv[1] = 1;
354     a.pb(0);
355     rep(i, 2, a.size()) inv[i] = (ll)(mod - mod / i) * inv[mod % i] % mod;
356     per(i, a.size() - 1, 1) a[i] = (ll)a[i - 1] * inv[i] % mod;
357     return a[0] = 0, a;
358 }
359 poly Ln(poly a, int n) {  // a[0] = 1
360     a = mul(Der(a), inv(a, n)), a.resize(n - 1);
361     return FIX(Int(a));
362 }
363 poly Exp(poly a, int n) {  // a[0] = 0
364     a.resize(n);
365     poly b, one(1, 1);
366     if (n == 1)
367         return one;
368     b = Exp(a, n + 1 >> 1);
369     b = mul(b, add(add(a, Ln(b, n), 1), one));
370     return b.resize(n), b;
371 }
372
373 poly Div(poly a, poly b) {
374     poly c;
375     int n = a.size() - 1, m = b.size() - 1;
376     if (n < m)
377         return c;
378     reverse(a.begin(), a.end());
379     a.resize(n - m + 1);
380     reverse(b.begin(), b.end());
381     b.resize(n - m + 1);
382     c = mul(a, inv(b, n - m + 1));
383     c.resize(n - m + 1);
384     return reverse(c.begin(), c.end()), c;
385 }
386 poly Mod(poly a, poly b) { return FIX(add(a, mul(Div(a, b), b), 1)); }
387 inline int chk(int x) { return power(x, (mod - 1) / 2) == 1; }
388 inline int R() { return rand() % mod; }
389 inline pii mul(pii a, pii b, int w) {
390     return mp(((ll)a.fi * b.fi + (ll)a.se * b.se % mod * w) % mod, ((ll)a.fi * b.se + (
    ll)a.se * b.fi) % mod);
391 }
392 inline int Sqrt(int x) {
393     if (!chk(x))
394         return -1;
395     int a = R();
396     while (chk(((ll)a * a - x + mod) % mod)) a = R();
397     int w = ((ll)a * a - x + mod) % mod, p = (mod + 1) / 2;
398     pii res = mp(1, 0), t = mp(a, 1);
399     for (; p; p >>= 1, t = mul(t, t, w))
400         if (p & 1)
401             res = mul(res, t, w);
402     assert(!res.se);
403     return min(res.fi, mod - res.fi);
```

```
}
poly Sqrt(poly a, int n) {
    if (n == 1) {
        poly b(1, Sqrt(a[0]));
        return b;
    }
    a.resize(n);
    poly b = Sqrt(a, n + 1 >> 1);
    b = mul(add(b, mul(a, inv(b, n))), (mod + 1) / 2);
    return b.resize(n), b;
}
poly fastpow(poly a, ll k, int n) {
    a.resize(n), a = FIX(a);
    if (!a.size())
        return a;
    int st = 0, base = 0;
    while (!a[st]) ++st;
    if (st * k >= n)
        return a.resize(0), a;
    REP(i, a.size() - st) a[i] = a[i + st];
    if (st)
        a.resize(a.size() - st);
    base = a[0];
    ll inv = power(base, mod - 2);
    REP(i, a.size()) a[i] = a[i] * inv % mod;
    a = FIX(Exp(mul(Ln(a, n), k % mod), n));
    ;
    if (st) {
        reverse(a.begin(), a.end());
        a.resize(a.size() + st * k);
        reverse(a.begin(), a.end());
        a.resize(n), a = FIX(a);
    }
    base = power(base, k);
    REP(i, a.size()) a[i] = (ll)a[i] * base % mod;
    return FIX(a);
}
}  // namespace Poly
using namespace Poly;
int main() {
    int n = buf + 1, K = buf;
    poly a(n, 0), b;
    REP(i, n) a[i] = buf;
    b = add(a, Exp(Int(inv(Sqrt(a, n), n)), n), 1);
    b[0] = (b[0] + 2 - a[0] + mod) % mod;
    a = Ln(b, n);
    a[0]++;
    b = Der(fastpow(a, K, n));
    b.resize(n - 1);
    REP(i, n - 1) buf.out(b[i]);
    return 0;
}
```

4.1.8 Lagrange Polynomial

$$L(x) = \sum_{i=0}^{n} y_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

```cpp
//O(n^2)
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
typedef pair<int, int> P;
const int MAXN = 3005, mod = 998244353;
int exgcd(int a, int b, int &x, int &y) {
    int d = a;
    if(b != 0) {
        d = exgcd(b, a % b, y, x);
        y -= (a / b) * x;
    }
    else {
        x = 1; y = 0;
    }
    return d;
}
int inv(int a) {
    int x, y;
    exgcd(a, mod, x, y);
    return (x % mod + mod) % mod;
}
struct Lagrange {
    int n, a[MAXN][2];
    void init() {
        for(int i = 0; i <= n; i++) a[i][0] = a[i][1] = 0;
        n = 0;
        a[0][1] = 1;
    }
    int query(int x, int q = 0) {
        int res = 0;
        for(int i = n; i >= 0; i--) res = ((LL)res * x + a[i][q]) % mod;
        return res;
    }
    void update(int x, int y) {
        a[n][0] = 0;
        int v = (LL)(y - query(x) + mod) % mod * inv(query(x, 1)) % mod;
        for(int i = 0; i <= n; i++) a[i][0] = (a[i][0] + (LL)a[i][1] * v) % mod;
        a[++n][1] = 0;
        for(int i = n; i; i--) a[i][1] = (a[i - 1][1] + (LL)a[i][1] * (mod - x)) % mod;
        a[0][1] = (LL)a[0][1] * (mod - x) % mod;
    }
}p;
int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
    int Q;
    cin >> Q;
    int op, x, y;
    p.n = 0;
    p.init();
    while(Q--) {
        cin >> op >> x;
        if(op == 1) {
            cin >> y;
            p.update(x, y);
        }
        else cout << p.query(x) << endl;
    }
    return 0;
}
//f(x) = y, 求f(z_i)
```

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
int n, m;
vector<int> x, y, z;
namespace Poly{
    const int P = 998244353;
    vector<int> ans;// for Evaluate()
    vector<vector<int>> p;// for Evaluate() & Interpolate()
    inline int Pow(ll x, int y=P-2){ // x^y
        int ans=1;
        for(; y; y>>=1, x=x*x%P) if(y&1) ans=ans*x%P;
        return ans;
    }
    inline int Ge(int x){ int n=1; while(n<=x) n<<=1; return n;}
    inline int Mod(int x){ return x<P?x:x-P;}
    inline void NTT(vector<int> &f, int g, int n){
        f.resize(n);
        for(int i=0, j=0; i<n; ++i){
            if(i>j) swap(f[i], f[j]);
            for(int k=n>>1; (j^=k)<k; k>>=1);
        }
        vector<int> w(n>>1);
        for(int i=1; i<n; i<<=1){
            for(int j=w[0]=1, w0=(g==1?Pow(3, (P-1)/i/2):Pow(Pow(3, (P-1)/i/2))); j<i; ++j) w[j]=(ll)w[j-1]*w0%P;
            for(int j=0; j<n; j+=i<<1){
                for(int k=j; k<j+i; ++k){
                    int t=(ll)f[k+i]*w[k-j]%P;
                    f[k+i]=Mod(f[k]-t+P);
                    f[k]=Mod(f[k]+t);
                }
            }
        }
        if(g==-1) for(int i=0, I=Pow(n); i<n; ++i) f[i]=(ll)f[i]*I%P;
    }
    inline vector<int> Add(const vector<int> &f, const vector<int> &g){
        vector<int> ans=f;
        for(unsigned i=0; i<f.size(); ++i) (ans[i]+=g[i])%=P;
        return ans;
    }
    inline vector<int> Mul(const vector<int> &f, const vector<int> &g){// f*g
        vector<int> F=f, G=g;
        int p=Ge(f.size()+g.size()-2);
        NTT(F, 1, p), NTT(G, 1, p);
        for(int i=0; i<p; ++i) F[i]=(ll)F[i]*G[i]%P;
        NTT(F, -1, p);
        return F.resize(f.size()+g.size()-1), F;
    }
    inline vector<int> PolyInv(const vector<int> &f, int n=-1){// 1/f
        if(n==-1) n=f.size();
        vector<int> ans;
        if(n==1) return ans.push_back(Pow(f[0])), ans;
        ans=PolyInv(f, (n+1)/2);
        vector<int> tmp(&f[0], &f[0]+n);
        int p=Ge(n*2-2);
        NTT(tmp, 1, p), NTT(ans, 1, p);
        for(int i=0; i<p; ++i) ans[i]=(2-(ll)ans[i]*tmp[i]%P+P)*ans[i]%P;
        NTT(ans, -1, p);
        return ans.resize(n), ans;
    }
```

```
122     inline void PolyDiv(const vector<int> &a, const vector<int> &b, vector<int> &d,
        vector<int> &r){//a=d*b+r
123         if(b.size()>a.size()) return (void)(d.clear(), r=a);
124 
125         vector<int> A=a, B=b, iB;
126         int n=a.size(), m=b.size();
127         reverse(A.begin(), A.end()), reverse(B.begin(), B.end());
128         B.resize(n-m+1), iB=PolyInv(B, n-m+1);
129         d=Mul(A, iB);
130         d.resize(n-m+1), reverse(d.begin(), d.end());
131 
132         r=Mul(b, d);
133         for(int i=0; i<m-1; ++i) r[i]=(P+a[i]-r[i])%P;
134         r.resize(m-1);
135     }
136     inline vector<int> Derivative(const vector<int> &a){ // a'
137         vector<int> ans;
138         ans.resize(a.size()-1);
139         for(unsigned i=1; i<a.size(); ++i) ans[i-1]=(ll)a[i]*i%P;
140         return ans;
141     }
142     void Evaluate_Interpolate_Init(int l, int r, int t, const vector<int> &a){
143         if(l==r) return p[t].clear(), p[t].push_back(P-a[l]), p[t].push_back(1);
144         int mid=(l+r)/2, k=t<<1;
145         Evaluate_Interpolate_Init(l, mid, k, a), Evaluate_Interpolate_Init(mid+1, r, k
        |1, a);
146         p[t]=Mul(p[k], p[k|1]);
147     }
148     inline void Evaluate(int l, int r, int t, const vector<int> &f, const vector<int> &a
        ){
149         if(r-l+1<=512){
150             for(int i=l; i<=r; ++i){
151                 int x=0, j=f.size(), a1=a[i], a2=(ll)a[i]*a[i]%P, a3=(ll)a[i]*a2%P, a4=(
            ll)a[i]*a3%P, a5=(ll)a[i]*a4%P, a6=(ll)a[i]*a5%P, a7=(ll)a[i]*a6%P, a8=(ll)a[i]*a7%P
            ;
152                 while(j>=8)
153                 x=((ll)x*a8+(ll)f[j-1]*a7+(ll)f[j-2]*a6+(ll)f[j-3]*a5+(ll)f[j-4]*a4+(ll)
            f[j-5]*a3+(ll)f[j-6]*a2+(ll)f[j-7]*a1+f[j-8])%P, j-=8;
154                 while(j--) x=((ll)x*a[i]+f[j])%P;
155                 ans.push_back(x);
156             }
157             return;
158         }
159         vector<int> tmp;
160         PolyDiv(f, p[t], tmp, tmp);
161         Evaluate(l, (l+r)/2, t<<1, tmp, a), Evaluate((l+r)/2+1, r, t<<1|1, tmp, a);
162     }
163     inline vector<int> Evaluate(const vector<int> &f, const vector<int> &a, int flag=-1)
        {// f(a_i)
164         if(flag==-1) p.resize(a.size()<<2), Evaluate_Interpolate_Init(0, a.size()-1, 1,
        a);
165         ans.clear(), Evaluate(0, a.size()-1, 1, f, a);
166         return ans;
167     }
168     vector<int> Interpolate(int l, int r, int t, const vector<int> &x, const vector<int>
         &f){
169         if(l==r){
170             vector<int> ans;
171             return ans.push_back(f[l]), ans;
172         }
173         int mid=(l+r)/2, k=t<<1;
```

```
174         return Add(Mul(Interpolate(l, mid, k, x, f), p[k|1]), Mul(Interpolate(mid+1, r,
        k|1, x, f), p[k]));
175     }
176     inline vector<int> Interpolate(const vector<int> &x, const vector<int> &y){// (x_i,
        y_i)
177         int n=x.size();
178         p.resize(n<<2), Evaluate_Interpolate_Init(0, n-1, 1, x);
179         vector<int> f=Evaluate(Derivative(p[1]), x, 0);
180         for(int i=0; i<n; ++i) f[i]=(ll)y[i]*Pow(f[i])%P;
181         return Interpolate(0, n-1, 1, x, f);
182     }
183 }
184 using namespace Poly;
185 int main() {
186     cin >> n; x.resize(n), y.resize(n);
187     for(int i=0; i<n; ++i) cin >> x[i], cin >> y[i];
188     cin >> m, z.resize(m);
189     for(int i=0; i<m; ++i) cin >> z[i];
190     x=Evaluate(Interpolate(x, y), z);
191     for(int i:x) cout << i << " ";
192     return 0;
193 }
```

4.1.9 BM Alogrithm

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define rep(i,a,n) for (int i=a;i<n;i++)
4  #define per(i,a,n) for (int i=n-1;i>=a;i--)
5  #define pb push_back
6  #define mp make_pair
7  #define all(x) (x).begin(),(x).end()
8  #define fi first
9  #define se second
10 #define SZ(x) ((int)(x).size())
11 typedef vector<int> VI;
12 typedef long long ll;
13 typedef pair<int,int> PII;
14 const ll mod=1000000007;
15 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;}
16 // head
17 namespace linear_seq {
18     const int N=10010;
19     ll res[N],base[N],_c[N],_md[N];
20
21     vector<int> Md;
22     void mul(ll *a,ll *b,int k) {
23         rep(i,0,k+k) _c[i]=0;
24         rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
25         for (int i=k+k-1;i>=k;i--) if (_c[i])
26             rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
27         rep(i,0,k) a[i]=_c[i];
28     }
29     int solve(ll n,VI a,VI b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]+...
30 //        printf("%d\n",SZ(b));
31         ll ans=0,pnt=0;
32         int k=SZ(a);
33         assert(SZ(a)==SZ(b));
34         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() {
    while (~scanf("%d",&n)) {
        vector<int>v;
        v.push_back(1);
        v.push_back(2);
        v.push_back(4);
        v.push_back(7);
        v.push_back(13);
        v.push_back(24);
        //VI{1,2,4,7,13,24}
        printf("%d\n",linear_seq::gao(v,n-1));
    }
}
```

4.2 Math Theory

4.2.1 Inverse

```
//O(logn)求n的逆元
const int mod = 1e6 + 3;
int exgcd(int a, int b, int &x, int &y) {
    int d = a;
    if(b != 0) {
        d = exgcd(b, a % b, y, x);
        y -= (a / b) * x;
    }
    else {
        x = 1; y = 0;
    }
    return d;
}
int inverse(int a) {
    int x, y;
    exgcd(a, mod, x, y);
    return (x % mod + mod) % mod;
}
int inverse(int a) { return qpow(a, mod - 2); }
//O(n)求1~n的逆元
int inv[MAXN];
void init() {
    inv[0] = inv[1] = 1;
    for(int i = 2; i < MAXN; i++) inv[i] = (long long)(mod - mod / i) * inv[mod % i] %
    mod;
}
```

4.2.2 Lucas

```
//mod很小可以预处理逆元的情况
void init() {
    fac[0] = 1;
    for(int i = 1; i < mod; i++) fac[i] = (long long)fac[i - 1] * i % mod;
    inv[0] = inv[1] = 1;
    for(int i = 2; i < mod; i++) inv[i] = (long long)(mod - mod / i) * inv[mod % i] %
    mod;
    for(int i = 1; i < mod; i++) inv[i] = (long long)inv[i] * inv[i - 1] % mod;
}
int C(int a, int b) {
    if(b > a) return 0;
    if(a < mod) return (long long)fac[a] * inv[b] % mod * inv[a - b] % mod;
    return (long long)C(a / mod, b / mod) * C(a % mod, b % mod) % mod;
}
//mod过大不能预处理逆元的情况
LL qpow(LL x, LL y) {
    LL res = 1;
    while(y) {
        if(y & 1) res = res * x % mod;
        x = x * x % mod;
        y >>= 1;
    }
    return res;
}
LL C(LL a, LL b) {
    if(b > a) return 0;
```

```
    if(b > a - b) b = a - b;
    LL s1 = 1, s2 = 1;
    for(LL i = 0; i < b; i++) {
        s1 = s1 * (a - i) % mod;
        s2 = s2 * (i + 1) % mod;
    }
    return s1 * qpow(s2, mod - 2) % mod;
}
LL lucas(LL a, LL b) {
    if(a < mod) return C(a, b);
    return lucas(a / mod, b / mod) * C(a % mod, b % mod);
}
```

4.2.3 CRT && exCRT

$$x \equiv a_i \pmod{m_i}$$

```
namespace CRT {
    LL m[MAXN], a[MAXN];
    LL exgcd(LL _a, LL _b, LL &x, LL &y) {
        if(!_b) {
            x = 1; y = 0;
            return _a;
        }
        LL d = exgcd(_b, _a % _b, y, x);
        y -= (_a / _b) * x;
        return d;
    }
    LL crt(int n) {
        LL M = 1, tmp, res = 0, x, y;
        for(int i = 1; i <= n; i++) M *= m[i];
        for(int i = 1; i <= n; i++) {
            tmp = M / m[i];
            exgcd(tmp, m[i], x, y);
            x = (x + m[i]) % m[i];
            res = (a[i] * x % M * tmp % M + res) % M;
        }
        return res;
    }
}
namespace EXCRT {
    LL m[MAXN], a[MAXN];
    LL exgcd(LL _a, LL _b, LL &x, LL &y) {
        if(!_b) {
            x = 1; y = 0;
            return _a;
        }
        LL d = exgcd(_b, _a % _b, y, x);
        y -= (_a / _b) * x;
        return d;
    }
    LL excrt(int n) {
        LL M = m[1], A = a[1], x, y, d, tmp;
        for(int i = 2; i <= n; i++) {
            d = exgcd(M, m[i], x, y);
            if((A - a[i]) % d) return -1; //No solution
            tmp = M / d; M *= m[i] / d;
            y = (A - a[i]) / d % M * y % M;
```

```
            y = (y + tmp) % tmp;
            A = (m[i] % M * y % M + a[i]) % M;
            A = (A + M) % M;
        }
        return A;
    }
        LL inv(LL _a, LL _b) {
        LL x, y;
        exgcd(_a, _b, x, y);
        return (x % _b + _b) % _b;
    }
    LL excrt(int n) {
        LL M = m[1], A = a[1], x, y, d, c, tmp;
        for(int i = 2; i <= n; i++) {
            d = exgcd(M, m[i], x, y);
            c = a[i] - A;
            if(c % d) return -1;
            c = (c % m[i] + m[i]) % m[i];
            M /= d; m[i] /= d;
            c = c / d * inv(M % m[i], m[i]) % m[i];
            tmp = M;
            M *= m[i] * d;
            A = (c * tmp % M * d % M + A) % M;
        }
        return A;
    }
}
```

4.2.4 BSGS

```
const int MOD = 76543;
int hs[MOD + 5], head[MOD + 5], nxt[MOD + 5], id[MOD + 5], ecnt;
void insert(int x, int y) {
    int k = x % MOD;
    hs[ecnt] = x, id[ecnt] = y, nxt[ecnt] = head[k], head[k] = ecnt++;
}
int find(int x) {
    int k = x % MOD;
    for(int i = head[k]; i; i = nxt[i])
        if(hs[i] == x)
            return id[i];
    return -1;
}
int BSGS(int a, int b, int c){
    memset(head, 0, sizeof head); ecnt = 1;
    if(b == 1) return 0;
    int m = sqrt(c * 1.0), j;
    LL x = 1, p = 1;
    for(int i = 0; i < m; i++, p = p * a % c)
        insert(p * b % c, i);
    for(LL i = m; ;i += m){
        if((j = find(x = x * p % c)) != -1) return i - j;
        if(i > c) break;
    }
    return -1;
}
```

4.2.5 Miller-Rabin && PollardRho

```
LL ksc(LL a,LL n,LL mod){
    LL ret=0;
    for(;n;n>>=1){
        if(n&1){ret+=a;if(ret>=mod)ret-=mod;}
        a<<=1;if(a>=mod)a-=mod;
    }
    return ret;
}
LL ksm(LL a,LL n,LL mod){
    LL ret = 1;
    for(;n;n>>=1){
        if(n&1)ret=ksc(ret,a,mod);
        a=ksc(a,a,mod);
    }
    return ret;
}
int millerRabin(LL n){
    if(n<2 || (n!=2 && !(n&1)))return 0;
    LL d=n-1;for(;!(d&1);d>>=1);
    for(int i=0;i<20;++i){
        LL a=rand()%(n-1)+1;
        LL t=d,m=ksm(a,d,n);
        for(;t!=n-1 && m!=1 && m!=n-1;m=ksc(m,m,n),t<<=1);
        if(m!=n-1 && !(t&1)) return 0;
    }
    return 1;
}
LL cnt,fact[100];
LL gcd(LL a,LL b){return !b?a:gcd(b,a%b);}
LL pollardRho(LL n, int a){
    LL x=rand()%n,y=x,d=1,k=0,i=1;
    while(d==1){
        ++k;
        x=ksc(x,x,n)+a;if(x>=n)x-=n;
        d=gcd(x>y?x-y:y-x,n);
        if(k==i){y=x;i<<=1;}
    }
    if(d==n)return pollardRho(n,a+1);
    return d;
}
void findfac(LL n){
    if(millerRabin(n)){fact[++cnt]=n;return;}
    LL p=pollardRho(n,rand()%(n-1)+1);
    findfac(p);
    findfac(n/p);
}
```

4.2.6 $\varphi(n)$

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

4.2.7 Euler Sieve

```
int prime[MAXN], cnt, phi[MAXN], mu[MAXN];
bool isp[MAXN];

int min_pow[MAXN];   //最小质因子最高次幂
int min_sum[MAXN];   //1+p+p^2+...+p^k
int div_sum[MAXN];   //约数和

int min_index[MAXN]; //最小质因子的指数
int div_num[MAXN];   //约数个数
void Euler(int n) {
    mu[1] = phi[1] = div_num[1] = div_sum[1] = 1;
    for(int i = 2; i <= n; i++) {
        if(!isp[i]) {
            prime[++cnt] = min_pow[i] = i;
            phi[i] = i - 1;
            mu[i] = -1;
            min_index[i] = 1; div_num[i] = 2;
            div_sum[i] = min_sum[i] = i + 1;
        }
        for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
            isp[i * prime[j]] = 1;
            if(i % prime[j] == 0) {
                phi[i * prime[j]] = phi[i] * prime[j];
                mu[i * prime[j]] = 0;

                min_index[i * prime[j]] = min_index[i] + 1;
                div_num[i * prime[j]] = div_num[i] / (min_index[i] + 1) * (min_index[i * prime[j]] + 1);

                min_sum[i * prime[j]] = min_sum[i] + min_pow[i] * prime[j];
                div_sum[i * prime[j]] = div_sum[i] / min_sum[i] * min_sum[i * prime[j]];
                min_pow[i * prime[j]] = min_pow[i] * prime[j];
                break;
            }
            phi[i * prime[j]] = phi[i] * (prime[j] - 1);
            mu[i * prime[j]] = -mu[i];

            div_num[i * prime[j]] = div_num[i] << 1;
            min_index[i * prime[j]] = 1;

            div_sum[i * prime[j]] = div_sum[i] * (prime[j] + 1);
            min_pow[i * prime[j]] = prime[j];
            min_sum[i * prime[j]] = prime[j] + 1;
        }
    }
}
```

4.2.8 DuJiao Sieve

$$\sum_{i=1}^{n} \varphi(i)$$

```
vector<int> prime;
int phi[MAXN], P[MAXN];
bool isp[MAXN];
unordered_map<LL, int> mp;
```

```
void Euler(int n) {
    phi[1] = 1;
    for(int i = 2; i <= n; i++) {
        if(!isp[i]) {
            prime.push_back(i);
            phi[i] = i - 1;
        }
        for(auto x : prime) {
            if(i * x > n) break;
            isp[i * x] = 1;
            if(i % x == 0) {
                phi[i * x] = phi[i] * x;
                break;
            }
            phi[i * x] = phi[i] * (x - 1);
        }
    }
    for(int i = 1; i <= n; i++) P[i] = (P[i - 1] + phi[i]) % mod;
}
LL cal(LL n) {
    if(n < MAXN) return P[n];
    if(mp.count(n)) return mp[n];
    LL res = 0;
    for(LL i = 2, last; i <= n; i = last + 1) {
        last = n / (n / i);
        res += (last - i + 1) % mod * cal(n / i) % mod;
        res %= mod;
    }
    mp[n] = ((__int128)n * (n + 1) / 2 % mod + mod - res) % mod;
    return mp[n];
}
```

$$\sum_{i=1}^{n} \mu(i)$$

```
LL cal(LL n) {
    if(n < MAXN) return M[n];
    if(mp.count(n)) return mp[n];
    LL res = 0;
    for(LL i = 2, last; i <= n; i = last + 1) {
        last = n / (n / i);
        res += (last - i + 1) * cal(n / i);
    }
    mp[n] = 1 - res;
    return 1 - res;
}
```

4.2.9 Min_25 Sieve

思路为把结果分为 i 为质数的和，i 为合数的和，i=1 的和

g(n, j) 表示从 1 累加到 n 的 f(i)，其中的 i 满足要么 i 自己是质数，要么 i 的最小质因子大于第 j 个质数

要求 $f(p)(p\ is\ prime)$ 可被多项式表示,$f(p^k)$ 可快速计算

分别计算多项式 x^0, x^1, x^2 的 g(n,j) 和 h(n,j)

h 由欧拉筛计算，递归起点 g(n, 0)

公式

$$\sum_{i}^{n} f(i)$$

$$g(n,j)=\begin{cases} g(n,j-1) & p_j^2>n \\ g(n,j-1)-f(p_j)[g(\frac{n}{p_j},j-1)-g(p_j-1,j-1)] & p_j^2\le n \end{cases}$$

其中

$$g(p_j-1,j-1)=\sum_{i}^{j-1}f(p_i)$$

可用 h 表示

S(n,j) 表示从 1 累加到 n 的 f(i)，同样的，i 满足要么 i 自己是质数，要么 i 的最小质因子大于第 j 个质数和 g 顺序相反，最后答案为 S(n, 1)，递推式为

$$S(n,j)=g(n,|P|)-\sum_{i=1}^{j}f(p_i)+\sum_{k\ge j}\sum_{p_k^{e+1}<n}(f(p_k^e)S(\frac{n}{p_k^e},k+1)+f(p_k^{e+1}))$$

前一部分 $g(n,|P|)-\sum_{i=1}^{j}f(p_i)$ 是相应的质数部分和。

例题：欧拉函数前缀和

$$\sum_{i=1}^{n}\varphi(i)$$

$$g_{k,n}\ and\ h_{k,n}\ Count$$

$$\sum_{i=1}^{n}i^k$$

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int MAXN = 1e6 + 5, mod = 1e9 + 7;
const int inv2 = (mod + 1) / 2, inv6 = (mod + 1) / 6;
int prime[MAXN], isp[MAXN], cnt;
LL g[3][MAXN << 1], h[3][MAXN << 1];
LL w[MAXN << 1];
int id1[MAXN], id2[MAXN];
inline int MOD(LL x) { return x >= mod ? x - mod : x; }
//inline int MOD(LL x) { return x % mod; }
inline int add(LL x, LL y) { return MOD(MOD(x) + MOD(y)); }
void Euler(int n) {
    for(int i = 2; i <= n; i++) {
        if(!isp[i]) {
            prime[++cnt] = i;
            h[0][cnt] = h[0][cnt - 1] + 1;
            h[1][cnt] = add(h[1][cnt - 1], i);
            h[2][cnt] = add(h[2][cnt - 1], (LL)i * i % mod);
        }
        for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
            isp[i * prime[j]] = 1;
            if(i % prime[j] == 0) {
                break;
            }
        }
    }
}
LL n;
int sz, m;
inline int id(LL x) {
```

```
    return x <= sz ? id1[x] : id2[n / x];
}
//f(p ^ k)
inline int f(int p, LL pk) {
    return pk / p * (p - 1) % mod;
}
LL S(LL x, int y) {
    if(x <= 1 || prime[y] > x) return 0;
    //G(x) - H(j - 1) (first part)
    LL res = add(add(g[1][id(x)], mod - g[0][id(x)]), mod - add(h[1][y - 1], mod - h[0][y - 1]));
    for(int j = y, k = 1; j <= cnt && (LL)prime[j] * prime[j] <= x; j++, k = 1) {
        for(LL pk = prime[j]; pk * prime[j] <= x; pk *= prime[j], k++) {
            res = add(res, S(x / pk, j + 1) * f(prime[j], pk) % mod + f(prime[j], pk * prime[j]));
        }
    }
    return res;
}
int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
    cin >> n;
    sz = sqrt(n);
    Euler(sz);
    for(LL i = 1, last, t; i <= n; i = last + 1) {
        last = n / (n / i);
        w[++m] = n / i, t = n / i % mod;
        w[m] <= sz ? id1[w[m]] = m : id2[last] = m;
        g[0][m] = MOD(t + mod - 1);
        g[1][m] = add(t * (t + 1) % mod * inv2 % mod, mod - 1);
        g[2][m] = add((2 * t + 1) % mod * t * (t + 1) % mod * inv6 % mod, mod - 1);
    }
    for(int j = 1; j <= cnt; j++) {
        for(int i = 1; i <= m && (LL)prime[j] * prime[j] <= w[i]; i++) {
            g[0][i] = MOD(g[0][i] + mod - (g[0][id(w[i] / prime[j])] - h[0][j - 1]));
            g[1][i] = MOD(g[1][i] + mod - ((LL)prime[j] * MOD(g[1][id(w[i] / prime[j])] + mod - h[1][j - 1]) % mod));
            g[2][i] = MOD(g[2][i] + mod - ((LL)prime[j] * prime[j] % mod * MOD(g[2][id(w[i] / prime[j])] + mod - h[2][j - 1]) % mod));
        }
    }
    //S(n, 1) + F(1);
    LL ans = MOD(S(n, 1) + 1);
    cout << ans << endl;
    return 0;
}
```

$$\sum_{i=1}^{n} \mu(i)$$

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int MAXN = 1e6 + 5;
int prime[MAXN], isp[MAXN], cnt;
LL g[3][MAXN << 1], h[3][MAXN << 1];
LL w[MAXN << 1];
int id1[MAXN], id2[MAXN];
void Euler(int n) {
    for(int i = 2; i <= n; i++) {
        if(!isp[i]) {
```

```
            prime[++cnt] = i;
            h[0][cnt] = h[0][cnt - 1] + 1;
        }
        for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
            isp[i * prime[j]] = 1;
            if(i % prime[j] == 0) {
                break;
            }
        }
    }
}
LL a, b;
LL n;
int sz, m;
inline int id(LL x) {
    return x <= sz ? id1[x] : id2[n / x];
}
//f(p ^ k)
inline int f(int p, int k) {
    return k == 1 ? -1 : 0;
}
LL S(LL x, int y) {
    if(x <= 1 || prime[y] > x) return 0;
    //g(x) - h(j - 1)
    LL res = - g[0][id(x)] + h[0][y - 1];
    for(int j = y, k = 1; j <= cnt && (LL)prime[j] * prime[j] <= x; j++, k = 1) {
        for(LL pk = prime[j]; pk * prime[j] <= x; pk *= prime[j], k++) {
            res += S(x / pk, j + 1) * f(prime[j], k) + f(prime[j], k + 1);
        }
    }
    return res;
}
LL cal(LL x) {
    n = x;
    m = 0;
    sz = sqrt(n);
    for(LL i = 1, last, t; i <= n; i = last + 1) {
        last = n / (n / i);
        w[++m] = n / i, t = n / i;
        w[m] <= sz ? id1[w[m]] = m : id2[last] = m;
        g[0][m] = t - 1;
    }
    for(int j = 1; j <= cnt; j++) {
        for(int i = 1; i <= m && (LL)prime[j] * prime[j] <= w[i]; i++) {
            g[0][i] = g[0][i] - (g[0][id(w[i] / prime[j])] - h[0][j - 1]);
        }
    }
    return S(x, 1) + 1;
}
int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
    cin >> a >> b;
    Euler(sqrt(b));

    //S(n, 1) + F(1);
    cout << cal(b) - cal(a - 1) << endl;
    return 0;
}
```

4.2.10 Möbius Inversion

$$\sum_{i}^{n}\sum_{j}^{m} lcm(i,j) \pmod p$$

```
int mu[MAXN], prime[MAXN], sum[MAXN], cnt;
bool isp[MAXN];
void getmu(int n) {
    mu[1] = 1;
    for(int i = 2; i <= n; i++) {
        if(!isp[i]) {
            mu[i] = -1;
            prime[++cnt] = i;
        }
        for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
            isp[i * prime[j]] = 1;
            if(i % prime[j] == 0) {
                mu[i * prime[j]] = 0;
                break;
            }
            mu[i * prime[j]] = -mu[i];
        }
    }
}
ll n, m, ans;
ll query(ll x, ll y) { return (x * (x + 1) / 2 % mod) * (y * (y + 1) / 2 % mod) % mod; }
ll F(ll x, ll y) {
    ll res = 0, last;
    for(ll i = 1; i <= min(x, y); i = last + 1) {
        last = min(x / (x / i), y / (y / i));
        res = (res + (sum[last] - sum[i - 1]) * query(x / i, y / i) % mod) % mod;
    }
    return res;
}
int main() {
    cin>>n>>m;
    getmu(min(n, m));
    for(ll i = 1; i <= min(n, m); i++) sum[i] = (sum[i - 1] + (i * i * mu[i]) % mod) %
    mod;
    ll last;
    for(ll d = 1; d <= min(n, m); d = last + 1) {
        last = min(n / (n / d), m / (m / d));
        ans = (ans + (last - d + 1) * (d + last) / 2 % mod * F(n / d, m / d) % mod) %
    mod;
    }
    ans = (ans + mod) % mod;
    cout<<ans<<endl;
    return 0;
}
```

5 Geometry

5.1 Commonly Definition and Functions

5.1.1 Const and Functions

```
namespace CG{
    #define Point Vector
    const double pi=acos(-1.0);
    const double inf=1e100;
    const double eps=1e-9;
    template <typename T> inline T Abs(T x){return x>0?x:-x;}
    template <typename T> inline bool operator == (T x,T y){return Abs(x-y)<eps;}
    int sgn(double x){
        if (Abs(x)<eps) return 0;
        if (x>0) return 1;
        else return -1;
    }
}
```

5.1.2 Point Definition

```
namespace CG{
    struct Point{
        double x,y;
        Point(double x=0,double y=0):x(x),y(y){}
    };
    Vector operator + (const Vector a,const Vector b){return Vector(a.x+b.x,a.y+b.y);}
    Vector operator - (const Vector a,const Vector b){return Vector(a.x-b.x,a.y-b.y);}
    Vector operator * (const Vector a,const double k){return Vector(a.x*k,a.y*k);}
    Vector operator / (const Vector a,const double k){return Vector(a.x/k,a.y/k);}
    bool operator < (const Vector a,const Vector b) {return a.x==b.x?a.y<b.y:a.x<b.x;}
    bool operator == (const Vector a,const Vector b) {return a.x==b.x && a.y==b.y;}
    double Dot(const Vector a,const Vector b){return a.x*b.x+a.y*b.y;}
    double Cross(const Vector a,const Vector b){return a.x*b.y-a.y*b.x;}
    double mult_Cross(const Vector a,const Vector b,const Vector c){return (a.x-c.x)*(b.
    y-c.y)-(b.x-c.x)*(a.y-c.y);}
    double mult_Dot(const Vector a,const Vector b,const Vector c){return (a.x-c.x)*(b.x-
    c.x)+(a.y-c.y)*(b.y-c.y);}
    double Norm(const Vector a){return sqrt(Dot(a,a));}
    double Angle(const Vector a,const Vector b){return acos(Dot(a,b)/Norm(a)/Norm(b));}
    Vector Rotate(const Vector a,const double theta){return Vector(a.x*cos(theta)-a.y*
    sin(theta),a.x*sin(theta)+a.y*cos(theta));}
    bool ToLeftTest(const Vector a,const Vector b){return Cross(a,b)<0;}
    double DisPP(const Vector a,const Vector b){return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y
    )*(a.y-b.y));}
}
```

5.1.3 Line Definition

```
namespace CG{
    struct Line{
        Point p0,v,p1;
        double t,theta;
        Line(Point _p0=0,Point _v=0,double _t=1):p0(_p0),v(_v),t(_t){p1=p0+v*t; theta=
    atan2(v.y,v.x);}
```

```
        // Line(Point _p0=0,Point _v=0,double _t=1):p0(_p0),p1(_v){v=(p1-p0)/t; theta=atan2(v.y,v.x);}
    };
    bool operator < (const Line n,const Line m) {return n.theta<m.theta;}
    Point GetIntersection(const Line n,const Line m){return n.p0+n.v*Cross(m.v,(n.p0-m.p0))/Cross(n.v,m.v);}
    bool OnLine(const Vector a,const Line l){return Cross(l.p0-a,l.p1-a)==0;}
    bool OnSegment(const Point a,const Line l){return sgn(Cross(l.p0-a,l.p1-a))==0 && sgn(Dot(l.p0-a,l.p1-a))<0;}
    double DisPL(const Point a,const Line l){return Abs(Cross(l.p1-l.p0,a-l.p0)/Norm(l.p1-l.p0));}
    double DisPS(const Point a,const Line l){
        if (l.p0==l.p1) return Norm(a-l.p0);
        Vector v1=l.p1-l.p0,v2=a-l.p0,v3=a-l.p1;
        if (sgn(Dot(v1,v2))<0) return Norm(v2);
        if (sgn(Dot(v1,v3))>0) return Norm(v3);
        return DisPL(a,l);
    }
    Point GetProjection(const Point a,const Line l){
        Vector v=l.p1-l.p0;
        return l.p0+v*(Dot(v,a-l.p0)/Dot(v,v));
    }
    bool SegmentIntersection(const Line n,const Line m,bool p){
        double c1=Cross(n.p1-n.p0,m.p1-m.p0);
        double c2=Cross(n.p1-n.p0,m.p1-n.p0);
        double c3=Cross(m.p1-m.p0,n.p0-m.p0);
        double c4=Cross(m.p1-m.p0,n.p1-m.p0);
        if (p){
            if (!sgn(c1) || !sgn(c2) || !sgn(c3) || !sgn(c4)){
                return OnSegment(n.p0,m) || OnSegment(n.p1,m) || OnSegment(m.p0,n) || OnSegment(m.p0,m);

            }
        }
        return (sgn(c1)*sgn(c2)<0 && sgn(c3)*sgn(c4)<0);
    }
}
```

5.1.4 Get Area

```
namespace CG{
    double GetArea(Point *p,int n){
        double area=Cross(p[n],p[1]);
        for (int i=2;i<=n;i++) area+=0.5*Cross(p[i-1],p[i]);
        return Abs(area);
    }
}
```

5.1.5 Get Circumference

```
namespace CG{
    double GetCircumference(Point *p,int n){
        double Circumference=DisPP(p[n],p[1]);
        for (int i=2;i<=n;i++) Circumference+=DisPP(p[i-1],p[i]);
        return Circumference;
    }
}
```

5.1.6 Anticlockwise Sort

```
namespace CG{
    \\p为一个凸包,只是不知其点集是否为逆时针
    void clockwise_sort(Point *p,int n){
        for(int i=0;i<n-2;i++){
            double tmp = mult_Cross(p[i+1],p[i+2],p[i]);
            if(tmp>0) return;
            else if(tmp<0){
                reverse(p,p+n);
                return;
            }
        }
    }
}
```

5.2 Convex Hull

5.2.1 Get Convex Hull

```
namespace CG{
    Point p[MAXN],s[MAXN];
    int ConvexHull(Point *p,int n,Point *s){
        sort(p,p+n,cmp); //x从小到大,y从小到大;
        int m=0;
        for (int i=0;i<n;i++){
            for (;m>=2 && Cross(s[m-1]-s[m-2],p[i]-s[m-1])<=0;m--);
            s[++m]=p[i];
        }
        int k=m;
        for (int i=n-2;i;i--){
            for (;m>=k+1 && Cross(s[m-1]-s[m-2],p[i]-s[m-1])<=0;m--);
            s[++m]=p[i];
        }
        return m-1;
    }
}
```

5.2.2 Point in Convex Hull

```
namespace CG{
    bool PointInConvexHull(Point A){
        int l=1,r=tot-2,mid;
        while(l<=r){
            mid=(l+r)>>1;
            double a1=Cross(p[mid]-p[0],A-p[0]);
            double a2=Cross(p[mid+1]-p[0],A-p[0]);
            if(a1>=0 && a2<=0){
                if(Cross(p[mid+1]-p[mid],A-p[mid])>=0) return true;
                return false;
            }
            else if(a1<0) r=mid-1;
            else l=mid+1;
        }
        return false;
    }
}
```

5.3 Minkowski Sum

```
namespace CG{
    void Minkowski(Point *C1,int n,Point *C2,int m){
        for(int i=1;i<=n;i++) s1[i]=C1[i]-C1[i-1];
        for(int i=1;i<=m;i++) s2[i]=C2[i]-C2[i-1];
        A[tot=1]=C1[1]+C2[1];
        int p1=1,p2=1;
        while (p1<=n && p2<=m) ++tot,A[tot]=A[tot-1]+(s1[p1]*s2[p2]>=0?s1[p1++]:s2[p2
    ++]);
        while (p1<=n) ++tot,A[tot]=A[tot-1]+s1[p1++];
        while (p2<=m) ++tot,A[tot]=A[tot-1]+s2[p2++];
        tot=ConvexHull(A,tot);
    }
}
```

5.4 Rotating Calipers

5.4.1 The Diameter of Convex Hull

```
namespace CG{
    double RotatingCalipers(Point *p,int n){
        double dis=0;
        for(int i=0,j=2;i<n;++i){
            while (abs(Cross(p[i+1]-p[i],p[j]-p[i]))<abs(Cross(p[i+1]-p[i],p[j+1]-p[i]))
    ) j=(j+1)%n;
            dis=max(dis,max(DisPP(p[j],p[i]),DisPP(p[j],p[i+1])));
        }
        return dis;
    }
}
```

5.4.2 The Min Distance Bewteen two Convex Hull

```
namespace CG{
    ///点c到线段ab的最短距离
    double GetDist(Point a,Point b,Point c){
        if(dis(a,b)<esp) return dis(b,c); ///a,b是同一个点
        if(mult_Dot(b,c,a)<-esp) return dis(a,c); ///投影
        if(mult_Dot(a,c,b)<-esp) return dis(b,c);
        return fabs(mult_Cross(b,c,a)/dis(a,b));

    }
    ///求一条线段ab的两端点到另外一条线段bc的距离，反过来一样，共4种情况
    double MinDist(Point a,Point b,Point c,Point d){
        return min(min(GetDist(a,b,c),GetDist(a,b,d)),min(GetDist(c,d,a),GetDist(c,d,b))
    );
    }
    double RotatingCalipers(Point *p,int n,Point *q,int m){
        int yminP = 0,ymaxQ=0;
        for(int i=1;i<n;i++){  ///找到点集p组成的凸包的左下角
            if(p[i].y<p[yminP].y||(p[i].y==p[yminP].y)&&(p[i].x<p[yminP].x)) yminP = i;
        }
        for(int i=1;i<m;i++){  ///找到点集q组成的凸包的右上角
            if(q[i].y>q[ymaxQ].y||(q[i].y==q[ymaxQ].y)&&(q[i].x>q[ymaxQ].x)) ymaxQ = i;
        }
        double ans = DisPP(p[yminP],q[ymaxQ]); ///距离(yminP,ymaxQ)维护为当前最小值。
```

```
        for(int i=0;i<n;i++){
            double tmp;
            while(tmp=(mult_Cross(q[ymaxQ+1],p[yminP],p[yminP+1])-mult_Cross(q[ymaxQ],p[
    yminP],p[yminP+1]))>esp)
                ymaxQ = (ymaxQ+1)%m;
            if(tmp<-esp) ans = min(ans,GetDist(p[yminP],p[yminP+1],q[ymaxQ]));
            else ans=min(ans,MinDist(p[yminP],p[yminP+1],q[ymaxQ],q[ymaxQ+1]));
            yminP = (yminP+1)%n;
        }
        return ans;
    }
}
```

5.5 Half Plane Intersection

```
namespace CG{
    void HalfPlaneIntersection(Line l[],int n){
        deque <Point> p;
        sort(l+1,l+1+n);
        deque <Line> q;
        q.push_back(l[1]);
        for (int i=2;i<=n;i++){
            for (;!p.empty() && !ToLeftTest(p.back()-l[i].p0,l[i].v);q.pop_back(),p.
    pop_back());
            for (;!p.empty() && !ToLeftTest(p.front()-l[i].p0,l[i].v);q.pop_front(),p.
    pop_front());
            if (sgn(Cross(l[i].v,q.back().v))==0)
                if (ToLeftTest(l[i].p0-q.back().p0),q.back().v){
                    q.pop_back();
                    if (!p.empty()) p.pop_back();
                }
            if (!q.empty()) p.push_back(GetIntersection(q.back(),l[i]));
            q.push_back(l[i]);
        }
        for (;!p.empty() && !ToLeftTest(p.back()-q.front().p0,q.front().v);q.pop_back(),
    p.pop_back());
        p.push_back(GetIntersection(q.back(),q.front()));
        if (p.size() < 3) printf("0\n");
        else{
            cerr << "!" << endl;
            double area = 0.5 * Cross(p.back(), p.front());
            Point last = p.front();
            for (p.pop_front(); !p.empty(); last = p.front(), p.pop_front())
                area += 0.5 * Cross(last, p.front());
            printf("%.8lf\n", fabs(area));
        }
    }
}
```

5.6 Min Circle Cover

```
namespace CG{
    Point GetCircleCenter(const Point a,const Point b,const Point c){
        Point p=(a+b)/2.0,q=(a+c)/2.0;
        Vector v=Rotate(b-a,pi/2.0),w=Rotate(c-a,pi/2.0);
        if (sgn(Norm(Cross(v,w)))==0){
            if (sgn(Norm(a-b)+Norm(b-c)-Norm(a-c))==0) return (a+c)/2;
            if (sgn(Norm(b-a)+Norm(a-c)-Norm(b-c))==0) return (b+c)/2;
```

```
            if (sgn(Norm(a-c)+Norm(c-b)-Norm(a-b))==0) return (a+c)/2;
        }
        return GetIntersection(Line(p,v),Line(q,w));
    }
    void MinCircleCover(Point p[],int n){
        random_shuffle(p+1,p+1+n);
        Point c=p[1];
        double r=0;
        for (int i=2;i<=n;i++)
            if (sgn(Norm(c-p[i])-r)>0){
                c=p[i],r=0;
                for (int j=1;j<i;j++)
                    if (sgn(Norm(c-p[j])-r)>0){
                        c=(p[i]+p[j])/2.0;
                        r=Norm(c-p[i]);
                        for (int k=1;k<j;k++)
                            if (sgn(Norm(c-p[k])-r)>0){
                                c=GetCircleCenter(p[i],p[j],p[k]);
                                r=Norm(c-p[i]);
                            }
                    }
            }
        printf("%.10f\n%.10f %.10f",r,c.x,c.y);
    }
}
```

5.7 Circle Union Area

```
//k次覆盖
//圆并去重后s[0]
typedef pair<double, int> P;
const double pi = acos(-1.0);
const int MAXN = 10003;
P arc[MAXN << 1];
int acnt, cnt;
double s[1003];
bool del[1003];
void add(double st, double en) {
    if(st < -pi) {
        add(st + 2 * pi, pi);
        add(-pi, en);
        return;
    }
    if(en > pi) {
        add(st, pi);
        add(-pi, en - 2 * pi);
        return;
    }
    arc[++acnt] = P(st, 1);
    arc[++acnt] = P(en, -1);
}
double F(double x) {
    return (x - sin(x)) / 2;
}
struct Node {
    int x, y, r;
    Node(int _x = 0, int _y = 0, int _r = 0):x(_x), y(_y), r(_r) {}
    bool operator == (const Node& t) {
        return x == t.x && y == t.y && r == t.r;
```

```
32      }
33      inline void read() {
34          scanf("%d%d%d", &x, &y, &r);
35      }
36  }a[1003];
37  int main() {
38      int n;
39      scanf("%d", &n);
40      for(int i = 1; i <= n; i++) a[i].read();
41      /*
42      //去重
43      int nn = 0;
44      for(int i = 1; i <= n; i++) {
45          bool same = 0;
46          for(int j = 1; j < i; j++) {
47              if(a[i] == a[j]) {
48                  same = 1; break;
49              }
50          }
51          if(!same) a[++nn] = a[i];
52      }
53      n = nn;
54      //去包含
55      for(int i = 1; i <= n; i++) {
56          for(int j = 1; j <= n; j++) if(i != j) {
57              if(hypot(a[i].x - a[j].x, a[i].y - a[j].y) < (double)(a[i].r - a[j].r)) del[
    j] = 1;
58          }
59      }
60      nn = 0;
61      for(int i = 1; i <= n; i++) if(!del[i]) {
62          a[++nn] = a[i];
63      }
64      n = nn;
65      */
66      for(int i = 1; i <= n; i++) {
67          acnt = 0;
68          for(int j = 1; j <= n; j++) if(i != j) {
69              int dis = (a[i].x - a[j].x) * (a[i].x - a[j].x) + (a[i].y - a[j].y) * (a[i].
    y - a[j].y);
70              if(a[j].r > a[i].r && dis <= (a[j].r - a[i].r) * (a[j].r - a[i].r)) add(-pi,
     pi);
71              else if(dis > (a[i].r - a[j].r) * (a[i].r - a[j].r) && dis < (a[i].r + a[j].
    r) * (a[i].r + a[j].r)){
72                  double c = sqrt(dis);
73                  double angle = acos((a[i].r * a[i].r + c * c - a[j].r * a[j].r) / (2 * a
    [i].r * c));
74                  double k = atan2(a[j].y - a[i].y, a[j].x - a[i].x);
75                  add(k - angle, k + angle);
76              }
77          }
78          arc[++acnt] = P(pi, -1);
79          sort(arc + 1, arc + acnt + 1);
80          cnt = 0;
81          double last = -pi;
82          for(int j = 1; j <= acnt; j++) {
83              s[cnt] += F(arc[j].first - last) * a[i].r * a[i].r; //扇形 - 三角形
84              double xa = a[i].x + a[i].r * cos(last);
85              double ya = a[i].y + a[i].r * sin(last);
86              last = arc[j].first;
87              double xb = a[i].x + a[i].r * cos(last);
```

```
            double yb = a[i].y + a[i].r * sin(last);
            s[cnt] += (xa * yb - xb * ya) / 2; //到圓心的三角形面積
            cnt += arc[j].second;
        }
    }
    //printf("%.3f\n", s[0]);
    for (int i = 0; i < n; i++) {
        printf("[%d] = %.3f\n", i + 1, s[i] - s[i + 1]);
    }
    return 0;
}
```

5.8 Simpson Integrate

```
double Simpson(double l,double r){
    return (r-l)*(F(l)+4*F((l+r)/2)+F(r))/6;
}
double Integrate(double l,double r,double S){
    double mid=(l+r)/2;
    double A=Simpson(l,mid);
    double B=Simpson(mid,r);
    if(A+B-S<eps)return S;
    return Integrate(l,mid,A)+Integrate(mid,r,B);
}
```

5.9 Closest Point

```
vector <Point> P;
DB CP(int l, int r) {
    if (l == r) return DB_INF;
    if (l + 1 == r) return DisPP(P[l], P[r]);
    int mid = (l + r) >> 1;
    DB d = min(CP(l, mid), CP(mid + 1, r));
    vector <Point> tmp;
    for (int i = l; i <= r; i++)
        if (fabs(P[mid].x - P[i].x) < d) tmp.push_back(P[i]);
    sort(tmp.begin(), tmp.end(), cmpy);
    for (int i = 0; i < tmp.size(); i++)
        for (int j = i + 1; j < tmp.size() && tmp[j].y - tmp[i].y < d; j++)
            d = min(d, DisPP(tmp[i], tmp[j]));
    return d;
}
```

5.10 K-D Tree

```
#include <iostream>
#include <algorithm>
#include <stack>
#include <math.h>
using namespace std;
/*function of this program: build a 2d tree using the input trainingdata
 the input is exm_set which contains a list of tuples (x,y)
 the output is a 2d tree pointer*/


struct data
```

```
12 {
13     double x = 0;
14     double y = 0;
15 };
16
17 struct Tnode
18 {
19     struct data dom_elt;
20     int split;
21     struct Tnode * left;
22     struct Tnode * right;
23 };
24
25 bool cmp1(data a, data b){
26     return a.x < b.x;
27 }
28
29 bool cmp2(data a, data b){
30     return a.y < b.y;
31 }
32
33 bool equal(data a, data b){
34     if (a.x == b.x && a.y == b.y)
35     {
36         return true;
37     }
38     else{
39         return false;
40     }
41 }
42
43 void ChooseSplit(data exm_set[], int size, int &split, data &SplitChoice{
44     /*compute the variance on every dimension. Set split as thedismension that have the
       biggest
45      variance. Then choose the instance which is the median on thissplit dimension.*/
46     /*compute variance on the x,y dimension. DX=EX^2-(EX)^2*/
47     double tmp1,tmp2;
48     tmp1 = tmp2 = 0;
49     for (int i = 0; i < size; ++i)
50     {
51         tmp1 += 1.0 / (double)size * exm_set[i].x * exm_set[i].x;
52         tmp2 += 1.0 / (double)size * exm_set[i].x;
53     }
54     double v1 = tmp1 - tmp2 * tmp2;  //compute variance on the xdimension
55
56     tmp1 = tmp2 = 0;
57     for (int i = 0; i < size; ++i)
58     {
59         tmp1 += 1.0 / (double)size * exm_set[i].y * exm_set[i].y;
60         tmp2 += 1.0 / (double)size * exm_set[i].y;
61     }
62     double v2 = tmp1 - tmp2 * tmp2;  //compute variance on the ydimension
63
64     split = v1 > v2 ? 0:1; //set the split dimension
65
66     if (split == 0)
67     {
68         sort(exm_set,exm_set + size, cmp1);
69     }
70     else{
71         sort(exm_set,exm_set + size, cmp2);
```

```
72      }
73
74      //set the split point value
75      SplitChoice.x = exm_set[size / 2].x;
76      SplitChoice.y = exm_set[size / 2].y;
77
78  }
79
80  Tnode* build_kdtree(data exm_set[], int size, Tnode* T){
81      //call function ChooseSplit to choose the split dimension and splitpoint
82      if (size == 0){
83          return NULL;
84      }
85      else{
86          int split;
87          data dom_elt;
88          ChooseSplit(exm_set, size, split, dom_elt);
89          data exm_set_right [100];
90          data exm_set_left [100];
91          int sizeleft ,sizeright;
92          sizeleft = sizeright = 0;
93
94          if (split == 0)
95          {
96              for (int i = 0; i < size; ++i)
97              {
98
99                  if (!equal(exm_set[i],dom_elt) && exm_set[i].x <=dom_elt.x)
100                 {
101                     exm_set_left[sizeleft].x = exm_set[i].x;
102                     exm_set_left[sizeleft].y = exm_set[i].y;
103                     sizeleft++;
104                 }
105                 else if (!equal(exm_set[i],dom_elt) && exm_set[i].x >dom_elt.x)
106                 {
107                     exm_set_right[sizeright].x = exm_set[i].x;
108                     exm_set_right[sizeright].y = exm_set[i].y;
109                     sizeright++;
110                 }
111             }
112         }
113         else{
114             for (int i = 0; i < size; ++i)
115             {
116
117                 if (!equal(exm_set[i],dom_elt) && exm_set[i].y <=dom_elt.y)
118                 {
119                     exm_set_left[sizeleft].x = exm_set[i].x;
120                     exm_set_left[sizeleft].y = exm_set[i].y;
121                     sizeleft++;
122                 }
123                 else if (!equal(exm_set[i],dom_elt) && exm_set[i].y >dom_elt.y)
124                 {
125                     exm_set_right[sizeright].x = exm_set[i].x;
126                     exm_set_right[sizeright].y = exm_set[i].y;
127                     sizeright++;
128                 }
129             }
130         }
131         T = new Tnode;
132         T->dom_elt.x = dom_elt.x;
```

```
        T->dom_elt.y = dom_elt.y;
        T->split = split;
        T->left = build_kdtree(exm_set_left, sizeleft, T->left);
        T->right = build_kdtree(exm_set_right, sizeright, T->right);
        return T;

    }
}


double Distance(data a, data b){
    double tmp = (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y);
    return sqrt(tmp);
}


void searchNearest(Tnode * Kd, data target, data &nearestpoint, double &distance){

    //1. 如果Kd是空的，则设dist为无穷大返回

    //2. 向下搜索直到叶子结点

    stack<Tnode*> search_path;
    Tnode* pSearch = Kd;
    data nearest;
    double dist;

    while(pSearch != NULL)
    {
        //pSearch加入到search_path中；
        search_path.push(pSearch);

        if (pSearch->split == 0)
        {
            if(target.x <= pSearch->dom_elt.x) /* 如果小于就进入左子树 */
            {
                pSearch = pSearch->left;
            }
            else
            {
                pSearch = pSearch->right;
            }
        }
        else{
            if(target.y <= pSearch->dom_elt.y) /* 如果小于就进入左子树 */
            {
                pSearch = pSearch->left;
            }
            else
            {
                pSearch = pSearch->right;
            }
        }
    }
    //取出search_path最后一个赋给nearest
    nearest.x = search_path.top()->dom_elt.x;
    nearest.y = search_path.top()->dom_elt.y;
    search_path.pop();


    dist = Distance(nearest, target);
```

```
194     //3. 回溯搜索路径
195
196     Tnode* pBack;
197
198     while(search_path.size() != 0)
199     {
200         //取出search_path最后一个结点赋给pBack
201         pBack = search_path.top();
202         search_path.pop();
203
204         if(pBack->left == NULL && pBack->right == NULL) /* 如果pBack为子结点 */
205
206         {
207
208             if( Distance(nearest, target) > Distance(pBack->dom_elt,target) )
209             {
210                 nearest = pBack->dom_elt;
211                 dist = Distance(pBack->dom_elt, target);
212             }
213
214         }
215
216         else
217
218         {
219
220             int s = pBack->split;
221             if (s == 0)
222             {
223                 if( fabs(pBack->dom_elt.x - target.x) < dist) /* 如果target为中心的圆
    （球或超球），半径为dist的圆与分割超平面交， 那么就要跳到另一边的子空间去搜索 */
224                 {
225                     if( Distance(nearest, target) > Distanc(pBack->dom_elt, target) )
226                     {
227                         nearest = pBack->dom_elt;
228                         dist = Distance(pBack->dom_elt, target);
229                     }
230                     if(target.x <= pBack->dom_elt.x) /* 如果target位pBack的左子空间，那
    么就要跳到右子空间去搜索 */
231                         pSearch = pBack->right;
232                     else
233                         pSearch = pBack->left; /* 如果target位于pBack的子空间，那么就要
    跳到左子空间去搜索 */
234                     if(pSearch != NULL)
235                         //pSearch加入到search_path中
236                         search_path.push(pSearch);
237                 }
238             }
239             else {
240                 if( fabs(pBack->dom_elt.y - target.y) < dist) /* 如果target为中心的圆
    （球或超球），半径为dist的圆与分割超平面交， 那么就要跳到另一边的子空间去搜索 */
241                 {
242                     if( Distance(nearest, target) > Distanc(pBack->dom_elt, target) )
243                     {
244                         nearest = pBack->dom_elt;
245                         dist = Distance(pBack->dom_elt, target);
246                     }
247                     if(target.y <= pBack->dom_elt.y) /* 如果target位pBack的左子空间，那
    么就要跳到右子空间去搜索 */
248                         pSearch = pBack->right;
249                     else
```

```
250                         pSearch = pBack->left; /* 如果target位于pBack的子空间，那么就要
    跳到左子空间去搜索 */
251                     if(pSearch != NULL)
252                         // pSearch加入到search_path中
253                         search_path.push(pSearch);
254                 }
255             }
256 
257         }
258     }
259 
260     nearestpoint.x = nearest.x;
261     nearestpoint.y = nearest.y;
262     distance = dist;
263 
264 }
265 
266 int main(){
267     data exm_set[100]; //assume the max training set size is 100
268     double x,y;
269     int id = 0;
270     cout<<"Please input the training data in the form x y. One instanceper line. Enter
    -1 -1 to stop."<<endl;
271     while (cin>>x>>y){
272         if (x == -1)
273         {
274             break;
275         }
276         else{
277             exm_set[id].x = x;
278             exm_set[id].y = y;
279             id++;
280         }
281     }
282     struct Tnode * root = NULL;
283     root = build_kdtree(exm_set, id, root);
284 
285     data nearestpoint;
286     double distance;
287     data target;
288     cout <<"Enter search point"<<endl;
289     while (cin>>target.x>>target.y)
290     {
291         searchNearest(root, target, nearestpoint, distance);
292         cout<<"The nearest distance is "<<distance<<",and the nearestpoint is "<<
    nearestpoint.x<<","<<nearestpoint.y<<endl;
293         cout <<"Enter search point"<<endl;
294 
295     }
296 }
```

6 Conclusion

6.1 Game Theory

6.1.1 Bash's Game

Bash's Game 巴什博弈
有一堆个数为 n 的石子，游戏双方依次从中拿取，满足:
1. 每次至少取 1 个，最多取 m 个.
最后取光者得胜。
结论: n = t(m+1) + r, 必败态:r = 0;
巴什博弈变种:
取一个指定集合的石头个数
取到最后一个石子输, n = t(m + 1) + r, r = 1;

6.1.2 Wythoff' s Game

Wythoff' s Game（威佐夫博弈）
有两堆分别为 (an, bn) 的石子, 游戏双方依次从中拿取，满足:
1. 从任意一堆中取任意个 > 1。2. 从两堆中取同样多个。最后取完者胜.
结论: 对于任意的局势 (a, b)(a $<$ b), 必败点为 (b-a)*(sqrt(5)+1)/2=a.

6.1.3 Fibonacci' s Game / Zeckendorf's theory

Fibonacci' s Game（斐波那契博弈）
有一堆个数为 n 的石子，游戏双方轮流取石子，满足:
1. 先手不能在第一次把所有的石子取完;
2. 之后每次可以取的石子数介于 1 到对手刚取的石子数的 2 倍之间（包含 1 和对手刚取的石子数的 2 倍）。
结论: 必败点是斐波那契数

齐肯多夫定理: 任何正整数可以表示为若干个不连续的 Fibonacci 数之和

6.1.4 Nim' s Game / Anti-Nim's Game / K-Nim's Game / Anti-K-Nim's Game

Nim' s Game（尼姆博弈）
石子的个数可以等价成某个游戏的 SG 函数。

有 n 堆石子，游戏双方依次从中拿取, 满足:
1. 规定每次只能从一堆中取若干根，可将一堆全取走，但不可不取.
最后取完者为胜。
结论:
T 态: 所有火柴数异或和为 0
S 态: 所有火柴数异或和不为 0

必胜态:S

有 n 堆石子, 游戏双方依次从中拿取, 满足:
1. 规定每次只能从一堆中取若干根，可将一堆全取走，但不可不取.
最后取完者为败。
结论:
S0 态: 即仅有奇数个孤单堆
T0 态: 即仅有偶数个孤单堆
S1 态: 异或和大于 0, 且有 1 个充裕堆
T1 态: 不存在
S2 态: 异或和大于 0, 且有多个充裕堆
T2 态: 异或和等于 0, 且有多个充裕堆
必胜态:T0,S1,S2
必败态:S0,T2

有 n 堆石子，游戏双方依次从中拿取, 满足:
1. 规定每次只能至多 k 堆中取若干根，可将 k 堆全取走，但不可不取.
最后取完者为胜。
结论:
对于每一堆，把它石子的个数用二进制表示
必败态: 对所有的石子堆，如果在任何一个二进制位上 1 的个数总是 k+1 的整数倍

有 n 堆石子，游戏双方依次从中拿取, 满足:
1. 规定每次只能至多 k 堆中取若干根，可将 k 堆全取走，但不可不取
最后取完者为败。
结论:
1. 对于每一堆，把它石子的个数用二进制表示
2. 所有的堆（非零堆，下同）全是 1, 此时如果 1 堆个数模 k+1 的结果是 1 则必败，否则必胜 (我们可以通过拿走 0 到 k 个堆来随意调整当前状态模的结果，然后再将所有大于 1 的堆降到 1 就行了)
3. 有多于 k 个堆的个数大于 1。必胜

6.1.5 阶梯博弈

有 n 个阶梯呈升序排列，每个阶梯上有若干个石子，游戏双方轮流取石子, 满足:
1. 将一个阶梯上的石子移任意个（>0）到前一个台阶。
当没有可行操作时（所有石子都被移动到了地面，即第 0 号台阶）输。
结论:
奇数号台阶的 Nim 游戏
变种 1: 树上, 每个石子只能往父亲节点移动.
变种 2:
游戏双方在一个 1*N 的格子内挪动棋子，刚开始在若干个位置上有棋子, 每个位置至多一个棋子
每一个选手可以进行的操作时选择一个棋子并把它向左方移动，当然不能越过其它的棋子，也不能超出边界。
谁不能移动谁就输了。求谁会赢?

结论:
将棋子位置按升序排列, 然后从后往前两两绑成一对, 如果个数是奇数, 那么将第一个和边界外绑定.
一对棋子的前一个和前一对棋子的后一个之间有多少个空位置对最终的结果是没有影响的。
于是我们只需要考虑同一对的两个棋子之间有多少空位, 将同一对棋子间的空位视为石子, 做 nim 游戏
两对棋子间的空格数当奇数位石子, 其他当偶数位石子, 石子相右边移动
变种 3:
山上有 n 个人，每个人给出距离山顶的距离，给出其中一个人为 king，每次能挑选一个人向上移动，不能越过其他人，最后将 king 移动到山顶者获胜。问获胜者。
结论:
只要把 King 当作普通人一样处理即可。除了两种特殊情况：
1. 当 King 是第一个人时，Alice 直接胜
2. 当 King 是第二个人且一共有奇数个人时，第一堆的大小需要减 1。

6.1.6 Multi-Nim

有 n 堆石子，游戏双方依次从中拿取, 满足:
1. 任意一堆石子中拿任意多个石子 (不能不拿)
2. 把一堆数量不少于 2 石子分为两堆不为空的石子
最后取完者为胜。
结论:
操作一与普通的 Nim 游戏等价
操作二实际上是将一个游戏分解为两个游戏，根据 SG 定理，我们可以通过异或运算把两个游戏连接到一起，作为一个后继状态

$$SG(x) \equiv \begin{cases} x-1 & (xmod4=0) \\ x & (xmod4=1or2) \\ x+1 & (xmod4=3) \end{cases} \tag{1}$$

Multi-SG 游戏规定，在符合拓扑原则的前提下，一个单一游戏的后继可以为多个单一游戏。
Multi-SG 其他规则与 SG 游戏相同。
注意在这里要分清楚后继与多个单一游戏
对于一个状态来说，不同的划分方法会产生多个不同的后继，而在一个后继中可能含有多个独立的游戏
一个后继状态的 SG 值即为后继状态中独立游戏的异或和
该状态的 SG 值即为后继状态的 SG 值中未出现过的最小值

6.1.7 Every-SG

给定一张无向图，上面有一些棋子，两个顶尖聪明的人在做游戏，每人每次必须将可以移动的棋子进行移动，不能移动的人
因为两个人都顶尖聪明，因此当一个人知道某一个游戏一定会输的话，它一定会尽力缩短游戏的时间，当它知道某一个游戏一定会赢的话，一定会尽力延长游戏的时间
对于还没有结束的单一游戏，游戏者必须对该游戏进行一步决策；
其他规则与普通 SG 游戏相同

Every-SG 游戏与普通 SG 游戏最大的不同就是它多了一维时间
对于 SG 值为 0 的点，我们需要知道最少需要多少步才能走到结束
对于 SG 值不为 0 的点，我们需要知道最多需要多少步结束
这样我们用 step 变量来记录这个步数

$$step(x) \equiv \begin{cases} 0 & u \\ maxstep(v) & sg(u)! = 0vusg(v) = 0 \\ minstep(v) & sg(u) = 0vu \end{cases} \tag{2}$$

6.1.8 树的删边游戏

给出一个有 N 个点的树，有一个点作为树的根节点。游戏者轮流从树中删去边，删去一条边后，不与根节点相连的部分将被移走。谁无法移动谁输。
结论:
Colon Principle: 对于树上的某一个点，ta 的分支可以转化成以这个点为根的一根竹子，这个竹子的长度就是 ta 各个分支的边的数量的异或和
叶子节点的 SG 值为 0；中间节点的 SG 值为它的所有子节点的 SG 值加 1 后的异或和。

6.1.9 Chomp's theory?

取一个无关紧要的位置, 如果对方必胜, 则学习其策略, 我方必胜.

6.1.10 Other's theory?

有 n 堆石子，游戏双方依次从中拿取, 满足:
1. 规定每次能从任意多堆中取 1 根, 不可不取.
最后取完者为胜。
结论: 如果全是偶数, 先手必败, 否者先手必胜

一个无相联通图，有一个点作为图的根。
游戏者轮流从图中删去边，删去一条边后，不与根节点相连的部分将被移走。
谁无路可走谁输。
结论:
Fusion Principle: 环上的点可以融合，且不改变图的 SG 值, 我们可以把一个带有奇数边的环等价成只有一个端点的一条边而偶数边的环等价于一个点

6.1.11 SG Theory

```
memset(mex, 0, sizeof mex);
for (int i = 1; i < maxN;++i) {
    for (int j = 1; j <= n;++j) {
        if (a[j] <= i)
            mex[SG[i - a[j]]] = i;
            for (int k = 1; i - k - a[j] > 0;++k)
                mex[SG[k] ^ SG[i - k - a[j]]] = 1;
    }
```

```
    for (int j = 0;; ++j)
        if (mex[j]!=i){
            SG[i] = j;
            break;
        }
}
```

6.1.12 SJ Theory

反公平游戏 Anti-SG Game
DAG 上没有出度的点为胜利状态，其它定义与一般游戏相同。
现在的问题是解决多个反公平游戏的合并。
SJ 定理说明：先手必胜，当且仅当以下两个条件同时成立或同时不成立：
1. 合并的 SG 值为 0；
2. 所有游戏的 SG 值不超过 1。

6.1.13 Surreal Number Theory

6.2 Math

6.2.1 Euler's Theorem

$$a^b \equiv \begin{cases} a^{b\%\varphi(p)} & \gcd(a,p)=1 \\ a^b & \gcd(a,p)\neq 1, b<\varphi(p) \\ a^{b\%\varphi(p)+\varphi(p)} & \gcd(a,p)\neq 1, b\geq\varphi(p) \end{cases} (mod\ p)$$

6.2.2 Möbius Inversion

Dirichlet Convolution is $(f\times g)(N)=\sum_{d|N} f(d)*g(\frac{N}{d})$
Theorem:

$$\begin{cases} f=g\times 1 \\ g=f\times\mu \end{cases}$$

$$\begin{cases} id(n)=\sum\limits_{d|n}\varphi(d) \\ e(n)=\sum\limits_{d|n}\mu(d) \end{cases} \tag{3}$$

$$\begin{cases} \sum\limits_{i}^{n}\sum\limits_{j}^{m} gcd(i,j)=\sum\limits_{d}^{\max(n,m)}\varphi(d)*\lfloor\frac{n}{d}\rfloor\lfloor\frac{m}{d}\rfloor \\ \sum\limits_{i}^{n}\sum\limits_{j}^{m} e(gcd(i,j))=\sum\limits_{d}^{\min(n,m)}\mu(d)*\lfloor\frac{n}{d}\rfloor\lfloor\frac{m}{d}\rfloor \\ \sum\limits_{i=1}^{n}|\mu(i)|=\sum\limits_{i=1}^{\lfloor\sqrt{n}\rfloor}\mu(i)*\lfloor\frac{n}{i*i}\rfloor \end{cases} \tag{4}$$

$$\begin{cases} sum(x,y) = \sum_{i}^{x}\sum_{j}^{y} i*j = \frac{x*(x+1)}{2} * \frac{y*(y+1)}{2} \\ F(x,y) = \sum_{i=1}^{\min(x,y)} i^2 * \mu(i) * sum(\lfloor\frac{x}{i}\rfloor, \lfloor\frac{y}{i}\rfloor) \\ \sum_{i}^{n}\sum_{j}^{m} lcm(i,j) = \sum_{i=1}^{\min(n,m)} d * F(\lfloor\frac{n}{i}\rfloor, \lfloor\frac{y}{i}\rfloor) \end{cases} \tag{5}$$

6.2.3 Sieve Tips

$$\varphi(nm) = \varphi(n)\cdot\varphi(m)\cdot\frac{gcd(n,m)}{\varphi(gcd(n,m))} \tag{6}$$

$$\varphi(n) = \sum_{i=1}^{n}[(n,i)=1]\cdot i = \frac{n*\varphi(n)+[n=1]}{2} \tag{7}$$

$$\begin{cases} id = \varphi \times 1 \\ \frac{n\cdot(n+1)}{2} = \sum_{i=1}^{n} i = \sum_{i=1}^{n}\sum_{d|i}\varphi(d) = \sum_{\frac{i}{d}=1}^{n}\sum_{d=1}^{\lfloor\frac{n}{\frac{i}{d}}\rfloor}\varphi(d) = \sum_{i=1}^{n}\phi(\lfloor\frac{n}{i}\rfloor) \end{cases} \tag{8}$$

$$\begin{cases} e = \mu \times 1 \\ 1 = \sum_{i=1}^{n}[i=1] = \sum_{i=1}^{n}\sum_{d|i}\mu(d) = \sum_{i=1}^{n}\sum_{d=1}^{\lfloor\frac{n}{i}\rfloor}\mu(d) = \sum_{i=1}^{n} M(\lfloor\frac{n}{i}\rfloor) \end{cases} \tag{9}$$

$$\begin{cases} id^2 = (id\cdot\varphi)\times id \\ \phi'(n) = \sum_{i=1}^{n} i\cdot\varphi(i) \\ \frac{n\cdot(n+1)\cdot(2n+1)}{6} = \sum_{i=1}^{n} i^2 = \sum_{i=1}^{n}\sum_{d|i} d\cdot\varphi(d)\cdot\frac{i}{d} = \sum_{\frac{i}{d}=1}^{n}\frac{i}{d}\sum_{d=1}^{\lfloor\frac{n}{\frac{i}{d}}\rfloor} d\cdot\varphi(d) = \sum_{i=1}^{n} i\cdot\phi'(\lfloor\frac{n}{i}\rfloor) \end{cases} \tag{10}$$

6.2.4 Newton's method

Iff

$$f^{'}(x) \neq 0 \ \&\& \ f^{''}(x) continuous$$

Then

$$x_{n+1} = x_n - \frac{f(x_n)}{f^{'}(x_n)}$$

6.2.5 Cantor Expansion

康托展开是一个全排列到一个自然数的双射，常用于构建哈希表时的空间压缩。康托展开的实质是计算当前排列在所有由小到大全排列中的顺序，因此是可逆的。以下称第 x 个全排列是都是指由小到大的顺序。

$$X = a_n(n-1)! + a_{n-1}(n-2)! + \cdots + a_1\cdot 0!$$

其中 a_i 为排列的第 i 个元素值
例如，3 5 7 4 1 2 9 6 8 展开为 98884。因为 X=2*8!+3*7!+4*6!+2*5!+0*4!+0*3!+2*2!+0*1!+0*0!=98884.
解释：
排列的第一位是 3，比 3 小的数有两个，以这样的数开始的排列有 8! 个，因此第一项为 2*8!
排列的第二位是 5，比 5 小的数有 1、2、3、4，由于 3 已经出现，因此共有 3 个比 5 小的数，这样的排列有 7! 个，因此第二项为 3*7!
以此类推，直至 0*0!
用途：
显然，n 位（0 n-1）全排列后，其康托展开唯一且最大约为 n!，因此可以由更小的空间来储存这些排列。由公式可将 X 逆推出唯一的一个排列。
逆运算：
如 n=5,x=96 时：
首先用 96-1 得到 95，说明 x 之前有 95 个排列.(将此数本身减去 1)
用 95 去除 4! 得到 3 余 23，说明有 3 个数比第 1 位小，所以第一位是 4.
用 23 去除 3! 得到 3 余 5，说明有 3 个数比第 2 位小，所以是 4，但是 4 已出现过，因此是 5.
用 5 去除 2! 得到 2 余 1，类似地，这一位是 3.
用 1 去除 1! 得到 1 余 0，这一位是 2.
最后一位只能是 1.
所以这个数是 45321.
按以上方法可以得出通用的算法。

6.2.6 $\sum_{i=1}^{n} i^k$

$$\sum_{i}^{n} i = \frac{n(n+1)}{2}$$
$$\sum_{i}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$
$$\sum_{i}^{n} i^3 = [\frac{n(n+1)}{2}]^2$$

差分和组合数:$O(k^2)$

考虑数列 $\{1_k, 2_k, \cdots, i_k\}$
相邻两项做差之后，得到的数列的每项应该是一个 k - 1 次关于 i 的多项式。
再次相邻两项做差之后，得到的数列的每项应该是一个 k - 2 次关于 i 的多项式。
如此进行 k 次，得到的数列的每项应该是一个 0 次关于 i 的多项式，即常数数列。
再次相邻两项做差之后，一定会得到一个全是 0 的数列。
假设经过 i 次差分数列之后的数列第一项为 r_i 那么答案就是

$$\sum_{i=0}^{k} r_i \binom{n}{i+1}$$

举例来说，考虑数列
1,8,27,64,125,216,...

求差分可得
7,19,37,61,91,...
12,18,24,30,...
6,6,6,...
0,0,...
每个数列的第一项为 1, 7, 12, 6。所以最终的答案即为

$$\binom{n}{1}+7\binom{n}{2}+12\binom{n}{3}+6\binom{n}{4}=\frac{n^2(n+1)^2}{4}$$

拉格朗日插值:$O(k)$

6.2.7 Generating Function

生成函数

普通型

$$A(x)=\sum_{i=0}^{\infty}a_ix^i$$

已知 $\{a_i\}$, $\{b_i\}$ 的生成函数分别是 A(x), B(x)。
那么 $\{a_i \pm b_i\}$ 的生成函数是 A(x) ± B(x)。
值得注意的是数列 $\{a_i\}$, $\{b_i\}$ 的卷积的生成函数，恰好是 C(x) = A(x)B(x)。

$$c_i=\sum_j a_jb_{i-j}$$

关键公式

$$\frac{1}{1-x}=1+x+x^2+x^3+\cdots=\sum_i x^i$$
$$\frac{1}{1-x^2}=1+x^2+x^4+x^6+\cdots=\sum_i x^{2i}$$

推广的二项式定理

$$(1+x)^n=\binom{n}{0}x^0+\binom{n}{1}x^1+\binom{n}{2}x^2+\cdots=\sum_i\binom{n}{i}x^i$$

其中有 β 函数和 Γ 函数

$$\Gamma(s)=\int_0^{+\infty}x^{s-1}e^{-x}dx$$
$$B(p,q)=\binom{p+q}{p}=\int_0^1 x^{p-1}(1-x)^{q-1}dx=\frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)}$$

分别有递推公式

$$\begin{aligned}\Gamma(s+1) &= s\Gamma(s)\\ B(p,q) &= B(q,p)\\ &= \frac{q-1}{p+q-1}B(p,q-1)\\ &= \frac{p-1}{p+q-1}B(p-1,q)\\ &= \frac{(p-1)(q-1)}{(p+q-1)(p+q-2)}B(p-1,q-1)\end{aligned}$$

例子

Fibonacci 生成函数

构造一个与 f_n 有关得幂级数

$$F(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_n x^n + \cdots$$

亦即

$$F(x) = xF(x) + x^2F(x) + f_0 + (f_1 - f_0)x$$

带入 $f_0 = 0, f_1 = 1$

$$F(x) = \frac{x}{1-x-x^2}$$

如果想继续计算通项，我们需要解方程得到

$$\frac{x}{1-x-x^2} = \frac{a}{1-\alpha x} + \frac{b}{1-\beta x}$$

解出

$$\begin{cases}\alpha = \dfrac{1+\sqrt{5}}{2}\\ \beta = \dfrac{1-\sqrt{5}}{2}\end{cases}$$

带入方程有

$$a = \frac{1}{\sqrt{5}}, b = -\frac{1}{\sqrt{5}}$$

则

$$\begin{aligned}F(x) &= \frac{a}{1-\alpha x} + \frac{b}{1-\beta x}\\ &= a(1+\alpha x+\alpha^2x^2+\cdots) + b(1+\beta x+\beta^2x^2+\cdots)\end{aligned}$$

蕴含通项为

$$f_n = a\alpha^n + b\beta^n = \frac{(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n}{\sqrt{5}}$$

Catalan 数

$$c_n = \sum_{i=1}^{n-1} c_i c_{n-i}$$

生成函数

$$C(x) = c_1 x + x_2 x^2 + \cdots$$

亦即

$$C(x) = xC(x)^2 + 1$$

解得

$$C(x) = \frac{1 - \sqrt{1-4x}}{2x}$$

利用推广的二项式定理展开 $\sqrt{1-4x}$ 可得通项

$$c_n = \frac{1}{n}\binom{2n-2}{n-1}$$

指数型生成函数

$$A(x) = \sum_{i=0}^{\infty} \frac{a_i x^i}{i!}$$

已知 $\{a_i\}$, $\{b_i\}$ 的生成函数分别是 A(x), B(x)。
那么 $\{a_i \pm b_i\}$ 的生成函数是 A(x) ± B(x)。
值得注意的是数列 $\{a_i\}$, $\{b_i\}$ 的卷积的生成函数，恰好是 C(x) = A(x)B(x)。

$$c_i = \sum_j \binom{i}{j} a_j b_{i-j}$$

关键公式

$$e^x = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \cdots = \sum_i \frac{x^i}{i!}$$

$$e^{-x} = 1 - x + \frac{1}{2}x^2 - \frac{1}{6}x^3 + \cdots = \sum_i \frac{(-x)^i}{i!}$$

$$\frac{e^x + x^{-x}}{2} = 1 + \frac{1}{2}x^2 + \frac{1}{24}x^4 + \cdots = \sum_i \frac{x^{2i}}{(2i)!}$$

6.2.8 Polya

设 G 是 p 个对象的一个置换群，用 k 种颜色去染这 p 个对象，若一种染色方案在群 G 的作用下变为另一种方案，则这两个方案当作是同一种方案，这样的不同染色方案数为:

$$\frac{1}{|G|} \times \sum (k^{C(f)}), f \in G$$

C(f) 为循环节，|G| 表示群的置换方法数
对于有 n 个位置的手镯，有 n 种旋转置换和 n 种翻转置换
对于旋转置换:

$$C(f_i) = gcd(n, i)$$

i 表示一次转过 i 颗宝石，i = 0 时 c = n;
对于翻转置换:
如果 n 为偶数: 则有 $\frac{n}{2}$ 个置换 $C(f) = \frac{n}{2}$，有 $\frac{n}{2}$ 个置换 $C(f) = \frac{n}{2} + 1$
如果 n 为奇数: $C(f) = \frac{n}{2} + 1$

6.2.9 FWT

$$\begin{cases} C_k = \sum_{i \oplus j = k} A_i * B_j \\ DWT(A)_i = \sum_{j}^{n} A_j * f_{i,j} \\ DWT(C)_i = DWT(A)_i * DWT(B)_i \\ f_{i,j} \cdot f_{i,k} = f_{i,j \oplus k} \\ f_{i,j} = [i \ and \ j == i] \quad (and) \\ f_{i,j} = [i \ and \ j == j] \quad (or) \\ f_{i,j} = (-1)^{|i \ and \ j|} \quad (xor) \end{cases}$$

6.3 Graph Theory

6.3.1 Matrix-Tree Theorem

基尔霍夫矩阵树定理
构造一个矩阵 A。
如果 i = j，那么 Aij 为点 i(j) 的度数。
如果 i=j，那么 Aij 为 i 到 j 的边数的相反数。最终得到的矩阵，删掉任意一行，任意一列之后的矩阵行列式求值，可得到原图中生成树的个数。

6.4 Geometry

6.4.1 The Number of Ingeter Point on a Circle

Set $r = const$ is the radius of the circle.

$$r^2 = {p_1}^{a_1} + {p_2}^{a_2} + \cdots + {p_m}^{a_m} = \sum_{i=1}^{m} {p_i}^{a_i}$$

Define

$$\chi(n) = \begin{cases} 1 & n\%4 = 1 \\ -1 & n\%4 = 3 \\ 0 & n\%2 = 0 \end{cases}$$

By the way, $\chi(n)$ is a multiplicative function.

Define

$$\Gamma(p_i,a_i)=\sum_{j=0}^{a_i}\chi(p_i^j)=\begin{cases}1 & p_i=2 \quad || \quad (p_i\%4=3 \quad \&\& \quad a_i\%2=0)\\ 0 & p_i\%4=3 \quad \&\& \quad a_i\%2=1\\ a_i+1 & p_i\%4=1\end{cases}$$

Define cnt is the number of integer point on circle

$$cnt(r)=4\prod_{i=1}^{m}\sum_{j=0}^{a_i}\chi(p_i^j)=4\prod_{i=1}^{m}\Gamma(p_i,a_i)=4\sum_{k|r^2}\chi(k)$$

Define CNT is the number of integer point in circle

$$CNT(r)=1+\sum_{i=1}^{r^2}cnt(i)=1+\sum_{i=1}^{r^2}\lfloor\frac{r^2}{i}\rfloor\chi(i)$$

6.5 Josephus

6.5.1 $J(n,m)$: The Last Surviving Person

based-0, $m-1$ was the first be killed.

$$J_0(n,m)=\begin{cases}0 & n=1;\\ (J_0(n-1,m)+m)\%n & 1<n<m;\\ \lfloor\frac{k((f(n',k)-nmodk)modn')}{k-1}\rfloor \ where\ n'=n-\lfloor\frac{n}{k}\rfloor & m<=n\end{cases}$$

$$J_k(n,m)=(J_0(n,m)+k)\%n$$

```
int J0(int n, int m) {
    if (m == 1) return n - 1;
    int ans = 0;
    for (int i = 2; i <= n; ) {
        if (ans + m >= i) {
            ans = (ans + m) % i;
            i++;
            continue;
        }
        int step = (i - 1 - ans - 1) / (m - 1);
        if (i + step > n) {
            ans += (n - (i - 1)) * m;
            break;
        }
        i += step; ans += step * m;
    }
    return ans;
}
```

6.5.2 $aJ(n,1,K)$: Survival Time of K-th Person

based-1, 1 was the first be killed.

```
int aJ1(int size, int be, int goal){
    if (be > size) be = (be - 1) % size + 1;
    if (goal % m == be % m) return (goal - be) / m + 1;
```

```
    return (size - be) / m + 1 + get(size - (size - be) / m - 1, (be + ((size - be) / m
    + 1) * m) - size, goal > be ? goal - (goal - be) / m - 1 : goal);
}
\\ size : the size of current group;
\\ be : the start person of current step;
\\ goal : index of the asked person.
```

7 Others

7.1 Offline Algorithm

7.1.1 CDQ Divide and Conquer

```
struct Node {
    int x, y, z, ans;
    Node() {}
    Node(int _x, int _y, int _z):x(_x), y(_y), z(_z) {}
    bool operator < (const Node &b) const {
        if(y == b.y) {
            if(z == b.z) return x < b.x;
            return z < b.z;
        }
        return y < b.y;
    }
}A[MAXN], B[MAXN], C[MAXN];
int bit[MAXN];
void add(int k, int v) {
    for(; k <= m; k += k & -k) bit[k] = max(bit[k], v);
}
void clear(int k) {
    for(; k <= m; k += k & -k) bit[k] = 0;
}
int sum(int k) {
    int res = 0;
    for(; k; k -= k & -k) res = max(res, bit[k]);
    return res;
}
void solve(int l, int r) {
    if(l == r) {
        B[l] = A[l];
        return;
    }
    int mid = (l + r) >> 1;
    solve(l, mid);
    for(int i = mid + 1; i <= r; i++) B[i] = A[i];
    //sort(B + l, B + mid + 1);
    sort(B + mid + 1, B + r + 1);
    int L = l;
    for(int R = mid + 1; R <= r; R++) {
        while(L <= mid && B[L].y < B[R].y) add(B[L].z, B[L].ans), L++;
        A[B[R].x].ans = max(A[B[R].x].ans, sum(B[R].z - 1) + 1);
        B[R].ans = A[B[R].x].ans;
    }
    for(int i = l; i <= L; i++) clear(B[i].z);
    solve(mid + 1, r);
    L = l;
    int p = l, q = mid + 1;
    while(p <= mid || q <= r) {
        if(q > r || (p <= mid && B[p].y <= B[q].y)) C[L++] = B[p++];
        else C[L++] = B[q++];
    }
    for(int i = l; i <= r; i++) B[i] = C[i];
}
```

7.1.2 Mo' s Algorithm

```
struct Node{
    int l, r, t, id;
    bool operator < (const Node& a) const {
        if(l /sz == a.l / sz) {
            if(r == a.r) return t < a.t;
            return r < a.r;
        }
        return l / sz < a.l / sz;
    }
}q[MAXN];
void solve() {
    while (t < q[i].t) addTime(t++, 1);
    while (t > q[i].t) addTime(--t, -1);
    while(L < q[i].l) add(L++, -1);
    while(L > q[i].l) add(--L, 1);
    while(R < q[i].r) add(++R, 1);
    while(R > q[i].r) add(R--, -1);
}
```

7.1.3 Mo's Algorithm On Tree

```
struct Edge {
    int to, nxt;
}e[MAXN << 1];
int head[MAXN], ecnt;
int stack[MAXN], top, belong[MAXN], cnt, sz;
struct Node {
    int l, r, id, ti;
    bool operator < (const Node &x) const {
        return belong[l] < belong[x.l] || (belong[l] == belong[x.l] && belong[r] <
    belong[x.r]) || (belong[l] == belong[x.l] && belong[r] == belong[x.r] && ti < x.ti);
    }
}q[MAXN];
struct Node2 {
    int l, r, ti;
}qq[MAXN];
int n, m, Q, Q0, Q1;
int V[MAXN], W[MAXN], C[MAXN];
int fa[MAXN][S + 3], dep[MAXN];
long long ans[MAXN], tans;
int vis[MAXN], cur[MAXN];
long long sum[MAXN];
int l, r, tm;
inline int read() {
    int x = 0; char ch = getchar(); bool fg = 0;
    while(ch < '0' || ch > '9') { if(ch == '-') fg = 1; ch = getchar(); }
    while(ch >= '0' && ch <= '9') { x = x * 10 + ch - '0'; ch = getchar(); }
    return fg ? -x : x;
}
inline void add_edge(int u, int v) {
    e[++ecnt] = (Edge) {v, head[u]}; head[u] = ecnt;
    e[++ecnt] = (Edge) {u, head[v]}; head[v] = ecnt;
}
void dfs(int u, int f) {
    fa[u][0] = f;
    dep[u] = dep[f] + 1;
    int bot = top;
```

```
    for(int i = head[u]; i; i = e[i].nxt) {
        int v = e[i].to;
        if(v == f) continue;
        dfs(v, u);
        if(top - bot >= sz) {
            cnt++;
            while(top != bot) belong[stack[top--]] = cnt;
        }
    }
    stack[++top] = u;
}
void G(int &u, int step) {
    for(int i = 0; i < S; i++) if((1 << i) & step) u = fa[u][i];
}
int lca(int u, int v) {
    if(dep[u] > dep[v]) swap(u, v);
    G(v, dep[v] - dep[u]);
    if(u == v) return u;
    for(int i = S; i >= 0; i--) if(fa[u][i] != fa[v][i]) {
        u = fa[u][i]; v = fa[v][i];
    }
    return fa[u][0];
}
inline void modify(int u) {
    tans -= V[C[u]] * sum[cur[C[u]]];
    cur[C[u]] += vis[u];
    vis[u] = -vis[u];
    tans += V[C[u]] * sum[cur[C[u]]];
}
inline void update(int u, int v) {
    if(u == v) return;
    if(dep[u] > dep[v]) swap(u, v);
    while(dep[v] > dep[u]) {
        modify(v);
        v = fa[v][0];
    }
    while(u != v) {
        modify(u); modify(v);
        u = fa[u][0]; v = fa[v][0];
    }
}
inline void upd(int t) {
    if(vis[qq[t].l] == -1) {
        modify(qq[t].l);
        swap(C[qq[t].l], qq[t].r);
        modify(qq[t].l);
    }
    else swap(C[qq[t].l], qq[t].r);
}
inline void moveto(int u, int v) {
    update(l, u); update(r, v);
    l = u; r = v;
}
int main() {
    n = read(); m = read(); Q = read();
    sz = (int)pow(n, 2.0 / 3.0);
    for(int i = 1; i <= m; i++) V[i] = read();
    for(int i = 1; i <= n; i++) W[i] = read();
    for(int i = 1, u, v; i < n; i++) {
        u = read(); v = read();
        add_edge(u, v);
```

```
    }
    for(int i = 1; i <= n; i++) {
        C[i] = read();
        vis[i] = 1;
        sum[i] = sum[i - 1] + W[i];
    }
    for(int i = 1, tp; i <= Q; i++) {
        tp = read();
        if(tp) {
            ++Q1;
            q[Q1].l = read(); q[Q1].r = read();
            q[Q1].id = Q1;
            q[Q1].ti = i;
        }
        else {
            ++Q0;
            qq[Q0].l = read(); qq[Q0].r = read();
            qq[Q0].ti = i;
        }
    }
    dfs(1, 0);
    while(top) belong[stack[top--]] = cnt;
    sort(q + 1, q + Q1 + 1);
    for(int k = 1; k <= S; k++) {
        for(int i = 1; i <= n; i++) {
            fa[i][k] = fa[fa[i][k - 1]][k - 1];
        }
    }
    for(int i = 1; i <= Q1; i++) {
        if(belong[q[i].l] > belong[q[i].r]) swap(q[i].l, q[i].r);
        moveto(q[i].l, q[i].r);
        int lc = lca(l, r);
        modify(lc);
        while(qq[tm + 1].ti < q[i].ti && tm < Q0) upd(++tm);
        while(qq[tm].ti > q[i].ti) upd(tm--);
        ans[q[i].id] = tans;
        modify(lc);
    }
    for(int i = 1; i <= Q1; i++) printf("%lld\n", ans[i]);
    return 0;
}
```

7.2 Randomized Algorithm

7.2.1 Simulated Annealing

```
void solve() {
    while(T > eps) {
        double alpha = ((rand() % 30001) / 15000.0) * pi;
        double theta = ((rand() % 10001) / 10000.0) * pi;
        tmp.x = cur.x + T * sin(theta) * cos(alpha);
        tmp.y = cur.y + T * sin(theta) * sin(alpha);
        tmp.z = cur.z + T * cos(theta);
        tmp.dis = cal(tmp);
        if(tmp.dis < cur.dis || (tmp.dis * 0.999 < cur.dis && (rand() & 7) == 7)) cur =
    tmp;
        //if(exp((cur.d - tmp.d)  / T) > ((double)rand() / RAND_MAX)) cur = tmp;

        T *= 0.999;
```

```
    }
}
```

7.3 Other Method

7.3.1 Enumerate Subset

```
for(int i = 0; i < (1 << k); i++) {
    for(int j = i; ; --j &= i) {
        // work();
        if(j == 0) break;
    }
}
```

7.3.2 Enumerate $\lfloor \frac{n}{d} \rfloor \lfloor \frac{m}{d} \rfloor$

```
int cal(int n, int m) {
    if(n > m) swap(n, m);
    int res = 0, last;
    for(int i = 1; i <= n; i = last + 1) {
        last = min(n / (n / i), m / (m / i));
        res += (n / i) * (m / i) * (sum(last) - sum(i - 1));
    }
    return res;
}
```

7.3.3 Find Primitive Root Modulo N

```
for i in range(1,mod):
    if 3 ** i % mod == 1:
        if i == mod - 1:
            print("yes")
            break
        print("no")
```

8 Samples

8.1 vimrc

```
set cindent
set number
set mouse=a
set tabstop=4
set shiftwidth=4
syntax on
inoremap { {}<left>
map <F9> :w<CR> :! g++ % -o %< -Wall --std=c++14 -g && ./%< <CR>
"ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
```

8.2 Check

```
while true; do
    ./data > in
    ./tmp < in > out
    ./std < in > ans
    diff out ans
    if [ $? -ne 0 ] ; then exit; fi
    echo Passed
done
```

8.3 FastIO

```
//普通情况
namespace IO {
    const int MB = 1048576;
    const int RMAX = 16 * MB;
    const int WMAX = 16 * MB;
    #define getchar() *(rp++)
    #define putchar(x) (*(wp++) = (x))
    char rb[RMAX], *rp = rb,  wb[WMAX], *wp = wb;
    inline void init() {
        fread(rb, sizeof(char), RMAX, stdin);
    }
    template <class _T> inline void read(_T &_a) {
        _a = 0; bool _f = 0; int _c = getchar();
        while (_c < '0' || _c > '9') _f |= _c == '-', _c = getchar();
        while (_c >= '0' && _c <= '9') _a = _a * 10 + (_c ^ '0'), _c = getchar();
        _a = _f ? -_a : _a;
    }
    template <class _T> inline void write(_T _a) {
        static char buf[20], *top = buf;
        if (_a) {
            while (_a) {
                _T tm = _a / 10;
                *(++top) = char(_a - tm * 10) | '0';
                _a = tm;
            }
            while (top != buf) putchar(*(top--));
        }
        else putchar('0');
    }
```

```
    void output() {
        fwrite(wb, sizeof(char), wp - wb, stdout);
    }
}
//EOF结尾+分块读入
#define likely(x) __builtin_expect(!!(x), 1)
#define unlikely(x) __builtin_expect(!!(x), 0)
namespace IO {
    const int MB = 1048576;
    const int RMAX = 4 * MB;
    const int WMAX = 4 * MB;
    unsigned long long filesize;
    #define putchar(x) (*(wp++) = (x))
    char rb[RMAX], wb[WMAX], *wp = wb;
    int rp = 0;
    inline void init() {
        filesize = fread(rb, sizeof(char), RMAX, stdin);
        rp = 0;
        wp = wb;
    }
    void output() {
        fwrite(wb, sizeof(char), wp - wb, stdout);
    }
    inline char getCHAR(){
        if(unlikely(rp == filesize)){
            fwrite(wb, sizeof(char), wp - wb, stdout);
            init();
            if(unlikely(filesize == 0)) {
                //cerr << 1.0 * (clock() - st) / CLOCKS_PER_SEC << endl;
                exit(0);
            }
        }
        return rb[rp++];
    }
    template <class _T> inline void read(_T &_a) {
        _a = 0; static bool _f = 0; static int _c;
        _f = 0; _c = getCHAR();
        while (_c < '0' || _c > '9') _f |= _c == '-', _c = getCHAR();
        while (_c >= '0' && _c <= '9') _a = _a * 10 + (_c ^ '0'), _c = getCHAR();
        _a = _f ? -_a : _a;
    }
    template <class _T> inline void write(_T _a) {
        static char buf[20], *top = buf;
        if (_a) {
            while (_a) {
                _T tm = _a / 10;
                *(++top) = char(_a - tm * 10) | '0';
                _a = tm;
            }
            while (top != buf) putchar(*(top--));
        }
        else putchar('0');
        putchar('\n');
    }
}
```

8.4 Java BigNum

```
import java.math.*;
```

```
import java.util.*;
import java.lang.*;

public class Main{
    public static void main(String []args){}
}
//IO
Scanner in = new Scanner(System.in);
while(in.hasNext()){} //EOF
//fast-IO
public static void main(String argv[]) throws IOException{}
StreamTokenizer cin = new StreamTokenizer(new BufferedReader(new InputStreamReader(
    System.in)));
PrintWriter cout = new PrintWriter(new OutputStreamWriter(System.out));
while(cin.nextToken() != StreamTokenizer.TT_EOF) ;//EOF
cin.nextToken();int n = (int)cin.nval;String s = cin.sval;
cout.println( Type );cout.flush();
cin.ordinaryChar('/');

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
br.ready()//EOF
while ((valueString=bf.readLine())!=null);
br.close();
//true fast-IO
static class InputReader {
    public BufferedReader reader;
    public StringTokenizer tokenizer;

    public InputReader(InputStream stream) {
        reader = new BufferedReader(new InputStreamReader(stream), 32768);
        tokenizer = null;
    }

    public String next() {
        while (tokenizer == null || !tokenizer.hasMoreTokens()) {
            try {
                tokenizer = new StringTokenizer(reader.readLine());
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
        return tokenizer.nextToken();
    }

    public int nextInt() {
        return Integer.parseInt(next());
    }

}
//类 Number
//doubleValue()
//intValue()
//longValue()
//shortValue()
//类 BigDecimal
//ROUND_CEILING 接近正无穷大的舍入模式。
//ROUND_FLOOR 接近负无穷大的舍入模式。
//ROUND_DOWN 接近零的舍入模式
//ROUND_HALF_UP 四舍五入 >=0.5向上舍入
//ROUND_HALF_DOWN 四舍五入 >0.5向上舍入
//BigDecimal(BigInteger val)
```

```
//BigDecimal(BigInteger unscaledVal, int scale)
//BigDecimal(char[] in, int offset, int len, MathContext mc)
//BigDecimal(double val, MathContext mc)不建议
//BigDecimal(int val, MathContext mc)
//BigDecimal(long val, MathContext mc)
//BigDecimal(String val, MathContext mc)
//abs()
//add(BigDecimal augend, MathContext mc)
//compareTo(BigDecimal val)
//divide(BigDecimal divisor,MathContext mc)
//divideToIntegralValue(BigDecimal divisor, MathContext mc)
//max(BigDecimal val)
//min(BigDecimal val)
//multiply(BigDecimal multiplicand, MathContext mc)
//negate()  其值为 (-this), 其标度为 this.scale()
//pow(int n)
//remainder(BigDecimal divisor) 返回其值为 (this % divisor) 的 BigDecimal
//round(MathContext mc) 返回根据 MathContext 设置进行舍入后的 BigDecimal。
//caleByPowerOfTen(int n) 返回其数值等于 (this * 10^n) 的 BigDecimal。
//subtract(BigDecimal subtrahend, MathContext mc)
//setScale(int newScale,RoundingMode roundingMode)
//toString()
//ulp()返回此 BigDecimal 的 ulp（最后一位的单位）的大小
//String s = b.stripTrailingZeros().toPlainString();让bigdecimal不用科学计数法显示
//类 BigInteger
//parseInt
//BigInteger zero = BigInteger.valueOf(0);
//BigInteger a = in.nextBigInteger();
//abs()
//and(BigInteger val) 返回其值为 (this & val)
//or(BigInteger val) 返回其值为 (this | val)
//andNot(BigInteger val) 返回其值为 (this & ~val)
//compareTo(BigInteger val)
//add(BigInteger val)
//divide(BigInteger val)
//BigInteger[]  divideAndRemainder(BigInteger val) 返回包含 (this / val) 后跟 (this %
    val) 的两个 BigInteger 的数组。
//equals(Object x)
//gcd(BigInteger val)
//isProbablePrime(int certainty) e.g: a.isProbablePrime(4)
//max(BigInteger val) min(BigInteger val)
//mod(BigInteger m)
//modInverse(BigInteger m) 返回其值为 (this^-1 mod m)
//modPow(BigInteger exponent, BigInteger m) 返回其值为 (thisexponent mod m)
//multiply(BigInteger val)
//not() 返回其值为 (~this)
//shiftLeft(int n) 返回其值为 (this << n)
//shiftRight(int n) 返回其值为 (this >> n)
//toString()
//valueOf(long val)
//xor(BigInteger val) 返回其值为 (this ^ val)
//other
//Arrays.sort(array);
```

8.5 pb_ds

```
//P.S.:无脑正确使用pb_ds代替std::set/map/priority_queue不会变慢
//可持久化平衡树,不过时间和空间都不太行
#include <ext/rope>
```

```
using namespace __gnu_cxx;
int a[1000];
rope<int> x;
rope<int> x(a,a + n);
rope<int> a(x);
x->at(10);x[10];
x->push_back(x)      // 在末尾添加x
x->insert(pos,x)     // 在pos插入x
x->erase(pos,x)      // 从pos开始删除x个
x->replace(pos,x)    // 从pos开始换成x
x->substr(pos,x)     // 提取pos开始x个

//树
//不支持低级操作（如交换左右子树）
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
定义一颗红黑树
tree<int,null_type,less<int>,rb_tree_tag,tree_order_statistics_node_update>t, other;
int 关键字类型
null_type
无映射(低版本g++为null_mapped_type)（无映射为类似set，有映射类似map)
less<int>从小到大排序
rb_tree_tag 红黑树（splay_tree_tag)
tree_order_statistics_node_update结点更新(统计子树size，可自写)，不写不支持order_of_key
    以及find_by_order
插入:t.insert();
删除:t.erase();
比x小的个数:t.order_of_key(x);
第x+1值:t.find_by_order(x);
前驱:t.lower_bound();
后继:t.upper_bound();
合并:t.join(other); (other和*this值域不能相交)
分裂:t.split(x, other); (清空other，将t中比x小的元素移至other)

//自定义节点更新
template <class Node_CItr , class Node_Itr , class Cmp_Fn , class _Alloc >
struct my_node_update {
    virtual Node_CItr node_begin () const = 0;
    virtual Node_CItr node_end() const = 0;
    typedef char metadata_type;//节点上记录的额外信息的类型
    //以上为固定格式

    //operator()的功能是将节点it的信息更新为其左右孩子的信息之和，传入的end_it表示空节点
    //对Node_Itr可以做的事情有：用get_l_child,get_r_child获取左右孩子，用两个星号获取节
    点信息，用get_metadata获取节点额外信息
    inline void operator()(Node_Itr it, Node_CItr end_it) {
        Node_Itr l = it.get_l_child(), r = it.get_r_child();
        int left = 0, right = 0;
        if(l != end_it) left = l.get_metadata();
        if(r != end_it) right = r.get_metadata();
        const_cast<metadata_type &>(it.get_metadata()) = left + right + (*it)->second;
        //it是node_Itr，取*后变为iterator，再取->second变成mapped_value
    }
    inline int prefix_sum(int x) {
        int ans = 0;
        Node_CItr it = node_begin ();
        while(it != node_end()) {
            Node_CItr l = it.get_l_child(), r = it.get_r_child();
            if(Cmp_Fn()(x, (*it)->first)) it = l;
            else {
```

```
                ans += (*it)->second;
                if(l != node_end()) ans += l.get_metadata();
                it = r;
            }
        }
        return ans;
    }
    inline int interval_sum(int l, int r) {
        return prefix_sum(r) - prefix_sum(l - 1);
    }
};

tree<int, char, less<int>, rb_tree_tag, my_node_update> T;//map
int main() {
    T[2] = 'a'; T[3] = 'b'; T[4] = 1;
    cout << (char)T.interval_sum(3, 4) << endl;//c
    return 0;
}
//堆
#include <ext/pb_ds/priority_queue.hpp>
using namespace __gnu_pbds;
__gnu_pbds::priority_queue<int, std::less<int>, __gnu_pbds::pairing_heap_tag> q;

template <typename Value_Type ,
typename Cmp_Fn = std::less<Value_Type>,
typename Tag = pairing_heap_tag ,
typename Allocator = std::allocator<char> >
class priority_queue
Tag可以是binary_heap_tag（二叉堆）binomial_heap_tag（二项堆）rc_binomial_heap_tag
    pairing_heap_tag（配对堆）thin_heap_tag
用begin()和end()获取迭代器从而遍历
删除单个元素 void erase(point_iterator)
更改一个元素的值 void modify(point_iterator, const_reference)
合并 void join(priority_queue &other)，把other合并到*this，并把other清空
push()会返回迭代器
五种操作：push、pop、modify、erase、join
● pairing_heap_tag：push和join O(1)，其余均摊O(logn)
● binary_heap_tag：只支持push和pop，均为均摊O(logn)
● binomial_heap_tag：push为均摊O(1)，其余为O(logn)
● rc_binomial_heap_tag：push为O(1)，其余为O(logn)
● thin_heap_tag：push为O(1)，不支持join，其余为O(logn)；但是如果只有increase_key，modify
    均摊O(1)
● 不支持不是不能用，而是用起来很慢
经过实践检测得到的结论：
● Dijkstra算法中应用pairing_heap_tag，速度与手写数据结构相当。
● binary_heap_tag在绝大多数情况下优于std::priority_queue
● pairing_heap_tag在绝大多数情况优于binomial_heap_tag和rc_binomial_heap_tag
● 只有push，pop和join操作时，binary_heap_tag速度较快
● 有modify操作时，可以考虑thin_heap_tag或者pairing_heap_tag，或手写数据结构。

//hash_table
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/hash_policy.hpp>
using namespace __gnu_pbds;
__gnu_pbds::cc_hash_table <Key, Mapped> mp; //使用链地址法解决哈希冲突
__gnu_pbds::gp_hash_table <Key, Mapped> mp; //使用探测法解决哈希冲突
//用法和map一样
```