**更新于 20141015。**

**图论：**

匈牙利算法 + 判断某个点是否一定在最大匹配中

最大流 Dinic && dyf 式判断环（二分图上找 >2 的环；多校题目）

最小费用流

Tarjan 点强连通分量

（非递归）Tarjan 点强连通分量 by zb

（一年前）无向图点双连通分量

（一年前）无向图边双连通分量

倍增求 LCA

最大团 (大概跑到 n = 60 左右)

树的点分治（HDU4918 Query on the subtree；要求在线修改点权 + 询问距离一个点距离为 d 以内的点权和）

**数据结构：**

Trie 图

二进制 Trie 树

后缀数组，二分所有字串，找到字串的对应位置 —— HDU 5030

一维线段树，处理 Lazy 标记之间的关系（此题是操作为区间 0, 1 覆盖和翻转）

利用线段树维护哈希（CF452F，给个排列判断是否存在 (a+b)/2 在 a 和 b 之中间）

二维线段树（2013 长春 Regional 题目，单点修改，区间查询最值）

树状数组 1D 区间修改区间查询，利用差分技巧

树状数组 2D 区间修改区间查询，利用差分技巧，可以扩展到异或

笛卡尔树（根值最大）

Splay 平衡树

Deque 模拟块状链表，两个操作：把一段区间循环右移；求一段区间值为 k 的数量（k <= 1e5）

莫队算法（Acdream 1108，求一段区间所有数的频数的第 k 大）

可持续化线段树（HDU4866 多校第一场Shooting，平面上给定区间，询问一个 x 轴上可以打到的前 K 个区间距离和。把区间变成端点 +1，-1；离散化区间后在线点查询）

KD树（Acdream1197 在三维空间内一个立方体内的点数；用 KD 树暴力插入查询，最坏复杂度为平方级的，不过对付随机数据很好用）

KD树，求平面内距离一个点的最近 m 个点

CDQ 分治（Acdream 1157，三个操作，插入删除线段，查询完全包含某个线段的个数；将插入删除查询一起排序，按时间戳分治）

树链剖分（离线 DFS 序，对颜色建立线段树）求每个点最多被染上的颜色

正常的树链剖分（对树上的链建立线段树）上海网赛卖东西的题目

线段树套 Splay（用得上就见鬼了）

**动态规划：**

数位 DP （Acdream 1166；求在 [li, ri]，i <= 4 的范围内， x0 ^ .. x3 = 0 的 count 和 sum；将进位当成状态）

数位DP（求 n 以内的数满足 d0-d1+d2 .. = 0 的数有多少个；d0表示最高位，以此类推；因为最高位不能放零，需要记录起始位的是否存在）

数位DP（上海网赛 B 题，一共 m 个数每个限制为 0 <= ai <= Ri 求在 p 进制下求和不发生进位的方案数；每次DP放一个数的一个数位会很好写）

二维 Sparse-Table（一维显然同理）

斜率优化（玩具装箱）

使用的条件：较优的决策点对后续状态影响具有持续性，简单来说就是如果在状态I处，选择了J作为决策点，那在I+1处的决策点一定不可能在J之前；然后把DP方程写出来，左边化为J，K的形式，右边化为I的形式。

最长上升子序列 (NlgN)

**数学：**

浮点数高斯消元（一定有解）

同余高斯消元（一定有解；如果 P 很大，代回过程需要解模方程）

枚举自由变元（当然前提是有解，此题仅为二进制）

二进制高斯消元求基（CF 472F，将 A[] 通过异或变成 B[]，要求不用多余的槽位）

矩阵乘法

矩阵的逆

行列式求解（复杂度 O(n^3lgn)；不懂原理，只有板子）

辛普森积分

**数论：**

线性筛(N) + 快速幂(lgN) + 扩展欧几里得(lgN) + 逆元(lgN) + 分解质因数(NsqrtN)

离散对数，BSGS（A^X = B (mod P)，P 可以不为质数，时间复杂度 O(hash\*sqrtN) ；P为质数的短版本见白书）

卢卡斯定理 + 线性模方程 + 互质版中国剩余定理

非互质版中国剩余定理(两两合并模方程)

MillerRabin判素 + PollardRho找约数（模板题 Poj1181）

求解二次剩余（求解 x^2 = d (mod P)，P 为奇质数；URAL 1132）

合并加法的技巧（Acdream1136，求树上相邻点对乘积 <=P 的方案数）

**组合：**

Polya定理 (Hdu4633，有 n 种颜色，颜色可以无限用，正方体 8 个点，12 条边，54 个小面进行染色的方案数)

Burnside定理 （Acdream1022 给定 m 个颜色，颜色个数有限制，对正方体边长进行染色，求旋转不相同的方案数）

快速傅里叶变换 FFT (by cxlove)

题目一 —— 2013 多校，给定 1e5 个边长问组成三角形的方案数

题目二 —— CF472G，求两个 01 串之间的曼哈顿距离（还要分块）

三次方程卡尔丹公式（by thu 保证有解时调用，全程 complex）

**几何：**

头文件

最近点对

给定 N 个圆求每个圆被几个圆完美包含（HDU 3511，利用 set 做花式扫描线）

求最大被至少两个矩形完美覆盖的圆半径（ASC某题，做法为二分半径，缩小矩形后判断矩形是否有重合；具体做法为离散化矩形左右侧端点，将 set 中的线段元素的小于号重载成不相交）

**其他：**

求最长回文字串 Manacher 算法

KMP + EXTKMP

DLX 精确覆盖（HDU4069，花式数独判唯一性）

DLX 最小重复覆盖（HDU3498）

矩形离散化（UVA1092，对线段离散化要对块离散化简单不少，后者感觉完全不可写）

可重复K短路（A\*）

**图论——**

**匈牙利算法 + 判断某个点是否一定在最大匹配中**

bool gao(int u, bool f1, bool f2){ // f1 means left or right, f2 means change match or not ..

for(int i = first[u]; i != -1; i = e[i].next){

int v = e[i].v;

if(!vis[v]){

vis[v] = true;

if(f1 == 0){

if(matchy[v] == -1 || gao(matchy[v], f1, f2)){

if(!f2) matchy[v] = u, matchx[u] = v;

return true;

}

} else{

if(matchx[v] == -1 || gao(matchx[v], f1, f2)){

if(!f2) matchx[v] = u, matchy[u] = v;

return true;

}

}

}

}

return false;

}

int Hungary(){

memset(matchx, -1, sizeof(matchx));

memset(matchy, -1, sizeof(matchy));

int ret = 0;

for(int i = 0; i < X.size(); i ++){

memset(vis, 0, sizeof(vis));

if(gao(X[i], 0, 0)) ret ++;

}

return ret;

}

**最大流 Dinic && dyf 式判断环（二分图上找 >2 的环；多校题目）**

struct edge{

int u, v, next, flow;

}e[M];

int first[N], nw[N], level[N], ecnt, pre[M];

void addedge(int u, int v, int flow){

e[++ ecnt].next = first[u], first[u] = ecnt, e[ecnt].u = u, e[ecnt].v = v, e[ecnt].flow = flow;

e[++ ecnt].next = first[v], first[v] = ecnt, e[ecnt].u = v, e[ecnt].v = u, e[ecnt].flow = 0;

}

/\* by ztb by mt

\*/

bool Make\_level(int S, int T){

memset(level, 0, sizeof(level));

memcpy(nw, first, sizeof(first));

level[S] = 1;

queue <int> Q;

Q.push(S);

while(!Q.empty()){

int f = Q.front(); Q.pop();

for(int i = first[f]; i != -1; i = e[i].next)

if(e[i].flow && !level[e[i].v]){

level[e[i].v] = level[f] + 1;

Q.push(e[i].v);

if(e[i].v == T) return true;

}

}

return false;

}

int Maxflow(int S, int T){

int i = S, j, flow = 0;

while(level[S]){

for(j = nw[i]; j != -1; j = e[j].next)

if(e[j].flow && level[i] + 1 == level[e[j].v])

break;

nw[i] = j;

if(j != -1){

i = e[j].v, pre[i] = j;

if(i == T){

int delta = INF;

for(int k = T; k != S; k = e[pre[k]].u)

if(e[pre[k]].flow < delta)

delta = e[pre[k]].flow, i = e[pre[k]].u;

flow += delta;

for(int k = T; k != S; k = e[pre[k]].u)

e[pre[k] ^ 1].flow += delta, e[pre[k]].flow -= delta;

}

} else{

level[i] = 0;

i = e[pre[i]].u;

}

}

return flow;

}

int Dinic(int S, int T){

int res = 0;

while(Make\_level(S, T))

res += Maxflow(S, T);

return res;

}

/\* by dyf

\*/

bool used[N]; stack <int> stk;

int ans[N][N];

bool Dfs(int u, int pre) {

int biu = -1;

used[u] = true;

for(int i = first[u]; i != -1; i = e[i].next){

int v = e[i].v;

if(e[i].v == pre){

biu = i;

continue;

}

if(e[i].flow > 0){

if(used[v]) return true;

if(Dfs(v, u)) return true;

}

if(biu == -1) first[u] = e[i].next;

else e[biu].next = e[i].next;

biu = i;

}

used[u] = false;

return false;

}

int n, m, K, r, c;

bool Check(){

for(int i = 1; i <= n; i ++){

int sum = 0;

for(int j = first[i]; j != -1; j = e[j].next){

int v = e[j].v;

if(v == 0) continue;

ans[i][v - n] = K - e[j].flow;

}

}

memset(used, 0, sizeof(used));

for(int i = 0; i <= n + m + 1; i ++){

if(Dfs(i, -1))

return false;

}

return true;

}

int main(){

// freopen("1002.in", "r", stdin);

// freopen("out.txt", "w", stdout);

while(scanf("%d%d%d", &n, &m, &K) == 3){

memset(first, -1, sizeof(first));

ecnt = -1;

int sumr = 0, sumc = 0;

for(int i = 1; i <= n; i ++) scanf("%d", &r), addedge(0, i, r), sumr += r;

for(int i = 1; i <= m; i ++) scanf("%d", &c), addedge(n + i, n + m + 1, c), sumc += c;

for(int i = 1; i <= n; i ++)

for(int j = 1; j <= m; j ++)

addedge(i, n + j, K);

int res = Dinic(0, n + m + 1);

if(res != sumr || res != sumc){

puts("Impossible"); continue;

}

if(!Check()){

puts("Not Unique");

} else{

puts("Unique");

for(int i = 1; i <= n; i ++)

for(int j = 1; j <= m; j ++)

printf("%d%c", ans[i][j], j == m ? '\n' : ' ');

}

}

}

**最小费用流**

struct edge{

int u, v, next, flow, cost;

}e[MAXM];

int ecnt, first[MAXN], dist[MAXN], pre[MAXN];

bool flag[MAXN];

void addedge(int u, int v, int flow, int cost){

//cout << u << ' ' << v << ' ' << flow << ' ' << cost << endl;

e[++ ecnt].next = first[u], first[u] = ecnt, e[ecnt].u = u, e[ecnt].v = v, e[ecnt].flow = flow, e[ecnt].cost = cost;

e[++ ecnt].next = first[v], first[v] = ecnt, e[ecnt].u = v, e[ecnt].v = u, e[ecnt].flow = 0, e[ecnt].cost = -cost;

}

bool make\_map(int S, int T){

for(int i = 0; i < MAXN ; i ++)

dist[i] = INF, pre[i] = -1;

memset(flag, 0, sizeof(flag));

dist[S] = 0;

queue <int> Q;

Q.push(S);

while(!Q.empty()){

int f = Q.front(); Q.pop();

flag[f] = false;

for(int i = first[f]; i != -1; i = e[i].next)

if(e[i].flow && dist[e[i].v] > dist[f] + e[i].cost){

dist[e[i].v] = dist[f] + e[i].cost,

pre[e[i].v] = i;

if(!flag[e[i].v])

Q.push(e[i].v), flag[e[i].v]=true;

}

}

return dist[T] != INF;

}

int mincost\_maxflow(int S, int T){

int ans = 0;

while(make\_map(S, T)){

int delta = INF, cost = 0;

for(int i = pre[T]; i != -1; i = pre[e[i].u])

delta = min(e[i].flow, delta), cost += e[i].cost;

for(int i = pre[T]; i != -1; i = pre[e[i].u])

e[i].flow -= delta, e[i ^ 1].flow += delta;

ans += delta \* cost;

}

return ans;

}

**Tarjan 点强连通分量**

int dfn[MAXN], low[MAXN], scc\_id[MAXN], cnt, idx, scc\_cnt;

vector <int> vec[MAXN];

stack <int> stk;

void dfs(int u){

stk.push(u);

dfn[u] = low[u] = ++ idx;

for(int i = first[u]; i != -1; i = e[i].next){

int v = e[i].v;

if(!dfn[v])

dfs(v), low[u] = min(low[u], low[v]);

else if(!scc\_id[v])

low[u] = min(low[u], dfn[v]);

}

if(dfn[u] == low[u]){

scc\_cnt ++;

while(1){

int x = stk.top(); stk.pop();

vec[scc\_cnt].pb(x);

scc\_id[x] = scc\_cnt;

if(x == u) break;

}

}

}

int n, x;

void gao(){

cnt = idx = 0;

for(int i = 1; i <= n; i ++)

if(!dfn[i]) dfs(i);

}

**（非递归）Tarjan 点强连通分量 by zb**

struct Snode{

int x, now, opt;

Snode(){}

Snode(int \_x, int \_now, int \_opt){x = \_x, now = \_now, opt = \_opt;}

};

struct Edge{

int y, next;

} e[M];

int tot, head[N], fa[N], cnt[N], dfn[N], low[N], v[N], x[M], y[M], out[N], inz[N];

Snode s[N];

stack <int> z;

void Addedge(int x, int y){

e[++tot].y = y; e[tot].next = head[x]; head[x] = tot;

}

int top, tim;

void Pop(int x){

fa[x] = x, cnt[x]++;

while (z.top() != x){

inz[z.top()] = 0;

fa[z.top()] = x;

cnt[x]++;

z.pop();

}

inz[x] = 0;

z.pop();

}

void Push(int x){

s[++top] = Snode(x, head[x], 0);

v[x] = 1;

z.push(x), inz[x] = 1;

dfn[x] = low[x] = ++tim;

}

void Tarjan(int st){

top = 0, tim = 0;

Push(st);

while (top){

int x = s[top].x, now = s[top].now, opt = s[top].opt;

if (now == -1){

if (low[x] == dfn[x]) Pop(x);

top--;

v[x] = 2;

}else if (opt){

s[top] = Snode(x, e[now].next, 0);

int y = e[now].y;

low[x] = min(low[x], low[y]);

}else{

int y = e[now].y;

if (!v[y]){

s[top] = Snode(x, now, 1);

Push(y);

}else{

if (inz[y]) low[x] = min(low[x], dfn[y]);

s[top] = Snode(x, e[now].next, 0);

}

}

}

}

**（一年前）无向图点双连通分量**

void dfs(int x,int f){

low[x]=dfn[x]=++index;

for(int i=0,s=e[x].size();i<s;i++){

int v=e[x][i];

if(!dfn[e[x][i]]){

stack[++cnt][1]=x,stack[cnt][2]=v;

dfs(e[x][i],x);

low[x]=min(low[x],low[e[x][i]]);

if(low[v]>=dfn[x]){

ne[++scnt].clear();

while(1){

int xx=stack[cnt][1],vv=stack[cnt--][2];

if(belong[xx]!=scnt)

ne[scnt].pb(xx),belong[xx]=scnt;

if(belong[vv]!=scnt)

ne[scnt].pb(vv),belong[vv]=scnt;

if(xx==x && vv==v)break;

}

}

}

else if(dfn[v]<dfn[x] && v!=f)

low[x]=min(low[x],dfn[v]);

}

}

**（一年前）无向图边双连通分量**

void dfs(int x){

low[x]=dfn[x]=++id;

for(int i=first[x];i!=-1;i=e[i].next)if(!e[i].vis){

int v=e[i].v;

e[i].vis=e[i^1].vis=true;

if(!dfn[v])dfs(v),low[x]=min(low[x],low[v]);

else low[x]=min(low[x],dfn[v]);

}

}

for(int i=1;i<=n;i++)

if(!dfn[i])

dfs(i);

for(int i=0;i<=cnt;i++)

if(low[e[i].v]>dfn[e[i].u])

e[i].cut=e[i^1].cut=true,ans++;

**倍增求 LCA**

void init\_lca(){

for(int i = 1; (1 << i) < MAXN; i ++)

for(int j = 1; j < MAXN; j ++)

father[j][i] = father[ father[j][i - 1] ][i - 1];

}

int lca(int x, int y){

if(deep[x] < deep[y])

swap(x, y);

int temp = deep[x] - deep[y];

for(int i = 0; (1 << i) <= temp; i++)

if(temp & (1 << i))

x = father[x][i];

if(x == y) return x;

for(int i = 20; i >= 0; i --)

if(father[x][i] != father[y][i])

x = father[x][i], y = father[y][i];

return father[x][0];

}

**最大团 (大概跑到 n = 60 左右)**

int dp[55];

//int x[maxn]; // 取决于是否需要最优解.若需要，还要增加一个path[maxn]数组

bool dfs(int \*v, int tot, int depth){

int t[55];

if(tot == 0){

if(best < depth){

//for (int i = 0; i < depth; i ++) path[i] = x[i]; //路径信息2

best = depth;

return true;

}

return false;

}

for(int i = 0; i < tot; i ++){

if(depth + (tot - i) <= best) return false;

if(depth + dp[v[i]] <= best) return false;

//x[depth] = v[i]; //路径信息3

int k = 0;

for(int j = i + 1; j < tot; j ++)

if(mat[v[i]][v[j]]) t[k ++] = v[j];

if(dfs(t, k, depth + 1)) return true;

}

return false;

}

int MaximumClique(int n){

int v[N];

best = 0;

for(int i = n - 1; i >= 0; i --){

//x[0] = i; 路径信息1

int k = 0;

for(int j = i + 1; j < n; j ++)

if(mat[i][j]) v[k ++] = j;

dfs(v, k, 1);

dp[i] = best;

}

return best;

}

**树的点分治（HDU4918 Query on the subtree；要求在线修改点权 + 询问距离一个点距离为 d 以内的点权和）**

#pragma comment(linker, "/STACK:36777216")

#include <map>

#include <set>

#include <cmath>

#include <ctime>

#include <deque>

#include <queue>

#include <stack>

#include <cstdio>

#include <string>

#include <vector>

#include <iomanip>

#include <cassert>

#include <cstdlib>

#include <cstring>

#include <iostream>

#include <algorithm>

#define lc(x) (x << 1)

#define rc(x) (lc(x) + 1)

#define lowbit(x) (x & (-x))

#define PI (acos(-1))

#define lowbit(x) (x & (-x))

#define LL long long

#define DB long double

#define ULL unsigned long long

#define PII pair<int, int>

#define PLL pair<LL, LL>

#define PB push\_back

#define MP make\_pair

using namespace std;

const int N = 1e5 + 7;

const int INF = 0x3fffffff;

const int MOD = 1e9 + 7;

const DB EPS = 1e-8;

int sum[N \* 55], st[N \* 55], delta[N \* 55], totID, totLen;

void Allocate(int id, int len){

st[id] = totLen;

delta[id] = len;

for(int i = st[id] + 1; i <= st[id] + len; i ++) sum[i] = 0;

totLen += len;

}

void Add(int id, int x, int v){

for(; x <= delta[id]; x += lowbit(x)){

sum[st[id] + x] += v;

}

}

int Query(int id, int x){

int res = 0;

x = min(x, delta[id]);

for(; x; x -= lowbit(x)){

res += sum[st[id] + x];

}

return res;

}

map <int, int> M[N];

struct Node{

int fa, sub, dist;

Node(){}

Node(int \_fa, int \_sub, int \_dist){

fa = \_fa, sub = \_sub, dist = \_dist;

}

};

vector <Node> Entries[N];

//////////////////////////////////////////////////////////

int first[N], ecnt, size[N], w[N];

int heavy, totsize, MN;

bool del[N];

struct edge{

int next, v;

} e[N << 1];

void addedge(int u, int v){

e[++ ecnt].next = first[u], first[u] = ecnt, e[ecnt].v = v;

}

void dfs\_size(int u, int pre){

size[u] = 1;

for(int i = first[u]; i != -1; i = e[i].next)

if(e[i].v != pre && !del[e[i].v])

dfs\_size(e[i].v, u), size[u] += size[e[i].v];

}

void dp(int u, int pre){

int tmp = 0;

for(int i = first[u]; i != -1; i = e[i].next)

if(e[i].v != pre && !del[e[i].v])

dp(e[i].v, u), tmp = max(tmp, size[e[i].v]);

tmp = max(tmp, totsize - size[u]);

if(tmp < MN) MN = tmp, heavy = u;

}

void dfs\_get\_heavy(int u){

heavy = -1, MN = INF;

dfs\_size(u, 0), totsize = size[u];

dp(u, 0);

}

vector <PII> vec;

void gao(int u, int pre, int dist){

if(dist != 0) vec.PB(MP(dist, u));

for(int i = first[u]; i != -1; i = e[i].next){

int v = e[i].v;

if(del[v]) continue;

if(v == pre) continue;

gao(v, u, dist + 1);

}

}

void dfs(int u){

dfs\_get\_heavy(u);

u = heavy;

del[u] = true;

if(totsize == 1) return;

M[u][-1] = ++ totID;

Allocate(totID, totsize - 1);

Entries[u].PB(Node(u, -1, 0));

for(int i = first[u]; i != -1; i = e[i].next){

int v = e[i].v;

if(del[v]) continue;

vec.clear();

M[u][v] = ++ totID;

gao(v, u, 1);

Allocate(totID, vec.size());

for(int j = 0; j < vec.size(); j ++){

Entries[vec[j].second].PB(Node(u, v, vec[j].first));

Entries[vec[j].second].PB(Node(u, -1, vec[j].first));

Add(M[u][v], vec[j].first, w[vec[j].second]);

Add(M[u][-1], vec[j].first, w[vec[j].second]);

}

}

for(int i = first[u]; i != -1; i = e[i].next)

if(!del[e[i].v])

dfs(e[i].v);

}

//////////////////////////////////////////////////////////

int n, m, u, v, p[N]; char op;

int main(){

int size = 256 << 20; // 256MB

char \*p = (char\*)malloc(size) + size;

\_\_asm\_\_("movl %0, %%esp\n" :: "r"(p));

// freopen("in.txt", "r", stdin);

while(scanf("%d%d", &n, &m) == 2){

for(int i = 1; i <= n; i ++){

first[i] = -1, del[i] = 0;

Entries[i].clear();

M[i].clear();

}

totID = totLen = 0;

ecnt = -1;

for(int i = 1; i <= n; i ++)

scanf("%d", &w[i]);

for(int i = 1; i < n; i ++){

scanf("%d%d", &u, &v);

addedge(u, v), addedge(v, u);

}

dfs(1);

while(m --){

scanf(" %c%d%d", &op, &u, &v);

if(op == '!'){

int dd = v - w[u];

for(int i = 0; i < Entries[u].size(); i ++){

int fa = Entries[u][i].fa, sub = Entries[u][i].sub, di = Entries[u][i].dist;

if(di == 0) continue;

Add(M[fa][sub], di, dd);

}

w[u] = v;

} else{

int res = w[u];

for(int i = 0; i < Entries[u].size(); i ++){

int fa = Entries[u][i].fa, sub = Entries[u][i].sub, di = Entries[u][i].dist;

if(v < di) continue;

if(sub == -1 && u != fa) res += w[fa];

if(sub == -1) res += Query(M[fa][-1], v - di);

else res -= Query(M[fa][sub], v - di);

}

printf("%d\n", res);

}

}

}

}

**数据结构——**

**Trie 图**

struct trie{

int fail, son[4];

bool flag;

} tree[N];

int totsz = 0;

void insert(char \*word){

int p = 0;

for(int i = 0, len = strlen(word); i < len; i ++){

int t = H(word[i]);

if(!tree[p].son[t])

tree[p].son[t] = ++ totsz;

p = tree[p].son[t];

}

tree[p].flag = true;

}

void build\_trie\_map(){

queue <int> Q;

for(int i = 0; i < 4; i ++)

if(tree[0].son[i])

Q.push(tree[0].son[i]);

while(!Q.empty()){

int u = Q.front(); Q.pop();

for(int i = 0; i < 4; i ++){

if(tree[u].son[i]){

tree[tree[u].son[i]].fail = tree[tree[u].fail].son[i];

if(tree[tree[tree[u].fail].son[i]].flag)

tree[tree[u].son[i]].flag = true;

Q.push(tree[u].son[i]);

}

else

tree[u].son[i] = tree[tree[u].fail].son[i];

}

}

}

**二进制 Trie 树**

void insert(ll x){

int p=0;//注意建树的顺序..应该从高位往低位建..

for(int i=50;i>=0;i--){

bool temp=x&(1LL<<i);

if(!tree[p].son[temp])

tree[p].son[temp]=++size;

p=tree[p].son[temp];

}

}

ll check(ll x){

ll ans=0;

int p=0;

for(int i=50;i>=0;i--){

bool temp=!(x&((1LL)<<i));

if(tree[p].son[temp])

p=tree[p].son[temp],ans|=(1LL<<i);

else

p=tree[p].son[!temp];

}

return ans;

}

**后缀数组，二分所有字串，找到字串的对应位置 —— HDU 5030**

int wa[N], wb[N], S[N];

int sa[N], rank[N], height[N];

void da(char \*ch, int n, int m){

// 不用初始化 ..

int i, j, p = 0, \*X = wa, \*Y = wb;

for(i = 0; i < m; i ++) S[i] = 0;

for(i = 0; i < n; i ++) S[X[i] = ch[i]] ++;

for(i = 1; i < m; i ++) S[i] += S[i - 1];

for(i = n - 1; i >= 0; i --) sa[-- S[X[i]]] = i;

for(j = 1; p < n; j \*= 2, m = p){

for(p = 0, i = n - j; i < n; i ++) Y[p ++] = i;

for(i = 0; i < n; i ++) if(sa[i] >= j) Y[p ++] = sa[i] - j;

for(i = 0; i < m; i ++) S[i] = 0;

for(i = 0; i < n; i ++) S[X[Y[i]]] ++;

for(i = 1; i < m; i ++) S[i] += S[i - 1];

for(i = n - 1; i >= 0; i --) sa[-- S[X[Y[i]]]] = Y[i];

swap(X, Y);

for(p = 1, X[sa[0]] = 0, i = 1; i < n; i ++)

X[sa[i]] = (Y[sa[i-1]] == Y[sa[i]] && Y[sa[i - 1] + j] == Y[sa[i] + j]) ? p - 1 : p ++;

}

}

void calcheight(char \*ch, int n){

int i, j, k = 0;

for(i = 0; i <= n; i ++) rank[sa[i]] = i;

for(i = 0; i < n; height[rank[i ++]] = k)

for(k ? k -- : 0, j = sa[rank[i] - 1]; ch[i + k] == ch[j + k]; k ++);

}

int n, m, vis[N]; LL sum[N];

char ch[N];

PII calc(LL x){

// 二分找到字串位置 ..

int u = lower\_bound(sum, sum + 1 + n, x) - sum;

return MP(u, x - sum[u - 1] + height[u]);

}

bool check(LL x){

PII u = calc(x);

int pos = u.first, len = u.second;

memset(vis, -1, sizeof(vis));

vis[sa[pos] + len] = sa[pos];

int minv = len;

for(int i = pos + 1; i <= n; i ++){

minv = std::min(minv, height[i]);

vis[sa[i] + minv] = std::max(vis[sa[i] + minv], sa[i]);

if(minv == 0) return 0;

}

int last = 0, cnt = 1;

for(int i = 0; i < n; i ++){

if(vis[i] >= last)

cnt ++, last = i;

}

return cnt <= m;

}

int main(){

while(scanf("%d", &m) == 1){

if(m == 0) break;

scanf(" %s", ch);

if(m == 1){

printf("%s\n", ch);

continue;

}

n = strlen(ch);

da(ch, n + 1, 200), calcheight(ch, n);

sum[0] = 0;

// 二分所有字串 .. 注意 LL

for(int i = 1; i <= n; i ++){

sum[i] = sum[i - 1] + n - sa[i] - height[i];

}

LL l = 1, r = sum[n];

while(l < r){

LL mid = l + r >> 1;

if(check(mid)) r = mid;

else l = mid + 1;

}

PII u = calc(l);

int pos = u.first, len = u.second;

for(int i = sa[pos]; i < sa[pos] + len; i ++)

printf("%c", ch[i]);

puts("");

}

}

**一维线段树，处理 Lazy 标记之间的关系**

**（此题是操作为区间 0, 1 覆盖和翻转）**

struct seg{

int mk, sum;

} tree[4][MAXN << 2];

int ll[MAXN << 2], rr[MAXN << 2];

int a[MAXN];

void init\_tree(int l, int r, int x){

ll[x] = l, rr[x] = r;

for(int i = 0; i <= 3; i ++)

tree[i][x].mk = 0;

if(l == r){

for(int i = 0; i <= 3; i ++)

tree[i][x].sum = (a[l] >> i & 1);

return;

}

int mid = l + r >> 1;

init\_tree(l, mid, lc(x));

init\_tree(mid + 1, r, rc(x));

for(int i = 0; i <= 3; i ++)

tree[i][x].sum = tree[i][lc(x)].sum + tree[i][rc(x)].sum;

}

int pd(int x){

for(int i = 0; i <= 3; i ++){

if(tree[i][x].mk != 0){

if(tree[i][x].mk == 1){

tree[i][lc(x)].sum = tree[i][rc(x)].sum = 0;

}

else if(tree[i][x].mk == 2){

tree[i][lc(x)].sum = rr[lc(x)] - ll[lc(x)] + 1;

tree[i][rc(x)].sum = rr[rc(x)] - ll[rc(x)] + 1;

}

else if(tree[i][x].mk == 3){

tree[i][lc(x)].sum = rr[lc(x)] - ll[lc(x)] + 1 - tree[i][lc(x)].sum;

tree[i][rc(x)].sum = rr[rc(x)] - ll[rc(x)] + 1 - tree[i][rc(x)].sum;

}

tree[i][lc(x)].mk = tree[i][rc(x)].mk = tree[i][x].mk;

tree[i][x].mk = 0;

}

}

}

int id, op;

void gao(int l, int r, int x){

if(l <= ll[x] && rr[x] <= r){

for(int i = 0; i <= 3; i ++){

int k = (op >> i & 1);

if(id == 1){

if(k == 0) tree[i][x].sum = 0, tree[i][x].mk = 1;

}

if(id == 2){

if(k == 1) tree[i][x].sum = rr[x] - ll[x] + 1, tree[i][x].mk = 2;

}

if(id == 3){

if(k == 1) tree[i][x].sum = rr[x] - ll[x] + 1 - tree[i][x].sum, tree[i][x].mk = 3;

}

}

return;

}

pd(x);

int mid = ll[x] + rr[x] >> 1;

if(l <= mid) gao(l, r, lc(x));

if(r > mid) gao(l, r, rc(x));

for(int i = 0; i <= 3; i ++)

tree[i][x].sum = tree[i][lc(x)].sum + tree[i][rc(x)].sum;

}

int query(int l, int r, int x){

if(l <= ll[x] && rr[x] <= r){

int res = 0;

for(int i = 0; i <= 3; i ++)

res += (tree[i][x].sum \* (1 << i));

return res;

}

pd(x);

int mid = ll[x] + rr[x] >> 1;

int res = 0;

if(l <= mid) res += query(l, r, lc(x));

if(r > mid) res += query(l, r, rc(x));

return res;

}

int t, n, m, l, r;

char ch[10];

int main(){

//freopen("in.txt", "r", stdin);

cin >> t;

while(t --){

scanf("%d%d", &n, &m);

for(int i = 1; i <= n; i ++)

scanf("%d", &a[i]);

init\_tree(1, n, 1);

while(m --){

scanf(" %s", &ch);

if(ch[0] == 'S'){

scanf("%d%d", &l, &r); l ++, r ++;

int k = query(l, r, 1);

printf("%d\n", k);

}

if(ch[0] == 'A'){

id = 1;

scanf("%d%d%d", &op, &l, &r); l ++, r ++;

gao(l, r, 1);

}

if(ch[0] == 'O'){

id = 2;

scanf("%d%d%d", &op, &l, &r); l ++, r ++;

gao(l, r, 1);

}

if(ch[0] == 'X'){

id = 3;

scanf("%d%d%d", &op, &l, &r); l ++, r ++;

gao(l, r, 1);

}

}

}

}

**利用线段树维护哈希**

**（CF452F，给个排列判断是否存在 (a+b)/2 在 a 和 b 之中间）**

const int co = 777;

LL po[N];

struct seg{

int l, r, sz; LL f, g;

int mid(){

return l + r >> 1;

}

} tree[N << 2];

void init\_tree(int l, int r, int x){

tree[x].l = l, tree[x].r = r, tree[x].sz = r - l + 1;

if(l == r) return;

int mid = l + r >> 1;

init\_tree(l, mid, lc(x));

init\_tree(mid + 1, r, rc(x));

}

void insert(int pos, int x){

if(tree[x].l == tree[x].r){

tree[x].f = tree[x].g = 1;

return;

}

int mid = tree[x].mid();

if(pos <= mid) insert(pos, lc(x));

else insert(pos, rc(x));

tree[x].f = (tree[lc(x)].f \* po[tree[rc(x)].sz] + tree[rc(x)].f);

tree[x].g = (tree[rc(x)].g \* po[tree[lc(x)].sz] + tree[lc(x)].g);

}

LL query(int l, int r, int x, int id){

if(l <= tree[x].l && tree[x].r <= r){

if(id == 1) return tree[x].f;

return tree[x].g;

}

int mid = tree[x].mid(); LL res;

if(id == 1){

if(l <= mid && r > mid) res = query(l, r, lc(x), id) \* po[min(tree[x].r, r) - mid] + query(l, r, rc(x), id);

else if(l <= mid) res = query(l, r, lc(x), id);

else res = query(l, r, rc(x), id);

}

if(id == 2){

if(l <= mid && r > mid) res = query(l, r, rc(x), id) \* po[mid - max(tree[x].l, l) + 1] + query(l, r, lc(x), id);

else if(l <= mid) res = query(l, r, lc(x), id);

else res = query(l, r, rc(x), id);

}

return res;

}

int n, x;

int main(){

//freopen("in.txt", "r", stdin);

po[0] = 1;

for(int i = 1; i < N; i ++)

po[i] = po[i - 1] \* co;

cin >> n;

init\_tree(1, n, 1);

for(int i = 1; i <= n; i ++){

scanf("%d", &x);

insert(x, 1);

int mn = max(1, 2 \* x - n);

int mx = min(n, 2 \* x - 1);

//cout << mn << ' ' << x << ' ' << mx << endl;

//cout << query(mn, x, 1, 1) << ' ' << query(x, mx, 1, 2) << endl;

if(query(mn, x, 1, 1) != query(x, mx, 1, 2)){

puts("YES");

return 0;

}

}

puts("NO");

}

**二维线段树（2013 长春 Regional 题目，单点修改，区间查询最值）**

struct \_2dseg{

int mn, mx;

} tree[MAXN << 2][MAXN << 2];

pair<int, int> calcy(int x, int y, int ll, int rr, int ly, int ry){

if(ly <= ll && rr <= ry)

return mp(tree[x][y].mn, tree[x][y].mx);

int mid = ll + rr >> 1;

int mx = -INF, mn = INF;

if(ly <= mid){

pair<int, int> tmp = calcy(x, lc(y), ll, mid, ly, ry);

mn = min(mn, tmp.first);

mx = max(mx, tmp.second);

}

if(mid < ry){

pair<int, int> tmp = calcy(x, rc(y), mid + 1, rr, ly, ry);

mn = min(mn, tmp.first);

mx = max(mx, tmp.second);

}

return mp(mn, mx);

}

pair<int, int> calcx(int x, int ll, int rr, int lx, int rx, int ly, int ry){

if(lx <= ll && rr <= rx)

return calcy(x, 1, 1, n, ly, ry);

int mid = ll + rr >> 1;

int mx = -INF, mn = INF;

if(lx <= mid){

pair<int, int> tmp = calcx(lc(x), ll, mid, lx, rx, ly, ry);

mn = min(mn, tmp.first);

mx = max(mx, tmp.second);

}

if(mid < rx){

pair<int, int> tmp = calcx(rc(x), mid + 1, rr, lx, rx, ly, ry);

mn = min(mn, tmp.first);

mx = max(mx, tmp.second);

}

return mp(mn, mx);

}

void gaoy(int x, int y, int ll, int rr, int posy, int num, bool leaf){

if(ll == rr){

if(leaf) tree[x][y].mn = tree[x][y].mx = num;

else{

tree[x][y].mn = min(tree[lc(x)][y].mn, tree[rc(x)][y].mn);

tree[x][y].mx = max(tree[lc(x)][y].mx, tree[rc(x)][y].mx);

}

return;

}

int mid = ll + rr >> 1;

if(posy <= mid) gaoy(x, lc(y), ll, mid, posy, num, leaf);

else gaoy(x, rc(y), mid + 1, rr, posy, num, leaf);

tree[x][y].mn = min(tree[x][lc(y)].mn, tree[x][rc(y)].mn);

tree[x][y].mx = max(tree[x][lc(y)].mx, tree[x][rc(y)].mx);

}

void gaox(int x, int ll, int rr, int posx, int posy, int num){

if(ll == rr){

gaoy(x, 1, 1, n, posy, num, true);

return;

}

int mid = ll + rr >> 1;

if(posx <= mid) gaox(lc(x), ll, mid, posx, posy, num);

else gaox(rc(x), mid + 1, rr, posx, posy, num);

gaoy(x, 1, 1, n, posy, num, false);

}

**树状数组 1D 区间修改区间查询，利用差分技巧**

ll sum\_1[maxn],sum\_2[maxn],a[maxn],v,n,q,l,r;

char ch;

void insert(ll sum[],int x,ll v){

for(;x<=n;x+=lowbit(x))

sum[x]+=v;

}

ll query(ll sum[],int x){

ll res=0;

for(;x;x-=lowbit(x))

res+=sum[x];

return res;

}

ll get\_sum(int x){

return (ll)(x+1)\*query(sum\_1,x)-query(sum\_2,x);

}

int main(){

while(scanf("%d%d",&n,&q)==2){

memset(sum\_1,0,sizeof(sum\_1));

memset(sum\_2,0,sizeof(sum\_2));

for(int i=1;i<=n;i++)

scanf("%lld",&a[i]);

for(int i=n;i>=2;i--){

a[i]=a[i]-a[i-1];

insert(sum\_1,i,a[i]);

insert(sum\_2,i,i\*a[i]);

}

insert(sum\_1,1,a[1]);

insert(sum\_2,1,1\*a[1]);

while(q--){

scanf(" %c",&ch);

if(ch=='Q'){

scanf("%d%d",&l,&r);

printf("%lld\n",get\_sum(r)-get\_sum(l-1));

}

else{

scanf("%lld%lld%lld",&l,&r,&v);

insert(sum\_1,l,v),insert(sum\_1,r+1,-v);

insert(sum\_2,l,l\*v),insert(sum\_2,r+1,-(r+1)\*v);

}

}

}

}

**树状数组 2D 区间修改区间查询，利用差分技巧，可以扩展到异或**

int sum[4][maxn][maxn],v,n,m,l0,r0,l1,r1,c;

char ch;

void insert(int s[][maxn],int x,int y,int v){

for(;x<=n;x+=lowbit(x))

for(int t=y;t<=m;t+=lowbit(t))

s[x][t]+=v;

}

int query(int s[][maxn],int x,int y){

ll res=0;

for(;x;x-=lowbit(x))

for(int t=y;t;t-=lowbit(t))

res+=s[x][t];

return res;

}

int get\_sum(int x,int y){

ll res0=query(sum[0],x,y)\*(x+1)\*(y+1);

ll res1=query(sum[1],x,y)\*(x+1);

ll res2=query(sum[2],x,y)\*(y+1);

ll res3=query(sum[3],x,y);

//cout<<res0<<' '<<res1<<' '<<res2<<' '<<res3<<endl;

return res0-res1-res2+res3;

}

int main(){

scanf("%c%d%d",&ch,&n,&m);

while(scanf(" %c",&ch)==1){

if(ch=='L'){

scanf("%d%d%d%d%d",&l0,&r0,&l1,&r1,&c);

insert(sum[0],l0,r0,c);insert(sum[0],l1+1,r0,-c);insert(sum[0],l0,r1+1,-c);insert(sum[0],l1+1,r1+1,c);

insert(sum[1],l0,r0,c\*r0);insert(sum[1],l1+1,r0,-c\*r0);insert(sum[1],l0,r1+1,-c\*(r1+1));insert(sum[1],l1+1,r1+1,c\*(r1+1));

insert(sum[2],l0,r0,c\*l0);insert(sum[2],l1+1,r0,-c\*(l1+1));insert(sum[2],l0,r1+1,-c\*l0);insert(sum[2],l1+1,r1+1,c\*(l1+1));

insert(sum[3],l0,r0,c\*l0\*r0);insert(sum[3],l1+1,r0,-c\*(l1+1)\*r0);insert(sum[3],l0,r1+1,-c\*l0\*(r1+1));insert(sum[3],l1+1,r1+1,c\*(l1+1)\*(r1+1));

}

if(ch=='k'){

scanf("%d%d%d%d",&l0,&r0,&l1,&r1);

//cout<<get\_sum(l1,r1)<<' '<<get\_sum(l0-1,r1)<<' '<<get\_sum(l1,r0-1)<<' '<<get\_sum(l0-1,r0-1)<<endl;

printf("%d\n",get\_sum(l1,r1)-get\_sum(l0-1,r1)-get\_sum(l1,r0-1)+get\_sum(l0-1,r0-1));

}

}

}

**笛卡尔树（根值最大）**

tree[root=0].val=0x7fffffff;

void insert(int x){

int p=x-1;//因为x-1一定在最右的路径的末端(没有右儿子)

while(tree[p].val<tree[x].val)p=tree[p].f;

if(!p)root=x;

tree[x].l=tree[p].r,tree[tree[p].r].f=x;

tree[x].f=p,tree[p].r=x;

}

for(int i=1;i<=n;i++)

insert(i);

**Splay 平衡树**

struct splay{

int l, r, f, sz, num[2], gcd[2];

void init(){

l = r = f = sz = num[0] = num[1] = gcd[0] = gcd[1] = 0;

}

}tree[MAXN];

int root, sz;

void debug(){

cout << "Root : " << root << endl;

for(int i = 0; i <= sz; i ++){

cout << i << ' ' << tree[i].l << ' ' << tree[i].r << ' ' << tree[i].sz << ' ' << tree[i].num[0] << ' ' << tree[i].num[1] << ' ' << tree[i].gcd[0] << ' ' << tree[i].gcd[1] << endl;

}

}

int gcd(int x, int y){

return !y ? x : gcd(y, x % y);

}

void update(int x){

tree[x].gcd[0] = gcd(gcd(tree[x].num[0], tree[tree[x].l].gcd[0]), tree[tree[x].r].gcd[0]);

tree[x].gcd[1] = gcd(gcd(tree[x].num[1], tree[tree[x].l].gcd[1]), tree[tree[x].r].gcd[1]);

tree[x].sz = 1 + tree[tree[x].l].sz + tree[tree[x].r].sz;

}

void zig(int x){

int y = tree[x].f, z = tree[y].f, A = tree[x].r;

if(tree[z].l == y) tree[z].l=x;

else tree[z].r = x;

tree[x].f = z, tree[y].f = x;

tree[y].l = A, tree[x].r = y;

if(A) tree[A].f = y;

update(y);

}

void zag(int x){

int y = tree[x].f, z = tree[y].f, A = tree[x].l;

if(tree[z].l == y) tree[z].l = x;

else tree[z].r = x;

tree[x].f = z, tree[y].f = x;

tree[y].r = A, tree[x].l = y;

if(A) tree[A].f = y;

update(y);

}

void splay(int x){

int y = tree[x].f, z = tree[y].f;

while(y){

if(!z){

if(x == tree[y].l) zig(x);

else zag(x);

break;

}

if(x == tree[y].l && y == tree[z].l) zig(y), zig(x);

else if(x == tree[y].l && y == tree[z].r) zig(x), zag(x);

else if(x == tree[y].r && y == tree[z].r) zag(y), zag(x);

else zag(x), zig(x);

y = tree[x].f, z = tree[y].f;

}

update(x);

root = x;

}

int find(int M){

int x = root;

while(1){

if(tree[tree[x].l].sz == M - 1) break;

if(tree[tree[x].l].sz < M){

M -= (tree[tree[x].l].sz + 1), x = tree[x].r;

}

else x = tree[x].l;

}

return x;

}

void insert(int x, int num, int sta){

x = find(x), splay(x);

sz ++;

tree[sz].init();

tree[sz].sz = 1, tree[sz].gcd[sta] = tree[sz].num[sta] = num;

if(tree[x].r == 0){

tree[sz].f = x, tree[x].r = sz;

}

else{

x = tree[x].r;

while(tree[x].l) x = tree[x].l;

tree[sz].f = x, tree[x].l = sz;

}

splay(sz);

}

void erase(int x){

x = find(x), splay(x);

if(!tree[x].l){

tree[tree[x].r].f = 0;

root = tree[x].r;

}

else{

int y = tree[x].l;

tree[y].f = 0;

while(tree[y].r) y = tree[y].r;

splay(y);

tree[y].r = tree[x].r;

if(tree[x].r) tree[tree[x].r].f = y;

update(y);

}

tree[x].init();

}

void gao(int x, int num){

x = find(x), splay(x);

if(num == -1){

tree[x].gcd[0] = gcd(gcd(tree[x].num[1], tree[tree[x].l].gcd[0]), tree[tree[x].r].gcd[0]);

tree[x].gcd[1] = gcd(gcd(tree[x].num[0], tree[tree[x].l].gcd[1]), tree[tree[x].r].gcd[1]);

swap(tree[x].num[1], tree[x].num[0]);

}

else{

int sta;

if(tree[x].num[0] != 0) sta = 0; else sta = 1;

tree[x].num[sta] = num;

tree[x].gcd[sta] = gcd(gcd(tree[x].num[sta], tree[tree[x].l].gcd[sta]), tree[tree[x].r].gcd[sta]);

}

}

int query(int x0, int x1, int sta){

x0 = find(x0), splay(x0);

tree[tree[x0].r].f = 0;

x1 = find(x1), splay(x1);

root = x0;

tree[x1].f = x0, tree[x0].r = x1;

update(x0);

return tree[tree[x1].l].gcd[sta];

}

int n, num[MAXN], sta[MAXN];

int build(int l, int r, int f){

if(l > r) return 0;

int x = l + r >> 1;

tree[x].init();

if(f == 0) root = x;

tree[x].f = f;

if(x >= 2 && x <= n + 1)

tree[x].num[sta[x - 1]] = tree[x].gcd[sta[x - 1]] = num[x - 1];

tree[x].l = build(l, x - 1, x);

tree[x].r = build(x + 1, r, x);

update(x);

return x;

}

void init(){

for(int i = 1; i <= n; i ++)

scanf("%d%d", &num[i], &sta[i]);

build(1, n + 2, 0);

sz = n + 2;

//debug();

}

**Deque 模拟块状链表**

**两个操作：把一段区间循环右移；求一段区间值为 k 的数量（k <= 1e5）**

deque <int> Q[355];

int s[355][N];

int n, q, x, l, r, op, ans;

void gao(int &y){

y = (y + ans - 1) % n + 1;

}

int main(){

// freopen("in.txt", "r", stdin);

cin >> n;

int Limit = sqrt(n + 0.5);

for(int i = 0; i < n; i ++){

scanf("%d", &x);

Q[i / Limit].push\_back(x);

s[i / Limit][x] ++;

}

cin >> q;

while(q --){

scanf("%d", &op);

if(op == 1){

scanf("%d%d", &l, &r);

gao(l), gao(r);

l --, r --;

if(l > r) swap(l, r);

int ll = l / Limit, rr = r / Limit;

if(ll == rr){

int t = Q[rr][r % Limit];

Q[rr].erase(Q[rr].begin() + (r % Limit));

Q[ll].insert(Q[ll].begin() + (l % Limit), t);

} else{

for(int i = ll; i < rr; i ++){

int t = Q[i].back(); Q[i].pop\_back();

Q[i + 1].push\_front(t);

s[i][t] --, s[i + 1][t] ++;

}

int t = Q[rr][1 + r % Limit];

Q[rr].erase(Q[rr].begin() + (1 + r % Limit)); s[rr][t] --;

Q[ll].insert(Q[ll].begin() + (l % Limit), t); s[ll][t] ++;

}

} else{

scanf("%d%d%d", &l, &r, &x);

gao(l), gao(r), gao(x); ans = 0;

l --, r --;

if(l > r) swap(l, r);

int ll = l / Limit, rr = r / Limit;

if(ll == rr){

for(int i = l % Limit, len = r % Limit; i <= len; i ++){

ans += (Q[ll][i] == x);

}

} else{

for(int i = ll + 1; i < rr; i ++) ans += s[i][x];

for(int i = l % Limit; i < Limit; i ++) ans += (Q[ll][i] == x);

for(int i = 0, len = r % Limit; i <= len; i ++) ans += (Q[rr][i] == x);

}

printf("%d\n", ans);

}

}

}

**莫队算法（Acdream 1108，求一段区间所有数的频数的第 k 大）**

int limit, l[MAXN], r[MAXN];

bool cmp(const int &i, const int &j){

int k1 = l[i] / limit, k2 = l[j] / limit;

if(k1 != k2) return k1 < k2;

return r[i] < r[j];

}

int t, n, m, a[MAXN], c[MAXN], id[MAXN], fre[MAXN], cnt[MAXN], ans[MAXN];

int calc(int x){

int ll = 1, rr = MAXN;

while(ll < rr){

int mid = ll + rr + 1 >> 1;

if(fre[mid] < x) rr = mid - 1;

else ll = mid;

}

return ll;

}

int main(){

//freopen("in.txt", "r", stdin);

cin >> t;

while(t --){

memset(fre, 0, sizeof(fre));

memset(cnt, 0, sizeof(cnt));

scanf("%d%d", &n, &m);

limit = sqrt(n + 0.5);

for(int i = 1; i <= n; i ++) scanf("%d", &a[i]);

for(int i = 1; i <= m; i ++) scanf("%d%d%d", &l[i], &r[i], &c[i]), id[i] = i;

sort(id + 1, id + 1 + m, cmp);

int ll = 1, rr = 0;

for(int i = 1; i <= m; i ++){

int k = id[i];

while(rr < r[k]) fre[++ cnt[a[++ rr]]] ++;

while(rr > r[k]) fre[cnt[a[rr --]] --] --;

while(ll < l[k]) fre[cnt[a[ll ++]] --] --;

while(ll > l[k]) fre[++ cnt[a[-- ll]]] ++;

ans[k] = calc(c[k]);

}

for(int i = 1; i <= m; i ++)

printf("%d\n", ans[i]);

}

}

**可持续化线段树（HDU4866 多校第一场Shooting，平面上给定区间，询问一个 x 轴上可以打到的前 K 个区间距离和。把区间变成端点 +1，-1；离散化区间后在线点查询）**

struct seg{

int lson, rson, num; LL sum;

} tree[N \* 60]; // 既然不是线段树的写法, 就不用开 \*4 的空间了 ..

int T[N], tot;

int init\_tree(int l, int r){

int root = ++ tot;

tree[root].sum = tree[root].num = 0;

if(l != r){

int mid = l + r >> 1;

tree[root].lson = init\_tree(l, mid);

tree[root].rson = init\_tree(mid + 1, r);

}

return root;

}

int update(int root, int pos, int nn, int l, int r){

int newroot = ++ tot;

tree[newroot].num = tree[root].num + (nn > 0 ? 1 : -1);

tree[newroot].sum = tree[root].sum + nn;

if(l != r){

int mid = l + r >> 1;

if(pos <= mid){

tree[newroot].lson = update(tree[root].lson, pos, nn, l, mid);

tree[newroot].rson = tree[root].rson;

}

else{

tree[newroot].lson = tree[root].lson;

tree[newroot].rson = update(tree[root].rson, pos, nn, mid + 1, r);

}

}

return newroot;

}

vector <int> lsh;

LL query(int root, int k, int l, int r){

if(l == r){

return (LL) k \* lsh[l - 1];

}

int mid = l + r >> 1;

if(k <= tree[tree[root].lson].num){

return query(tree[root].lson, k, l, mid);

}

else{

LL ll = tree[tree[root].lson].sum;

return ll + query(tree[root].rson, k - tree[tree[root].lson].num, mid + 1, r);

}

}

vector <PII> vec;

void init(){

tot = 0;

lsh.clear(), vec.clear();

}

int n, m, x, P, a, b, c, K, l[N], r[N], d[N];

int main(){

while(scanf("%d%d%d%d", &n, &m, &x, &P) == 4){

init();

for(int i = 1; i <= n; i ++) scanf("%d%d%d", &l[i], &r[i], &d[i]), lsh.PB(d[i]), r[i] ++;

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

for(int i = 1; i <= n; i ++){

d[i] = lower\_bound(lsh.begin(), lsh.end(), d[i]) - lsh.begin() + 1;

vec.PB(MP(l[i], d[i])), vec.PB(MP(r[i], -d[i]));

}

int sz = lsh.size();

sort(vec.begin(), vec.end());

T[0] = init\_tree(1, sz);

for(int i = 1; i <= vec.size(); i ++){

int v = lsh[abs(vec[i - 1].second) - 1], pos = vec[i - 1].second;

if(pos < 0) v = -v;

pos = abs(pos);

T[i] = update(T[i - 1], pos, v, 1, sz);

}

LL pre = 1;

for(int i = 1; i <= m; i ++){

scanf("%d%d%d%d", &x, &a, &b, &c);

K = ((LL) a \* pre + b) % c;

int rr = upper\_bound(vec.begin(), vec.end(), MP(x + 1, -INF)) - vec.begin();

LL ans = query(T[rr], K, 1, sz);

if(pre > P) ans \*= 2;

printf("%I64d\n", ans);

pre = ans;

}

}}

**KD树（Acdream1197 在三维空间内一个立方体内的点数；用 KD 树暴力插入查询，最坏复杂度为平方级的，不过对付随机数据很好用）**

#define M 50001

struct node{

int x[5];

int l, r, p;

}kd[M];

const int D = 3;

int n, splitD, A[5], B[5];

void insert(int u, int a, int d)

{

d %= D;

if (kd[a].x[d] < kd[u].x[d])

{

if (kd[u].l == -1) kd[u].l = a, kd[a].p = u;

else insert(kd[u].l, a, d+1);

}

else

{

if (kd[u].r == -1) kd[u].r = a, kd[a].p = u;

else insert(kd[u].r, a, d+1);

}

}

int sum;

void search(int u, int d)

{

if (u == -1) return ;

int i;

for (i = 0; i < D; i++)

if (kd[u].x[i] < A[i] || kd[u].x[i] > B[i])

break;

if (i == D) ++sum;

d %= D;

if (kd[u].x[d] >= A[d])

search(kd[u].l, d+1);

if (kd[u].x[d] <= B[d])

search(kd[u].r, d+1);

}

int main()

{

int n, m, root, i, k, cc = 0;

while (~scanf("%d", &n))

{

root = -1;

for (k = 0; k < n; k++)

{

kd[k].l = kd[k].r = kd[k].p = -1;

for (i = 0; i < D; i++)

scanf("%d", kd[k].x+i);

if (root == -1) root = 0;

else insert(root, k, 0);

}

scanf("%d", &m);

printf("Case #%d:\n", ++cc);

while (m--)

{

for (i = 0; i < D; i++)

scanf("%d", A+i);

for (i = 0; i < D; i++)

{

scanf("%d", B+i);

if (A[i] > B[i])

{

swap(A[i], B[i]);

}

}

sum = 0;

search(root, 0);

printf("%d\n", sum);

}

}

return 0;

}

**KD树，求平面内距离一个点的最近 m 个点**

int tt,n,k,Q,m,son[maxn<<2];

struct pnt{int d[6];}a[maxn],tar,tree[maxn<<2];

bool operator <(const pnt &t1,const pnt &t2){return t1.d[tt]<t2.d[tt];}

priority\_queue< pair<ll,pnt> >q;

ll sqr(ll x){return x\*x;}

void init\_tree(int l,int r,int dep,int x){

if(l>r)return;

int mid=(l+r)>>1;

tt=dep%k;

//按照tt=dep%k第tt维来排序

son[x]=true;

nth\_element(a+l,a+mid,a+r+1);

//第mid大的元素放在mid处,左边的比他小,右边的比他大

tree[x]=a[mid];

init\_tree(l,mid-1,dep+1,lc(x));

init\_tree(mid+1,r,dep+1,rc(x));

}

void query(int dep,int x){

if(!son[x])return;

int lx=lc(x),rx=rc(x),flag=0,cur=dep%k;

//这里的dep%k不能再用tt存了,不然会出错..why??

pair<ll,pnt>res=make\_pair(0,tree[x]);

for(int i=0;i<k;i++)

res.first+=sqr(tar.d[i]-tree[x].d[i]);

if(tar.d[cur]>=tree[x].d[cur])swap(lx,rx);

//把lx换为target所在的那一边

query(dep+1,lx);

//这里应该先递归左子树再判断这个点,可以减少大量时间..因为减少了递归右子树的可能性..

if(q.size()<m)q.push(res);

else if(q.top().first>res.first)q.pop(),q.push(res);

//这里进行一下判断,是KD树优秀复杂度的保证

if(sqr(tree[x].d[cur]-tar.d[cur])<q.top().first)flag=true;

if(flag)query(dep+1,rx);

}

int main(){

while(scanf("%d%d",&n,&k)==2){

for(int i=1;i<=n;i++)

for(int j=0;j<k;j++)

scanf("%d",&a[i].d[j]);

memset(son,0,sizeof(son));

init\_tree(1,n,0,1);

scanf("%d",&Q);

while(Q--){

for(int i=0;i<k;i++)

scanf("%d",&tar.d[i]);

scanf("%d",&m);

query(0,1);

pnt ans[21];

int cnt=0;

while(!q.empty())ans[++cnt]=q.top().second,q.pop();

printf("the closest %d points are:\n",m);

for(int i=m;i>=1;i--)

for(int j=0;j<k;j++)

printf("%d%s",ans[i].d[j],j==k-1?"\n":" ");

}

}

}

**CDQ 分治（Acdream 1157，三个操作，插入删除线段，查询完全包含某个线段的个数；将插入删除查询一起排序，按时间戳分治）**

int sum[2 \* N];

void Add(int x, int v){

x = 2 \* N - x;

for(; x < 2 \* N; x += lowbit(x))

sum[x] += v;

}

int Query(int x){

int ret = 0;

x = 2 \* N - x;

for(; x; x -= lowbit(x))

ret += sum[x];

return ret;

}

struct Node{

int op, id, l, r;

Node(){}

Node(int \_op, int \_id, int \_l, int \_r){

op = \_op, id = \_id, l = \_l, r = \_r;

}

} Q[N];

bool cmp\_l(const Node &A, const Node &B){

if(A.l != B.l) return A.l < B.l;

return A.r > B.r;

}

bool cmp\_id(const Node &A, const Node &B){

return A.id < B.id;

}

int ans[N], n;

void gao(int l, int r){

if(l >= r) return;

int mid = l + r >> 1;

gao(l, mid);

sort(Q + l, Q + 1 + r, cmp\_l);

for(int i = l; i <= r; i ++){

if(Q[i].id <= mid){

if(Q[i].op == 1) Add(Q[i].r, 1);

else if(Q[i].op == -1) Add(Q[i].r, -1);

} else{

if(Q[i].op == 2) ans[Q[i].id] += Query(Q[i].r);

}

}

for(int i = l; i <= r; i ++){

if(Q[i].id <= mid){

if(Q[i].op == 1) Add(Q[i].r, -1);

else if(Q[i].op == -1) Add(Q[i].r, 1);

}

}

sort(Q + l, Q + 1 + r, cmp\_id);

gao(mid + 1, r);

}

char op; int l, r, id, \_l[N], \_r[N], qcnt;

vector <int> lsh;

int main(){

//freopen("in.txt", "r", stdin);

while(scanf("%d", &n) == 1){

qcnt = 0;

lsh.clear();

memset(ans, 0, sizeof(ans));

for(int i = 1; i <= n; i ++){

scanf(" %c", &op);

if(op == 'D'){

scanf("%d%d", &l, &r);

Q[i] = Node(1, i, l, r);

\_l[++ qcnt] = l, \_r[qcnt] = r;

lsh.PB(l), lsh.PB(r);

} else if(op == 'Q'){

scanf("%d%d", &l, &r);

lsh.PB(l), lsh.PB(r);

Q[i] = Node(2, i, l, r);

} else{

scanf("%d", &id);

Q[i] = Node(-1, i, \_l[id], \_r[id]);

}

}

sort(lsh.begin(), lsh.end());

lsh.erase(unique(lsh.begin(), lsh.end()), lsh.end());

for(int i = 1; i <= n; i ++){

Q[i].l = lower\_bound(lsh.begin(), lsh.end(), Q[i].l) - lsh.begin() + 1;

Q[i].r = lower\_bound(lsh.begin(), lsh.end(), Q[i].r) - lsh.begin() + 1;

//cout << i << ' ' << Q[i].op << ' ' << Q[i].l << ' ' << Q[i].r << endl;

}

gao(1, n);

for(int i = 1; i <= n; i ++)

if(Q[i].op == 2)

printf("%d\n", ans[i]);

}

}

**树链剖分（离线 DFS 序，对颜色建立线段树）求每个点最多被染上的颜色**

struct Seg{

int l, r, mx, pos;

int mid(){

return l + r >> 1;

}

} tree[N << 2];

void Init\_tree(int l, int r, int x){

tree[x].l = l, tree[x].r = r, tree[x].mx = 0;

if(l == r){

tree[x].pos = l;

return;

}

int mid = l + r >> 1;

Init\_tree(l, mid, lc(x));

Init\_tree(mid + 1, r, rc(x));

}

void Insert(int pos, int num, int x){

if(tree[x].l == tree[x].r){

tree[x].mx += num;

return;

}

int mid = tree[x].mid();

if(pos <= mid) Insert(pos, num, lc(x));

else Insert(pos, num, rc(x));

if(tree[lc(x)].mx >= tree[rc(x)].mx) tree[x].mx = tree[lc(x)].mx, tree[x].pos = tree[lc(x)].pos;

else tree[x].mx = tree[rc(x)].mx, tree[x].pos = tree[rc(x)].pos;

}

struct edge{

int v, next;

} e[N << 1];

int first[N], ecnt, fa[N], seq[N], id[N], head[N], sz[N], son[N], dep[N], tot;

vector <PII> vec[N];

void Addedge(int u, int v){

e[++ ecnt].v = v, e[ecnt].next = first[u], first[u] = ecnt;

}

void Dfs(int u, int pre){

fa[u] = pre; sz[u] = 1;

dep[u] = dep[pre] + 1;

son[u] = 0;

for(int i = first[u]; i != -1; i = e[i].next){

int v = e[i].v;

if(v == pre) continue;

Dfs(v, u);

if(sz[v] > sz[son[u]])

son[u] = v;

sz[u] += sz[v];

}

}

void Build(int u, int st){

head[u] = st;

seq[++ tot] = u;

id[u] = tot;

if(son[u] != 0) Build(son[u], st);

for(int i = first[u]; i != -1; i = e[i].next){

int v = e[i].v;

if(v == fa[u] || v == son[u]) continue;

Build(v, v);

}

}

void Gao(int u, int v, int w){

int fu = head[u], fv = head[v];

while(fu != fv){

if(dep[fu] < dep[fv])

swap(fu, fv), swap(u, v);

vec[fu].PB(MP(w, 1));

if(id[u] < tot) vec[seq[id[u] + 1]].PB(MP(w, -1));

u = fa[fu], fu = head[u];

}

if(dep[u] < dep[v])

swap(u, v);

vec[v].PB(MP(w, 1));

if(id[u] < tot) vec[seq[id[u] + 1]].PB(MP(w, -1));

}

int T, u, v, w, n, m, ans[N];

int main(){

// freopen("in.txt", "r", stdin);

while(scanf("%d%d", &n, &m) == 2){

if(!n && !m) break;

memset(first, -1, sizeof(first));

ecnt = -1, tot = 0;

for(int i = 1; i <= n; i ++)

vec[i].clear();

for(int i = 1; i < n; i ++){

scanf("%d%d", &u, &v);

Addedge(u, v), Addedge(v, u);

}

Dfs(1, 0);

Build(1, 1);

// for(int i = 1; i <= n; i ++)

// printf("id[%d] = %d, son[%d] = %d, sz[%d] = %d, head[%d] = %d\n", i, id[i], i, son[i], i, sz[i], i, head[i]);

Init\_tree(1, N, 1);

for(int i = 1; i <= m; i ++){

scanf("%d%d%d", &u, &v, &w);

Gao(u, v, w);

}

for(int i = 1; i <= n; i ++){

u = seq[i];

for(int j = 0; j < vec[u].size(); j ++)

Insert(vec[u][j].first, vec[u][j].second, 1);

ans[u] = (tree[1].mx == 0 ? 0 : tree[1].pos);

}

for(int i = 1; i <= n; i ++)

printf("%d\n", ans[i]);

}

}

**正常的树链剖分（对树上的链建立线段树）上海网赛卖东西的题目**

const int N = 5e4 + 7;

const int INF = 2e9 + 7;

const int MOD = 1e9 + 7;

const DB EPS = 1e-8;

inline int max(int x, int y){

return x > y ? x : y;

}

inline int min(int x, int y){

return x < y ? x : y;

}

struct Seg{

int l, r, mx, mn, mk, res[2];

int mid(){

return l + r >> 1;

}

} tree[N << 2];

int seq[N], p[N];

inline void Pushup(int x){

tree[x].mx = max(tree[lc(x)].mx, tree[rc(x)].mx);

tree[x].mn = min(tree[lc(x)].mn, tree[rc(x)].mn);

tree[x].res[0] = max(max(tree[lc(x)].res[0], tree[rc(x)].res[0]), tree[lc(x)].mx - tree[rc(x)].mn);

tree[x].res[1] = max(max(tree[lc(x)].res[1], tree[rc(x)].res[1]), tree[rc(x)].mx - tree[lc(x)].mn);

}

inline void Init\_tree(int l, int r, int x){

tree[x].l = l, tree[x].r = r;

tree[x].mk = 0;

if(l == r){

tree[x].mx = tree[x].mn = p[seq[l]];

tree[x].res[0] = tree[x].res[1] = -1;

return;

}

int mid = l + r >> 1;

Init\_tree(l, mid, lc(x));

Init\_tree(mid + 1, r, rc(x));

Pushup(x);

}

inline void Pushdown(int x){

if(tree[x].mk){

int t = tree[x].mk;

tree[lc(x)].mx += t, tree[rc(x)].mx += t;

tree[lc(x)].mn += t, tree[rc(x)].mn += t;

tree[lc(x)].mk += t, tree[rc(x)].mk += t;

tree[x].mk = 0;

}

}

int Query(int l, int r, int &mx, int &mn, int op, int x){

if(l <= tree[x].l && tree[x].r <= r){

mx = tree[x].mx;

mn = tree[x].mn;

return tree[x].res[op];

}

Pushdown(x);

int mid = tree[x].mid(), ret = -1;

if(r <= mid) ret = Query(l, r, mx, mn, op, lc(x));

else if(l > mid) ret = Query(l, r, mx, mn, op, rc(x));

else{

int mxl, mxr, mnl, mnr;

ret = max(ret, Query(l, r, mxl, mnl, op, lc(x)));

ret = max(ret, Query(l, r, mxr, mnr, op, rc(x)));

if(op == 0) ret = max(ret, mxl - mnr);

else ret = max(ret, mxr - mnl);

mx = max(mxl, mxr);

mn = min(mnl, mnr);

}

return ret;

}

void Update(int l, int r, int w, int x){

if(l <= tree[x].l && tree[x].r <= r){

tree[x].mx += w, tree[x].mn += w;

tree[x].mk += w;

return;

}

Pushdown(x);

int mid = tree[x].mid();

if(l <= mid) Update(l, r, w, lc(x));

if(r > mid) Update(l, r, w, rc(x));

Pushup(x);

}

struct edge{

int u, v, next;

} e[N << 1];

int first[N], ecnt, fa[N], id[N], head[N], sz[N], son[N], dep[N], tot;

void Addedge(int u, int v){

e[++ ecnt].v = v, e[ecnt].next = first[u], first[u] = ecnt;

}

void Dfs(int u, int pre){

fa[u] = pre; sz[u] = 1;

dep[u] = dep[pre] + 1;

son[u] = 0;

for(int i = first[u]; i != -1; i = e[i].next){

int v = e[i].v;

if(v == pre) continue;

Dfs(v, u);

if(sz[v] > sz[son[u]])

son[u] = v;

sz[u] += sz[v];

}

}

void Build(int u, int st){

head[u] = st;

seq[++ tot] = u;

id[u] = tot;

if(son[u] != 0) Build(son[u], st);

for(int i = first[u]; i != -1; i = e[i].next){

int v = e[i].v;

if(v == fa[u] || v == son[u]) continue;

Build(v, v);

}

}

int Calc(int u, int v){

int fu = head[u], fv = head[v], ret = -1;

int mxv = -1, mnu = INF, tmx, tmn;

while(fu != fv){

if(dep[fu] > dep[fv]){

int res = Query(id[fu], id[u], tmx, tmn, 0, 1);

ret = max(ret, tmx - mnu);

ret = max(ret, res);

mnu = min(mnu, tmn);

u = fa[fu], fu = head[u];

} else{

int res = Query(id[fv], id[v], tmx, tmn, 1, 1);

ret = max(ret, mxv - tmn);

ret = max(ret, res);

mxv = max(mxv, tmx);

v = fa[fv], fv = head[v];

}

}

if(dep[u] > dep[v]){

int res = Query(id[v], id[u], tmx, tmn, 0, 1);

ret = max(ret, res);

} else{

int res = Query(id[u], id[v], tmx, tmn, 1, 1);

ret = max(ret, res);

}

ret = max(ret, tmx - mnu);

ret = max(ret, mxv - tmn);

ret = max(ret, mxv - mnu);

return ret < 0 ? 0 : ret;

}

void Gao(int u, int v, int w){

int fu = head[u], fv = head[v];

while(fu != fv){

if(dep[fu] < dep[fv])

swap(fu, fv), swap(u, v);

Update(id[fu], id[u], w, 1);

u = fa[fu], fu = head[u];

}

if(dep[u] < dep[v])

swap(u, v);

Update(id[v], id[u], w, 1);

}

int T, u, v, w, n, m;

int main(){

// freopen("in.txt", "r", stdin);

cin >> T;

while(T --){

scanf("%d", &n);

for(int i = 1; i <= n; i ++)

scanf("%d", &p[i]);

memset(first, -1, sizeof(first));

ecnt = -1, tot = 0;

for(int i = 1; i < n; i ++){

scanf("%d%d", &u, &v);

Addedge(u, v), Addedge(v, u);

}

Dfs(1, 0);

Build(1, 1);

// for(int i = 1; i <= n; i ++)

// printf("fa[%d] = %d, id[%d] = %d, son[%d] = %d, sz[%d] = %d\n", i, fa[i], i, id[i], i, son[i], i, sz[i]);

Init\_tree(1, n, 1);

scanf("%d", &m);

for(int i = 1; i <= m; i ++){

scanf("%d%d%d", &u, &v, &w);

printf("%d\n", Calc(u, v));

// return 0;

Gao(u, v, w);

}

}

}

**线段树套 Splay（用得上就见鬼了）**

//maintain(北京集训队)

#include <iostream>

#include <math.h>

#include <stdio.h>

#define maxn 1000005

#define lc(x) (x<<1)

#define rc(x) (lc(x)+1)

using namespace std;

struct seg\_tree{int l,r;int mid(){return l+r>>1;}}seg[4\*maxn];

struct splay{int l,r,f,size,key;}tree[4\*maxn];

int n,m,color[maxn],prev[maxn],temp[maxn],root[4\*maxn],tree\_size;

/\*void pushdown(int x){

//平衡树的标记下传和线段树一样,在给一个节点表上标记后同时对这个节点进行值的修改

//理论上说每次操作都要pushdown的,但如果每次操作之前都先进行提根的splay操作的话就不用刻意pushdown了..因此pushdown可以只在splay和update函数中用到

if(tree[x].rev){

if(tree[x].l){

tree[tree[x].l].rev ^= 1;

swap(tree[tree[x].l].l,tree[tree[x].l].r);

swap(tree[tree[x].l].lmax,tree[tree[x].l].rmax);

}

if(tree[x].r){

tree[tree[x].r].rev ^= 1;

swap(tree[tree[x].r].l,tree[tree[x].r].r);

swap(tree[tree[x].r].lmax,tree[tree[x].r].rmax);

}

tree[x].rev = 0;

}

if(tree[x].mark){

if(tree[x].l){

tree[tree[x].l].key = tree[x].key, tree[tree[x].l].mark = true;

tree[tree[x].l].sum = tree[tree[x].l].size \* tree[tree[x].l].key;

tree[tree[x].l].max = tree[tree[x].l].lmax = tree[tree[x].l].rmax = tree[tree[x].l].key < 0 ? tree[tree[x].l].key: tree[tree[x].l].sum;

}

if(tree[x].r){

tree[tree[x].r].key = tree[x].key, tree[tree[x].r].mark = true;

tree[tree[x].r].sum = tree[tree[x].r].size \* tree[tree[x].r].key;

tree[tree[x].r].max = tree[tree[x].r].lmax = tree[tree[x].r].rmax = tree[tree[x].r].key < 0 ? tree[tree[x].r].key: tree[tree[x].r].sum;

}

tree[x].mark=0;

}

}\*/

void update(int x){

pushdown(x);

tree[x].size=tree[tree[x].l].size+tree[tree[x].r].size+1;

}

void zig(int x){

int y=tree[x].f,z=tree[y].f,A=tree[x].r;

pushdown(y),pushdown(x);

if(tree[z].l==y)tree[z].l=x;

else tree[z].r=x;

tree[x].f=z,tree[y].f=x;

tree[y].l=A,tree[x].r=y;

if(A)tree[A].f=y;

update(y);

}

void zag(int x){

int y=tree[x].f,z=tree[y].f,A=tree[x].l;

pushdown(y),pushdown(x);

if(tree[z].l==y)tree[z].l=x;

else tree[z].r=x;

tree[x].f=z,tree[y].f=x;

tree[y].r=A,tree[x].l=y;

if(A)tree[A].f=y;

update(y);

}

void splay(int x){

int y=tree[x].f,z=tree[y].f;

//这里不用root=x,每次进行splay后最后再改变root..

while(y){

pushdown(x);

if(!z){

if(x==tree[y].l)zig(x);

else zag(x);

break;

}

if(x==tree[y].l&&y==tree[z].l)

zig(y),zig(x);

else if(x==tree[y].l&&y==tree[z].r)

zig(x),zag(x);

else if(x==tree[y].r&&y==tree[z].r)

zag(y),zag(x);

else

zag(x),zig(x);

y=tree[x].f,z=tree[y].f;

}update(x);

}

void insert(int &x,int key,int father){

//lgn建立节点的正确姿势

//如果是利用splay对一条链的性质的话则这个insert完全用不上..

if(!x){

tree\_size++;

tree[tree\_size].f=father,tree[tree\_size].key=key,tree[tree\_size].size=1;

x=tree\_size;

return;

}

if(key<=tree[x].key)insert(tree[x].l,key,x);

else insert(tree[x].r,key,x);

update(x);

}

int find(int x,int key){

//找小于key的节点个数(小于等于key-1的节点个数)

int temp=0;key--;

while(x){

if(key>=tree[x].key)temp+=tree[tree[x].l].size+1,x=tree[x].r;

else x=tree[x].l;

}

return temp;

}

int Find(int x,int M){

//找排名M的数

while(1){

if(tree[tree[x].l].size == M-1)break;

if(tree[tree[x].l].size < M)

x = tree[x].r , M -= (tree[tree[x].l].size+1);//这里要注意多减一个1

else

x = tree[x].l;

}

return x;

}

void erase(int &X,int key){

//删除一个节点值为key的节点

int x=X;

while(1){

if(key==tree[x].key)break;

if(key<tree[x].key)x=tree[x].l;

else x=tree[x].r;

}

splay(x);

if(!tree[x].l){

tree[tree[x].r].f=0;

X=tree[x].r;return;

}

int p=tree[x].l;tree[p].f=0;

while(tree[p].r)p=tree[p].r;

splay(p);

//把它左子树的最后节点提上来

tree[p].r=tree[x].r;

if(tree[x].r)tree[tree[x].r].f=p;

update(p);

X=p;

}

void buildtree(int l,int r,int x){

seg[x].l=l,seg[x].r=r;

//splay之后改变root

for(int i=l;i<=r;i++)

insert(root[x],prev[i],0),splay(tree\_size),root[x]=tree\_size;

if(l==r)return;

int mid=seg[x].mid();

buildtree(l,mid,lc(x));

buildtree(mid+1,r,rc(x));

}

int query(int l,int r,int x){

if(l<=seg[x].l&&seg[x].r<=r)

return find(root[x],l);

int mid=seg[x].mid();

if(r<=mid)return query(l,r,lc(x));

else if(l>mid)return query(l,r,rc(x));

else return query(l,r,lc(x))+query(l,r,rc(x));

}

void change(int pre,int now,int pos,int x){

erase(root[x],pre);

insert(root[x],now,0);

if(seg[x].l==seg[x].r)return;

int mid=seg[x].mid();

if(pos<=mid)change(pre,now,pos,lc(x));

else change(pre,now,pos,rc(x));

}

void cchange(int x,int col){

int tmp;

prev[x]=0;

for(int i=1;i<=n;i++){

if(prev[i]==x){

change(x,prev[x],i,1);

prev[i]=prev[prev[x]];

}

if(i<x&&color[i]==col)prev[x]=i;

if(i>x&&prev[i]<x&&color[i]==col){

change(prev[i],x,i,1);

prev[i]=x;

}

}

color[x]=col;

}

int main(){

scanf("%d%d",&n,&m);

for(int i=1;i<=n;i++)

scanf("%d",&color[i]);

for(int i=1;i<=n;i++)

prev[i]=temp[color[i]],temp[color[i]]=i;

buildtree(1,n,1);

char A;int x,y;

while(m--){

scanf(" %c%d%d",&A,&x,&y);

if(A=='Q')printf("%d\n",query(x,y,1));

else{

int p=prev[x];

cchange(x,y),change(p,prev[x],x,1);

}

}

return 0;

}

**动态规划 ——**

**数位 DP （Acdream 1166；求在 [li, ri]，i <= 4 的范围内， x0 ^ .. x3 = 0 的 count 和 sum；将进位当成状态）**

PII f[35][2][2][2][2]; int dig[4][35];

bool vis[35][2][2][2][2];

PII dp(int pos, bool f0, bool f1, bool f2, bool f3){

if(pos == 0) return MP(1, 0);

if(vis[pos][f0][f1][f2][f3] != 0)

return f[pos][f0][f1][f2][f3];

vis[pos][f0][f1][f2][f3] = 1;

int res1 = 0, res2 = 0;

for(int i0 = 0, r0 = f0 ? dig[0][pos] : 1; i0 <= r0; i0 ++)

for(int i1 = 0, r1 = f1 ? dig[1][pos] : 1; i1 <= r1; i1 ++)

for(int i2 = 0, r2 = f2 ? dig[2][pos] : 1; i2 <= r2; i2 ++){

int i3 = i0 ^ i1 ^ i2;

if(i3 && f3 && dig[3][pos] == 0) continue;

bool t0 = (f0 && (i0 == r0)), t1 = (f1 && (i1 == r1)), t2 = (f2 && (i2 == r2));

bool t3 = (f3 && (i3 == dig[3][pos]));

PII tmp = dp(pos - 1, t0, t1, t2, t3);

res1 += tmp.first; if(res1 >= MOD) res1 -= MOD;

res2 += tmp.second; if(res2 >= MOD) res2 -= MOD;

int s = i0 + i1 + i2 + i3;

res2 += ((LL) tmp.first \* (1 << (pos - 1)) \* s) % MOD;

if(res2 >= MOD) res2 -= MOD;

}

return f[pos][f0][f1][f2][f3] = MP(res1, res2);

}

int x[4];

PII calc(){

memset(dig, 0, sizeof(dig));

for(int i = 0; i < 4; i ++){

int cnt = 0;

while(x[i]) dig[i][++ cnt] = x[i] % 2, x[i] /= 2;

}

return dp(30, 1, 1, 1, 1);

}

int ans1, ans2, a[4][2];

int main(){

while(scanf("%d%d%d%d%d%d%d%d", &a[0][0], &a[0][1], &a[1][0], &a[1][1], &a[2][0], &a[2][1], &a[3][0], &a[3][1]) == 8){

for(int i = 0; i < 4; i ++) a[i][0] --;

ans1 = ans2 = 0;

for(int i = 0; i < (1 << 4); i ++){

int cnt = 0;

for(int j = 0; j < 4; j ++){

if(i >> j & 1) x[j] = a[j][0], cnt ++;

else x[j] = a[j][1];

}

memset(vis, 0, sizeof(vis));

PII tmp = calc();

if(cnt % 2 == 0) ans1 = (ans1 + tmp.first) % MOD, ans2 = (ans2 + tmp.second) % MOD;

else ans1 = (ans1 - tmp.first + MOD) % MOD, ans2 = (ans2 - tmp.second + MOD) % MOD;

}

printf("%d %d\n", ans1, ans2);

}

}

**数位DP（求 n 以内的数满足 d0-d1+d2 .. = 0 的数有多少个；d0表示最高位，以此类推；因为最高位不能放零，需要记录起始位的是否存在）**

int dp(int pos, int num, int sta, int flag, bool first){

if(pos == 0) return 1;

if(flag == 0 && f[pos][num][sta][first] != -1) return f[pos][num][sta][first];

int res = 0;

for(int i = 0, r = flag ? dig[pos] : 9; i <= r; i ++){

if(i == 0 && first == true) res = res + dp(pos - 1, 9, 0, 0, 1);

else if(sta == 0 && i <= num) res = res + dp(pos - 1, i, sta ^ 1, flag && (i == r), 0);

else if(sta == 1 && i >= num) res = res + dp(pos - 1, i, sta ^ 1, flag && (i == r), 0);

}

return flag ? res : f[pos][num][sta][first] = res;

}

**数位DP（上海网赛 B 题，一共 m 个数每个限制为 0 <= ai <= Ri 求在 p 进制下求和不发生进位的方案数；每次DP放一个数的一个数位会很好写）**

int dp(int pos, int num, int sta, int flag, bool first){

if(pos == 0) return 1;

if(flag == 0 && f[pos][num][sta][first] != -1) return f[pos][num][sta][first];

int res = 0;

for(int i = 0, r = flag ? dig[pos] : 9; i <= r; i ++){

if(i == 0 && first == true) res = res + dp(pos - 1, 9, 0, 0, 1);

else if(sta == 0 && i <= num) res = res + dp(pos - 1, i, sta ^ 1, flag && (i == r), 0);

else if(sta == 1 && i >= num) res = res + dp(pos - 1, i, sta ^ 1, flag && (i == r), 0);

}

return flag ? res : f[pos][num][sta][first] = res;

}

**二维 Sparse-Table（一维显然同理）**

int \_log[N], f[N][N][9][9], n, m, Q, lx, ly, rx, ry;

int main(){

// freopen("in.txt", "r", stdin);

for(int i = 2; i < N; i ++)

\_log[i] = \_log[i / 2] + 1;

while(scanf("%d%d", &n, &m) == 2){

for(int i = 1; i <= n; i ++)

for(int j = 1; j <= m; j ++)

scanf("%d", &f[i][j][0][0]);

for(int i1 = 0; i1 <= \_log[n]; i1 ++)

for(int i2 = 0; i2 <= \_log[m]; i2 ++)

for(int j1 = 1; j1 + (1 << i1) - 1 <= n; j1 ++)

for(int j2 = 1; j2 + (1 << i2) - 1 <= m; j2 ++){

if(i1) f[j1][j2][i1][i2] = max(f[j1][j2][i1 - 1][i2], f[j1 + (1 << (i1 - 1))][j2][i1 - 1][i2]);

if(i2) f[j1][j2][i1][i2] = max(f[j1][j2][i1][i2 - 1], f[j1][j2 + (1 << (i2 - 1))][i1][i2 - 1]);

// cout << j1 << ' ' << j2 << ' ' << i1 << ' ' << i2 << ' ' << f[j1][j2][i1][i2] << endl;

}

scanf("%d", &Q);

while(Q --){

scanf("%d%d%d%d", &lx, &ly, &rx, &ry);

int tx = \_log[rx - lx + 1];

int ty = \_log[ry - ly + 1];

int a = f[lx][ly][tx][ty];

int b = f[lx][ry - (1 << ty) + 1][tx][ty];

int c = f[rx - (1 << tx) + 1][ly][tx][ty];

int d = f[rx - (1 << tx) + 1][ry - (1 << ty) + 1][tx][ty];

// cout << a << ' ' << b << ' ' << c << ' ' << d << endl;

int mx = max(a, max(b, max(c, d)));

printf("%d ", mx);

if(mx == f[lx][ly][0][0] ||

mx == f[lx][ry][0][0] ||

mx == f[rx][ly][0][0] ||

mx == f[rx][ry][0][0]

) puts("yes"); else puts("no");

}

}

}

**斜率优化（玩具装箱）**

**使用的条件：较优的决策点对后续状态影响具有持续性，简单来说就是如果在状态I处，选择了J作为决策点，那在I+1处的决策点一定不可能在J之前；然后把DP方程写出来，左边化为J，K的形式，右边化为I的形式。**

ll a[maxn],stk[maxn],dp[maxn],f[maxn],sum[maxn],c;

ll G(int x,int y){return dp[x]-dp[y]+(f[x]+c)\*(f[x]+c)-(f[y]+c)\*(f[y]+c);}

ll S(int x,int y){return 2\*(f[x]-f[y]);}

double X(int x,int y){return double(double(G(x,y))/double(S(x,y)));}

int main(){

int n,head=0,tail=0;

scanf("%d%lld",&n,&c);

c++;

for(int i=1;i<=n;i++){

scanf("%d",&a[i]);

sum[i]=sum[i-1]+a[i];

f[i]=sum[i]+i;

while(head<tail && X(stk[head+1],stk[head])<=f[i])

head++;

dp[i]=dp[stk[head]]+(f[i]-f[stk[head]]-c)\*(f[i]-f[stk[head]]-c);

while(tail>head && !(X(stk[tail],stk[tail-1])<X(i,stk[tail])))

tail--;

stk[++tail]=i;

}

printf("%lld\n",dp[n]);

//system("pause");

return 0;

}

**最长上升子序列 (NlgN)**

dp[1]=1,lis[1]=aa[1],cnt=1;

for(int i=2;i<=n;i++){

int t=lower\_bound(lis+1,lis+cnt+1,aa[i])-lis;

if(t>cnt)lis[++cnt]=aa[i],dp[i]=cnt;

else lis[t]=aa[i],dp[i]=t;

}

**数学——矩阵**

**浮点数高斯消元（一定有解）**

void Gauss(int n, int m){

int i, j, k;

for(i = 0, j = 0; i < n && j < m; j ++){

k = i;

for(int r = i + 1; r < n; r ++)

if(fabs(mat[k][j]) < fabs(mat[r][j]))

k = r;

//找到一个大一点的精度好控制一些..但是这种写法是基于一定有解的

if(k < n){

if(i != k) for(int t = 0; t <= m; t ++) swap(mat[i][t], mat[k][t]);

//如果这里是从0开始枚举的话很容易就精度溢出了..因为每次都得消一遍..

for(int r = i + 1; r < n; r ++)

if(fabs(mat[r][j]) > EPS){//这里一定要用eps来判断..

double tt = mat[i][j] / mat[r][j];

//注意这里tt的顺序如果是mat[r][j]/mat[i][j]的话就很可能精度不够..不过是为啥..?..

for(int t = m; t >= 0; t --)

mat[r][t] -= mat[i][t] / tt;

}

i ++;

}

}

for(int r = m - 1; r >= 0; r --){

if(mat[r][r] == 0.0) continue;//有必要..因为某些点本身不会被涉及到..

double tmp = 0.0;

for(int t = r + 1; t < m; t ++)

tmp += mat[r][t] \* ans[t];

ans[r] = (mat[r][m] - tmp) / mat[r][r];

}

}

**同余高斯消元（一定有解；如果 P 很大，代回过程需要解模方程）**

void gauss(int n,int m){

//n和m为系数矩阵的大小

int i,j,k;

for(i=0,j=0;i<n&&j<m;j++){

for(k=i;k<n;k++)if(a[k][j])break;

if(k<n){

for(int r=0;r<=m;r++)swap(a[i][r],a[k][r]);

//这里从第i+1行开始消就可以了..只要保证是行阶梯就可以

for(int r=i+1;r<n;r++)

if(a[r][j]){

int b1=a[i][j],b2=a[r][j];

for(int t=0;t<=m;t++)

a[r][t]=((b1\*a[r][t]-b2\*a[i][t])%7+7)%7;

}

i++;

}

}//判断无解一定要在最后..

for(int r=i;r<n;r++)if(a[r][m]!=0){

printf("Inconsistent data.\n");

return;

}

if(i<m){

printf("Multiple solutions.\n");

return;

}

//从m-1开始是因为如果一定有唯一解的话一定是一个严格的倒三角形式..

for(int r=m-1;r>=0;r--){

int tmp=0;

for(int t=r+1;t<m;t++)

tmp+=a[r][t]\*ans[t],tmp%=7;

while((a[r][m]-tmp)%a[r][r])a[r][m]+=7;

//这里能够这么做的原因是7是一个质数..因此保证了这么加一定可以在6步之内加出一个解来..

ans[r]=(a[r][m]-tmp)/a[r][r],ans[r]=(ans[r]%7+7)%7;

}

for(int i=0;i<m;i++){

if(ans[i]<3)ans[i]+=7;

printf("%d%s",ans[i],i==m-1?"\n":" ");

}

}

**枚举自由变元（当然前提是有解，此题仅为二进制）**

void dfs(int r,int co,int tot){

if(tot>ANS)return;

if(co==-1){

ANS=min(ANS,tot);return;

}

int first;

for(int i=n-1;i>=0;i--)if(a[r][i])first=i;

if(co==first){

ans[first]=a[r][n];

for(int j=first+1;j<n;j++)

ans[first]^=(ans[j]\*a[r][j]);

dfs(r-1,first-1,tot+ans[first]);

}

else{

ans[co]=1;dfs(r,co-1,tot+1);

ans[co]=0;dfs(r,co-1,tot);

}

}

void gauss(int n,int m){

//n和m为系数矩阵的大小

int i,j,k;

for(i=0,j=0;i<n&&j<m;j++){

k=i;

for(;k<n;k++)if(a[k][j])break;

if(k<n){

for(int r=0;r<=m;r++)swap(a[i][r],a[k][r]);

//这里从第i+1行开始消就可以了..只要保证是行阶梯就可以

for(int r=i+1;r<n;r++)

if(a[r][j])

for(int t=0;t<=m;t++)

a[r][t]^=a[i][t];

i++;

}

}//判断无解一定要在最后..

for(int r=i;r<n;r++)if(a[r][m]!=0){

printf("impossible\n");

return;

}

dfs(i-1,m-1,0);

printf("%d\n",ANS);

}

**二进制高斯消元求基（CF 472F，将 A[] 通过异或变成 B[]，要求不用多余的槽位）**

void Debug(int a[], int n){

for(int i = 1; i <= n; i ++){

for(int j = 30; j >= 0; j --)

printf("%d", a[i] >> j & 1);

puts("");

}

puts("");

}

void Gao(int a[], int n, vector <PII> &vec){

int r = 1;

for(int i = 30; i >= 0; i --){

if((a[r] >> i & 1) == 0){

int pos = -1;

for(int j = r; j <= n; j ++)

if(a[j] >> i & 1){

pos = j; break;

}

if(pos == -1) continue;

swap(a[r], a[pos]);

vec.PB(MP(r, pos));

vec.PB(MP(pos, r));

vec.PB(MP(r, pos));

}

for(int j = 1; j <= n; j ++)

if((a[j] >> i & 1) && j != r){

a[j] ^= a[r];

vec.PB(MP(j, r));

}

pos[r] = i;

r ++;

}

}

int Lcp(int a, int b){

int tot = 0;

for(int i = 30; i >= 0; i --){

if((a >> i & 1) != (b >> i & 1))

break;

tot ++;

}

return tot;

}

bool Check(int a[], int b[], int n, vector <PII> &vec){

int r = 1;

for(int i = 1; i <= 30; i ++){

if(a[i] == b[i]) continue;

for(int j = i; j <= 30; j ++){

if(Lcp(a[i] ^ a[j], b[i]) > Lcp(a[i], b[i]))

a[i] ^= a[j], vec.PB(MP(i, j));

}

if(a[i] != b[i]) return false;

}

return true;

}

int n, a[N], b[N];

vector <PII> vec1, vec2;

int main(){

cin >> n;

for(int i = 1; i <= n; i ++) scanf("%d", &a[i]);

for(int i = 1; i <= n; i ++) scanf("%d", &b[i]);

vector <PII> vec1, vec2;

Gao(a, n, vec1);

Gao(b, n, vec2);

// Debug(a, n), Debug(b, n);

if(!Check(a, b, n, vec1)){

puts("-1");

} else{

for(int i = vec2.size() - 1; i >= 0; i --)

vec1.PB(vec2[i]);

printf("%d\n", vec1.size());

for(int i = 0; i < vec1.size(); i ++)

printf("%d %d\n", vec1[i].first, vec1[i].second);

}

}

**矩阵乘法**

int size;

struct Matrix{

LL base[88][88];

void init(){

memset(base, 0, sizeof(base));

}

void init2(){

init();

for(int i = 0; i < size; i ++) base[i][i]=1;

}

void debug(){

for(int i = 0; i < size; i ++){

for(int j = 0; j < size; j ++)

cout << base[i][j] << ' ';

cout << endl;

}

cout << endl;

}

}start, trans, temp, ans;

void copy(Matrix &A, Matrix B){

for(int i = 0; i < size; i ++)

for(int j = 0; j < size; j ++)

A.base[i][j] = B.base[i][j];

}

void go(Matrix &A, Matrix B){

Matrix temp;

temp.init();

for(int i = 0; i < size; i ++)

for(int j = 0; j < size; j ++)

for(int k = 0; k < size; k ++)

temp.base[i][j] += A.base[i][k] \* B.base[k][j],

temp.base[i][j] %= mod;

copy(A, temp);

}

void Mul(LL x){

while(x){

if(x & 1) go(start, trans);

go(trans, trans);

x >>= 1;

}

}

**矩阵的逆**

Matrix Gauss\_Inverse(Matrix A){

Matrix B; B.init2();

for(int i = 0; i < size; i ++) {

if(A.base[i][i] == 0){

int cur;

for(int j = 0; j < size; j ++)

if(A.base[j][i] != 0){

cur = j; break;

}

for(int j = 0; j < size; j ++){

swap(A.base[i][j], A.base[cur][j]);

swap(B.base[i][j], B.base[cur][j]);

}

}

// A.debug(); B.debug();

int t = A.base[i][i];

for(int j = 0; j < size; j ++)

A.base[i][j] /= t, B.base[i][j] /= t;

for(int j = 0; j < size; j ++)

if(j != i) {

t = A.base[j][i];

for(int k = 0; k < size; k ++){

A.base[j][k] -= t \* A.base[i][k];

B.base[j][k] -= t \* B.base[i][k];

}

}

}

return B;

}

**行列式求解（复杂度 O(n^3lgn)；不懂原理，只有板子）**

ll calc\_det(ll a[201][201],int n){

ll ret=1;

for(int i=0;i<n;i++){

for(int j=i+1;j<n;j++)

while(a[j][i]){

ll t=a[i][i]/a[j][i];

for(int k=i;k<n;k++)

a[i][k]=(a[i][k]-a[j][k]\*t)%mod;

for(int k=i;k<n;k++)

swap(a[i][k],a[j][k]);

ret=-ret;

}

if(a[i][i]==0)return 0;

ret=ret\*a[i][i],ret%=mod;

}

ret=((ret%mod)+mod)%mod;

return ret;

}

**辛普森积分**

double F(double x){

//被积函数

return sqrt(1+4\*a\*a\*x\*x);

}

double simpson(double l,double r){

double m=(r+l)/2.0;

return (F(l)+F(r)+4\*F(m))\*(r-l)/6.0;

}

double asr(double l,double r,double epsilon,double A){

double m=(r+l)/2.0;

double L=simpson(l,m),R=simpson(m,r);

if(fabs(L+R-A)<15\*epsilon)return L+R+(L+R-A)/15.0;

return asr(l,m,epsilon/2.0,L)+asr(m,r,epsilon/2.0,R);

}

double asr(double l,double r,double epsilon){

return asr(l,r,epsilon,simpson(l,r));

}

double calc(double w,double h){

a=4.0\*h/w/w;

//上边界,下边界,精度控制

return asr(0,w/2,eps)\*2;

}

**数学——数论**

**线性筛(N) + 快速幂(lgN) + 扩展欧几里得(lgN) + 逆元(lgN) + 分解质因数(NsqrtN)**

void prime\_table(){

memset(is\_prime,1,sizeof(is\_prime));

for(int i=2;i<maxn;i++){

if(is\_prime[i])

prime[++cnt]=i;

for(int j=1;j<=cnt && i\*prime[j]<maxn;j++){

is\_prime[i\*prime[j]]=false;

if(i%prime[j]==0)break;

}

}

}//线性筛

void prime(ll x){

for(int i=2;i\*i<=x;i++)if(x%i==0){

p[++cnt][1]=i;

while(x%i==0)p[cnt][2]++,x/=i;

}

if(x>1)p[++cnt][1]=x,p[cnt][2]=1;

}//分解质因数

ll pow\_mod(ll x,ll y){

ll res=1;

while(y){

if(y&1)res\*=x,res%=mod;

x\*=x,x%=mod,y>>=1;

}

return res%mod;

}//快速幂

void ext\_gcd(ll a,ll b,ll &d,ll &x,ll &y){

if(!b){d=a,x=1,y=0;}

else{ext\_gcd(b,a%b,d,y,x),y-=x\*(a/b);}

}//扩展欧几里得

ll inv(ll a,ll c){

ll d,x,y;

ext\_gcd(a,c,d,x,y);

return (x+c)%c;

}//逆元

void phi\_table(){

phi[1]=1;

for(int i=2;i<maxn;i++){

if(!phi[i])phi[i]=i-1,prime[++cnt]=i;

for(int j=1;j<=cnt&&i\*prime[j]<maxn;j++){

if(i%prime[j]==0){phi[i\*prime[j]]=phi[i]\*prime[j];break;}

else phi[i\*prime[j]]=phi[i]\*(prime[j]-1);

}

}

}//欧拉函数线性筛

void mu\_table(){

mu[1] = 1;

for(int i = 0; i < N; i ++) is\_prime[i] = true;

for(int i = 2 ; i < N; i ++){

if(is\_prime[i]) prime.PB(i), mu[i] = -1;

for(int j = 0; j < prime.size() && i \* prime[j] < N; j ++){

is\_prime[i \* prime[j]] = false;

if(i % prime[j] == 0) break;

mu[i \* prime[j]] = -mu[i];

}

}

for(int i = 1; i < N; i ++) if(mu[i] != 0)

for(int j = i; j < N; j += i)

vec[j].PB(i);

}//莫比乌斯函数线性筛

**离散对数，BSGS（A^X = B (mod P)，P 可以不为质数，时间复杂度 O(hash\*sqrtN) ；P为质数的短版本见白书）**

int gcd(int x,int y){return !y?x:gcd(y,x%y);}

struct edge{int next,v,id;}e[mod];

int first[mod],cnt;

void add\_hash(int u,int num,int id){

e[++cnt].next=first[u],first[u]=cnt,e[cnt].v=num,e[cnt].id=id;

}

int \_hash(ll x){

int res=inf;

for(int i=first[x%mod];i!=-1;i=e[i].next)

if(e[i].v==x)res=min(res,e[i].id);

return res==inf?0:res;

}

void ext\_gcd(ll a,ll b,ll &d,ll &x,ll &y){

if(!b){d=a,x=1,y=0;}

else{ext\_gcd(b,a%b,d,y,x),y-=x\*(a/b);}

}

ll inv(ll a,ll c){

ll d,x,y;

ext\_gcd(a,c,d,x,y);

return (x+c)%c;

}

ll pow\_mod(ll x,ll y,ll mm){

ll res=1;

while(y){

if(y&1)res\*=x,res%=mm;

x\*=x,x%=mm,y>>=1;

}

return res%mm;

}

ll baby\_step(ll a,ll b,ll c){//a^x = b (mod c)

ll tt=1,tmp,co=0,d=1;

memset(first,-1,sizeof(first)),cnt=-1;

for(int i=0;i<=100;tt=tt\*a%c,i++)if(tt==b)return i;

while((tmp=gcd(a,c))!=1){

if(b%tmp!=0)return -1;

co++,b/=tmp,c/=tmp,d=d\*a/tmp%c;

}

int m=(int)(sqrt(c+0.5));

tt=1;

for(int i=0;i<=m;i++)

add\_hash(tt%mod,tt,i),tt\*=a,tt%=c;

tt=pow\_mod(a,m,c);

for(int i=0;i<=m;i++){

if(tmp=\_hash(b\*inv(d,c)%c))return i\*m+tmp+co;

d=d\*tt%c;

}

return -1;

}

**卢卡斯定理 + 线性模方程 + 互质版中国剩余定理**

ll calc(ll x,ll y,ll c){

if(x<y)return 0;

ll res1=1,res2=1;

for(int i=1;i<=y;i++)

res1\*=(x-i+1),res1%=c,res2\*=i,res2%=c;

res2=inv(res2,c),res1\*=res2,res1%=c;

return res1;

}//C(x,y)%c,直接算

ll lucas(ll x,ll y,ll c){

ll res=1;

while(x&&y&&res)res\*=calc(x%c,y%c,c),res%=c,x/=c,y/=c;

return res;

}//C(x,y)%c,Lucas定理

ll equ(ll a,ll b,ll c){

// ax + cy = b

ll d,x,y;ext\_gcd(a,c,d,x,y);

if(b%d!=0)return -1;

x\*=(b/d),x%=(c/d);if(x<0)x+=(c/d);

return x;

}//求解模方程ax = b mod c, 最小满足要求的正整数x

ll China(int N,ll a[],ll m[]){

ll M=1,res=0;

for(int i=0;i<N;i++)M\*=m[i];

//cout<<M<<endl;

for(int i=0;i<N;i++){

ll w=M/m[i];

res+=w\*equ(w,a[i],m[i]),res%=(mod-1);

}

return res;

}//中国剩余定理 x mod m[i] = a[i]..要求m[i]之间互质

ll go(ll x){

// a^b % c = a^(b%phi(c)+phi(c)) % c (b>phi(c))

// 这里不用判断b>phi(c)是因为c是质数,因此一定有a^phi(c) = 1 (mod c)..因此也就无所谓了

// phi(c)不为质数,分解质因数后用中国剩余定理合并

ll a[4];

for(int i=0;i<4;i++)a[i]=lucas(n,x,fac[i]);

/\*for(int i=0;i<4;i++)cout<<a[i]<<endl;

cout<<endl;\*/

ll res=China(4,a,fac);

return res;//+mod-1;

}

**非互质版中国剩余定理(两两合并模方程)**

ll China(int N,ll a[],ll b[]){

ll d,x,y;

a[0]=1,b[0]=1;

for(int i=1;i<=N;i++){

ext\_gcd(a[i-1],a[i],d,x,y);

if((b[i]-b[i-1])%d!=0)return -1;

x\*=(b[i]-b[i-1])/d;

x%=(a[i]/d);if(x<0)x+=a[i]/d;

int lcm=a[i]\*a[i-1]/gcd(a[i],a[i-1]);

a[i]=lcm;

b[i]=(a[i-1]\*x+b[i-1])%lcm;

}

return b[N];

}//N mod a[i] = b[i]

**MillerRabin判素 + PollardRho找约数（模板题 Poj1181）**

ll mul\_mod(ll x,ll y,ll k){

x%=k,y%=k;

ll res=0;

while(y){

if(y&1){

res+=x;

if(res>=k)res-=k;

}

x+=x,y>>=1;

if(x>=k)x-=k;

}

return res;

}

ll pow\_mod(ll x,ll y,ll k){

ll res=1;

while(y){

if(y&1)res=mul\_mod(res,x,k);

x=mul\_mod(x,x,k),y>>=1;

}

return res;

}

ll witness(ll a,ll b,ll c){

if(b==0)return 1;

ll x,y,t=0;

while((b&1)==0)

b>>=1,t++;

y=x=pow\_mod(a,b,c);

while(t--){

y=mul\_mod(x,x,c);

if(y==1 && x!=1 && x!=c-1)

return false;

x=y;

}

return y==1;

}

bool miller\_rabin(ll n) {//..质数为true, 非质数为false..

if(n==2)return true;

if(n<2 || (n&1)==0)return false;

for(int i=0;i<3;i++)

if(witness(rand()%(n-2)+2,n-1,n)!=1)

return false;

return true;

}

ll pollard\_rho(ll n,ll c){//..随机返回一个 n 的约数..

if(n%2==0)return 2;

ll i=1,k=2,x=rand()%n,y=x,d;

while(1){

i++;

x=(mul\_mod(x,x,n)+c)%n;

d=gcd(y-x,n);

if(d==n)return n;

if(d!=n && d>1)return d;

if(i==k) y=x,k<<=1;

}

}

void calc(ll n,ll c=240){//寻找最小的约数..

if(n==1)return;

if(miller\_rabin(n)){

ans=min(ans,n);

return;

}

ll k=n;

while(k==n)k=pollard\_rho(n,c--);

calc(k,c),calc(n/k,c);

}

ll n;

int t;

int main(){

cin>>t;

while(t--){

scanf("%lld",&n);

if(miller\_rabin(n))printf("Prime\n");

else{

ans=inf;

calc(n);

printf("%lld\n",ans);

}

}

}

**求解二次剩余（求解 x^2 = d (mod P)，P 为奇质数；URAL 1132）**

LL pow\_mod(LL x, LL y, LL z) {

LL res = 1;

while(y) {

if(y & 1) res \*= x, res %= z;

x \*= x, x %= z, y >>= 1;

}

return res % z;

}

struct Complex{

LL x, y;

Complex(){}

Complex(LL \_x, LL \_y){

x = \_x, y = \_y;

}

};

LL w;

Complex Mul(Complex A, Complex B, LL p){

Complex ret;

ret.x = (A.x \* B.x % p + A.y \* B.y % p \* w % p) % p;

ret.y = (A.x \* B.y % p + A.y \* B.x % p) % p;

return ret;

}

Complex Pow\_mod(Complex x, LL y, LL p){

Complex ret(1, 0);

while(y){

if(y & 1) ret = Mul(ret, x, p);

x = Mul(x, x, p), y >>= 1;

}

return ret;

}

LL Legendre(LL x, LL y){ // 勒让德符号，仅在 d ^ ((p - 1) / 2) 之时 d 为 p 的二次剩余；

return pow\_mod(x, (y - 1) / 2, y);

}

LL gao(LL d, LL p){

if(p == 2) return 1;

if(Legendre(d, p) + 1 == p) return -1;

LL a;

while(1){

a = rand() % p;

w = ((a \* a - d) % p + p) % p;

// cout << a << ' ' << w << ' ' << Legendre(w, p) << endl;

if(Legendre(w, p) + 1 == p) break;

}

Complex ret = Pow\_mod(Complex(a, 1), (p + 1) / 2, p);

/\*

简单来说就是找到一个 a 使得 w = a^2 - d 为 p 的二次非剩余

（这里随机 a 就好，因为有二次非剩余有一半的概率出现）；

然后 (a + sqrt(w)) ^ ((p+1)/2) 就是一个解。。这里带根号的乘法类似于复数域的乘法。。

\*/

// cout << ret.x <<endl;

return ret.x;

}

int t; LL d, p;

int main(){

// freopen("in.txt", "r", stdin);

cin >> t;

while(t --){

scanf("%lld%lld", &d, &p);

d %= p;

if(p == 2){

if(d == 1) puts("1");

else puts("No root");

} else{

LL k = gao(d, p);

if(k == -1){

puts("No root");

} else{

if(k > p - k) k = p - k;

printf("%lld %lld\n", k, p - k);

}

}

}

}

**合并加法的技巧（Acdream1136，求树上相邻点对乘积 <=P 的方案数）**

int sum[N][2005], f[N][2005];

int num[2005], cnt;

// P = 5

// 1, 2, 3~5

// 合并加法的技巧中，其实每个数的区间一定有奇数个（2 除外，2 只有 2 个），因为 <sqrt(n) 的每个区间都只有一个数，而且能够一一对应到 >sqrt(n) 的区间中。

void calc(int u, int pre){

for(int i = 1; i <= cnt; i ++)

f[u][i] = num[i];

for(int i = first[u]; i != -1; i = e[i].next){

int v = e[i].v;

if(v == pre) continue;

calc(v, u);

for(int j = 1; j <= cnt; j ++){

f[u][j] = ((LL) f[u][j] \* sum[v][cnt - j + 1]) % MOD;

}

}

for(int i = 1; i <= cnt; i ++){

sum[u][i] = sum[u][i - 1] + f[u][i];

if(sum[u][i] >= MOD) sum[u][i] -= MOD;

}

}

int n, P, u, v;

int main(){

//freopen("in.txt", "r", stdin);

cin >> n >> P;

memset(first, -1, sizeof(first));

ecnt = -1; cnt = 0;

for(int i = 1; i < n; i ++){

scanf("%d%d", &u, &v);

addedge(u, v), addedge(v, u);

}

for(int i = 1; i <= P; i ++){

int next = P / (P / i);

num[++ cnt] = next - i + 1;

i = next;

}

calc(1, 0);

printf("%d\n", sum[1][cnt]);

}

**数学——组合 + FFT + 卡尔丹公式**

**Polya定理 (Hdu4633，有 n 种颜色，颜色可以无限用，正方体 8 个点，12 条边，54 个小面进行染色的方案数)**

int main(){

cin>>t;

for(int cas=1;cas<=t;cas++){

scanf("%d",&n);//n -> color number

ans=pow\_mod(n,74);

ans+=2\*3\*pow\_mod(n,2+3+15);//face,90,270

ans+=3\*pow\_mod(n,4+6+28);//face,180

ans+=6\*pow\_mod(n,4+7+27);//edge,180

ans+=2\*4\*pow\_mod(n,4+4+18);//pnt,120,240

ans%=mod;

ll p=inv(24,mod);

printf("Case %d: %I64d\n",cas,(ans\*p)%mod);

}

return 0;

}

**Burnside定理 （Acdream1022 给定 m 个颜色，颜色个数有限制，对正方体边长进行染色，求旋转不相同的方案数）**

int m, a[10], b[10];

LL ans, C[205][205], res;

void dfs(int sum, int x, LL num, int pos, int left){

if(sum > left) return;

if(pos == m + 1){

if(sum == 0) res += num, res %= MOD;

return;

}

for(int i = 0; i <= b[pos]; i ++) if(sum - i >= 0)

dfs(sum - i, x, num \* C[sum][i] % MOD, pos + 1, left - b[pos]);

}

LL calc(int x, bool flag){

res = 0;

int n = 12 / x, sum = 0;

if(flag) n = 10 / x;

for(int i = 1; i <= m; i ++) b[i] /= x, sum += b[i];

//cout << sum << endl;

if(sum < n) return 0;

dfs(n, x, 1, 1, sum);

return res;

}

void get\_face(){

memcpy(b, a, sizeof(b));

ans += 6 \* calc(4, 0), ans %= MOD;//90,270

memcpy(b, a, sizeof(b));

ans += 3 \* calc(2, 0), ans %= MOD;//180

}

void get\_edge(){

for(int i = 1; i <= m; i ++)

for(int j = 1; j <= m; j ++){

memcpy(b, a, sizeof(b));

b[i] --, b[j] --;

if(b[i] < 0 || b[j] < 0)continue;

ans += 6 \* calc(2, 1), ans %= MOD;

}

}

void get\_point(){

memcpy(b,a,sizeof(b));

ans += 8 \* calc(3, 0), ans %= MOD;

}

int t, cas;

int main(){

C[0][0] = 1;

for(int i = 1; i <= 200;i ++){

C[i][0] = 1;

for(int j = 1; j <= i; j ++)

C[i][j] = (C[i - 1][j - 1] + C[i - 1][j]) % MOD;

}

cin >> t;

while(t --){

scanf("%d", &m);

for(int i = 1; i <= m; i ++) scanf("%d", &a[i]);

memcpy(b, a, sizeof(b));

ans = 0;

ans += calc(1, 0), ans %= MOD;

get\_face();

get\_edge();

get\_point();

cout << "Case #" << ++ cas << ": ";

printf("%lld\n", ans \* inv(24, MOD) % MOD);

}

return 0;

}

**快速傅里叶变换 FFT (by cxlove)**

**题目一 —— 2013 多校，给定 1e5 个边长问组成三角形的方案数**

struct Complex{

double a,b;

Complex(){}

Complex(double \_a,double \_b):a(\_a),b(\_b){}

Complex operator + (const Complex &c) const {return Complex(a + c.a , b + c.b);}

Complex operator - (const Complex &c) const {return Complex(a - c.a , b - c.b);}

Complex operator \* (const Complex &c) const {return Complex(a \* c.a - b \* c.b , a \* c.b + b \* c.a);}

};

void change (Complex y[],int len) {

for (int i=1,j=len/2;i<len-1;i++) {

if (i<j)swap(y[i],y[j]);

int k=len/2;

while(j>=k)

j-=k,k/=2;

if(j<k)j+=k;

}

}

void fft(int len,int on,Complex y[]){

change (y,len);

for (int h=2;h<=len;h<<=1) {

Complex wn(cos(-on\*2\*pi/h),sin(-on\*2\*pi/h));

for (int j=0;j<len;j+=h){

Complex w(1,0);

for (int k=j;k<j+h/2;k++) {

Complex u=y[k];

Complex t=w\*y[k+h/2];

y[k]=u+t,y[k+h/2]=u-t,w=w\*wn;

}

}

}

if(on==-1)

for(int i=0;i<len;i++)

y[i].a=(y[i].a/len);

}

Complex a[maxn<<2];

ll ans[maxn<<2],len,s[maxn<<2],mx,t,n,x,pp[maxn<<2],sum[maxn<<2];

void init(){

memset(ans,0,sizeof(ans));

memset(s,0,sizeof(s));

memset(sum,0,sizeof(sum));

mx=0,len=1;

}

int main(){

cin>>t;

while(t--){

scanf("%d",&n);

init();

for(int i=1;i<=n;i++){

scanf("%d",&pp[i]);

s[pp[i]]++,mx=max(mx,pp[i]);

}

mx++;//注意fft中是从0到n-1进行的，所以mx要++

while(len<2\*mx)len<<=1;

for(int i=0;i<mx;i++)a[i]=Complex((double)s[i],0);

for(int i=mx;i<len;i++)a[i]=Complex(0,0);

fft(len,1,a);

for(int i=0;i<len;i++)a[i]=a[i]\*a[i];

fft(len,-1,a);

for(int i=0;i<len;i++)ans[i]=(ll)(a[i].a+0.5);

for(int i=1;i<=n;i++)

ans[2\*pp[i]]--;//自己加自己

for(int i=0;i<len;i++)

ans[i]/=2;//a+b,b+a重复计算

for(int i=1;i<len;i++)

sum[i]=sum[i-1]+ans[i];

ll cnt=0,tot=(ll)(n)\*(ll)(n-1)\*(ll)(n-2)/6;

for(int i=1;i<=n;i++){ // 枚举最长边，另外两条边的 sum 要大于该边

cnt+=(sum[len-1]-sum[pp[i]]);

cnt-=(n-1); // 减掉和自己相加

cnt-=(n-i)\*(n-i-1)/2; // 减掉大于自己的两条边相加

cnt-=(n-i)\*(i-1); // 减掉大于 + 小于

}

printf("%.7f\n",(double)cnt/tot);

}

return 0;

}

**题目二 —— CF472G，求两个 01 串之间的曼哈顿距离（还要分块）**

struct Complex{

DB a, b;

Complex(){}

Complex(DB \_a, DB \_b){

a = \_a, b = \_b;

}

Complex operator + (const Complex &c) const {return Complex(a + c.a , b + c.b);}

Complex operator - (const Complex &c) const {return Complex(a - c.a , b - c.b);}

Complex operator \* (const Complex &c) const {return Complex(a \* c.a - b \* c.b , a \* c.b + b \* c.a);}

};

void Change(Complex y[], int len) {

for(int i = 1 ,j = len / 2; i < len - 1; i ++){

if(i < j) swap(y[i], y[j]);

int k = len / 2;

while(j >= k)

j -= k, k /= 2;

if(j < k) j += k;

}

}

void FFT(int len, int on, Complex y[]){

Change(y, len);

for (int h = 2; h <= len; h <<= 1){

Complex wn(cos(-on \* 2 \* PI / h), sin(-on \*2 \* PI / h));

for (int j = 0; j < len; j += h){

Complex w(1, 0);

for (int k = j; k < j + h / 2; k ++){

Complex u = y[k];

Complex t = w \* y[k + h / 2];

y[k] = u + t, y[k + h / 2] = u - t, w = w \* wn;

}

}

}

if(on == -1)

for(int i = 0; i < len; i ++)

y[i].a = (y[i].a / len);

}

Complex A[N << 2], B[N << 2];

char ch1[N], ch2[N]; int a[N], b[N], Q[205][N];

int main(){

// freopen("in.txt", "r", stdin);

scanf(" %s", ch1); int n1 = strlen(ch1);

scanf(" %s", ch2); int n2 = strlen(ch2);

for(int i = 0; i < n1; i ++) a[i] = ch1[i] - '0' == 1 ? 1 : -1;

for(int i = 0; i < n2; i ++) b[i] = ch2[i] - '0' == 1 ? 1 : -1;

reverse(b, b + n2);

int sz = sqrt(n1 \* 16), tot = (n1 - 1) / sz;

int len = 1; while(len < n2 + sz) len <<= 1;

for(int i = 0; i < n2; i ++) B[i] = Complex(b[i], 0);

for(int i = n2; i < len; i ++) B[i] = Complex(0, 0);

reverse(b, b + n2);

FFT(len, 1, B);

for(int i = 0; i <= tot; i ++){

int st = sz \* i, cur = min(sz, n1 - st), ed = st + cur, t = 0;

for(int j = st; j < ed; j ++) A[t ++] = Complex(a[j], 0);

for(; t < len; t ++) A[t] = Complex(0, 0);

FFT(len, 1, A);

for(int j = 0; j < len; j ++) A[j] = A[j] \* B[j];

FFT(len, -1, A);

for(int j = 0; j <= n2 - cur; j ++) Q[i][j] = (cur - (floor)(A[n2 - 1 - j].a + 0.5)) / 2;

}

int l1, l2, r1, m;

cin >> m;

while(m --){

scanf("%d%d%d", &l1, &l2, &len);

r1 = l1 + len - 1;

int ll = l1 / sz, rr = r1 / sz;

int res = 0, t = 0;

if(ll == rr){

for(int i = l1; i <= r1; i ++) res += (a[i] != b[l2 + t]), t ++;

} else{

for(int i = l1; i < sz \* (ll + 1); i ++) res += (a[i] != b[l2 + t]), t ++;

for(int i = ll + 1; i < rr; i ++) res += Q[i][l2 + t], t += sz;

for(int i = sz \* rr; i <= r1; i ++) res += (a[i] != b[l2 + t]), t ++;

}

printf("%d\n", res);

}

}

**三次方程卡尔丹公式（by thu 保证有解时调用，全程 complex）**

void solve(ld a,ld b,ld c,ld d)

{

//printf("%f %f %f %f\n",a,b,c,d);

ld xx=(b\*c)/(6\*a\*a)-(b\*b\*b)/(27\*a\*a\*a)-(d)/(2\*a);

ld yy=(c)/(3\*a)-(b\*b)/(9\*a\*a);

complex<ld> delta=sqrt(complex<ld>(xx\*xx+yy\*yy\*yy));

complex<ld> aa=pow(xx+delta,1./3);

complex<ld> bb=pow(xx-delta,1./3);

complex<ld> cc=-b/(3\*a);

complex<ld> x1=cc+aa+bb;

complex<ld> w1=complex<ld>(-.5,sqrt(3)\*.5);

complex<ld> w2=complex<ld>(-.5,-sqrt(3)\*.5);

complex<ld> x2=cc+aa\*w1+bb\*w2;

complex<ld> x3=cc+aa\*w2+bb\*w1;

}

**几何——**

**头文件**

#pragma comment(linker, "/STACK:36777216")

#include <map>

#include <set>

#include <cmath>

#include <ctime>

#include <queue>

#include <stack>

#include <cstdio>

#include <string>

#include <vector>

#include <iomanip>

#include <cassert>

#include <cstdlib>

#include <cstring>

#include <iostream>

#include <algorithm>

#define lc(x) (x << 1)

#define rc(x) (lc(x) + 1)

#define lowbit(x) (x & (-x))

#define PI (acos(-1))

#define lowbit(x) (x & (-x))

#define LL long long

#define DB double

#define ULL unsigned long long

#define PII pair<int, int>

#define PLL pair<LL, LL>

#define PB push\_back

#define MP make\_pair

using namespace std;

/\*

=========================================================================================================

< Beginning of Geometry Template >

=========================================================================================================

\*/

const DB EPS = 1e-8;

const int INF = 0x3fffffff;

inline int sgn(DB x) {

return x < -EPS ? -1 : x < EPS ? 0 : 1;

}

struct Point {

DB x, y;

Point() {}

Point(DB \_x, DB \_y) {

x = \_x, y = \_y;

}

inline bool operator == (const Point &A) const {

return (sgn(x - A.x) == 0 && sgn(y - A.y) == 0);

}

inline bool operator < (const Point A) const {

if(sgn(x - A.x) != 0) return sgn(x - A.x) < 0;

return sgn(y - A.y) < 0;

}

inline Point operator + (const Point &A) const {

return Point(x + A.x, y + A.y);

}

inline Point operator - (const Point &A) const {

return Point(x - A.x, y - A.y);

}

inline Point operator \* (const DB &A) const {

return Point(x \* A, y \* A);

}

inline Point operator / (const DB &A) const {

return Point(x / A, y / A);

}

inline DB operator \* (const Point &A) const {

return x \* A.y - y \* A.x;

}

inline DB operator & (const Point &A) const {

return x \* A.x + y \* A.y;

}

inline void transXY(DB ceta) {

DB tx = x, ty = y;

x = tx \* cos(ceta) - ty \* sin(ceta);

y = tx \* sin(ceta) + ty \* cos(ceta);

}

};

inline DB Dist(Point A, Point B) {

return sqrt((A.x - B.x) \* (A.x - B.x) + (A.y - B.y) \* (A.y - B.y));

}

inline DB Angle(Point A, Point B) {

return atan2(B.y - A.y, B.x - A.x);

}

inline bool IsPointOnSegment(Point p, Point a1, Point a2){

return sgn((a1 - p) \* (a2 - p)) == 0 && sgn((a1 - p) & (a2 - p)) < 0;

}

inline int IsPointInPolygon(Point p, Point poly[], int n){

int wn = 0;

for(int i = 0; i < n; i ++){

if(IsPointOnSegment(p, poly[i], poly[(i + 1) % n])) return -1; // on the segment

int k = sgn((poly[(i + 1) % n] - poly[i]) \* (p - poly[i]));

int d1 = sgn(poly[i].y - p.y);

int d2 = sgn(poly[(i + 1) % n].y - p.y);

if(k > 0 && d1 <= 0 && d2 > 0) wn ++;

if(k < 0 && d2 <= 0 && d1 > 0) wn --;

}

if(wn != 0) return 1; // inside ..

return 0; // outside

}

int ConvexHull(vector <Point> p, Point \*ch){

sort(p.begin(), p.end());

int m = 0;

for(int i = 0; i < p.size(); i ++){

while(m > 1 && (ch[m - 1] - ch[m - 2]) \* (p[i] - ch[m - 2]) <= 0) m --; // Must be <= , so that there are no 3 points 1 line

ch[m ++] = p[i];

}

int k = m;

for(int i = p.size() - 2; i >= 0; i --){

while(m > k && (ch[m - 1] - ch[m - 2]) \* (p[i] - ch[m - 2]) <= 0) m --;

ch[m ++] = p[i];

}

if(p.size() > 1) m --;

return m;

}

struct Line {

Point st, ed, ve;

DB k;

Line() {}

Line(Point \_st, Point \_ed) {

st = \_st, ed = \_ed;

k = atan2(ed.y - st.y, ed.x - st.x);

ve = st - ed;

ve = ve / sqrt(ve.x \* ve.x + ve.y \* ve.y);

}

inline Point point(DB len) {

return st + (ve \* len);

}

};

inline Point GetLineProjection(Point P, Line L) {

return L.st + L.ve \* (L.ve & (P - L.st) / (L.ve & L.ve));

}

struct Circle {

Point c; DB r;

Circle() {}

Circle(Point \_c, DB \_r) {

c = \_c, r = \_r;

}

inline Point point(DB ceta) {

return Point(c.x + cos(ceta) \* r, c.y + sin(ceta) \* r);

}

};

// Intersection between circles .. return the state of intersecting ..

inline int GetCircleCircleIntersection(Circle C1, Circle C2, vector <Point> &sol) {

DB d = Dist(C1.c, C2.c);

if(sgn(d) == 0) {

if(sgn(C1.r - C2.r) == 0) return -1; // overlap ..

return 0;

}

if(sgn(C1.r + C2.r - d) < 0) return 0;

if(sgn(fabs(C1.r - C2.r) - d) > 0) return 0;

DB a = Angle(C1.c, C2.c);

DB da = acos((C1.r \* C1.r + d \* d - C2.r \* C2.r) / (2 \* C1.r \* d));

Point p1 = C1.point(a - da), p2 = C1.point(a + da);

if(p1 == p2) {

sol.PB(p1);

return 1;

} else {

sol.PB(p1), sol.PB(p2);

return 2;

}

}

// Intersection between line ans circle .. return the state of intersecting .. using analytic geometry

inline int GetLineCircleIntersection(Line L, Circle C, vector <Point> &sol) {

DB a = L.ve.x, b = L.st.x - C.c.x, c = L.ve.y, d = L.st.y - C.c.y;

DB e = a \* a + c \* c, f = 2 \* (a \* b + c \* d), g = b \* b + d \* d - C.r \* C.r;

DB delta = f \* f - 4 \* e \* g;

if(sgn(delta) < 0) return 0;

else if(sgn(delta) == 0) {

DB t1 = -f / (2 \* e);

sol.PB(L.point(t1));

return 1;

} else {

DB t1 = (-f - sqrt(delta)) / (2 \* e);

DB t2 = (-f + sqrt(delta)) / (2 \* e);

sol.PB(L.point(t1)), sol.PB(L.point(t2));

return 2;

}

}

/\*

=========================================================================================================

< End of Geometry Template >

\*/

**最近点对**

struct Point{

double x, y;

} p[MAXN];

bool cmpx(const Point &A, const Point &B){

if(A.x != B.x) return A.x < B.x;

return A.y < B.y;

}

bool cmpy(const Point &A, const Point &B){

return A.y < B.y;

}

double sqr(double x){

return x \* x;

}

double dist(Point A, Point B){

return sqrt(sqr(A.x - B.x) + sqr(A.y - B.y));

}

double ans;

vector <Point> vec;

void gao(int l, int r){

if(r - l <= 4){

for(int i = l; i < r; i ++) for(int j = i + 1; j <= r; j ++) ans = min(ans, dist(p[i], p[j]));

sort(p + l, p + r + 1, cmpy);

return;

}

int mid = l + r >> 1;

double midx = p[mid].x;

gao(l, mid), gao(mid + 1, r);

inplace\_merge(p + l, p + mid + 1, p + r, cmpy);

vec.clear();

for(int i = l; i <= r; i ++) if(fabs(midx - p[i].x) <= ans) vec.pb(p[i]);

for(int i = 0; i < vec.size(); i ++)

for(int j = i + 1; j < vec.size() && vec[j].y - vec[i].y <= ans; j ++)

ans = min(ans, dist(vec[i], vec[j]));

}

**给定 N 个圆求每个圆被几个圆完美包含（HDU 3511，利用 set 做花式扫描线）**

struct Circle{

int x, y, r;

} c[MAXN];

struct vecNode{

int x, mk, id;

vecNode(){};

vecNode(int a, int b, int c){

x = a, mk = b, id = c;

}

};

bool operator < (const vecNode &A, const vecNode &B){

if(A.x != B.x) return A.x < B.x;

return A.mk < B.mk;

}

struct SNode{

int ud, id;

SNode(){}

SNode(int a, int b){

ud = a, id = b;

}

};

int now;

DB sqr(DB x){

return x \* x;

}

bool operator < (const SNode &A, const SNode &B){

double Ay = (DB) c[A.id].y + A.ud \* sqrt(sqr((DB) c[A.id].r) - sqr((DB) now - c[A.id].x) + EPS);

double By = (DB) c[B.id].y + B.ud \* sqrt(sqr((DB) c[B.id].r) - sqr((DB) now - c[B.id].x) + EPS);

return Ay < By;

}

set <SNode> S;

set <SNode> :: iterator it, it2;

vector <vecNode> vec;

int f[MAXN], n;

int main(){

//freopen("in.txt", "r", stdin);

while(scanf("%d", &n) == 1){

vec.clear(), S.clear();

for(int i = 1; i <= n; i ++) f[i] = 0;

for(int i = 1; i <= n; i ++){

scanf("%d%d%d", &c[i].x, &c[i].y, &c[i].r);

vec.pb(vecNode(c[i].x + c[i].r, 0, i)), vec.pb(vecNode(c[i].x - c[i].r, 1, i));

// 0 means right, 1 means left ..

}

sort(vec.begin(), vec.end());

for(int i = 0; i < vec.size(); i ++){

now = vec[i].x;

if(vec[i].mk == 1){

it = S.lower\_bound(SNode(1, vec[i].id));

if(it != S.end() && it != S.begin()){

it2 = -- it; it ++;

int u = (\*it).id, v = (\*it2).id;

if(u == v) f[vec[i].id] = f[u] + 1;

else f[vec[i].id] = max(f[u], f[v]);

}

S.insert(SNode(1, vec[i].id)); // 1 means up ..

S.insert(SNode(-1, vec[i].id)); // -1 means down ..

}

else{

S.erase(SNode(1, vec[i].id));

S.erase(SNode(-1, vec[i].id));

}

}

int ans = 0;

for(int i = 1; i <= n; i ++) ans = max(ans, f[i]);

printf("%d\n", ans + 1);

}

}

**求最大被至少两个矩形完美覆盖的圆半径（ASC某题，做法为二分半径，缩小矩形后判断矩形是否有重合；具体做法为离散化矩形左右侧端点，将 set 中的线段元素的小于号重载成不相交）**

int n, x[N][2], y[N][2]; DB X, Y;

struct Node{

DB x, l, r; int t;

Node(){}

Node(DB \_x, int \_t, DB \_l, DB \_r){

x = \_x, t = \_t, l = \_l, r = \_r;

}

bool operator < (const Node &A) const{

if(fabs(x - A.x) < EPS)

return t > A.t;

return x < A.x;

}

};

struct Seg{

DB l, r;

Seg(){}

Seg(DB \_l, DB \_r){

l = \_l, r = \_r;

}

bool operator < (const Seg &A) const{

return r < A.l;

}

};

vector <Node> vec;

set <Seg> S;

bool check(DB r){

vec.clear();

for(int i = 1; i <= n; i ++){

if(min(x[i][1] - x[i][0], y[i][1] - y[i][0]) >= 2 \* r - EPS){

vec.PB(Node(x[i][0] + r, 1, y[i][0] + r, y[i][1] - r));

vec.PB(Node(x[i][1] - r, -1, y[i][0] + r, y[i][1] - r));

}

}

sort(vec.begin(), vec.end());

S.clear();

for(int i = 0; i < vec.size(); i ++){

if(vec[i].t == -1) S.erase(Seg(vec[i].l, vec[i].r));

else{

if(S.count(Seg(vec[i].l, vec[i].r))){

set <Seg> :: iterator it = S.find(Seg(vec[i].l, vec[i].r));

X = vec[i].x, Y = vec[i].l < (\*it).l ? (\*it).l : vec[i].l;

return true;

}

S.insert(Seg(vec[i].l, vec[i].r));

}

}

return false;

}

int main(){

// freopen("in.txt", "r", stdin);

while(scanf("%d", &n) == 1){

for(int i = 1; i <= n; i ++)

scanf("%d%d%d%d", &x[i][0], &y[i][0], &x[i][1], &y[i][1]);

DB l = 0, r = 1e6;

for(int i = 0; i <= 30; i ++){

DB mid = (l + r) / 2.0;

if(check(mid)) l = mid;

else r = mid;

}

if(!check(l)) puts("Impossible");

else printf("%.1f %.1f %.1f\n", X, Y, l);

}

}

**其他——**

**求最长回文字串 Manacher 算法**

char ch[N], str[N]; int p[N];

void Manacher(int n){

int mx = 0, id;

for(int i = 1; i <= n; i ++){

if(mx > i) p[i] = min(p[2 \* id - i], mx - i);

else p[i] = 1;

for(; str[i + p[i]] == str[i - p[i]] && i + p[i] <= n && i + p[i] >= 0; p[i] ++);

if(p[i] + i > mx)

mx = p[i] + i, id = i;

}

}

int cnt, ans, cas;

int main(){

//freopen("in.txt", "r", stdin);

while(scanf(" %s", ch + 1) == 1){

if(ch[1] == 'E') break;

str[0] = '#';

cnt = 0;

for(int i = 1, len = strlen(ch + 1); i <= len; i ++)

str[++ cnt] = ch[i], str[++ cnt] = '#';

Manacher(cnt);

ans = 0;

for(int i = 1; i <= cnt; i ++)

ans = max(ans, p[i] - 1);

printf("Case %d: %d\n", ++ cas, ans);

}

}

**KMP + EXTKMP**

int ext[MAXN], len, next[MAXN];

void extend\_kmp(char \*ch){

// ext 下标从 0 开始，ext[0] 无意义强制为 0 。。

memset(ext, 0, sizeof(ext));

len = strlen(ch);

int j = 0;

while(1 + j < len && ch[j] == ch[1 + j]) j ++;

ext[1] = j;

int k = 1;

for(int i = 2; i < len; i ++){

int Len = k + ext[k] - 1, L = ext[i - k];

if(L < Len - i + 1)

ext[i] = L;

else{

j = max(0, Len - i + 1);

while(i + j < len && ch[i + j] == ch[j]) j ++;

ext[i] = j, k = i;

}

}

}

void kmp(char \*ch){

// next 下标从 1 开始，next[1] 无意义强制为 0 。。

memset(next, 0, sizeof(next));

len = strlen(ch);

for(int i = len; i >= 1; i --) ch[i] = ch[i - 1];

int j = 0;

for(int i = 2; i <= len; i ++){

while(j && ch[i] != ch[j + 1]) j = next[j];

if(ch[i] == ch[j + 1]) j ++;

next[i] = j;

}

for(int i = 0; i < len; i ++) ch[i] = ch[i + 1]; ch[len] = '\0';

// 事后还原 。。

}

**DLX 精确覆盖（HDU4069，花式数独判唯一性）**

const int MAXC = 400, MAXR = 1000, MAXNODE = 4 \* 1000;

struct DLX{

int n, m, sz, H[MAXR], S[MAXC]; // S[] 是每列的节点数

int row[MAXNODE], col[MAXNODE];

int L[MAXNODE], R[MAXNODE], U[MAXNODE], D[MAXNODE];

int ans[MAXR], ansd; // ansd 是答案长度

void init(int \_n, int \_m){

n = \_n, m = \_m;

for(int i = 0; i <= m; i ++)

S[i] = 0, U[i] = D[i] = i, L[i] = i - 1, R[i] = i + 1;

R[m] = 0, L[0] = m;

sz = m;

for(int i = 1; i <= n; i ++) H[i] = -1;

}

void Link(int r, int c){ // Must be added in ascending order ..

++ S[col[++ sz] = c];

row[sz] = r;

D[sz] = D[c], U[sz] = c;

U[D[c]] = sz, D[c] = sz;

if(H[r] < 0) H[r] = L[sz] = R[sz] = sz;

else {

R[sz] = R[H[r]];

L[R[H[r]]] = sz;

L[sz] = H[r];

R[H[r]] = sz;

}

}

inline void rmv(int c){

L[R[c]] = L[c], R[L[c]] = R[c];

for(int i = D[c]; i != c; i = D[i])

for(int j = R[i]; j != i; j = R[j])

U[D[j]] = U[j], D[U[j]] = D[j], S[col[j]] --;

}

inline void rst(int c){

for(int i = U[c]; i != c; i = U[i])

for(int j = L[i]; j != i; j = L[j])

S[col[j]] ++, U[D[j]] = j, D[U[j]] = j;

L[R[c]] = c, R[L[c]] = c;

}

void dfs(int d){

if(num > 1) return;

if(R[0] == 0){

num ++;

for(int i = 0; i < d; i ++){

ch[(ans[i] - 1) / 9] = (ans[i] - 1) % 9 + '1';

}

return;

// ansd = d; return true;

}

int c = R[0];

for(int i = R[0]; i != 0; i = R[i])

if(S[i] < S[c]) c = i;

rmv(c);

for(int i = D[c]; i != c; i = D[i]){

ans[d] = row[i];

for(int j = R[i]; j != i; j = R[j]) rmv(col[j]);

dfs(d + 1);

if(num > 1) return;

for(int j = L[i]; j != i; j = L[j]) rst(col[j]);

}

rst(c);

}

void solve(vector <int> &v){

dfs(0);

return;

// v.clear();

// if(!dfs(0)) return false;

// for(int i = 0; i < ansd; i ++) v.PB(ans[i]);

// return true;

}

};

DLX Solver;

int a[10][10], vis[10][10], d[4][2] = {{-1, 0}, {0, 1}, {1, 0}, {0, -1}};

void bfs(int x, int y, int id) {

vis[x][y] = id;

queue <PII> Q;

Q.push(MP(x, y));

while(!Q.empty()) {

PII u = Q.front(); Q.pop();

for(int i = 0; i < 4; i ++){

if(a[u.first][u.second] >> (i + 4) & 1) continue;

int xx = u.first + d[i][0], yy = u.second + d[i][1];

if(xx < 0 || yy < 0 || xx >= 9 || yy >= 9) continue;

if(vis[xx][yy] == -1) Q.push(MP(xx, yy)), vis[xx][yy] = id;

}

}

}

inline int encode(int a, int b, int c){

return a \* 9 \* 9 + b \* 9 + c + 1;

}

inline void decode(int x, int &a, int &b, int &c){

x --;

a = x / 81; x -= a \* 81;

b = x / 9; x -= b \* 9;

c = x;

}

int cas, T;

int main() {

// freopen("in.txt", "r", stdin);

scanf("%d", &T);

while(T --) {

for(int i = 0; i < 9; i ++)

for(int j = 0; j < 9; j ++)

scanf("%d", &a[i][j]);

Solver.init(729, 324);

int cnt = 0;

memset(vis, -1, sizeof(vis));

for(int i = 0; i < 9; i ++)

for(int j = 0; j < 9; j ++)

if(vis[i][j] == -1)

bfs(i, j, cnt ++);

for(int r = 0; r < 9; r ++)

for(int c = 0; c < 9; c ++) {

int t = (a[r][c] & 15);

if(t == 0) {

for(int v = 0; v < 9; v ++) {

int R = encode(r, c, v);

Solver.Link(R, encode(0, r, c));

Solver.Link(R, encode(1, r, v));

Solver.Link(R, encode(2, c, v));

Solver.Link(R, encode(3, vis[r][c], v));

}

} else {

t --;

int R = encode(r, c, t);

Solver.Link(R, encode(0, r, c));

Solver.Link(R, encode(1, r, t));

Solver.Link(R, encode(2, c, t));

Solver.Link(R, encode(3, vis[r][c], t));

}

}

printf("Case %d:\n", ++ cas);

num = 0;

vector <int> vec;

Solver.solve(vec);

if(num > 1) {

puts("Multiple Solutions");

continue;

} else if(num == 0) {

puts("No solution");

continue;

}

for(int i = 0; i < 81; i ++){

printf("%c", ch[i]);

if(i % 9 == 8) puts("");

}

}

}

**DLX 最小重复覆盖（HDU3498）**

const int MAXC = 60, MAXR = 60, MAXNODE = 60 \* 60;

struct DLX{

int n, m, sz, H[MAXR], S[MAXC]; // S[] 是每列的节点数

int row[MAXNODE], col[MAXNODE];

int L[MAXNODE], R[MAXNODE], U[MAXNODE], D[MAXNODE];

int ans[MAXR], ansd; // ansd 是答案长度

void init(int \_n, int \_m){

n = \_n, m = \_m;

for(int i = 0; i <= m; i ++)

S[i] = 0, U[i] = D[i] = i, L[i] = i - 1, R[i] = i + 1;

R[m] = 0, L[0] = m;

sz = m;

for(int i = 1; i <= n; i ++) H[i] = -1;

ansd = INF; // Here is different from accurate cover ..

}

void Link(int r, int c){ // Must be added in ascending order ..

++ S[col[++ sz] = c];

row[sz] = r;

D[sz] = D[c], U[sz] = c;

U[D[c]] = sz, D[c] = sz;

if(H[r] < 0) H[r] = L[sz] = R[sz] = sz;

else {

R[sz] = R[H[r]];

L[R[H[r]]] = sz;

L[sz] = H[r];

R[H[r]] = sz;

}

}

void rmv(int c){

for(int i = D[c]; i != c; i = D[i])

L[R[i]] = L[i], R[L[i]] = R[i];

}

void rst(int c){

for(int i = U[c]; i != c; i = U[i])

L[R[i]] = i, R[L[i]] = i;

}

int h(){

bool vis[MAXC];

memset(vis, 0, sizeof(vis));

int ret = 0;

for(int i = R[0]; i != 0; i = R[i])

if(!vis[i]){

ret ++;

vis[i] = true;

for(int j = D[i]; j != i; j = D[j])

for(int k = R[j]; k != j; k = R[k])

vis[col[k]] = true;

}

return ret;

}

void dfs(int d){

if(d + h() >= ansd) return;

if(R[0] == 0){

ansd = d;

return;

}

int c = R[0];

for(int i = R[0]; i != 0; i = R[i])

if(S[i] < S[c]) c = i;

for(int i = D[c]; i != c; i = D[i]){

rmv(i);

ans[d] = row[i];

for(int j = R[i]; j != i; j = R[j]) rmv(j);

dfs(d + 1);

for(int j = L[i]; j != i; j = L[j]) rst(j);

rst(i);

}

}

bool solve(vector <int> &v){

v.clear();

dfs(0);

if(ansd == INF) return false;

for(int i = 0; i < ansd; i ++) v.PB(ans[i]);

return true;

}

};

DLX Solver;

int n, m, u, v;

int main(){

// freopen("in.txt", "r", stdin);

while(scanf("%d%d", &n, &m) == 2){

Solver.init(n, n);

for(int i = 1; i <= n; i ++)

Solver.Link(i, i);

for(int i = 1; i <= m; i ++){

scanf("%d%d", &u, &v);

Solver.Link(u, v);

Solver.Link(v, u);

}

vector <int> vec;

Solver.solve(vec);

printf("%d\n", vec.size());

//for(int i = 0; i < vec[0].size(); i ++) cout << vec[0][i] << ' '; cout << endl;

}

}

**矩形离散化（UVA1092，对线段离散化要对块离散化简单不少，后者感觉完全不可写）**

int r, c, K, cas, x\_1[N], x\_2[N], y\_1[N], y\_2[N]; LL ans;

bool A[N << 1][N << 1], f[N << 1][N << 1];

vector <int> X, Y;

int main(){

// freopen("in.txt", "r", stdin);

while(scanf("%d%d%d", &r, &c, &K) == 3){

if(!r && !c && !K) break;

X.clear(), Y.clear();

for(int i = 1; i <= K; i ++){

scanf("%d%d%d%d", &x\_1[i], &y\_1[i], &x\_2[i], &y\_2[i]);

X.PB(x\_1[i]), X.PB(x\_2[i] + 1);

Y.PB(y\_1[i]), Y.PB(y\_2[i] + 1);

}

X.PB(0), X.PB(c);

Y.PB(0), Y.PB(r);

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

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

memset(A, 0, sizeof(A));

memset(f, 0, sizeof(f));

for(int i = 1; i <= K; i ++){

int k = lower\_bound(Y.begin(), Y.end(), y\_2[i] + 1) - Y.begin();

for(int j = 0; j < X.size(); j ++)

if(X[j] > x\_1[i] && X[j] <= x\_2[i] + 1)

A[j][k] = 1;

}

f[X.size() - 1][Y.size() - 1] = 1; ans = 0;

for(int i = X.size() - 1; i; i --)

for(int j = Y.size() - 1; j; j --){

if(A[i][j]) continue;

if(!f[i][j])

ans += (LL) (X[i] - X[i - 1]) \* (Y[j] - Y[j - 1]);

else

f[i - 1][j] |= f[i][j], f[i][j - 1] |= f[i][j];

}

printf("Case %d: %lld\n", ++ cas, ans);

}

}

**可重复K短路（A\*）**

#include <iostream>

#include <vector>

#include <queue>

#define MAXN 1005

#define MAXM 100005

#define inf 0x3fffffff

using namespace std;

struct AStar{int F,G,V;}TTemp;//G=F+V..V用dijkstra预处理

struct Edge{int V,Cost;}Temp;

int N,M,S,T,K,Dist[MAXN],Cnt[MAXN];

vector<Edge>E[MAXM],EE[MAXM];

bool operator < (const AStar &A,const AStar &B){return A.F>B.F;}

void Spfa(){

bool Flag[MAXN];

memset(Flag,true,sizeof(Flag));

for(int i=1;i<=N;i++)Dist[i]=inf;

queue<int>Q;

Q.push(T);Dist[T]=0;

while(!Q.empty()){

int F=Q.front();

Q.pop();

Flag[F]=true;

for(int i=0,SS=EE[F].size();i<SS;i++)

if(Dist[EE[F][i].V]>EE[F][i].Cost+Dist[F]){

Dist[EE[F][i].V]=EE[F][i].Cost+Dist[F];

if(Flag[EE[F][i].V]){

Q.push(EE[F][i].V);

Flag[EE[F][i].V]=false;

}

}

}

//for(int i=1;i<=N;i++)

// cout<<i<<' '<<Dist[i]<<endl;

}

int A\_Star(){

if(Dist[S]==inf)return -1;

priority\_queue <AStar> Q;

TTemp.F=Dist[S];TTemp.V=S;TTemp.G=0;

Q.push(TTemp);

while(!Q.empty()){

TTemp=Q.top();

Q.pop();

int F=TTemp.V;

//cout<<endl<<TTemp.V<<' '<<TTemp.G<<' '<<TTemp.F<<endl<<"-------------"<<endl;

Cnt[F]++;

if(Cnt[T]==K)return TTemp.G;

if(Cnt[F]>K)break;

for(int i=0,SS=E[F].size();i<SS;i++){

AStar AA;

AA.G=TTemp.G+E[F][i].Cost;

AA.F=AA.G+Dist[E[F][i].V];

AA.V=E[F][i].V;

//cout<<AA.V<<' '<<AA.G<<' '<<AA.F<<endl;

Q.push(AA);

}

}

return -1;

}

int main(){

cin>>N>>M;

int x,y,z;

for(int i=1;i<=M;i++){

scanf("%d%d%d",&x,&y,&z);

Temp.V=y,Temp.Cost=z;

E[x].push\_back(Temp);

Temp.V=x;EE[y].push\_back(Temp);

}

scanf("%d%d%d",&S,&T,&K);

if(S==T)K++;

Spfa();

cout<<A\_Star()<<endl;

system("pause");

return 0;

}

**JAVA——**

**JAVA快速读入**

import java.util.\*;

import java.io.\*;

import java.math.\*;

/\*

\* Thanks to Petr providing this template.

\*/

public class Main {

public static void main(String[] args) {

InputStream inputStream = System.in;

OutputStream outputStream = System.out;

InputReader in = new InputReader(inputStream);

PrintWriter out = new PrintWriter(outputStream);

int T;

T = in.nextInt();

for(int kase = 1; kase <= T; kase ++){

String A = in.next();

String B = in.next();

BigInteger a = new BigInteger (A, 2);

BigInteger b = new BigInteger (B, 2);

a = a.gcd(b);

out.println("Case #" + kase + ": " + a.toString(2));

}

out.flush();

// out.close();

}

}

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());

}

}

**JAVA高精度卡特兰数**

import java.util.\*;

import java.math.\*;

public class Main{

public static void main(String[] args){

Scanner cin = new Scanner(System.in);

BigDecimal \_3[] = new BigDecimal[105];

BigDecimal C[][] = new BigDecimal[105][105];

for(int i = 0; i <= 100; i ++){

for(int j = 0; j <= 100; j ++){

C[i][j] = BigDecimal.valueOf(0);

}

}

// Must initalize !!

\_3[0] = BigDecimal.valueOf(1);

for(int i = 1; i <= 100; i ++){

\_3[i] = \_3[i - 1].multiply(BigDecimal.valueOf(3));

}

C[0][0] = BigDecimal.valueOf(1);

for(int i = 1; i <= 100; i ++){

C[i][0] = BigDecimal.valueOf(1);

for(int j = 1; j <= 100; j ++){

C[i][j] = C[i - 1][j - 1].add(C[i - 1][j]);

}

}

BigDecimal ans[] = new BigDecimal[105];

ans[1] = BigDecimal.valueOf(0);

ans[2] = BigDecimal.valueOf(1.5);

for(int i = 3; i <= 100; i ++){

BigDecimal s = BigDecimal.valueOf(0), t = BigDecimal.valueOf(0);

for(int j = 1; j <= i - 1; j ++){

s = s.add(ans[j].multiply( C[i][j].multiply(BigDecimal.valueOf(3)) ) );

t = t.add(C[i][j].multiply(BigDecimal.valueOf(3)));

}

s = s.add(\_3[i]);

ans[i] = s.divide(t, 50, BigDecimal.ROUND\_HALF\_EVEN);

}

int n;

while(cin.hasNext()){

n = cin.nextInt();

System.out.println(ans[n]);

}

}}