



作者：oneplus

目录

[比赛常用模板 1](#_Toc5432)

[手动扩栈 11](#_Toc29375)

[字符串 11](#_Toc8859)

[kmp 11](#_Toc22495)

[AC自动机 12](#_Toc20212)

[后缀自动机 15](#_Toc13511)

[后缀数组 16](#_Toc13071)

[回文树 17](#_Toc13857)

[序列自动机 20](#_Toc19353)

[Lyndon分解 21](#_Toc9688)

[马拉车算法 23](#_Toc8983)

[广义后缀自动机 24](#_Toc13202)

[数论 29](#_Toc9449)

[逆元 29](#_Toc30269)

[扩展欧几里得 29](#_Toc10535)

[费马小定理 29](#_Toc5864)

[线性递推 29](#_Toc2194)

[线性求阶乘的逆元 29](#_Toc22396)

[线性筛同时求质数、欧拉函数、莫比乌斯函数、莫比乌斯函数前缀和 30](#_Toc16632)

[1 <= x <= n, 1 <= y <= m, gcd(x, y) = k的个数 30](#_Toc31755)

[FWT 31](#_Toc13745)

[FFT 31](#_Toc5073)

[数值积分 33](#_Toc13610)

[球盒问题 34](#_Toc18212)

[第二类斯特林数 35](#_Toc29589)

[第一类斯特林数 36](#_Toc14810)

[Catalan数 36](#_Toc27694)

[错排 38](#_Toc19958)

[康拓展开 38](#_Toc26053)

[莫比乌斯反演 39](#_Toc12436)

[欧拉函数 40](#_Toc31843)

[ax + by = c的整数解 41](#_Toc11121)

[Lucas定理 41](#_Toc24900)

[模线性方程组 42](#_Toc18310)

[Heron法开方 43](#_Toc19327)

[平方剩余 44](#_Toc29060)

[约瑟夫环（编号为0~n-1） 44](#_Toc10003)

[Miller-Rabin素性测试 44](#_Toc5948)

[O(1) gcd 48](#_Toc15665)

[费马平方和定理 49](#_Toc11572)

[拉格朗日插值法模板 50](#_Toc10232)

[拉姆齐定理(r(3,3)) 53](#_Toc17911)

[牛顿迭代 54](#_Toc13573)

[求1~n中二进制数第k位为1的种数 55](#_Toc28907)

[求等差数列整除整数下取整求和 56](#_Toc13694)

[原根 57](#_Toc1187)

[斐波那契数列 59](#_Toc28551)

[二次剩余模板(mod为奇质数) 59](#_Toc22802)

[BM模板 60](#_Toc17469)

[Kummer定理 73](#_Toc22537)

[BSGS大步小步算法 76](#_Toc30379)

[高斯-约旦消元法 78](#_Toc16097)

[线性基 79](#_Toc9051)

[min\_25筛 83](#_Toc582)

[洲阁筛 85](#_Toc13676)

[杜教筛 88](#_Toc3446)

[裴蜀定理 91](#_Toc19795)

[中国剩余定理 92](#_Toc8316)

[蔡勒公式 94](#_Toc28340)

[超级卡特兰数 95](#_Toc27941)

[数据结构 95](#_Toc13813)

[左偏树 95](#_Toc20679)

[单调队列 97](#_Toc7036)

[ST表 98](#_Toc14608)

[线段树 101](#_Toc14767)

[区间合并 101](#_Toc29811)

[势能线段树 104](#_Toc29748)

[李超树 107](#_Toc29431)

[主席树 108](#_Toc4618)

[Treap 110](#_Toc25196)

[Splay 114](#_Toc10597)

[WBLT 117](#_Toc10577)

[AVL 树 119](#_Toc29559)

[替罪羊树 121](#_Toc7669)

[插入 122](#_Toc3989)

[删除 123](#_Toc923)

[upper\_bound 123](#_Toc14412)

[at 124](#_Toc26013)

[前驱后继 124](#_Toc17003)

[笛卡尔树 124](#_Toc26864)

[可持久化数组 126](#_Toc15626)

[可持久化平衡树 127](#_Toc11887)

[可持久化字典树 130](#_Toc787)

[线段树套线段树 132](#_Toc29914)

[K-D Tree 133](#_Toc20961)

[偏序 139](#_Toc24426)

[LCT 145](#_Toc21988)

[维护树链信息 145](#_Toc11880)

[维护连通性质 148](#_Toc25974)

[维护边双连通分量 150](#_Toc24881)

[维护边权 153](#_Toc9024)

[维护子树信息 156](#_Toc13696)

[图论 160](#_Toc19683)

[树的直径 160](#_Toc22019)

[dfs 160](#_Toc24508)

[树形 DP 162](#_Toc23159)

[LCA 163](#_Toc19711)

[倍增 163](#_Toc3564)

[Tarjan 算法 166](#_Toc32641)

[用欧拉序列转化为 RMQ 问题 170](#_Toc21471)

[树的重心 171](#_Toc18657)

[定义 171](#_Toc2877)

[性质 171](#_Toc14579)

[求法 171](#_Toc1696)

[树链剖分 172](#_Toc14751)

[Dsu on tree 179](#_Toc11067)

[虚树 183](#_Toc23184)

[树分治 191](#_Toc5195)

[动态点分治 200](#_Toc21339)

[AHU算法 213](#_Toc18196)

[树哈希 219](#_Toc14362)

[「BJOI2015」树的同构 219](#_Toc9130)

[HDU 6647 223](#_Toc31617)

[矩阵树定理 230](#_Toc24855)

[生成树 234](#_Toc29432)

[最小生成树唯一性 234](#_Toc6044)

[严格次小生成树 236](#_Toc29770)

[Kruskal 重构树 243](#_Toc12835)

[斯坦纳树 249](#_Toc8666)

[最小树形图 251](#_Toc10271)

[朱刘算法 252](#_Toc24965)

[Tarjan 的 DMST 算法 253](#_Toc11104)

[最短路 259](#_Toc7068)

[差分约束 263](#_Toc4527)

[例题 luogu P1993 小 K 的农场 263](#_Toc2710)

[k 短路 265](#_Toc16430)

[同余最短路 270](#_Toc26989)

[连通性相关 274](#_Toc20175)

[强连通分量 274](#_Toc32191)

[双连通分量 275](#_Toc3163)

[割点 275](#_Toc9264)

[割边 278](#_Toc4069)

[2-SAT 279](#_Toc30878)

[欧拉图 282](#_Toc10905)

[最小环 286](#_Toc16674)

[Floyd 286](#_Toc25930)

[平面图 288](#_Toc14839)

[网络流 290](#_Toc20271)

[最大流 290](#_Toc4541)

[最小割 300](#_Toc20386)

[费用流 303](#_Toc8372)

[Prufer 序列 312](#_Toc31470)

[LGV 引理 317](#_Toc31453)

[弦图 320](#_Toc10857)

[计算几何 321](#_Toc15402)

[Pick 定理 321](#_Toc23143)

[三角剖分 323](#_Toc10018)

[凸包 323](#_Toc9423)

[扫描线 324](#_Toc12702)

[Atlantis 问题 324](#_Toc9563)

[旋转卡壳 328](#_Toc14485)

[随机增量法 333](#_Toc15558)

[DP 336](#_Toc6920)

[背包dp 336](#_Toc4723)

[区间dp 339](#_Toc15758)

[树形dp 339](#_Toc28761)

[动态dp 341](#_Toc17174)

[DP优化 341](#_Toc13105)

[单调队列 341](#_Toc27861)

[斜率优化 343](#_Toc6463)

[数位dp 343](#_Toc25204)

[插头dp 345](#_Toc19495)

# 比赛常用模板

/\*\*/  
#pragma GCC optimize(3,"Ofast","inline")  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
const long long mod = 1e9 + 7;  
const int maxn = 1e5 + 5;  
  
int t, n, m, tot, head[maxn], prim[maxn];  
char s[maxn];  
bool vis[maxn];  
  
LL ksm(LL a, LL b){  
 return (a \* b - (LL)((long double)a / mod \* b) \* mod + mod) % mod;  
}  
  
void read(int &x){  
 char ch = getchar();x = 0;  
 for (; ch < '0' || ch > '9'; ch = getchar());  
 for (; ch >='0' && ch <= '9'; ch = getchar()) x = x \* 10 + ch - '0';  
}  
  
void read(int &x){  
 char ch = getchar();x = 0;  
 int f = 1;  
 for (; ch < '0' || ch > '9'; ch = getchar()) if(ch == '-') f = -1;  
 for (; ch >='0' && ch <= '9'; ch = getchar()) x = x \* 10 + ch - '0';  
 x = f \* x;  
}  
  
namespace fastIO  
{  
 static char buf[100000], \*h = buf, \*dp = buf;//缓存开大可减少读入时间，看题目给的空间  
 #define gc h == dp && (dp = (h = buf) + fread(buf, 1, 100000, stdin), h == dp) ? EOF : \*h++//不能用fread则换成getchar  
 template<typename T>  
 inline void read(T &x)  
 {  
 int f = 1;x = 0;  
 register char c(gc);  
 while(c > '9' || c < '0'){  
 if(c == '-') f = -1;  
 c = gc;  
 }  
 while(c <= '9' && c >= '0') x = (x << 1) + (x << 3) + (c ^ 48), c = gc;  
 x \*= f;  
 }  
 template<typename T>  
 void output(T x)  
 {  
 if(x < 0){putchar('-'); x = ~(x - 1);}  
 static int s[20], top = 0;  
 while(x){s[++top]= x%10; x /= 10;}  
 if(!top) s[++top] = 0;  
 while(top) putchar(s[top--] + '0');  
 }  
}  
using namespace fastIO;  
  
inline \_\_int128 read(){  
 \_\_int128 x = 0, f = 1;  
 char ch = getchar();  
 while(ch < '0'|| ch > '9'){  
 if(ch == '-')  
 f = -1;  
 ch = getchar();  
 }  
 while(ch >= '0' && ch <= '9'){  
 x = x \* 10 + ch - '0';  
 ch = getchar();  
 }  
 return x \* f;  
}  
  
inline void print(\_\_int128 x){  
 if(x < 0){  
 putchar('-');  
 x = -x;  
 }  
 if(x > 9) print(x / 10);  
 putchar(x % 10 + '0');  
}  
  
LL gcd(LL a, LL b)  
{  
 if(a < 0) a =- a;  
 if(b < 0) b =- b;  
 if(a && b)  
 while(a %= b ^= a ^= b ^= a);  
 return a + b;  
}  
  
  
LL poww(LL x, LL num){  
 LL res = 1;  
 while(num){  
 if(num & 1) res = res \* x % mod;  
 x = x \* x % mod;  
 num >>= 1;  
 }  
 return res;  
}  
  
LL ex\_gcd(LL a, LL b, LL &x, LL &y){  
 if(!b){  
 x = 1, y = 0;  
 return ;  
 }  
 ex\_gcd(b, a % b, y, x);  
 y -= a / b \* x;  
}  
  
LL phi(LL x){  
 LL res = x;  
 for (LL i = 2; i \* i <= x; i++){  
 if(x % i == 0){  
 res = res / i \* (i - 1);  
 while(x % i == 0) x /= i;  
 }  
 }  
 if(x > 1) res = res / x \* (x - 1);  
 return res;  
}  
  
void get(){  
 for (int i = 2; i <= maxn; i++){  
 if(!vis[i]) prim[tot++] = i;  
 for (int j = 0; j < tot && i \* prim[j] <= maxn; j++){  
 vis[prim[j] \* i] = true;  
 if(i % prim[j] == 0) break;  
 }  
 }  
}  
  
void get(){ // 线性求逆元  
 inv[1] = 1;  
 for (int i = 2; i <= 10000000; i++) inv[i] = (LL)(mod - mod / i) \* inv[mod % i] % mod;  
}  
  
void kmp(){  
 int len = strlen(s + 1);  
 for (int i = 2, j = 0; i <= len; i++){  
 j = nxt[i - 1];  
 while(j && s[i] != s[j + 1]) j = nxt[j];  
 j += s[j + 1] == s[i];  
 nxt[i] = j;  
 }  
 for (int i = 1; i <= len; i++) printf("%d ", nxt[i]);  
 printf("\n");  
}  
  
void get\_kmp(){  
 int mx = 0, len1 = strlen(T + 1);  
 for (int i = 1, j = 0; i <= len1; i++){  
 while(j && T[i] != s[j + 1]) j = nxt[j];  
 j += s[j + 1] == T[i];  
 mx = max(mx, j);  
 }  
}  
  
//下标从1开始，nxt[i]和extend[i] 多1  
void Extend(){  
 int len = strlen(T + 1);  
 nxt[1] = len + 1;  
 int i = 1;  
 while(T[i] == T[i + 1] && i + 1 <= len) i++;  
 nxt[2] = i;  
 for (int i = 3, j = 2; i <= len; i++){  
 if(nxt[i - j + 1] + i < nxt[j] + j) nxt[i] = nxt[i - j + 1];  
 else{  
 int x = max(1, nxt[j] + j - i);  
 while(T[x] == T[i + x - 1] && i + x - 1 <= len) x++;  
 nxt[i] = x;  
 j = i;  
 }  
 }  
}  
  
void get\_extend(){  
 Extend();  
 int lens = strlen(s + 1), lent = strlen(T + 1);  
 int i = 1;  
 while(s[i] == T[i] && i <= min(lens, lent)) i++;  
 extend[1] = i;  
 for (int i = 2, j = 1; i <= lens; i++){  
 if(nxt[i - j + 1] + i < extend[j] + j) extend[i] = nxt[i - j + 1];  
 else{  
 int x = max(1, extend[j] + j - i);  
 while(T[x] == s[x + i - 1] && x <= lent && x + i - 1 <= lens) x++;  
 extend[i] = x;  
 j = i;  
 }  
 }  
 // for (int i = 1; i <= lens; i++) printf("%d ", extend[i]);  
 // printf("\n");  
}  
  
  
//下标从0开始  
void Extend(){  
 int len = strlen(T);  
 nxt[0] = len;  
 int i = 0;  
 while(T[i] == T[i + 1] && i + 1 < len) i++;  
 nxt[1] = i;  
 for (int i = 2, j = 1; i < len; i++){  
 if(nxt[i - j] + i < nxt[j] + j) nxt[i] = nxt[i - j];  
 else{  
 int x = max(0, nxt[j] + j - i);  
 while(T[x] == T[i + x] && i + x < len) x++;  
 nxt[i] = x;  
 j = i;  
 }  
 }  
}  
  
void get\_extend(){  
 Extend();  
 int lens = strlen(s), lent = strlen(T);  
 int i = 0;  
 while(s[i] == T[i] && i < min(lens, lent)) i++;  
 extend[0] = i;  
 for (int i = 1, j = 0; i < lens; i++){  
 if(nxt[i - j] + i < extend[j] + j) extend[i] = nxt[i - j];  
 else{  
 int x = max(0, extend[j] + j - i);  
 // printf("%d %d\n", i, x);  
 while(T[x] == s[x + i] && x < lent && x + i < lens) x++;  
 extend[i] = x;  
 j = i;  
 }  
 }  
 for (int i = 0; i < lens; i++) printf("%d ", extend[i]);  
 printf("\n");  
}  
  
int lim, hib, rt = 3, rev[1 << n | 1]; //2^n, rt为mod的原根  
  
void init(int len){ //数组长度  
 lim = 1, hib = -1;  
 while(lim <= len) lim <<= 1, hib++;  
 for (int i = 0; i < lim; i++){  
 rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << hib);  
 }  
}  
  
void fwt\_or(int \*a, int type){  
 for (int i = 1; i < lim; i <<= 1){  
 for (int p = i << 1, j = 0; j < lim; j += p){  
 for (int k = 0; k < i; k++){  
 if(type == 1) a[i + j + k] = (a[j + k] + a[i + j + k]) % mod;  
 else a[i + j + k] = (a[i + j + k] + mod - a[j + k]) % mod;  
 }  
 }  
 }  
}  
  
void fwt\_and(int \*a, int type){  
 for (int i = 1; i < lim; i <<= 1){  
 for (int p = i << 1, j = 0; j < lim; j += p){  
 for (int k = 0; k < i; k++){  
 if(type == 1) a[j + k] = (a[j + k] + a[i + j + k]) % mod;  
 else a[j + k] = (a[j + k] + mod - a[i + j + k]) % mod;  
 }  
 }  
 }  
}  
  
void fwt\_xor(int \*a, int type){  
 for (int i = 1; i < lim; i <<= 1){  
 for (int p = i << 1, j = 0; j < lim; j += p){  
 for (int k = 0; k < i; k++){  
 int x = a[j + k], y = a[i + j + k];  
 a[j + k] = (x + y) % mod, a[i + j + k] = (x + mod - y) % mod;  
 if(!type) a[j + k] = 1LL \* a[j + k] \* inv2 % mod,  
 a[i + j + k] = 1LL \* a[i + j + k] \* inv2 % mod;  
 }  
 }  
 }  
}  
  
mod = 998244353;  
lim = 1, G = 3, Gi = 332748118; //Gi = poww(3, mod - 2)  
  
void ntt(int \*a, int type){  
 for (int i = 0; i < lim; i++) if(i < rev[i]) swap(a[i], a[rev[i]]);  
 static int x, y;  
 for (int mid = 1; mid < lim; mid <<= 1){  
 int len = mid << 1, wn = poww(type ? G : Gi , (mod - 1) / (mid << 1));  
 for (int i = 0; i < lim; i += len){  
 for (int j = 0, w = 1; j < mid; j++, w = 1LL \* w \* wn % mod){  
 x = a[i + j], y = 1LL \* w \* a[i + mid + j] % mod;  
 a[i + j] = (x + y) % mod, a[i + mid + j] = (x + mod - y) % mod;  
 }  
 }  
 }  
 if(!type){  
 LL inv = poww(lim, mod - 2);  
 for (int i = 0; i < lim; i++) a[i] = 1LL \* inv \* a[i] % mod;  
 }  
}  
  
while(lim <= n \* 2) lim <<= 1, L++;  
for (int i = 0; i < lim; i++) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (L - 1));  
  
int find1(int x)  
{  
 return x==fa[x]?x:fa[x]=find1(fa[x]);  
}  
inline int find2(int x)  
{  
 return x==fa[x]?x:fa[x]=find2(fa[x]);  
}  
inline int find3(int x)  
{  
 while(x!=fa[x])x=fa[x]=fa[fa[x]];return x;  
}  
inline int find4(int x)  
{  
 int se=0;st[0]=x;  
 while(fa[x]!=x)st[++se]=x=fa[x];  
 while(se)fa[st[--se]]=x;  
 return x;  
}  
inline int find5(int x)  
{  
 int t=x,p;  
 while(x!=fa[x])x=fa[x];  
 while(t!=x)p=fa[t],fa[t]=x,t=p;  
 return x;  
}  
int find6(int x)  
{  
 return 0>fa[x]?x:fa[x]=find6(fa[x]);  
}  
inline int find7(int x)  
{  
 return 0>fa[x]?x:fa[x]=find7(fa[x]);  
}  
inline int find8(int x)  
{  
 while(fa[x]>=0&&fa[fa[x]]>=0)x=fa[x]=fa[fa[x]];return fa[x]<0?x:fa[x];  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 return 0;  
}  
/\*\*/  
  
#include <cstdio>  
#include <vector>  
#include <queue>  
#include <cstring>  
#include <cmath>  
#include <map>  
#include <set>  
#include <string>  
#include <iostream>  
#include <algorithm>  
#include <iomanip>  
#include <stack>  
#include <queue>  
using namespace std;  
#define sd(n) scanf("%d", &n)  
#define sdd(n, m) scanf("%d%d", &n, &m)  
#define sddd(n, m, k) scanf("%d%d%d", &n, &m, &k)  
#define pd(n) printf("%d\n", n)  
#define pc(n) printf("%c", n)  
#define pdd(n, m) printf("%d %d\n", n, m)  
#define pddd(n, m, z) printf("%d %d %d\n", n, m, z)  
#define pld(n) printf("%lld\n", n)  
#define pldd(n, m) printf("%lld %lld\n", n, m)  
#define plddd(n, m, z) printf("%lld %lld %lld\n", n, m, z)  
#define sld(n) scanf("%lld", &n)  
#define sldd(n, m) scanf("%lld%lld", &n, &m)  
#define slddd(n, m, k) scanf("%lld%lld%lld", &n, &m, &k)  
#define sf(n) scanf("%lf", &n)  
#define sc(n) scanf("%c", &n)  
#define sff(n, m) scanf("%lf%lf", &n, &m)  
#define sfff(n, m, k) scanf("%lf%lf%lf", &n, &m, &k)  
#define ss(str) scanf("%s", str)  
#define rep(i, a, n) for (int i = a; i <= n; i++)  
#define per(i, a, n) for (int i = a; i >= n; i--)  
#define mem(a, n) memset(a, n, sizeof(a))  
#define debug(x) cout << #x << ": " << x << endl  
#define pb push\_back  
#define all(x) (x).begin(), (x).end()  
#define fi first  
#define se second  
#define ce i == n ? '\n' : ' '  
#define mod(x) ((x) % MOD)  
#define gcd(a, b) \_\_gcd(a, b)  
#define lowbit(x) (x & -x)  
typedef pair<int, int> PII;  
typedef long long ll;  
typedef unsigned long long ull;  
typedef long double ld;  
const int MOD = 1e9 + 7;  
const double eps = 1e-9;  
const ll INF = 0x3f3f3f3f3f3f3f3fll;  
const int inf = 0x3f3f3f3f;  
inline int read()  
{  
 int ret = 0, sgn = 1;  
 char ch = getchar();  
 while (ch < '0' || ch > '9')  
 {  
 if (ch == '-')  
 sgn = -1;  
 ch = getchar();  
 }  
 while (ch >= '0' && ch <= '9')  
 {  
 ret = ret \* 10 + ch - '0';  
 ch = getchar();  
 }  
 return ret \* sgn;  
}  
inline void Out(int a) //ê?3?ía1ò  
{  
 if (a > 9)  
 Out(a / 10);  
 putchar(a % 10 + '0');  
}  
   
ll gcd(ll a, ll b)  
{  
 return b == 0 ? a : gcd(b, a % b);  
}  
   
ll lcm(ll a, ll b)  
{  
 return a \* b / gcd(a, b);  
}  
///快速幂m^k%mod  
ll qpow(ll x, ll n, ll mod)  
{  
 ll res = 1;  
 while (n)  
 {  
 if (n & 1)  
 res = (res \* x) % mod;  
 x = x \* x % mod, n >>= 1;  
 }  
 return res;  
}  
   
const int N = 1e5 + 50;  
int n;  
char s1[N], s2[N];  
int cnt, res, pos;  
vector<int> v;

# 手动扩栈

g++  
int size = 256 << 20; //256M  
char\* p = (char\*)malloc(size) + size;  
\_\_asm\_\_("movl %0, %%esp\n" :: "r"(p));  
  
g++ x64  
extern void main2() \_\_asm\_\_("main2");  
void main2() {  
 ...  
 exit(0);  
}  
int main() {  
 int size = 256 << 20;  
 \_\_asm\_\_ \_\_volatile\_\_("movq %0, %%rsp\njmp main2"::"r"((char\*)malloc(size)+size));  
}  
  
c++  
#pragma comment(linker, "/STACK:102400000,102400000")

# 字符串

## kmp

#include <cstdio>  
#include <cstring>  
   
int ns, nss;  
char s[10001], ss[1000001];  
int Next[10001], fail[10001];  
   
void makeNext() {  
 int k = 0;  
 for (int i = 1; i <= ns; i++) {  
 Next[i] = k;  
 fail[i] = s[i]==s[k]?fail[k]:k;  
 while (k && s[i] != s[k])  
 k = fail[k];  
 if (s[i] == s[k])  
 k++;  
 }  
}  
   
// 可重叠查找  
int match() {  
 int rst = 0;  
 int j = 0;  
 for (int i = 0; i < nss; i++) {  
 while (j && ss[i] != s[j])  
 j = fail[j];  
 if (ss[i] == s[j]) {  
 j++;  
 if (j == ns) {  
 rst++;  
 }  
 }  
 }  
 return rst;  
}  
   
// 循环节长度  
int repetend() {  
 int len = ns-Next[ns];  
 return ns%len?ns:len;  
}  
   
int main() {  
 int T;  
 scanf("%d", &T);  
 while (T--) {  
 scanf("%s%s", s, ss);  
 ns = strlen(s);  
 nss = strlen(ss);  
 makeNext();  
 printf("%d\n", match());  
 }  
 return 0;  
}

## AC自动机

/// HDU 3065  
#include <cstdio>  
#include <cstring>  
#include <algorithm>  
#include <queue>  
using namespace std;  
   
const int MAXN = 50 \* 1000 + 10;  
int n;  
char mode[1010][60];  
char a[2000000 + 10];  
int ans[1010];  
   
struct Node {  
 int val;  
 int fail;  
 int next[26];  
} node[MAXN];  
int con;  
   
void insertTrie(char str[], int id) {  
 int p = 0;  
 for (int i = 0; str[i]; i++) {  
 int ha = str[i] - 'A';  
 if (!node[p].next[ha]) {  
 node[p].next[ha] = con++;  
 }  
 p = node[p].next[ha];  
 }  
 node[p].val = id;  
}  
   
void makeFail() {  
 queue<int> que;  
 que.push(0);  
 while (!que.empty()) {  
 int p = que.front();  
 que.pop();  
 for (int i = 0; i<26; i++) {  
 if (!node[p].next[i]) continue;  
 if (p == 0) node[node[p].next[i]].fail = 0;  
 else {  
 int temp = node[p].fail;  
 while (temp) {  
 if (node[temp].next[i]) {  
 node[node[p].next[i]].fail = node[temp].next[i];  
 break;  
 }  
 temp = node[temp].fail;  
 }  
 if (!temp)  
 node[node[p].next[i]].fail = node[0].next[i];  
 }  
 que.push(node[p].next[i]);  
 }  
 }  
}  
   
void match(char str[]) {  
 int p = 0;  
 for (int i = 0; str[i]; i++) {  
 int ha = str[i] - 'A';  
 while (p && (!(ha >= 0 && ha<26) || !node[p].next[ha]))  
 p = node[p].fail;  
 if (ha >= 0 && ha<26 && node[p].next[ha])  
 p = node[p].next[ha];  
 int temp = p;  
 while (temp) {  
 if (node[temp].val)  
 ans[node[temp].val]++;  
 temp = node[temp].fail;  
 }  
 }  
}  
   
void print(int p) {  
 printf("%d:\n", p);  
 printf("val: %d\n", node[p].val);  
 printf("fail: %d\n", node[p].fail);  
 printf("edge:\n");  
 for (int i = 0; i < 26; i++) {  
 if (node[p].next[i]) {  
 printf("%c: %d\n", 'A'+i, node[p].next[i]);  
 }  
 }  
 printf("\n");  
 for (int i = 0; i < 26; i++) {  
 if (node[p].next[i]) {  
 print(node[p].next[i]);  
 }  
 }  
}  
   
int main() {  
 while (scanf("%d", &n) == 1) {  
 memset(ans, 0, sizeof ans);  
 memset(node, 0, sizeof node);  
 con = 1;  
 for (int i = 1; i <= n; i++) {  
 scanf("%s", mode[i]);  
 insertTrie(mode[i], i);  
 }  
 makeFail();  
 //print(0);  
 scanf("%s", a);  
 match(a);  
 for (int i = 1; i <= n; i++) {  
 if (ans[i]) {  
 printf("%s: %d\n", mode[i], ans[i]);  
 }  
 }  
 }  
 return 0;  
}

## 后缀自动机

/// hihoCoder 1445  
#include <cstdio>  
#include <algorithm>  
#include <cstring>  
using namespace std;  
   
const int MAXL = 1e6+10, MAXHA = 26;  
struct SuffixAutomaton {  
 struct {  
 int par, ch[MAXHA];  
 int maxlen;  
 } v[MAXL\*2];  
   
 int cos = 1, s = 1, last = 1;  
 int newState() {  
 return ++cos;  
 }  
 int push(int ha) {  
 int p = last;  
 const int np = newState();  
 last = np;  
 v[np].maxlen = v[p].maxlen + 1;  
 while (p && !v[p].ch[ha]) {  
 v[p].ch[ha] = np;  
 p = v[p].par;  
 }  
 if (!p) {  
 v[np].par = s;  
 return np;  
 }  
 const int q = v[p].ch[ha];  
 if (v[q].maxlen == v[p].maxlen + 1)  
 v[np].par = q;  
 else {  
 const int nq = newState();  
 v[nq].maxlen = v[p].maxlen + 1;  
 memcpy(v[nq].ch, v[q].ch, sizeof v[q].ch);  
 v[nq].par = v[q].par;  
 v[q].par = v[np].par = nq;  
 while (v[p].ch[ha] == q) {  
 v[p].ch[ha] = nq;  
 p = v[p].par;  
 }  
 }  
 return np;  
 }  
};  
   
char a[MAXL];  
SuffixAutomaton sa;  
   
int main() {  
 scanf("%s", a);  
 for (char\* p = a; \*p; ++p) {  
 sa.push(\*p - 'a');  
 }  
 long long ans = 0;  
 for (int i = 2; i <= sa.cos; i++) {  
 ans += sa.v[i].maxlen - sa.v[sa.v[i].par].maxlen;  
 }  
 printf("%lld\n", ans);  
 return 0;  
}

## 后缀数组

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
   
typedef long long LL;  
using namespace std;  
  
int n, m, sa[100005], x[100005], y[100005], c[100005], h[100005], rk[100005];  
char s[100005];  
  
void get\_sa(){  
 memset(c, 0, sizeof(c));  
 memset(x, 0, sizeof(x));  
 memset(y, 0, sizeof(y));  
 for (int i = 1; i <= n; i++) ++c[x[i] = s[i] - 'a' + 1];  
 for (int i = 2; i <= m; i++) c[i] += c[i - 1];  
 for (int i = n; i >= 1; i--) sa[c[x[i]]--] = i;  
 for (int k = 1; k <= n; k <<= 1){  
 int cnt = 0;  
 for (int i = n - k + 1; i <= n; i++) y[++cnt] = i;  
 for (int i = 1; i <= n; i++) if(sa[i] > k) y[++cnt] = sa[i] - k;  
 for (int i = 1; i <= m; i++) c[i] = 0;  
 for (int i = 1; i <= n; i++) ++c[x[i]];  
 for (int i = 2; i <= m; i++) c[i] += c[i - 1];  
 for (int i = n; i >= 1; i--) sa[c[x[y[i]]]--] = y[i], y[i] = 0;  
 swap(x, y);  
 x[sa[1]] = 1, cnt = 1;  
 for (int i = 2; i <= n; i++) x[sa[i]] = (y[sa[i]] == y[sa[i - 1]] && y[sa[i] + k] == y[sa[i - 1] + k] ? cnt : ++cnt);  
 if(cnt == n) break;  
 m = cnt;  
 }  
}  
  
void get\_h(){  
 int k = 0;  
 for (int i = 1; i <= n; i++) rk[sa[i]] = i;  
 for (int i = 1; i <= n; i++){  
 if(rk[i] == 1) continue;  
 if(k) k--;  
 int j = sa[rk[i] - 1];  
 while(i + k <= n && j + k <= n && s[i + k] == s[j + k]) k++;  
 h[rk[i]] = k;  
 }  
}  
  
int main()  
{  
 //freopen("18.in", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
   
 scanf("%s", s + 1);  
 n = strlen(s + 1), m = 26;  
 get\_sa(), get\_h();  
 //for (int i = 1; i <= n; i++) printf("%d%c", sa[i], i == n ? '\n' : ' ');  
 //for (int i = 1; i <= n; i++) printf("%d%c", h[i], i == n ? '\n' : ' ');  
 LL ans = 0;//求不重复的字串有多少种  
 for (int i = 1; i <= n; i++) ans += n - sa[i] + 1 - h[i];  
 printf("%lld\n", ans);  
   
 return 0;  
}  
/\*\*/

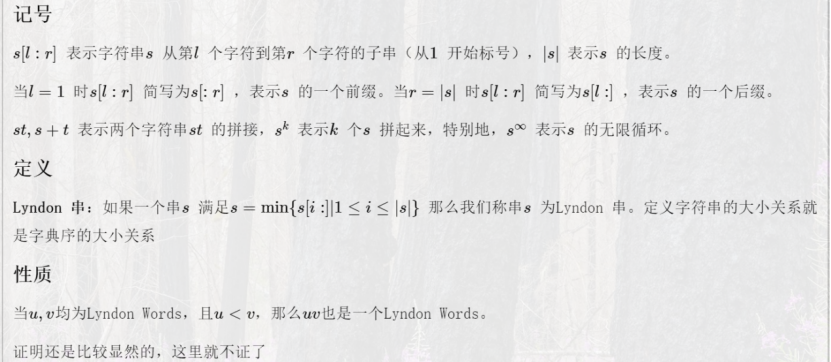
## 回文树

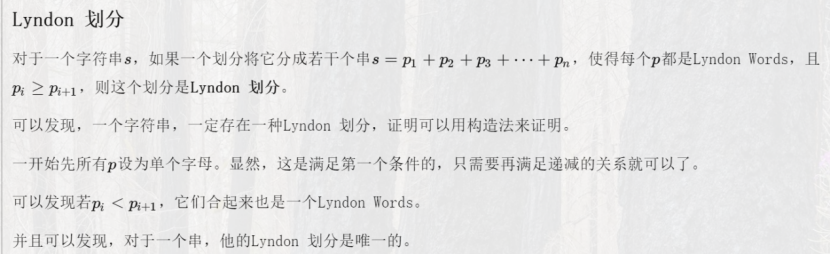
/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
const int N = 310005;  
  
char s[300005], str[300005 << 1];  
int num[300005], p[300005 << 1];  
  
struct Palindrome\_Automaton{//回文自动机  
 int len[N], next[N][26], fail[N], cnt[N], last, S[N], p, n, id[N];  
 int newnode(int l){//新建节点  
 for(int i = 0; i < 26; i++) next[p][i] = 0;//新建的节点为p，先消除它的子节点  
 cnt[p] = 0;  
 len[p] = l;  
 return p++;  
 }  
 inline void init(){//初始化  
 p = n = last = 0;  
 newnode(0);  
 newnode(-1);  
 S[0] = -1;  
 fail[0] = 1;  
 }  
 int get\_fail(int x){  
 while(S[n - len[x] - 1] != S[n]) x = fail[x];  
 return x;  
 }  
 inline void add(int c){//插字符  
 c -= 'a';  
 S[++n] = c;  
 int cur = get\_fail(last);  
 if(!next[cur][c]){  
 int now = newnode(len[cur] + 2);  
 fail[now] = next[get\_fail(fail[cur])][c];  
 next[cur][c] = now;  
 }  
 last = next[cur][c], id[last] = n;  
 cnt[last]++;  
 }  
 void count(){//统计本质相同的回文串的出现次数  
 for (int i = p - 1; i >= 0; i--){//逆序累加，保证每个点都会比它的父亲节点先算完，于是父亲节点能加到所有子孙  
 cnt[fail[i]] += cnt[i];  
 }  
 }  
}run;  
  
inline bool check(int L, int R) {  
 int mid = (L + R) >> 1;  
 if ((L & 1) == (R & 1)) return p[(mid << 1) + 2] >= R - L + 1;  
 return p[(mid << 1) + 3] >= R - L + 1;  
}  
  
void manacher(char s[], int len){  
 for (int i = 0; i < len; i++){  
 str[(i << 1) + 1] = '#';  
 str[(i << 1) + 2] = s[i];  
 }  
 len <<= 1;  
 str[0] = '?', str[len + 1] = '#', str[len + 2] = '\0';  
 int maxlen = 0, id = 0;  
 for (int i = 2; i < len + 1; i++){  
 if(p[id] + id > i) p[i] = min(p[(id << 1) - i], p[id] + id - i);  
 else p[i] = 1;  
 while(i - p[i] && i + p[i] < len + 2 && str[i + p[i]] == str[i - p[i]]) p[i]++;  
 if(id + p[id] < p[i] + i) id = i;  
 if(maxlen < p[i]) maxlen = p[i];  
 }  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 while(scanf("%s", s) == 1){  
 memset(num, 0, sizeof(num));  
 run.init();  
 int len = strlen(s);  
 for (int i = 0; i < len; i++) run.add(s[i]);  
 manacher(s, len);  
 run.count();  
 for (int i = 2; i < run.p; i++){  
 int t = run.id[i] - run.len[i] / 2 - 1;  
 if(check(run.id[i] - run.len[i], t)) num[run.len[i]] += run.cnt[i];  
 }  
 for (int i = 1; i <= len; i++) printf("%d%c", num[i], i == len ? '\n' : ' ');  
 }  
  
 return 0;  
}  
/\*\*/

## 序列自动机

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
typedef long long LL;  
using namespace std;  
char s[1000005], str[1005];  
int n, net[100005][105], p[105];  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 memset(p, -1, sizeof(p));  
 scanf("%s %d", s, &n);  
 int len = strlen(s);  
 for (int i = len - 1; i >= 0; i--){  
 for (int j = 0; j < 105; j++){  
 net[i][j] = p[j];  
 }  
 p[s[i] - ' '] = i;  
 }  
 getchar();  
 while(n--){  
 gets(str);  
 int t = p[str[0] - ' '];  
 if(t == -1) printf("No\n");  
 else {  
 len = strlen(str);  
 int f = 0;  
 for (int i = 1; i < len; i++){  
 t = net[t][fun(str[i])];  
 if(t == -1) {f = 1; break;}  
 }  
 if(!f) printf("YES\n");  
 else printf("NO\n");  
 }  
 }  
 return 0;  
}  
/\*\*/

## Lyndon分解





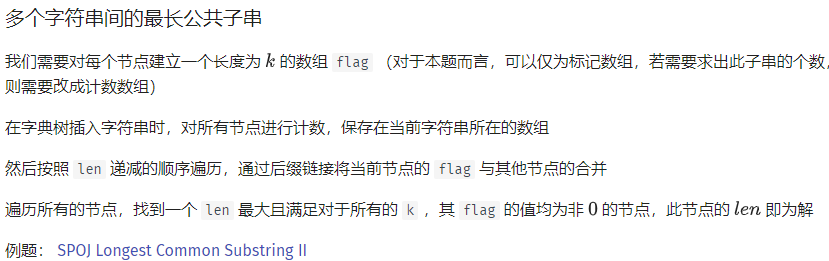
/\*  
hdu 6761  
[Lyndon分解]  
\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
const long long mod = 1e9 + 7;  
  
int t, n, ans[1000005];  
char s[1000005];  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 scanf("%d", &t);  
 while(t--){  
 scanf("%s", s + 1);  
 n = strlen(s + 1);  
 for (int i = 1; i <= n; ){  
 int j = i, k = i + 1;  
 ans[i] = i;  
 while(j <= n && s[j] <= s[k]){  
 if(s[j] < s[k]) ans[k] = ans[i], j = i;  
 else ans[k] = ans[j] + k - j, j++;  
 k++;  
 }  
 while(i <= j) i += k - j;  
 }  
 LL res = 0, p = 1;  
 for (int i = 1; i <= n; i++){  
 res = (res + ans[i] \* p % mod) % mod;  
 p = p \* 1112 % mod;  
 }  
 printf("%lld\n", res);  
 }  
  
 return 0;  
}  
/\*\*/

## 马拉车算法

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
char s[110005], str[220005];  
int p[220005];  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 while(scanf("%s", s) == 1){  
 int len = strlen(s);  
 for (int i = 0; i < len; i++){  
 str[(i << 1) + 1] = '#';  
 str[(i << 1) + 2] = s[i];  
 }  
 len <<= 1;  
 str[0] = '?', str[len + 1] = '#', str[len + 2] = '\0';  
 int maxlen = 0, id = 0;  
 for (int i = 2; i < len + 1; i++){  
 if(p[id] + id > i) p[i] = min(p[(id << 1) - i], p[id] + id - i);  
 else p[i] = 1;  
 while(str[i + p[i]] == str[i - p[i]]) p[i]++;  
 if(id + p[id] < p[i] + i) id = i;  
 if(maxlen < p[i]) maxlen = p[i];  
 }  
 for (int i = 2; i < len + 1; i++) printf("%c ", str[i]);  
 printf("\n");  
 for (int i = 2; i < len + 1; i++) printf("%d ", p[i]);  
 printf("\n");  
 printf("%d\n", maxlen - 1);  
 }  
  
 return 0;  
}  
/\*\*/

## 广义后缀自动机

所有字符中不同子串个数  
#include <bits/stdc++.h>  
using namespace std;  
#define MAXN 2000000 // 双倍字符串长度  
#define CHAR\_NUM 30 // 字符集个数，注意修改下方的 (-'a')  
struct exSAM {  
 int len[MAXN]; // 节点长度  
 int link[MAXN]; // 后缀链接，link  
 int next[MAXN][CHAR\_NUM]; // 转移  
 int tot; // 节点总数：[0, tot)  
 void init() {  
 tot = 1;  
 link[0] = -1;  
 }  
 int insertSAM(int last, int c) {  
 int cur = next[last][c];  
 if (len[cur]) return cur;  
 len[cur] = len[last] + 1;  
 int p = link[last];  
 while (p != -1) {  
 if (!next[p][c])  
 next[p][c] = cur;  
 else  
 break;  
 p = link[p];  
 }  
 if (p == -1) {  
 link[cur] = 0;  
 return cur;  
 }  
 int q = next[p][c];  
 if (len[p] + 1 == len[q]) {  
 link[cur] = q;  
 return cur;  
 }  
 int clone = tot++;  
 for (int i = 0; i < CHAR\_NUM; ++i)  
 next[clone][i] = len[next[q][i]] != 0 ? next[q][i] : 0;  
 len[clone] = len[p] + 1;  
 while (p != -1 && next[p][c] == q) {  
 next[p][c] = clone;  
 p = link[p];  
 }  
 link[clone] = link[q];  
 link[cur] = clone;  
 link[q] = clone;  
 return cur;  
 }  
 int insertTrie(int cur, int c) {  
 if (next[cur][c]) return next[cur][c];  
 return next[cur][c] = tot++;  
 }  
 void insert(const string &s) {  
 int root = 0;  
 for (auto ch : s) root = insertTrie(root, ch - 'a');  
 }  
 void insert(const char \*s, int n) {  
 int root = 0;  
 for (int i = 0; i < n; ++i) root = insertTrie(root, s[i] - 'a');  
 }  
 void build() {  
 queue<pair<int, int>> q;  
 for (int i = 0; i < 26; ++i)  
 if (next[0][i]) q.push({i, 0});  
 while (!q.empty()) {  
 auto item = q.front();  
 q.pop();  
 auto last = insertSAM(item.second, item.first);  
 for (int i = 0; i < 26; ++i)  
 if (next[last][i]) q.push({i, last});  
 }  
 }  
} exSam;  
char s[1000100];  
int main() {  
 int n;  
 cin >> n;  
 exSam.init();  
 for (int i = 0; i < n; ++i) {  
 cin >> s;  
 int len = strlen(s);  
 exSam.insert(s, len);  
 }  
 exSam.build();  
 long long ans = 0;  
 for (int i = 1; i < exSam.tot; ++i) {  
 ans += exSam.len[i] - exSam.len[exSam.link[i]];  
 }  
 cout << ans << endl;  
}



#include <bits/stdc++.h>  
using namespace std;  
#define MAXN 2000000 // 双倍字符串长度  
#define CHAR\_NUM 30 // 字符集个数，注意修改下方的 (-'a')  
#define NUM 15 // 字符串个数  
struct exSAM {  
 int len[MAXN]; // 节点长度  
 int link[MAXN]; // 后缀链接，link  
 int next[MAXN][CHAR\_NUM]; // 转移  
 int tot; // 节点总数：[0, tot)  
 int lenSorted[MAXN]; // 按照 len 排序后的数组，仅排序 [1, tot)  
 // 部分，最终下标范围 [0, tot - 1)  
 int sizeC[MAXN][NUM]; // 表示某个字符串的子串个数  
 int curString; // 字符串实际个数  
 /\*\*  
 \* 计数排序使用的辅助空间数组  
 \*/  
 int lc[MAXN]; // 统计个数  
 void init() {  
 tot = 1;  
 link[0] = -1;  
 }  
 int insertSAM(int last, int c) {  
 int cur = next[last][c];  
 len[cur] = len[last] + 1;  
 int p = link[last];  
 while (p != -1) {  
 if (!next[p][c])  
 next[p][c] = cur;  
 else  
 break;  
 p = link[p];  
 }  
 if (p == -1) {  
 link[cur] = 0;  
 return cur;  
 }  
 int q = next[p][c];  
 if (len[p] + 1 == len[q]) {  
 link[cur] = q;  
 return cur;  
 }  
 int clone = tot++;  
 for (int i = 0; i < CHAR\_NUM; ++i)  
 next[clone][i] = len[next[q][i]] != 0 ? next[q][i] : 0;  
 len[clone] = len[p] + 1;  
 while (p != -1 && next[p][c] == q) {  
 next[p][c] = clone;  
 p = link[p];  
 }  
 link[clone] = link[q];  
 link[cur] = clone;  
 link[q] = clone;  
 return cur;  
 }  
 int insertTrie(int cur, int c) {  
 if (!next[cur][c]) next[cur][c] = tot++;  
 sizeC[next[cur][c]][curString]++;  
 return next[cur][c];  
 }  
 void insert(const string &s) {  
 int root = 0;  
 for (auto ch : s) root = insertTrie(root, ch - 'a');  
 curString++;  
 }  
 void insert(const char \*s, int n) {  
 int root = 0;  
 for (int i = 0; i < n; ++i) root = insertTrie(root, s[i] - 'a');  
 curString++;  
 }  
 void build() {  
 queue<pair<int, int>> q;  
 for (int i = 0; i < 26; ++i)  
 if (next[0][i]) q.push({i, 0});  
 while (!q.empty()) {  
 auto item = q.front();  
 q.pop();  
 auto last = insertSAM(item.second, item.first);  
 for (int i = 0; i < 26; ++i)  
 if (next[last][i]) q.push({i, last});  
 }  
 }  
 void sortLen() {  
 for (int i = 1; i < tot; ++i) lc[i] = 0;  
 for (int i = 1; i < tot; ++i) lc[len[i]]++;  
 for (int i = 2; i < tot; ++i) lc[i] += lc[i - 1];  
 for (int i = 1; i < tot; ++i) lenSorted[--lc[len[i]]] = i;  
 }  
 void getSizeLen() {  
 for (int i = tot - 2; i >= 0; --i)  
 for (int j = 0; j < curString; ++j)  
 sizeC[link[lenSorted[i]]][j] += sizeC[lenSorted[i]][j];  
 }  
 void debug() {  
 cout << " i len link ";  
 for (int i = 0; i < 26; ++i) cout << " " << (char)('a' + i);  
 cout << endl;  
 for (int i = 0; i < tot; ++i) {  
 cout << "i: " << setw(3) << i << " len: " << setw(3) << len[i]  
 << " link: " << setw(3) << link[i] << " Next: ";  
 for (int j = 0; j < CHAR\_NUM; ++j) {  
 cout << setw(3) << next[i][j];  
 }  
 cout << endl;  
 }  
 }  
} exSam;  
int main() {  
 exSam.init();  
 string s;  
 while (cin >> s) exSam.insert(s);  
 exSam.build();  
 exSam.sortLen();  
 exSam.getSizeLen();  
 int ans = 0;  
 for (int i = 0; i < exSam.tot; ++i) {  
 bool flag = true;  
 for (int j = 0; j < exSam.curString; ++j) {  
 if (!exSam.sizeC[i][j]) {  
 flag = false;  
 break;  
 }  
 }  
 if (flag) ans = max(ans, exSam.len[i]);  
 }  
 cout << ans << endl;  
}

# 数论

## 逆元

### 扩展欧几里得

要求a与mod互质

// ax+by=gcd(a,b)  
LL exgcd(LL a, LL b, LL& x, LL& y) {  
 if(b==0) {  
 x=1; y=0;  
 return a;  
 }  
 else {  
 LL g = exgcd(b, a%b, y, x);  
 y-=a/b\*x;  
 return g;  
 }  
}  
LL inverse(LL a) {  
 LL x,y;  
 exgcd(a, MOD, x, y);  
 return (x+MOD)%MOD;  
}

### 费马小定理

要求mod为质数

fastPow(a, MOD-2);

### 线性递推

要求mod为质数

LL inv[MAXN];  
void makeInv() {  
 inv[1] = 1;  
 for(int i=2; i<MAXN; i++) {  
 inv[i] = inv[MOD%i]\*(MOD-MOD/i)%MOD;  
 }  
}

### 线性求阶乘的逆元

LL fac[MAXN], invfac[MAXN];  
void makeFac() {  
 fac[0] = 1;  
 for(int i=1; i< MAXN; i++) {  
 fac[i] = fac[i-1]\*i%MOD;  
 }  
 invfac[MAXN-1] = fastPow(fac[MAXN-1], MOD-2); // 使用费马小定理要求MOD是质数  
 for(int i=MAXN-2; i>=0; i--) {  
 invfac[i] = invfac[i+1]\*(i+1)%MOD;  
 }  
}

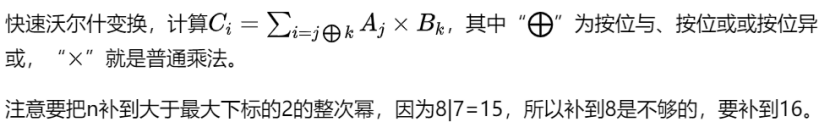
## 线性筛同时求质数、欧拉函数、莫比乌斯函数、莫比乌斯函数前缀和

int np;  
LL prime[MAXN], phi[MAXN], mu[MAXN], smu[MAXN];  
bool notp[MAXN];  
   
void linearSieve() {  
 mu[1] = smu[1] = phi[1] = 1;  
 for (int i = 2; i < MAXN; i++) {  
 if (!notp[i]) {  
 prime[np++] = i;  
 mu[i] = -1;  
 phi[i] = i-1;  
 }  
 for (int j = 0; j < np; j++) {  
 if(i\*prime[j]>=MAXN) break;  
 notp[i\*prime[j]] = true;  
 if (i%prime[j] == 0) {  
 mu[i\*prime[j]] = 0;  
 phi[i\*prime[j]] = phi[i]\*prime[j];  
 break;  
 }  
 else {  
 mu[i\*prime[j]] = -mu[i];  
 phi[i\*prime[j]] = phi[i]\*(prime[j]-1);  
 }  
 }  
 smu[i] = smu[i-1]+mu[i];  
 }  
}

## 1 <= x <= n, 1 <= y <= m, gcd(x, y) = k的个数

LL nGcdEqK(LL n, LL m, LL k) {  
 n/=k; m/=k;  
 LL last, rst = 0;  
 for (LL i = 1; i <= n && i <= m; i = last + 1) {  
 last = min(n/(n/i), m/(m/i));  
 rst += (n/i)\*(m/i)%p\*(smu[last]-smu[i-1])%p;  
 rst %= p;  
 }  
 return rst;  
}

## FWT



void FWT(LL a[],int n)  
{  
 for(int d=1;d<n;d<<=1)  
 for(int m=d<<1,i=0;i<n;i+=m)  
 for(int j=0;j<d;j++)  
 {  
 LL x=a[i+j],y=a[i+j+d];  
 //xor:a[i+j]=x+y,a[i+j+d]=x-y;  
 //and:a[i+j]=x+y;  
 //or:a[i+j+d]=x+y;  
 }  
}  
void UFWT(LL a[],int n)  
{  
 for(int d=1;d<n;d<<=1)  
 for(int m=d<<1,i=0;i<n;i+=m)  
 for(int j=0;j<d;j++)  
 {  
 LL x=a[i+j],y=a[i+j+d];  
 //xor:a[i+j]=(x+y)/2,a[i+j+d]=(x-y)/2;  
 //and:a[i+j]=x-y;  
 //or:a[i+j+d]=y-x;  
 }  
}  
void solve(LL a[], LL b[], int n)  
{  
 FWT(a,n);  
 FWT(b,n);  
 for(int i=0;i<n;i++) a[i]=a[i]\*b[i];  
 UFWT(a,n);  
}

## FFT

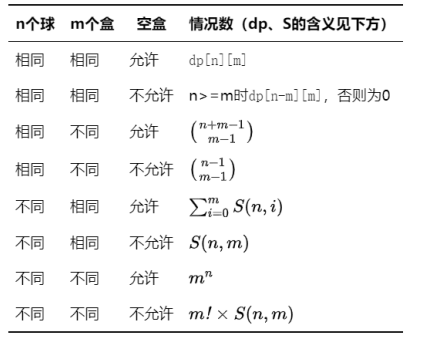


#include <cstdio>  
#include <algorithm>  
#include <cstring>  
#include <cmath>  
#include <complex>  
using namespace std;  
   
typedef complex<double> Complex;  
const double PI = acos(-1);  
const int MAXN = (1<<20)+10;  
int N; // 大于等于结果项数(n1+n2-1)的2的整次幂，init会赋值  
Complex omega[MAXN], omegaInv[MAXN];  
int n1, n2;  
double a[MAXN], b[MAXN], c[MAXN];  
   
void init(int n1, int n2) {  
 for(N=1; N<(n1+n2-1); N<<=1);  
 for(int i=0; i<N; i++) {  
 omega[i] = Complex(cos(2\*PI\*i/N), sin(2\*PI\*i/N));  
 omegaInv[i] = conj(omega[i]);  
 }  
}  
   
void dft(Complex a[], Complex ome[] = omega) {  
 for(int i=0,k=log2(N); i<N; i++) {  
 int t = 0;  
 for(int j=0; j<k; j++) {  
 if(i&(1<<j))  
 t|=1<<(k-j-1);  
 }  
 if(i<t) swap(a[i], a[t]);  
 }  
 for(int l=2; l<=N; l<<=1) {  
 int m = l>>1;  
 for(Complex\* p=a; p<a+N; p+=l) {  
 for(int i=0; i<m; i++) {  
 Complex t = ome[N/l\*i]\*p[m+i];  
 p[m+i]=p[i]-t;  
 p[i]+=t;  
 }  
 }  
 }  
}  
   
void idft(Complex a[]) {  
 dft(a, omegaInv);  
 for(int i=0; i<N; i++)  
 a[i]/=N;  
}  
   
// 多项式相乘，f(x)=a[i]\*x^i, g(x)=b[i]\*x^i, i=0..n-1，n1, n2是项数  
void multiply(const double a[], int n1, const double b[], int n2, double rst[]) {  
 static Complex ia[MAXN], ib[MAXN];  
 init(n1, n2);  
 copy(a, a+n1, ia);  
 fill(ia+n1, ia+N, Complex());  
 copy(b, b+n2, ib);  
 fill(ib+n2, ib+N, Complex());  
 dft(ia); dft(ib);  
 for(int i=0; i<N; i++)  
 ia[i]\*=ib[i];  
 idft(ia);  
 for(int i=0; i<n1+n2-1; i++)  
 rst[i] = ia[i].real();  
}  
   
int main() {  
 while(scanf("%d%d", &n1, &n2)==2) {  
 for(int i=0; i<n1; i++) {  
 scanf("%lf", a+i);  
 }  
 for(int i=0; i<n2; i++) {  
 scanf("%lf", b+i);  
 }  
 multiply(a, n1, b, n2, c);  
 for(int i=0; i<n1+n2-1; i++) {  
 printf("%f%c", c[i], " \n"[i==n1+n2-2]);  
 }  
 }  
 return 0;  
}

## 数值积分

const double EPS = 1e-10;  
double F(double x) {  
 return x\*x\*x+2\*x\*x+0.4\*x+3.332;  
}  
// 阶数为4或5用科特斯公式  
double cotes(double a, double b) {  
 double d=(a+b)/2, c=(a+d)/2, e=(d+b)/2;  
 return (7\*F(a)+32\*F(c)+12\*F(d)+32\*F(e)+7\*F(b))\*(b-a)/90;  
}  
// 阶数小于等于3用辛普森公式  
double simpson(double a, double b) {  
 double c = (a+b)/2;  
 return (F(a)+4\*F(c)+F(b))\*(b-a)/6;  
}  
double asr(double a, double b, double eps, double A) {  
 double c = (a+b)/2;  
 double L = simpson(a, c), R = simpson(c, b);  
 //double L = cotes(a, c), R = cotes(c, b);  
 if(abs(L+R-A) <= 15\*eps) return L+R+(L+R-A)/15;  
 return asr(a, c, eps/2, L) + asr(c, b, eps/2, R);  
}  
// 调用这个  
double asr(double a, double b, double eps){  
 return asr(a, b, eps, simpson(a,b));  
 //return asr(a, b, eps, cotes(a,b));  
}

## 球盒问题



球、盒均相同的DP方法

OEIS A026820  
dp[i][j]的含义是，把非负整数i划分为j个非负整数之和的方案数。  
dp[0][i]=1, i>=0 // 0个球放在i个盒子中  
dp[1][i]=1, i>=1 // 1个球放在i个盒子中  
dp[i][1]=1, i>=0 // i个球放在1个盒子中  
dp[i][j]=dp[i][j-1]+dp[i-j][j], i>=2, j<=i // i个球放在j-1个盒子中；或i-j个球放在j个盒子中，再给每个盒子里放一个球  
dp[i][j]=dp[i][j-1], i>=2, j>i // i个球放在j-1个盒子中  
其余情况为0  
LL dp[MAXN][MAXN];  
void init() {  
 for(int i=0; i<MAXN; i++)  
 dp[0][i] = 1;  
 for(int i=1; i<MAXN; i++)  
 dp[1][i] = 1;  
 for(int i=0; i<MAXN; i++)  
 dp[i][1] = 1;  
 for(int i=2; i<MAXN; i++) {  
 for(int j=2; j<MAXN; j++) {  
 dp[i][j] = dp[i][j-1];  
 if(i>j) dp[i][j] += dp[i-j][j];  
 }  
 }  
}  
1  
1 2  
1 2 3  
1 3 4 5  
1 3 5 6 7  
1 4 7 9 10 11  
1 4 8 11 13 14 15  
1 5 10 15 18 20 21 22  
1 5 12 18 23 26 28 29 30  
1 6 14 23 30 35 38 40 41 42  
1 6 16 27 37 44 49 52 54 55 56  
1 7 19 34 47 58 65 70 73 75 76 77

## 第二类斯特林数

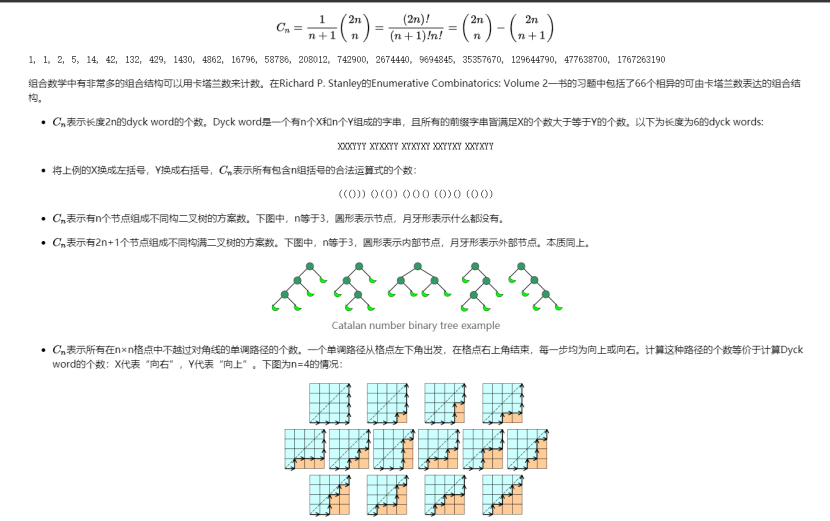
OEIS A008277  
S[i][j]的含义是，把大小为i的集合划分为j个非空子集合的方案数。  
S[i][i]=1, i>=0 // i个球放入i个盒子  
S[i][j]=S[i-1][j]\*j+S[i-1][j-1], i>=2, 1<=j<=i-1 // 前i-1个球在j个盒子中，第i个球随意放；或前i-1个球在j-1个盒子中，第i个球只能放在第j个盒子中  
其余情况为0  
LL S[MAXN][MAXN];  
void init() {  
 for(int i=0; i<MAXN; i++) {  
 S[i][i] = 1;  
 }  
 for(int i=2; i<MAXN; i++) {  
 for(int j=1; j<i; j++) {  
 S[i][j]=S[i-1][j]\*j+S[i-1][j-1];  
 }  
 }  
}  
1  
1 1  
1 3 1  
1 7 6 1  
1 15 25 10 1  
1 31 90 65 15 1  
1 63 301 350 140 21 1  
1 127 966 1701 1050 266 28 1  
1 255 3025 7770 6951 2646 462 36 1  
1 511 9330 34105 42525 22827 5880 750 45 1  
1 1023 28501 145750 246730 179487 63987 11880 1155 55 1

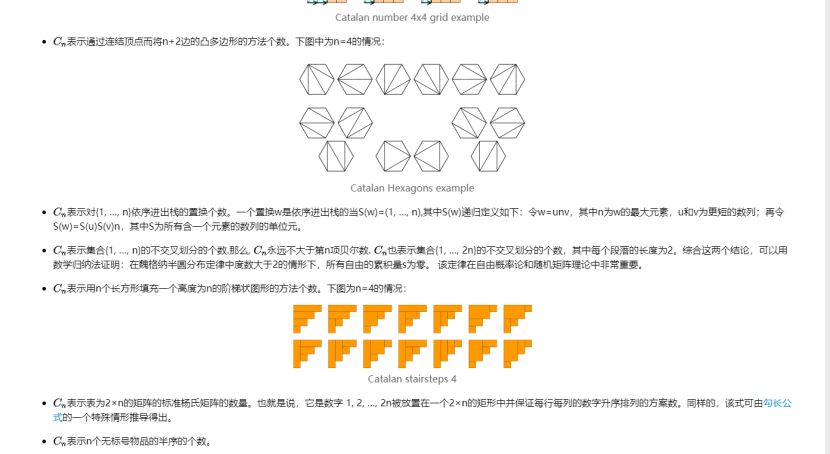
## 第一类斯特林数

S[i][j]的含义是，把i个物品排成j个非空循环排列的方案数。  
  
S[0][0]=1  
S[i][j]=(i-1)\*S[i-1][j]+S[i-1][j-1], i>=1, j>=1  
其余为0  
LL S[MAXN][MAXN];  
void init() {  
 S[0][0] = 1;  
 for(int i=1; i<MAXN; i++) {  
 for(int j=1; j<=i; j++) {  
 S[i][j] = (i-1)\*S[i-1][j]+S[i-1][j-1];  
 }  
 }  
}  
1  
1 1  
2 3 1  
6 11 6 1  
24 50 35 10 1  
120 274 225 85 15 1  
720 1764 1624 735 175 21 1  
5040 13068 13132 6769 1960 322 28 1  
40320 109584 118124 67284 22449 4536 546 36 1  
362880 1026576 1172700 723680 269325 63273 9450 870 45 1

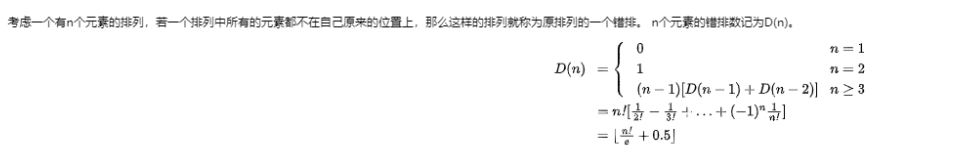
## Catalan数

OEIS A000108

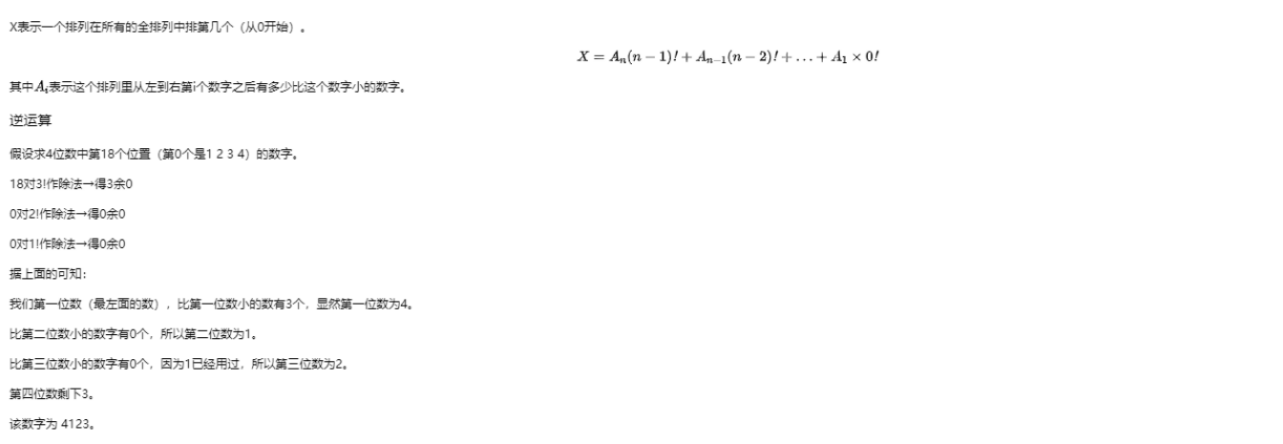




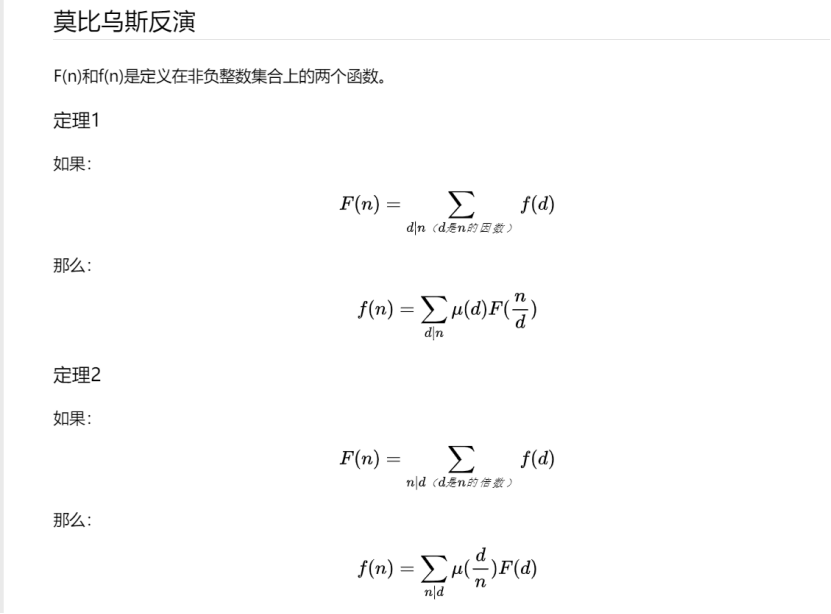
## 错排



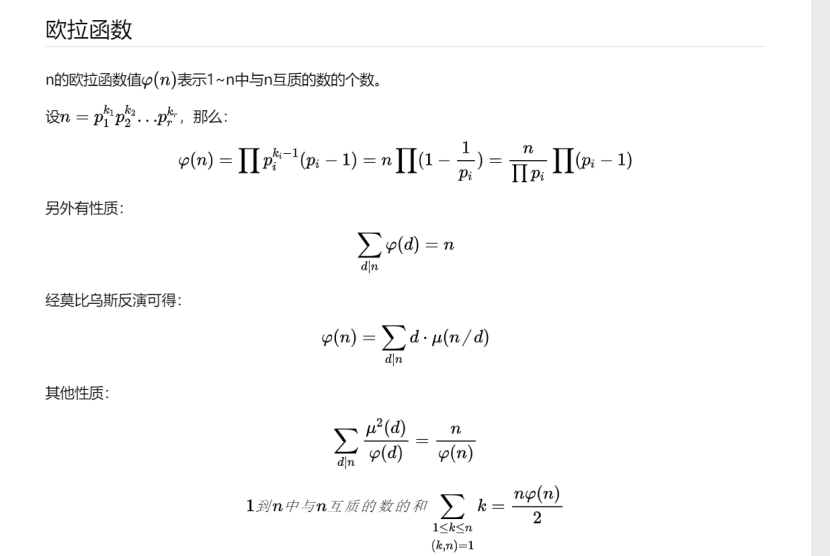
## 康拓展开



## 莫比乌斯反演

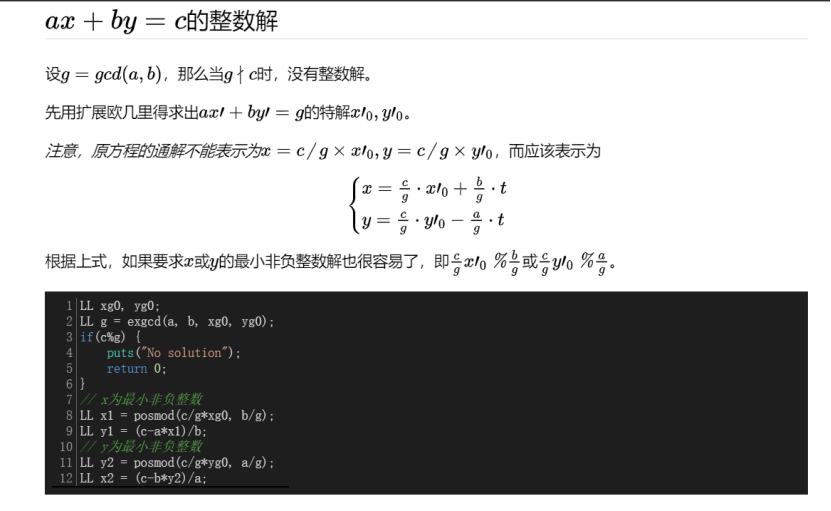


## 欧拉函数

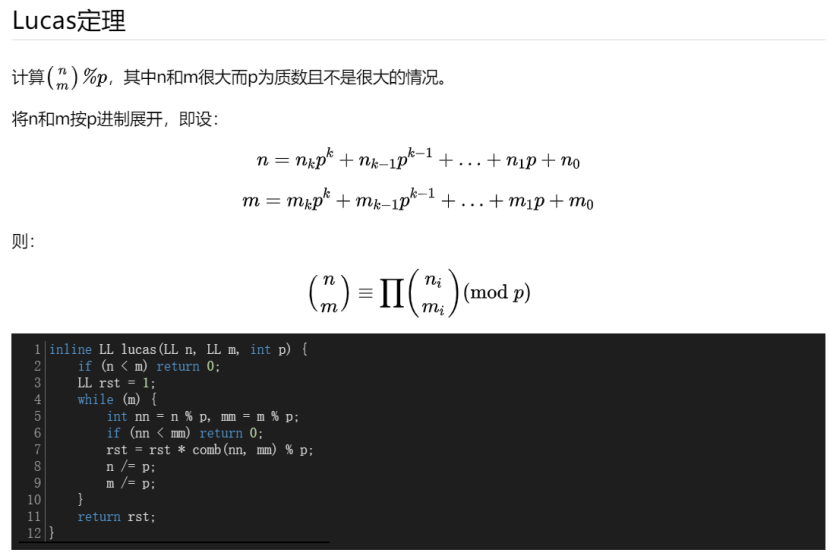


LL phi(LL n) {  
 LL rst = n;  
 for (int i = 2; (LL)i\*i <= n; i++) {  
 if (n%i == 0) {  
 rst -= rst / i;  
 do n /= i; while (n%i == 0);  
 }  
 }  
 if (n > 1)  
 rst -= rst / n;  
 return rst;  
}

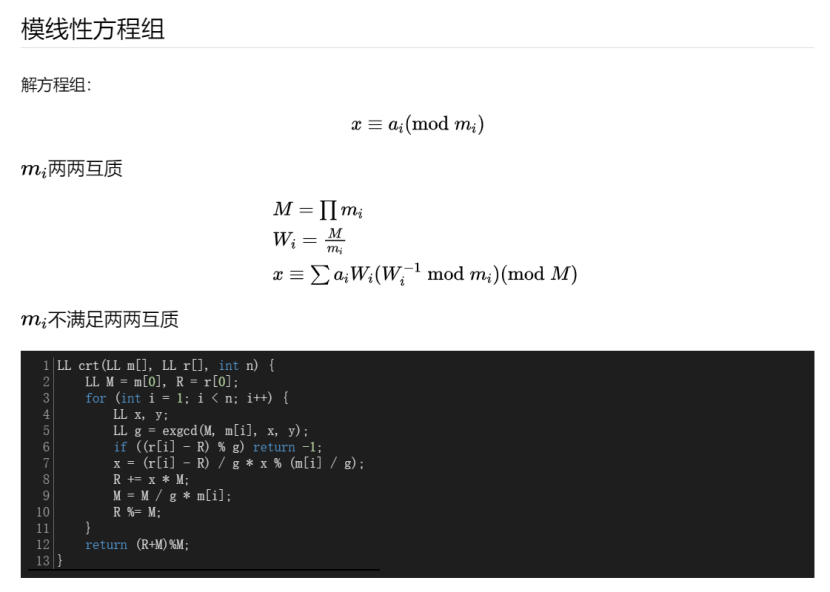
## ax + by = c的整数解



## Lucas定理

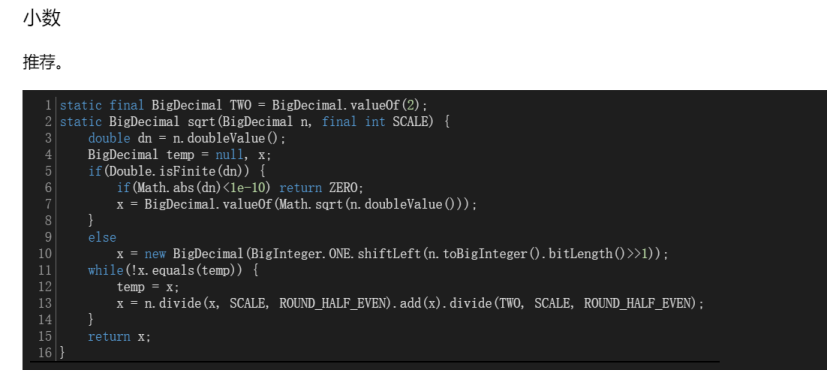


## 模线性方程组



## Heron法开方





## 平方剩余



## 约瑟夫环（编号为0~n-1）

f(1)=0  
f(i)=(f(i-1)+k)%i  
  
LL josephus(LL n, LL k) {  
 if (k == 1) return n - 1;  
 LL ans = 0;  
 for (LL i = 2; i <= n;) {  
 if (ans + k >= i) {  
 ans = (ans + k) % i;  
 i++;  
 continue;  
 }  
 LL step = (i - ans - 2) / (k - 1);  
 if (i + step > n) {  
 ans += (n - (i - 1))\*k;  
 break;  
 }  
 i += step; ans += step \* k;  
 }  
 return ans%n;  
}

## Miller-Rabin素性测试

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
#include <time.h>  
#include <random>  
  
typedef long long LL;  
using namespace std;  
  
int t, n;  
LL x, y, a[100005];  
mt19937 rd(time(0));  
  
LL ksm(LL a, LL b, LL mod){  
 return (a \* b - (LL)((long double)a / mod \* b) \* mod + mod) % mod;  
}  
  
LL poww(LL x, LL num, LL mod){  
 LL res = 1;  
 x %= mod;  
 while(num){  
 if(num & 1) res = ksm(res, x, mod);  
 x = ksm(x, x, mod);  
 num >>= 1;  
 }  
 return res;  
}  
  
struct Mill{  
   
 LL n, fac[22000][2], bk[22000]; int tot;  
  
 const int C = 2307;  
 const int S = 8;  
  
 bool check(LL a, LL n){  
 LL m = n - 1, x, y = 0;  
 int j = 0;  
 while (!(m & 1)) m >>= 1, j++;  
 x = poww(a, m, n);  
 for (int i = 1; i <= j; x = y, i++){  
 y = ksm(x, x, n);  
 if (y == 1 && x != 1 && x != n - 1) return 1;  
 }  
 return y != 1;  
 }  
  
 bool miller\_rabin(LL n){  
 if(n < 2){  
 return 0;  
 }else if (n == 2){  
 return 1;  
 }else if (!(n & 1)){  
 return 0;  
 }  
 for (int i = 0; i < S; ++i){  
 if (check(rd() % (n - 1) + 1, n)) return 0;  
 }  
 return 1;  
 }  
  
 LL pollard\_rho(LL n, int c){  
 LL i = 1, k = 2, x = rd() % n, y = x, d;  
 while (1) {  
 ++i; x = (ksm(x, x, n) + c) % n;  
 d = \_\_gcd(y - x, n);  
 if(d > 1 && d < n){  
 return d;  
 }  
 if(y == x){  
 return n;  
 }  
 if(i == k){  
 y = x;  
 k <<= 1;  
 }  
 }  
 }  
  
 void findfac(LL n, int c){  
 if(n == 1){  
 return ;  
 }  
 if(miller\_rabin(n)){  
 bk[++\*bk] = n;  
 return ;  
 }  
 LL m = n;  
 while(m == n){  
 m = pollard\_rho(n, c--);  
 }  
 findfac(m, c);  
 findfac(n / m, c);  
 }  
 void gao(LL \_n){  
 n = \_n; \*bk = 0;  
 findfac(n, C);  
 sort(bk + 1, bk + 1 + \*bk);  
 fac[1][0] = bk[1];  
 fac[1][1] = 1;  
 tot = 1;  
 for (int i = 2; i <= \*bk; i++){  
 if (bk[i] == bk[i - 1]) {  
 ++fac[tot][1];  
 }else{  
 ++tot;  
 fac[tot][0] = bk[i];  
 fac[tot][1] = 1;  
 }  
 }  
 }  
  
}mill;  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 scanf("%d", &t);  
 while(t--){  
 scanf("%d %lld %lld", &n, &x, &y);  
 for (int i = 1; i <= n; i++) scanf("%lld", &a[i]);  
 mill.gao(x);  
 LL ans = 1LL << 60;  
 for (int i = 1; i <= mill.tot; i++){  
 LL nu = 0, tp, w;  
 // printf("%lld %lld\n", fac[i], num[i]);  
 for (int j = 1; j <= n; j++){  
 tp = log(a[j]) / log(mill.fac[i][0]);  
 w = 1;  
 for (int k = 1; k <= tp; k++) w \*= mill.fac[i][0], nu += a[j] / w;  
 }  
 w = 1;  
 tp = log(y) / log(mill.fac[i][0]);  
 for (int k = 1; k <= tp; k++) w \*= mill.fac[i][0], nu -= y / w;  
 // printf("%lld %lld\n", fac[i], tp);  
 nu = max(0LL, -nu);  
 ans = min(ans, nu / mill.fac[i][1]);  
 }  
 printf("%lld\n", ans);  
 }  
  
 return 0;  
}  
/\*\*/

## O(1) gcd

#include <cstdio>  
#include <cstring>  
#define SIZE 1000  
#define U unsigned int  
using namespace std;  
   
 int ss[1000001],cnt,p[1000001];  
 short g[1001][1001],b[1000001];  
 int f[1000001][3];  
 int a[2001],bb[2001];  
  
 int pregcd(int x,int y){  
 if (y==0) {g[x][y]=x;return(x);}  
 if (g[x][y]!=-1) return(g[x][y]);  
 g[x][y]=pregcd(y,x%y);  
 return(g[x][y]);  
 }  
   
 void euler(){  
 p[1]=1;  
 for (int i=2;i<=SIZE\*SIZE;i++){  
 if (!b[i]){  
 p[i]=i;ss[++cnt]=i;  
 }   
 for (int j=1;ss[j]\*i<=SIZE\*SIZE&&j<=cnt;j++){  
 b[ss[j]\*i]=1;  
 p[ss[j]\*i]=ss[j];  
 if (i%ss[j]==0) break;  
 }  
 }  
 }  
   
 int gcd(int x,int y){  
 if (x<=SIZE&&y<=SIZE) return(g[x][y]);  
 int d=1;  
 for (int i=0;i<=2;i++)  
 if (p[f[x][i]]==f[x][i]){  
 if (y%f[x][i]==0){  
 d\*=f[x][i];y/=f[x][i];  
 }  
 }else{  
 int t=g[f[x][i]][y%f[x][i]];  
 d\*=t;y/=t;  
 }  
 return(d);   
 }  
   
 int main(){  
 freopen("a.in","r",stdin);  
   
 for (int i=0;i<=SIZE;i++)  
 for (int j=0;j<=SIZE;j++)  
 g[i][j]=-1;  
 for (int i=0;i<=SIZE;i++)  
 for (int j=0;j<=SIZE;j++)  
 pregcd(i,j);//预处理SIZE以内gcd  
   
 euler();  
   
 f[1][0]=f[1][1]=f[1][2]=1;  
 for (int i=2;i<=SIZE\*SIZE;i++){  
 memcpy(f[i],f[i/p[i]],sizeof(f[i]));   
 if (f[i][0]<=SIZE/p[i]) f[i][0]\*=p[i];else  
 if (f[i][1]<=SIZE/p[i]) f[i][1]\*=p[i];else  
 f[i][2]\*=p[i];  
 }//预处理所有数的分解  
   
 int T,n,m;  
 scanf("%d",&T);  
 while (T--){  
 scanf("%d%d",&n,&m);  
 U ans=0;  
 for (int i=0;i<n;i++) scanf("%d",&a[i]);  
 for (int i=0;i<m;i++) scanf("%d",&bb[i]);  
 for (int i=0;i<n;i++)  
 for (int j=0;j<m;j++)  
 ans+=gcd(a[i],bb[j])^i^j;  
 printf("%u\n",ans);  
 }  
 }

## 费马平方和定理

/\*奇质数能表示为两个平方数之和的充分必要条件是该质数被 4 除余 1，即 4∗n+1\*/  
  
/\*a ^ 2 + b ^ 2 = p \* q, p 和 q都是素数 2 <= p <= 1e9\*/  
  
// interesting  
#include <bits/stdc++.h>  
using namespace std;  
int T;  
int p, q;  
map<int, int> cnt;  
int main() {  
 scanf("%d", &T);  
 while (T--) {  
 scanf("%d%d", &p, &q);  
 cnt[p]++;  
 cnt[q]++;  
 int ans = 1;  
 for (auto x : cnt) {  
 int i = x.first;  
 if (cnt[i]) {  
 if (i % 4 == 1) {  
 ans \*= cnt[i] + 1; // 分解为一对共轭整数 (a + bi) \* (a - bi), (c + di) \* (c - di) 第一项和第三项乘，第二项和第四项乘，换位置，有两种，如果cnt[i] == 2,那么有A = 0  
 } else if (i % 4 == 3 && (cnt[i] & 1))  
 ans = 0; // 高斯素数无法分配  
 }  
 }  
 printf("%d\n", 4 \* ans); // 四个区间  
 cnt.clear();  
 }  
 return 0;  
}

## 拉格朗日插值法模板

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
const int D = 1e3 + 10;  
const long long mod = 9999991;  
  
LL a[D], f[D], g[D], p[D], p1[D], p2[D], b[D], h[D][2], C[D];  
  
LL poww(LL x, LL num){  
 LL res = 1;  
 x %= mod;  
 while(num){  
 if(num & 1) res = res \* x % mod;  
 x = x \* x % mod;  
 num >>= 1;  
 }  
 return res;  
}  
// 已知a\_i 的d次多项式，求第n项  
LL calcn(int d, LL \*a, LL n){// a[0].. a[d] a[n]  
 if(n <= d) return a[n];  
 p1[0] = p2[0] = 1;  
 for (int i = 0; i <= d; i++){  
 LL t = (n - i + mod) % mod;  
 p1[i + 1] = p1[i] \* t % mod;  
 }  
 for (int i = 0; i <= d; i++){  
 LL t = (n - d + i + mod) % mod;  
 p2[i + 1] = p2[i] \* t % mod;  
 }  
 LL ans = 0;  
 for (int i = 0; i <= d; i++){  
 LL t = g[i] \* g[d - i] % mod \* p1[i] % mod \* p2[d - i] % mod \* a[i] % mod;  
 if((d - i) & 1) ans = (ans - t + mod) % mod;  
 else ans = (ans + t) % mod;  
 }  
 return ans;  
}  
  
// 初始化，初始化的时候记得将D的值  
void init(int M){  
 f[0] = f[1] = g[0] = g[1] = 1;  
 for (int i = 2; i <= M + 4; i++) f[i] = f[i - 1] \* i % mod;  
 g[M + 4] = poww(f[M + 4], mod - 2);  
 for (int i = M + 3; i >= 1; i--) g[i] = g[i + 1] \* (i + 1) % mod;  
}  
  
// 已知a\_i，并且知道a\_i是m次多项式  
LL polysum(LL m, LL \*a, LL n){// a[0].. a[m] \sum\_{i=0}^{n} a[i]  
 if(n <= m){  
 LL sum = 0;  
 for (int i = 0; i <= n; i++) sum = (sum + a[i]) % mod;  
 return sum;  
 }  
 LL b[D];  
 for (int i = 0; i <= m; i++) b[i] = a[i];  
 b[m + 1] = calcn(m, b, m + 1);  
 for (int i = 1; i <= m + 1; i++) b[i] = (b[i - 1] + b[i]) % mod;  
 return calcn(m + 1, b, n);// m次多项式的和是m+1 次多项式  
}  
  
LL qpolysum(LL R, LL n, LL \*a, LL m){  
 // a[0].. a[m] \sum\_{i=0}^{n-1} a[i]\*R^i  
 if(R == 1) return polysum(n, a, m);  
 a[m + 1] = calcn(m, a, m + 1);  
 LL r = poww(R, mod - 2), p3 = 0, p4 = 0, c, ans;  
 h[0][0] = 0, h[0][1] = 1;  
 for (int i = 1; i <= m + 1; i++){  
 h[i][0] = (h[i - 1][0] + a[i - 1]) \* r % mod;  
 h[i][1] = h[i - 1][1] \* r % mod;  
 }  
 for (int i = 0; i <= m + 1; i++){  
 LL t = g[i] \* g[m + 1 - i] % mod;  
 if(i & 1) p3 = ((p3 - h[i][0] \* t) % mod + mod) % mod, p4 = ((p4 - h[i][1] \* t) % mod + mod) % mod;  
 else p3 = (p3 + h[i][0] \* t) % mod, p4 = (p4 + h[i][1] \* t) % mod;  
 }  
 c = poww(p4, mod - 2) \* (mod - p3) % mod;  
 for (int i = 0; i <= m + 1; i++) h[i][0] = (h[i][0] + h[i][1] \* c) % mod;  
 for (int i = 0; i <= m + 1; i++) C[i] = h[i][0];  
 ans = (calcn(m, C, n) \* poww(R, n) - c) % mod;  
 if(ans < 0) ans += mod;  
 return ans;  
}  
  
int main()  
{  
 //freopen("in1.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 int t, n, m;  
 init(1000);  
 scanf("%d", &t);  
 while(t--){  
 scanf("%d %d", &n, &m);  
 for (int i = 0; i <= n; i++){  
 scanf("%lld", &a[i]);  
 a[i] %= mod;  
 }  
 a[n + 1] = calcn(n, a, n + 1) % mod;  
 for (int i = 0; i <= n + 1; i++){  
 if(i) a[i] += a[i - 1], a[i] %= mod;  
 }  
 int l, r;  
 while(m--){  
 scanf("%d %d", &l, &r);  
 LL ans = calcn(n + 1, a, r) - calcn(n + 1, a, l - 1);  
 printf("%lld\n", (ans % mod + mod) % mod);  
 }  
 }  
   
 return 0;  
  
}  
/\*\*/  
  
/\*  
1  
3 2  
1 10 49 142  
6 7  
95000 100000  
\*/

## 拉姆齐定理(r(3,3))

/\*  
https://ac.nowcoder.com/acm/contest/5205/A  
\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
int t, n, m;  
int mp[7][7];  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 scanf("%d", &t);  
 while(t--){  
 scanf("%d %d", &n, &m);  
 if(n >= 6){  
 for (int i = 1, x, y; i <= m; i++) scanf("%d %d", &x, &y);  
 printf("yes\n"); continue;  
 }  
 memset(mp, 0, sizeof(mp));  
 for (int i = 1, x, y; i <= m; i++){  
 scanf("%d %d", &x, &y);  
 mp[x][y] = mp[y][x] = 1;  
 }  
 int f = 0;  
 for (int i = 1; i <= n; i++){  
 for (int j = i + 1; j <= n; j++){  
 for (int k = j + 1; k <= n; k++){  
 int ans = mp[i][j] + mp[j][k] + mp[k][i];  
 if(ans == 3 || ans == 0) {f = 1; break;}  
 }  
 }  
 }  
 if(f) printf("yes\n");  
 else printf("no\n");  
 }  
  
 return 0;  
}  
/\*\*/

## 牛顿迭代

/\*  
http://acm.hdu.edu.cn/showproblem.php?pid=2899  
\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
int t;  
double y;  
  
double f(double x){  
 return 6 \* x \* x \* x \* x \* x \* x \* x + 8 \* x \* x \* x \* x \* x \* x + 7 \* x \* x \* x + 5 \* x \* x - y \* x;  
}  
  
double \_f(double x){  
 return 42 \* x \* x \* x \* x \* x \* x + 48 \* x \* x \* x \* x \* x + 21 \* x \* x + 10 \* x - y;  
}  
  
double \_\_f(double x){  
 return 42 \* 6 \* x \* x \* x \* x \* x + 48 \* 5 \* x \* x \* x \* x + 21 \* 2 \* x + 10;  
}  
  
double newton(double x){  
 while(fabs(\_f(x)) > 1e-6){  
 x -= \_f(x) / \_\_f(x);  
 }  
 return x;  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 scanf("%d", &t);  
 while(t--){  
 scanf("%lf", &y);  
 double ans = f(0);  
 for (int i = 0; i <= 100; i++){  
 double x = newton(i);  
 if(0 <= x && x <= 100) ans = min(ans, f(x));  
 }  
 printf("%.4lf\n", ans);  
 }  
  
 return 0;  
}  
/\*\*/

## 求1~n中二进制数第k位为1的种数

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
LL fun(LL n, int k){ // 1~n中求二进制数第k位为1的所有种数  
 return n / (1LL << (k + 1)) \* (1LL << k) + max(0LL, (n % (1LL << (k + 1)) - (1LL << k) + 1));  
}  
  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 return 0;  
}  
/\*\*/

## 求等差数列整除整数下取整求和

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
const long long mod = 1e9 + 7;  
  
LL get(LL a, LL b, LL c, LL n){//a为公差，b为初始项，c为除数，n为项数  
 if(n <= 0) return 0;  
 if(n == 1) return (b / c) % mod;  
 LL tmp = 0;  
 tmp += (a / c) % mod \* ((n - 1) \* n / 2 % mod) % mod;  
 tmp %= mod;  
 tmp += (b / c) \* n % mod;  
 tmp %= mod;  
 a = a % c;  
 b = b % c;  
 if (a == 0) return tmp;  
 else return (tmp + get(c, (a \* n + b) % c, a, (a \* n + b) / c)) % mod;  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 return 0;  
}  
/\*\*/

## 原根

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
const int maxn = 1e6 + 5;  
  
int mod, cnt, tot;  
int prim[maxn], factor[1005], a[maxn];  
bool vis[maxn];  
  
int get\_phi(int x){  
 int res = x;  
 for (int i = 2; i \* i <= x; i++){  
 if(x % i == 0){  
 res = res / i \* (i - 1);  
 while(x % i == 0) x /= i;  
 }  
 }  
 if(x > 1) res = res / x \* (x - 1);  
 return res;  
}  
  
void get\_factor(int x){  
 for (int i = 2; i \* i <= x; i++){  
 if(x % i == 0){  
 factor[cnt++] = i;  
 while(x % i == 0) x /= i;  
 }  
 }  
 if(x > 1) factor[cnt++] = x;  
}  
  
LL poww(LL x, LL num, LL mod){  
 LL res = 1;  
 while(num){  
 if(num & 1) res = res \* x % mod;  
 x = x \* x % mod;  
 num >>= 1;  
 }  
 return res;  
}  
  
int get\_rt(int x){  
 for (int i = 2; i < x; i++){  
 int f = 0;  
 for (int j = 0; j < cnt; j++){  
 if(poww(i, (x - 1) / factor[j], x) == 1) {f = 1; break;}  
 }  
 if(!f) return i;  
 }  
 return -1;  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 scanf("%d", &mod);  
 get\_factor(get\_phi(mod));  
 int ans = get\_rt(mod);  
 printf("%d\n", ans);  
  
 return 0;  
}  
/\*\*/

## 斐波那契数列

前n项的平方 = f(n) \* f(n + 1)  
前偶数项的平方 = f(2n + 1) - f(1)  
前奇数项的平方 = f(2n) - f(2) + f(1)  
f(n) \* f(n) = f(n - 1) \* f(n + 1) + (-1) ^ (n - 1)  
f(2n - 2m - 2) \* (f(2n) + f(2n + 2)) = f(2m + 2) + f(4n - 2m) (n > m >= -1 && n >= 1)  
f(2n) / f(n) = f(n - 1) + f(n + 1)

## **二次剩余模板(mod为奇质数)**

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
using namespace std;  
typedef long long LL;  
  
const long long mod = 1e9 + 7;//奇质数  
LL ksm(LL k, LL n){  
 LL s = 1;  
 for(; n; n >>= 1, k = k \* k % mod) if(n & 1) s = s \* k % mod;  
 return s;  
}  
namespace number{  
 LL D;  
 struct Z{  
 LL x, y;  
 Z(LL \_x = 0, LL \_y = 0) {x = \_x, y = \_y;}  
 };  
 Z operator +(const Z &x, const Z &y) {return Z((x.x + y.x) % mod, (x.y + y.y) % mod);}  
 Z operator -(const Z &x, const Z &y) {return Z((x.x - y.x + mod) % mod, (x.y - y.y + mod) % mod);}  
 Z operator \*(const Z &x, const Z &y) {return Z((x.x \* y.x % mod + D \* x.y % mod \* y.y % mod + mod) % mod, (x.y \* y.x % mod + x.x \* y.y % mod) % mod);}  
 Z opt(const Z &x) {return Z(mod - x.x, mod - x.y);}  
 Z pwr(Z k, LL n){  
 Z s = Z(1, 0);  
 for(; n; n >>= 1, k = k \* k) if(n & 1) s = s \* k;  
 return s;  
 }  
}  
using namespace number;  
LL cipolla(LL k){  
 k %= mod;  
 if(ksm(k, (mod - 1) / 2) == mod - 1) return -1;// -1, -1  
 if(k == 0) return 0;//0, 0  
 LL a = rand() % mod;  
 while(ksm((a \* a % mod - k + mod) % mod, (mod - 1) / 2) <= 1) a = rand() % mod;  
 D=(a \* a % mod - k + mod) % mod;  
 LL v = (pwr(Z(a, 1), (mod + 1) / 2)).x;  
 return v;// 有两个解，一个是v, 一个是mod - v  
}  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 cout << cipolla(10);  
 return 0;  
}  
/\*\*/

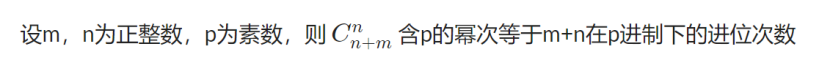
## BM模板

#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <algorithm>  
#include <vector>  
#include <string>  
#include <map>  
#include <set>  
#include <cassert>  
using namespace std;  
#define rep(i,a,n) for (int i=a;i<n;i++)  
#define per(i,a,n) for (int i=n-1;i>=a;i--)  
#define pb push\_back  
#define mp make\_pair  
#define all(x) (x).begin(),(x).end()  
#define fi first  
#define se second  
#define SZ(x) ((int)(x).size())  
typedef vector<int> VI;  
typedef long long ll;  
typedef pair<int,int> PII;  
const ll mod = 998244353;  
ll powmod(ll a,ll b)  
{  
 ll res=1;  
 a%=mod;  
 assert(b>=0);  
 for(;b;b>>=1)  
 {  
 if(b&1)  
 res=res\*a%mod;  
 a=a\*a%mod;  
 }  
 return res;  
}  
int \_, n;  
namespace linear\_seq {  
 const int N=10010;  
 ll res[N],base[N],\_c[N],\_md[N];  
 vector<int> Md;  
 void mul(ll \*a,ll \*b,int k) {  
 for(int i = 0 ; i < k + k ; ++i)  
 \_c[i]=0;  
 for(int i = 0 ; i < k ;++i)  
 if (a[i])  
 for(int j = 0 ;j < k ;++ j)  
 \_c[i+j]=(\_c[i+j]+a[i]\*b[j])%mod;  
 for (int i=k+k-1;i>=k;i--)  
 if (\_c[i])  
 for(int j = 0 ; j<(int ) Md.size() ; ++ j)  
 \_c[i-k+Md[j]]=(\_c[i-k+Md[j]]-\_c[i]\*\_md[Md[j]])%mod;  
 for(int i =0 ; i< k ; ++i)  
 a[i]=\_c[i];  
 }  
 int solve(ll n,VI a,VI b) {  
 ll ans=0,pnt=0;  
 int k=SZ(a);  
 assert( SZ(a) == SZ(b) );  
 for(int i = 0 ;i < k ; ++ i)  
 \_md[k-1-i] = -a[i] ; \_md[k] = 1 ;  
 Md.clear() ;  
 for(int i =0 ; i < k ; ++ i)  
 if (\_md[i]!=0)  
 Md.push\_back(i);  
 for(int i = 0; i< k ;++ i)  
 res[i]=base[i]=0;  
 res[0]=1;  
 while ((1ll<<pnt)<=n)  
 pnt++;  
 for (int p=pnt;p>=0;p--) {  
 mul(res,res,k);  
 if ((n>>p)&1) {  
 for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;  
 for(int j = 0 ;j < (int)Md.size() ; ++ j)  
 res[ Md[j] ]=(res[ Md[j] ]-res[k]\*\_md[Md[j]])%mod;  
 }  
 }  
 rep(i,0,k) ans=(ans+res[i]\*b[i])%mod;  
 if (ans<0) ans+=mod;  
 return ans;  
 }  
 VI BM(VI s) {  
 VI C(1,1),B(1,1);  
 int L=0,m=1,b=1;  
 for(int n= 0 ;n < (int)s.size(); ++ n ) {  
 ll d=0;  
 for(int i =0 ; i < L +1 ;++ i)  
 d=(d+(ll)C[i]\*s[n-i])%mod;  
 if (d==0) ++m;  
 else if (2\*L<=n) {  
 VI T=C;  
 ll c=mod-d\*powmod(b,mod-2)%mod;  
 while (SZ(C)<SZ(B)+m)  
 C.push\_back(0);  
 for(int i =0 ; i < (int)B.size(); ++ i)  
 C[i+m]=(C[i+m]+c\*B[i])%mod;  
 L=n+1-L; B=T; b=d; m=1;  
 } else {  
 ll c=mod-d\*powmod(b,mod-2)%mod;  
 while (SZ(C)<SZ(B)+m)  
 C.push\_back(0);  
 for(int i = 0 ;i <(int) B.size() ; ++ i)  
 C[i+m]=(C[i+m]+c\*B[i])%mod;  
 ++m;  
 }  
 }  
 return C;  
 }  
 ll gao(VI a,ll n) {  
 VI c=BM(a);  
 c.erase(c.begin());  
 for( int i = 0 ; i < (int)c.size( );++i )  
 c[i]=(mod-c[i])%mod;  
 return (ll)solve(n,c,VI(a.begin(),a.begin()+SZ(c)));  
 }  
};  
int main() {  
 long long n;  
 VI a;  
 int N,v;  
 a.push\_back(4);  
 a.push\_back(12);  
 a.push\_back(33);  
 a.push\_back(88);  
 a.push\_back(232);  
 a.push\_back(609);  
 a.push\_back(1596);  
 for (;~scanf("%lld",&n);)  
 printf("%lld\n",linear\_seq::gao(a,n-1));  
 return 0 ;  
}  
  
输出公式版  
#include <bits/stdc++.h>  
using namespace std;  
typedef long long ll;  
#define rep(i,a,n) for(int i=a;i<n;i++)  
namespace linear  
{  
 ll mo=1000000009;  
 vector<ll> v;  
 double a[105][105],del;  
 int k;  
 struct matrix  
 {  
 int n;  
 ll a[50][50];  
 matrix operator \* (const matrix & b)const  
 {  
 matrix c;  
 c.n=n;  
 rep(i,0,n)rep(j,0,n)c.a[i][j]=0;  
 rep(i,0,n)rep(j,0,n)rep(k,0,n)  
 c.a[i][j]=(c.a[i][j]+a[i][k]\*b.a[k][j]%mo)%mo;  
 return c;  
 }  
 }A;  
 bool solve(int n)  
 {  
 rep(i,1,n+1)  
 {  
 int t=i;  
 rep(j,i+1,n+1)if(fabs(a[j][i])>fabs(a[t][i]))t=j;  
 if(fabs(del=a[t][i])<1e-6)return false;  
 rep(j,i,n+2)swap(a[i][j],a[t][j]);  
 rep(j,i,n+2)a[i][j]/=del;  
 rep(t,1,n+1)if(t!=i)  
 {  
 del=a[t][i];  
 rep(j,i,n+2)a[t][j]-=a[i][j]\*del;  
 }  
 }  
 return true;  
 }  
 void build(vector<ll> V)  
 {  
 v=V;  
 int n=(v.size()-1)/2;  
 k=n;  
 while(1)  
 {  
 rep(i,0,k+1)  
 {  
 rep(j,0,k)a[i+1][j+1]=v[n-1+i-j];  
 a[i+1][k+1]=1;  
 a[i+1][k+2]=v[n+i];  
 }  
 if(solve(n+1))break;  
 n--;k--;  
 }  
 A.n=k+1;  
 rep(i,0,A.n)rep(j,0,A.n)A.a[i][j]=0;  
 rep(i,0,A.n)A.a[i][0]=(int)round(a[i+1][A.n+1]);  
 rep(i,0,A.n-2)A.a[i][i+1]=1;  
 A.a[A.n-1][A.n-1]=1;  
 }  
 void formula()  
 {  
 printf("f(n) =");  
 rep(i,0,A.n-1)printf(" (%lld)\*f(n-%d) +",A.a[i][0],i+1);  
 printf(" (%lld)\n",A.a[A.n-1][0]);  
 }  
 ll cal(ll n)  
 {  
 if(n<v.size())return v[n];  
 n=n-k+1;  
 matrix B,T=A;  
 B.n=A.n;  
 rep(i,0,B.n)rep(j,0,B.n)B.a[i][j]=i==j?1:0;  
 while(n)  
 {  
 if(n&1)B=B\*T;  
 n>>=1;  
 T=T\*T;  
 }  
 ll ans=0;  
 rep(i,0,B.n-1)ans=(ans+v[B.n-2-i]\*B.a[i][0]%mo)%mo;  
 ans=(ans+B.a[B.n-1][0])%mo;  
 while(ans<0)ans+=mo;  
 return ans;  
 }  
}  
int main(void){  
 vector<ll> V;  
 V.push\_back(3);  
 V.push\_back(9);  
 V.push\_back(20);  
 V.push\_back(46);  
 V.push\_back(106);  
 V.push\_back(244);  
 V.push\_back(560);  
 V.push\_back(1286);  
 V.push\_back(2956);  
 V.push\_back(6794);  
 V.push\_back(15610);  
 linear::build(V);  
 linear::formula();  
// ll n;  
// while(~scanf("%lld",&n))  
// {  
// printf("%lld\n",linear::cal(n-1));  
// }  
 return 0;  
}  
  
求任意模数的BM模板（模数可以为非质数）  
#include <bits/stdc++.h>  
   
#include<ext/pb\_ds/assoc\_container.hpp>  
#include<ext/pb\_ds/tree\_policy.hpp>//用tree  
   
using namespace \_\_gnu\_pbds;  
   
using namespace std;  
typedef long long ll;  
typedef unsigned long long ul;  
#define mp make\_pair  
#define pb push\_back  
#define sqr(x) ((x)\*(x))  
#define sz(x) (int)x.size()  
#define all(x) x.begin(), x.end()  
typedef pair<int, int> pii;  
typedef pair<ll, ll> pll;  
const int mn = 105;  
const int mod = 1000000000;  
   
int n, m;  
   
// given first m items init[0..m-1] and coefficents trans[0..m-1] or  
// given first 2 \*m items init[0..2m-1], it will compute trans[0..m-1]  
// for you. trans[0..m] should be given as that  
// init[m] = sum\_{i=0}^{m-1} init[i] \* trans[i]  
struct LinearRecurrence {  
 using int64 = long long;  
 using vec = std::vector<int64>;  
   
 static void extand(vec &a, size\_t d, int64 value = 0) {  
 if (d <= a.size()) return;  
 a.resize(d, value);  
 }  
   
 static vec BerlekampMassey(const vec &s, int64 mod) {  
 std::function<int64(int64)> inverse = [&](int64 a) {  
 return a == 1 ? 1 : (int64) (mod - mod / a) \* inverse(mod % a) % mod;  
 };  
 vec A = {1}, B = {1};  
 int64 b = s[0];  
 for (size\_t i = 1, m = 1; i < s.size(); ++i, m++) {  
 int64 d = 0;  
 for (size\_t j = 0; j < A.size(); ++j) {  
 d += A[j] \* s[i - j] % mod;  
 }  
 if (!(d %= mod)) continue;  
 if (2 \* (A.size() - 1) <= i) {  
 auto temp = A;  
 extand(A, B.size() + m);  
 int64 coef = d \* inverse(b) % mod;  
 for (size\_t j = 0; j < B.size(); ++j) {  
 A[j + m] -= coef \* B[j] % mod;  
 if (A[j + m] < 0) A[j + m] += mod;  
 }  
 B = temp, b = d, m = 0;  
 } else {  
 extand(A, B.size() + m);  
 int64 coef = d \* inverse(b) % mod;  
 for (size\_t j = 0; j < B.size(); ++j) {  
 A[j + m] -= coef \* B[j] % mod;  
 if (A[j + m] < 0) A[j + m] += mod;  
 }  
 }  
 }  
 return A;  
 }  
   
 static void exgcd(int64 a, int64 b, int64 &g, int64 &x, int64 &y) {  
 if (!b) x = 1, y = 0, g = a;  
 else {  
 exgcd(b, a % b, g, y, x);  
 y -= x \* (a / b);  
 }  
 }  
   
 static int64 crt(const vec &c, const vec &m) {  
 int n = c.size();  
 int64 M = 1, ans = 0;  
 for (int i = 0; i < n; ++i) M \*= m[i];  
 for (int i = 0; i < n; ++i) {  
 int64 x, y, g, tm = M / m[i];  
 exgcd(tm, m[i], g, x, y);  
 ans = (ans + tm \* x \* c[i] % M) % M;  
 }  
 return (ans + M) % M;  
 }  
   
 static vec ReedsSloane(const vec &s, int64 mod) {  
 auto inverse = [](int64 a, int64 m) {  
 int64 d, x, y;  
 exgcd(a, m, d, x, y);  
 return d == 1 ? (x % m + m) % m : -1;  
 };  
 auto L = [](const vec &a, const vec &b) {  
 int da = (a.size() > 1 || (a.size() == 1 && a[0])) ? a.size() - 1 : -1000;  
 int db = (b.size() > 1 || (b.size() == 1 && b[0])) ? b.size() - 1 : -1000;  
 return std::max(da, db + 1);  
 };  
 auto prime\_power = [&](const vec &s, int64 mod, int64 p, int64 e) {  
 // linear feedback shift register mod p^e, p is prime  
 std::vector<vec> a(e), b(e), an(e), bn(e), ao(e), bo(e);  
 vec t(e), u(e), r(e), to(e, 1), uo(e), pw(e + 1);;  
 pw[0] = 1;  
 for (int i = pw[0] = 1; i <= e; ++i) pw[i] = pw[i - 1] \* p;  
 for (int64 i = 0; i < e; ++i) {  
 a[i] = {pw[i]}, an[i] = {pw[i]};  
 b[i] = {0}, bn[i] = {s[0] \* pw[i] % mod};  
 t[i] = s[0] \* pw[i] % mod;  
 if (t[i] == 0) {  
 t[i] = 1, u[i] = e;  
 } else {  
 for (u[i] = 0; t[i] % p == 0; t[i] /= p, ++u[i]);  
 }  
 }  
 for (size\_t k = 1; k < s.size(); ++k) {  
 for (int g = 0; g < e; ++g) {  
 if (L(an[g], bn[g]) > L(a[g], b[g])) {  
 ao[g] = a[e - 1 - u[g]];  
 bo[g] = b[e - 1 - u[g]];  
 to[g] = t[e - 1 - u[g]];  
 uo[g] = u[e - 1 - u[g]];  
 r[g] = k - 1;  
 }  
 }  
 a = an, b = bn;  
 for (int o = 0; o < e; ++o) {  
 int64 d = 0;  
 for (size\_t i = 0; i < a[o].size() && i <= k; ++i) {  
 d = (d + a[o][i] \* s[k - i]) % mod;  
 }  
 if (d == 0) {  
 t[o] = 1, u[o] = e;  
 } else {  
 for (u[o] = 0, t[o] = d; t[o] % p == 0; t[o] /= p, ++u[o]);  
 int g = e - 1 - u[o];  
 if (L(a[g], b[g]) == 0) {  
 extand(bn[o], k + 1);  
 bn[o][k] = (bn[o][k] + d) % mod;  
 } else {  
 int64 coef = t[o] \* inverse(to[g], mod) % mod \* pw[u[o] - uo[g]] % mod;  
 int m = k - r[g];  
 extand(an[o], ao[g].size() + m);  
 extand(bn[o], bo[g].size() + m);  
 for (size\_t i = 0; i < ao[g].size(); ++i) {  
 an[o][i + m] -= coef \* ao[g][i] % mod;  
 if (an[o][i + m] < 0) an[o][i + m] += mod;  
 }  
 while (an[o].size() && an[o].back() == 0) an[o].pop\_back();  
 for (size\_t i = 0; i < bo[g].size(); ++i) {  
 bn[o][i + m] -= coef \* bo[g][i] % mod;  
 if (bn[o][i + m] < 0) bn[o][i + m] -= mod;  
 }  
 while (bn[o].size() && bn[o].back() == 0) bn[o].pop\_back();  
 }  
 }  
 }  
 }  
 return std::make\_pair(an[0], bn[0]);  
 };  
   
 std::vector<std::tuple<int64, int64, int>> fac;  
 for (int64 i = 2; i \* i <= mod; ++i)  
 if (mod % i == 0) {  
 int64 cnt = 0, pw = 1;  
 while (mod % i == 0) mod /= i, ++cnt, pw \*= i;  
 fac.emplace\_back(pw, i, cnt);  
 }  
 if (mod > 1) fac.emplace\_back(mod, mod, 1);  
 std::vector<vec> as;  
 size\_t n = 0;  
 for (auto &&x: fac) {  
 int64 mod, p, e;  
 vec a, b;  
 std::tie(mod, p, e) = x;  
 auto ss = s;  
 for (auto &&x: ss) x %= mod;  
 std::tie(a, b) = prime\_power(ss, mod, p, e);  
 as.emplace\_back(a);  
 n = std::max(n, a.size());  
 }  
 vec a(n), c(as.size()), m(as.size());  
 for (size\_t i = 0; i < n; ++i) {  
 for (size\_t j = 0; j < as.size(); ++j) {  
 m[j] = std::get<0>(fac[j]);  
 c[j] = i < as[j].size() ? as[j][i] : 0;  
 }  
 a[i] = crt(c, m);  
 }  
 return a;  
 }  
   
 LinearRecurrence(const vec &s, const vec &c, int64 mod) :  
 init(s), trans(c), mod(mod), m(s.size()) {}  
   
 LinearRecurrence(const vec &s, int64 mod, bool is\_prime = true) : mod(mod) {  
 vec A;  
 if (is\_prime) A = BerlekampMassey(s, mod);  
 else A = ReedsSloane(s, mod);  
 if (A.empty()) A = {0};  
 m = A.size() - 1;  
 trans.resize(m);  
 for (int i = 0; i < m; ++i) {  
 trans[i] = (mod - A[i + 1]) % mod;  
 }  
 std::reverse(trans.begin(), trans.end());  
 init = {s.begin(), s.begin() + m};  
 }  
   
 int64 calc(int64 n) {  
 if (mod == 1) return 0;  
 if (n < m) return init[n];  
 vec v(m), u(m << 1);  
 int msk = !!n;  
 for (int64 m = n; m > 1; m >>= 1) msk <<= 1;  
 v[0] = 1 % mod;  
 for (int x = 0; msk; msk >>= 1, x <<= 1) {  
 std::fill\_n(u.begin(), m \* 2, 0);  
 x |= !!(n & msk);  
 if (x < m) u[x] = 1 % mod;  
 else {// can be optimized by fft/ntt  
 for (int i = 0; i < m; ++i) {  
 for (int j = 0, t = i + (x & 1); j < m; ++j, ++t) {  
 u[t] = (u[t] + v[i] \* v[j]) % mod;  
 }  
 }  
 for (int i = m \* 2 - 1; i >= m; --i) {  
 for (int j = 0, t = i - m; j < m; ++j, ++t) {  
 u[t] = (u[t] + trans[j] \* u[i]) % mod;  
 }  
 }  
 }  
 v = {u.begin(), u.begin() + m};  
 }  
 int64 ret = 0;  
 for (int i = 0; i < m; ++i) {  
 ret = (ret + v[i] \* init[i]) % mod;  
 }  
 return ret;  
 }  
   
 vec init, trans;  
 int64 mod;  
 int m;  
};  
   
ll f[25005];  
   
ll powmod(ll a, ll b, ll P) {  
 ll t = 1;  
 for (; b; b >>= 1, a = a \* a % P)  
 if (b & 1) t = t \* a % P;  
 return t;  
}  
   
int main() {  
#ifdef trote  
 //freopen("in.txt", "r", stdin);  
 freopen("../1.txt", "r", stdin);  
 //freopen("out2.txt", "w", stdout);  
 size\_t st = clock();  
#endif  
 scanf("%d%d", &n, &m);  
 f[0] = 0;  
 f[1] = 1;  
 for (int i = 2; i <= 25000; i++) {  
 f[i] = (f[i - 1] + f[i - 2]) % mod;  
 }  
 vector<ll> v;  
 ll S = 0;  
 for (int i = 0; i < 900; i++) {  
 S = (S + powmod(f[i], m, mod)) % mod;  
 v.pb(S);  
 }  
 LinearRecurrence solver(v, mod, false);  
 printf("%lld\n", solver.calc(n));  
   
 return 0;  
}

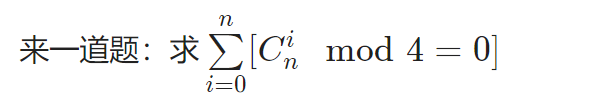
模板二

#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <algorithm>  
#include <vector>  
#include <string>  
#include <map>  
#include <set>  
#include <cassert>  
#include<bits/stdc++.h>  
using namespace std;  
#define rep(i,a,n) for (int i=a;i<n;i++)  
#define per(i,a,n) for (int i=n-1;i>=a;i--)  
#define pb push\_back  
#define mp make\_pair  
#define all(x) (x).begin(),(x).end()  
#define fi first  
#define se second  
#define SZ(x) ((int)(x).size())  
typedef vector<int> VI;  
typedef long long ll;  
typedef pair<int,int> PII;  
const ll mod=1000000007;  
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1){if(b&1)res=res\*a%mod;a=a\*a%mod;}return res;}  
// head  
  
ll n;  
namespace linear\_seq {  
 const int N=10010;  
 ll res[N],base[N],\_c[N],\_md[N];  
  
 vector<int> Md;  
 void mul(ll \*a,ll \*b,int k) {  
 rep(i,0,k+k) \_c[i]=0;  
 rep(i,0,k) if (a[i]) rep(j,0,k) \_c[i+j]=(\_c[i+j]+a[i]\*b[j])%mod;  
 for (int i=k+k-1;i>=k;i--) if (\_c[i])  
 rep(j,0,SZ(Md)) \_c[i-k+Md[j]]=(\_c[i-k+Md[j]]-\_c[i]\*\_md[Md[j]])%mod;  
 rep(i,0,k) a[i]=\_c[i];  
 }  
 int solve(ll n,VI a,VI b) { // a 系数 b 初值 b[n+1]=a[0]\*b[n]+...  
 ll ans=0,pnt=0;  
 int k=SZ(a);  
 assert(SZ(a)==SZ(b));  
 rep(i,0,k) \_md[k-1-i]=-a[i];\_md[k]=1;  
 Md.clear();  
 rep(i,0,k) if (\_md[i]!=0) Md.push\_back(i);  
 rep(i,0,k) res[i]=base[i]=0;  
 res[0]=1;  
 while ((1ll<<pnt)<=n) pnt++;  
 for (int p=pnt;p>=0;p--) {  
 mul(res,res,k);  
 if ((n>>p)&1) {  
 for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;  
 rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]\*\_md[Md[j]])%mod;  
 }  
 }  
 rep(i,0,k) ans=(ans+res[i]\*b[i])%mod;  
 if (ans<0) ans+=mod;  
 return ans;  
 }  
 VI BM(VI s) {  
 VI C(1,1),B(1,1);  
 int L=0,m=1,b=1;  
 rep(n,0,SZ(s)) {  
 ll d=0;  
 rep(i,0,L+1) d=(d+(ll)C[i]\*s[n-i])%mod;  
 if (d==0) ++m;  
 else if (2\*L<=n) {  
 VI T=C;  
 ll c=mod-d\*powmod(b,mod-2)%mod;  
 while (SZ(C)<SZ(B)+m) C.pb(0);  
 rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c\*B[i])%mod;  
 L=n+1-L; B=T; b=d; m=1;  
 } else {  
 ll c=mod-d\*powmod(b,mod-2)%mod;  
 while (SZ(C)<SZ(B)+m) C.pb(0);  
 rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c\*B[i])%mod;  
 ++m;  
 }  
 }  
 return C;  
 }  
 int gao(VI a,ll n) {  
 VI c=BM(a);  
 c.erase(c.begin());  
 rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;  
 return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));  
 }  
};  
  
int main() {  
 /\*push\_back 进去前 8~10 项左右、最后调用 gao 得第 n 项\*/  
 vector<int>v;  
 v.push\_back(3);  
 v.push\_back(9);  
 v.push\_back(20);  
 v.push\_back(46);  
 v.push\_back(106);  
 v.push\_back(244);  
 v.push\_back(560);  
 v.push\_back(1286);  
 v.push\_back(2956);  
 v.push\_back(6794);  
 int nCase;  
 scanf("%d", &nCase);  
 while(nCase--){  
 scanf("%lld", &n);  
 printf("%lld\n",1LL \* linear\_seq::gao(v,n-1) % mod);  
 }  
}

## Kummer定理



证明：

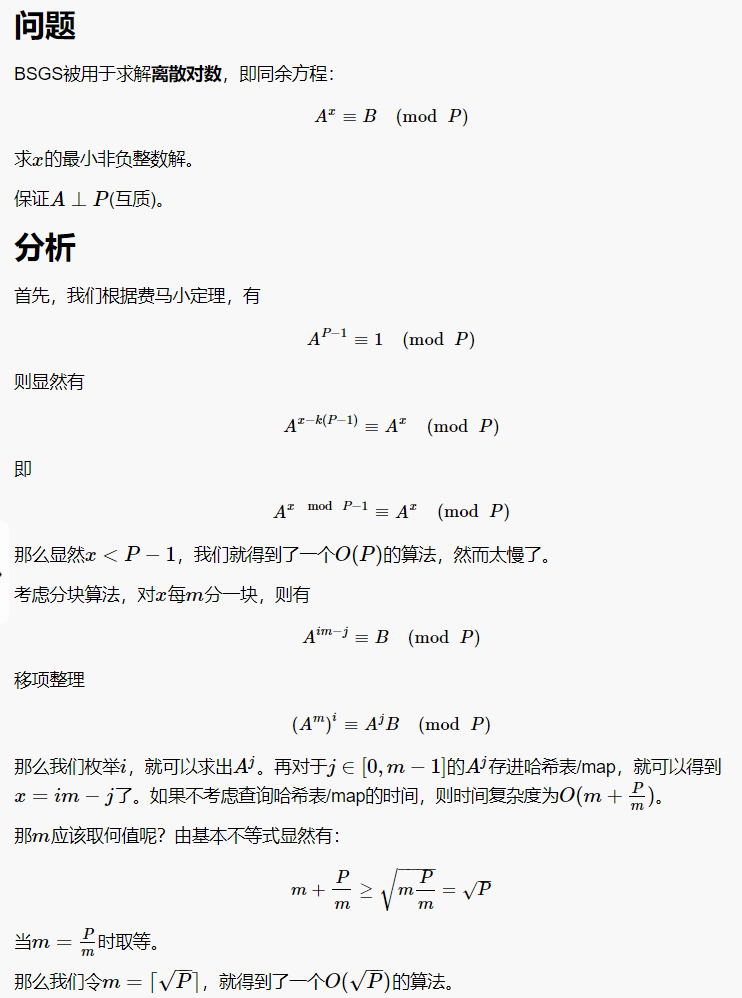


我们用总数减去不符合规定的数量：进位0次或1次的。我们可以枚举进位的位置。

一道类似的题，求9的倍数：

il int calcS(int n,int up,int ex)  
{  
 if(!up) return sub((LL)(n+2)\*(n+1)/2%md,ex\*(n+1));  
 else return add((LL)(p-1+p-1-n)\*(n+1)/2%md,ex\*(n+1));  
}  
il int calcP(int n,int up,int ex)  
{  
 if(!up) return n+1-ex;  
 else return p-1-n+ex;  
}  
 f[0][0][0][1]=1;  
 for(ri i=1; i<=n; ++i)  
 for(ri j=0; j<=a; ++j)  
 for(ri k=0; k<=1; ++k)  
 for(ri eq=0; eq<=1; ++eq)  
 {  
 int t=f[i-1][j][k][eq];  
 if(!t) continue;  
 for(ri up=0; up<=1; ++up)  
 {  
 int nj=min(j+up,a);  
 if(!eq) inc(f[i][nj][up][0],mul(t,calcS(p-1,k,up)));  
 else  
 {  
 inc(f[i][nj][up][1],mul(t,calcP(A[i],k,up)));  
 if(A[i]) inc(f[i][nj][up][0],mul(t,calcS(A[i]-1,k,up)));  
 }  
 }  
 }

## BSGS大步小步算法





int gcd(int a, int b) { return a ? gcd(b % a, a) : b; }  
int powmod(int a, int b, int p) {  
 int res = 1;  
 while (b > 0) {  
 if (b & 1) res = res \* a % p;  
 a = a \* a % p, b >>= 1;  
 }  
 return res;  
}  
// Finds the primitive root modulo p  
int generator(int p) {  
 vector<int> fact;  
 int phi = p - 1, n = phi;  
 for (int i = 2; i \* i <= n; ++i) {  
 if (n % i == 0) {  
 fact.push\_back(i);  
 while (n % i == 0) n /= i;  
 }  
 }  
 if (n > 1) fact.push\_back(n);  
 for (int res = 2; res <= p; ++res) {  
 bool ok = true;  
 for (int factor : fact) {  
 if (powmod(res, phi / factor, p) == 1) {  
 ok = false;  
 break;  
 }  
 }  
 if (ok) return res;  
 }  
 return -1;  
}  
// This program finds all numbers x such that x^k=a (mod n)  
int main() {  
 int n, k, a;  
 scanf("%d %d %d", &n, &k, &a);  
 if (a == 0) return puts("1\n0"), 0;  
 int g = generator(n);  
 // Baby-step giant-step discrete logarithm algorithm  
 int sq = (int)sqrt(n + .0) + 1;  
 vector<pair<int, int>> dec(sq);  
 for (int i = 1; i <= sq; ++i)  
 dec[i - 1] = {powmod(g, i \* sq \* k % (n - 1), n), i};  
 sort(dec.begin(), dec.end());  
 int any\_ans = -1;  
 for (int i = 0; i < sq; ++i) {  
 int my = powmod(g, i \* k % (n - 1), n) \* a % n;  
 auto it = lower\_bound(dec.begin(), dec.end(), make\_pair(my, 0));  
 if (it != dec.end() && it->first == my) {  
 any\_ans = it->second \* sq - i;  
 break;  
 }  
 }  
 if (any\_ans == -1) return puts("0"), 0;  
 // Print all possible answers  
 int delta = (n - 1) / gcd(k, n - 1);  
 vector<int> ans;  
 for (int cur = any\_ans % delta; cur < n - 1; cur += delta)  
 ans.push\_back(powmod(g, cur, n));  
 sort(ans.begin(), ans.end());  
 printf("%d\n", ans.size());  
 for (int answer : ans) printf("%d ", answer);  
}

## 高斯-约旦消元法

#include<cstdio>  
#include<cmath>  
#include<algorithm>  
using namespace std;  
  
const double eps=1e-8;  
const int maxn=105;  
int n;  
double a[maxn][maxn];  
  
void Gauss(){  
 for(int i=1;i<=n;++i){  
 int r=i;  
 for(int j=i+1;j<=n;++j) //找主元系数的绝对值最大的一行  
 if(fabs(a[j][i])>fabs(a[r][i]))  
 r=j;  
 if(fabs(a[r][i])<eps){ //最大为0,无解  
 printf("No Solution\n");  
 return;  
 }  
 if(r!=i)  
 for(int j=1;j<=n+1;++j)  
 swap(a[i][j],a[r][j]);  
 for(int j=1;j<=n;++j) //消元  
 if(j!=i){  
 double tmp=a[j][i]/a[i][i];  
 for(int k=i;k<=n+1;++k)  
 a[j][k]-=a[i][k]\*tmp;  
 }  
 }  
 for(int i=1;i<=n;++i)  
 printf("%.2f\n",a[i][n+1]/a[i][i]);  
}  
  
int main(){  
 scanf("%d",&n);  
 for(int i=1;i<=n;++i)  
 for(int j=1;j<=n+1;++j)  
 scanf("%lf",&a[i][j]);  
 Gauss();  
 return 0;  
}

## 线性基

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
const int maxn = 1e5 + 5;  
const int maxbit = 63;  
  
LL a[maxn], p[maxbit];  
int n;  
  
struct LinerBase  
{  
 int cnt;  
 LL b[65];  
  
 void init(){  
 memset(b, 0, sizeof(b));  
 }  
  
 bool insert(LL x){  
 for (int i = 63; i >= 0; i--){  
 if(x & (1LL << i)){  
 if(!b[i]) {b[i] = x, cnt++; return true;}  
 x ^= b[i];  
 }  
 }  
 return false;  
 }  
  
 bool check(LL x){  
 for (int i = 63; i >= 0; i--){  
 if(x & (1LL << i)) x ^= b[i];  
 }  
 return x == 0;  
 }  
}p1, p2, p3;  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 scanf("%d", &n);  
 for (int i = 1; i <= n; i++){  
 scanf("%lld", &a[i]);  
 }  
 for (int i = 1; i <= n; i++){  
 for (int j = maxbit - 1; j >= 0; j--){  
 if(a[i] >> j & 1){  
 if(!p[j]) {p[j] = a[i]; break;}  
 a[i] ^= p[j];  
 }  
 }  
 }  
 int m = 0;  
 for (int i = maxbit - 1; i >= 0; i--){  
 if(p[i]) m++;  
 }  
 LL q;  
 while(scanf("%lld", &q) == 1){  
 if(!q){//特判0的情况  
 if(m == n) printf("false\n");  
 else printf("true\n");  
 }else{  
 int f = 0;  
 for (int i = maxbit - 1; i >= 0; i--){  
 if(q >> i & 1){  
 if(!p[i]) {f = 1; break;}  
 }  
 }  
 if(!f) printf("true\n");  
 else printf("false\n");  
 }  
 }  
  
 return 0;  
}  
/\*\*/  
  
//https://www.bilibili.com/video/av40884251/?p=4

## min\_25筛

#include <bits/stdc++.h>  
#include <unordered\_map>  
#include <unordered\_set>  
  
using namespace std;  
#define fi first  
#define se second  
#define pb push\_back  
#define ins insert  
#define Vector Point  
#define ENDL "\n"  
#define lowbit(x) (x&(-x))  
#define mkp(x, y) make\_pair(x,y)  
#define mem(a, x) memset(a,x,sizeof a);  
typedef long long ll;  
typedef long double ld;  
typedef unsigned long long ull;  
typedef pair<int, int> pii;  
typedef pair<ll, ll> pll;  
typedef pair<double, double> pdd;  
const double eps = 1e-8;  
const double pi = acos(-1.0);  
const int inf = 0x3f3f3f3f;  
const double dinf = 1e300;  
const ll INF = 1e18;  
const int maxn = 1e6 + 10;  
  
ll prime[maxn], sum[maxn];  
ll w[maxn], id1[maxn], id2[maxn], g[maxn];  
bitset<maxn> vis;  
int m, cnt, tot;  
ll n, Mod, inv;  
  
inline ll qkp(ll x, ll n, ll p) {  
 ll ans = 1;  
 x %= p;  
 while (n) {  
 if (n & 1) ans = ans \* x % p;  
 x = x \* x % p;  
 n >>= 1;  
 }  
 return ans;  
}  
  
inline ll getInv(ll a, ll p) {  
 return qkp(a, p - 2, p);  
}  
  
inline ll getID(ll x) {  
 return x <= m ? id1[x] : id2[n / x];  
}  
  
inline ll cal(ll x) {  
 return x \* (x + 1) / 2;  
}  
  
inline void getMod(ll &x) {  
 x %= Mod;  
 if (x < 0) x += Mod;  
}  
  
inline void init() {  
 vis.reset();  
 vis[1] = 1;  
 m = sqrt(n), tot = cnt = 0;  
 for (int i = 2; i <= m; i++) {  
 if (!vis[i]) {  
 prime[++cnt] = i;  
 sum[cnt] = sum[cnt - 1] + i;  
 }  
 for (int j = 1; j <= cnt && 1LL \* i \* prime[j] <= m; j++) {  
 vis[i \* prime[j]] = 1;  
 if (i % prime[j] == 0) break;  
 }  
 }  
}  
  
ll solve() { //求质数和  
 init();  
 for (ll l = 1, r; l <= n; l = r + 1) {  
 w[++tot] = n / l;  
 r = n / w[tot];  
 g[tot] = cal(w[tot]) - 1;  
 if (w[tot] <= m) id1[w[tot]] = tot;  
 else id2[r] = tot;  
 }  
  
 for (int i = 1; i <= cnt; i++) {  
 for (int j = 1; j <= tot && prime[i] \* prime[i] <= w[j]; j++) {  
 ll k = getID(w[j] / prime[i]);  
 g[j] -= (g[k] - sum[i - 1]) \* prime[i];  
 }  
 }  
 return g[1];  
}  
  
int main() {  
 //freopen("in.txt","r",stdin);  
 //freopen("out.txt","w",stdout);  
 //ios\_base::sync\_with\_stdio(0), cin.tie(0), cout.tie(0);  
 int t;  
 scanf("%d", &t);  
 while (t--) {  
 scanf("%lld%lld", &n, &Mod);  
// clock\_t start = clock(), end;  
 inv = getInv(2, Mod);  
 ll ans = (n - 1) % Mod \* (n + 4) % Mod \* inv % Mod; //[3,n+1]求和  
 n++;  
 ans = (ans + solve() - 2 + Mod) % Mod;  
 printf("%lld\n", ans);  
// end = clock();  
// printf("time: %d ms\n", (end - start));  
 }  
 return 0;  
}

## 洲阁筛

#pragma GCC optimize(2)  
#include <bits/stdc++.h>  
using namespace std;  
#define LL long long  
int L0[1000005],L1[1000005],sn,m,P[1000005],x,ga[1000005],gb[1000005];  
LL n,a[1000005],b[1000005];  
int main(){  
 n=100000000000LL;  
 // scanf("%lld",&n);  
 sn=sqrt(n);  
 for (int i=2;i<=sn;++i){  
 if (!P[i]) P[++m]=i;  
 for (int j=1;j<=m;++j){  
 x=P[j]\*i;  
 if (x>sn) break;  
 P[x]=1;  
 if (i%P[j]==0) break;  
 }  
 }  
 for (int i=1;i<=sn;++i) a[i]=i,b[i]=n/i;  
 for (int i=1,j=1;i<=sn;++i){  
 while (j<=m&&P[j]\*P[j]<=i) ++j;  
 L0[i]=j;  
 }  
 for (int i=sn,j=L0[sn];i;--i){  
 while (j<=m&&P[j]\*P[j]<=b[i]) ++j;  
 L1[i]=j;  
 }  
 for (int i=1;i<=m;++i){  
 for (int j=1;j<=sn&&i<L1[j];++j){  
 LL y=j\*P[i]; gb[j]=i;  
 if (y<=sn) b[j]-=b[y]+gb[y]+1-i;  
 else b[j]-=a[n/y]+ga[n/y]+1-i;  
 }  
 for (int j=sn;j&&i<L0[j];--j){  
 LL y=j/P[i]; ga[j]=i;  
 a[j]-=a[y]+ga[y]+1-i;  
 }  
 }  
 printf("%lld\n",b[1]+m-1);  
 return 0;  
}

例题：

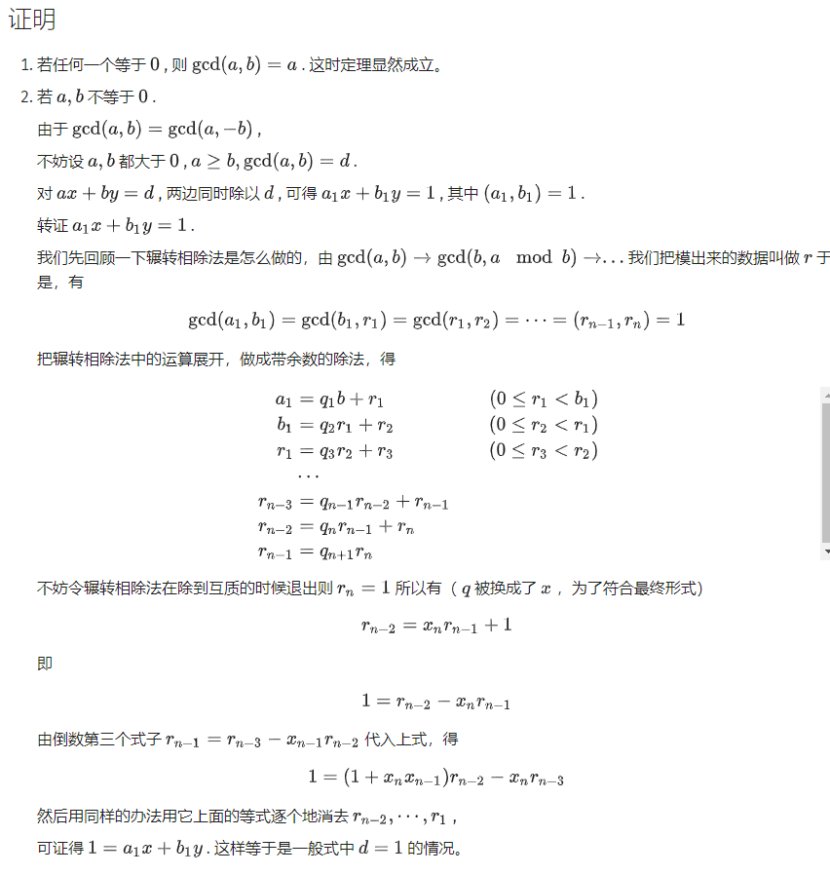
http://www.spoj.com/problems/DIVCNT3/

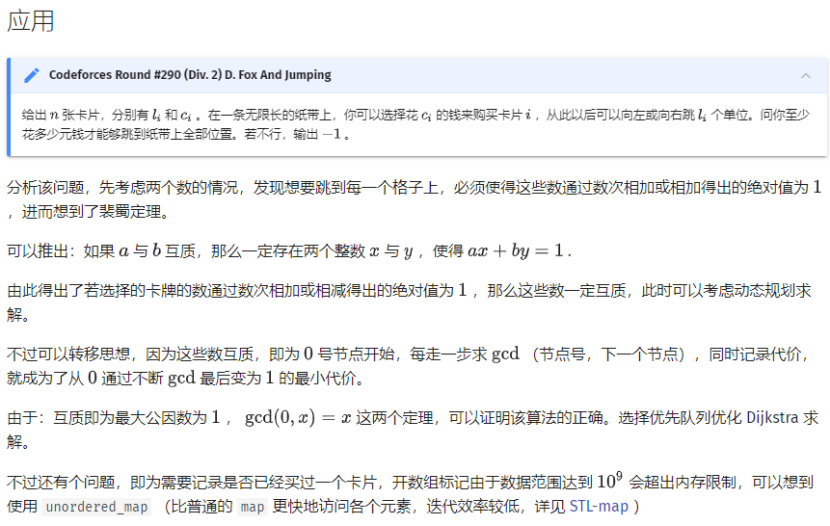
1 #pragma GCC optimize(2)  
 2 #include <bits/stdc++.h>  
 3 using namespace std;  
 4 #define LL long long  
 5 int T,W[340005],L0[340005],L1[340005],L2[340005],sn,m,P[340005],x,ga[340005],gb[340005];  
 6 LL n,a[340005],b[340005],F[340005];  
 7 void hh(){  
 8 scanf("%lld",&n);  
 9 sn=sqrt(n); F[1]=1; m=0;  
10 for (int i=2;i<=sn;++i){  
11 if (!P[i]) P[++m]=i,F[i]=4,W[i]=1;  
12 for (int j=1;j<=m;++j){  
13 x=P[j]\*i;  
14 if (x>sn) break;  
15 P[x]=1;  
16 if (i%P[j]==0){  
17 W[x]=W[i]+1;  
18 F[x]=F[i]/(3\*W[i]+1)\*(3\*W[x]+1);  
19 break;  
20 }else{  
21 W[x]=1; F[x]=F[i]\*4;  
22 }  
23 }  
24 }  
25 for (int i=1;i<=sn;++i)  
26 a[i]=i,b[i]=n/i,ga[i]=gb[i]=0;  
27 for (int i=1,j=1;i<=sn;++i){  
28 while (j<=m&&P[j]\*P[j]<=i) ++j;  
29 L0[i]=j;  
30 }  
31 for (int i=sn,j=L0[sn];i;--i){  
32 while (j<=m&&1ll\*P[j]\*P[j]<=b[i]) ++j;  
33 L1[i]=j;  
34 }  
35 for (int i=1,j=1;i<=sn;++i){  
36 while (j<=m&&P[j]<=i) ++j;  
37 L2[i]=j;  
38 }  
39 for (int i=1;i<=m;++i){  
40 for (int j=1;j<=sn&&i<L1[j];++j){  
41 LL y=j\*P[i]; gb[j]=i;  
42 if (y<=sn) b[j]-=b[y]+gb[y]+1-i;  
43 else b[j]-=a[n/y]+ga[n/y]+1-i;  
44 }  
45 for (int j=sn;j&&i<L0[j];--j){  
46 int y=j/P[i]; ga[j]=i;  
47 a[j]-=a[y]+ga[y]+1-i;  
48 }  
49 }  
50 LL ans=0;  
51 for (int i=1;i<=sn;++i)  
52 ans+=F[i]\*((b[i]-m+gb[i]-1)\*4+1);  
53 for (int i=1;i<=sn;++i)  
54 a[i]=b[i]=1,ga[i]=gb[i]=m+1;  
55 for (int i=m;i;--i){  
56 for (int j=1;j<=sn&&i<L1[j];++j){  
57 LL y=n/j;  
58 if (gb[j]!=i+1) b[j]=(m-i)\*4+1;  
59 y/=P[i]; gb[j]=i;  
60 for (int c=1;y;++c,y/=P[i])  
61 if (y<=sn)  
62 if (ga[y]==i+1) b[j]+=a[y]\*(3\*c+1);  
63 else b[j]+=(max(0,L2[y]-i-1)\*4+1)\*(3\*c+1);  
64 else  
65 if (gb[n/y]==i+1) b[j]+=b[n/y]\*(3\*c+1);  
66 else b[j]+=((m-i)\*4+1)\*(3\*c+1);  
67 }  
68 for (int j=sn;j&&i<L0[j];--j){  
69 int y=j;  
70 if (ga[j]!=i+1) a[j]=max(0,L2[y]-i-1)\*4+1;  
71 y/=P[i]; ga[j]=i;  
72 for (int c=1;y;++c,y/=P[i])  
73 if (ga[y]==i+1) a[j]+=a[y]\*(3\*c+1);  
74 else a[j]+=(max(0,L2[y]-i-1)\*4+1)\*(3\*c+1);  
75 }  
76 }  
77 ans+=b[1]-(ga[sn]==1?a[sn]:m\*4+1);  
78 for (int i=1;i<=sn;++i) P[i]=0;  
79 printf("%lld\n",ans);  
80 }  
81 int main(){  
82 scanf("%d",&T);  
83 while (T--) hh();  
84 return 0;  
85 }

## 杜教筛

/\*  
https://www.luogu.com.cn/problem/P4213  
\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
#include <unordered\_map>  
  
typedef long long LL;  
using namespace std;  
  
const int M = 6000001;  
  
int t, tot;  
LL n;  
int prim[M], mu[M], p[M], sumu[M];  
LL sump[M];  
bool vis[M];  
  
unordered\_map<int, int> mpu;  
unordered\_map<LL, LL> mpp;  
  
void get(){  
 mu[1] = p[1] = 1;  
 for (int i = 2; i < M; i++){  
 if(!vis[i]) prim[tot++] = i, mu[i] = -1, p[i] = i - 1;  
 for (int j = 0; j < tot && prim[j] \* i < M; j++){  
 vis[prim[j] \* i] = true;  
 if(i % prim[j] == 0){  
 p[i \* prim[j]] = p[i] \* prim[j];  
 break;  
 }else{  
 p[i \* prim[j]] = p[i] \* (prim[j] - 1);  
 mu[i \* prim[j]] = -mu[i];  
 }  
 }  
 }  
 for (int i = 1; i < M; i++){  
 sumu[i] = sumu[i - 1] + mu[i];  
 sump[i] = sump[i - 1] + p[i];  
 }  
}  
  
int get\_mu(LL n){  
 if(n < M) return sumu[n];  
 if(mpu[n]) return mpu[n];  
 int ans = 1;  
 for (int l = 2, r; l <= n && l >= 0; l = r + 1){  
 r = n / (n / l);  
 ans -= (r - l + 1) \* get\_mu(n / l);  
 }  
 return mpu[n] = ans;  
}  
  
LL get\_phi(LL n){  
 if(n < M) return sump[n];  
 if(mpp[n]) return mpp[n];  
 LL ans = (n + 1) \* n / 2;  
 for (int l = 2, r; l <= n && l >= 0; l = r + 1){  
 r = n / (n / l);  
 ans -= (r - l + 1) \* get\_phi(n / l);  
 }  
 return mpp[n] = ans;  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 get();  
 scanf("%d", &t);  
 while(t--){  
 scanf("%lld", &n);  
 printf("%lld %d\n", get\_phi(n), get\_mu(n));  
 }  
  
 return 0;  
}  
/\*\*/

## 裴蜀定理





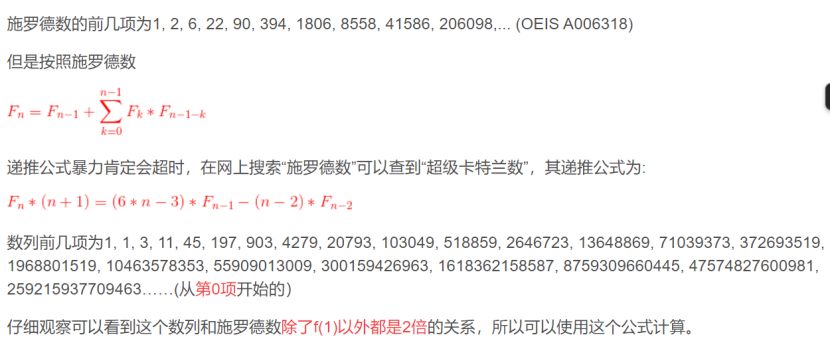
## 中国剩余定理

Mod是互质  
/\*  
同余方程组 ：  
设正整数m1.m2.mk两两互素，则方程组  
x ≡ a1 (mod m1)  
x ≡ a2 (mod m2)  
x ≡ a3 (mod m3)  
.  
.  
x ≡ ak (mod mk)  
有整数解，  
解为 x ≡ (a1 \* M1 \* 1/M1 + a2 \* M2 \* 1/M2 + a3 \* M3 \* 1/M3 + …… +ak \* Mk \* 1/Mk) mod M  
其中 M = M1 \* M2 \* M3 \* …… \* Mk, Mi为M/mi, 1/Mi为Mi的逆元  
\*/  
  
void exgcd(int a, int b, int &x, int &y)  
{  
 if(b == 0) {  
 x = 1;  
 y = 0;  
 return a;  
 }  
 exgcd(b, a % b, x, y);  
 int t = x;  
 x = y;  
 y = t - a / b \* y;  
 return r;  
}  
  
LL CRT(int m[], int a[], int n) // m 是 mod 的数, a 是余数, n 是方程组组数  
{  
 LL M = 1, ans = 0;  
 for(int i = 0; i < n; i++)  
 M \*= m[i];  
 for(int i = 0; i < n; i++) {  
 int x, y;  
 LL Mi = M / m[i];  
 exgcd(Mi, m[i], x, y); //求出的 x 即 Mi 的逆元  
 ans = (ans + Mi \* a[i] \* x % M + M) % M;  
 }  
 return ans;  
}  
Mod不互质：  
https://www.luogu.com.cn/problem/P4777  
#include<iostream>  
#include<vector>  
#include<algorithm>  
#include<queue>  
#include<cstring>  
#include<cstdio>  
using namespace std;  
typedef long long lt;  
  
lt read()  
{  
 lt f=1,x=0;  
 char ss=getchar();  
 while(ss<'0'||ss>'9'){if(ss=='-')f=-1;ss=getchar();}  
 while(ss>='0'&&ss<='9'){x=x\*10+ss-'0';ss=getchar();}  
 return f\*x;  
}  
  
const int maxn=100010;  
int n;  
lt ai[maxn],bi[maxn];  
  
lt mul(lt a,lt b,lt mod)  
{  
 lt res=0;  
 while(b>0)  
 {  
 if(b&1) res=(res+a)%mod;  
 a=(a+a)%mod;  
 b>>=1;  
 }  
 return res;  
}  
  
lt exgcd(lt a,lt b,lt &x,lt &y)  
{  
 if(b==0){x=1;y=0;return a;}  
 lt gcd=exgcd(b,a%b,x,y);  
 lt tp=x;  
 x=y; y=tp-a/b\*y;  
 return gcd;  
}  
  
lt excrt()  
{  
 lt x,y,k;  
 lt M=bi[1],ans=ai[1];//第一个方程的解特判  
 for(int i=2;i<=n;i++)  
 {  
 lt a=M,b=bi[i],c=(ai[i]-ans%b+b)%b;//ax≡c(mod b)  
 lt gcd=exgcd(a,b,x,y),bg=b/gcd;  
 if(c%gcd!=0) return -1; //判断是否无解，然而这题其实不用  
   
 x=mul(x,c/gcd,bg);  
 ans+=x\*M;//更新前k个方程组的答案  
 M\*=bg;//M为前k个m的lcm  
 ans=(ans%M+M)%M;  
 }  
 return (ans%M+M)%M;  
}  
int main()  
{  
 n=read();  
 for(int i=1;i<=n;++i)  
 bi[i]=read(),ai[i]=read();  
 printf("%lld",excrt());  
 return 0;  
}

## 蔡勒公式

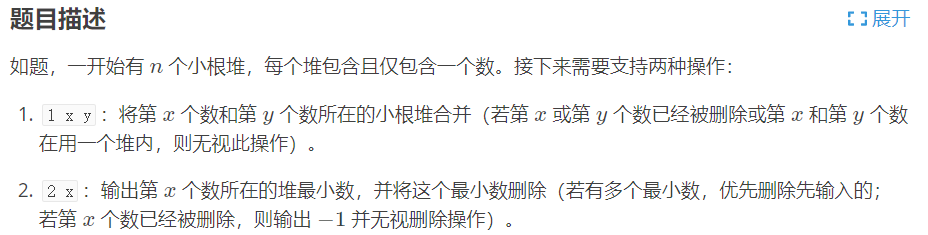
1582年10月4日或之前：w = y + y / 4 + c / 4 - 2 \* c + 13 \* (m + 1) / 5 + d + 2 取模  
之后：w = y + y / 4 + c / 4 - 2 \* c + 13 \* (m + 1) / 5 + d - 1 取模  
m为月份 如果m = 1 或者 2，要看作上一年的13，14月份来计算  
c为世纪 比如2012即为 c = 20 公元前253年 c = -3  
y一般为年份的后两位整数，如果是公元前的等于 c % 100 + 100

## 超级卡特兰数



# 数据结构

## 左偏树



#include <cstdio>  
#define N 100010  
using namespace std;  
int inline read()  
{  
 int x=0,f=1;char ch=getchar();  
 while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}  
 while(ch>='0'&&ch<='9'){x=x\*10+ch-'0';ch=getchar();}  
 return x\*f;  
}  
void swap(int &x,int &y){int t=x;x=y,y=t;}  
int ch[N][2],val[N],dis[N],f[N],n,m;  
int merge(int x,int y)  
{  
 if (x==0 || y==0)  
 return x+y;  
 if (val[x]>val[y] || (val[x]==val[y] && x>y))  
 swap(x,y);  
 ch[x][1]=merge(ch[x][1],y);  
 f[ch[x][1]]=x;  
 if (dis[ch[x][0]]<dis[ch[x][1]])  
 swap(ch[x][0],ch[x][1]);  
 dis[x]=dis[ch[x][1]]+1;  
 return x;  
}  
int getf(int x)  
{  
 while(f[x]) x=f[x];  
 return x;  
}  
void pop(int x)  
{  
 val[x]=-1;  
 f[ch[x][0]]=f[ch[x][1]]=0;  
 merge(ch[x][0],ch[x][1]);  
}  
main()  
{  
 n=read(),m=read();  
 dis[0]=-1;  
 for (int i=1;i<=n;i++)  
 val[i]=read();  
 for (int i=1;i<=m;i++)  
 {  
 int com=read();  
 if (com==1)  
 {  
 int x=read(),y=read();  
 if (val[x]==-1 || val[y]==-1)  
 continue;  
 if (x==y)  
 continue;  
 int fx=getf(x),fy=getf(y);  
 merge(fx,fy);  
 }  
 else  
 {  
 int x=read();  
 if (val[x]==-1)  
 puts("-1");  
 else  
 {  
 int y=getf(x);  
 printf("%d\n",val[y]);  
 pop(y);  
 }  
 }  
 }  
}

## 单调队列

#include <cstdio>  
#include <cstdlib>  
#include <cstring>  
#include <iostream>  
#define maxn 1000100  
using namespace std;  
int q[maxn], a[maxn];  
int n, k;  
void getmin() {  
 int head = 0, tail = 0;  
 for (int i = 1; i < k; i++) {  
 while (head <= tail && a[q[tail]] >= a[i]) tail--;  
 q[++tail] = i;  
 }  
 for (int i = k; i <= n; i++) {  
 while (head <= tail && a[q[tail]] >= a[i]) tail--;  
 q[++tail] = i;  
 while (q[head] <= i - k) head++;  
 printf("%d ", a[q[head]]);  
 }  
}  
  
void getmax() {  
 int head = 0, tail = 0;  
 for (int i = 1; i < k; i++) {  
 while (head <= tail && a[q[tail]] <= a[i]) tail--;  
 q[++tail] = i;  
 }  
 for (int i = k; i <= n; i++) {  
 while (head <= tail && a[q[tail]] <= a[i]) tail--;  
 q[++tail] = i;  
 while (q[head] <= i - k) head++;  
 printf("%d ", a[q[head]]);  
 }  
}  
  
int main() {  
 scanf("%d%d", &n, &k);  
 for (int i = 1; i <= n; i++) scanf("%d", &a[i]);  
 getmin();  
 printf("\n");  
 getmax();  
 printf("\n");  
 return 0;  
}

## ST表

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
int n, m, seed, seedx, seedy;  
int bin[25], dp[25][100005], Log[100005], val[100005];  
  
int get\_max\_num(int x, int y){  
 return val[x] >= val[y] ? x : y;  
}  
  
int main()  
{  
  
 scanf("%d %d %d %d %d", &n, &m, &seed, &seedx, &seedy);  
 bin[0] = 1;  
 for (int i = 1; i <= 20; i++) bin[i] = bin[i - 1] \* 2;  
 Log[0] = -1;  
 for (int i = 1; i <= n; i++) Log[i] = Log[i / 2] + 1;  
 unsigned x = seed;  
 for (int i = 1; i <= n; ++i){  
 x ^= x << 13;  
 x ^= x >> 17;  
 x ^= x << 5;  
 val[i] = x % 100;  
 dp[0][i] = i;  
 }  
 for (int i = 1; i <= Log[n]; i++){  
 for (int j = 1; j <= n; j++){  
 if(j + bin[i] - 1 <= n) dp[i][j] = get\_max\_num(dp[i - 1][j], dp[i - 1][j + bin[i - 1]]);  
 }  
 }  
 x = seedx;  
 unsigned y = seedy, lastans = 0;  
 int an = 0;  
 for (int i = 1; i <= m; ++i){  
 x ^= x << 13;  
 x ^= x >> 17;  
 x ^= x << 5;  
 y ^= y << 13;  
 y ^= y >> 17;  
 y ^= y << 5;  
 unsigned L = (x ^ lastans) % n + 1, R = (y ^ lastans) % n + 1;  
 if (L > R) swap(L, R);  
 int t = Log[R - L + 1];  
 lastans = get\_max\_num(dp[t][L], dp[t][R - bin[t] + 1]);  
 an ^= lastans;  
 }  
 printf("%d\n", an);  
  
 return 0;  
}  
  
/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
const int maxn = 2e7 + 5;  
const int num = 4500;  
  
int n, m, s;  
int bin[20], Log[4505];  
int a[maxn], l[maxn], r[maxn], dp[20][4505];  
  
namespace GenHelper  
{  
 unsigned z1, z2, z3, z4, b;  
 unsigned rand\_()  
 {  
 b = ((z1 << 6) ^ z1) >> 13;  
 z1 = ((z1 & 4294967294U) << 18) ^ b;  
 b = ((z2 << 2) ^ z2) >> 27;  
 z2 = ((z2 & 4294967288U) << 2) ^ b;  
 b = ((z3 << 13) ^ z3) >> 21;  
 z3 = ((z3 & 4294967280U) << 7) ^ b;  
 b = ((z4 << 3) ^ z4) >> 12;  
 z4 = ((z4 & 4294967168U) << 13) ^ b;  
 return (z1 ^ z2 ^ z3 ^ z4);  
 }  
}  
void srand(unsigned x)  
{  
 using namespace GenHelper;  
 z1 = x; z2 = (~x) ^ 0x233333333U; z3 = x ^ 0x1234598766U; z4 = (~x) + 51;  
}  
  
int read()  
{  
 using namespace GenHelper;  
 int a = rand\_() & 32767;  
 int b = rand\_() & 32767;  
 return a \* 32768 + b;  
}  
  
int ceil(int x){  
 return (x + num - 1) / num;  
}  
  
int solve(int x, int y){  
 int maxx = 0;  
 for (int i = x; i <= y; i++) maxx = max(maxx, a[i]);  
 return maxx;  
}  
  
int work(int x, int y){  
 int t = Log[y - x + 1];  
 return max(dp[t][x], dp[t][y - bin[t] + 1]);  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 bin[0] = 1, Log[0] = -1;  
 for (int i = 1; i <= 15; i++) bin[i] = bin[i - 1] << 1;  
 for (int i = 1; i <= 4500; i++) Log[i] = Log[i / 2] + 1;  
 scanf("%d %d %d", &n, &m, &s);  
 srand(s);  
 for (int i = 1, j = 1; i <= n; i++, j = ceil(i)) a[i] = read(), dp[0][j] = max(dp[0][j], a[i]);  
 int size = ceil(n);  
 for (int i = 1; i <= Log[size]; i++){  
 for (int j = 1; j + bin[i] - 1 <= size; j++){  
 dp[i][j] = max(dp[i - 1][j], dp[i - 1][j + bin[i - 1]]);  
 }  
 }  
 for (int i = 1; i <= n; i++){  
 if(i % num == 1) l[i] = a[i];  
 else l[i] = max(l[i - 1], a[i]);  
 }  
 for (int i = n; i >= 1; i--){  
 if(i % num == 0) r[i] = a[i];  
 else r[i] = max(r[i + 1], a[i]);  
 }  
 unsigned long long ans = 0;  
 for (int i = 1, x, y; i <= m; i++){  
 x = read() % n + 1, y = read() % n + 1;  
 // if(x <= 0 || y <= 0) {printf("%d %d\n", x, y);}  
 if(x > y) swap(x, y);  
 int num1 = ceil(x), num2 = ceil(y);  
 int maxx = max(r[x], l[y]);  
 if(num1 == num2) ans += solve(x, y);  
 else if(num2 - num1 == 1) ans += maxx;  
 else{  
 maxx = max(maxx, work(num1 + 1, num2 - 1));  
 ans += maxx;  
 }  
 }  
 printf("%llu\n", ans);  
  
 return 0;  
}  
/\*\*/

## 线段树

### 区间合并

/\*  
有一排n个座位，m次操作。A操作：将a名客人安置到最左的连续a个空位中，没有则不操作。L操作：[a,b]的客人离开。  
每个节点（区间）记录三个变量:  
  
sl[i]:从左往右有连续多少个空位  
  
sr[i]:从右往左有连续多少个空位  
  
s[i]:这个区间中最长连续多少个空位  
\*/  
/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
#define ls rt << 1  
#define rs rt << 1 | 1  
  
int n, m, s[500005 << 2], sl[500005 << 2], sr[500005 << 2], col[500005 << 2];  
char str[2];  
  
void up(int rt, int l, int r){  
 int mid = (l + r) >> 1;  
 s[rt] = max(max(s[ls], s[rs]), sr[ls] + sl[rs]);  
 sl[rt] = sl[ls] + (sl[ls] == mid - l + 1) \* sl[rs];  
 sr[rt] = sr[rs] + (sr[rs] == r - mid) \* sr[ls];  
}  
  
void down(int rt, int l, int r){  
 int mid = (l + r) >> 1;  
 if(col[rt]){  
 col[ls] = col[rs] = col[rt];  
 s[ls] = sl[ls] = sr[ls] = (col[rt] > 0) \* (mid - l + 1);  
 s[rs] = sl[rs] = sr[rs] = (col[rt] > 0) \* (r - mid);  
 col[rt] = 0;  
 }  
}  
  
void build(int rt, int l, int r){  
 s[rt] = sl[rt] = sr[rt] = r - l + 1;  
 if(l == r) return ;  
 int mid = (l + r) >> 1;  
 build(ls, l, mid);  
 build(rs, mid + 1, r);  
}  
  
void update(int rt, int l, int r, int L, int R, int x){  
 if(L <= l && r <= R){  
 s[rt] = sl[rt] = sr[rt] = (x > 0) \* (r - l + 1);  
 col[rt] = x;  
 return;  
 }  
 down(rt, l, r);  
 int mid = (l + r) >> 1;  
 if(L <= mid) update(ls, l, mid, L, R, x);  
 if(mid < R) update(rs, mid + 1, r, L, R, x);  
 up(rt, l, r);  
}  
  
int quary(int rt, int l, int r, int x){  
 down(rt, l, r);  
 int mid = (l + r) >> 1;  
 if(s[ls] >= x) return quary(ls, l, mid, x);  
 else if(sr[ls] + sl[rs] >= x) return mid - sr[ls] + 1;  
 else return quary(rs, mid + 1, r, x);  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 scanf("%d %d", &n, &m);  
 build(1, 1, n);  
 int ans = 0, l, r;  
 while(m--){  
 scanf("%s", str);  
 if(str[0] == 'A'){  
 scanf("%d", &l);  
 if(s[1] < l) ans++;  
 else{  
 int p = quary(1, 1, n, l);  
 update(1, 1, n, p, p + l - 1, -1);  
 }  
 }else{  
 scanf("%d %d", &l, &r);  
 update(1, 1, n, l, r, 1);  
 }  
 }  
 printf("%d\n", ans);  
  
 return 0;  
}  
/\*\*/

### 势能线段树

区间或区间与区间最小值

/\*  
https://loj.ac/problem/6507  
\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
const int maxn = 5e5 + 5;  
const int inf = (1LL << 31) - 1;  
  
int n, m, a[maxn], lzy\_or[maxn << 2], lzy\_and[maxn << 2];  
  
struct node  
{  
 int mn, \_or, \_and;  
}tr[maxn << 2];  
  
void up(int rt){  
 tr[rt].mn = min(tr[rt << 1].mn, tr[rt << 1 | 1].mn);  
 tr[rt].\_or = tr[rt << 1].\_or | tr[rt << 1 | 1].\_or;  
 tr[rt].\_and = tr[rt << 1].\_and & tr[rt << 1 | 1].\_and;  
}  
  
void down(int rt){  
 if(lzy\_or[rt]){  
 lzy\_or[rt << 1] |= lzy\_or[rt];  
 lzy\_and[rt << 1] |= lzy\_or[rt];  
 tr[rt << 1].mn |= lzy\_or[rt];  
 tr[rt << 1].\_or |= lzy\_or[rt];  
 tr[rt << 1].\_and |= lzy\_or[rt];  
 lzy\_or[rt << 1 | 1] |= lzy\_or[rt];  
 lzy\_and[rt << 1 | 1] |= lzy\_or[rt];  
 tr[rt << 1 | 1].mn |= lzy\_or[rt];  
 tr[rt << 1 | 1].\_or |= lzy\_or[rt];  
 tr[rt << 1 | 1].\_and |= lzy\_or[rt];  
 lzy\_or[rt] = 0;  
 }  
 if(lzy\_and[rt] != inf){  
 lzy\_or[rt << 1] &= lzy\_and[rt];  
 lzy\_and[rt << 1] &= lzy\_and[rt];  
 tr[rt << 1].mn &= lzy\_and[rt];  
 tr[rt << 1].\_or &= lzy\_and[rt];  
 tr[rt << 1].\_and &= lzy\_and[rt];  
 lzy\_and[rt << 1 | 1] &= lzy\_and[rt];  
 lzy\_or[rt << 1 | 1] &= lzy\_and[rt];  
 tr[rt << 1 | 1].mn &= lzy\_and[rt];  
 tr[rt << 1 | 1].\_or &= lzy\_and[rt];  
 tr[rt << 1 | 1].\_and &= lzy\_and[rt];  
 lzy\_and[rt] = inf;  
 }  
}  
  
void build(int rt, int l, int r){  
 lzy\_or[rt] = 0, lzy\_and[rt] = inf;  
 if(l == r){  
 tr[rt].mn = tr[rt].\_or = tr[rt].\_and = a[l];  
 return ;  
 }  
 int mid = (l + r) >> 1;  
 build(rt << 1, l, mid);  
 build(rt << 1 | 1, mid + 1, r);  
 up(rt);  
}  
  
void update\_and(int rt, int l, int r, int L, int R, int x){  
 if((tr[rt].\_or & x) == tr[rt].\_or) return ;  
 if(L <= l && r <= R && (tr[rt].\_and & x) == (tr[rt].\_or & x)){  
 tr[rt].\_and &= x;  
 tr[rt].\_or &= x;  
 lzy\_and[rt] &= x;  
 lzy\_or[rt] &= x;  
 tr[rt].mn &= x;  
 return ;  
 }  
 down(rt);  
 int mid = (l + r) >> 1;  
 if(mid >= L) update\_and(rt << 1, l, mid, L, R, x);  
 if(mid < R) update\_and(rt << 1 | 1, mid + 1, r, L, R, x);  
 up(rt);  
}  
  
void update\_or(int rt, int l, int r, int L, int R, int x){  
 if((tr[rt].\_and | x) == tr[rt].\_and) return ;  
 if(L <= l && r <= R && (tr[rt].\_and | x) == (tr[rt].\_or | x)){  
 tr[rt].\_and |= x;  
 tr[rt].\_or |= x;  
 lzy\_and[rt] |= x;  
 lzy\_or[rt] |= x;  
 tr[rt].mn |= x;  
 return ;  
 }  
 down(rt);  
 int mid = (l + r) >> 1;  
 if(mid >= L) update\_or(rt << 1, l, mid, L, R, x);  
 if(mid < R) update\_or(rt << 1 | 1, mid + 1, r, L, R, x);  
 up(rt);  
}  
  
int query(int rt, int l, int r, int L, int R){  
 if(L <= l && r <= R) return tr[rt].mn;  
 down(rt);  
 int mid = (l + r) >> 1, ans = inf;  
 if(mid >= L) ans = min(ans, query(rt << 1, l, mid, L, R));  
 if(mid < R) ans = min(ans, query(rt << 1 | 1, mid + 1, r, L, R));  
 return ans;  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 scanf("%d %d", &n, &m);  
 for (int i = 1; i <= n; i++) scanf("%d", &a[i]);  
 build(1, 1, n);  
 for (int i = 1, op, l, r, x; i <= m; i++){  
 scanf("%d %d %d", &op, &l, &r);  
 if(l > r) swap(l, r);  
 if(op == 3) printf("%d\n", query(1, 1, n, l, r));  
 else{  
 scanf("%d", &x);  
 if(op == 1) update\_and(1, 1, n, l, r, x);  
 else update\_or(1, 1, n, l, r, x);  
 }  
 }  
  
 return 0;  
}  
/\*\*/

## 李超树

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 > File Name: bzoj1568.cpp  
# File Name: bzoj1568.cpp  
# Author : xiaobuxie  
# QQ : 760427180  
# Email:760427180@qq.com  
# Created Time: 2019年09月03日 星期二 18时22分07秒  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
#include<iostream>  
#include<cstdio>  
#include<map>  
#include<cmath>  
#include<cstring>  
#include<set>  
#include<queue>  
#include<vector>  
#include<algorithm>  
using namespace std;  
typedef long long ll;  
#define inf 0x3f3f3f3f  
#define pq priority\_queue<int,vector<int>,greater<int> >  
ll gcd(ll a,ll b){  
 if(a<b) return gcd(b,a);  
 return b==0?a:gcd(b,a%b);  
}  
char s[10];  
int cnt=0;  
const int N=1e5+8;  
const int mx=5e4+9;  
struct line{  
 double k,b;  
}a[N];  
int tr[mx<<2];  
double f(int id,int x){  
 return a[id].k\*(x-1)+a[id].b;  
}  
void change(int o,int l,int r,int id){  
 if(l==r){  
 if( (f(id,l) > f(tr[o],l) )) tr[o]=id;  
 return;  
 }  
 int m=(l+r)>>1;  
 if( a[tr[o]].k < a[id].k){  
 if( f(id,m) > f(tr[o],m)){  
 change(o<<1,l,m,tr[o]);  
 tr[o]=id;  
 }  
 else change(o<<1|1,m+1,r,id);  
 }  
 if( a[tr[o]].k > a[id].k){  
 if( f(id,m) > f(tr[o],m) ){  
 change(o<<1|1,m+1,r,tr[o]);  
 tr[o]=id;  
 }  
 else change(o<<1,l,m,id);  
 }  
}  
double query(int o,int l,int r,int x){  
 if(l==r) return f(tr[o],x);  
 int m=(l+r)>>1;  
 if(x<=m) return max( f(tr[o],x), query(o<<1,l,m,x));  
 else return max( f(tr[o],x), query(o<<1|1,m+1,r,x));  
}  
  
int main(){  
 int n; scanf("%d",&n);  
 for(int i=1;i<=n;++i){  
 scanf("%s",s);  
 if(s[0]=='P'){  
 ++cnt;  
 scanf("%lf%lf",&a[cnt].b,&a[cnt].k);  
 change(1,1,mx-10,cnt);  
 }  
 else{  
 int x; scanf("%d",&x);  
 printf("%d\n",(int)(query(1,1,mx-10,x)/100));  
 }  
 }  
 return 0;  
}

## 主席树

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
int n, q, a[100005], b[100005], num[100005];  
int cnt, rt[100005];  
  
struct node  
{  
 int l, r;  
 LL w;  
}tr[100005 \* 20];  
  
void read(int &x){  
 char ch = getchar();x = 0;  
 for (; ch < '0' || ch > '9'; ch = getchar());  
 for (; ch >='0' && ch <= '9'; ch = getchar()) x = x \* 10 + ch - '0';  
}  
  
void cpy(int &now, int old){  
 now = ++cnt;  
 tr[now] = tr[old];  
}  
  
void up(int &now){  
 tr[now].w = tr[tr[now].l].w + tr[tr[now].r].w;  
}  
  
void build(int &now, int l, int r){  
 now = ++cnt;  
 tr[now].w = 0;  
 if(l == r) return ;  
 int mid = (l + r) >> 1;  
 build(tr[now].l, l, mid);  
 build(tr[now].r, mid + 1, r);  
}  
  
void update(int &now, int old, int l, int r, int x){  
 cpy(now, old);  
 if(l == r && x == l){  
 tr[now].w++;  
 return ;  
 }  
 int mid = (l + r) >> 1;  
 if(x <= mid) update(tr[now].l, tr[old].l, l, mid, x);  
 else update(tr[now].r, tr[old].r, mid + 1, r, x);  
 up(now);  
}  
  
int query(int s, int t, int l, int r, int k){  
 if(l == r) return l;  
 int mid = (l + r) >> 1;  
 int ans = tr[tr[t].l].w - tr[tr[s].l].w;  
 if(k <= ans) return query(tr[s].l, tr[t].l, l, mid, k);  
 else return query(tr[s].r, tr[t].r, mid + 1, r, k - ans);  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 while(scanf("%d %d", &n, &q) == 2){  
 for (int i = 0; i < n; i++){  
 read(a[i]);  
 b[i] = a[i];  
 }  
 sort(b, b + n);  
 int pos = unique(b, b + n) - b;  
 for (int i = 0; i < n; i++) a[i] = lower\_bound(b, b + pos, a[i]) - b + 1;  
 cnt = 0;  
 build(rt[0], 1, pos);  
 for (int i = 1; i <= n; i++) update(rt[i], rt[i - 1], 1, pos, a[i - 1]);  
 for (int i = 1, l, r; i <= q; i++){  
 read(l), read(r);  
 LL ans = -1;  
 int tot = 0;  
 for (int j = r - l + 1; j >= max(1, r - l - 43); j--){  
 num[j] = b[query(rt[l - 1], rt[r], 1, pos, j) - 1];  
 if(++tot >= 3 && num[j] + num[j + 1] > num[j + 2]) {ans = (LL)num[j] + num[j + 1] + num[j + 2]; break;}  
 }  
 printf("%lld\n", ans);  
 }  
 }  
  
 return 0;  
}  
/\*\*/

## Treap

treap 是一种弱平衡的二叉搜索树。treap 这个单词是 tree 和 heap 的组合，表明 treap 是一种由树和堆组合形成的数据结构。treap 的每个结点上要额外储存一个值 priority。treap 除了要满足二叉搜索树的性质之外，还需满足父节点的priority大于等于两个儿子的priority。而priority是每个结点建立时随机生成的，因此 treap 是期望平衡的。

treap 分为旋转式和无旋式两种。两种 treap 都易于编写，但无旋式 treap 的操作方式使得它天生支持维护序列、可持久化等特性。这里以重新实现 set<int> （不可重集合）为例，介绍无旋式 treap。

#include <algorithm>  
#include <cstdio>  
#include <iostream>  
  
#define maxn 100005  
#define INF (1 << 30)  
  
int n;  
  
struct treap {  
 int l[maxn], r[maxn], val[maxn], rnd[maxn], size[maxn], w[maxn];  
 int sz, ans, rt;  
 inline void pushup(int x) { size[x] = size[l[x]] + size[r[x]] + w[x]; }  
 void lrotate(int &k) {  
 int t = r[k];  
 r[k] = l[t];  
 l[t] = k;  
 size[t] = size[k];  
 pushup(k);  
 k = t;  
 }  
 void rrotate(int &k) {  
 int t = l[k];  
 l[k] = r[t];  
 r[t] = k;  
 size[t] = size[k];  
 pushup(k);  
 k = t;  
 }  
 void insert(int &k, int x) {  
 if (!k) {  
 sz++;  
 k = sz;  
 size[k] = 1;  
 w[k] = 1;  
 val[k] = x;  
 rnd[k] = rand();  
 return;  
 }  
 size[k]++;  
 if (val[k] == x) {  
 w[k]++;  
 } else if (val[k] < x) {  
 insert(r[k], x);  
 if (rnd[r[k]] < rnd[k]) lrotate(k);  
 } else {  
 insert(l[k], x);  
 if (rnd[l[k]] < rnd[k]) rrotate(k);  
 }  
 }  
  
 void del(int &k, int x) {  
 if (!k) return;  
 if (val[k] == x) {  
 if (w[k] > 1) {  
 w[k]--;  
 size[k]--;  
 return;  
 }  
 if (l[k] == 0 || r[k] == 0)  
 k = l[k] + r[k];  
 else if (rnd[l[k]] < rnd[r[k]]) {  
 rrotate(k);  
 del(k, x);  
 } else {  
 lrotate(k);  
 del(k, x);  
 }  
 } else if (val[k] < x) {  
 size[k]--;  
 del(r[k], x);  
 } else {  
 size[k]--;  
 del(l[k], x);  
 }  
 }  
  
 int queryrank(int k, int x) {  
 if (!k) return 0;  
 if (val[k] == x)  
 return size[l[k]] + 1;  
 else if (x > val[k]) {  
 return size[l[k]] + w[k] + queryrank(r[k], x);  
 } else  
 return queryrank(l[k], x);  
 }  
  
 int querynum(int k, int x) {  
 if (!k) return 0;  
 if (x <= size[l[k]])  
 return querynum(l[k], x);  
 else if (x > size[l[k]] + w[k])  
 return querynum(r[k], x - size[l[k]] - w[k]);  
 else  
 return val[k];  
 }  
  
 void querypre(int k, int x) {  
 if (!k) return;  
 if (val[k] < x)  
 ans = k, querypre(r[k], x);  
 else  
 querypre(l[k], x);  
 }  
  
 void querysub(int k, int x) {  
 if (!k) return;  
 if (val[k] > x)  
 ans = k, querysub(l[k], x);  
 else  
 querysub(r[k], x);  
 }  
} T;  
  
int main() {  
 srand(123);  
 scanf("%d", &n);  
 int opt, x;  
 for (int i = 1; i <= n; i++) {  
 scanf("%d%d", &opt, &x);  
 if (opt == 1)  
 T.insert(T.rt, x);  
 else if (opt == 2)  
 T.del(T.rt, x);  
 else if (opt == 3) {  
 printf("%d\n", T.queryrank(T.rt, x));  
 } else if (opt == 4) {  
 printf("%d\n", T.querynum(T.rt, x));  
 } else if (opt == 5) {  
 T.ans = 0;  
 T.querypre(T.rt, x);  
 printf("%d\n", T.val[T.ans]);  
 } else if (opt == 6) {  
 T.ans = 0;  
 T.querysub(T.rt, x);  
 printf("%d\n", T.val[T.ans]);  
 }  
 }  
 return 0;  
}

## Splay

#include <cstdio>  
const int N = 100005;  
int rt, tot, fa[N], ch[N][2], val[N], cnt[N], sz[N];  
struct Splay {  
 void maintain(int x) { sz[x] = sz[ch[x][0]] + sz[ch[x][1]] + cnt[x]; }  
 bool get(int x) { return x == ch[fa[x]][1]; }  
 void clear(int x) {  
 ch[x][0] = ch[x][1] = fa[x] = val[x] = sz[x] = cnt[x] = 0;  
 }  
 void rotate(int x) {  
 int y = fa[x], z = fa[y], chk = get(x);  
 ch[y][chk] = ch[x][chk ^ 1];  
 fa[ch[x][chk ^ 1]] = y;  
 ch[x][chk ^ 1] = y;  
 fa[y] = x;  
 fa[x] = z;  
 if (z) ch[z][y == ch[z][1]] = x;  
 maintain(x);  
 maintain(y);  
 }  
 void splay(int x) {  
 for (int f = fa[x]; f = fa[x], f; rotate(x))  
 if (fa[f]) rotate(get(x) == get(f) ? f : x);  
 rt = x;  
 }  
 void ins(int k) {  
 if (!rt) {  
 val[++tot] = k;  
 cnt[tot]++;  
 rt = tot;  
 maintain(rt);  
 return;  
 }  
 int cnr = rt, f = 0;  
 while (1) {  
 if (val[cnr] == k) {  
 cnt[cnr]++;  
 maintain(cnr);  
 maintain(f);  
 splay(cnr);  
 break;  
 }  
 f = cnr;  
 cnr = ch[cnr][val[cnr] < k];  
 if (!cnr) {  
 val[++tot] = k;  
 cnt[tot]++;  
 fa[tot] = f;  
 ch[f][val[f] < k] = tot;  
 maintain(tot);  
 maintain(f);  
 splay(tot);  
 break;  
 }  
 }  
 }  
 int rk(int k) {  
 int res = 0, cnr = rt;  
 while (1) {  
 if (k < val[cnr]) {  
 cnr = ch[cnr][0];  
 } else {  
 res += sz[ch[cnr][0]];  
 if (k == val[cnr]) {  
 splay(cnr);  
 return res + 1;  
 }  
 res += cnt[cnr];  
 cnr = ch[cnr][1];  
 }  
 }  
 }  
 int kth(int k) {  
 int cnr = rt;  
 while (1) {  
 if (ch[cnr][0] && k <= sz[ch[cnr][0]]) {  
 cnr = ch[cnr][0];  
 } else {  
 k -= cnt[cnr] + sz[ch[cnr][0]];  
 if (k <= 0) {  
 splay(cnr);  
 return val[cnr];  
 }  
 cnr = ch[cnr][1];  
 }  
 }  
 }  
 int pre() {  
 int cnr = ch[rt][0];  
 while (ch[cnr][1]) cnr = ch[cnr][1];  
 splay(cnr);  
 return cnr;  
 }  
 int nxt() {  
 int cnr = ch[rt][1];  
 while (ch[cnr][0]) cnr = ch[cnr][0];  
 splay(cnr);  
 return cnr;  
 }  
 void del(int k) {  
 rk(k);  
 if (cnt[rt] > 1) {  
 cnt[rt]--;  
 maintain(rt);  
 return;  
 }  
 if (!ch[rt][0] && !ch[rt][1]) {  
 clear(rt);  
 rt = 0;  
 return;  
 }  
 if (!ch[rt][0]) {  
 int cnr = rt;  
 rt = ch[rt][1];  
 fa[rt] = 0;  
 clear(cnr);  
 return;  
 }  
 if (!ch[rt][1]) {  
 int cnr = rt;  
 rt = ch[rt][0];  
 fa[rt] = 0;  
 clear(cnr);  
 return;  
 }  
 int cnr = rt;  
 int x = pre();  
 fa[ch[cnr][1]] = x;  
 ch[x][1] = ch[cnr][1];  
 clear(cnr);  
 maintain(rt);  
 }  
} tree;  
  
int main() {  
 int n, opt, x;  
 for (scanf("%d", &n); n; --n) {  
 scanf("%d%d", &opt, &x);  
 if (opt == 1)  
 tree.ins(x);  
 else if (opt == 2)  
 tree.del(x);  
 else if (opt == 3)  
 printf("%d\n", tree.rk(x));  
 else if (opt == 4)  
 printf("%d\n", tree.kth(x));  
 else if (opt == 5)  
 tree.ins(x), printf("%d\n", val[tree.pre()]), tree.del(x);  
 else  
 tree.ins(x), printf("%d\n", val[tree.nxt()]), tree.del(x);  
 }  
 return 0;  
}

## WBLT

WBLT，全称 Weight Balanced Leafy Tree，一种不常见的平衡树写法，但是具有常数较小，可以当做可并堆使用的优点。

类似于 WBT（weight-balanced trees），WBLT 体现了 leafy 的性质，即节点多，怎么多呢？

对于 n 个数，不同于 treap 等，WBLT 会建立 2n 个节点，每个节点的权值为其右儿子的权值，且右儿子的权值大于等于左儿子

每次插入，类似于堆，逐次向下交换并向上 pushup 更新即可，删除也是同理

当然，如果输入数据递增或递减，WBLT 会退化成链状，于是我们采用旋转来维护平衡。

因为 WBLT 同时满足堆的性质，我们可以用它来实现堆和可并堆。

而在旋转的过程中，会产生很多垃圾节点，我们采用垃圾回收的方式就可以回收废弃节点，将建立节点的操作稍作修改即可。

附上普通平衡树代码：

#include <iostream>  
  
using namespace std;  
  
const int maxn = 400005;  
  
const int ratio = 5;  
int n, cnt, fa, root;  
int size[maxn], ls[maxn], rs[maxn], val[maxn];  
  
void newnode(int &cur, int v) {  
 cur = ++cnt;  
 size[cur] = 1;  
 val[cur] = v;  
}  
void copynode(int x, int y) {  
 size[x] = size[y];  
 ls[x] = ls[y];  
 rs[x] = rs[y];  
 val[x] = val[y];  
}  
void merge(int l, int r) {  
 size[++cnt] = size[l] + size[r];  
 val[cnt] = val[r];  
 ls[cnt] = l, rs[cnt] = r;  
}  
void rotate(int cur, bool flag) {  
 if (flag) {  
 merge(ls[cur], ls[rs[cur]]);  
 ls[cur] = cnt;  
 rs[cur] = rs[rs[cur]];  
 } else {  
 merge(rs[ls[cur]], rs[cur]);  
 rs[cur] = cnt;  
 ls[cur] = ls[ls[cur]];  
 }  
}  
void maintain(int cur) {  
 if (size[ls[cur]] > size[rs[cur]] \* ratio)  
 rotate(cur, 0);  
 else if (size[rs[cur]] > size[ls[cur]] \* ratio)  
 rotate(cur, 1);  
 if (size[ls[cur]] > size[rs[cur]] \* ratio)  
 rotate(ls[cur], 1), rotate(cur, 0);  
 else if (size[rs[cur]] > size[ls[cur]] \* ratio)  
 rotate(rs[cur], 0), rotate(cur, 1);  
}  
void pushup(int cur) {  
 if (!size[ls[cur]]) return;  
 size[cur] = size[ls[cur]] + size[rs[cur]];  
 val[cur] = val[rs[cur]];  
}  
void insert(int cur, int x) {  
 if (size[cur] == 1) {  
 newnode(ls[cur], min(x, val[cur]));  
 newnode(rs[cur], max(x, val[cur]));  
 pushup(cur);  
 return;  
 }  
 maintain(cur);  
 insert(x > val[ls[cur]] ? rs[cur] : ls[cur], x);  
 pushup(cur);  
}  
void erase(int cur, int x) {  
 if (size[cur] == 1) {  
 cur = ls[fa] == cur ? rs[fa] : ls[fa];  
 copynode(fa, cur);  
 return;  
 }  
 maintain(cur);  
 fa = cur;  
 erase(x > val[ls[cur]] ? rs[cur] : ls[cur], x);  
 pushup(cur);  
}  
int find(int cur, int x) {  
 if (size[cur] == x) return val[cur];  
 maintain(cur);  
 if (x > size[ls[cur]]) return find(rs[cur], x - size[ls[cur]]);  
 return find(ls[cur], x);  
}  
int rnk(int cur, int x) {  
 if (size[cur] == 1) return 1;  
 maintain(cur); // asdasdasdasd  
 if (x > val[ls[cur]]) return rnk(rs[cur], x) + size[ls[cur]];  
 return rnk(ls[cur], x);  
}  
int main() {  
 scanf("%d", &n);  
 newnode(root, 2147383647); // 使根不改变  
 while (n--) {  
 int s, a;  
 scanf("%d %d", &s, &a);  
 if (s == 1) insert(root, a);  
 if (s == 2) erase(root, a);  
 if (s == 3) printf("%d\n", rnk(root, a));  
 if (s == 4) printf("%d\n", find(root, a));  
 if (s == 5) printf("%d\n", find(root, rnk(root, a) - 1));  
 if (s == 6) printf("%d\n", find(root, rnk(root, a + 1)));  
 }  
 return 0;  
}

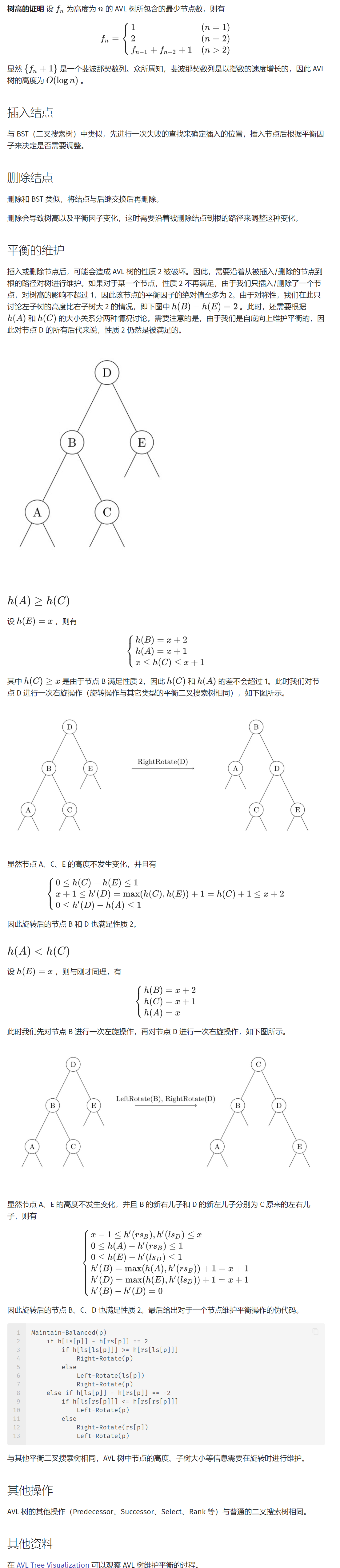
## AVL 树

AVL 树，是一种平衡的二叉搜索树。由于各种算法教材上对 AVL 的介绍十分冗长，造成了很多人对 AVL 树复杂、不实用的印象。但实际上，AVL 树的原理简单，实现也并不复杂。

性质:

1. 空二叉树是一个 AVL 树
2. 如果 T 是一棵 AVL 树，那么其左右子树也是 AVL 树，并且|h(ls) - h(rs)|<= 1 ，h 是其左右子树的高度
3. 树高为 O(logn)

平衡因子：右子树高度 - 左子树高度

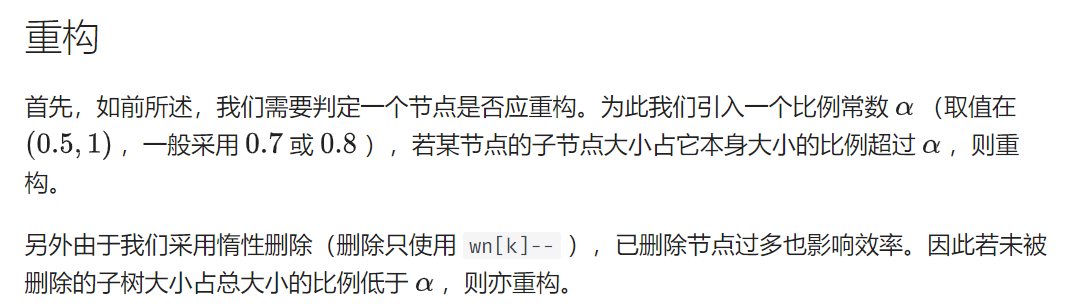


## 替罪羊树

**替罪羊树** 是一种依靠重构操作维持平衡的重量平衡树。替罪羊树会在插入、删除操作时，检测途经的节点，若发现失衡，则将以该节点为根的子树重构。

我们在此实现一个可重的权值平衡树。

int cnt, // 树中元素总数  
 rt, // 根节点，初值为 0 代表空树  
 w[MAXN], // 点中的数据 / 权值  
 lc[MAXN], rc[MAXN], // 左右子树  
 wn[MAXN], // 本数据出现次数（为 0 代表已删除）  
 s[MAXN], // 以本节点为根的子树大小  
 sd[MAXN]; // 已删除节点不计的子树大小  
  
void Calc(int k) {  
 // 重新计算以 k 为根的子树大小  
 s[k] = s[lc[k]] + s[rc[k]] + 1;  
 sd[k] = sd[lc[k]] + sd[rc[k]] + wn[k];  
}



inline bool CanRbu(int k) {  
 // 判断节点 k 是否需要重构  
 return wn[k] && (alpha \* s[k] <= (double)std::max(s[lc[k]], s[rc[k]]) ||  
 (double)sd[k] <= alpha \* s[k]);  
}

重构分为两个步骤——先中序遍历展开存入数组，再二分重建成树。

void Rbu\_Flatten(int& ldc, int k) {  
 // 中序遍历展开以 k 节点为根子树  
 if (!k) return;  
 Rbu\_Flatten(ldc, lc[k]);  
 if (wn[k]) ldr[ldc++] = k;  
 // 若当前节点已删除则不保留  
 Rbu\_Flatten(ldc, rc[k]);  
}  
  
int Rbu\_Build(int l, int r) {  
 // 将 ldr[] 数组内 [l, r) 区间重建成树，返回根节点  
 int mid = l + r >> 1; // 选取中间为根使其平衡  
 if (l >= r) return 0;  
 lc[ldr[mid]] = Rbu\_Build(l, mid);  
 rc[ldr[mid]] = Rbu\_Build(mid + 1, r); // 建左右子树  
 Calc(ldr[mid]);  
 return ldr[mid];  
}  
  
void Rbu(int& k) {  
 // 重构节点 k 的全过程  
 int ldc = 0;  
 Rbu\_Flatten(ldc, k);  
 k = Rbu\_Build(0, ldc);  
}

基本操作

几种操作的处理方式较为类似，都规定了 **到达空结点** 与 **找到对应结点** 的行为，之后按 **小于向左、大于向右** 的方式向下递归。

### 插入

插入时，到达空结点则新建节点，找到对应结点则 wn[k]++ 。递归结束后，途经的节点可重构的要重构。

void Ins(int& k, int p) {  
 // 在以 k 为根的子树内添加权值为 p 节点  
 if (!k) {  
 k = ++cnt;  
 if (!rt) rt = 1;  
 w[k] = p;  
 lc[k] = rc[k] = 0;  
 wn[k] = s[k] = sd[k] = 1;  
 } else {  
 if (w[k] == p)  
 wn[k]++;  
 else if (w[k] < p)  
 Ins(rc[k], p);  
 else  
 Ins(lc[k], p);  
 Calc(k);  
 if (CanRbu(k)) Rbu(k);  
 }  
}

### 删除

惰性删除，到达空结点则忽略，找到对应结点则 wn[k]-- 。递归结束后，可重构节点要重构。

void Del(int& k, int p) {  
 // 从以 k 为根子树移除权值为 p 节点  
 if (!k)  
 return;  
 else {  
 if (w[k] == p) {  
 if (wn[k]) wn[k]--;  
 } else {  
 if (w[k] < p)  
 Del(rc[k], p);  
 else  
 Del(lc[k], p);  
 }  
 Calc(k);  
 if (CanRbu(k)) Rbu(k);  
 }  
}

### upper\_bound

返回权值严格大于某值的最小名次。

到达空结点则返回 1，因为只有该子树左边的数均小于查找数才会递归至此。找到对应结点，则返回该节点所占据的最后一个名次 + 1。

int MyUprBd(int k, int p) {  
 // 在以 k 为根子树中，大于 p 的最小数的名次  
 if (!k)  
 return 1;  
 else if (w[k] == p && wn[k])  
 return sd[lc[k]] + 1 + wn[k];  
 else if (p < w[k])  
 return MyUprBd(lc[k], p);  
 else  
 return sd[lc[k]] + wn[k] + MyUprBd(rc[k], p);  
}

以下是反义函数，相当于采用 std::greater<> 比较，即返回权值严格小于某值的最大名次。查询一个数的排名可以用 MyUprGrt(rt, x) + 1 。

int MyUprGrt(int k, int p) {  
 if (!k)  
 return 0;  
 else if (w[k] == p && wn[k])  
 return sd[lc[k]];  
 else if (w[k] < p)  
 return sd[lc[k]] + wn[k] + MyUprGrt(rc[k], p);  
 else  
 return MyUprGrt(lc[k], p);  
}

### at

给定名次，返回该名次上的权值。到达空结点说明无此名次，找到对应结点则返回其权值。

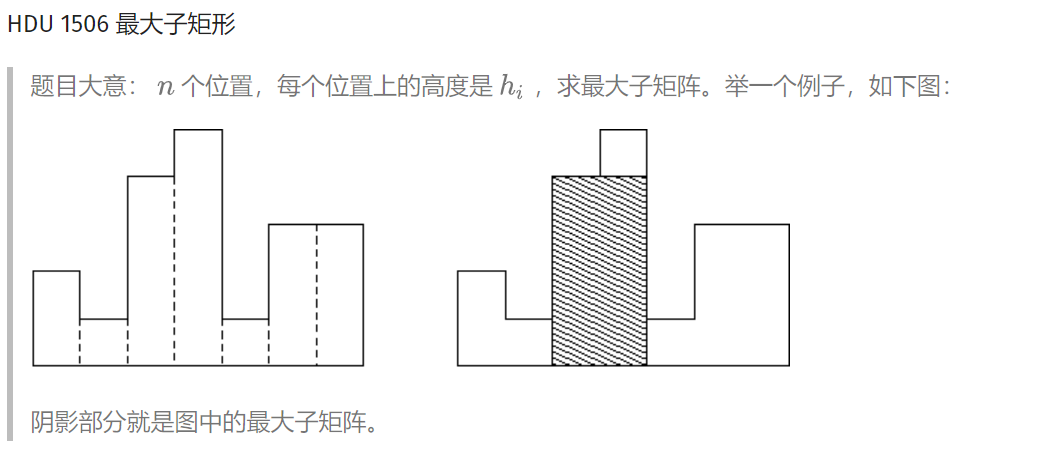
int MyAt(int k, int p) {  
 // 以 k 为根的子树中，名次为 p 的权值  
 if (!k)  
 return 0;  
 else if (sd[lc[k]] < p && p <= sd[lc[k]] + wn[k])  
 return w[k];  
 else if (sd[lc[k]] + wn[k] < p)  
 return MyAt(rc[k], p - sd[lc[k]] - wn[k]);  
 else  
 return MyAt(lc[k], p);  
}

### 前驱后继

以上两种功能结合即可。

inline int MyPre(int k, int p) { return MyAt(k, MyUprGrt(k, p)); }  
inline int MyPost(int k, int p) { return MyAt(k, MyUprBd(k, p)); }

## 笛卡尔树



#include <algorithm>  
#include <cstdio>  
#include <cstring>  
#include <iostream>  
using namespace std;  
typedef long long ll;  
const int N = 100000 + 10, INF = 0x3f3f3f3f;  
  
struct node {  
 int idx, val, par, ch[2];  
 friend bool operator<(node a, node b) { return a.idx < b.idx; }  
 void init(int \_idx, int \_val, int \_par) {  
 idx = \_idx, val = \_val, par = \_par, ch[0] = ch[1] = 0;  
 }  
} tree[N];  
  
int root, top, stk[N];  
ll ans;  
int cartesian\_build(int n) {  
 for (int i = 1; i <= n; i++) {  
 int k = i - 1;  
 while (tree[k].val > tree[i].val) k = tree[k].par;  
 tree[i].ch[0] = tree[k].ch[1];  
 tree[k].ch[1] = i;  
 tree[i].par = k;  
 tree[tree[i].ch[0]].par = i;  
 }  
 return tree[0].ch[1];  
}  
int dfs(int x) {  
 if (!x) return 0;  
 int sz = dfs(tree[x].ch[0]);  
 sz += dfs(tree[x].ch[1]);  
 ans = max(ans, (ll)(sz + 1) \* tree[x].val);  
 return sz + 1;  
}  
int main() {  
 int n, hi;  
 while (scanf("%d", &n), n) {  
 tree[0].init(0, 0, 0);  
 for (int i = 1; i <= n; i++) {  
 scanf("%d", &hi);  
 tree[i].init(i, hi, 0);  
 }  
 root = cartesian\_build(n);  
 ans = 0;  
 dfs(root);  
 printf("%lld\n", ans);  
 }  
 return 0;  
}

## 可持久化数组

如题，你需要维护这样的一个长度为 N*N* 的数组，支持如下几种操作

1. 在某个历史版本上修改某一个位置上的值
2. 访问某个历史版本上的某一位置的值

此外，每进行一次操作（**对于操作2，即为生成一个完全一样的版本，不作任何改动**），就会生成一个新的版本。版本编号即为当前操作的编号（从1开始编号，版本0表示初始状态数组）

#include<iostream>  
#include<cstdio>  
#include<algorithm>  
#define mid ((l+r)>>1)  
using namespace std;  
struct chairman{  
 //参见空间复杂度，要开大数组  
 int rt[1000001],T[20000001],L[20000001],R[20000001];  
 int cnt;//尾节点，插入节点用  
 int build(int l,int r){  
 int root=++cnt;  
 if(l==r){  
 scanf("%d",&T[root]);return root;  
 //建树与读入合二为一  
 }  
 //递归建子区间  
 L[root]=build(l,mid);R[root]=build(mid+1,r);  
 return root;  
 }  
 int update(int pre,int l,int r,int &x,int &c){  
 int root=++cnt;  
 if(l==r){  
 T[root]=c;return root;//修改  
 }  
 L[root]=L[pre];R[root]=R[pre];//先把子节点指向前驱结点以备复用  
 //递归修改子区间  
 if(x<=mid)L[root]=update(L[pre],l,mid,x,c);  
 else R[root]=update(R[pre],mid+1,r,x,c);  
 return root;  
 }  
 void query(int pre,int l,int r,int& x){  
 //普通的线段树查询  
 if(l==r){  
 printf("%d\n",T[pre]);return;  
 }  
 if(x<=mid)query(L[pre],l,mid,x);  
 else query(R[pre],mid+1,r,x);  
 }  
}hsf; //某本校大佬的名字  
int main(){  
 hsf.cnt=0;  
 int n,m;  
 scanf("%d%d",&n,&m);  
 hsf.build(1,n);  
 int v,cd,x,y;  
 hsf.rt[0]=1;//第零版本  
 for(int i=1;i<=m;++i){  
 scanf("%d%d%d",&v,&cd,&x);  
 if(cd==1){  
 scanf("%d",&y);  
 hsf.rt[i]=hsf.update(hsf.rt[v],1,n,x,y);//修改  
 }  
 if(cd==2){  
 hsf.rt[i]=hsf.rt[v];  
 hsf.query(hsf.rt[v],1,n,x);//查询  
 }  
 }  
 return 0;  
}

## 可持久化平衡树

您需要写一种数据结构（可参考题目标题），来维护一个可重整数集合，其中需要提供以下操作（ **对于各个以往的历史版本** ）：

1、 插入 x

2、 删除 x（若有多个相同的数，应只删除一个，**如果没有请忽略该操作**）

3、 查询 x 的排名（排名定义为比当前数小的数的个数 +1）

4、查询排名为 x的数

5、 求 x的前驱（前驱定义为小于 x，且最大的数，**如不存在输出** -2^{31}+1）

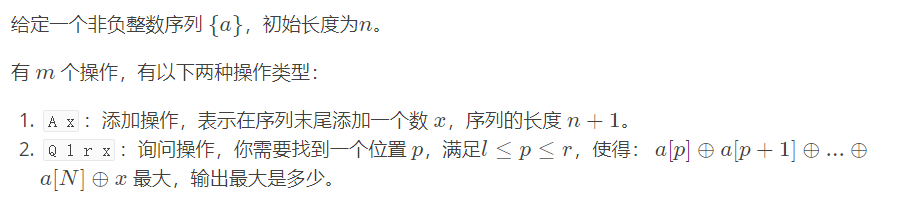
6、求 x的后继（后继定义为大于 x，且最小的数，**如不存在输出** 2^{31}-1）

**和原本平衡树不同的一点是，每一次的任何操作都是基于某一个历史版本，同时生成一个新的版本。（操作3, 4, 5, 6即保持原版本无变化）**

每个版本的编号即为操作的序号（版本0即为初始状态，空树）

#include<bits/stdc++.h>  
using namespace std;  
struct node  
{  
 int l,r;int size,rnd,v;  
}t[500005\*50];  
int cnt,rt[500005];  
void update(int k)  
{  
 t[k].size=t[t[k].l].size+t[t[k].r].size+1;  
}  
void newnode(int &k,int x)  
{  
 t[k=++cnt].v=x;t[k].size=1;t[k].rnd=rand();  
}  
int merge(int a,int b)  
{  
 if(!a||!b)return a+b;  
 if(t[a].rnd>t[b].rnd)  
 {  
 int p=++cnt;t[p]=t[a];  
 t[p].r=merge(t[p].r,b);  
 update(p);return p;  
 }  
 else  
 {  
 int p=++cnt;t[p]=t[b];  
 t[p].l=merge(a,t[p].l);  
 update(p);return p;  
 }  
}  
void split(int now,int k,int &x,int &y)  
{  
 if(!now)x=y=0;  
 else  
 {  
 if(t[now].v<=k)  
 {  
 x=++cnt;t[x]=t[now];  
 split(t[x].r,k,t[x].r,y);  
 update(x);  
 }  
 else   
 {  
 y=++cnt;t[y]=t[now];  
 split(t[y].l,k,x,t[y].l);  
 update(y);  
 }  
 }  
}  
void Delete(int &root,int w)  
{  
 int x=0,y=0,z=0;  
 split(root,w,x,z);  
 split(x,w-1,x,y);  
 y=merge(t[y].l,t[y].r);  
 root=merge(merge(x,y),z);  
}  
void Insert(int &root,int w)  
{  
 int x=0,y=0,z=0;  
 split(root,w,x,y);  
 newnode(z,w);  
 root=merge(merge(x,z),y);  
}  
int getval(int k,int w)  
{  
 if(w==t[t[k].l].size+1)return t[k].v;  
 else if(w<=t[t[k].l].size)return getval(t[k].l,w);  
 else return getval(t[k].r,w-t[t[k].l].size-1);  
}  
int getkth(int &root,int w)  
{  
 int x,y;  
 split(root,w-1,x,y);  
 int ans=t[x].size+1;  
 root=merge(x,y);  
 return ans;  
}  
int getpre(int &root,int w)  
{  
 int x,y,k,ans;  
 split(root,w-1,x,y);  
 if(!x)return -2147483647;  
 k=t[x].size;  
 ans=getval(x,k);  
 root=merge(x,y);  
 return ans;  
}  
int getnex(int &root,int w)  
{  
 int x,y,ans;  
 split(root,w,x,y);  
 if(!y)return 2147483647;  
 else ans=getval(y,1);  
 root=merge(x,y);  
 return ans;  
}  
int main()  
{  
 int n,f,w,tim;  
 scanf("%d",&n);  
 for(int i=1;i<=n;++i)  
 {  
 scanf("%d%d%d",&tim,&f,&w);  
 rt[i]=rt[tim];  
 if(f==1)Insert(rt[i],w);  
 else if(f==2)Delete(rt[i],w);  
 else if(f==3)printf("%d\n",getkth(rt[i],w));  
 else if(f==4)printf("%d\n",getval(rt[i],w));  
 else if(f==5)printf("%d\n",getpre(rt[i],w));  
 else printf("%d\n",getnex(rt[i],w));  
 }  
 return 0;  
}

## 可持久化字典树



/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
int n, m, sum, sz;  
int rt[600005], tr[600005 \* 24][2], cnt[600005 \* 24];  
char s[2];  
  
void insert(int now, int old, int t, int w){  
 if(t < 0) return ;  
 int x = (w >> t) & 1;  
 tr[now][!x] = tr[old][!x];  
 tr[now][x] = ++sz;  
 cnt[sz] = cnt[tr[old][x]] + 1;  
 insert(tr[now][x], tr[old][x], t - 1, w);  
}  
  
int query(int now, int old, int t, int w){  
 if(t < 0) return 0;  
 int x = (w >> t) & 1;  
 if(cnt[tr[now][!x]] < cnt[tr[old][!x]]) return (1 << t) + query(tr[now][!x], tr[old][!x], t - 1, w);  
 return query(tr[now][x], tr[old][x], t - 1, w);  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 scanf("%d %d", &n, &m);  
 rt[0] = ++sz;  
 insert(rt[0], 0, 24, 0);  
 for (int i = 1, x; i <= n; i++){  
 scanf("%d", &x);  
 rt[i] = ++sz;  
 sum ^= x;  
 insert(rt[i], rt[i - 1], 24, sum);  
   
 }  
 for (int i = 1, l, r, x; i <= m; i++){  
 scanf("%s", s);  
 if(s[0] == 'A'){  
 scanf("%d", &x);  
 sum ^= x;  
 rt[++n] = ++sz;  
 insert(rt[n], rt[n - 1], 24, sum);  
 }else{  
 scanf("%d %d %d", &l, &r, &x);  
 l--, r--;  
 if(!l) printf("%d\n", query(0, rt[r], 24, sum ^ x));  
 else printf("%d\n", query(rt[l - 1], rt[r], 24, sum ^ x));  
 }  
 }  
  
 return 0;  
}  
/\*\*/

## 线段树套线段树

第二维查询

int tree\_query(int k, int l, int r, int x) {  
 if (k == 0) return 0;  
 if (1 <= l && r <= sec[x].y) return vec\_query(ou\_root[k], 1, p, 1, sec[x].z);  
 int mid = l + r >> 1, res = 0;  
 if (1 <= mid) res += tree\_query(ou\_ch[k][0], l, mid, x);  
 if (sec[x].y > mid) res += tree\_query(ou\_ch[k][1], mid + 1, r, x);  
 return res;  
}

第二维修改

void tree\_insert(int &k, int l, int r, int x) {  
 if (k == 0) k = ++ou\_tot;  
 vec\_insert(ou\_root[k], 1, p, sec[x].z);  
 if (l == r) return;  
 int mid = l + r >> 1;  
 if (sec[x].y <= mid)  
 tree\_insert(ou\_ch[k][0], l, mid, x);  
 else  
 tree\_insert(ou\_ch[k][1], mid + 1, r, x);  
}

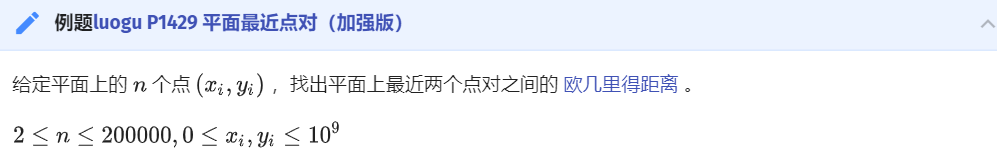
第三维查询

int vec\_query(int k, int l, int r, int x, int y) {  
 if (k == 0) return 0;  
 if (x <= l && r <= y) return data[k];  
 int mid = l + r >> 1, res = 0;  
 if (x <= mid) res += vec\_query(ch[k][0], l, mid, x, y);  
 if (y > mid) res += vec\_query(ch[k][1], mid + 1, r, x, y);  
 return res;  
}

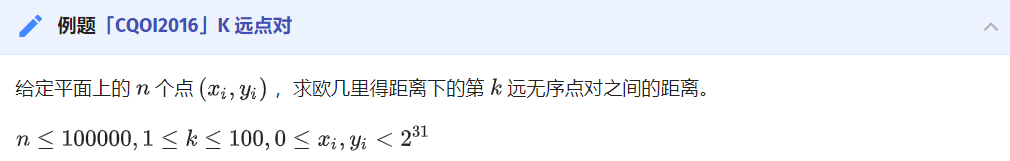
第三维修改

void vec\_insert(int &k, int l, int r, int loc) {  
 if (k == 0) k = ++tot;  
 data[k]++;  
 if (l == r) return;  
 int mid = l + r >> 1;  
 if (loc <= mid) vec\_insert(ch[k][0], l, mid, loc);  
 if (loc > mid) vec\_insert(ch[k][1], mid + 1, r, loc);  
}

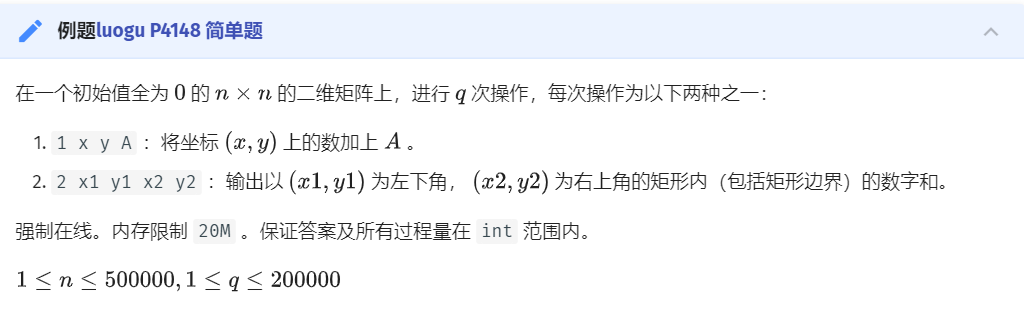
## K-D Tree



#include <algorithm>  
#include <cmath>  
#include <cstdio>  
#include <cstdlib>  
#include <cstring>  
using namespace std;  
const int maxn = 200010;  
int n, d[maxn], lc[maxn], rc[maxn];  
double ans = 2e18;  
struct node {  
 double x, y;  
} s[maxn];  
double L[maxn], R[maxn], D[maxn], U[maxn];  
double dist(int a, int b) {  
 return (s[a].x - s[b].x) \* (s[a].x - s[b].x) +  
 (s[a].y - s[b].y) \* (s[a].y - s[b].y);  
}  
bool cmp1(node a, node b) { return a.x < b.x; }  
bool cmp2(node a, node b) { return a.y < b.y; }  
void maintain(int x) {  
 L[x] = R[x] = s[x].x;  
 D[x] = U[x] = s[x].y;  
 if (lc[x])  
 L[x] = min(L[x], L[lc[x]]), R[x] = max(R[x], R[lc[x]]),  
 D[x] = min(D[x], D[lc[x]]), U[x] = max(U[x], U[lc[x]]);  
 if (rc[x])  
 L[x] = min(L[x], L[rc[x]]), R[x] = max(R[x], R[rc[x]]),  
 D[x] = min(D[x], D[rc[x]]), U[x] = max(U[x], U[rc[x]]);  
}  
int build(int l, int r) {  
 if (l >= r) return 0;  
 int mid = (l + r) >> 1;  
 double avx = 0, avy = 0, vax = 0, vay = 0; // average variance  
 for (int i = l; i <= r; i++) avx += s[i].x, avy += s[i].y;  
 avx /= (double)(r - l + 1);  
 avy /= (double)(r - l + 1);  
 for (int i = l; i <= r; i++)  
 vax += (s[i].x - avx) \* (s[i].x - avx),  
 vay += (s[i].y - avy) \* (s[i].y - avy);  
 if (vax >= vay)  
 d[mid] = 1, nth\_element(s + l, s + mid, s + r + 1, cmp1);  
 else  
 d[mid] = 2, nth\_element(s + l, s + mid, s + r + 1, cmp2);  
 lc[mid] = build(l, mid - 1), rc[mid] = build(mid + 1, r);  
 maintain(mid);  
 return mid;  
}  
double f(int a, int b) {  
 double ret = 0;  
 if (L[b] > s[a].x) ret += (L[b] - s[a].x) \* (L[b] - s[a].x);  
 if (R[b] < s[a].x) ret += (s[a].x - R[b]) \* (s[a].x - R[b]);  
 if (D[b] > s[a].y) ret += (D[b] - s[a].y) \* (D[b] - s[a].y);  
 if (U[b] < s[a].y) ret += (s[a].y - U[b]) \* (s[a].y - U[b]);  
 return ret;  
}  
void query(int l, int r, int x) {  
 if (l > r) return;  
 int mid = (l + r) >> 1;  
 if (mid != x) ans = min(ans, dist(x, mid));  
 if (l == r) return;  
 double distl = f(x, lc[mid]), distr = f(x, rc[mid]);  
 if (distl < ans && distr < ans) {  
 if (distl < distr) {  
 query(l, mid - 1, x);  
 if (distr < ans) query(mid + 1, r, x);  
 } else {  
 query(mid + 1, r, x);  
 if (distl < ans) query(l, mid - 1, x);  
 }  
 } else {  
 if (distl < ans) query(l, mid - 1, x);  
 if (distr < ans) query(mid + 1, r, x);  
 }  
}  
int main() {  
 scanf("%d", &n);  
 for (int i = 1; i <= n; i++) scanf("%lf%lf", &s[i].x, &s[i].y);  
 build(1, n);  
 for (int i = 1; i <= n; i++) query(1, n, i);  
 printf("%.4lf\n", sqrt(ans));  
 return 0;  
}



#include <algorithm>  
#include <cstdio>  
#include <cstring>  
#include <queue>  
using namespace std;  
#define int long long  
const int maxn = 100010;  
int n, k;  
priority\_queue<int, vector<int>, greater<int> > q;  
struct node {  
 int x, y;  
} s[maxn];  
bool cmp1(node a, node b) { return a.x < b.x; }  
bool cmp2(node a, node b) { return a.y < b.y; }  
int lc[maxn], rc[maxn], L[maxn], R[maxn], D[maxn], U[maxn];  
void maintain(int x) {  
 L[x] = R[x] = s[x].x;  
 D[x] = U[x] = s[x].y;  
 if (lc[x])  
 L[x] = min(L[x], L[lc[x]]), R[x] = max(R[x], R[lc[x]]),  
 D[x] = min(D[x], D[lc[x]]), U[x] = max(U[x], U[lc[x]]);  
 if (rc[x])  
 L[x] = min(L[x], L[rc[x]]), R[x] = max(R[x], R[rc[x]]),  
 D[x] = min(D[x], D[rc[x]]), U[x] = max(U[x], U[rc[x]]);  
}  
int build(int l, int r) {  
 if (l > r) return 0;  
 int mid = (l + r) >> 1;  
 double av1 = 0, av2 = 0, va1 = 0, va2 = 0; // average variance  
 for (int i = l; i <= r; i++) av1 += s[i].x, av2 += s[i].y;  
 av1 /= (r - l + 1);  
 av2 /= (r - l + 1);  
 for (int i = l; i <= r; i++)  
 va1 += (av1 - s[i].x) \* (av1 - s[i].x),  
 va2 += (av2 - s[i].y) \* (av2 - s[i].y);  
 if (va1 > va2)  
 nth\_element(s + l, s + mid, s + r + 1, cmp1);  
 else  
 nth\_element(s + l, s + mid, s + r + 1, cmp2);  
 lc[mid] = build(l, mid - 1);  
 rc[mid] = build(mid + 1, r);  
 maintain(mid);  
 return mid;  
}  
int sq(int x) { return x \* x; }  
int dist(int a, int b) {  
 return max(sq(s[a].x - L[b]), sq(s[a].x - R[b])) +  
 max(sq(s[a].y - D[b]), sq(s[a].y - U[b]));  
}  
void query(int l, int r, int x) {  
 if (l > r) return;  
 int mid = (l + r) >> 1, t = sq(s[mid].x - s[x].x) + sq(s[mid].y - s[x].y);  
 if (t > q.top()) q.pop(), q.push(t);  
 int distl = dist(x, lc[mid]), distr = dist(x, rc[mid]);  
 if (distl > q.top() && distr > q.top()) {  
 if (distl > distr) {  
 query(l, mid - 1, x);  
 if (distr > q.top()) query(mid + 1, r, x);  
 } else {  
 query(mid + 1, r, x);  
 if (distl > q.top()) query(l, mid - 1, x);  
 }  
 } else {  
 if (distl > q.top()) query(l, mid - 1, x);  
 if (distr > q.top()) query(mid + 1, r, x);  
 }  
}  
main() {  
 scanf("%lld%lld", &n, &k);  
 k \*= 2;  
 for (int i = 1; i <= k; i++) q.push(0);  
 for (int i = 1; i <= n; i++) scanf("%lld%lld", &s[i].x, &s[i].y);  
 build(1, n);  
 for (int i = 1; i <= n; i++) query(1, n, i);  
 printf("%lld\n", q.top());  
 return 0;  
}



#include <algorithm>  
#include <cstdio>  
#include <cstring>  
using namespace std;  
const int maxn = 200010;  
int n, op, xl, xr, yl, yr, lstans;  
struct node {  
 int x, y, v;  
} s[maxn];  
bool cmp1(int a, int b) { return s[a].x < s[b].x; }  
bool cmp2(int a, int b) { return s[a].y < s[b].y; }  
double a = 0.725;  
int rt, cur, d[maxn], lc[maxn], rc[maxn], L[maxn], R[maxn], D[maxn], U[maxn],  
 siz[maxn], sum[maxn];  
int g[maxn], t;  
void print(int x) {  
 if (!x) return;  
 print(lc[x]);  
 g[++t] = x;  
 print(rc[x]);  
}  
void maintain(int x) {  
 siz[x] = siz[lc[x]] + siz[rc[x]] + 1;  
 sum[x] = sum[lc[x]] + sum[rc[x]] + s[x].v;  
 L[x] = R[x] = s[x].x;  
 D[x] = U[x] = s[x].y;  
 if (lc[x])  
 L[x] = min(L[x], L[lc[x]]), R[x] = max(R[x], R[lc[x]]),  
 D[x] = min(D[x], D[lc[x]]), U[x] = max(U[x], U[lc[x]]);  
 if (rc[x])  
 L[x] = min(L[x], L[rc[x]]), R[x] = max(R[x], R[rc[x]]),  
 D[x] = min(D[x], D[rc[x]]), U[x] = max(U[x], U[rc[x]]);  
}  
int build(int l, int r) {  
 if (l > r) return 0;  
 int mid = (l + r) >> 1;  
 double av1 = 0, av2 = 0, va1 = 0, va2 = 0;  
 for (int i = l; i <= r; i++) av1 += s[g[i]].x, av2 += s[g[i]].y;  
 av1 /= (r - l + 1);  
 av2 /= (r - l + 1);  
 for (int i = l; i <= r; i++)  
 va1 += (av1 - s[g[i]].x) \* (av1 - s[g[i]].x),  
 va2 += (av2 - s[g[i]].y) \* (av2 - s[g[i]].y);  
 if (va1 > va2)  
 nth\_element(g + l, g + mid, g + r + 1, cmp1), d[g[mid]] = 1;  
 else  
 nth\_element(g + l, g + mid, g + r + 1, cmp2), d[g[mid]] = 2;  
 lc[g[mid]] = build(l, mid - 1);  
 rc[g[mid]] = build(mid + 1, r);  
 maintain(g[mid]);  
 return g[mid];  
}  
void rebuild(int& x) {  
 t = 0;  
 print(x);  
 x = build(1, t);  
}  
bool bad(int x) { return a \* siz[x] <= (double)max(siz[lc[x]], siz[rc[x]]); }  
void insert(int& x, int v) {  
 if (!x) {  
 x = v;  
 maintain(x);  
 return;  
 }  
 if (d[x] == 1) {  
 if (s[v].x <= s[x].x)  
 insert(lc[x], v);  
 else  
 insert(rc[x], v);  
 } else {  
 if (s[v].y <= s[x].y)  
 insert(lc[x], v);  
 else  
 insert(rc[x], v);  
 }  
 maintain(x);  
 if (bad(x)) rebuild(x);  
}  
int query(int x) {  
 if (!x || xr < L[x] || xl > R[x] || yr < D[x] || yl > U[x]) return 0;  
 if (xl <= L[x] && R[x] <= xr && yl <= D[x] && U[x] <= yr) return sum[x];  
 int ret = 0;  
 if (xl <= s[x].x && s[x].x <= xr && yl <= s[x].y && s[x].y <= yr)  
 ret += s[x].v;  
 return query(lc[x]) + query(rc[x]) + ret;  
}  
int main() {  
 scanf("%d", &n);  
 while (~scanf("%d", &op)) {  
 if (op == 1) {  
 cur++, scanf("%d%d%d", &s[cur].x, &s[cur].y, &s[cur].v);  
 s[cur].x ^= lstans;  
 s[cur].y ^= lstans;  
 s[cur].v ^= lstans;  
 insert(rt, cur);  
 }  
 if (op == 2) {  
 scanf("%d%d%d%d", &xl, &yl, &xr, &yr);  
 xl ^= lstans;  
 yl ^= lstans;  
 xr ^= lstans;  
 yr ^= lstans;  
 printf("%d\n", lstans = query(rt));  
 }  
 if (op == 3) return 0;  
 }  
}

## 偏序

链接：https://ac.nowcoder.com/acm/contest/8021/F  
来源：牛客网

给定三维空间上n个点，每个点都有xi,yi,zi三个坐标值．找出其中的一对点的距离，使得在这n个点的所有点对中，该距离为所有点对中最小的．

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
const int maxn = 1e5 + 5;  
const double inf = 1e18;  
  
int n, p[maxn];  
  
struct node  
{  
 int x, y, z;  
}a[maxn];  
  
bool cmpx(node a, node b){  
 return a.x < b.x;  
}  
  
bool cmpyz(int num1, int num2){  
 return a[num1].y == a[num2].y ? a[num1].z < a[num2].z : a[num1].y < a[num2].y;  
}  
  
double dis(node a, node b){  
 return (a.x - b.x) \* (a.x - b.x) + (a.y - b.y) \* (a.y - b.y) + (a.z - b.z) \* (a.z - b.z);  
}  
  
double cbq(int l, int r){  
 if(l == r) return inf;  
 if(r - l == 1) return dis(a[l], a[r]);  
 int mid = (l + r) >> 1;  
 double ans = min(cbq(l, mid), cbq(mid + 1, r));  
 int cnt = 0;  
 for (int i = l; i <= r; i++){  
 if(a[i].x > a[mid].x - ans && a[i].x < a[mid].x + ans) p[++cnt] = i;  
 }  
 sort(p + 1, p + 1 + cnt, cmpyz);  
 for (int i = 1; i <= cnt; i++){  
 for (int j = i + 1; j <= cnt; j++){  
 if(a[p[j]].y - a[p[i]].y >= ans) break;  
 ans = min(ans, dis(a[p[i]], a[p[j]]));  
 }  
 }  
 return ans;  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 scanf("%d", &n);  
 for (int i = 1; i <= n; i++) scanf("%d %d %d", &a[i].x, &a[i].y, &a[i].z);  
 sort(a + 1, a + 1 + n, cmpx);  
 double ans = cbq(1, n);  
 ans = sqrt(ans);  
 printf("%.3lf\n", ans);  
  
 return 0;  
}  
/\*\*/

珂朵莉树

/\*  
http://codeforces.com/contest/896/problem/C  
\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
int n, m;  
LL seed, vmax, a[100005];  
  
const long long mod = 1e9 + 7;  
  
struct node  
{  
 int l, r;  
 mutable LL v;  
 node(int l, int r = -1, LL v = 0) : l(l), r(r), v(v) {}  
 bool operator <(const node &rhs)const{  
 return l < rhs.l;  
 }  
};  
  
set<node> s;  
  
LL rnd(){  
 LL ret = seed;  
 seed = (seed \* 7 + 13) % mod;  
 return ret;  
}  
  
LL poww(LL x, LL num, LL mod){  
 LL res = 1;  
 x %= mod;  
 while(num){  
 if(num & 1) res = res \* x % mod;  
 x = x \* x % mod;  
 num >>= 1;  
 }  
 return res;  
}  
  
  
  
void add(int l, int r, LL x = 1){  
 split(l);  
 auto itr = split(r + 1), itl = split(l);  
 for (; itl != itr; itl++) itl->v += x;  
}  
  
void assign(int l, int r, LL x = 0){  
 split(l);  
 auto itr = split(r + 1), itl = split(l);  
 s.erase(itl, itr);  
 s.insert(node(l, r, x));  
}  
  
//求区间第K大值，reversed控制顺序与逆序  
LL k\_min(int l, int r, LL x, int reverse = 0){  
 if(reverse) x = r - l + 2 - x;  
 split(l);  
 auto itr = split(r + 1), itl = split(l);  
 vector<pair<LL, int> > v;  
 for (; itl != itr; itl++) v.push\_back({itl->v, itl->r - itl->l + 1});  
 sort(v.begin(), v.end());  
 for (int i = 0; i < (int)v.size(); i++){  
 x -= v[i].second;  
 if(x <= 0) return v[i].first;  
 }  
 return -1LL;  
}  
  
LL mul(int l, int r, LL x, LL mod){  
 split(l);  
 auto itr = split(r + 1), itl = split(l);  
 LL ans = 0;  
 for (; itl != itr; itl++) ans = (ans + poww(itl->v, x, mod) \* (itl->r - itl->l + 1) % mod) % mod;  
 return ans;  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 scanf("%d %d %lld %lld", &n, &m, &seed, &vmax);  
 for (int i = 1; i <= n; i++){  
 a[i] = rnd() % vmax + 1;  
 s.insert(node(i, i, a[i]));  
 }  
 s.insert(node(n + 1, n + 1, 0));  
 for (int i = 1; i <= m; i++){  
 int op = rnd() % 4 + 1;  
 int l = rnd() % n + 1, r = rnd() % n + 1;  
 if(l > r) swap(l, r);  
 int x, y;  
 if(op == 3) x = rnd() % (r - l + 1) + 1;  
 else x = rnd() % vmax + 1;  
 if(op == 4) y = rnd() % vmax + 1;  
 if(op == 1) add(l, r, x);  
 else if(op == 2) assign(l, r, x);  
 else if(op == 3) printf("%lld\n", k\_min(l, r, x));  
 else printf("%lld\n", mul(l, r, x, y));  
 }  
  
 return 0;  
}  
/\*\*/  
  
/\*  
https://ac.nowcoder.com/acm/contest/3121/L  
\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
int n, q;  
  
struct node  
{  
 int l, r;  
 mutable LL v;  
 node(int l, int r = -1, LL v = 0) : l(l), r(r), v(v) {}  
 bool operator <(const node &rhs)const{  
 return l < rhs.l;  
 }  
};  
  
set<node> s;  
  
auto split(int x){  
 auto it = s.lower\_bound(node(x));  
 if(it != s.end() && it->l == x) return it;  
 --it;  
 // if(it->r < x) return s.end();  
 int l = it->l, r = it->r;  
 LL v = it->v;  
 s.erase(it);  
 s.insert(node(l, x - 1, v));  
 return s.insert(node{x, r, v}).first;  
}  
  
void assign(int l, int r, LL x){  
 split(l);  
 auto itr = split(r + 1), itl = split(l);  
 s.erase(itl, itr);  
 s.insert(node(l, r, x));  
}  
  
LL sum(int l, int r){  
 LL res = 0;  
 auto itr = split(r + 1), itl = split(l);  
 for (; itl != itr; itl++) res += (LL)itl->v \* (itl->r - itl->l + 1);  
 return res;  
}  
  
void change(int l, int r){  
 LL res = sum(l, r);  
 auto itr = split(r + 1), itl = split(l);  
 s.erase(itl, itr);  
 if(l < r) s.insert(node(l, r - 1, 0));  
 s.insert(node(r, r, res));  
}  
  
int solve(int l, int r){  
 split(l);  
 auto itr = split(r + 1), itl = split(l);  
 set<LL> st;  
 for (; itl != itr; itl++) if(itl->v > 0) st.insert(itl->v);  
 return st.size();  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 scanf("%d %d", &n, &q);  
 s.insert(node(1, n, 1));  
 s.insert(node(n + 1, n + 1, 0));  
 for (int i = 1, l, r, op; i <= q; i++){  
 scanf("%d %d %d", &op, &l, &r);  
 if(op == 1) assign(l, r, 1);  
 else if(op == 2){  
 change(l, r);  
 }  
 else printf("%d\n", solve(l, r));  
 }  
  
 return 0;  
}  
/\*\*/

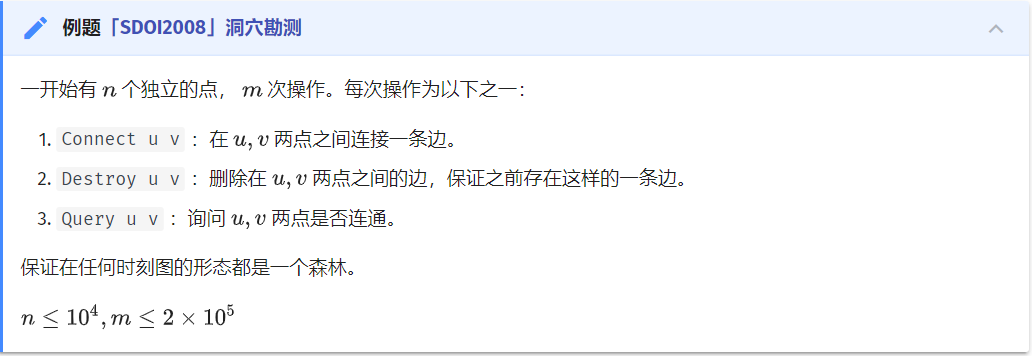
## LCT

### 维护树链信息



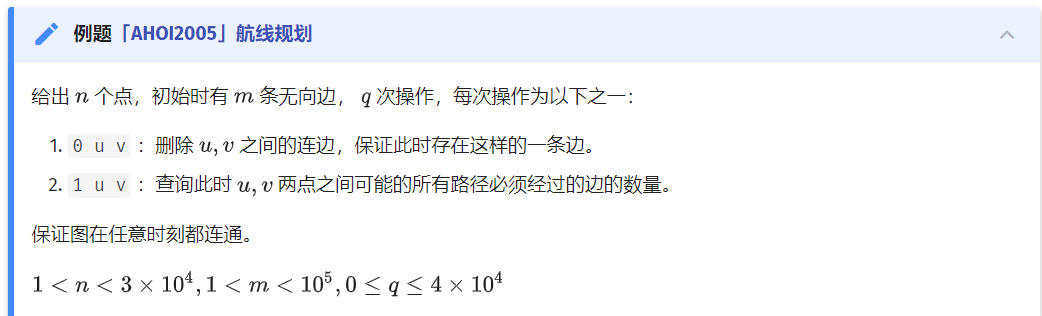
#include <algorithm>  
#include <cstdio>  
#include <cstring>  
using namespace std;  
#define int long long  
const int maxn = 100010;  
const int mod = 51061;  
int n, q, u, v, c;  
char op;  
struct Splay {  
 int ch[maxn][2], fa[maxn], siz[maxn], val[maxn], sum[maxn], rev[maxn],  
 add[maxn], mul[maxn];  
 void clear(int x) {  
 ch[x][0] = ch[x][1] = fa[x] = siz[x] = val[x] = sum[x] = rev[x] = add[x] =  
 0;  
 mul[x] = 1;  
 }  
 int getch(int x) { return (ch[fa[x]][1] == x); }  
 int isroot(int x) {  
 clear(0);  
 return ch[fa[x]][0] != x && ch[fa[x]][1] != x;  
 }  
 void maintain(int x) {  
 clear(0);  
 siz[x] = (siz[ch[x][0]] + 1 + siz[ch[x][1]]) % mod;  
 sum[x] = (sum[ch[x][0]] + val[x] + sum[ch[x][1]]) % mod;  
 }  
 void pushdown(int x) {  
 clear(0);  
 if (mul[x] != 1) {  
 if (ch[x][0])  
 mul[ch[x][0]] = (mul[x] \* mul[ch[x][0]]) % mod,  
 val[ch[x][0]] = (val[ch[x][0]] \* mul[x]) % mod,  
 sum[ch[x][0]] = (sum[ch[x][0]] \* mul[x]) % mod,  
 add[ch[x][0]] = (add[ch[x][0]] \* mul[x]) % mod;  
 if (ch[x][1])  
 mul[ch[x][1]] = (mul[x] \* mul[ch[x][1]]) % mod,  
 val[ch[x][1]] = (val[ch[x][1]] \* mul[x]) % mod,  
 sum[ch[x][1]] = (sum[ch[x][1]] \* mul[x]) % mod,  
 add[ch[x][1]] = (add[ch[x][1]] \* mul[x]) % mod;  
 mul[x] = 1;  
 }  
 if (add[x]) {  
 if (ch[x][0])  
 add[ch[x][0]] = (add[ch[x][0]] + add[x]) % mod,  
 val[ch[x][0]] = (val[ch[x][0]] + add[x]) % mod,  
 sum[ch[x][0]] = (sum[ch[x][0]] + add[x] \* siz[ch[x][0]] % mod) % mod;  
 if (ch[x][1])  
 add[ch[x][1]] = (add[ch[x][1]] + add[x]) % mod,  
 val[ch[x][1]] = (val[ch[x][1]] + add[x]) % mod,  
 sum[ch[x][1]] = (sum[ch[x][1]] + add[x] \* siz[ch[x][1]] % mod) % mod;  
 add[x] = 0;  
 }  
 if (rev[x]) {  
 if (ch[x][0]) rev[ch[x][0]] ^= 1, swap(ch[ch[x][0]][0], ch[ch[x][0]][1]);  
 if (ch[x][1]) rev[ch[x][1]] ^= 1, swap(ch[ch[x][1]][0], ch[ch[x][1]][1]);  
 rev[x] = 0;  
 }  
 }  
 void update(int x) {  
 if (!isroot(x)) update(fa[x]);  
 pushdown(x);  
 }  
 void print(int x) {  
 if (!x) return;  
 pushdown(x);  
 print(ch[x][0]);  
 printf("%lld ", x);  
 print(ch[x][1]);  
 }  
 void rotate(int x) {  
 int y = fa[x], z = fa[y], chx = getch(x), chy = getch(y);  
 fa[x] = z;  
 if (!isroot(y)) ch[z][chy] = x;  
 ch[y][chx] = ch[x][chx ^ 1];  
 fa[ch[x][chx ^ 1]] = y;  
 ch[x][chx ^ 1] = y;  
 fa[y] = x;  
 maintain(y);  
 maintain(x);  
 maintain(z);  
 }  
 void splay(int x) {  
 update(x);  
 for (int f = fa[x]; f = fa[x], !isroot(x); rotate(x))  
 if (!isroot(f)) rotate(getch(x) == getch(f) ? f : x);  
 }  
 void access(int x) {  
 for (int f = 0; x; f = x, x = fa[x]) splay(x), ch[x][1] = f, maintain(x);  
 }  
 void makeroot(int x) {  
 access(x);  
 splay(x);  
 swap(ch[x][0], ch[x][1]);  
 rev[x] ^= 1;  
 }  
 int find(int x) {  
 access(x);  
 splay(x);  
 while (ch[x][0]) x = ch[x][0];  
 splay(x);  
 return x;  
 }  
} st;  
main() {  
 scanf("%lld%lld", &n, &q);  
 for (int i = 1; i <= n; i++) st.val[i] = 1, st.maintain(i);  
 for (int i = 1; i < n; i++) {  
 scanf("%lld%lld", &u, &v);  
 if (st.find(u) != st.find(v)) st.makeroot(u), st.fa[u] = v;  
 }  
 while (q--) {  
 scanf(" %c%lld%lld", &op, &u, &v);  
 if (op == '+') {  
 scanf("%lld", &c);  
 st.makeroot(u), st.access(v), st.splay(v);  
 st.val[v] = (st.val[v] + c) % mod;  
 st.sum[v] = (st.sum[v] + st.siz[v] \* c % mod) % mod;  
 st.add[v] = (st.add[v] + c) % mod;  
 }  
 if (op == '-') {  
 st.makeroot(u);  
 st.access(v);  
 st.splay(v);  
 if (st.ch[v][0] == u && !st.ch[u][1]) st.ch[v][0] = st.fa[u] = 0;  
 scanf("%lld%lld", &u, &v);  
 if (st.find(u) != st.find(v)) st.makeroot(u), st.fa[u] = v;  
 }  
 if (op == '\*') {  
 scanf("%lld", &c);  
 st.makeroot(u), st.access(v), st.splay(v);  
 st.val[v] = st.val[v] \* c % mod;  
 st.sum[v] = st.sum[v] \* c % mod;  
 st.mul[v] = st.mul[v] \* c % mod;  
 }  
 if (op == '/')  
 st.makeroot(u), st.access(v), st.splay(v), printf("%lld\n", st.sum[v]);  
 }  
 return 0;  
}

### 维护连通性质



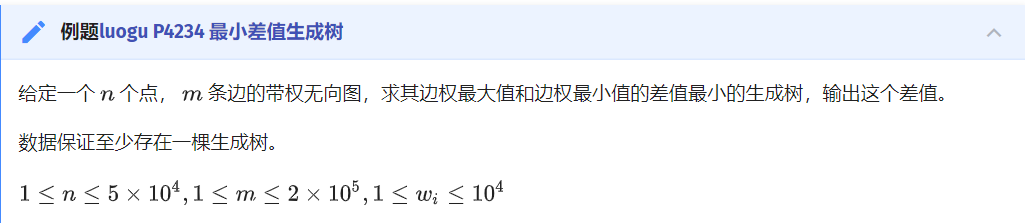
#include <algorithm>  
#include <cstdio>  
#include <cstring>  
using namespace std;  
const int maxn = 10010;  
struct Splay {  
 int ch[maxn][2], fa[maxn], tag[maxn];  
 void clear(int x) { ch[x][0] = ch[x][1] = fa[x] = tag[x] = 0; }  
 int getch(int x) { return ch[fa[x]][1] == x; }  
 int isroot(int x) { return ch[fa[x]][0] != x && ch[fa[x]][1] != x; }  
 void pushdown(int x) {  
 if (tag[x]) {  
 if (ch[x][0]) swap(ch[ch[x][0]][0], ch[ch[x][0]][1]), tag[ch[x][0]] ^= 1;  
 if (ch[x][1]) swap(ch[ch[x][1]][0], ch[ch[x][1]][1]), tag[ch[x][1]] ^= 1;  
 tag[x] = 0;  
 }  
 }  
 void update(int x) {  
 if (!isroot(x)) update(fa[x]);  
 pushdown(x);  
 }  
 void rotate(int x) {  
 int y = fa[x], z = fa[y], chx = getch(x), chy = getch(y);  
 fa[x] = z;  
 if (!isroot(y)) ch[z][chy] = x;  
 ch[y][chx] = ch[x][chx ^ 1];  
 fa[ch[x][chx ^ 1]] = y;  
 ch[x][chx ^ 1] = y;  
 fa[y] = x;  
 }  
 void splay(int x) {  
 update(x);  
 for (int f = fa[x]; f = fa[x], !isroot(x); rotate(x))  
 if (!isroot(f)) rotate(getch(x) == getch(f) ? f : x);  
 }  
 void access(int x) {  
 for (int f = 0; x; f = x, x = fa[x]) splay(x), ch[x][1] = f;  
 }  
 void makeroot(int x) {  
 access(x);  
 splay(x);  
 swap(ch[x][0], ch[x][1]);  
 tag[x] ^= 1;  
 }  
 int find(int x) {  
 access(x);  
 splay(x);  
 while (ch[x][0]) x = ch[x][0];  
 splay(x);  
 return x;  
 }  
} st;  
int n, q, x, y;  
char op[maxn];  
int main() {  
 scanf("%d%d", &n, &q);  
 while (q--) {  
 scanf("%s%d%d", op, &x, &y);  
 if (op[0] == 'Q') {  
 if (st.find(x) == st.find(y))  
 printf("Yes\n");  
 else  
 printf("No\n");  
 }  
 if (op[0] == 'C')  
 if (st.find(x) != st.find(y)) st.makeroot(x), st.fa[x] = y;  
 if (op[0] == 'D') {  
 st.makeroot(x);  
 st.access(y);  
 st.splay(y);  
 if (st.ch[y][0] == x && !st.ch[x][1]) st.ch[y][0] = st.fa[x] = 0;  
 }  
 }  
 return 0;  
}

### 维护边双连通分量



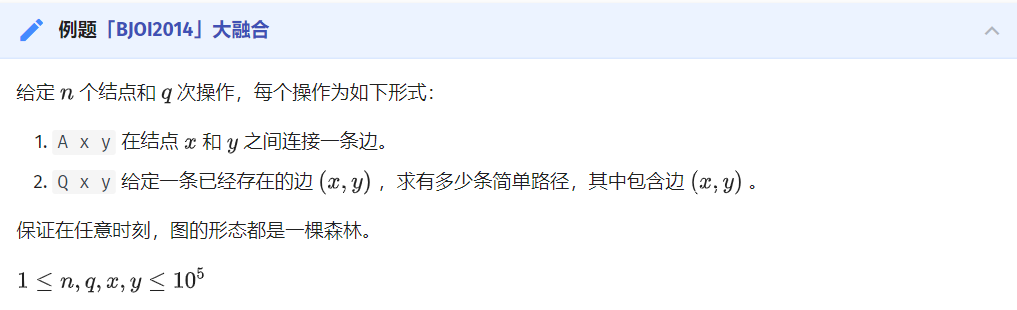
#include <algorithm>  
#include <cstdio>  
#include <cstring>  
#include <map>  
using namespace std;  
const int maxn = 200010;  
int f[maxn];  
int findp(int x) { return f[x] ? f[x] = findp(f[x]) : x; }  
void merge(int x, int y) {  
 x = findp(x);  
 y = findp(y);  
 if (x != y) f[x] = y;  
}  
struct Splay {  
 int ch[maxn][2], fa[maxn], tag[maxn], siz[maxn];  
 void clear(int x) { ch[x][0] = ch[x][1] = fa[x] = tag[x] = siz[x] = 0; }  
 int getch(int x) { return ch[findp(fa[x])][1] == x; }  
 int isroot(int x) {  
 return ch[findp(fa[x])][0] != x && ch[findp(fa[x])][1] != x;  
 }  
 void maintain(int x) {  
 clear(0);  
 if (x) siz[x] = siz[ch[x][0]] + 1 + siz[ch[x][1]];  
 }  
 void pushdown(int x) {  
 if (tag[x]) {  
 if (ch[x][0]) tag[ch[x][0]] ^= 1, swap(ch[ch[x][0]][0], ch[ch[x][0]][1]);  
 if (ch[x][1]) tag[ch[x][1]] ^= 1, swap(ch[ch[x][1]][0], ch[ch[x][1]][1]);  
 tag[x] = 0;  
 }  
 }  
 void print(int x) {  
 if (!x) return;  
 pushdown(x);  
 print(ch[x][0]);  
 printf("%d ", x);  
 print(ch[x][1]);  
 }  
 void update(int x) {  
 if (!isroot(x)) update(findp(fa[x]));  
 pushdown(x);  
 }  
 void rotate(int x) {  
 x = findp(x);  
 int y = findp(fa[x]), z = findp(fa[y]), chx = getch(x), chy = getch(y);  
 fa[x] = z;  
 if (!isroot(y)) ch[z][chy] = x;  
 ch[y][chx] = ch[x][chx ^ 1];  
 fa[ch[x][chx ^ 1]] = y;  
 ch[x][chx ^ 1] = y;  
 fa[y] = x;  
 maintain(y);  
 maintain(x);  
 if (z) maintain(z);  
 }  
 void splay(int x) {  
 x = findp(x);  
 update(x);  
 for (int f = findp(fa[x]); f = findp(fa[x]), !isroot(x); rotate(x))  
 if (!isroot(f)) rotate(getch(x) == getch(f) ? f : x);  
 }  
 void access(int x) {  
 for (int f = 0; x; f = x, x = findp(fa[x]))  
 splay(x), ch[x][1] = f, maintain(x);  
 }  
 void makeroot(int x) {  
 x = findp(x);  
 access(x);  
 splay(x);  
 tag[x] ^= 1;  
 swap(ch[x][0], ch[x][1]);  
 }  
 int find(int x) {  
 x = findp(x);  
 access(x);  
 splay(x);  
 while (ch[x][0]) x = ch[x][0];  
 splay(x);  
 return x;  
 }  
 void dfs(int x) {  
 pushdown(x);  
 if (ch[x][0]) dfs(ch[x][0]), merge(ch[x][0], x);  
 if (ch[x][1]) dfs(ch[x][1]), merge(ch[x][1], x);  
 }  
} st;  
int n, m, q, x, y, cur, ans[maxn];  
struct oper {  
 int op, a, b;  
} s[maxn];  
map<pair<int, int>, int> mp;  
int main() {  
 scanf("%d%d", &n, &m);  
 for (int i = 1; i <= n; i++) st.maintain(i);  
 for (int i = 1; i <= m; i++)  
 scanf("%d%d", &x, &y), mp[{x, y}] = mp[{y, x}] = 1;  
 while (scanf("%d", &s[++q].op)) {  
 if (s[q].op == -1) {  
 q--;  
 break;  
 }  
 scanf("%d%d", &s[q].a, &s[q].b);  
 if (!s[q].op) mp[{s[q].a, s[q].b}] = mp[{s[q].b, s[q].a}] = 0;  
 }  
 reverse(s + 1, s + q + 1);  
 for (map<pair<int, int>, int>::iterator it = mp.begin(); it != mp.end(); it++)  
 if (it->second) {  
 mp[{it->first.second, it->first.first}] = 0;  
 x = findp(it->first.first);  
 y = findp(it->first.second);  
 if (st.find(x) != st.find(y))  
 st.makeroot(x), st.fa[x] = y;  
 else {  
 if (x == y) continue;  
 st.makeroot(x);  
 st.access(y);  
 st.splay(y);  
 st.dfs(y);  
 int t = findp(y);  
 st.fa[t] = findp(st.fa[y]);  
 st.ch[t][0] = st.ch[t][1] = 0;  
 st.maintain(t);  
 }  
 }  
 for (int i = 1; i <= q; i++) {  
 if (s[i].op == 0) {  
 x = findp(s[i].a);  
 y = findp(s[i].b);  
 st.makeroot(x);  
 st.access(y);  
 st.splay(y);  
 st.dfs(y);  
 int t = findp(y);  
 st.fa[t] = st.fa[y];  
 st.ch[t][0] = st.ch[t][1] = 0;  
 st.maintain(t);  
 }  
 if (s[i].op == 1) {  
 x = findp(s[i].a);  
 y = findp(s[i].b);  
 st.makeroot(x);  
 st.access(y);  
 st.splay(y);  
 ans[++cur] = st.siz[y] - 1;  
 }  
 }  
 for (int i = cur; i >= 1; i--) printf("%d\n", ans[i]);  
 return 0;  
}

### 维护边权



#include <algorithm>  
#include <cstdio>  
#include <cstring>  
#include <set>  
using namespace std;  
const int maxn = 5000010;  
struct Splay {  
 int ch[maxn][2], fa[maxn], tag[maxn], val[maxn], minn[maxn];  
 void clear(int x) {  
 ch[x][0] = ch[x][1] = fa[x] = tag[x] = val[x] = minn[x] = 0;  
 }  
 int getch(int x) { return ch[fa[x]][1] == x; }  
 int isroot(int x) { return ch[fa[x]][0] != x && ch[fa[x]][1] != x; }  
 void maintain(int x) {  
 if (!x) return;  
 minn[x] = x;  
 if (ch[x][0]) {  
 if (val[minn[ch[x][0]]] < val[minn[x]]) minn[x] = minn[ch[x][0]];  
 }  
 if (ch[x][1]) {  
 if (val[minn[ch[x][1]]] < val[minn[x]]) minn[x] = minn[ch[x][1]];  
 }  
 }  
 void pushdown(int x) {  
 if (tag[x]) {  
 if (ch[x][0]) tag[ch[x][0]] ^= 1, swap(ch[ch[x][0]][0], ch[ch[x][0]][1]);  
 if (ch[x][1]) tag[ch[x][1]] ^= 1, swap(ch[ch[x][1]][0], ch[ch[x][1]][1]);  
 tag[x] = 0;  
 }  
 }  
 void update(int x) {  
 if (!isroot(x)) update(fa[x]);  
 pushdown(x);  
 }  
 void print(int x) {  
 if (!x) return;  
 pushdown(x);  
 print(ch[x][0]);  
 printf("%d ", x);  
 print(ch[x][1]);  
 }  
 void rotate(int x) {  
 int y = fa[x], z = fa[y], chx = getch(x), chy = getch(y);  
 fa[x] = z;  
 if (!isroot(y)) ch[z][chy] = x;  
 ch[y][chx] = ch[x][chx ^ 1];  
 fa[ch[x][chx ^ 1]] = y;  
 ch[x][chx ^ 1] = y;  
 fa[y] = x;  
 maintain(y);  
 maintain(x);  
 if (z) maintain(z);  
 }  
 void splay(int x) {  
 update(x);  
 for (int f = fa[x]; f = fa[x], !isroot(x); rotate(x))  
 if (!isroot(f)) rotate(getch(x) == getch(f) ? f : x);  
 }  
 void access(int x) {  
 for (int f = 0; x; f = x, x = fa[x]) splay(x), ch[x][1] = f, maintain(x);  
 }  
 void makeroot(int x) {  
 access(x);  
 splay(x);  
 tag[x] ^= 1;  
 swap(ch[x][0], ch[x][1]);  
 }  
 int find(int x) {  
 access(x);  
 splay(x);  
 while (ch[x][0]) x = ch[x][0];  
 splay(x);  
 return x;  
 }  
 void link(int x, int y) {  
 makeroot(x);  
 fa[x] = y;  
 }  
 void cut(int x, int y) {  
 makeroot(x);  
 access(y);  
 splay(y);  
 ch[y][0] = fa[x] = 0;  
 maintain(y);  
 }  
} st;  
const int inf = 2e9 + 1;  
int n, m, ans, nww, x, y;  
struct Edge {  
 int u, v, w;  
 bool operator<(Edge x) const { return w < x.w; };  
} s[maxn];  
multiset<int> mp;  
int main() {  
 scanf("%d%d", &n, &m);  
 for (int i = 1; i <= n; i++) st.val[i] = inf, st.maintain(i);  
 for (int i = 1; i <= m; i++) scanf("%d%d%d", &s[i].u, &s[i].v, &s[i].w);  
 sort(s + 1, s + m + 1);  
 for (int i = 1; i <= m; i++) st.val[n + i] = s[i].w, st.maintain(n + i);  
 for (int i = 1; i <= m; i++) {  
 x = s[i].u;  
 y = s[i].v;  
 if (x == y) continue;  
 if (st.find(x) != st.find(y)) {  
 nww++;  
 st.link(x, n + i);  
 st.link(n + i, y);  
 mp.insert(s[i].w);  
 if (nww == n - 1) ans = s[i].w - (\*(mp.begin()++));  
 } else {  
 st.makeroot(x);  
 st.access(y);  
 st.splay(y);  
 int t = st.minn[y] - n;  
 st.cut(s[t].u, t + n);  
 st.cut(t + n, s[t].v);  
 mp.erase(mp.find(s[t].w));  
 st.link(x, n + i);  
 st.link(n + i, y);  
 mp.insert(s[i].w);  
 if (nww == n - 1) ans = min(ans, s[i].w - (\*(mp.begin()++)));  
 }  
 }  
 printf("%d\n", ans);  
 return 0;  
}

### 维护子树信息



#include <algorithm>  
#include <cstdio>  
#include <cstring>  
using namespace std;  
const int maxn = 100010;  
typedef long long ll;  
struct Splay {  
 int ch[maxn][2], fa[maxn], siz[maxn], siz2[maxn], tag[maxn];  
 void clear(int x) {  
 ch[x][0] = ch[x][1] = fa[x] = siz[x] = siz2[x] = tag[x] = 0;  
 }  
 int getch(int x) { return ch[fa[x]][1] == x; }  
 int isroot(int x) { return ch[fa[x]][0] != x && ch[fa[x]][1] != x; }  
 void maintain(int x) {  
 clear(0);  
 if (x) siz[x] = siz[ch[x][0]] + 1 + siz[ch[x][1]] + siz2[x];  
 }  
 void pushdown(int x) {  
 if (tag[x]) {  
 if (ch[x][0]) swap(ch[ch[x][0]][0], ch[ch[x][0]][1]), tag[ch[x][0]] ^= 1;  
 if (ch[x][1]) swap(ch[ch[x][1]][0], ch[ch[x][1]][1]), tag[ch[x][1]] ^= 1;  
 tag[x] = 0;  
 }  
 }  
 void update(int x) {  
 if (!isroot(x)) update(fa[x]);  
 pushdown(x);  
 }  
 void rotate(int x) {  
 int y = fa[x], z = fa[y], chx = getch(x), chy = getch(y);  
 fa[x] = z;  
 if (!isroot(y)) ch[z][chy] = x;  
 ch[y][chx] = ch[x][chx ^ 1];  
 fa[ch[x][chx ^ 1]] = y;  
 ch[x][chx ^ 1] = y;  
 fa[y] = x;  
 maintain(y);  
 maintain(x);  
 maintain(z);  
 }  
 void splay(int x) {  
 update(x);  
 for (int f = fa[x]; f = fa[x], !isroot(x); rotate(x))  
 if (!isroot(f)) rotate(getch(x) == getch(f) ? f : x);  
 }  
 void access(int x) {  
 for (int f = 0; x; f = x, x = fa[x])  
 splay(x), siz2[x] += siz[ch[x][1]] - siz[f], ch[x][1] = f, maintain(x);  
 }  
 void makeroot(int x) {  
 access(x);  
 splay(x);  
 swap(ch[x][0], ch[x][1]);  
 tag[x] ^= 1;  
 }  
 int find(int x) {  
 access(x);  
 splay(x);  
 while (ch[x][0]) x = ch[x][0];  
 splay(x);  
 return x;  
 }  
} st;  
int n, q, x, y;  
char op;  
int main() {  
 scanf("%d%d", &n, &q);  
 while (q--) {  
 scanf(" %c%d%d", &op, &x, &y);  
 if (op == 'A') {  
 st.makeroot(x);  
 st.makeroot(y);  
 st.fa[x] = y;  
 st.siz2[y] += st.siz[x];  
 }  
 if (op == 'Q') {  
 st.makeroot(x);  
 st.access(y);  
 st.splay(y);  
 st.ch[y][0] = st.fa[x] = 0;  
 st.maintain(x);  
 st.makeroot(x);  
 st.makeroot(y);  
 printf("%lld\n", (ll)(st.siz[x] \* st.siz[y]));  
 st.makeroot(x);  
 st.makeroot(y);  
 st.fa[x] = y;  
 st.siz2[y] += st.siz[x];  
 }  
 }  
 return 0;  
}

# 图论

## 树的直径

### dfs

const int N = 10009;  
VI adj[N];  
int d[N], c;  
int n;  
#define v (\*it)  
void dfs(int u) {  
 ECH(it, adj[u]) if (!d[v]) {  
 d[v] = d[u] + 1;  
 if (d[v] > d[c]) c = v;  
 dfs(v);  
 }  
}  
#undef v  
int main() {  
 REP\_C(i, RD(n) - 1) {  
 int a, b;  
 RD(a, b);  
 --a, --b;  
 adj[a].PB(b), adj[b].PB(a);  
 }  
  
 d[0] = 1, dfs(0);  
 RST(d), d[c] = 1, dfs(c), OT(d[c] - 1);  
}

### 树形 DP

#include <iostream>  
#include <vector>  
using namespace std;  
  
const int N = int(1e4) + 9;  
vector<int> adj[N];  
int n, d;  
int dfs(int u = 1, int p = -1) {  
 int d1 = 0, d2 = 0;  
 for (auto v : adj[u]) {  
 if (v == p) continue;  
 int d = dfs(v, u) + 1;  
 if (d > d1)  
 d2 = d1, d1 = d;  
 else if (d > d2)  
 d2 = d;  
 }  
 d = max(d, d1 + d2);  
 return d1;  
}  
int main() {  
 cin >> n;  
 for (int i = 0; i < n - 1; ++i) {  
 int a, b;  
 cin >> a >> b;  
 adj[a].push\_back(b);  
 adj[b].push\_back(a);  
 }  
 dfs();  
 cout << d << endl;  
}

## LCA

### 倍增

#include <cstdio>  
#include <cstring>  
#include <iostream>  
#include <vector>  
#define MXN 50007  
using namespace std;  
std::vector<int> v[MXN];  
std::vector<int> w[MXN];  
  
int fa[MXN][31], cost[MXN][31], dep[MXN];  
int n, m;  
int a, b, c;  
void dfs(int root, int fno) {  
 fa[root][0] = fno;  
 dep[root] = dep[fa[root][0]] + 1;  
 for (int i = 1; i < 31; ++i) {  
 fa[root][i] = fa[fa[root][i - 1]][i - 1];  
 cost[root][i] = cost[fa[root][i - 1]][i - 1] + cost[root][i - 1];  
 }  
 int sz = v[root].size();  
 for (int i = 0; i < sz; ++i) {  
 if (v[root][i] == fno) continue;  
 cost[v[root][i]][0] = w[root][i];  
 dfs(v[root][i], root);  
 }  
}  
int lca(int x, int y) {  
 if (dep[x] > dep[y]) swap(x, y);  
 int tmp = dep[y] - dep[x], ans = 0;  
 for (int j = 0; tmp; ++j, tmp >>= 1)  
 if (tmp & 1) ans += cost[y][j], y = fa[y][j];  
 if (y == x) return ans;  
 for (int j = 30; j >= 0 && y != x; --j) {  
 if (fa[x][j] != fa[y][j]) {  
 ans += cost[x][j] + cost[y][j];  
 x = fa[x][j];  
 y = fa[y][j];  
 }  
 }  
 ans += cost[x][0] + cost[y][0];  
 return ans;  
}  
int main() {  
 memset(fa, 0, sizeof(fa));  
 memset(cost, 0, sizeof(cost));  
 memset(dep, 0, sizeof(dep));  
 scanf("%d", &n);  
 for (int i = 1; i < n; ++i) {  
 scanf("%d %d %d", &a, &b, &c);  
 ++a, ++b;  
 v[a].push\_back(b);  
 v[b].push\_back(a);  
 w[a].push\_back(c);  
 w[b].push\_back(c);  
 }  
 dfs(1, 0);  
 scanf("%d", &m);  
 for (int i = 0; i < m; ++i) {  
 scanf("%d %d", &a, &b);  
 ++a, ++b;  
 printf("%d\n", lca(a, b));  
 }  
 return 0;  
}

### Tarjan 算法

#include <algorithm>  
#include <iostream>  
using namespace std;  
  
class Edge {  
 public:  
 int toVertex, fromVertex;  
 int next;  
 int LCA;  
 Edge() : toVertex(-1), fromVertex(-1), next(-1), LCA(-1){};  
 Edge(int u, int v, int n) : fromVertex(u), toVertex(v), next(n), LCA(-1){};  
};  
  
const int MAX = 100;  
int head[MAX], queryHead[MAX];  
Edge edge[MAX], queryEdge[MAX];  
int parent[MAX], visited[MAX];  
int vertexCount, edgeCount, queryCount;  
  
void init() {  
 for (int i = 0; i <= vertexCount; i++) {  
 parent[i] = i;  
 }  
}  
  
int find(int x) {  
 if (parent[x] == x) {  
 return x;  
 } else {  
 return find(parent[x]);  
 }  
}  
  
void tarjan(int u) {  
 parent[u] = u;  
 visited[u] = 1;  
  
 for (int i = head[u]; i != -1; i = edge[i].next) {  
 Edge& e = edge[i];  
 if (!visited[e.toVertex]) {  
 tarjan(e.toVertex);  
 parent[e.toVertex] = u;  
 }  
 }  
  
 for (int i = queryHead[u]; i != -1; i = queryEdge[i].next) {  
 Edge& e = queryEdge[i];  
 if (visited[e.toVertex]) {  
 queryEdge[i ^ 1].LCA = e.LCA = find(e.toVertex);  
 }  
 }  
}  
  
int main() {  
 memset(head, 0xff, sizeof(head));  
 memset(queryHead, 0xff, sizeof(queryHead));  
  
 cin >> vertexCount >> edgeCount >> queryCount;  
 int count = 0;  
 for (int i = 0; i < edgeCount; i++) {  
 int start = 0, end = 0;  
 cin >> start >> end;  
  
 edge[count] = Edge(start, end, head[start]);  
 head[start] = count;  
 count++;  
  
 edge[count] = Edge(end, start, head[end]);  
 head[end] = count;  
 count++;  
 }  
  
 count = 0;  
 for (int i = 0; i < queryCount; i++) {  
 int start = 0, end = 0;  
 cin >> start >> end;  
  
 queryEdge[count] = Edge(start, end, queryHead[start]);  
 queryHead[start] = count;  
 count++;  
  
 queryEdge[count] = Edge(end, start, queryHead[end]);  
 queryHead[end] = count;  
 count++;  
 }  
  
 init();  
 tarjan(1);  
  
 for (int i = 0; i < queryCount; i++) {  
 Edge& e = queryEdge[i \* 2];  
 cout << "(" << e.fromVertex << "," << e.toVertex << ") " << e.LCA << endl;  
 }  
  
 return 0;  
}

### 用欧拉序列转化为 RMQ 问题

int dfn[N << 1], dep[N << 1], dfntot = 0;  
void dfs(int t, int depth) {  
 dfn[++dfntot] = t;  
 pos[t] = dfntot;  
 dep[dfntot] = depth;  
 for (int i = head[t]; i; i = side[i].next) {  
 dfs(side[i].to, t, depth + 1);  
 dfn[++dfntot] = t;  
 dep[dfntot] = depth;  
 }  
}  
void st\_preprocess() {  
 lg[0] = -1; // 预处理 lg 代替库函数 log2 来优化常数  
 for (int i = 1; i <= (N << 1); ++i) lg[i] = lg[i >> 1] + 1;  
 for (int i = 1; i <= (N << 1) - 1; ++i) st[0][i] = dfn[i];  
 for (int i = 1; i <= lg[(N << 1) - 1]; ++i)  
 for (int j = 1; j + (1 << i) - 1 <= ((N << 1) - 1); ++j)  
 st[i][j] = dep[st[i - 1][j]] < dep[st[i - 1][j + (1 << i - 1)]  
 ? st[i - 1][j]  
 : st[i - 1][j + (1 << i - 1)];  
}

## 树的重心

### 定义

对于树上的每一个点，计算其所有子树中最大的子树节点数，这个值最小的点就是这棵树的重心。

（这里以及下文中的“子树”都是指无根树的子树，即包括“向上”的那棵子树，并且不包括整棵树自身。）

### 性质

以树的重心为根时，所有子树的大小都不超过整棵树大小的一半。

树中所有点到某个点的距离和中，到重心的距离和是最小的；如果有两个重心，那么到它们的距离和一样。

把两棵树通过一条边相连得到一棵新的树，那么新的树的重心在连接原来两棵树的重心的路径上。

在一棵树上添加或删除一个叶子，那么它的重心最多只移动一条边的距离。

### 求法

在 DFS 中计算每个子树的大小，记录“向下”的子树的最大大小，利用总点数 - 当前子树（这里的子树指有根树的子树）的大小得到“向上”的子树的大小，然后就可以依据定义找到重心了。

// 这份代码默认节点编号从 1 开始，即 i ∈ [1,n]  
int size[MAXN], // 这个节点的“大小”（所有子树上节点数 + 该节点）  
 weight[MAXN], // 这个节点的“重量”  
 centroid[2]; // 用于记录树的重心（存的是节点编号）  
void GetCentroid(int cur, int fa) { // cur 表示当前节点 (current)  
 size[cur] = 1;  
 weight[cur] = 0;  
 for (int i = head[cur]; i != -1; i = e[i].nxt) {  
 if (e[i].to != fa) { // e[i].to 表示这条有向边所通向的节点。  
 GetCentroid(e[i].to, cur);  
 size[cur] += size[e[i].to];  
 weight[cur] = max(weight[cur], size[e[i].to]);  
 }  
 }  
 weight[cur] = max(weight[cur], n - size[cur]);  
 if (weight[cur] <= n / 2) { // 依照树的重心的定义统计  
 centroid[centroid[0] != 0] = cur;  
 }  
}

## 树链剖分

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
LL n, m, r, p, v[100005];  
int tot, head[100005], size[100005], d[100005], son[100005], top[100005], rk[100005], id[100005], cnt, f[100005];  
  
struct Node  
{  
 int v, next;  
}a[100005 << 1];  
  
struct node  
{  
 int l, r, ls, rs, lazy;  
 LL sum;  
}tree[100005 << 2];  
  
  
void dfs1(int x){  
 d[x] = d[f[x]] + 1, size[x] = 1;  
 for (int v, i = head[x]; i != -1; i = a[i].next){  
 v = a[i].v;  
 if(v == f[x]) continue;  
 f[v] = x;  
 dfs1(v);  
 size[x] += size[v];  
 if(size[v] > size[son[x]]) son[x] = v;  
 }  
}  
  
void dfs2(int x, int tp){  
 top[x] = tp, id[x] = ++cnt, rk[cnt] = x;  
 if(son[x]) dfs2(son[x], tp);  
 for (int v, i = head[x]; i != -1; i = a[i].next){  
 v = a[i].v;  
 if(v == f[x] || v == son[x]) continue;  
 dfs2(v, v);  
 }  
}  
  
void up(int rt){  
 tree[rt].sum = (tree[tree[rt].ls].sum + tree[tree[rt].rs].sum) % p;  
}  
  
int len(int rt){  
 return tree[rt].r - tree[rt].l + 1;  
}  
  
void down(int rt){  
 if(tree[rt].lazy){  
 int ls = tree[rt].ls, rs = tree[rt].rs, lz = tree[rt].lazy;  
 (tree[ls].lazy += lz) %= p, (tree[rs].lazy += lz) %= p;  
 (tree[ls].sum += lz \* len(ls)) %= p, (tree[rs].sum += lz \* len(rs)) %= p;  
 tree[rt].lazy = 0;  
 }  
}  
  
void build(int l, int r, int rt){  
 if(l == r){  
 tree[rt].sum = v[rk[l]], tree[rt].l = tree[rt].r = l;  
 return ;  
 }  
 int mid = (l + r) >> 1;  
 tree[rt].ls = cnt++, tree[rt].rs = cnt++;  
 build(l, mid, tree[rt].ls), build(mid + 1, r, tree[rt].rs);  
 tree[rt].l = tree[tree[rt].ls].l, tree[rt].r = tree[tree[rt].rs].r;  
 up(rt);  
}  
  
void update(int l, int r, LL c, int rt){  
 if(tree[rt].l >= l && tree[rt].r <= r){  
 (tree[rt].lazy += c) %= p, (tree[rt].sum += len(rt) \* c) %= p;  
 return ;  
 }  
 down(rt);  
 int mid = (tree[rt].l + tree[rt].r) >> 1;  
 if(mid >= l) update(l, r, c, tree[rt].ls);  
 if(mid < r) update(l, r, c, tree[rt].rs);  
 up(rt);  
}  
  
LL query(int l, int r, int rt){  
 if(tree[rt].l >= l && tree[rt].r <= r){  
 return tree[rt].sum;  
 }  
 down(rt);  
 int mid = (tree[rt].l + tree[rt].r) >> 1;  
 LL ans = 0;  
 if(mid >= l) ans += query(l, r, tree[rt].ls);  
 if(mid < r) ans += query(l, r, tree[rt].rs);  
 return ans % p;  
}  
  
void updates(int x, int y, LL c){  
 while(top[x] != top[y]){  
 if(d[top[x]] < d[top[y]]) swap(x, y);  
 update(id[top[x]], id[x], c, 0);  
 x = f[top[x]];  
 }  
 if(id[x] > id[y]) swap(x, y);  
 update(id[x], id[y], c, 0);  
}  
  
LL sum(int x, int y){  
 LL ans = 0;  
 while(top[x] != top[y]){  
 if(d[top[x]] < d[top[y]]) swap(x, y);  
 (ans += query(id[top[x]], id[x], 0)) %= p;  
 x = f[top[x]];  
 }  
 if(id[x] > id[y]) swap(x, y);  
 ans += query(id[x], id[y], 0);  
 return ans % p;  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 memset(head, -1, sizeof(head));  
 scanf("%lld %lld %lld %lld", &n, &m, &r, &p);  
 for (int i = 1; i <= n; i++){  
 scanf("%lld", &v[i]);  
 }  
 for (int u, v, i = 1; i < n; i++){  
 scanf("%d %d", &u, &v);  
 a[tot] = Node{v, head[u]}, head[u] = tot++;  
 a[tot] = Node{u, head[v]}, head[v] = tot++;  
 }  
 cnt = 0, dfs1(r), dfs2(r, r);  
 cnt = 1, build(1, n, 0);  
 for (int op, x, y, z, i = 1; i <= m; i++){  
 scanf("%d", &op);  
 if(op == 1){  
 scanf("%d %d %d", &x, &y, &z);  
 updates(x, y, z);  
 }else if(op == 2){  
 scanf("%d %d", &x, &y);  
 printf("%lld\n", sum(x, y));  
 }else if(op == 3){  
 scanf("%d %d", &x, &y);  
 update(id[x], id[x] + size[x] - 1, y, 0);  
 }else{  
 scanf("%d", &x);  
 printf("%lld\n", query(id[x], id[x] + size[x] - 1, 0));  
 }  
 }  
  
 return 0;  
}  
/\*\*/

## Dsu on tree

求树上子树最多能匹配多少个对颜色不同告诉你每个点的颜色和数目

/\*  
https://ac.nowcoder.com/acm/contest/6037/F  
\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
int n, tot, res;  
int head[200005], a[200005], son[200005], col[200005], ans[200005], sz[200005], cnt[200005], sum[200005];  
bool vis[200005];  
  
struct node  
{  
 int v, next;  
}e[200005 << 1];  
  
void dfs(int x, int pre){  
 sz[x] = 1;  
 sum[x] = a[x];  
 for (int i = head[x]; i != -1; i = e[i].next){  
 int v = e[i].v;  
 if(v == pre) continue;  
 dfs(v, x);  
 sz[x] += sz[v];  
 sum[x] += sum[v];  
 if(sz[son[x]] < sz[v] || !son[x]) son[x] = v;  
 }  
}  
  
void calc(int x, int fa, int val){  
 if(val > 0){  
 cnt[col[x]] += a[x];  
 res = max(res, cnt[col[x]]);  
 }else{  
 cnt[col[x]] -= a[x];  
 }  
 for (int i = head[x]; i != -1; i = e[i].next){  
 int v = e[i].v;  
 if(v == fa || vis[v]) continue;  
 calc(v, x, val);  
 }  
}  
  
void dfs1(int x, int pre, int keep){  
 for (int i = head[x]; i != -1; i = e[i].next){  
 int v = e[i].v;  
 if(v == pre || v == son[x]) continue;  
 dfs1(v, x, 0);  
 }  
 if(son[x]) dfs1(son[x], x, 1), vis[son[x]] = true;  
 calc(x, pre, 1);  
 vis[son[x]] = false;  
 if(res \* 2 <= sum[x]) ans[x] = sum[x] / 2;  
 else ans[x] = sum[x] - res;  
 if(!keep) calc(x, pre, -1), res = 0;  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 memset(head, -1, sizeof(head));  
 scanf("%d", &n);  
 for (int i = 1, u, v; i < n; i++){  
 scanf("%d %d", &u, &v);  
 e[tot] = node{v, head[u]}, head[u] = tot++;  
 e[tot] = node{u, head[v]}, head[v] = tot++;  
 }  
 for (int i = 1; i <= n; i++) scanf("%d %d", &col[i], &a[i]);  
 dfs(1, 0);  
 dfs1(1, 0, 0);  
 for (int i = 1; i <= n; i++) printf("%d\n", ans[i]);  
  
 return 0;  
}  
/\*\*/

## 虚树

inline bool cmp(const int x, const int y) { return id[x] < id[y]; }  
  
void build() {  
 sort(h + 1, h + k + 1, cmp);  
 sta[top = 1] = 1, g.sz = 0, g.head[1] = -1;  
 // 1 号节点入栈，清空 1 号节点对应的邻接表，设置邻接表边数为 1  
 for (int i = 1, l; i <= k; ++i)  
 if (h[i] != 1) {  
 //如果 1 号节点是关键节点就不要重复添加  
 l = lca(h[i], sta[top]);  
 //计算当前节点与栈顶节点的 LCA  
 if (l != sta[top]) {  
 //如果 LCA 和栈顶元素不同，则说明当前节点不再当前栈所存的链上  
 while (id[l] < id[sta[top - 1]])  
 //当次大节点的 Dfs 序大于 LCA 的 Dfs 序  
 g.push(sta[top - 1], sta[top]), top--;  
 //把与当前节点所在的链不重合的链连接掉并且弹出  
 if (id[l] > id[sta[top - 1]])  
 //如果 LCA 不等于次大节点（这里的大于其实和不等于没有区别）  
 g.head[l] = -1, g.push(l, sta[top]), sta[top] = l;  
 //说明 LCA 是第一次入栈，清空其邻接表，连边后弹出栈顶元素，并将 LCA 入栈  
 else  
 g.push(l, sta[top--]);  
 //说明 LCA 就是次大节点，直接弹出栈顶元素  
 }  
 g.head[h[i]] = -1, sta[++top] = h[i];  
 //当前节点必然是第一次入栈，清空邻接表并入栈  
 }  
 for (int i = 1; i < top; ++i)  
 g.push(sta[i], sta[i + 1]); //剩余的最后一条链连接一下  
 return;  
}

例题：求树上不同不同颜色点的距离和

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
const long long mod = 998244353;  
const int maxn = 200005;  
  
int n, type, tot, cnt, top, len;  
int c[maxn], b[maxn];  
int head[maxn], sz[maxn], son[maxn], topf[maxn], f[maxn], dep[maxn], dfn[maxn];  
LL ans, res, dis[maxn];  
int e[maxn], s[maxn], dp[maxn];  
bool vis[maxn];  
  
vector<int> v[maxn];  
vector<pair<int, LL> > g[maxn];  
  
struct node  
{  
 int v, w, next;  
}a[maxn << 1];  
  
bool cmp(int x, int y){  
 return dfn[x] < dfn[y];  
}  
  
void dfs(int x, int pre){  
 sz[x] = 1;  
 dep[x] = dep[pre] + 1;  
 f[x] = pre;  
 for (int i = head[x]; i != -1; i = a[i].next){  
 int v = a[i].v;  
 if(v == pre) continue;  
 dis[v] = (dis[x] + a[i].w) % mod;  
 dfs(v, x);  
 ans = (ans + 1LL \* sz[v] \* (n - sz[v]) % mod \* a[i].w % mod) % mod;  
 sz[x] += sz[v];  
 if(sz[son[x]] < sz[v]) son[x] = v;  
 }  
}  
  
void dfs1(int x, int topfa){  
 topf[x] = topfa;  
 dfn[x] = ++cnt;  
 if(!son[x]) return ;  
 dfs1(son[x], topfa);  
 for (int i = head[x]; i != -1; i = a[i].next){  
 int v = a[i].v;  
 if(topf[v]) continue;  
 dfs1(v, v);  
 }  
}  
  
int LCA(int x, int y){  
 while(topf[x] != topf[y]){  
 if(dep[topf[x]] < dep[topf[y]]) swap(x, y);  
 x = f[topf[x]];  
 }  
 if(dep[x] > dep[y]) swap(x, y);  
 return x;  
}  
  
void add\_edge(int u, int v){  
 if(u == n + 1) g[u].emplace\_back(make\_pair(v, 0));  
 else g[u].emplace\_back(make\_pair(v, (dis[v] - dis[u] + mod) % mod));  
}  
  
void insert(int x){  
 if(top <= 1){  
 s[++top] = x;  
 return ;  
 }  
 int lca = LCA(s[top], x);  
 if(lca == s[top]){  
 s[++top] = x;  
 return ;  
 }  
 while(top > 1 && dfn[lca] <= dfn[s[top - 1]]){  
 add\_edge(s[top - 1], s[top]);  
 top--;  
 }  
 if(lca != s[top]) add\_edge(lca, s[top]), s[top] = lca;  
 s[++top] = x;  
}  
  
void dfs2(int u){  
 dp[u] = vis[u];  
 for (auto x : g[u]){  
 int v = x.first;  
 LL w = x.second;  
 dfs2(v);  
 dp[u] += dp[v];  
 res = (res + 1LL \* dp[v] \* (len - dp[v]) % mod \* w % mod) % mod;  
 }  
 g[u].clear();  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 memset(head, -1, sizeof(head));  
 scanf("%d %d", &n, &type);  
 for (int i = 1; i <= n; i++) scanf("%d", &c[i]), b[i] = c[i];  
 sort(b + 1, b + 1 + n);  
 int num = unique(b + 1, b + 1 + n) - b - 1;  
 for (int i = 1; i <= n; i++) c[i] = lower\_bound(b + 1, b + 1 + num, c[i]) - b;  
 for (int i = 1; i <= n; i++) v[c[i]].emplace\_back(i);  
 for (int i = 1, u, v, w; i < n; i++){  
 scanf("%d %d %d", &u, &v, &w);  
 a[tot] = node{v, w, head[u]}, head[u] = tot++;  
 a[tot] = node{u, w, head[v]}, head[v] = tot++;  
 }  
 dfs(1, 0);  
 dfs1(1, 1);  
 for (int i = 1; i <= num; i++){  
 if(v[i].empty()) continue;  
 len = v[i].size();  
 for (int j = 0; j < len; j++) e[j + 1] = v[i][j], vis[e[j + 1]] = true;  
 sort(e + 1, e + 1 + len, cmp);  
 s[top = 1] = n + 1;  
 for (int j = 1; j <= len; j++) insert(e[j]);  
 while(top > 1) add\_edge(s[top - 1], s[top]), top--;  
 res = 0;  
 dfs2(n + 1);  
 for (int j = 1; j <= len; j++) vis[e[j]] = false;  
 ans = (ans - res + mod) % mod;  
 }  
 printf("%lld\n", (ans << 1) % mod);  
  
 return 0;  
}  
/\*  
8 3  
1 2 3 1 3 3 1 2  
1 2 1  
2 4 2  
2 5 2  
5 6 3  
5 7 3  
1 3 4  
3 8 4  
\*/

## 树分治

[luogu P3806【模板】点分治 1]:给定一棵有 n 个点的树，询问树上距离为 k的点对是否存在。

#include <algorithm>  
#include <cstdio>  
#include <cstring>  
#include <queue>  
using namespace std;  
const int maxn = 20010;  
const int inf = 2e9;  
int n, m, a, b, c, q[maxn], rt, siz[maxn], maxx[maxn], dist[maxn];  
int cur, h[maxn], nxt[maxn], p[maxn], w[maxn];  
bool tf[10000010], ret[maxn], vis[maxn];  
void add\_edge(int x, int y, int z) {  
 cur++;  
 nxt[cur] = h[x];  
 h[x] = cur;  
 p[cur] = y;  
 w[cur] = z;  
}  
int sum;  
void calcsiz(int x, int fa) {  
 siz[x] = 1;  
 maxx[x] = 0;  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != fa && !vis[p[j]]) {  
 calcsiz(p[j], x);  
 maxx[x] = max(maxx[x], siz[p[j]]);  
 siz[x] += siz[p[j]];  
 }  
 maxx[x] = max(maxx[x], sum - siz[x]);  
 if (maxx[x] < maxx[rt]) rt = x;  
}  
int dd[maxn], cnt;  
void calcdist(int x, int fa) {  
 dd[++cnt] = dist[x];  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != fa && !vis[p[j]])  
 dist[p[j]] = dist[x] + w[j], calcdist(p[j], x);  
}  
queue<int> tag;  
void dfz(int x, int fa) {  
 tf[0] = true;  
 tag.push(0);  
 vis[x] = true;  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != fa && !vis[p[j]]) {  
 dist[p[j]] = w[j];  
 calcdist(p[j], x);  
 for (int k = 1; k <= cnt; k++)  
 for (int i = 1; i <= m; i++)  
 if (q[i] >= dd[k]) ret[i] |= tf[q[i] - dd[k]];  
 for (int k = 1; k <= cnt; k++)  
 if (dd[k] < 10000010) tag.push(dd[k]), tf[dd[k]] = true;  
 cnt = 0;  
 }  
 while (!tag.empty()) tf[tag.front()] = false, tag.pop();  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != fa && !vis[p[j]]) {  
 sum = siz[p[j]];  
 rt = 0;  
 maxx[rt] = inf;  
 calcsiz(p[j], x);  
 calcsiz(rt, -1);  
 dfz(rt, x);  
 }  
}  
int main() {  
 scanf("%d%d", &n, &m);  
 for (int i = 1; i < n; i++)  
 scanf("%d%d%d", &a, &b, &c), add\_edge(a, b, c), add\_edge(b, a, c);  
 for (int i = 1; i <= m; i++) scanf("%d", q + i);  
 rt = 0;  
 maxx[rt] = inf;  
 sum = n;  
 calcsiz(1, -1);  
 calcsiz(rt, -1);  
 dfz(rt, -1);  
 for (int i = 1; i <= m; i++)  
 if (ret[i])  
 printf("AYE\n");  
 else  
 printf("NAY\n");  
 return 0;  
}

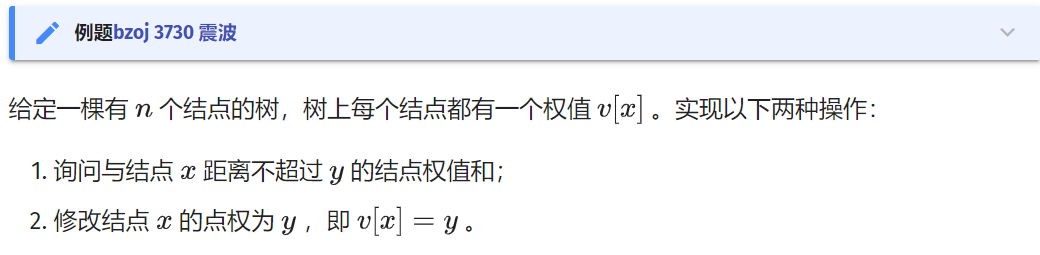
[luogu P4178 Tree]:由于这里查询的是树上距离为[0, k]的点对数量，所以我们用线段树来支持维护和查询。

#include <algorithm>  
#include <cstdio>  
#include <cstring>  
#include <queue>  
#define int long long  
using namespace std;  
const int maxn = 2000010;  
const int inf = 2e9;  
int n, a, b, c, q, rt, siz[maxn], maxx[maxn], dist[maxn];  
int cur, h[maxn], nxt[maxn], p[maxn], w[maxn], ret;  
bool vis[maxn];  
void add\_edge(int x, int y, int z) {  
 cur++;  
 nxt[cur] = h[x];  
 h[x] = cur;  
 p[cur] = y;  
 w[cur] = z;  
}  
int sum;  
void calcsiz(int x, int fa) {  
 siz[x] = 1;  
 maxx[x] = 0;  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != fa && !vis[p[j]]) {  
 calcsiz(p[j], x);  
 maxx[x] = max(maxx[x], siz[p[j]]);  
 siz[x] += siz[p[j]];  
 }  
 maxx[x] = max(maxx[x], sum - siz[x]);  
 if (maxx[x] < maxx[rt]) rt = x;  
}  
int dd[maxn], cnt;  
void calcdist(int x, int fa) {  
 dd[++cnt] = dist[x];  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != fa && !vis[p[j]])  
 dist[p[j]] = dist[x] + w[j], calcdist(p[j], x);  
}  
queue<int> tag;  
struct segtree {  
 int cnt, rt, lc[maxn], rc[maxn], sum[maxn];  
 void clear() {  
 while (!tag.empty()) update(rt, 1, 20000000, tag.front(), -1), tag.pop();  
 cnt = 0;  
 }  
 void print(int o, int l, int r) {  
 if (!o || !sum[o]) return;  
 if (l == r) {  
 printf("%lld %lld\n", l, sum[o]);  
 return;  
 }  
 int mid = (l + r) >> 1;  
 print(lc[o], l, mid);  
 print(rc[o], mid + 1, r);  
 }  
 void update(int& o, int l, int r, int x, int v) {  
 if (!o) o = ++cnt;  
 if (l == r) {  
 sum[o] += v;  
 if (!sum[o]) o = 0;  
 return;  
 }  
 int mid = (l + r) >> 1;  
 if (x <= mid)  
 update(lc[o], l, mid, x, v);  
 else  
 update(rc[o], mid + 1, r, x, v);  
 sum[o] = sum[lc[o]] + sum[rc[o]];  
 if (!sum[o]) o = 0;  
 }  
 int query(int o, int l, int r, int ql, int qr) {  
 if (!o) return 0;  
 if (r < ql || l > qr) return 0;  
 if (ql <= l && r <= qr) return sum[o];  
 int mid = (l + r) >> 1;  
 return query(lc[o], l, mid, ql, qr) + query(rc[o], mid + 1, r, ql, qr);  
 }  
} st;  
void dfz(int x, int fa) {  
 // tf[0]=true;tag.push(0);  
 st.update(st.rt, 1, 20000000, 1, 1);  
 tag.push(1);  
 vis[x] = true;  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != fa && !vis[p[j]]) {  
 dist[p[j]] = w[j];  
 calcdist(p[j], x);  
 for (int k = 1; k <= cnt; k++)  
 if (q - dd[k] >= 0)  
 ret += st.query(st.rt, 1, 20000000, max(0ll, 1 - dd[k]) + 1,  
 max(0ll, q - dd[k]) + 1);  
 for (int k = 1; k <= cnt; k++)  
 st.update(st.rt, 1, 20000000, dd[k] + 1, 1), tag.push(dd[k] + 1);  
 cnt = 0;  
 }  
 st.clear();  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != fa && !vis[p[j]]) {  
 sum = siz[p[j]];  
 rt = 0;  
 maxx[rt] = inf;  
 calcsiz(p[j], x);  
 calcsiz(rt, -1);  
 dfz(rt, x);  
 }  
}  
int main() {  
 scanf("%lld", &n);  
 for (int i = 1; i < n; i++)  
 scanf("%lld%lld%lld", &a, &b, &c), add\_edge(a, b, c), add\_edge(b, a, c);  
 scanf("%lld", &q);  
 rt = 0;  
 maxx[rt] = inf;  
 sum = n;  
 calcsiz(1, -1);  
 calcsiz(rt, -1);  
 dfz(rt, -1);  
 printf("%lld\n", ret);  
 return 0;  
}

## 动态点分治



#include <algorithm>  
#include <cstdio>  
#include <cstring>  
#include <queue>  
using namespace std;  
const int maxn = 100010;  
const int inf = 2e9;  
int n, a, b, m, x, col[maxn];  
// 0 off 1 on  
char op;  
int cur, h[maxn \* 2], nxt[maxn \* 2], p[maxn \* 2];  
void add\_edge(int x, int y) {  
 cur++;  
 nxt[cur] = h[x];  
 h[x] = cur;  
 p[cur] = y;  
}  
bool vis[maxn];  
int rt, sum, siz[maxn], maxx[maxn], fa[maxn], dep[maxn];  
void calcsiz(int x, int f) {  
 siz[x] = 1;  
 maxx[x] = 0;  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != f && !vis[p[j]]) {  
 calcsiz(p[j], x);  
 siz[x] += siz[p[j]];  
 maxx[x] = max(maxx[x], siz[p[j]]);  
 }  
 maxx[x] = max(maxx[x], sum - siz[x]);  
 if (maxx[x] < maxx[rt]) rt = x;  
}  
struct heap {  
 priority\_queue<int> A, B; // heap=A-B  
 void insert(int x) { A.push(x); }  
 void erase(int x) { B.push(x); }  
 int top() {  
 while (!B.empty() && A.top() == B.top()) A.pop(), B.pop();  
 return A.top();  
 }  
 void pop() {  
 while (!B.empty() && A.top() == B.top()) A.pop(), B.pop();  
 A.pop();  
 }  
 int top2() {  
 int t = top(), ret;  
 pop();  
 ret = top();  
 A.push(t);  
 return ret;  
 }  
 int size() { return A.size() - B.size(); }  
} dist[maxn], ch[maxn], ans;  
void dfs(int x, int f, int d, heap& y) {  
 y.insert(d);  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != f && !vis[p[j]]) dfs(p[j], x, d + 1, y);  
}  
void pre(int x) {  
 vis[x] = true;  
 for (int j = h[x]; j; j = nxt[j])  
 if (!vis[p[j]]) {  
 rt = 0;  
 maxx[rt] = inf;  
 sum = siz[p[j]];  
 calcsiz(p[j], -1);  
 calcsiz(rt, -1);  
 fa[rt] = x;  
 dfs(p[j], -1, 1, dist[rt]);  
 ch[x].insert(dist[rt].top());  
 dep[rt] = dep[x] + 1;  
 pre(rt);  
 }  
 ch[x].insert(0);  
 if (ch[x].size() >= 2)  
 ans.insert(ch[x].top() + ch[x].top2());  
 else if (ch[x].size())  
 ans.insert(ch[x].top());  
}  
struct LCA {  
 int dep[maxn], lg[maxn], fa[maxn][20];  
 void dfs(int x, int f) {  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != f) dep[p[j]] = dep[x] + 1, fa[p[j]][0] = x, dfs(p[j], x);  
 }  
 void init() {  
 dfs(1, -1);  
 for (int i = 2; i <= n; i++) lg[i] = lg[i / 2] + 1;  
 for (int j = 1; j <= lg[n]; j++)  
 for (int i = 1; i <= n; i++) fa[i][j] = fa[fa[i][j - 1]][j - 1];  
 }  
 int query(int x, int y) {  
 if (dep[x] > dep[y]) swap(x, y);  
 int k = dep[y] - dep[x];  
 for (int i = 0; k; k = k / 2, i++)  
 if (k & 1) y = fa[y][i];  
 if (x == y) return x;  
 k = dep[x];  
 for (int i = lg[k]; i >= 0; i--)  
 if (fa[x][i] != fa[y][i]) x = fa[x][i], y = fa[y][i];  
 return fa[x][0];  
 }  
 int dist(int x, int y) { return dep[x] + dep[y] - 2 \* dep[query(x, y)]; }  
} lca;  
int d[maxn][20];  
int main() {  
 scanf("%d", &n);  
 for (int i = 1; i < n; i++)  
 scanf("%d%d", &a, &b), add\_edge(a, b), add\_edge(b, a);  
 lca.init();  
 rt = 0;  
 maxx[rt] = inf;  
 sum = n;  
 calcsiz(1, -1);  
 calcsiz(rt, -1);  
 pre(rt);  
 // for(int i=1;i<=n;i++)printf("%d ",fa[i]);printf("\n");  
 for (int i = 1; i <= n; i++)  
 for (int j = i; j; j = fa[j]) d[i][dep[i] - dep[j]] = lca.dist(i, j);  
 scanf("%d", &m);  
 while (m--) {  
 scanf(" %c", &op);  
 if (op == 'G') {  
 if (ans.size())  
 printf("%d\n", ans.top());  
 else  
 printf("-1\n");  
 } else {  
 scanf("%d", &x);  
 if (!col[x]) {  
 if (ch[x].size() >= 2) ans.erase(ch[x].top() + ch[x].top2());  
 ch[x].erase(0);  
 if (ch[x].size() >= 2) ans.insert(ch[x].top() + ch[x].top2());  
 for (int i = x; fa[i]; i = fa[i]) {  
 if (ch[fa[i]].size() >= 2)  
 ans.erase(ch[fa[i]].top() + ch[fa[i]].top2());  
 ch[fa[i]].erase(dist[i].top());  
 dist[i].erase(d[x][dep[x] - dep[fa[i]]]);  
 if (dist[i].size()) ch[fa[i]].insert(dist[i].top());  
 if (ch[fa[i]].size() >= 2)  
 ans.insert(ch[fa[i]].top() + ch[fa[i]].top2());  
 }  
 } else {  
 if (ch[x].size() >= 2) ans.erase(ch[x].top() + ch[x].top2());  
 ch[x].insert(0);  
 if (ch[x].size() >= 2) ans.insert(ch[x].top() + ch[x].top2());  
 for (int i = x; fa[i]; i = fa[i]) {  
 if (ch[fa[i]].size() >= 2)  
 ans.erase(ch[fa[i]].top() + ch[fa[i]].top2());  
 if (dist[i].size()) ch[fa[i]].erase(dist[i].top());  
 dist[i].insert(d[x][dep[x] - dep[fa[i]]]);  
 ch[fa[i]].insert(dist[i].top());  
 if (ch[fa[i]].size() >= 2)  
 ans.insert(ch[fa[i]].top() + ch[fa[i]].top2());  
 }  
 }  
 col[x] ^= 1;  
 }  
 }  
 return 0;  
}



#include <algorithm>  
#include <cstdio>  
#include <cstring>  
using namespace std;  
const int maxn = 100010;  
const int inf = 2e9;  
const int ddd = 6000010;  
struct Segtree {  
 int cnt, rt[maxn], sum[ddd], lc[ddd], rc[ddd];  
 void update(int& o, int l, int r, int x, int v) {  
 if (!o) o = ++cnt;  
 if (l == r) {  
 sum[o] += v;  
 return;  
 }  
 int mid = (l + r) >> 1;  
 if (x <= mid)  
 update(lc[o], l, mid, x, v);  
 else  
 update(rc[o], mid + 1, r, x, v);  
 sum[o] = sum[lc[o]] + sum[rc[o]];  
 }  
 int query(int o, int l, int r, int ql, int qr) {  
 if (!o || r < ql || l > qr) return 0;  
 if (ql <= l && r <= qr) return sum[o];  
 int mid = (l + r) >> 1;  
 return query(lc[o], l, mid, ql, qr) + query(rc[o], mid + 1, r, ql, qr);  
 }  
} dist, ch;  
int n, m, val[maxn], u, v, op, x, y, lstans;  
int cur, h[maxn \* 2], nxt[maxn \* 2], p[maxn \* 2];  
void add\_edge(int x, int y) {  
 cur++;  
 nxt[cur] = h[x];  
 h[x] = cur;  
 p[cur] = y;  
}  
struct LCA {  
 int dep[maxn], lg[maxn], fa[maxn][20];  
 void dfs(int x, int f) {  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != f) dep[p[j]] = dep[x] + 1, fa[p[j]][0] = x, dfs(p[j], x);  
 }  
 void init() {  
 dep[1] = 1;  
 dfs(1, -1);  
 for (int i = 2; i <= n; i++) lg[i] = lg[i / 2] + 1;  
 for (int j = 1; j <= lg[n]; j++)  
 for (int i = 1; i <= n; i++) fa[i][j] = fa[fa[i][j - 1]][j - 1];  
 }  
 int query(int x, int y) {  
 if (dep[x] > dep[y]) swap(x, y);  
 int k = dep[y] - dep[x];  
 for (int i = 0; k; k = k / 2, i++)  
 if (k & 1) y = fa[y][i];  
 if (x == y) return x;  
 k = dep[x];  
 for (int i = lg[k]; i >= 0; i--)  
 if (fa[x][i] != fa[y][i]) x = fa[x][i], y = fa[y][i];  
 return fa[x][0];  
 }  
 int dist(int x, int y) { return dep[x] + dep[y] - 2 \* dep[query(x, y)]; }  
} lca;  
int rt, sum, siz[maxn], maxx[maxn], fa[maxn];  
int d[maxn][20], dep[maxn];  
bool vis[maxn];  
void calcsiz(int x, int fa) {  
 siz[x] = 1;  
 maxx[x] = 0;  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != fa && !vis[p[j]]) {  
 calcsiz(p[j], x);  
 siz[x] += siz[p[j]];  
 maxx[x] = max(maxx[x], siz[p[j]]);  
 }  
 maxx[x] = max(maxx[x], sum - siz[x]);  
 if (maxx[x] < maxx[rt]) rt = x;  
}  
void dfs1(int x, int fa, int y, int d) {  
 ch.update(ch.rt[y], 0, n, d, val[x]);  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != fa && !vis[p[j]]) dfs1(p[j], x, y, d + 1);  
}  
void dfs2(int x, int fa, int y, int d) {  
 dist.update(dist.rt[y], 0, n, d, val[x]);  
 for (int j = h[x]; j; j = nxt[j])  
 if (p[j] != fa && !vis[p[j]]) dfs2(p[j], x, y, d + 1);  
}  
void pre(int x) {  
 vis[x] = true;  
 dfs2(x, -1, x, 0);  
 for (int j = h[x]; j; j = nxt[j])  
 if (!vis[p[j]]) {  
 rt = 0;  
 maxx[rt] = inf;  
 sum = siz[p[j]];  
 calcsiz(p[j], -1);  
 calcsiz(rt, -1);  
 dfs1(p[j], -1, rt, 1);  
 fa[rt] = x;  
 dep[rt] = dep[x] + 1;  
 pre(rt);  
 }  
}  
int main() {  
 scanf("%d%d", &n, &m);  
 for (int i = 1; i <= n; i++) scanf("%d", val + i);  
 for (int i = 1; i < n; i++)  
 scanf("%d%d", &u, &v), add\_edge(u, v), add\_edge(v, u);  
 lca.init();  
 rt = 0;  
 maxx[rt] = inf;  
 sum = n;  
 calcsiz(1, -1);  
 calcsiz(rt, -1);  
 pre(rt);  
 // for(int i=1;i<=n;i++)printf("%d ",fa[i]);printf("\n");  
 for (int i = 1; i <= n; i++)  
 for (int j = i; j; j = fa[j]) d[i][dep[i] - dep[j]] = lca.dist(i, j);  
 while (m--) {  
 scanf("%d%d%d", &op, &x, &y);  
 x ^= lstans;  
 y ^= lstans;  
 if (op == 0) {  
 lstans = dist.query(dist.rt[x], 0, n, 0, y);  
 int nww = 0;  
 for (int i = x; fa[i]; i = fa[i]) {  
 nww = d[x][dep[x] - dep[fa[i]]]; // lca.dist(x,fa[i]);  
 lstans += dist.query(dist.rt[fa[i]], 0, n, 0, y - nww);  
 lstans -= ch.query(ch.rt[i], 0, n, 0, y - nww);  
 }  
 printf("%d\n", lstans);  
 }  
 if (op == 1) {  
 int nww = 0;  
 dist.update(dist.rt[x], 0, n, 0, y - val[x]);  
 for (int i = x; fa[i]; i = fa[i]) {  
 nww = d[x][dep[x] - dep[fa[i]]]; // lca.dist(x,fa[i]);  
 dist.update(dist.rt[fa[i]], 0, n, nww, y - val[x]);  
 ch.update(ch.rt[i], 0, n, nww, y - val[x]);  
 }  
 val[x] = y;  
 }  
 }  
 return 0;  
}

## AHU算法

[SPOJ-TREEISO](https://www.spoj.com/problems/TREEISO/en/)

AHU 算法用于判断两棵有根树是否同构。

// Tree Isomorphism, O(nlogn)  
// replace quick sort with radix sort ==> O(n)  
// Author: \_Backl1ght  
#include <bits/stdc++.h>  
using namespace std;  
typedef long long ll;  
const int N = 1e5 + 5;  
const int maxn = N << 1;  
  
int n;  
struct Edge {  
 int v, nxt;  
} e[maxn << 1];  
int head[maxn], sz[maxn], f[maxn], maxv[maxn], tag[maxn], tot, Max;  
vector<int> center[2], L[maxn], subtree\_tags[maxn];  
void addedge(int u, int v) {  
 e[tot].v = v;  
 e[tot].nxt = head[u];  
 head[u] = tot++;  
 e[tot].v = u;  
 e[tot].nxt = head[v];  
 head[v] = tot++;  
}  
  
void dfs\_size(int u, int fa) {  
 sz[u] = 1;  
 maxv[u] = 0;  
 for (int i = head[u]; i; i = e[i].nxt) {  
 int v = e[i].v;  
 if (v == fa) continue;  
 dfs\_size(v, u);  
 sz[u] += sz[v];  
 maxv[u] = max(maxv[u], sz[v]);  
 }  
}  
  
void dfs\_center(int rt, int u, int fa, int id) {  
 maxv[u] = max(maxv[u], sz[rt] - sz[u]);  
 if (Max > maxv[u]) {  
 center[id].clear();  
 Max = maxv[u];  
 }  
 if (Max == maxv[u]) center[id].push\_back(u);  
 for (int i = head[u]; i; i = e[i].nxt) {  
 int v = e[i].v;  
 if (v == fa) continue;  
 dfs\_center(rt, v, u, id);  
 }  
}  
  
int dfs\_height(int u, int fa, int depth) {  
 L[depth].push\_back(u);  
 f[u] = fa;  
 int h = 0;  
 for (int i = head[u]; i; i = e[i].nxt) {  
 int v = e[i].v;  
 if (v == fa) continue;  
 h = max(h, dfs\_height(v, u, depth + 1));  
 }  
 return h + 1;  
}  
  
void init(int n) {  
 for (int i = 1; i <= 2 \* n; i++) head[i] = 0;  
 tot = 1;  
 center[0].clear();  
 center[1].clear();  
  
 int u, v;  
 for (int i = 1; i <= n - 1; i++) {  
 scanf("%d %d", &u, &v);  
 addedge(u, v);  
 }  
 dfs\_size(1, -1);  
 Max = n;  
 dfs\_center(1, 1, -1, 0);  
  
 for (int i = 1; i <= n - 1; i++) {  
 scanf("%d %d", &u, &v);  
 addedge(u + n, v + n);  
 }  
 dfs\_size(1 + n, -1);  
 Max = n;  
 dfs\_center(1 + n, 1 + n, -1, 1);  
}  
  
bool cmp(int u, int v) { return subtree\_tags[u] < subtree\_tags[v]; }  
  
bool rootedTreeIsomorphism(int rt1, int rt2) {  
 for (int i = 0; i <= 2 \* n + 1; i++) L[i].clear(), subtree\_tags[i].clear();  
 int h1 = dfs\_height(rt1, -1, 0);  
 int h2 = dfs\_height(rt2, -1, 0);  
 if (h1 != h2) return false;  
 int h = h1 - 1;  
 for (int j = 0; j < (int)L[h].size(); j++) tag[L[h][j]] = 0;  
 for (int i = h - 1; i >= 0; i--) {  
 for (int j = 0; j < (int)L[i + 1].size(); j++) {  
 int v = L[i + 1][j];  
 subtree\_tags[f[v]].push\_back(tag[v]);  
 }  
  
 sort(L[i].begin(), L[i].end(), cmp);  
  
 for (int j = 0, cnt = 0; j < (int)L[i].size(); j++) {  
 if (j && subtree\_tags[L[i][j]] != subtree\_tags[L[i][j - 1]]) ++cnt;  
 tag[L[i][j]] = cnt;  
 }  
 }  
 return subtree\_tags[rt1] == subtree\_tags[rt2];  
}  
  
bool treeIsomorphism() {  
 if (center[0].size() == center[1].size()) {  
 if (rootedTreeIsomorphism(center[0][0], center[1][0])) return true;  
 if (center[0].size() > 1)  
 return rootedTreeIsomorphism(center[0][0], center[1][1]);  
 }  
 return false;  
}  
  
int main() {  
 int T;  
 scanf("%d", &T);  
 while (T--) {  
 scanf("%d", &n);  
 init(n);  
 puts(treeIsomorphism() ? "YES" : "NO");  
 }  
 return 0;  
}

## 树哈希

### [「BJOI2015」树的同构](https://www.luogu.com.cn/problem/P5043)

#include <algorithm>  
#include <cstdio>  
#include <tr1/unordered\_map>  
#include <vector>  
  
class Solution {  
 private:  
 typedef unsigned long long ull;  
 typedef std::vector<int>::iterator it;  
 static const ull seed = 2333233233;  
 static const int maxn = 107;  
  
 int n, m, size[maxn], lastRoot, root, lastMax, Max, ans;  
 ull hashval[maxn], res;  
 std::vector<int> e[maxn];  
 std::tr1::unordered\_map<ull, int> id;  
  
 ull getHash(int now, int fa) {  
 size[now] = 1;  
 hashval[now] = 1;  
 for (register it i = e[now].begin(); i != e[now].end(); ++i) {  
 int v = \*i;  
 if (v == fa) {  
 continue;  
 }  
 hashval[now] ^= getHash(v, now) \* seed + size[v];  
 size[now] += size[v];  
 }  
 return hashval[now];  
 }  
  
 void getRoot(int now, int fa) {  
 int max = 0;  
 size[now] = 1;  
 for (register it i = e[now].begin(); i != e[now].end(); ++i) {  
 int v = \*i;  
 if (v == fa) {  
 continue;  
 }  
 getRoot(v, now);  
 size[now] += size[v];  
 max = std::max(max, size[v]);  
 }  
 max = std::max(max, n - size[now]);  
 if (max < Max && now != lastRoot) {  
 root = now;  
 Max = max;  
 }  
 }  
  
 public:  
 Solution() {  
 get();  
 solve();  
 }  
  
 void get() {  
 scanf("%d", &m);  
 for (register int i = 1; i <= m; i++) {  
 scanf("%d", &n);  
 for (register int j = 1; j <= n; j++) {  
 std::vector<int>().swap(e[j]);  
 }  
 for (register int j = 1, fa; j <= n; j++) {  
 scanf("%d", &fa);  
 if (!fa) {  
 root = j;  
 } else {  
 e[fa].push\_back(j);  
 e[j].push\_back(fa);  
 }  
 }  
 lastRoot = root = 0;  
 Max = n;  
 getRoot(1, 0);  
 lastRoot = root, lastMax = Max;  
 res = getHash(root, 0);  
 if (!id.count(res)) {  
 id[res] = i;  
 }  
 ans = id[res];  
  
 Max = n;  
 getRoot(1, 0);  
 if (lastMax == Max) {  
 res = getHash(root, 0);  
 if (!id.count(res)) {  
 id[res] = i;  
 }  
 ans = std::min(ans, id[res]);  
 }  
 printf("%d\n", ans);  
 }  
 }  
  
 void solve() {}  
};  
Solution sol;  
  
int main() {}

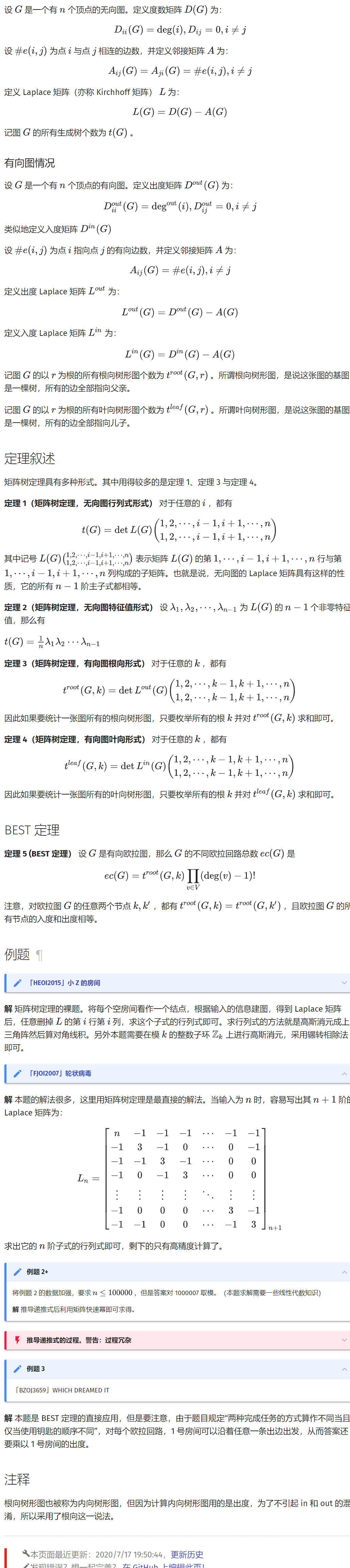
### [HDU 6647](http://acm.hdu.edu.cn/showproblem.php?pid=6647)

题目要求的是遍历一棵无根树产生的本质不同括号序列方案数。

#include <algorithm>  
#include <cmath>  
#include <cstdio>  
#include <cstring>  
#include <iostream>  
#include <map>  
#include <set>  
#include <utility>  
#include <vector>  
using namespace std;  
typedef long long ll;  
typedef unsigned long long ull;  
typedef pair<int, int> PII;  
const int mod = 998244353;  
const int inf = 1 << 30;  
const int maxn = 100000 + 5;  
namespace sieve {  
const int maxp = 2000000 + 5;  
int vis[maxp], prime[maxp], tot;  
void init() {  
 ms(vis, 0);  
 for (int i = 2; i < maxp; i++) {  
 if (!vis[i]) prime[++tot] = i;  
 for (int j = 1; j <= tot && prime[j] \* i < maxp; j++) {  
 vis[i \* prime[j]] = 1;  
 if (i % prime[j] == 0) break;  
 }  
 }  
}  
} // namespace sieve  
namespace MyIO {  
struct fastIO {  
 char s[100000];  
 int it, len;  
 fastIO() { it = len = 0; }  
 inline char get() {  
 if (it < len) return s[it++];  
 it = 0;  
 len = fread(s, 1, 100000, stdin);  
 if (len == 0)  
 return EOF;  
 else  
 return s[it++];  
 }  
 bool notend() {  
 char c = get();  
 while (c == ' ' || c == '\n') c = get();  
 if (it > 0) it--;  
 return c != EOF;  
 }  
} buff;  
inline int gi() {  
 int r = 0;  
 bool ng = 0;  
 char c = buff.get();  
 while (c != '-' && (c < '0' || c > '9')) c = buff.get();  
 if (c == '-') ng = 1, c = buff.get();  
 while (c >= '0' && c <= '9') r = r \* 10 + c - '0', c = buff.get();  
 return ng ? -r : r;  
}  
} // namespace MyIO  
namespace {  
inline int add(int x, int y) {  
 x += y;  
 return x >= mod ? x -= mod : x;  
}  
inline int sub(int x, int y) {  
 x -= y;  
 return x < 0 ? x += mod : x;  
}  
inline int mul(int x, int y) { return 1ll \* x \* y % mod; }  
inline int qpow(int x, ll n) {  
 int r = 1;  
 while (n > 0) {  
 if (n & 1) r = 1ll \* r \* x % mod;  
 n >>= 1;  
 x = 1ll \* x \* x % mod;  
 }  
 return r;  
}  
inline int inv(int x) { return qpow(x, mod - 2); }  
} // namespace  
using MyIO::gi;  
using sieve::prime;  
int ping[maxn], pingv[maxn];  
int n, ans, siz[maxn];  
vector<int> edge[maxn];  
map<ull, int> uqc[maxn];  
map<ull, int>::iterator it;  
ull hashval[maxn], hashrt[maxn];  
ull srchashval[maxn], srchashrt[maxn];  
int dp[maxn], rdp[maxn];  
ull pack(ull val, int sz) { return 2ull + 3ull \* val + 7ull \* prime[sz + 1]; }  
void predfs(int u, int ff) {  
 siz[u] = dp[u] = 1;  
 hashval[u] = 1;  
 int sz = 0;  
 for (int v : edge[u]) {  
 if (v == ff) continue;  
 predfs(v, u);  
 sz++;  
 siz[u] += siz[v];  
 dp[u] = mul(dp[u], dp[v]);  
 uqc[u][hashval[v]]++;  
 hashval[u] += hashval[v] \* prime[siz[v]];  
 }  
 srchashval[u] = hashval[u];  
 hashval[u] = pack(hashval[u], siz[u]);  
 dp[u] = mul(dp[u], ping[sz]);  
 for (it = uqc[u].begin(); it != uqc[u].end(); it++) {  
 dp[u] = mul(dp[u], pingv[it->second]);  
 }  
}  
set<ull> qc;  
void dfs(int u, int ff) {  
 if (!qc.count(hashrt[u])) {  
 qc.insert(hashrt[u]);  
 ans = add(ans, rdp[u]);  
 }  
 for (int v : edge[u]) {  
 if (v == ff) continue;  
 ull tmp = srchashrt[u] - hashval[v] \* prime[siz[v]];  
 tmp = pack(tmp, n - siz[v]);  
 uqc[v][tmp]++;  
 srchashrt[v] = srchashval[v] + tmp \* prime[n - siz[v]];  
 hashrt[v] = pack(srchashrt[v], n);  
 int tdp = mul(rdp[u], inv(dp[v]));  
 tdp = mul(tdp, inv((int)edge[u].size()));  
 tdp = mul(tdp, uqc[u][hashval[v]]);  
 rdp[v] = mul(dp[v], tdp);  
 rdp[v] = mul(rdp[v], (int)edge[v].size());  
 rdp[v] = mul(rdp[v], inv(uqc[v][tmp]));  
 dfs(v, u);  
 }  
}  
int main() {  
 sieve::init();  
 ping[0] = pingv[0] = 1;  
 for (int i = 1; i < maxn; i++) {  
 ping[i] = mul(ping[i - 1], i);  
 pingv[i] = mul(pingv[i - 1], inv(i));  
 }  
 int T = gi();  
 while (T--) {  
 n = gi();  
 for (int i = 2, u, v; i <= n; i++) {  
 u = gi();  
 v = gi();  
 edge[u].push\_back(v);  
 edge[v].push\_back(u);  
 }  
 predfs(1, 0);  
 ans = 0;  
 qc.clear();  
 rdp[1] = dp[1];  
 hashrt[1] = hashval[1];  
 srchashrt[1] = srchashval[1];  
 dfs(1, 0);  
 printf("%d\n", ans);  
 for (int i = 1; i <= n; i++) {  
 edge[i].clear();  
 uqc[i].clear();  
 }  
 }  
 return 0;  
}

## 矩阵树定理

Kirchhoff 矩阵树定理（简称矩阵树定理）解决了一张图的生成树个数计数问题。



HEOI2015」小 Z 的房间

#include <bits/stdc++.h>  
using namespace std;  
  
#define int long long  
const int N = 15, MOD = 1e9;  
int dx[4] = { -1, 0, 0, 1 };  
int dy[4] = { 0, 1, -1, 0 };  
int f[N \* N][N \* N], n, m, tot, a[N][N];  
char ch[N];  
  
int gauss() {  
 int ans = 1;  
 for (int i = 1; i < tot; i++) {  
 for (int j = i + 1; j < tot; j++) {  
 while (f[j][i]) {  
 int t = f[i][i] / f[j][i];  
 for (int k = i; k < tot; k++) f[i][k] = (f[i][k] - t \* f[j][k] + MOD) % MOD;  
 swap(f[i], f[j]);  
 ans = -ans;  
 }  
 }  
 ans = ans \* f[i][i] % MOD;  
 }  
 return (ans % MOD + MOD) % MOD;  
}  
  
signed main() {  
 scanf("%lld %lld", &n, &m);  
 for (int i = 1; i <= n; i++) {  
 scanf("%s", ch + 1);  
 for (int j = 1; j <= m; j++) {  
 if (ch[j] == '.')  
 a[i][j] = ++tot;  
 }  
 }  
 for (int i = 1; i <= n; i++)  
 for (int j = 1; j <= m; j++) {  
 for (int k = 0; k < 4; k++) {  
 int x = i + dx[k], y = j + dy[k];  
 if (x > n || x < 1 || y < 1 || y > m)  
 continue;  
 if (a[x][y] && a[i][j] && a[x][y] < a[i][j]) {  
 int u = a[x][y], v = a[i][j];  
 f[u][u]++, f[v][v]++;  
 f[v][u]--, f[u][v]--;  
 }  
 }  
 }  
 printf("%lld\n", gauss());  
 return 0;  
}

## 生成树

### 最小生成树唯一性

考虑最小生成树的唯一性。如果一条边 **不在最小生成树的边集中** ，并且可以替换与其 **权值相同、并且在最小生成树边集** 的另一条边。那么，这个最小生成树就是不唯一的。

对于 Kruskal 算法，只要计算为当前权值的边可以放几条，实际放了几条，如果这两个值不一样，那么就说明这几条边与之前的边产生了一个环（这个环中至少有两条当前权值的边，否则根据并查集，这条边是不能放的），即最小生成树不唯一。

寻找权值与当前边相同的边，我们只需要记录头尾指针，用单调队列即可在 O(α (m))（m 为边数）的时间复杂度里优秀解决这个问题（基本与原算法时间相同）

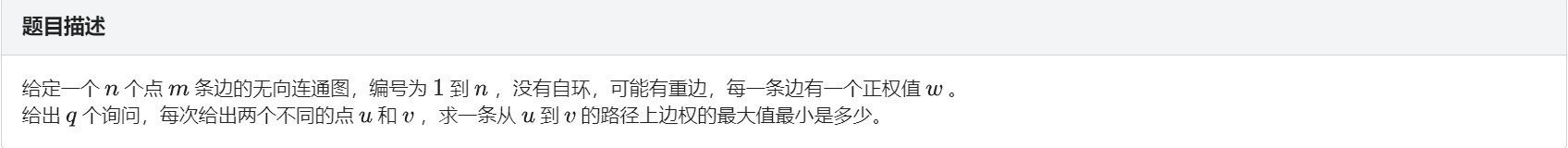
#include <algorithm>  
#include <cstdio>  
  
struct Edge {  
 int x, y, z;  
};  
int f[100001];  
Edge a[100001];  
int cmp(const Edge& a, const Edge& b) { return a.z < b.z; }  
int find(int x) { return f[x] == x ? x : f[x] = find(f[x]); }  
int main() {  
 int t;  
 scanf("%d", &t);  
 while (t--) {  
 int n, m;  
 scanf("%d%d", &n, &m);  
 for (int i = 1; i <= n; i++) f[i] = i;  
 for (int i = 1; i <= m; i++) scanf("%d%d%d", &a[i].x, &a[i].y, &a[i].z);  
 sort(a + 1, a + m + 1, cmp);  
 int num = 0, ans = 0, tail = 0, sum1 = 0, sum2 = 0;  
 bool flag = 1;  
 for (int i = 1; i <= m + 1; i++) {  
 if (i > tail) {  
 if (sum1 != sum2) {  
 flag = 0;  
 break;  
 }  
 sum1 = 0;  
 for (int j = i; j <= m + 1; j++) {  
 if (a[j].z != a[i].z) {  
 tail = j - 1;  
 break;  
 }  
 if (find(a[j].x) != find(a[j].y)) ++sum1;  
 }  
 sum2 = 0;  
 }  
 if (i > m) break;  
 int x = find(a[i].x);  
 int y = find(a[i].y);  
 if (x != y && num != n - 1) {  
 sum2++;  
 num++;  
 f[x] = f[y];  
 ans += a[i].z;  
 }  
 }  
 if (flag)  
 printf("%d\n", ans);  
 else  
 printf("Not Unique!\n");  
 }  
 return 0;  
}

### 严格次小生成树

#include <algorithm>  
#include <iostream>  
  
const int INF = 0x3fffffff;  
const long long INF64 = 0x3fffffffffffffffLL;  
  
struct Edge {  
 int u, v, val;  
 bool operator<(const Edge &other) const { return val < other.val; }  
};  
  
Edge e[300010];  
bool used[300010];  
  
int n, m;  
long long sum;  
  
class Tr {  
 private:  
 struct Edge {  
 int to, nxt, val;  
 } e[600010];  
 int cnt, head[100010];  
  
 int pnt[100010][22];  
 int dpth[100010];  
 // 到祖先的路径上边权最大的边  
 int maxx[100010][22];  
 // 到祖先的路径上边权次大的边，若不存在则为 -INF  
 int minn[100010][22];  
  
 public:  
 void addedge(int u, int v, int val) {  
 e[++cnt] = (Edge){v, head[u], val};  
 head[u] = cnt;  
 }  
  
 void insedge(int u, int v, int val) {  
 addedge(u, v, val);  
 addedge(v, u, val);  
 }  
  
 void dfs(int now, int fa) {  
 dpth[now] = dpth[fa] + 1;  
 pnt[now][0] = fa;  
 minn[now][0] = -INF;  
 for (int i = 1; (1 << i) <= dpth[now]; i++) {  
 pnt[now][i] = pnt[pnt[now][i - 1]][i - 1];  
 int kk[4] = {maxx[now][i - 1], maxx[pnt[now][i - 1]][i - 1],  
 minn[now][i - 1], minn[pnt[now][i - 1]][i - 1]};  
 // 从四个值中取得最大值  
 std::sort(kk, kk + 4);  
 maxx[now][i] = kk[3];  
 // 取得严格次大值  
 int ptr = 2;  
 while (ptr >= 0 && kk[ptr] == kk[3]) ptr--;  
 minn[now][i] = (ptr == -1 ? -INF : kk[ptr]);  
 }  
  
 for (int i = head[now]; i; i = e[i].nxt) {  
 if (e[i].to != fa) {  
 maxx[e[i].to][0] = e[i].val;  
 dfs(e[i].to, now);  
 }  
 }  
 }  
  
 int lca(int a, int b) {  
 if (dpth[a] < dpth[b]) std::swap(a, b);  
  
 for (int i = 21; i >= 0; i--)  
 if (dpth[pnt[a][i]] >= dpth[b]) a = pnt[a][i];  
  
 if (a == b) return a;  
  
 for (int i = 21; i >= 0; i--) {  
 if (pnt[a][i] != pnt[b][i]) {  
 a = pnt[a][i];  
 b = pnt[b][i];  
 }  
 }  
 return pnt[a][0];  
 }  
  
 int query(int a, int b, int val) {  
 int res = -INF;  
 for (int i = 21; i >= 0; i--) {  
 if (dpth[pnt[a][i]] >= dpth[b]) {  
 if (val != maxx[a][i])  
 res = std::max(res, maxx[a][i]);  
 else  
 res = std::max(res, minn[a][i]);  
 a = pnt[a][i];  
 }  
 }  
 return res;  
 }  
} tr;  
  
int fa[100010];  
int find(int x) { return fa[x] == x ? x : fa[x] = find(fa[x]); }  
  
void Kruskal() {  
 int tot = 0;  
 std::sort(e + 1, e + m + 1);  
 for (int i = 1; i <= n; i++) fa[i] = i;  
  
 for (int i = 1; i <= m; i++) {  
 int a = find(e[i].u);  
 int b = find(e[i].v);  
 if (a != b) {  
 fa[a] = b;  
 tot++;  
 tr.insedge(e[i].u, e[i].v, e[i].val);  
 sum += e[i].val;  
 used[i] = 1;  
 }  
 if (tot == n - 1) break;  
 }  
}  
  
int main() {  
 std::ios::sync\_with\_stdio(0);  
 std::cin.tie(0);  
 std::cout.tie(0);  
  
 std::cin >> n >> m;  
 for (int i = 1; i <= m; i++) {  
 int u, v, val;  
 std::cin >> u >> v >> val;  
 e[i] = (Edge){u, v, val};  
 }  
  
 Kruskal();  
 long long ans = INF64;  
 tr.dfs(1, 0);  
  
 for (int i = 1; i <= m; i++) {  
 if (!used[i]) {  
 int \_lca = tr.lca(e[i].u, e[i].v);  
 // 找到路径上不等于 e[i].val 的最大边权  
 long long tmpa = tr.query(e[i].u, \_lca, e[i].val);  
 long long tmpb = tr.query(e[i].v, \_lca, e[i].val);  
 // 这样的边可能不存在，只在这样的边存在时更新答案  
 if (std::max(tmpa, tmpb) > -INF)  
 ans = std::min(ans, sum - std::max(tmpa, tmpb) + e[i].val);  
 }  
 }  
 // 次小生成树不存在时输出 -1  
 std::cout << (ans == INF64 ? -1 : ans) << '\n';  
 return 0;  
}

### Kruskal 重构树

最小瓶颈路 加强版



#include <bits/stdc++.h>  
  
using namespace std;  
  
const int MAX\_VAL\_RANGE = 280010;  
  
int n, m, log2Values[MAX\_VAL\_RANGE + 1];  
  
namespace TR {  
struct Edge {  
 int to, nxt, val;  
} e[400010];  
int cnt, head[140010];  
  
void addedge(int u, int v, int val = 0) {  
 e[++cnt] = (Edge){v, head[u], val};  
 head[u] = cnt;  
}  
  
int val[140010];  
namespace LCA {  
int sec[280010], cnt;  
int pos[140010];  
int dpth[140010];  
  
void dfs(int now, int fa) {  
 dpth[now] = dpth[fa] + 1;  
 sec[++cnt] = now;  
 pos[now] = cnt;  
  
 for (int i = head[now]; i; i = e[i].nxt) {  
 if (fa != e[i].to) {  
 dfs(e[i].to, now);  
 sec[++cnt] = now;  
 }  
 }  
}  
  
int dp[280010][20];  
void init() {  
 dfs(2 \* n - 1, 0);  
 for (int i = 1; i <= 4 \* n; i++) {  
 dp[i][0] = sec[i];  
 }  
 for (int j = 1; j <= 19; j++) {  
 for (int i = 1; i + (1 << j) - 1 <= 4 \* n; i++) {  
 dp[i][j] = dpth[dp[i][j - 1]] < dpth[dp[i + (1 << (j - 1))][j - 1]]  
 ? dp[i][j - 1]  
 : dp[i + (1 << (j - 1))][j - 1];  
 }  
 }  
}  
  
int lca(int x, int y) {  
 int l = pos[x], r = pos[y];  
 if (l > r) {  
 swap(l, r);  
 }  
 int k = log2Values[r - l + 1];  
 return dpth[dp[l][k]] < dpth[dp[r - (1 << k) + 1][k]]  
 ? dp[l][k]  
 : dp[r - (1 << k) + 1][k];  
}  
} // namespace LCA  
} // namespace TR  
  
using TR::addedge;  
  
namespace GR {  
struct Edge {  
 int u, v, val;  
  
 bool operator<(const Edge &other) const { return val < other.val; }  
} e[100010];  
  
int fa[140010];  
  
int find(int x) { return fa[x] == 0 ? x : fa[x] = find(fa[x]); }  
  
void kruskal() {  
 int tot = 0, cnt = n;  
 sort(e + 1, e + m + 1);  
 for (int i = 1; i <= m; i++) {  
 int fau = find(e[i].u), fav = find(e[i].v);  
 if (fau != fav) {  
 cnt++;  
 fa[fau] = fa[fav] = cnt;  
 addedge(fau, cnt);  
 addedge(cnt, fau);  
 addedge(fav, cnt);  
 addedge(cnt, fav);  
 TR::val[cnt] = e[i].val;  
 tot++;  
 }  
 if (tot == n - 1) {  
 break;  
 }  
 }  
}  
} // namespace GR  
  
int ans;  
int A, B, C, P;  
inline int rnd() { return A = (A \* B + C) % P; }  
  
void initLog2() {  
 for (int i = 2; i <= MAX\_VAL\_RANGE; i++) {  
 log2Values[i] = log2Values[i >> 1] + 1;  
 }  
}  
  
int main() {  
 initLog2();  
 cin >> n >> m;  
 for (int i = 1; i <= m; i++) {  
 int u, v, val;  
 cin >> u >> v >> val;  
 GR::e[i] = (GR::Edge){u, v, val};  
 }  
 GR::kruskal();  
 TR::LCA::init();  
 int Q;  
 cin >> Q;  
 cin >> A >> B >> C >> P;  
  
 while (Q--) {  
 int u = rnd() % n + 1, v = rnd() % n + 1;  
 ans += TR::val[TR::LCA::lca(u, v)];  
 ans %= 1000000007;  
 }  
 cout << ans;  
 return 0;  
}

## 斯坦纳树

#include <bits/stdc++.h>  
  
using namespace std;  
  
const int maxn = 510;  
const int INF = 0x3f3f3f3f;  
typedef pair<int, int> P;  
int n, m, k;  
  
struct edge {  
 int to, next, w;  
} e[maxn << 1];  
  
int head[maxn << 1], tree[maxn << 1], tot;  
int dp[maxn][5000], vis[maxn];  
int key[maxn];  
priority\_queue<P, vector<P>, greater<P> > q;  
  
void add(int u, int v, int w) {  
 e[++tot] = edge{v, head[u], w};  
 head[u] = tot;  
}  
  
void dijkstra(int s) {  
 memset(vis, 0, sizeof(vis));  
 while (!q.empty()) {  
 P item = q.top();  
 q.pop();  
 if (vis[item.second]) continue;  
 vis[item.second] = 1;  
 for (int i = head[item.second]; i; i = e[i].next) {  
 if (dp[tree[i]][s] > dp[item.second][s] + e[i].w) {  
 dp[tree[i]][s] = dp[item.second][s] + e[i].w;  
 q.push(P(dp[tree[i]][s], tree[i]));  
 }  
 }  
 }  
}  
  
int main() {  
 memset(dp, INF, sizeof(dp));  
 scanf("%d %d %d", &n, &m, &k);  
 int u, v, w;  
 for (int i = 1; i <= m; i++) {  
 scanf("%d %d %d", &u, &v, &w);  
 add(u, v, w);  
 tree[tot] = v;  
 add(v, u, w);  
 tree[tot] = u;  
 }  
 for (int i = 1; i <= k; i++) {  
 scanf("%d", &key[i]);  
 dp[key[i]][1 << (i - 1)] = 0;  
 }  
 for (int s = 1; s < (1 << k); s++) {  
 for (int i = 1; i <= n; i++) {  
 for (int subs = s & (s - 1); subs; subs = s & (subs - 1))  
 dp[i][s] = min(dp[i][s], dp[i][subs] + dp[i][s ^ subs]);  
 if (dp[i][s] != INF) q.push(P(dp[i][s], i));  
 }  
 dijkstra(s);  
 }  
 printf("%d\n", dp[key[1]][(1 << k) - 1]);  
 return 0;  
}

## 最小树形图

有向图上的最小生成树（Directed Minimum Spanning Tree）称为最小树形图

### 朱刘算法

常用的算法是朱刘算法（也称 Edmonds 算法），可以在O(nm)时间内解决最小树形图问题

bool solve() {  
 ans = 0;  
 int u, v, root = 0;  
 for (;;) {  
 f(i, 0, n) in[i] = 1e100;  
 f(i, 0, m) {  
 u = e[i].s;  
 v = e[i].t;  
 if (u != v && e[i].w < in[v]) {  
 in[v] = e[i].w;  
 pre[v] = u;  
 }  
 }  
 f(i, 0, m) if (i != root && in[i] > 1e50) return 0;  
 int tn = 0;  
 memset(id, -1, sizeof id);  
 memset(vis, -1, sizeof vis);  
 in[root] = 0;  
 f(i, 0, n) {  
 ans += in[i];  
 v = i;  
 while (vis[v] != i && id[v] == -1 && v != root) {  
 vis[v] = i;  
 v = pre[v];  
 }  
 if (v != root && id[v] == -1) {  
 for (int u = pre[v]; u != v; u = pre[u]) id[u] = tn;  
 id[v] = tn++;  
 }  
 }  
 if (tn == 0) break;  
 f(i, 0, n) if (id[i] == -1) id[i] = tn++;  
 f(i, 0, m) {  
 u = e[i].s;  
 v = e[i].t;  
 e[i].s = id[u];  
 e[i].t = id[v];  
 if (e[i].s != e[i].t) e[i].w -= in[v];  
 }  
 n = tn;  
 root = id[root];  
 }  
 return ans;  
}

### Tarjan 的 DMST 算法

Tarjan 提出了一种能够在O(m + nlogn)时间内解决最小树形图问题的算法

#include <bits/stdc++.h>  
  
using namespace std;  
  
typedef long long ll;  
#define maxn 102  
#define INF 0x3f3f3f3f  
  
struct UnionFind {  
 int fa[maxn << 1];  
 UnionFind() { memset(fa, 0, sizeof(fa)); }  
 void clear(int n) { memset(fa + 1, 0, sizeof(int) \* n); }  
 int find(int x) { return fa[x] ? fa[x] = find(fa[x]) : x; }  
 int operator[](int x) { return find(x); }  
};  
struct Edge {  
 int u, v, w, w0;  
};  
struct Heap {  
 Edge \*e;  
 int rk, constant;  
 Heap \*lch, \*rch;  
 Heap(Edge \*\_e) : e(\_e), rk(1), constant(0), lch(NULL), rch(NULL) {}  
 void push() {  
 if (lch) lch->constant += constant;  
 if (rch) rch->constant += constant;  
 e->w += constant;  
 constant = 0;  
 }  
};  
Heap \*merge(Heap \*x, Heap \*y) {  
 if (!x) return y;  
 if (!y) return x;  
 if (x->e->w + x->constant > y->e->w + y->constant) swap(x, y);  
 x->push();  
 x->rch = merge(x->rch, y);  
 if (!x->lch || x->lch->rk < x->rch->rk) swap(x->lch, x->rch);  
 if (x->rch)  
 x->rk = x->rch->rk + 1;  
 else  
 x->rk = 1;  
 return x;  
}  
Edge \*extract(Heap \*&x) {  
 Edge \*r = x->e;  
 x->push();  
 x = merge(x->lch, x->rch);  
 return r;  
}  
  
vector<Edge> in[maxn];  
int n, m, fa[maxn << 1], nxt[maxn << 1];  
Edge \*ed[maxn << 1];  
Heap \*Q[maxn << 1];  
UnionFind id;  
  
void contract() {  
 bool mark[maxn << 1];  
 // 将图上的每一个结点与其相连的那些结点进行记录。  
 for (int i = 1; i <= n; i++) {  
 queue<Heap \*> q;  
 for (int j = 0; j < in[i].size(); j++) q.push(new Heap(&in[i][j]));  
 while (q.size() > 1) {  
 Heap \*u = q.front();  
 q.pop();  
 Heap \*v = q.front();  
 q.pop();  
 q.push(merge(u, v));  
 }  
 Q[i] = q.front();  
 }  
 mark[1] = true;  
 for (int a = 1, b = 1, p; Q[a]; b = a, mark[b] = true) {  
 //寻找最小入边以及其端点，保证无环。  
 do {  
 ed[a] = extract(Q[a]);  
 a = id[ed[a]->u];  
 } while (a == b && Q[a]);  
 if (a == b) break;  
 if (!mark[a]) continue;  
 // 对发现的环进行收缩，以及环内的结点重新编号，总权值更新。  
 for (a = b, n++; a != n; a = p) {  
 id.fa[a] = fa[a] = n;  
 if (Q[a]) Q[a]->constant -= ed[a]->w;  
 Q[n] = merge(Q[n], Q[a]);  
 p = id[ed[a]->u];  
 nxt[p == n ? b : p] = a;  
 }  
 }  
}  
  
ll expand(int x, int r);  
ll expand\_iter(int x) {  
 ll r = 0;  
 for (int u = nxt[x]; u != x; u = nxt[u]) {  
 if (ed[u]->w0 >= INF)  
 return INF;  
 else  
 r += expand(ed[u]->v, u) + ed[u]->w0;  
 }  
 return r;  
}  
ll expand(int x, int t) {  
 ll r = 0;  
 for (; x != t; x = fa[x]) {  
 r += expand\_iter(x);  
 if (r >= INF) return INF;  
 }  
 return r;  
}  
void link(int u, int v, int w) { in[v].push\_back({u, v, w, w}); }  
  
int main() {  
 int rt;  
 scanf("%d %d %d", &n, &m, &rt);  
 for (int i = 0; i < m; i++) {  
 int u, v, w;  
 scanf("%d %d %d", &u, &v, &w);  
 link(u, v, w);  
 }  
 // 保证强连通  
 for (int i = 1; i <= n; i++) link(i > 1 ? i - 1 : n, i, INF);  
 contract();  
 ll ans = expand(rt, n);  
 if (ans >= INF)  
 puts("-1");  
 else  
 printf("%lld\n", ans);  
 return 0;  
}

## 最短路

/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
struct get\_dijkstra  
{  
 static const int maxn = 100005;  
 static const long long inf = 1LL << 60;  
 struct node  
 {  
 int v;  
 LL w;  
 int next;  
 }edge[maxn << 2];  
 int s, tot, head[maxn], n;  
 // LL dist[maxn];  
 bool vis[maxn];  
 void init(int p){  
 memset(head, -1, sizeof(head));  
 n = p, tot = 0;  
 }  
 void add(int u, int v, LL w){  
 edge[tot] = node{v, w, head[u]}, head[u] = tot++;  
 edge[tot] = node{u, w, head[v]}, head[v] = tot++;  
 }  
 void dijk(int st, LL d[]){ // spfa  
 int u, dv;  
 LL dw;  
 for (int i = 0; i <= n; i++) vis[i] = false, d[i] = inf;  
 queue<int> q;  
 vis[st] = true, d[st] = 0;  
 q.push(st);  
 while(q.size()){  
 u = q.front();  
 q.pop();  
 vis[u] = 0;  
 for (int i = head[u]; i != -1; i = edge[i].next){  
 dv = edge[i].v, dw = edge[i].w;  
 if(d[dv] > d[u] + dw){  
 d[dv] = d[u] + dw;  
 if(vis[dv] == 0) q.push(dv), vis[dv] = 1;  
 }  
 }  
 }  
 }  
 void dijk(int st, LL d[]){  
 for (int i = 0; i <= n; i++) vis[i] = false, d[i] = inf;  
 priority\_queue<pair<LL, int> > q;  
 d[st] = 0;  
 q.push(make\_pair(-d[st], st));  
 while(q.size()){  
 int x = q.top().second;  
 q.pop();  
 if(vis[x]) continue;  
 vis[x] = true;  
 for (int i = head[x]; i != -1; i = edge[i].next){  
 int v = edge[i].v;  
 LL dw = edge[i].w;  
 if(d[v] > d[x] + dw){  
 d[v] = d[x] + dw;  
 q.push(make\_pair(-d[v], v));  
 }  
 }  
 }  
 }  
}p;  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 return 0;  
}  
/\*\*/

## 差分约束

### 例题 [luogu P1993 小 K 的农场](https://www.luogu.com.cn/problem/P1993)

跑判断负环，如果不存在负环，输出 Yes ，否则输出 No

#include <algorithm>  
#include <cstdio>  
#include <cstring>  
#include <queue>  
using namespace std;  
struct edge {  
 int v, w, next;  
} e[40005];  
int head[10005], vis[10005], tot[10005], cnt;  
long long ans, dist[10005];  
queue<int> q;  
inline void addedge(int u, int v, int w) {  
 e[++cnt].v = v;  
 e[cnt].w = w;  
 e[cnt].next = head[u];  
 head[u] = cnt;  
}  
int main() {  
 int n, m;  
 scanf("%d%d", &n, &m);  
 for (int i = 1; i <= m; i++) {  
 int op, x, y, z;  
 scanf("%d", &op);  
 if (op == 1) {  
 scanf("%d%d%d", &x, &y, &z);  
 addedge(y, x, z);  
 } else if (op == 2) {  
 scanf("%d%d%d", &x, &y, &z);  
 addedge(x, y, -z);  
 } else {  
 scanf("%d%d", &x, &y);  
 addedge(x, y, 0);  
 addedge(y, x, 0);  
 }  
 }  
 for (int i = 1; i <= n; i++) addedge(0, i, 0);  
 memset(dist, -0x3f, sizeof(dist));  
 dist[0] = 0;  
 vis[0] = 1;  
 q.push(0);  
 while (!q.empty()) {  
 int cur = q.front();  
 q.pop();  
 vis[cur] = 0;  
 for (int i = head[cur]; i; i = e[i].next)  
 if (dist[cur] + e[i].w > dist[e[i].v]) {  
 dist[e[i].v] = dist[cur] + e[i].w;  
 if (!vis[e[i].v]) {  
 vis[e[i].v] = 1;  
 q.push(e[i].v);  
 tot[e[i].v]++;  
 if (tot[e[i].v] >= n) {  
 puts("No");  
 return 0;  
 }  
 }  
 }  
 }  
 puts("Yes");  
 return 0;  
}

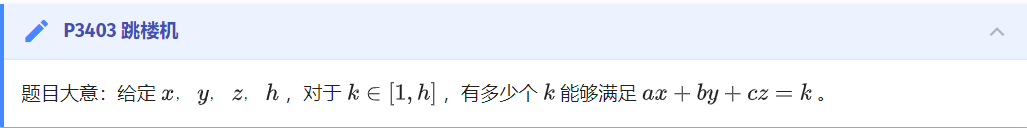
## k 短路

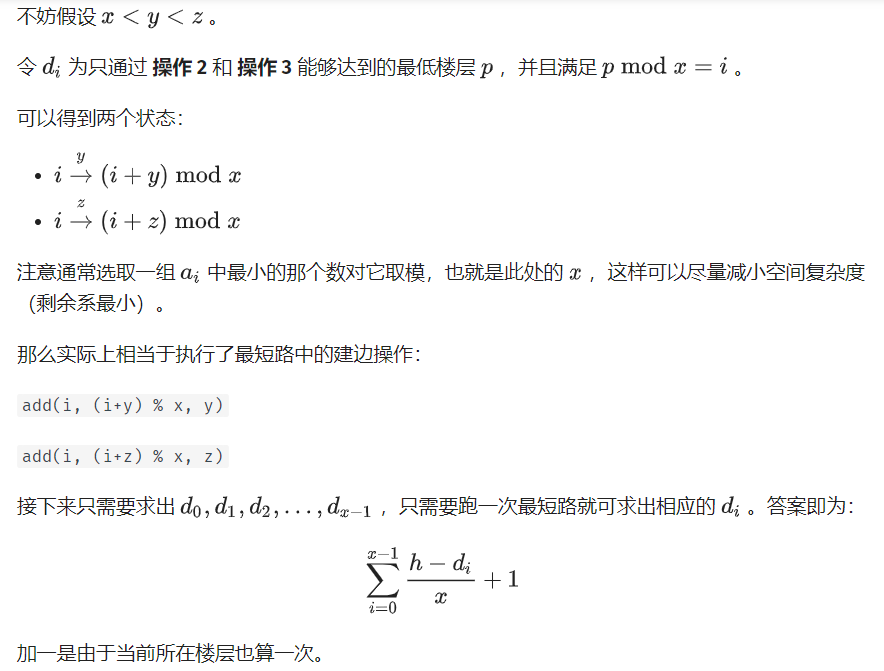
可持久化可并堆优化

#include <algorithm>  
#include <cstdio>  
#include <cstring>  
#include <queue>  
using namespace std;  
const int maxn = 200010;  
int n, m, s, t, k, x, y, ww, cnt, fa[maxn];  
struct Edge {  
 int cur, h[maxn], nxt[maxn], p[maxn], w[maxn];  
 void add\_edge(int x, int y, int z) {  
 cur++;  
 nxt[cur] = h[x];  
 h[x] = cur;  
 p[cur] = y;  
 w[cur] = z;  
 }  
} e1, e2;  
int dist[maxn];  
bool tf[maxn], vis[maxn], ontree[maxn];  
struct node {  
 int x, v;  
 node\* operator=(node a) {  
 x = a.x;  
 v = a.v;  
 return this;  
 }  
 bool operator<(node a) const { return v > a.v; }  
} a;  
priority\_queue<node> Q;  
void dfs(int x) {  
 vis[x] = true;  
 for (int j = e2.h[x]; j; j = e2.nxt[j])  
 if (!vis[e2.p[j]])  
 if (dist[e2.p[j]] == dist[x] + e2.w[j])  
 fa[e2.p[j]] = x, ontree[j] = true, dfs(e2.p[j]);  
}  
struct LeftistTree {  
 int cnt, rt[maxn], lc[maxn \* 20], rc[maxn \* 20], dist[maxn \* 20];  
 node v[maxn \* 20];  
 LeftistTree() { dist[0] = -1; }  
 int newnode(node w) {  
 cnt++;  
 v[cnt] = w;  
 return cnt;  
 }  
 int merge(int x, int y) {  
 if (!x || !y) return x + y;  
 if (v[x] < v[y]) swap(x, y);  
 int p = ++cnt;  
 lc[p] = lc[x];  
 v[p] = v[x];  
 rc[p] = merge(rc[x], y);  
 if (dist[lc[p]] < dist[rc[p]]) swap(lc[p], rc[p]);  
 dist[p] = dist[rc[p]] + 1;  
 return p;  
 }  
} st;  
void dfs2(int x) {  
 vis[x] = true;  
 if (fa[x]) st.rt[x] = st.merge(st.rt[x], st.rt[fa[x]]);  
 for (int j = e2.h[x]; j; j = e2.nxt[j])  
 if (fa[e2.p[j]] == x && !vis[e2.p[j]]) dfs2(e2.p[j]);  
}  
int main() {  
 scanf("%d%d%d%d%d", &n, &m, &s, &t, &k);  
 for (int i = 1; i <= m; i++)  
 scanf("%d%d%d", &x, &y, &ww), e1.add\_edge(x, y, ww), e2.add\_edge(y, x, ww);  
 Q.push({t, 0});  
 while (!Q.empty()) {  
 a = Q.top();  
 Q.pop();  
 if (tf[a.x]) continue;  
 tf[a.x] = true;  
 dist[a.x] = a.v;  
 for (int j = e2.h[a.x]; j; j = e2.nxt[j]) Q.push({e2.p[j], a.v + e2.w[j]});  
 }  
 if (k == 1) {  
 if (tf[s])  
 printf("%d\n", dist[s]);  
 else  
 printf("-1\n");  
 return 0;  
 }  
 dfs(t);  
 for (int i = 1; i <= n; i++)  
 if (tf[i])  
 for (int j = e1.h[i]; j; j = e1.nxt[j])  
 if (!ontree[j])  
 if (tf[e1.p[j]])  
 st.rt[i] = st.merge(  
 st.rt[i],  
 st.newnode({e1.p[j], dist[e1.p[j]] + e1.w[j] - dist[i]}));  
 for (int i = 1; i <= n; i++) vis[i] = false;  
 dfs2(t);  
 if (st.rt[s]) Q.push({st.rt[s], dist[s] + st.v[st.rt[s]].v});  
 while (!Q.empty()) {  
 a = Q.top();  
 Q.pop();  
 cnt++;  
 if (cnt == k - 1) {  
 printf("%d\n", a.v);  
 return 0;  
 }  
 if (st.lc[a.x])  
 Q.push({st.lc[a.x], a.v - st.v[a.x].v + st.v[st.lc[a.x]].v});  
 if (st.rc[a.x])  
 Q.push({st.rc[a.x], a.v - st.v[a.x].v + st.v[st.rc[a.x]].v});  
 x = st.rt[st.v[a.x].x];  
 if (x) Q.push({x, a.v + st.v[x].v});  
 }  
 printf("-1\n");  
 return 0;  
}

## 同余最短路

当出现形如“给定n个整数，求这n个整数能拼凑出多少的其他整数（n个整数可以重复取）”，以及“给定n个整数，求这n个整数不能拼凑出的最小（最大）的整数”的问题时可以使用同余最短路的方法





#include <bits/stdc++.h>  
  
using namespace std;  
typedef long long ll;  
const int maxn = 100010;  
const int INF = 0x3f3f3f3f;  
  
ll h, x, y, z;  
ll head[maxn << 1], tot;  
ll dis[maxn], vis[maxn];  
queue<int> q;  
  
struct edge {  
 ll to, next, w;  
} e[maxn << 1];  
  
void add(ll u, ll v, ll w) {  
 e[++tot] = edge{v, head[u], w};  
 head[u] = tot;  
}  
  
void spfa() {  
 dis[1] = 1;  
 vis[1] = 1;  
 q.push(1);  
 while (!q.empty()) {  
 int u = q.front();  
 q.pop();  
 vis[u] = 0;  
 for (int i = head[u]; i; i = e[i].next) {  
 int v = e[i].to, w = e[i].w;  
 if (dis[v] > dis[u] + w) {  
 dis[v] = dis[u] + w;  
 if (!vis[v]) {  
 q.push(v);  
 vis[v] = 1;  
 }  
 }  
 }  
 }  
}  
  
int main() {  
 memset(dis, INF, sizeof(dis));  
 scanf("%lld", &h);  
 scanf("%lld %lld %lld", &x, &y, &z);  
 if (x == 1 || y == 1 || z == 1) {  
 printf("%d\n", h);  
 return 0;  
 }  
 for (int i = 0; i < x; i++) {  
 add(i, (i + z) % x, z);  
 add(i, (i + y) % x, y);  
 }  
 spfa();  
 ll ans = 0;  
 for (int i = 0; i < x; i++) {  
 if (h >= dis[i]) ans += (h - dis[i]) / x + 1;  
 }  
 printf("%lld\n", ans);  
 return 0;  
}

## 连通性相关

### 强连通分量

int dfn[N], low[N], dfncnt, s[N], in\_stack[N], tp;  
int scc[N], sc; // 结点 i 所在 scc 的编号  
int sz[N]; // 强连通 i 的大小  
void tarjan(int u) {  
 low[u] = dfn[u] = ++dfncnt, s[++tp] = u, in\_stack[u] = 1;  
 for (int i = h[u]; i; i = e[i].nex) {  
 const int &v = e[i].t;  
 if (!dfn[v]) {  
 tarjan(v);  
 low[u] = min(low[u], low[v]);  
 } else if (in\_stack[v]) {  
 low[u] = min(low[u], dfn[v]);  
 }  
 }  
 if (dfn[u] == low[u]) {  
 ++sc;  
 while (s[tp] != u) {  
 scc[s[tp]] = sc;  
 sz[sc]++;  
 in\_stack[s[tp]] = 0;  
 --tp;  
 }  
 scc[s[tp]] = sc;  
 sz[sc]++;  
 in\_stack[s[tp]] = 0;  
 --tp;  
 }  
}

### 双连通分量

https://oi-wiki.org/graph/bcc/

### 割点

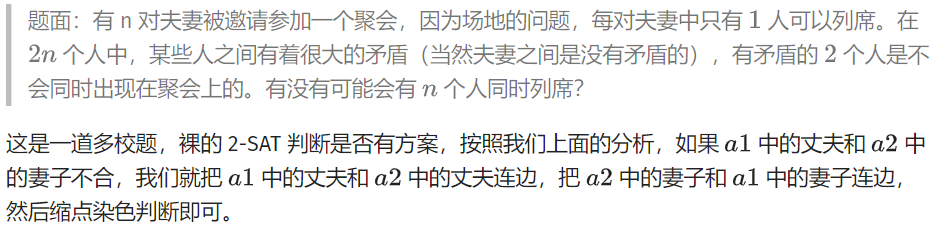
/\*  
洛谷 P3388 【模板】割点（割顶）  
\*/  
#include <bits/stdc++.h>  
using namespace std;  
int n, m; // n：点数 m：边数  
int num[100001], low[100001], inde, res;  
// num：记录每个点的时间戳  
// low：能不经过父亲到达最小的编号，inde：时间戳，res：答案数量  
bool vis[100001], flag[100001]; // flag: 答案 vis：标记是否重复  
vector<int> edge[100001]; // 存图用的  
void Tarjan(int u, int father) { // u 当前点的编号，father 自己爸爸的编号  
 vis[u] = true; // 标记  
 low[u] = num[u] = ++inde; // 打上时间戳  
 int child = 0; // 每一个点儿子数量  
 for (auto v : edge[u]) { // 访问这个点的所有邻居 （C++11）  
  
 if (!vis[v]) {  
 child++; // 多了一个儿子  
 Tarjan(v, u); // 继续  
 low[u] = min(low[u], low[v]); // 更新能到的最小节点编号  
 if (father != u && low[v] >= num[u] &&  
 !flag  
 [u]) // 主要代码  
 // 如果不是自己，且不通过父亲返回的最小点符合割点的要求，并且没有被标记过  
 // 要求即为：删了父亲连不上去了，即为最多连到父亲  
 {  
 flag[u] = true;  
 res++; // 记录答案  
 }  
 } else if (v != father)  
 low[u] =  
 min(low[u], num[v]); // 如果这个点不是自己，更新能到的最小节点编号  
 }  
 if (father == u && child >= 2 &&  
 !flag[u]) { // 主要代码，自己的话需要 2 个儿子才可以  
 flag[u] = true;  
 res++; // 记录答案  
 }  
}  
int main() {  
 cin >> n >> m; // 读入数据  
 for (int i = 1; i <= m; i++) { // 注意点是从 1 开始的  
 int x, y;  
 cin >> x >> y;  
 edge[x].push\_back(y);  
 edge[y].push\_back(x);  
 } // 使用 vector 存图  
 for (int i = 1; i <= n; i++) // 因为 Tarjan 图不一定连通  
 if (!vis[i]) {  
 inde = 0; // 时间戳初始为 0  
 Tarjan(i, i); // 从第 i 个点开始，父亲为自己  
 }  
 cout << res << endl;  
 for (int i = 1; i <= n; i++)  
 if (flag[i]) cout << i << " "; // 输出结果  
 return 0;  
}

### 割边

int low[MAXN], dfn[MAXN], iscut[MAXN], dfs\_clock;  
bool isbridge[MAXN];  
vector<int> G[MAXN];  
int cnt\_bridge;  
int father[MAXN];  
  
void tarjan(int u, int fa) {  
 father[u] = fa;  
 low[u] = dfn[u] = ++dfs\_clock;  
 for (int i = 0; i < G[u].size(); i++) {  
 int v = G[u][i];  
 if (!dfn[v]) {  
 tarjan(v, u);  
 low[u] = min(low[u], low[v]);  
 if (low[v] > dfn[u]) {  
 isbridge[v] = true;  
 ++cnt\_bridge;  
 }  
 } else if (dfn[v] < dfn[u] && v != fa) {  
 low[u] = min(low[u], dfn[v]);  
 }  
 }  
}

## 2-SAT

**HDU3062 [Party](http://acm.hdu.edu.cn/showproblem.php?pid=3062)**



// 作者：小黑 AWM  
#include <algorithm>  
#include <cstdio>  
#include <cstring>  
#include <iostream>  
#define maxn 2018  
#define maxm 4000400  
using namespace std;  
int Index, instack[maxn], DFN[maxn], LOW[maxn];  
int tot, color[maxn];  
int numedge, head[maxn];  
struct Edge {  
 int nxt, to;  
} edge[maxm];  
int sta[maxn], top;  
int n, m;  
void add(int x, int y) {  
 edge[++numedge].to = y;  
 edge[numedge].nxt = head[x];  
 head[x] = numedge;  
}  
void tarjan(int x) { // 缩点看不懂请移步强连通分量上面有一个链接可以点。  
 sta[++top] = x;  
 instack[x] = 1;  
 DFN[x] = LOW[x] = ++Index;  
 for (int i = head[x]; i; i = edge[i].nxt) {  
 int v = edge[i].to;  
 if (!DFN[v]) {  
 tarjan(v);  
 LOW[x] = min(LOW[x], LOW[v]);  
 } else if (instack[v])  
 LOW[x] = min(LOW[x], DFN[v]);  
 }  
 if (DFN[x] == LOW[x]) {  
 tot++;  
 do {  
 color[sta[top]] = tot; // 染色  
 instack[sta[top]] = 0;  
 } while (sta[top--] != x);  
 }  
}  
bool solve() {  
 for (int i = 0; i < 2 \* n; i++)  
 if (!DFN[i]) tarjan(i);  
 for (int i = 0; i < 2 \* n; i += 2)  
 if (color[i] == color[i + 1]) return 0;  
 return 1;  
}  
void init() {  
 top = 0;  
 tot = 0;  
 Index = 0;  
 numedge = 0;  
 memset(sta, 0, sizeof(sta));  
 memset(DFN, 0, sizeof(DFN));  
 memset(instack, 0, sizeof(instack));  
 memset(LOW, 0, sizeof(LOW));  
 memset(color, 0, sizeof(color));  
 memset(head, 0, sizeof(head));  
}  
int main() {  
 while (~scanf("%d%d", &n, &m)) {  
 init();  
 for (int i = 1; i <= m; i++) {  
 int a1, a2, c1, c2;  
 scanf("%d%d%d%d", &a1, &a2, &c1, &c2); // 自己做的时候别用 cin 会被卡  
 add(2 \* a1 + c1, 2 \* a2 + 1 - c2);  
 // 对于第 i 对夫妇，我们用 2i+1 表示丈夫，2i 表示妻子。  
 add(2 \* a2 + c2, 2 \* a1 + 1 - c1);  
 }  
 if (solve())  
 printf("YES\n");  
 else  
 printf("NO\n");  
 }  
 return 0;  
}

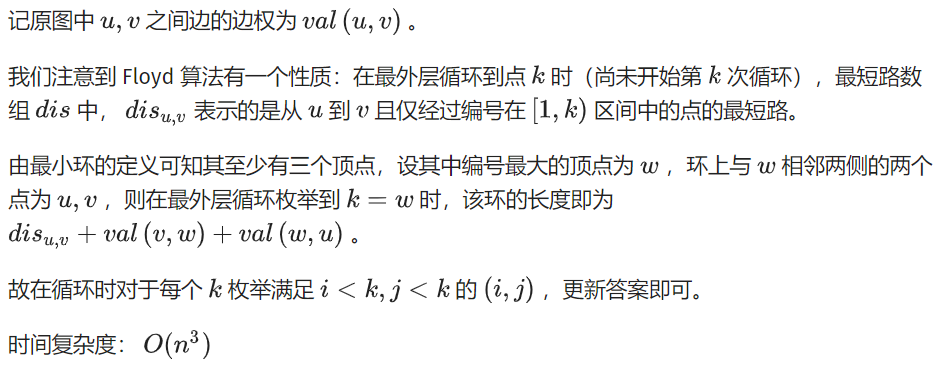
## 欧拉图

Hierholzer算法

/\*  
欧拉路径 Hierholzer算法  
\*/  
#pragma GCC optimize(3,"Ofast","inline")  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
int n, m, cnt, tot, in[10005], f[10005], head[10005], ans[1000005];  
bool vis[1000005];  
  
struct node  
{  
 int v, id, next;  
}a[1000005 << 1];  
  
void read(int &x){  
 char ch = getchar();x = 0;  
 for (; ch < '0' || ch > '9'; ch = getchar());  
 for (; ch >='0' && ch <= '9'; ch = getchar()) x = x \* 10 + ch - '0';  
}  
  
int Find(int x){  
 return x == f[x] ? x : f[x] = Find(f[x]);  
}  
  
void dfs(int x){  
 for (int i = head[x]; i != -1; i = a[i].next){  
 int v = a[i].v;  
 if(vis[a[i].id]) continue;  
 vis[a[i].id] = true;  
 head[x] = i;  
 dfs(v);  
 ans[cnt++] = a[i].id;  
 i = head[x];  
 }  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 memset(head, -1, sizeof(head));  
 read(n), read(m);  
 for (int i = 1; i <= n + 1; i++) f[i] = i;  
 for (int i = 1, u, v; i <= m; i++){  
 read(u), read(v);  
 in[u]++, in[v]++;  
 a[tot] = node{v, i, head[u]}, head[u] = tot++;  
 a[tot] = node{u, i, head[v]}, head[v] = tot++;  
 u = Find(u), v = Find(v);  
 if(u != v) f[u] = v;  
 }  
 int fg = 0, num = 0;  
 for (int i = 1; i <= n; i++) if(Find(i) == i) num++;  
 for (int i = 1; i <= n; i++) if(!in[i]) num--;  
 for (int i = 1; i <= n; i++){  
 if(in[i] & 1) {fg = 1; break;}  
 }  
 if(fg || num != 1) printf("No\n");  
 else{  
 printf("Yes\n");  
 dfs(1);  
 for (int i = 0; i < cnt; i++) printf("%d ", ans[i]);  
 printf("\n");  
 }  
  
 return 0;  
}  
/\*  
4 8  
1 2  
2 3  
3 4  
2 3  
2 3  
4 1  
3 4  
3 4  
\*/

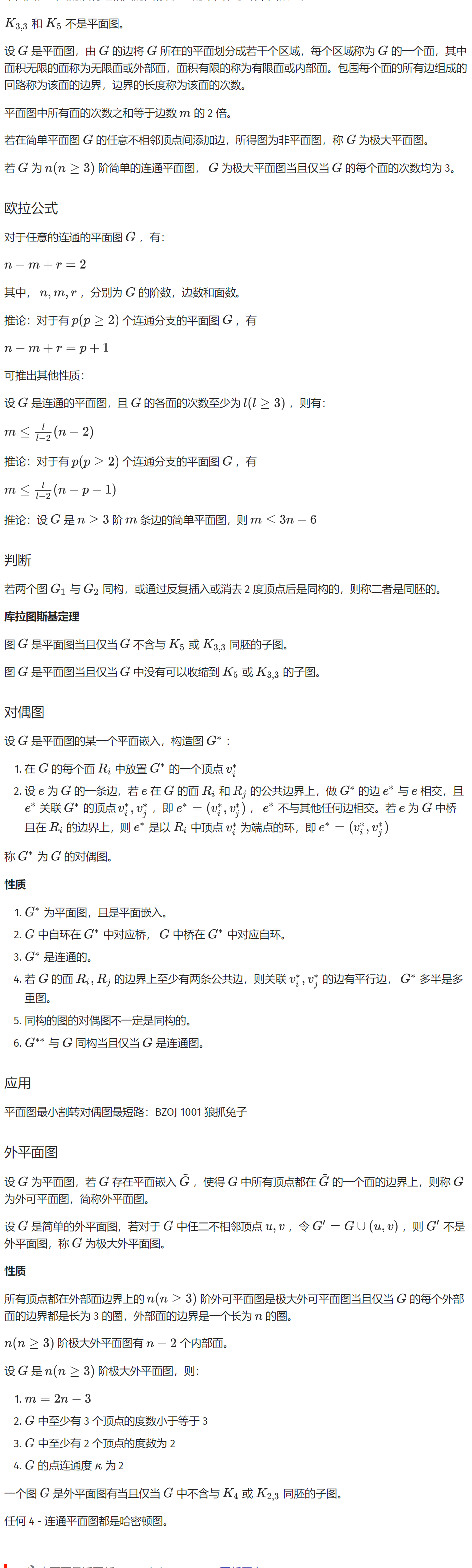
## 最小环

### Floyd



int val[maxn + 1][maxn + 1]; // 原图的邻接矩阵  
inline int floyd(const int &n) {  
 static int dis[maxn + 1][maxn + 1]; // 最短路矩阵  
 for (int i = 1; i <= n; ++i)  
 for (int j = 1; j <= n; ++j) dis[i][j] = val[i][j]; // 初始化最短路矩阵  
 int ans = inf;  
 for (int k = 1; k <= n; ++k) {  
 for (int i = 1; i < k; ++i)  
 for (int j = 1; j < i; ++j)  
 ans = std::min(ans, dis[i][j] + val[i][k] + val[k][j]); // 更新答案  
 for (int i = 1; i <= n; ++i)  
 for (int j = 1; j <= n; ++j)  
 dis[i][j] = std::min(  
 dis[i][j], dis[i][k] + dis[k][j]); // 正常的 floyd 更新最短路矩阵  
 }  
 return ans;  
}

## 平面图



## 网络流

### 最大流

#### EK 算法

时间复杂度O(nm^2)

#define maxn 250  
#define INF 0x3f3f3f3f  
  
struct Edge {  
 int from, to, cap, flow;  
 Edge(int u, int v, int c, int f) : from(u), to(v), cap(c), flow(f) {}  
};  
  
struct EK {  
 int n, m;  
 vector<Edge> edges;  
 vector<int> G[maxn];  
 int a[maxn], p[maxn];  
  
 void init(int n) {  
 for (int i = 0; i < n; i++) G[i].clear();  
 edges.clear();  
 }  
  
 void AddEdge(int from, int to, int cap) {  
 edges.push\_back(Edge(from, to, cap, 0));  
 edges.push\_back(Edge(to, from, 0, 0));  
 m = edges.size();  
 G[from].push\_back(m - 2);  
 G[to].push\_back(m - 1);  
 }  
  
 int Maxflow(int s, int t) {  
 int flow = 0;  
 for (;;) {  
 memset(a, 0, sizeof(a));  
 queue<int> Q;  
 Q.push(s);  
 a[s] = INF;  
 while (!Q.empty()) {  
 int x = Q.front();  
 Q.pop();  
 for (int i = 0; i < G[x].size(); i++) {  
 Edge& e = edges[G[x][i]];  
 if (!a[e.to] && e.cap > e.flow) {  
 p[e.to] = G[x][i];  
 a[e.to] = min(a[x], e.cap - e.flow);  
 Q.push(e.to);  
 }  
 }  
 if (a[t]) break;  
 }  
 if (!a[t]) break;  
 for (int u = t; u != s; u = edges[p[u]].from) {  
 edges[p[u]].flow += a[t];  
 edges[p[u] ^ 1].flow -= a[t];  
 }  
 flow += a[t];  
 }  
 return flow;  
 }  
};

#### Dinic 算法

时间复杂度O(msqrt(n))

#define maxn 250  
#define INF 0x3f3f3f3f  
  
struct Edge {  
 int from, to, cap, flow;  
 Edge(int u, int v, int c, int f) : from(u), to(v), cap(c), flow(f) {}  
};  
  
struct Dinic {  
 int n, m, s, t;  
 vector<Edge> edges;  
 vector<int> G[maxn];  
 int d[maxn], cur[maxn];  
 bool vis[maxn];  
  
 void init(int n) {  
 for (int i = 0; i < n; i++) G[i].clear();  
 edges.clear();  
 }  
  
 void AddEdge(int from, int to, int cap) {  
 edges.push\_back(Edge(from, to, cap, 0));  
 edges.push\_back(Edge(to, from, 0, 0));  
 m = edges.size();  
 G[from].push\_back(m - 2);  
 G[to].push\_back(m - 1);  
 }  
  
 bool BFS() {  
 memset(vis, 0, sizeof(vis));  
 queue<int> Q;  
 Q.push(s);  
 d[s] = 0;  
 vis[s] = 1;  
 while (!Q.empty()) {  
 int x = Q.front();  
 Q.pop();  
 for (int i = 0; i < G[x].size(); i++) {  
 Edge& e = edges[G[x][i]];  
 if (!vis[e.to] && e.cap > e.flow) {  
 vis[e.to] = 1;  
 d[e.to] = d[x] + 1;  
 Q.push(e.to);  
 }  
 }  
 }  
 return vis[t];  
 }  
  
 int DFS(int x, int a) {  
 if (x == t || a == 0) return a;  
 int flow = 0, f;  
 for (int& i = cur[x]; i < G[x].size(); i++) {  
 Edge& e = edges[G[x][i]];  
 if (d[x] + 1 == d[e.to] && (f = DFS(e.to, min(a, e.cap - e.flow))) > 0) {  
 e.flow += f;  
 edges[G[x][i] ^ 1].flow -= f;  
 flow += f;  
 a -= f;  
 if (a == 0) break;  
 }  
 }  
 return flow;  
 }  
  
 int Maxflow(int s, int t) {  
 this->s = s;  
 this->t = t;  
 int flow = 0;  
 while (BFS()) {  
 memset(cur, 0, sizeof(cur));  
 flow += DFS(s, INF);  
 }  
 return flow;  
 }  
};

#### ISAP

struct Edge {  
 int from, to, cap, flow;  
 Edge(int u, int v, int c, int f) : from(u), to(v), cap(c), flow(f) {}  
};  
  
bool operator<(const Edge& a, const Edge& b) {  
 return a.from < b.from || (a.from == b.from && a.to < b.to);  
}  
  
struct ISAP {  
 int n, m, s, t;  
 vector<Edge> edges;  
 vector<int> G[maxn];  
 bool vis[maxn];  
 int d[maxn];  
 int cur[maxn];  
 int p[maxn];  
 int num[maxn];  
  
 void AddEdge(int from, int to, int cap) {  
 edges.push\_back(Edge(from, to, cap, 0));  
 edges.push\_back(Edge(to, from, 0, 0));  
 m = edges.size();  
 G[from].push\_back(m - 2);  
 G[to].push\_back(m - 1);  
 }  
  
 bool BFS() {  
 memset(vis, 0, sizeof(vis));  
 queue<int> Q;  
 Q.push(t);  
 vis[t] = 1;  
 d[t] = 0;  
 while (!Q.empty()) {  
 int x = Q.front();  
 Q.pop();  
 for (int i = 0; i < G[x].size(); i++) {  
 Edge& e = edges[G[x][i] ^ 1];  
 if (!vis[e.from] && e.cap > e.flow) {  
 vis[e.from] = 1;  
 d[e.from] = d[x] + 1;  
 Q.push(e.from);  
 }  
 }  
 }  
 return vis[s];  
 }  
  
 void init(int n) {  
 this->n = n;  
 for (int i = 0; i < n; i++) G[i].clear();  
 edges.clear();  
 }  
  
 int Augment() {  
 int x = t, a = INF;  
 while (x != s) {  
 Edge& e = edges[p[x]];  
 a = min(a, e.cap - e.flow);  
 x = edges[p[x]].from;  
 }  
 x = t;  
 while (x != s) {  
 edges[p[x]].flow += a;  
 edges[p[x] ^ 1].flow -= a;  
 x = edges[p[x]].from;  
 }  
 return a;  
 }  
  
 int Maxflow(int s, int t) {  
 this->s = s;  
 this->t = t;  
 int flow = 0;  
 BFS();  
 memset(num, 0, sizeof(num));  
 for (int i = 0; i < n; i++) num[d[i]]++;  
 int x = s;  
 memset(cur, 0, sizeof(cur));  
 while (d[s] < n) {  
 if (x == t) {  
 flow += Augment();  
 x = s;  
 }  
 int ok = 0;  
 for (int i = cur[x]; i < G[x].size(); i++) {  
 Edge& e = edges[G[x][i]];  
 if (e.cap > e.flow && d[x] == d[e.to] + 1) {  
 ok = 1;  
 p[e.to] = G[x][i];  
 cur[x] = i;  
 x = e.to;  
 break;  
 }  
 }  
 if (!ok) {  
 int m = n - 1;  
 for (int i = 0; i < G[x].size(); i++) {  
 Edge& e = edges[G[x][i]];  
 if (e.cap > e.flow) m = min(m, d[e.to]);  
 }  
 if (--num[d[x]] == 0) break;  
 num[d[x] = m + 1]++;  
 cur[x] = 0;  
 if (x != s) x = edges[p[x]].from;  
 }  
 }  
 return flow;  
 }  
};

### 最小割

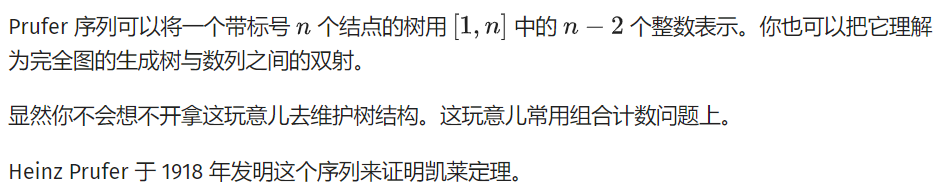
**最大流最小割定理**

#include <algorithm>  
#include <cstdio>  
#include <cstring>  
#include <queue>  
  
const int N = 1e4 + 5, M = 2e5 + 5;  
int n, m, s, t, tot = 1, lnk[N], ter[M], nxt[M], val[M], dep[N], cur[N];  
  
void add(int u, int v, int w) {  
 ter[++tot] = v, nxt[tot] = lnk[u], lnk[u] = tot, val[tot] = w;  
}  
void addedge(int u, int v, int w) { add(u, v, w), add(v, u, 0); }  
int bfs(int s, int t) {  
 memset(dep, 0, sizeof(dep));  
 memcpy(cur, lnk, sizeof(lnk));  
 std::queue<int> q;  
 q.push(s), dep[s] = 1;  
 while (!q.empty()) {  
 int u = q.front();  
 q.pop();  
 for (int i = lnk[u]; i; i = nxt[i]) {  
 int v = ter[i];  
 if (val[i] && !dep[v]) q.push(v), dep[v] = dep[u] + 1;  
 }  
 }  
 return dep[t];  
}  
int dfs(int u, int t, int flow) {  
 if (u == t) return flow;  
 int ans = 0;  
 for (int &i = cur[u]; i && ans < flow; i = nxt[i]) {  
 int v = ter[i];  
 if (val[i] && dep[v] == dep[u] + 1) {  
 int x = dfs(v, t, std::min(val[i], flow - ans));  
 if (x) val[i] -= x, val[i ^ 1] += x, ans += x;  
 }  
 }  
 if (ans < flow) dep[u] = -1;  
 return ans;  
}  
int dinic(int s, int t) {  
 int ans = 0;  
 while (bfs(s, t)) {  
 int x;  
 while ((x = dfs(s, t, 1 << 30))) ans += x;  
 }  
 return ans;  
}  
int main() {  
 scanf("%d%d%d%d", &n, &m, &s, &t);  
 while (m--) {  
 int u, v, w;  
 scanf("%d%d%d", &u, &v, &w);  
 addedge(u, v, w);  
 }  
 printf("%d\n", dinic(s, t));  
 return 0;  
}

### 费用流

/\*  
模板更新第二个  
\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
  
typedef long long LL;  
  
using namespace std;  
  
typedef pair<long long, int> P;  
typedef pair<int, int> pii;  
  
int T, n, k, s0, s1, t, a[2005];  
  
void read(int &x){  
 char ch = getchar();x = 0;  
 for (; ch < '0' || ch > '9'; ch = getchar());  
 for (; ch >='0' && ch <= '9'; ch = getchar()) x = x \* 10 + ch - '0';  
}  
  
  
  
struct MCMFvector{  
 static const int maxn = 4005;  
 static const long long inf = 1LL << 60;  
 struct node{int v, rev, c, w;};  
 vector<node> edge[maxn];  
 int n, pre[maxn];  
 LL dist[maxn], h[maxn];  
  
 void init(int point){  
 n = point;  
 for(int i = 0; i <= n; i++) {  
 h[i] = 0;  
 edge[i].clear();  
 }  
 }  
 void add(int u, int v, int c, int w){  
 edge[u].push\_back(node{v, int(edge[v].size()), c, w});  
 edge[v].push\_back(node{u, int(edge[u].size())-1, 0, -w});  
 }  
 void minCostMaxFlow(int s, int t, int &flow, LL &cost){  
 flow = cost = 0;  
 while(true){  
 for(int i = 0; i <= n; i++) dist[i] = inf;  
 dist[s] = 0;  
 priority\_queue<pii, vector<pii>, greater<pii> > q;  
 q.push(pii(dist[s], s));  
 while(!q.empty()){  
 int u = q.top().second, dis = q.top().first;  
 q.pop();  
 if (dist[u] < dis) continue;  
 for(int i = 0; i < (int)edge[u].size(); i++){  
 int v = edge[u][i].v, c = edge[u][i].c, w = edge[u][i].w;  
 if(c > 0 && dist[v] > dist[u] + w + h[u] - h[v]){  
 dist[v] = dist[u] + w + h[u] - h[v];  
 pre[v] = edge[u][i].rev;  
 q.push(pii(dist[v], v));  
 }  
 }  
 }  
 if(dist[t] == inf) break;  
 for(int i = 0; i <= n; i++) h[i] += dist[i];  
 int addf = 0x3f3f3f3f;  
 for(int x = t; x != s; x = edge[x][pre[x]].v) addf = min(addf, edge[edge[x][pre[x]].v][edge[x][pre[x]].rev].c);  
 for(int x = t; x != s; x = edge[x][pre[x]].v){  
 edge[x][pre[x]].c += addf;  
 edge[edge[x][pre[x]].v][edge[x][pre[x]].rev].c -= addf;  
 }  
 flow += addf;  
 cost += h[t] \* addf;  
 }  
 }  
  
 void maxCostMaxFlow(int s, int t, int &flow, LL &cost){  
 flow = cost = 0;  
 while(true){  
 for(int i = 0; i <= n; i++) dist[i] = -inf;  
 dist[s] = 0;  
 priority\_queue<pii> q;  
 q.push(pii(dist[s], s));  
 while(!q.empty()){  
 int u = q.top().second, dis = q.top().first;  
 q.pop();  
 if (dist[u] > dis) continue;  
 for(int i = 0; i < (int)edge[u].size(); i++){  
 int v = edge[u][i].v, c = edge[u][i].c, w = edge[u][i].w;  
 if(c > 0 && dist[v] < dist[u] + w + h[u] - h[v]){  
 dist[v] = dist[u] + w + h[u] - h[v];  
 pre[v] = edge[u][i].rev;  
 q.push(pii(dist[v], v));  
 }  
 }  
 }  
 if(dist[t] == -inf) break;  
 for(int i = 0; i <= n; i++) h[i] += dist[i];  
 int addf = 0x3f3f3f3f;  
 for(int x = t; x != s; x = edge[x][pre[x]].v) addf = min(addf, edge[edge[x][pre[x]].v][edge[x][pre[x]].rev].c);  
 for(int x = t; x != s; x = edge[x][pre[x]].v){  
 edge[x][pre[x]].c += addf;  
 edge[edge[x][pre[x]].v][edge[x][pre[x]].rev].c -= addf;  
 }  
 flow += addf;  
 cost += h[t] \* addf;  
 }  
 }  
}g;  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 read(T);  
 while(T--){  
 read(n), read(k);  
 for (int i = 1; i <= n; i++) read(a[i]);  
 s0 = 2 \* n + 1, s1 = 2 \* n + 2, t = 2 \* n + 3;  
 g.init(2 \* n + 3);  
 g.add(s0, s1, k, 0);  
 for (int i = 1; i <= n; i++) g.add(s1, i, 1, 0), g.add(i, n + i, 1, -a[i]), g.add(n + i, t, 1, 0);  
 for (int i = 1; i <= n; i++){  
 for (int j = i + 1; j <= n; j++){  
 if(a[i] <= a[j]) g.add(n + i, j, 1, 0);  
 }  
 }  
 int flow;  
 LL cost;  
 g.minCostMaxFlow(s0, t, flow, cost);  
 printf("%lld\n", -cost);  
 }  
  
 return 0;  
}  
/\*\*/  
  
struct MCMF {  
 int n, m;  
 std::vector<Edge> edges;  
 std::vector<int> G[maxn];  
 int inq[maxn]; //是否在队列中  
 i64 d[maxn]; // bellmanford  
 int p[maxn]; //上一条弧  
 i64 a[maxn]; //可改进量  
 void init(int n) {  
 this->n = n;  
 for (int i = 0; i < n; i++) G[i].clear();  
 edges.clear();  
 }  
 void AddEdge(int from, int to, int cap, i64 cost) {  
 edges.emplace\_back(from, to, cap, 0, cost);  
 edges.emplace\_back(to, from, 0, 0, -cost);  
 m = edges.size();  
 G[from].push\_back(m - 2);  
 G[to].push\_back(m - 1);  
 }  
 bool BellmanFord(int s, int t, int& flow, i64& cost) {  
 for (int i = 0; i < n; i++) d[i] = INF;  
 memset(inq, 0, sizeof(inq));  
 d[s] = 0;  
 inq[s] = 1;  
 p[s] = 0;  
 a[s] = INF;  
 std::queue<int> q;  
 q.push(s);  
 while (!q.empty()) {  
 int u = q.front();  
 q.pop();  
 inq[u] = 0;  
 for (int i = 0; i < int(G[u].size()); i++) {  
 Edge& e = edges[G[u][i]];  
 if (e.cap > e.flow && d[e.to] > d[u] + e.cost) {  
 d[e.to] = d[u] + e.cost;  
 p[e.to] = G[u][i];  
 a[e.to] = std::min<i64>(a[u], e.cap - e.flow);  
 if (!inq[e.to]) {  
 q.push(e.to);  
 inq[e.to] = 1;  
 }  
 }  
 }  
 }  
 if (d[t] == INF) return false; // 当没有可增广的路时退出  
 flow += a[t];  
 cost += (i64)d[t] \* (i64)a[t];  
 for (int u = t; u != s; u = edges[p[u]].from) {  
 edges[p[u]].flow += a[t];  
 edges[p[u] ^ 1].flow -= a[t];  
 }  
 return true;  
 }  
 int MincostMaxflow(int s, int t, i64& cost) {  
 int flow = 0;  
 cost = 0;  
 while (BellmanFord(s, t, flow, cost))  
 ;  
 return flow;  
 }  
};

## Prufer 序列

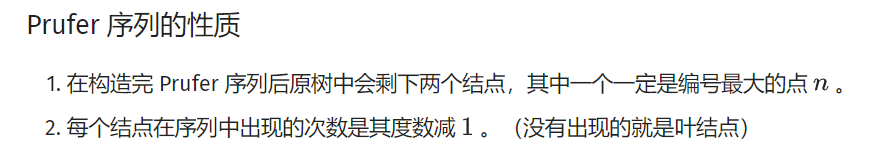


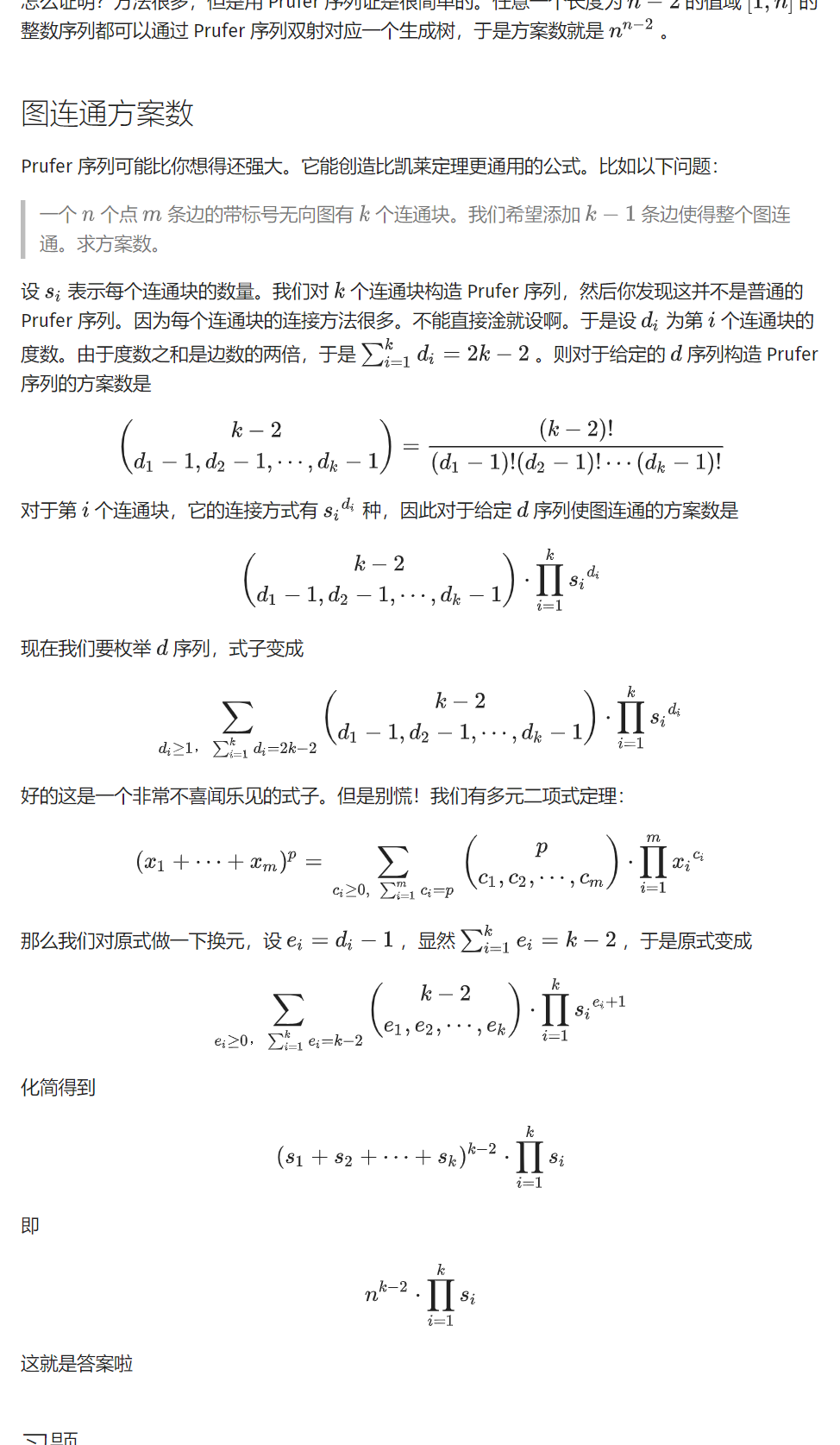
堆O(nlogn)

// 代码摘自原文，结点是从 0 标号的  
vector<vector<int>> adj;  
  
vector<int> pruefer\_code() {  
 int n = adj.size();  
 set<int> leafs;  
 vector<int> degree(n);  
 vector<bool> killed(n, false);  
 for (int i = 0; i < n; i++) {  
 degree[i] = adj[i].size();  
 if (degree[i] == 1) leafs.insert(i);  
 }  
  
 vector<int> code(n - 2);  
 for (int i = 0; i < n - 2; i++) {  
 int leaf = \*leafs.begin();  
 leafs.erase(leafs.begin());  
 killed[leaf] = true;  
 int v;  
 for (int u : adj[leaf])  
 if (!killed[u]) v = u;  
 code[i] = v;  
 if (--degree[v] == 1) leafs.insert(v);  
 }  
 return code;  
}

线性构造

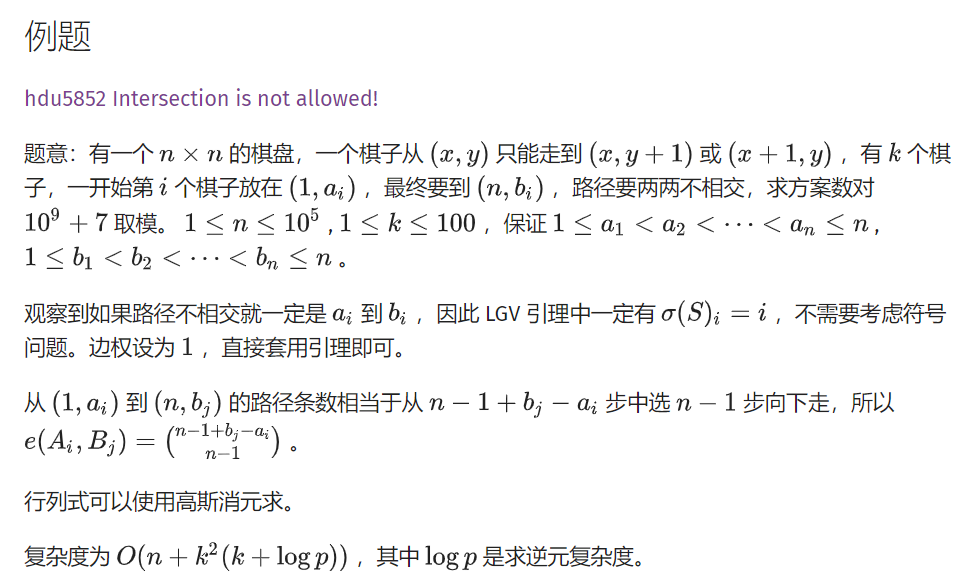
// 从原文摘的代码，同样以 0 为起点  
vector<vector<int>> adj;  
vector<int> parent;  
  
void dfs(int v) {  
 for (int u : adj[v]) {  
 if (u != parent[v]) parent[u] = v, dfs(u);  
 }  
}  
  
vector<int> pruefer\_code() {  
 int n = adj.size();  
 parent.resize(n), parent[n - 1] = -1;  
 dfs(n - 1);  
  
 int ptr = -1;  
 vector<int> degree(n);  
 for (int i = 0; i < n; i++) {  
 degree[i] = adj[i].size();  
 if (degree[i] == 1 && ptr == -1) ptr = i;  
 }  
  
 vector<int> code(n - 2);  
 int leaf = ptr;  
 for (int i = 0; i < n - 2; i++) {  
 int next = parent[leaf];  
 code[i] = next;  
 if (--degree[next] == 1 && next < ptr) {  
 leaf = next;  
 } else {  
 ptr++;  
 while (degree[ptr] != 1) ptr++;  
 leaf = ptr;  
 }  
 }  
 return code;  
}





## LGV 引理

Lindström–Gessel–Viennot lemma，即 LGV 引理，可以用来处理有向无环图上不相交路径计数等问题。



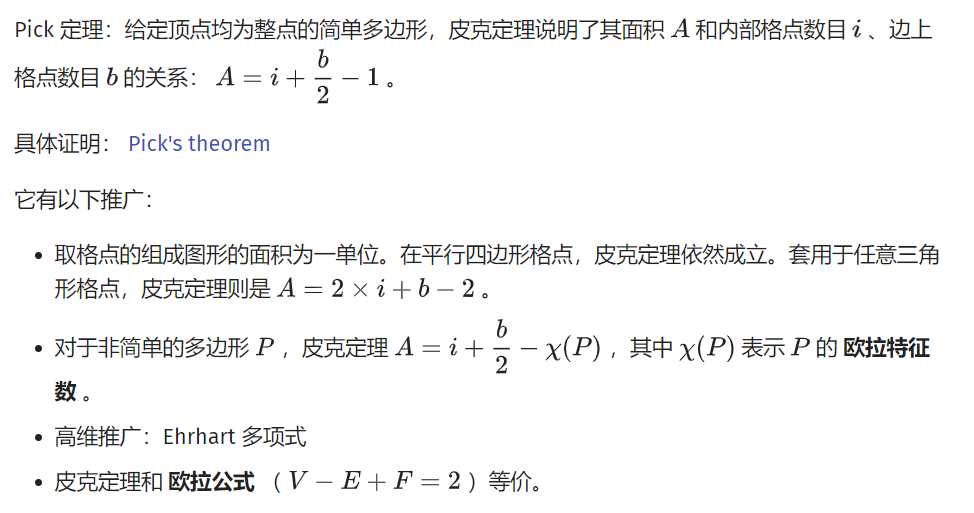
#include <algorithm>  
#include <cstdio>  
  
typedef long long ll;  
  
const int K = 105;  
const int N = 100005;  
const int mod = 1e9 + 7;  
  
int T, n, k, a[K], b[K], fact[N << 1], m[K][K];  
  
int qpow(int x, int y) {  
 int out = 1;  
 while (y) {  
 if (y & 1) out = (ll)out \* x % mod;  
 x = (ll)x \* x % mod;  
 y >>= 1;  
 }  
 return out;  
}  
int c(int x, int y) {  
 return (ll)fact[x] \* qpow(fact[y], mod - 2) % mod \*  
 qpow(fact[x - y], mod - 2) % mod;  
}  
int main() {  
 fact[0] = 1;  
 for (int i = 1; i < N \* 2; ++i) fact[i] = (ll)fact[i - 1] \* i % mod;  
  
 scanf("%d", &T);  
  
 while (T--) {  
 scanf("%d%d", &n, &k);  
  
 for (int i = 1; i <= k; ++i) scanf("%d", a + i);  
 for (int i = 1; i <= k; ++i) scanf("%d", b + i);  
  
 for (int i = 1; i <= k; ++i) {  
 for (int j = 1; j <= k; ++j) {  
 if (a[i] <= b[j])  
 m[i][j] = c(b[j] - a[i] + n - 1, n - 1);  
 else  
 m[i][j] = 0;  
 }  
 }  
  
 for (int i = 1; i < k; ++i) {  
 if (!m[i][i]) {  
 for (int j = i + 1; j <= k; ++j) {  
 if (m[j][i]) {  
 std::swap(m[i], m[j]);  
 break;  
 }  
 }  
 }  
 if (!m[i][i]) continue;  
 for (int j = i + 1; j <= k; ++j) {  
 if (!m[j][i]) continue;  
 int mul = (ll)m[j][i] \* qpow(m[i][i], mod - 2) % mod;  
 for (int p = i; p <= k; ++p) {  
 m[j][p] = (m[j][p] - (ll)m[i][p] \* mul % mod + mod) % mod;  
 }  
 }  
 }  
  
 int ans = 1;  
  
 for (int i = 1; i <= k; ++i) ans = (ll)ans \* m[i][i] % mod;  
  
 printf("%d\n", ans);  
 }  
  
 return 0;  
}

## 弦图

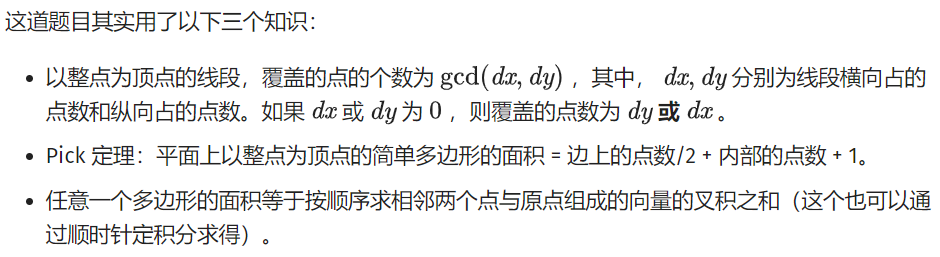
https://oi-wiki.org/graph/chord/

# 计算几何

## Pick 定理



例题：给一个平面上的简单多边形，求边上的点，多边形内的点，多边形面积。



#include <cmath>  
#include <cstdio>  
#include <iostream>  
using namespace std;  
const int MAXN = 110;  
struct node {  
 int x, y;  
} p[MAXN];  
inline int gcd(int x, int y) { return y == 0 ? x : gcd(y, x % y); }  
inline int area(int a, int b) { return p[a].x \* p[b].y - p[a].y \* p[b].x; }  
int main() {  
 int t, ncase = 1;  
 scanf("%d", &t);  
 while (t--) {  
 int n, dx, dy, x, y, num = 0, sum = 0;  
 scanf("%d", &n);  
 p[0].x = 0, p[0].y = 0;  
 for (int i = 1; i <= n; i++) {  
 scanf("%d%d", &x, &y);  
 p[i].x = x + p[i - 1].x, p[i].y = y + p[i - 1].y;  
 dx = x, dy = y;  
 if (x < 0) dx = -x;  
 if (y < 0) dy = -y;  
 num += gcd(dx, dy);  
 sum += area(i - 1, i);  
 }  
 if (sum < 0) sum = -sum;  
 printf("Scenario #%d:\n", ncase++);  
 printf("%d %d %.1f\n\n", (sum - num + 2) >> 1, num, sum \* 0.5);  
 }  
 return 0;  
}

## 三角剖分

https://oi-wiki.org/geometry/triangulation/

## 凸包

// stk[] 是整型，存的是下标  
// p[] 存储向量或点  
tp = 0; // 初始化栈  
std::sort(p + 1, p + 1 + n); // 对点进行排序  
stk[++tp] = 1;  
//栈内添加第一个元素，且不更新 used，使得 1 在最后封闭凸包时也对单调栈更新  
for (int i = 2; i <= n; ++i) {  
 while (tp >= 2 // 下一行 \* 操作符被重载为叉积  
 && (p[stk[tp]] - p[stk[tp - 1]]) \* (p[i] - p[stk[tp]]) <= 0)  
 used[stk[tp--]] = 0;  
 used[i] = 1; // used 表示在凸壳上  
 stk[++tp] = i;  
}  
int tmp = tp; // tmp 表示下凸壳大小  
for (int i = n - 1; i > 0; --i)  
 if (!used[i]) {  
 // ↓求上凸壳时不影响下凸壳  
 while (tp > tmp && (p[stk[tp]] - p[stk[tp - 1]]) \* (p[i] - p[stk[tp]]) <= 0)  
 used[stk[tp--]] = 0;  
 used[i] = 1;  
 stk[++tp] = i;  
 }  
for (int i = 1; i <= tp; ++i) // 复制到新数组中去  
 h[i] = p[stk[i]];  
int ans = tp - 1;

## 扫描线

### Atlantis 问题

在二维坐标系上，给出多个矩形的左下以及右上坐标，求出所有矩形构成的图形的面积。

#include <algorithm>  
#include <cstdio>  
#include <cstring>  
#define maxn 300  
using namespace std;  
  
int lazy[maxn << 3]; // 标记了这条线段出现的次数  
double s[maxn << 3];  
  
struct node1 {  
 double l, r;  
 double sum;  
} cl[maxn << 3]; // 线段树  
  
struct node2 {  
 double x, y1, y2;  
 int flag;  
} p[maxn << 3]; // 坐标  
  
//定义sort比较  
bool cmp(node2 a, node2 b) { return a.x < b.x; }  
  
//上传  
void pushup(int rt) {  
 if (lazy[rt] > 0)  
 cl[rt].sum = cl[rt].r - cl[rt].l;  
 else  
 cl[rt].sum = cl[rt \* 2].sum + cl[rt \* 2 + 1].sum;  
}  
  
//建树  
void build(int rt, int l, int r) {  
 if (r - l > 1) {  
 cl[rt].l = s[l];  
 cl[rt].r = s[r];  
 build(rt \* 2, l, (l + r) / 2);  
 build(rt \* 2 + 1, (l + r) / 2, r);  
 pushup(rt);  
 } else {  
 cl[rt].l = s[l];  
 cl[rt].r = s[r];  
 cl[rt].sum = 0;  
 }  
 return;  
}  
  
//更新  
void update(int rt, double y1, double y2, int flag) {  
 if (cl[rt].l == y1 && cl[rt].r == y2) {  
 lazy[rt] += flag;  
 pushup(rt);  
 return;  
 } else {  
 if (cl[rt \* 2].r > y1) update(rt \* 2, y1, min(cl[rt \* 2].r, y2), flag);  
 if (cl[rt \* 2 + 1].l < y2)  
 update(rt \* 2 + 1, max(cl[rt \* 2 + 1].l, y1), y2, flag);  
 pushup(rt);  
 }  
}  
  
int main() {  
 int temp = 1, n;  
 double x1, y1, x2, y2, ans;  
 while (scanf("%d", &n) && n) {  
 ans = 0;  
 for (int i = 0; i < n; i++) {  
 scanf("%lf %lf %lf %lf", &x1, &y1, &x2, &y2);  
 p[i].x = x1;  
 p[i].y1 = y1;  
 p[i].y2 = y2;  
 p[i].flag = 1;  
 p[i + n].x = x2;  
 p[i + n].y1 = y1;  
 p[i + n].y2 = y2;  
 p[i + n].flag = -1;  
 s[i + 1] = y1;  
 s[i + n + 1] = y2;  
 }  
 sort(s + 1, s + (2 \* n + 1)); // 离散化  
 sort(p, p + 2 \* n, cmp); // 把矩形的边的纵坐标从小到大排序  
 build(1, 1, 2 \* n); // 建树  
 memset(lazy, 0, sizeof(lazy));  
 update(1, p[0].y1, p[0].y2, p[0].flag);  
 for (int i = 1; i < 2 \* n; i++) {  
 ans += (p[i].x - p[i - 1].x) \* cl[1].sum;  
 update(1, p[i].y1, p[i].y2, p[i].flag);  
 }  
 printf("Test case #%d\nTotal explored area: %.2lf\n\n", temp++, ans);  
 }  
 return 0;  
}

## 旋转卡壳

#include <stdio.h>  
#include <string.h>  
#include <algorithm>  
#include <iostream>  
using namespace std;  
  
struct Point  
{  
 int x,y;  
 Point(int \_x = 0, int \_y = 0)  
 {  
 x = \_x;  
 y = \_y;  
 }  
 Point operator -(const Point &b)const  
 {  
 return Point(x - b.x, y - b.y);  
 }  
 int operator ^(const Point &b)const  
 {  
 return x\*b.y - y\*b.x;  
 }  
 int operator \*(const Point &b)const  
 {  
 return x\*b.x + y\*b.y;  
 }  
 void input()  
 {  
 scanf("%d%d",&x,&y);  
 }  
};  
int dist2(Point a,Point b)  
{  
 return (a-b)\*(a-b);  
}  
int cross(Point p0,Point p1,Point p2) //计算叉积 p0p1 X p0p2  
{  
 return (p1.x-p0.x)\*(p2.y-p0.y)-(p1.y-p0.y)\*(p2.x-p0.x);  
}  
const int MAXN = 50010;  
Point list[MAXN];  
int Stack[MAXN],top;  
bool \_cmp(Point p1,Point p2)  
{  
 int tmp = cross(p1,p2,list[0]);  
 if(tmp > 0)return true;  
 else if(tmp == 0 && dist2(p1,list[0]) <= dist2(p2,list[0]))  
 return true;  
 else return false;  
}  
void Graham(int n)  
{  
 Point p0;  
 int k = 0;  
 p0 = list[0];  
 for(int i = 1;i < n;i++)  
 if(p0.y > list[i].y || (p0.y == list[i].y && p0.x > list[i].x))  
 {  
 p0 = list[i];  
 k = i;  
 }  
 swap(list[0],list[k]);  
 sort(list+1,list+n,\_cmp);  
 if(n == 1)  
 {  
 top = 1;  
 Stack[0] = 0;  
 return;  
 }  
 if(n == 2)  
 {  
 top = 2;  
 Stack[0] = 0;  
 Stack[1] = 1;  
 return;  
 }  
 Stack[0] = 0;  
 Stack[1] = 1;  
 top = 2;  
 for(int i = 2;i < n;i++)  
 {  
 while(top > 1 && (cross(list[Stack[top-1]],list[i],list[Stack[top-2]])) <= 0 )  
 top--;  
 Stack[top++] = i;  
 }  
}  
//旋转卡壳，求两点间距离平方的最大值  
int rotating\_calipers(Point p[],int n)  
{  
 int ans = 0;  
 Point v;  
 int cur = 1;  
 for(int i = 0;i < n;i++)  
 {  
 v = p[i]-p[(i+1)%n];  
 while((v^(p[(cur+1)%n]-p[cur])) < 0)  
 cur = (cur+1)%n;  
 //printf("%d %d\n",i,cur);  
 ans = max(ans,max(dist2(p[i],p[cur]),dist2(p[(i+1)%n],p[(cur+1)%n])));  
 }  
 return ans;  
}  
Point p[MAXN];  
int main()  
{  
 int n;  
 while(scanf("%d",&n) == 1)  
 {  
 for(int i = 0;i < n;i++)  
 list[i].input();  
 Graham(n);  
 for(int i = 0;i < top;i++)  
 p[i] = list[Stack[i]];  
 printf("%d\n",rotating\_calipers(p,top));  
 }  
 return 0;  
}

## 随机增量法

在一个平面上有n个点，求一个半径最小的圆，能覆盖所有的点

空间复杂度和时间复杂度O(n)

#include <cmath>  
#include <cstdio>  
#include <cstdlib>  
#include <cstring>  
#include <iostream>  
  
using namespace std;  
  
int n;  
double r;  
  
struct point {  
 double x, y;  
} p[100005], o;  
  
inline double sqr(double x) { return x \* x; }  
  
inline double dis(point a, point b) {  
 return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y));  
}  
  
inline bool cmp(double a, double b) { return fabs(a - b) < 1e-8; }  
  
point geto(point a, point b, point c) {  
 double a1, a2, b1, b2, c1, c2;  
 point ans;  
 a1 = 2 \* (b.x - a.x), b1 = 2 \* (b.y - a.y),  
 c1 = sqr(b.x) - sqr(a.x) + sqr(b.y) - sqr(a.y);  
 a2 = 2 \* (c.x - a.x), b2 = 2 \* (c.y - a.y),  
 c2 = sqr(c.x) - sqr(a.x) + sqr(c.y) - sqr(a.y);  
 if (cmp(a1, 0)) {  
 ans.y = c1 / b1;  
 ans.x = (c2 - ans.y \* b2) / a2;  
 } else if (cmp(b1, 0)) {  
 ans.x = c1 / a1;  
 ans.y = (c2 - ans.x \* a2) / b2;  
 } else {  
 ans.x = (c2 \* b1 - c1 \* b2) / (a2 \* b1 - a1 \* b2);  
 ans.y = (c2 \* a1 - c1 \* a2) / (b2 \* a1 - b1 \* a2);  
 }  
 return ans;  
}  
  
int main() {  
 scanf("%d", &n);  
 for (int i = 1; i <= n; i++) scanf("%lf%lf", &p[i].x, &p[i].y);  
 for (int i = 1; i <= n; i++) swap(p[rand() % n + 1], p[rand() % n + 1]);  
 o = p[1];  
 for (int i = 1; i <= n; i++) {  
 if (dis(o, p[i]) < r || cmp(dis(o, p[i]), r)) continue;  
 o.x = (p[i].x + p[1].x) / 2;  
 o.y = (p[i].y + p[1].y) / 2;  
 r = dis(p[i], p[1]) / 2;  
 for (int j = 2; j < i; j++) {  
 if (dis(o, p[j]) < r || cmp(dis(o, p[j]), r)) continue;  
 o.x = (p[i].x + p[j].x) / 2;  
 o.y = (p[i].y + p[j].y) / 2;  
 r = dis(p[i], p[j]) / 2;  
 for (int k = 1; k < j; k++) {  
 if (dis(o, p[k]) < r || cmp(dis(o, p[k]), r)) continue;  
 o = geto(p[i], p[j], p[k]);  
 r = dis(o, p[i]);  
 }  
 }  
 }  
 printf("%.10lf\n%.10lf %.10lf", r, o.x, o.y);  
 return 0;  
}

# DP

## 背包dp

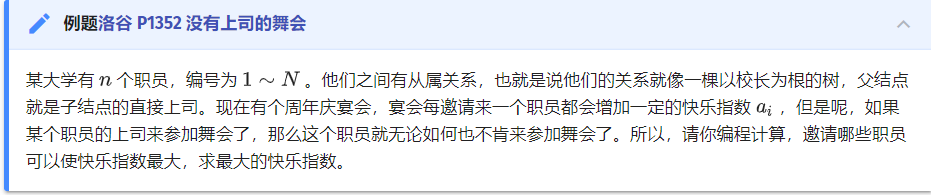
/\*\*/  
#include <cstdio>  
#include <cstring>  
#include <cmath>  
#include <cctype>  
#include <iostream>  
#include <algorithm>  
#include <map>  
#include <set>  
#include <vector>  
#include <string>  
#include <stack>  
#include <queue>  
  
typedef long long LL;  
using namespace std;  
  
const int MX = 500000;  
  
int n, m, k[205], a[205], dp[500005], V;  
  
struct Node{  
 int pos, value;  
}que[500005];  
  
void MultiPack(int num, int cost, int value){  
 if(V / cost < num) num = V / cost;  
 for(int mo = 0; mo < cost; mo++)  
 {  
 int head = 0, tail = 0;  
 for(int k = 0; k <= (V - mo) / cost; k++)   
 {  
 int x = k;  
 int y = dp[k \* cost + mo] - k \* value;  
 while(head < tail && que[head].pos < k - num) head++;  
 while(head < tail && que[tail - 1].value <= y) tail--;  
 que[tail].value = y, que[tail].pos = x;  
 tail++;  
 dp[k \* cost + mo] = que[head].value + k \* value;  
 // printf("==%d %d\n", k \* cost + mo, dp[k \* cost + mo]);  
 }  
 }  
}  
  
int main()  
{  
 //freopen("in.txt", "r", stdin);  
 //freopen("out.txt", "w", stdout);  
  
 scanf("%d %d", &n, &m);  
 for (int i = 1; i <= n; i++){  
 scanf("%d %d", &k[i], &a[i]);  
 V += k[i] \* a[i];  
 }  
 V = min(V, MX);  
 // dp[0] = 1;  
 for (int i = 1; i <= n; i++) MultiPack(a[i], k[i], k[i]);  
 for (int i = 1, x; i <= m; i++){  
 scanf("%d", &x);  
 if(dp[x] == x) printf("Yes\n");  
 else printf("No\n");  
 }  
  
 return 0;  
}  
/\*\*/

## 区间dp

核心代码

for (len = 1; len <= n; len++)  
 for (i = 1; i <= 2 \* n - 1; i++) {  
 int j = len + i - 1;  
 for (k = i; k < j && k <= 2 \* n - 1; k++)  
 f[i][j] = max(f[i][j], f[i][k] + f[k + 1][j] + sum[j] - sum[i - 1]);  
 }

## 树形dp



#include <algorithm>  
#include <cstdio>  
using namespace std;  
struct edge {  
 int v, next;  
} e[6005];  
int head[6005], n, cnt, f[6005][2], ans, is\_h[6005], vis[6005];  
void addedge(int u, int v) {  
 e[++cnt].v = v;  
 e[cnt].next = head[u];  
 head[u] = cnt;  
}  
void calc(int k) {  
 vis[k] = 1;  
 for (int i = head[k]; i; i = e[i].next) { // 枚举该结点的每个子结点  
 if (vis[e[i].v]) continue;  
 calc(e[i].v);  
 f[k][1] += f[e[i].v][0];  
 f[k][0] += max(f[e[i].v][0], f[e[i].v][1]);  
 }  
 return;  
}  
int main() {  
 scanf("%d", &n);  
 for (int i = 1; i <= n; i++) scanf("%d", &f[i][1]);  
 for (int i = 1; i < n; i++) {  
 int l, k;  
 scanf("%d%d", &l, &k);  
 is\_h[l] = 1;  
 addedge(k, l);  
 }  
 for (int i = 1; i <= n; i++)  
 if (!is\_h[i]) { // 从根结点开始DFS  
 calc(i);  
 printf("%d", max(f[i][1], f[i][0]));  
 return 0;  
 }  
}

## 动态dp

https://oi-wiki.org/dp/dynamic/

## DP优化

### 单调队列

#include <algorithm>  
#include <cmath>  
#include <cstdio>  
#include <cstring>  
#include <iostream>  
const int maxn = 150000 + 10;  
const int maxm = 300 + 10;  
const ll inf = 0xcfcfcfcfcfcfcfcf;  
  
ll f[2][maxn];  
ll a[maxm], b[maxm], t[maxm];  
int n, m, d;  
  
int que[maxn];  
int fl = 1;  
  
void init() {  
 memset(f, inf, sizeof(f));  
 memset(que, 0, sizeof(que));  
 \_rep(i, 1, n) f[0][i] = 0;  
 fl = 1;  
}  
  
void dp() {  
 init();  
 for (int i = 1; i <= m; i++) {  
 int l = 1, r = 0;  
  
 int k = 1;  
 for (int j = 1; j <= n; j++) {  
 for (; k <= min(1ll \* n, j + d \* (t[i] - t[i - 1])); k++) {  
 while (l <= r && f[fl ^ 1][que[r]] <= f[fl ^ 1][k]) r--;  
 que[++r] = k;  
 }  
  
 while (l <= r && que[l] < max(1ll, j - d \* (t[i] - t[i - 1]))) l++;  
 f[fl][j] = f[fl ^ 1][que[l]] - abs(a[i] - j) + b[i];  
 }  
  
 fl ^= 1;  
 }  
}  
  
int main() {  
 cin >> n >> m >> d;  
 \_rep(i, 1, m) { cin >> a[i] >> b[i] >> t[i]; }  
  
 dp();  
 ll ans = inf;  
 \_rep(i, 1, n) ans = max(ans, f[fl ^ 1][i]);  
 cout << ans << endl;  
}

### 斜率优化

## 数位dp



int dfs(int x, int st, int op) // op=1 =;op=0 <  
{  
 if (!x) return 1;  
 if (!op && ~f[x][st]) return f[x][st];  
 int maxx = op ? dim[x] : 9, ret = 0;  
 for (int i = 0; i <= maxx; i++) {  
 if (abs(st - i) < 2) continue;  
 if (st == 11 && i == 0)  
 ret += dfs(x - 1, 11, op & (i == maxx));  
 else  
 ret += dfs(x - 1, i, op & (i == maxx));  
 }  
 if (!op) f[x][st] = ret;  
 return ret;  
}  
int solve(int x) {  
 memset(f, -1, sizeof f);  
 dim.clear();  
 dim.push\_back(-1);  
 int t = x;  
 while (x) {  
 dim.push\_back(x % 10);  
 x /= 10;  
 }  
 return dfs(dim.size() - 1, 11, 1);  
}

## 插头dp

https://oi-wiki.org/dp/plug/