ACM Template

HouZAJ

2019.11.2

Table of Content

1 技巧

1.1 VIM 配置

```
syntax on
set cindent
set nu
set tabstop=4
set shiftwidth=4
set background=dark
set mouse=a

map <C-A> ggVG"+y
```

1.2 运行脚本

```
!/bin/bash

rm ./$1
g++ $1.cpp -std=gnu++14 -Wall -o $1
./$1
```

1.3 Fast IO

1.3.1 C++ 二进制快读

```
inline char get(void) {
    static char buf[1000000], *p1 = buf, *p2 = buf;
    if (p1 == p2) {
        p2 = (p1 = buf) + fread(buf, 1, 1000000, stdin);
        if (p1 == p2) return EOF;
    }
    return *p1++;
}

inline int read() {
    int x = 0; static char c; bool minus = false;
    for (; !(c >= '0' && c <= '9'); c = get()) if (c == '-') minus =
    ↪  true;
    for (; c >= '0' && c <= '9'; x = x * 10 + c - '0', c = get()); if
    ↪  (minus) x = -x;
    return x;
}
```

1.3.2 C++ 非二进制快读

```
inline int read() {
    int x = 0; static char c; bool minus = false;
    for (; !(c >= '0' && c <= '9'); c = getchar()) if (c == '-')
     ↪  minus = true;
    for (; c >= '0' && c <= '9'; x = x * 10 + c - '0', c =
     ↪  getchar()); if (minus) x = -x;
    return x;
}
```

1.3.3 C++ 手写输出

```
char WritellBuffer[1024];
template <typename T>
inline void write(T a,char end){
    ll cnt=0,fu=1;
    if(a<0){putchar('-');fu=-1;}
    do{WritellBuffer[++cnt]=fu*(a%10)+'0';a/=10;}while(a);
    while(cnt){putchar(WritellBuffer[cnt]);--cnt;}
    if(end) putchar(end);
}
```

1.3.4 Java IO 类

```
class InputReader {
    public BufferedReader reader;
    public StringTokenizer tokenizer;

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

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

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

```
    }
}
```

2 字符串

2.1 后缀自动机

```
char s[N];

struct SuffixAutomaton {
    int slink[N], len[N], trans[N][SIZE], cnt[N], buc[N], vec[N];
    ll dp[N];
    int lst, tot;

    inline void init() {
        tot = 0;
        lst = newNode();
        slink[lst] = len[lst] = 0;
    }

    inline int newNode() {
        int x = ++tot;
        memset(trans[x], 0, sizeof(trans[x]));
        dp[x] = idg[x] = cnt[x] = 0;
        return x;
    }

    inline void push(int val) {
        int p = lst, np = newNode();
        len[np] = len[p] + 1;
        cnt[np] = 1;

        for(; p && trans[p][val] == 0; p = slink[p]) {
            trans[p][val] = np;
        }

        if(p == 0) {
            slink[np] = 1;
        } else {
            int q = trans[p][val];
            if(len[q] == len[p] + 1) {
                slink[np] = q;
            } else {
                int nq = ++tot;
                cnt[nq] = dp[nq] = 0;
                slink[nq] = slink[q];
                len[nq] = len[p] + 1;
```

```
                memcpy(trans[nq], trans[q], sizeof(trans[q]));
                slink[np] = slink[q] = nq;
                for(; p && trans[p][val] == q; p = slink[p]) {
                    trans[p][val] = nq;
                }
            }
        }
        lst = np;
    }

    inline void getCnt() {
        memset(buc, 0, (tot + 1) * sizeof(int));
        for(int i = 2; i <= tot; i++) {
            buc[len[i]]++;
        }
        for(int i = 2; i <= tot; i++) {
            buc[i] += buc[i - 1];
        }
        for(int i = tot; i >= 2; i--) {
            vec[buc[len[i]]--] = i;
        }

        for(int i = tot - 1; i >= 1; i--) {
            int u = vec[i];
            cnt[slink[u]] += cnt[u];
        }
        cnt[1] = 0;
    }
};

SuffixAutomaton sam;
```

2.2 后缀数组

2.2.1 求最长公共子串

```
#include <bits/stdc++.h>
using namespace std;
const int N = (int)200000 + 15;

int t1[N], t2[N], buc[N], sa[N], rk[N], height[N];
char s[N];

inline void buildSA(char* s, int n, int m = 128) {
    int* x = t1, *y = t2;
    memset(buc + 1, 0, m * sizeof(int));
    for(int i = 1; i <= n; i++) {
        buc[x[i] = s[i]]++;
```

```
    }
    for(int i = 1; i <= m; i++) {
        buc[i] += buc[i - 1];
    }
    for(int i = n; i >= 1; i--) {
        sa[buc[x[i]]--] = i;
    }

    for(int k = 1; k <= n; k <<= 1) {
        int p = 0;
        for(int i = n - k + 1; i <= n; i++) {
            y[++p] = i;
        }
        for(int i = 1; i <= n; i++) {
            if(sa[i] > k) {
                y[++p] = sa[i] - k;
            }
        }
        memset(buc + 1, 0, m * sizeof(int));
        for(int i = 1; i <= n; i++) {
            buc[x[i]]++;
        }
        for(int i = 1; i <= m; i++) {
            buc[i] += buc[i - 1];
        }
        for(int i = n; i >= 1; i--) {
            sa[buc[x[y[i]]]--] = y[i];
        }
        swap(x, y);
        p = x[sa[1]] = 1;
        for(int i = 2; i <= n; i++) {
            int a = sa[i] + k > n ? -1 : y[sa[i] + k];
            int b = sa[i - 1] + k > n ? -1 : y[sa[i - 1] + k];
            x[sa[i]] = (y[sa[i]] == y[sa[i - 1]] && a == b) ? p :
             ↪  ++p;
        }
        if(p >= n) {
            break;
        }
        m = p;
    }
}

inline void getHeight(char* s, int n) {
    for(int i = 1; i <= n; i++) {
        rk[sa[i]] = i;
    }
    int k = 0;
```

```
    for(int i = 1; i <= n; i++) {
        if(rk[i] == 1) {
            continue;
        }
        if(k) {
            k--;
        }
        int j = sa[rk[i] - 1];
        while(s[i + k] == s[j + k]) {
            k++;
        }
        height[rk[i]] = k;
    }
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);

    while(cin >> (s + 1)) {
        int pos = strlen(s + 1) + 1;
        s[pos] = '$';
        cin >> (s + 1 + pos);
        int n = strlen(s + 1);
        buildSA(s, n);
        getHeight(s, n);

        int maxx = 0;
        for(int i = 2; i <= n; i++) {
            if(maxx < height[i] && (sa[i - 1] < pos && sa[i] > pos ||
            ↪  sa[i - 1] > pos && sa[i] < pos)) {
                maxx = height[i];
            }
        }
        cout << maxx << "\n";
    }
    cout.flush();
}
```

2.3 AC 自动机

```
char str[10000 + 1];
int  pos[N];
int  nxt[N][ascii_size];
int  fail[N];
int  tot;

int  que[N];
```

```
int  head, tail;

inline void newNode(int& x){
    x = tot++;
    memset(nxt[x], -1, sizeof(nxt[x]));
    fail[x] = 0;
    pos[x] = 0;
}

inline void init(){
    head = tail = 0;
    tot = 1;
    memset(nxt[0], -1, sizeof(nxt[0]));
    fail[0] = pos[0] = 0;
}

void push(char s[], int i){
    int p = 0;
    for(int i = 0; s[i]; i++){
        int idx = s[i] - 32;
        if(nxt[p][idx] == -1){
            newNode(nxt[p][idx]);
        }
        p = nxt[p][idx];
    }
    pos[p] = i;
}

void setFail(){
    for(int idx = 0; idx < ascii_size; idx++){
        if(nxt[0][idx] != -1){
            que[head++] = nxt[0][idx];
        }else{
            nxt[0][idx] = 0;
        }
    }
    while(tail != head){
        int p = que[tail++];
        for(int idx = 0; idx < ascii_size; idx++){
            if(~nxt[p][idx]){
                fail[nxt[p][idx]] = nxt[fail[p]][idx];
                que[head++] = nxt[p][idx];
            }else{
                nxt[p][idx] = nxt[fail[p]][idx];
            }
        }
    }
}
```

```
void query(char s[]){
    int p = 0;
    for(int i = 0; s[i]; i++){
        int idx = s[i] - 32;
        p = nxt[p][idx];
        for(int q = p; q; q = fail[q]){
            if(pos[q]) {
                // DO SOMETHING
            }
        }
    }
}
```

2.4 Z 函数

```
char s[N];
int z[N];

inline void zFunc(char* s, int n) {
    z[1] = 0;
    for(int i = 2, l = 1, r = 1; i <= n; i++) {
        z[i] = 0;
        if(i <= r) {
            z[i] = min(z[i - l + 1], r - i + 1);
        }
        while(i + z[i] <= n && s[1 + z[i]] == s[i + z[i]]) {
            z[i]++;
        }
        if(i + z[i] - 1 > r) {
            l = i;
            r = i + z[i] - 1;
        }
    }
}
```

2.5 KMP

```
int knxt[N];

void getNext(char* p){
    knxt[0] = -1;
    int k = -1, j = 0;
    while(p[j]){
        if(k == -1 || p[k] == p[j]){
            j++;
            k++;
```

```
            knxt[j] = (p[k] != p[j] ? k : knxt[k]);
            // knxt[j] = k; //未优化版本，可求循环节
        }else{
            k = knxt[k];
        }
    }
}

int kmpSearch(char* s, char* p){
    getNext(p);
    int i = 0, j = 0;
    int cnt = 0;
    while(s[i]){
        if(j == -1 || s[i] == p[j]){
            i++;
            j++;
        }else{
            j = knxt[j];
        }

        if(j>=0&&!p[j]){
            cnt++;
            j = 0;
        }
    }
    return cnt;
}
```

2.6 字典树

```
int  cnt[N];
int  nxt[N][26];
char str[12];
int  tot;

inline void init(){
    tot = 1;
    memset(cnt, 0, sizeof(cnt));
    memset(nxt[0], -1, sizeof(nxt));
}

void push(char* s){
    int p = 0;
    for(int i = 0; s[i]; i++){
        int idx = s[i] - 'a';
        if(nxt[p][idx] == -1){
            nxt[p][idx] = tot++;
        }
```

```
        cnt[nxt[p][idx]]++;
        p = nxt[p][idx];
    }
}

int query(char* s){
    int p = 0;
    for(int i = 0; s[i]; i++){
        int idx = s[i] - 'a';
        if(nxt[p][idx] == -1)   return 0;
        p = nxt[p][idx];
    }
    return cnt[p];
}
```

2.7 Manacher

```
const int N = (int)1e5 + 15;

char tmp[N], s[2 * N];
int d[2 * N];

inline void manacher(char* s, int n) {
    for(int i = 1, l = 0, r = -1; i <= n; i++) {
        int k = (i > r ? 1 : min(d[l + r - i], r - i + 1));
        while(i - k >= 1 && i + k <= n && s[i - k] == s[i + k]) {
            k++;
        }
        d[i] = k;
        if(i + k - 1 > r) {
            r = i + k - 1;
            l = i - k + 1;
        }
    }
}

int main() {
    while(~scanf("%s", tmp + 1)) {
        int n = 0, m = 0;
        s[++n] = '#';
        for(int i = 1; tmp[i]; i++) {
            s[++n] = tmp[i];
            s[++n] = '#';
            ++m;
        }
        s[n + 1] = 0;
        manacher(s, n);
        //...
```

```
    }
}
```

2.8 回文树

```
char s[N];

struct Node{
    int slink, len, cnt;
    int nxt[26];
};

struct PalindromicTree{
    int  tot;
    int  cursuffix;
    Node tree[N];

    void init(){
        tree[0].slink = 0, tree[0].len = -1, tree[0].cnt = 1;
        tree[1].slink = 0, tree[1].len = 0,  tree[1].cnt = 1;
        tot = 2, cursuffix = 1;
        initNode(1);
        initNode(0);
    }

    void initNode(int o){
        memset(tree[o].nxt, -1, sizeof(tree[o].nxt));
    }

    void add(int pos){
        int idx = s[pos] - 'a';
        int cur = cursuffix;
        while(true){
            int curlen = tree[cur].len;
            if(pos - 1 - curlen >= 0 && s[pos] == s[pos - 1 -
             ↪  curlen])      break;
            cur = tree[cur].slink;
        }

        if(tree[cur].nxt[idx] != -1){
            cursuffix = tree[cur].nxt[idx];
            return;
        }

        int nxt = tree[cur].nxt[idx] = tot++;
        initNode(nxt);
        tree[nxt].len = tree[cur].len + 2;
        cursuffix = nxt;
```

```cpp
        if(tree[nxt].len == 1){
            tree[nxt].cnt = 1;
            tree[nxt].slink = 1;
            return;
        }

        while(true){
            cur = tree[cur].slink;
            int curlen = tree[cur].len;
            if(pos - 1 - curlen >= 0 && s[pos] == s[pos - 1 -
             ↪  curlen]){
                tree[nxt].slink = tree[cur].nxt[idx];
                break;
            }
        }

        tree[nxt].cnt = tree[tree[nxt].slink].cnt + 1;
    }
};

PalindromicTree pt;
```

2.9 表达式

2.9.1 调度场算法（后缀表达式）

```cpp
void getSuffixExp(char s[]){
    pp = 0;
    stack<char> stk;
    for(int i = 0; s[i]; ){
        if(s[i] == ' '){
            i++;
            continue;
        }

        if(isdigit(s[i])){
            sscanf(s + i, "%d", &suffix_exp[pp].data);
            suffix_exp[pp++].type = 0;
            while(isdigit(s[i]))    i++;
        }else{
            while(!stk.empty() && getPriority(stk.top()) >=
             ↪  getPriority(s[i])){
                suffix_exp[pp++] = Data{stk.top(), 1};
                stk.pop();
            }

            stk.push(s[i]);
```

```
            i++;
        }
    }
    while(!stk.empty()){
        suffix_exp[pp++] = Data{stk.top(), 1};
        stk.pop();
    }
}

double solve(){
    stack<double> ans;
    for(int i = 0; i < pp; i++){
        if(suffix_exp[i].type == 0)     ans.push(suffix_exp[i].data);
        else{
            double t2 = ans.top();
            ans.pop();
            double t1 = ans.top();
            ans.pop();

            if(suffix_exp[i].data == '+')         ans.push(t1 + t2);
            else if(suffix_exp[i].data == '-')    ans.push(t1 - t2);
            else if(suffix_exp[i].data == '*')    ans.push(t1 * t2);
            else if(suffix_exp[i].data == '/')    ans.push(t1 / t2);
        }
    }
    return ans.top();
}
```

2.9.2 后缀表达式 + 表达式树（化简表达式）

```
int getPriority(char ch){
    if(ch == '*' || ch == '/')          return 2;
    else if(ch == '+' || ch == '-')     return 1;
    else                                return 0;
}

bool isOperation(char ch){
    return (ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch ==
     ↪  '(' || ch == ')');
}

void getSuffixExp(char s[]){
    stack<char> stk;
    pp = 0;
    for(int i = 0; s[i]; i++){
        if(isOperation(s[i])){
            if(s[i] == ')'){
                while(!stk.empty() && stk.top() != '('){
```

```
                suffix_exp[pp++] = make_pair(stk.top(), 1);
                stk.pop();
            }
            stk.pop();
        }else{
            while(s[i] != '(' && !stk.empty() &&
            ↪  getPriority(stk.top()) >= getPriority(s[i])){
            ↪  //)
                suffix_exp[pp++] = make_pair(stk.top(), 1);
                stk.pop();
            }
            stk.push(s[i]);
        }
    }else{
        suffix_exp[pp++] = make_pair(s[i], 0);
    }
  }
  while(!stk.empty()){
      suffix_exp[pp++] = make_pair(stk.top(), 1);
      stk.pop();
  }
}

void newNode(int rt, char val){
    tree[rt].ch[0] = tree[rt].ch[1] = 0;
    tree[rt].data = val;
}

int buildTree(){
    int rt = 0;
    stack<int> stk;
    for(int i = 0; i < pp; i++){
        if(suffix_exp[i].second == 0){
            newNode(++rt, suffix_exp[i].first);
            stk.push(rt);
        }else{
            int t2 = stk.top();
            stk.pop();
            int t1 = stk.top();
            stk.pop();

            newNode(++rt, suffix_exp[i].first);
            tree[rt].ch[0] = t1;
            tree[rt].ch[1] = t2;
            stk.push(rt);
        }
    }
    return rt;
```

```
}

void dfs(int u, int pre, bool is_left){
    if(!isOperation(tree[u].data)){
        putchar(tree[u].data);
        return;
    }

    bool flag = false;
    if(pre != -1 && is_left && getPriority(tree[u].data) <
    ↪  getPriority(tree[pre].data))                    flag = true;
    if(pre != -1 && !is_left){
        if(getPriority(tree[u].data) < getPriority(tree[pre].data))
        ↪  flag = true;
        else if(getPriority(tree[u].data) ==
        ↪  getPriority(tree[pre].data) && (tree[pre].data == '-' ||
        ↪  tree[pre].data == '/'))        flag = true;
    }

    if(flag)    putchar('(');
    dfs(tree[u].ch[0], u, true);
    putchar(tree[u].data);
    dfs(tree[u].ch[1], u, false);
    if(flag)    putchar(')');
}
```

3 数学

3.1 快速幂

3.1.1 快速幂

```
inline int quickPow(int a, int b) {
    int ans = 1, base = a;
    while(b) {
        if(b & 1LL) {
            ans = (ll)ans * base % MOD;
        }
        base = (ll)base * base % MOD;
        b >>= 1LL;
    }
    return ans;
}
```

3.1.2 矩阵快速幂

```
struct Matrix {
    int met[metSize][metSize];

    inline void initZero() {
        memset(met, 0, sizeof(met));
    }

    inline void initI() {
        for(int i = 0; i < metSize; i++) {
            for(int j = 0; j < metSize; j++) {
                met[i][j] = (i == j);
            }
        }
    }

    Matrix operator * (const Matrix& b) const {
        Matrix ret;
        for(int i = 0; i < metSize; i++) {
            for(int j = 0; j < metSize; j++) {
                ret.met[i][j] = 0;
                for(int k = 0; k < metSize; k++) {
                    ret.met[i][j] = (ret.met[i][j] + (ll)met[i][k] *
                    ↪ b.met[k][j]) % (MOD - 1);
                }
            }
        }
        return ret;
    }
};

inline Matrix quickPow(Matrix a, ll b) {
    Matrix ans;
    ans.initI();
    Matrix base = a;
    while(b) {
        if(b & 1) {
            ans = ans * base;
        }
        base = base * base;
        b >>= 1;
    }
    return ans;
}
```

3.2 线性筛

3.2.1 求素数与欧拉函数

```
int prime[N], phi[N];
int prime_tot;
bool used[N];

void euler(){
    memset(used, true, sizeof(used));
    prime_tot = 0;
    phi[1] = 1;

    for(int i = 2; i < N; i++){
        if(used[i]){
            prime[prime_tot++] = i;
            phi[i] = i - 1;
        }
        for(int j = 0; i*prime[j] < N; j++){
            used[i*prime[j]] = false;
            if(i%prime[j] == 0){
                phi[i*prime[j]] = phi[i] * prime[j];
                break;
            }else{
                phi[i*prime[j]] = phi[i] * (prime[j] - 1);
            }
        }
    }
}
```

3.2.2 求积性函数

```
ll f[N], sum[N];
bool isprime[N];
int  prime[N], tot;
int  cnt[N];

int quickPow(int a, int b){
    int ans = 1, base = a;
    while(b){
        if(b&1)     ans = ans * base;
        base = base * base;
        b >>= 1;
    }
    return ans;
}

void init(){
    memset(isprime, true, sizeof(isprime));
```

```
    tot = 0, f[1] = 1, sum[1] = 1;
    for(int i = 2; i < N; i++){
        if(isprime[i]){
            prime[tot++] = i;
            cnt[i]++;
        }
        for(int j = 0; j < tot && i * prime[j] < N; j++){
            isprime[i * prime[j]] = false;
            if(i % prime[j] == 0){
                cnt[i * prime[j]] = cnt[i] + 1;
                break;
            }else{

                cnt[i * prime[j]] = 1;
            }
        }
    }

    for(int i = 0; i < tot && (ll)prime[i] * prime[i] < N; i++){
        f[prime[i] * prime[i]] = 1;
        ll cur = (ll)prime[i] * prime[i] * prime[i];
        while(cur < N){
            f[cur] = 0;
            cur = cur * prime[i];
        }
    }

    for(int i = 2; i < N; i++){
        if(isprime[i]){
            f[i] = 2;
        }
        for(int j = 0; j < tot && i * prime[j] < N; j++){
            if(i % prime[j] == 0){
                int pk = quickPow(prime[j], cnt[i * prime[j]]);
                f[i * prime[j]] = f[pk] * f[i * prime[j] / pk];
                break;
            }else{
                f[i * prime[j]] = f[i] * f[prime[j]];
            }
        }
        sum[i] = f[i] + sum[i - 1];
    }
}
```

3.2.3 求莫比乌斯函数

```
ll mu[N], sum[N];
bool isprime[N];
```

```cpp
int prime[N], tot;

void init(){
    memset(isprime, true, sizeof(isprime));
    tot = 0, mu[1] = 1, sum[1] = 1;
    for(int i = 2; i < N; i++){
        if(isprime[i]){
            prime[tot++] = i;
            mu[i] = -1;
        }
        for(int j = 0; j < tot && i * prime[j] < N; j++){
            isprime[i * prime[j]] = false;
            if(i % prime[j] == 0){
                mu[i * prime[j]] = 0;
                break;
            }else{
                mu[i * prime[j]] = -mu[i];
            }
        }
        sum[i] = sum[i - 1] + mu[i];
    }
}
```

3.3 中国剩余定理

3.3.1 非互质

```cpp
// x = a[i] mod m[i]

ll a[N], m[N];

ll extgcd(ll a, ll b, ll& x, ll& y){
    ll d = a;
    if(!b){
        x = 1, y = 0;
    }else{
        d = extgcd(b, a%b, y, x);
        y -= a/b*x;
    }
    return d;
}

ll crt(ll n){
    if(n == 1){
        return (m[0] > a[0] ? a[0] : -1);
    }else{
        for(int i = 1; i < n; i++){
            if(m[i] <= a[i])    return -1;
```

```
            ll x, y;
            ll d = extgcd(m[0], m[i], x, y);
            if((a[i] - a[0])%d != 0)    return -1;
            ll t = m[i]/d;
            ll k = ((a[i] - a[0])/d*x%t + t)%t;
            a[0] = m[0] * k + a[0];
            m[0] = m[0] * m[i]/d;
            a[0] = (a[0]%m[0] + m[0])%m[0];
        }
    }
    return a[0];
}
```

3.3.2 互质

```
int m[N] = {23, 28, 33};
int a[N];
int m_tot = 23*28*33;

int extgcd(int a, int b, int& x, int& y){
    int d = a;
    if(!b){
        x = 1, y = 0;
    }else{
        d = extgcd(b, a%b, y, x);
        y -= a/b*x;
    }
    return d;
}

int res(){
    int ans = 0;
    for(int i = 0; i < N; i++){
        int mt = m_tot/m[i];
        int t, y;
        extgcd(mt, m[i], t, y);
        ans += a[i] * t * mt;
    }
    return (ans%m_tot + m_tot)%m_tot;
}
```

3.4 高斯消元

3.4.1 解线性方程组

```
inline bool Guass(int n) {
    for(int i = 1; i <= n; i++) {
```

```
        int r = i;
        for(int k = i + 1; k <= n; k++) {
            if(fabs(G[r][i]) < fabs(G[k][i])) {
                r = k;
            }
        }
        swap(G[i], G[r]);

        if(fabs(G[i][i]) < 1e-4) {
            return false;
        }

        for(int k = i + 1; k <= n; k++) {
            for(int j = n + 1; j >= i; j--) {
                G[k][j] -= G[i][j] / G[i][i] * G[k][i];
            }
        }
    }

    for(int i = n; i >= 1; i--) {
        for(int j = i - 1; j >= 1; j--) {
            G[j][n + 1] -= G[i][n + 1] / G[i][i] * G[j][i];
            G[j][i] = 0;
        }
    }
    return true;
}
```

3.4.2 辗转相除思想的高斯消元

```
inline bool Guass(int n) {
    bool flag = 0;
    for(int i = 1; i <= n; i++) {
        if(G[i][i] == 0) {
            return false;
        }

        for(int k = i + 1; k <= n; k++) {
            while(G[k][i]) {
                int r = G[i][i] / G[k][i];
                for(int j = i; j <= n; j++) {
                    G[i][j] = addMod(G[i][j], MOD - 1LL * G[k][j] * r
                     ↪ % MOD);
                }
                swap(G[k], G[i]);
                flag ^= 1;
            }
        }
```

```
    }
    return flag;
}
```

3.5 牛顿迭代法

3.5.1 求多项式的根

```
inline double fun(const vector<double>& a, double x) {
    double res = 0;
    for(int i = 0; i < (int)a.size(); i++) {
        res += pow(x, i) * a[i];
    }
    return res;
}

inline double newton(const vector<double>& a, const vector<double>&
 ↪  aDiff) {
    double x = -30;
    for(int i = 0; i < 1000000; i++) {
        double nx = x - fun(a, x) / fun(aDiff, x);
        if(dcmp(x - nx) == 0) {
            return nx;
        }
        x = nx;
    }
    return x;
}

inline vector<double> getRoots(vector<double> a) {
    vector<double> res;
    while(a.size() > 1) {
        vector<double> aDiff, na;
        for(int i = 1; i < (int)a.size(); i++) {
            aDiff.push_back(i * a[i]);
        }
        double x0 = newton(a, aDiff);
        res.push_back(x0);
        for(int i = a.size() - 1; i >= 1; i--) {
            a[i - 1] = a[i - 1] + x0 * a[i];
        }

        a.erase(a.begin());
    }
    return res;
}
```

3.5.2 给定整数 N，求最大的满足 $x^2 \leq N$ 的 x

```java
public static BigInteger newton(BigInteger n) {
    BigInteger a = BigInteger.ONE.shiftLeft(n.bitLength() / 2);
    boolean isDec = false;
    while(true) {
        BigInteger b = n.divide(a).add(a).shiftRight(1);
        if (a.compareTo(b) == 0 || a.compareTo(b) < 0 && isDec)
            break;
        isDec = a.compareTo(b) > 0;
        a = b;
    }
    return a;
}
```

3.6 拉格朗日插值法

3.6.1 拉格朗日插值法

```cpp
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
const int MOD = 998244353;
const int N = 2000 + 15;

int x[N], y[N];

int quickPow(int a, int b, int MOD) {
    int ans = 1, base = a;
    while(b) {
        if(b & 1) {
            ans = (ll)ans * base % MOD;
        }
        base = (ll)base * base % MOD;
        b >>= 1;
    }
    return ans;
}

inline int calc(int k, int n) {
    int sum = 0;
    for(int i = 1; i <= n; i++) {
        int cur = y[i];
        for(int j = 1; j <= n; j++) {
            if(i == j) {
                continue;
            }
            cur = (ll)cur * (k - x[j] + MOD) % MOD * quickPow((x[i] -
             ↪ x[j] + MOD) % MOD, MOD - 2, MOD) % MOD;
```

```
        }
        sum = (sum + cur) % MOD;
    }
    return sum;
}

int main() {
    int n, k;
    while(~scanf("%d%d", &n, &k)) {
        for(int i = 1; i <= n; i++) {
            scanf("%d%d", &x[i], &y[i]);
        }
        printf("%d\n", calc(k, n));
    }
}
```

3.6.2 求 i^k 的前缀和

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
const int MOD = 1000000007;
const int N = 50000 + 15;

int inv[N];
int prime[N], tot;
int fi[N];
int ans[N];

inline int quickPow(int a, int b) {
    int ans = 1, base = a;
    while(b) {
        if(b & 1) {
            ans = (ll)ans * base % MOD;
        }
        base = (ll)base * base % MOD;
        b >>= 1;
    }
    return ans;
}

inline void initInv() {
    inv[1] = 1;
    for(int i = 2; i < N; i++) {
        inv[i] = ((-(ll)(MOD / i) * inv[MOD % i]) % MOD + MOD) % MOD;
    }
}
```

```
inline void initFi(int k) {
    memset(fi, 0, sizeof(fi));
    tot = 0;
    fi[1] = 1;
    for(int i = 2; i <= k + 1; i++) {
        if(!fi[i]) {
            fi[i] = quickPow(i, k);
            prime[tot++] = i;
            for(int j = 0; i * prime[j] < N; j++) {
                fi[i * prime[j]] = (ll)fi[i] * fi[prime[j]] % MOD;
                if(i % prime[j] == 0) {
                    break;
                }
            }
        }
    }
    for(int i = 2; i <= k + 1; i++) {
        fi[i] = (fi[i] + fi[i - 1]) % MOD;
    }
}

inline void Lagrange(ll n, int k) {
    int p = 1;
    for(int i = 1; i <= k + 1; i++) {
        ans[i] = (i + k + 1) & 1 ? MOD - fi[i] : fi[i];
        p = (ll)p * ((n - i + 1) % MOD) % MOD * inv[i] % MOD;
        ans[i] = (ll)ans[i] * p % MOD;
    }
    p = 1;
    for(int i = k; i >= 1; i--) {
        p = (ll)p * ((n - i - 1) % MOD) % MOD * inv[k + 1 - i] % MOD;
        ans[i] = (ll)ans[i] * p % MOD;
    }
}

int main() {
    initInv();

    int t;
    scanf("%d", &t);
    while(t--) {
        ll n;
        int k;
        scanf("%lld%d", &n, &k);
        initFi(k);
        Lagrange(n, k);
        int res = accumulate(ans, ans + k + 2, 0, [] (int a, int b)
         ↪  -> int { return (a + b) % MOD; });
```

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

3.7 线性基

3.7.1 求异或和值的第 k 小数

```
#include <cstdio>
#include <algorithm>
#include <cstring>
using namespace std;
const int HIGHEST = 63;
const int N = 1e4 + 15;
typedef long long ll;

ll a[N], b[HIGHEST + 5], c[HIGHEST + 5], pp;

void build(int n) {
    for(int i = 0; i < n; i++) {
        for(int j = HIGHEST; j >= 0; j--) {
            if((a[i] >> j & 1) == 0)    continue;
            if(b[j])    a[i] ^= b[j];
            else {
                b[j] = a[i];
                for(int k = j - 1; k >= 0; k--)         if(b[k] &&
                ↪  (b[j] >> k & 1))   b[j] ^= b[k];
                for(int k = j + 1; k <= HIGHEST; k++)   if(b[k] >> j
                ↪  & 1)             b[k] ^= b[j];
                break;
            }
        }
    }
}

int main() {
    int t, csn = 1;
    scanf("%d", &t);
    while(t--) {
        memset(b, 0, sizeof(b));
        int n;
        scanf("%d", &n);
        for(int i = 0; i < n; i++)      scanf("%lld", &a[i]);
        build(n);

        pp = 0;
        for(int j = 0; j <= HIGHEST; j++) {
            if(b[j])    c[pp++] = b[j];
```

```
        }

        printf("Case #%d:\n", csn++);
        int q;
        scanf("%d", &q);
        while(q--) {
            ll k;
            scanf("%lld", &k);
            if(pp < n)      k--;
            if(k == 0) {
                puts("0");
            } else {
                ll xorsum = 0;
                for(int i = 0; i <= HIGHEST && k; i++, k >>= 1) {
                    if(i >= pp) {
                        xorsum = -1;
                        break;
                    }
                    if(k&1)     xorsum ^= c[i];
                }
                printf("%lld\n", xorsum);
            }
        }
    }
}
```

3.8 欧拉降幂

3.8.1 计算 $a^{a^{a^{\cdots}}}$

```
ll quickPow(int a, int b, int p) {
    int ans = 1, base = a;
    while(b) {
        if(b & 1) {
            ans = (ll)ans * base % p;
        }
        base = (ll)base * base % p;
        b >>= 1;
    }
    return ans;
}

ll quickPow(int a, int b) {
    ll ans = 1, base = a;
    while(b) {
        if(b & 1) {
            ans = min((ll)ans * base, (ll)1e7);
        }
```

```
        base = min((ll)base * base, (ll)1e7);
        b >>= 1;
    }
    return ans;
}

pair<int, ll> fun(int a, int b, int p) {
    if(b == 0) {
        return make_pair(a % p, a);
    }
    if(p == 1) {
        auto ret = make_pair(0, (ll)1e7);
        if(a == 1) {
            ret.second = a;
        } else if(b <= 10) {
            ret.second = a;
            for(int i = 0; i < b; i++) {
                ret.second = quickPow(ret.second, a);
            }
        }
        return ret;
    }

    auto ele = fun(a, b - 1, phi[p]);
    if(ele.second < phi[p]) {
        ele.first = quickPow(a, ele.first, p);
        ele.second = quickPow(a, ele.second);
    } else {
        ele.first = quickPow(a, ele.first % phi[p] + phi[p], p);
        ele.second = quickPow(a, ele.second);
    }
    return ele;
}
```

3.9 FFT 与 NTT

3.9.1 FFT

```
const double PI = acos(-1.0);
struct Complex {
  double x, y;
  Complex(double _x = 0.0, double _y = 0.0) {
    x = _x;
    y = _y;
  }
  Complex operator-(const Complex &b) const {
    return Complex(x - b.x, y - b.y);
  }
```

```
  Complex operator+(const Complex &b) const {
    return Complex(x + b.x, y + b.y);
  }
  Complex operator*(const Complex &b) const {
    return Complex(x * b.x - y * b.y, x * b.y + y * b.x);
  }
};

void change(Complex y[], int len) {
  int i, j, k;
  for (int i = 1, j = len / 2; i < len - 1; i++) {
    if (i < j) swap(y[i], y[j]);
    k = len / 2;
    while (j >= k) {
      j = j - k;
      k = k / 2;
    }
    if (j < k) j += k;
  }
}

void fft(Complex y[], int len, int on) {
  change(y, len);
  for (int h = 2; h <= len; h <<= 1) {
    Complex wn(cos(2 * PI / h), sin(on * 2 * PI / h));
    for (int j = 0; j < len; j += h) {
      Complex w(1, 0);
      for (int k = j; k < j + h / 2; k++) {
        Complex u = y[k];
        Complex t = w * y[k + h / 2];
        y[k] = u + t;
        y[k + h / 2] = u - t;
        w = w * wn;
      }
    }
  }
  if (on == -1) {
    for (int i = 0; i < len; i++) {
      y[i].x /= len;
    }
  }
}

const int MAXN = 200020;
Complex x1[MAXN], x2[MAXN];
char str1[MAXN / 2], str2[MAXN / 2];
int sum[MAXN];
```

```cpp
int main() {
  while (scanf("%s%s", str1, str2) == 2) {
    int len1 = strlen(str1);
    int len2 = strlen(str2);
    int len = 1;
    while (len < len1 * 2 || len < len2 * 2) len <<= 1;
    for (int i = 0; i < len1; i++) x1[i] = Complex(str1[len1 - 1 - i] - '0', 0);
    for (int i = len1; i < len; i++) x1[i] = Complex(0, 0);
    for (int i = 0; i < len2; i++) x2[i] = Complex(str2[len2 - 1 - i] - '0', 0);
    for (int i = len2; i < len; i++) x2[i] = Complex(0, 0);
    fft(x1, len, 1);
    fft(x2, len, 1);
    for (int i = 0; i < len; i++) x1[i] = x1[i] * x2[i];
    fft(x1, len, -1);
    for (int i = 0; i < len; i++) sum[i] = int(x1[i].x + 0.5);
    for (int i = 0; i < len; i++) {
      sum[i + 1] += sum[i] / 10;
      sum[i] %= 10;
    }
    len = len1 + len2 - 1;
    while (sum[len] == 0 && len > 0) len--;
    for (int i = len; i >= 0; i--) printf("%c", sum[i] + '0');
    printf("\n");
  }
  return 0;
}
```

3.9.2 NTT

```cpp
inline int addMod(int a, int b) {
    return a + b >= MOD ? a + b - MOD : a + b;
}

inline int quickPow(int a, int b) {
    int ans = 1, base = a;
    while(b) {
        if(b & 1) {
            ans = (ll)ans * base % MOD;
        }
        base = (ll)base * base % MOD;
        b >>= 1;
    }
    return ans;
}

void change(vector<int>& y, int len) {
```

```
    for(int i = 1, j = len / 2; i < len - 1; i++) {
        if(i < j) {
            swap(y[i], y[j]);
        }
        int k = len / 2;
        while(j >= k) {
            j = j - k;
            k = k / 2;
        }
        if(j < k) {
            j += k;
        }
    }
}

void ntt(vector<int>& y, int len, int on) {
    change(y, len);
    for(int h = 2; h <= len; h <<= 1) {
        int gn = quickPow(3, (MOD - 1) / h);
        for (int j = 0; j < len; j += h) {
            int g = 1;
            for (int k = j; k < j + h / 2; k++) {
                int u = y[k];
                int t = (ll)g * y[k + h / 2] % MOD;
                y[k] = addMod(u, t);
                y[k + h / 2] = addMod(u, MOD - t);
                g = (ll)g * gn % MOD;
            }
        }
    }
    if(on == -1) {
        reverse(y.begin() + 1, y.begin() + len);
        for(int i = 0, inv = quickPow(len, MOD - 2); i < len; i++) {
            y[i] = (ll)y[i] * inv % MOD;
        }
    }
}

inline void mul(vector<int> a, vector<int> b, vector<int>& res, int
↪   alen, int blen) {
    int len = rpow2[alen + blen];
    ntt(a, len, 1);
    ntt(b, len, 1);
    for(int i = 0; i < len; i++) {
        res[i] = (ll)a[i] * b[i] % MOD;
    }
    ntt(res, len, -1);
```

```
}
```

3.10 生成函数

3.10.1 普通型

```
// 有无数种硬币, 为 2, 4, 8, 16, ... 各两个, 问组成 n 的方案数
// (1 + x^2 + x^4)(1 + x^4 + x^8)...
void solve(){
    int lim = 2;
    memset(c1, 0, sizeof(c1));
    memset(c2, 0, sizeof(c2));
    c1[0] = c1[1] = c1[2] = 1;

    for(int i = 2; i < N; i <<= 1){
        for(int k = 0; k <= (i << 1); k += i){
            for(int j = 0; j <= lim && j + k < N; j++){
                c2[j + k] += c1[j];
            }
            lim += k;
        }
        for(int j = 0; j <= lim && j < N; j++){
            c1[j] = c2[j];
            c2[j] = 0;
        }
    }
}
```

3.11 BSGS

```
int bsgs(int a, int b) {
    unordered_map<int, int> mp;
    int sqrtP = ceil(sqrt(MOD));
    for(int y = 0, sum = b; y <= sqrtP; y++, sum = 1LL * sum * a %
    ↪  MOD) {
        mp[sum] = y;
    }

    int z = quickPow(a, sqrtP);
    for(int x = 1, sum = z; x <= sqrtP; x++, sum = 1LL * sum * z %
    ↪  MOD) {
        if(mp.count(sum)) {
            return x * sqrtP - mp[sum];
        }
    }
    return -1;
}
```

3.12 自适应辛普森积分

```
double a, b, c, d;

inline double f(double x) {
    return (c * x + d) / (a * x + b);
}

inline double simpson(double l, double r) {
    double mid = (l + r) / 2;
    return (r - l) * (f(l) + f(r) + 4 * f(mid)) / 6;
}

inline double calc(double l, double r, double eps, double ans) {
    double mid = (l + r) / 2;
    double lres = simpson(l, mid), rres = simpson(mid, r);
    if(fabs(lres + rres - ans) <= 15 * eps) {
        return lres + rres + (lres + rres - ans) / 15;
    }
    return calc(l, mid, eps / 2, lres) + calc(mid, r, eps / 2, rres);
}

int main() {
    double l, r;
    while(~scanf("%lf%lf%lf%lf%lf%lf", &a, &b, &c, &d, &l, &r)) {
        printf("%.6f\n", calc(l, r, 1e-6, simpson(l, r)));
    }
}
```

3.13 扩展欧几里德

```
ll extgcd(ll a, ll b, ll& x, ll& y){
    ll d = a;
    if(!b){
        x = 1, y = 0;
    }else{
        d = extgcd(b, a%b, y, x);
        y -= (a/b) * x;
    }
    return d;
}
```

3.14 反素数

```
void dfs(int depth, ull cnt, ull num, ull& upper, ull& max_num, ull&
↪ ans){  //传入的 n 作为上界
    if(cnt > max_num || (cnt == max_num && ans > num)){   //因子数大
    ↪ 于 或 因子数相同但值小则更新
```

```
        ans = num;
        max_num = cnt;
    }
    for(ull i = 1; i <= prime_nums[depth - 1]; i++){
        if(num > upper/prime[depth])     break;
        prime_nums[depth] = i;
        num*=prime[depth];
        dfs(depth + 1, cnt*(i + 1), num, upper, max_num, ans);
    }
    prime_nums[depth] = 0;
}
```

3.15 康托展开

```
// 康托展开，从末位为第 1 位，首位为第 n 位
// X = a[n]*(n-1)! + a[n - 1]*(n-2)! + ... + a[1]*0!

const int fac[] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320};

int cantor(int a[N]){
    int sum = 0;
    for(int i = 0; i < N; i++){
        int cnt = 0;
        for(int j = i + 1; j < N; j++){
            if(a[j] < a[i])     cnt++;      //后面的数比 a[i] 小的有几
            ↪ 个
        }
        sum += cnt*fac[N - i - 1];          //sum += a[n] * (n - 1)!
    }
    return sum;
}

void inv_cantor(int num, int a[]){
    bool used[N + 1] = {0};
    for(int i = 0; i < N; i++){
        int cnt = num/fac[N - i - 1];   //比 a[i] 小且未用过的数有
        ↪ cnt 个
        num %= fac[N - i - 1];

        int j;
        for(j = 0; j < N; j++){         //回推该位数字是几
            if(!used[j]){
                if(cnt == 0)    break;
                cnt--;
            }
        }
        a[i] = j;
        used[j] = true;
```

```
    }
}
```

3.16 杜教筛

3.16.1 求积性函数前缀和

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<ll, ll> pii;
const int N = 3e6 + 5;

bool used[1005];
bool isprime[N];
int tot;
int prime[N];
int mu[N], phi[N];
ll arr_sum_mu[N], arr_sum_phi[N];
ll mp_mu[1005], mp_phi[1005];

pii getAns(int n, int lim) {
    if(n < N)              return make_pair(arr_sum_mu[n],
     ↪  arr_sum_phi[n]);
    if(used[lim / n]) {
        return make_pair(mp_mu[lim / n], mp_phi[lim / n]);
    }

    used[lim / n] = true;
    mp_mu[lim / n] = 1;
    mp_phi[lim / n] = (ll)n * ((ll)n + 1) / 2;
    for(ll l = 2, r; l <= n && l > 0; l = r + 1) {
        r = n / (n / l);

        pair<ll, ll> tmp = getAns(n / l, lim);
        ll sum_g = (ll)r - l + 1;
        mp_mu[lim / n] -= sum_g * tmp.first;
        mp_phi[lim / n] -= sum_g * tmp.second;
    }
    return make_pair(mp_mu[lim / n], mp_phi[lim / n]);
}

inline void init() {
    memset(isprime, true, sizeof(isprime));
    isprime[0] = isprime[1] = false;
    mu[1] = 1, phi[1] = 1;
    arr_sum_mu[1] = 1, arr_sum_phi[1] = 1;
    tot = 0;
```

```cpp
    for(int i = 2; i < N; i++) {
        if(isprime[i]) {
            prime[tot++] = i;
            phi[i] = i - 1;
            mu[i] = -1;
        }

        for(int j = 0; j < tot && prime[j] * i < N; j++) {
            isprime[i * prime[j]] = false;
            if(i % prime[j] == 0) {
                phi[i * prime[j]] = phi[i] * prime[j];
                mu[i * prime[j]] = 0;
                break;
            }

            phi[i * prime[j]] = phi[i] * (prime[j] - 1);
            mu[i * prime[j]] = -mu[i];
        }

        arr_sum_mu[i] = arr_sum_mu[i - 1] + mu[i];
        arr_sum_phi[i] = arr_sum_phi[i - 1] + phi[i];
    }
}

int main() {
    init();

    int t;
    scanf("%d", &t);
    while(t--) {
        memset(used, false, sizeof(used));
        int n;
        scanf("%d", &n);
        pair<ll, ll> tmp = getAns(n, n);
        printf("%lld %lld\n", tmp.second, tmp.first);
    }
}
```

3.17 Min_25 筛

3.17.1 求积性函数前缀和

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = (int)1e6 + 15;
const ll MOD = (ll)1e9 + 7;
```

```
const ll inv6 = 166666668;
bool isNotPrime[N];
int prime[N], tot;
int sum1[N], sum2[N], g1[N], g2[N];
ll w[N];
int idx1[N], idx2[N];
int gTot;
int sqrtN;

inline void init(int n) {
    tot = 0;
    for(int i = 2; i <= n; i++) {
        if(!isNotPrime[i]) {
            prime[++tot] = i;
            sum1[tot] = (sum1[tot - 1] + i) % MOD;
            sum2[tot] = (sum2[tot - 1] + (ll)i * i) % MOD;
        }
        for(int j = 1; i * prime[j] <= n; j++) {
            isNotPrime[i * prime[j]] = true;
            if(i % prime[j] == 0) {
                break;
            }
        }
    }
}

inline void initG(ll n) {
    // calculate g1[gTot] = 1 + 2 + 3 + ...
    // calculate g2[gTot] = 1^2 + 2^2 + 3^3 + ...
    gTot = 0;
    for(ll i = 1, r; i <= n; i = r + 1) {
        w[++gTot] = n / i;
        r = n / (n / i);
        if(n / i <= sqrtN) {
            idx1[n / i] = gTot;
        } else {
            idx2[n / (n / i)] = gTot;
        }

        ll x = w[gTot] % MOD;
        g1[gTot] = (x * (x + 1) / 2 + MOD - 1) % MOD;
        g2[gTot] = (x * (x + 1) % MOD * (2 * x + 1) % MOD * inv6 +
         ↪  MOD - 1) % MOD;
    }
    // enumerate j, then enumerate n
    // g(i, j - 1) can be calculated before g(n, j)
    for(int i = 1; i <= tot; i++) {
```

```
        for(int j = 1; j <= gTot && (ll)prime[i] * prime[i] <= w[j]; j++) {
            ll k = w[j] / prime[i] <= sqrtN ? idx1[w[j] / prime[i]] : idx2[n / (w[j] / prime[i])];
            g1[j] = (g1[j] - (ll)prime[i] * (g1[k] - sum1[i - 1] + MOD) % MOD + MOD) % MOD;
            g2[j] = (g2[j] - (ll)prime[i] * prime[i] % MOD * (g2[k] - sum2[i - 1] + MOD) % MOD + MOD) % MOD;
        }
    }
}

inline ll calcS(ll x, int j, ll n) {
    if(prime[j] >= x) {
        return 0;
    }
    ll k = x <= sqrtN ? idx1[x] : idx2[n / x];
    ll ans = ((ll)g2[k] - g1[k] - (sum2[j] - sum1[j]) + 2LL * MOD) % MOD;
    for(int i = j + 1; i <= tot && (ll)prime[i] * prime[i] <= x; i++) {
        for(ll e = 1, pie = prime[i]; pie <= x; pie *= prime[i], e++) {
            ll xx = pie % MOD;
            ans = (ans + xx * (xx - 1) % MOD * (calcS(x / pie, i, n) + (e != 1))) % MOD;
        }
    }
    return ans;
}

int main() {
    ll n;
    while(~scanf("%lld", &n)) {
        sqrtN = sqrt(n);
        init(sqrtN);
        initG(n);
        printf("%lld\n", (calcS(n, 0, n) + 1) % MOD);
    }
}
```

4 树

4.1 树状数组

```
ll tree[N];
```

```
inline int lowbit(int x) {
    return x & -x;
}

inline ll query(int x) {
    ll ret = 0;
    for(; x > 0; x -= lowbit(x)) {
        ret += tree[x];
    }
    return ret;
}

inline void update(int x, ll val) {
    for(; x < N; x += lowbit(x)) {
        tree[x] += val;
    }
}
```

4.2 线段树

4.2.1 线段树合并

```
int merge(int o1, int o2, int fa, int l, int r){
      if(o1 == 0 || o2 == 0)   return o1 ^ o2;
      if(l == r){
          ans[fa] = max(ans[fa], l);
          return o1;
      }
      int m = (l + r) >> 1;
      ls[o1] = merge(ls[o1], ls[o2], fa, lson);
      rs[o1] = merge(rs[o1], rs[o2], fa, rson);
      return o1;
}
```

4.3 主席树

4.3.1 主席树

```
void build(int& o, int l, int r) {  //建立空树
    o = tot++;
    sum[o] = 0;
    if(l == r)  return;
    int m = (l + r) >> 1;
    build(ls[o], lson);
    build(rs[o], rson);
}

void update(int& o, int pre, int pos, int l, int r) {
```

```
    o = tot++;
    ls[o] = ls[pre];
    rs[o] = rs[pre];
    sum[o] = sum[pre] + 1;
    if(l == r)  return;

    int m = (l + r) >> 1;
    if(pos <= m) {
        update(ls[o], ls[pre], pos, lson);
    } else {
        update(rs[o], rs[pre], pos, rson);
    }
}

int query(int o, int qr, int l, int r) {
    if(r <= qr) {
        return sum[o];
    }
    int m = (l + r) >> 1;
    int ans = query(ls[o], qr, lson);
    if(m < qr) {
        ans += query(rs[o], qr, rson);
    }
    return ans;
}
```

4.3.2 在 $[x_1, x_2]$ 中 lower_bound(y)

```
int sum[N * 40], ls[N * 40], rs[N * 40], root[N], a[N];
int tot;

inline void build(int& x, int l, int r) {
    x = tot++;
    sum[x] = 0;
    if(l == r) {
        return;
    }
    int m = (l + r) >> 1;
    build(ls[x], lson);
    build(rs[x], rson);
}

inline void update(int& x, int pre, int pos, int l, int r) {
    x = tot++;
    ls[x] = ls[pre], rs[x] = rs[pre];
    sum[x] = sum[pre] + 1;
    if(l == r) {
        return;
```

```
    }
    int m = (l + r) >> 1;
    if(pos <= m) {
        update(ls[x], ls[pre], pos, lson);
    } else {
        update(rs[x], rs[pre], pos, rson);
    }
}

inline int querySum(int x, int pre, int pos, int l, int r) {
    if(r <= pos) {
        return sum[x] - sum[pre];
    }
    int m = (l + r) >> 1;
    int ret = querySum(ls[x], ls[pre], pos, lson);
    if(m < pos) {
        ret += querySum(rs[x], rs[pre], pos, rson);
    }
    return ret;
}

inline int findK(int x, int pre, int k, int l, int r) {
    if(l == r) {
        return l;
    }
    int tmp = sum[ls[x]] - sum[ls[pre]];
    int m = (l + r) >> 1;
    if(tmp >= k) {
        return findK(ls[x], ls[pre], k, lson);
    } else {
        return findK(rs[x], rs[pre], k - tmp, rson);
    }
}
```

4.4 平衡树

4.4.1 Treap

```
struct Treap{
    int key[N], weight[N], sz[N], pre[N], ch[N][2];
    int tot, root;

    void newNode(int& x, int pkey){
        x = ++tot;
        key[x]    = pkey;
        weight[x] = rand();
        ch[x][0] = ch[x][1] = pre[x] = 0;
        sz[x] = 1;
```

```
    }

    void pushUp(int x){
        sz[x] = sz[ch[x][0]] + sz[ch[x][1]] + 1;
    }

    void rotate(int x, int p){
        int y = pre[x], z = pre[y];
        ch[y][p^1] = ch[x][p];
        pre[ch[x][p]] = y;
        pre[x] = z;
        if(z)   ch[z][ch[z][1] == y] = x;
        else    root = x;
        ch[x][p] = y;
        pre[y] = x;
        pushUp(y);
        pushUp(x);
    }

    void insert(int x, int pkey, int fa = 0){
        if(x == 0){
            newNode(x, pkey);
            if(fa)  ch[fa][!(pkey < key[fa])] = x;
            pre[x] = fa;
            return;
        }
        insert(ch[x][!(pkey < key[x])], pkey, x);
        pushUp(x);
        int y = ch[x][!(pkey < key[x])];
        if(weight[y] > weight[x])  rotate(y, (key[y] < key[x]));
    }

    void init(){
        tot = 0, root = 1;
        insert(0, inf);
        insert(root, -inf);
    }

    void query(int x, int pkey, int& ans){
        if(x == 0)     return;
        if(key[x] <= pkey){
            ans = max(ans, key[x]);
            query(ch[x][1], pkey, ans);
        }else{
            query(ch[x][0], pkey, ans);
        }
    }
};
```

```
Treap tp;
```

4.4.2 Splay

```
struct SplayTree{
    int ch[N][2], pre[N];
    ll  sum[N], lzy[N], key[N], val[N], sz[N];
    int root, tot;

    void color(int o, ll delta){
        lzy[o] += delta;
        sum[o] += sz[o] * delta;
        val[o] += delta;
    }

    void pushUp(int o){
        sz[o] = 1, sum[o] = val[o];
        if(~ch[o][0]){
            sz[o] += sz[ch[o][0]];
            sum[o] += sum[ch[o][0]];
        }
        if(~ch[o][1]){
            sz[o] += sz[ch[o][1]];
            sum[o] += sum[ch[o][1]];
        }
    }

    void pushDown(int o){
        if(lzy[o]){
            if(~ch[o][0])    color(ch[o][0], lzy[o]);
            if(~ch[o][1])    color(ch[o][1], lzy[o]);
            lzy[o] = 0;
        }
    }

    void rotate(int x, int p){
        int y = pre[x], z = pre[y];
        pushDown(y), pushDown(x);
        ch[y][p^1] = ch[x][p];
        pre[ch[x][p]] = y;
        pre[x] = z;
        if(~z)   ch[z][ch[z][1] == y] = x;
        ch[x][p] = y;
        pre[y] = x;
        pushUp(y), pushUp(x);
    }
```

```cpp
void splay(int x, int goal){
    while(pre[x] != goal){
        if(pre[pre[x]] == goal)     rotate(x, ch[pre[x]][0] ==
        ↪  x);
        else{
            int y = pre[x], z = pre[y];
            int p = (ch[z][0] == y);
            if(ch[y][p^1] == x)     rotate(y, p), rotate(x, p);
            else                    rotate(x, p^1), rotate(x, p);
        }
    }

    if(goal == -1)          root = x;
    else                    pushUp(goal);
}

void newNode(int& o, int pkey, ll pval){
    o = tot++;
    ch[o][0] = ch[o][1] = pre[o] = -1;
    lzy[o] = 0;
    key[o] = pkey;
    sum[o] = val[o] = pval;
    sz[o] = 1;
}

void insert(int& o, int pkey, ll pval, int fa){
    if(o == -1){
        newNode(o, pkey, pval);
        pre[o] = fa;
        splay(o, -1);
        return;
    }
    if(key[o] == pkey){
        pushDown(o);
        splay(o, -1);
        return;
    }

    pushDown(o);
    if(pkey < key[o])    insert(ch[o][0], pkey, pval, o);
    else                 insert(ch[o][1], pkey, pval, o);
}

void init(){
    root = -1, tot = 0;
    insert(root, inf, 0, -1);
    insert(root, -inf, 0, -1);
}
```

```
    int findPrev(int o, int pkey){
        if(o == -1)  return -1;
        if(key[o] < pkey){
            int ret = findPrev(ch[o][1], pkey);
            return ret == -1 ? o : ret;
        }else{
            return findPrev(ch[o][0], pkey);
        }
    }

    int findSucc(int o, int pkey){
        if(o == -1)  return -1;
        if(key[o] > pkey){
            int ret = findSucc(ch[o][0], pkey);
            return ret == -1 ? o : ret;
        }else{
            return findSucc(ch[o][1], pkey);
        }
    }

    void splayLR(int lkey, int rkey){
        int l = findPrev(root, lkey), r = findSucc(root, rkey);
        splay(l, -1);
        splay(r, root);
    }

    ll query(int lkey, int rkey){
        splayLR(lkey, rkey);
        return sum[ch[ch[root][1]][0]];
    }

    void changeInterval(int lkey, int rkey, ll pval){
        splayLR(lkey, rkey);
        color(ch[ch[root][1]][0], pval);
    }

    void delInterval(int lkey, int rkey){
        splayLR(lkey, rkey);
        ch[ch[root][1]][0] = -1;
        pushUp(ch[root][1]);
    }
};

SplayTree spt;
```

4.4.3 可持久化 FHQ Treap

```
int ch[N << 7][2], w[N << 7], pri[N << 7], sz[N << 7];
int root[N];
int tot;

inline int mrand() {
    static int seed = 123456;
    return seed = (1LL * seed * 2333 + 1234567891) % 998244353;
}

inline int newNode(int v) {
    int x = ++tot;
    pri[x] = mrand();
    sz[x] = 1;
    w[x] = v;
    return x;
}

inline int copyNode(int v) {
    if(!v) {
        return 0;
    }
    int x = ++tot;
    pri[x] = pri[v];
    sz[x] = sz[v];
    w[x] = w[v];
    ch[x][0] = ch[v][0];
    ch[x][1] = ch[v][1];
    return x;
}


inline void pushUp(int x) {
    sz[x] = sz[ch[x][0]] + sz[ch[x][1]] + 1;
}

inline int merge(int x, int y) {
    if(!x || !y) {
        return x | y;
    }
    if(pri[x] < pri[y]) {
        ch[x][1] = merge(ch[x][1], y);
        pushUp(x);
        return x;
    } else {
        ch[y][0] = merge(x, ch[y][0]);
        pushUp(y);
```

```
        return y;
    }
}

inline void split(int rt, int k, int& x, int& y) {
    if(!rt) {
        x = y = 0;
        return;
    }
    rt = copyNode(rt);
    if(w[rt] <= k) {
        x = rt;
        split(ch[rt][1], k, ch[x][1], y);
    } else {
        y = rt;
        split(ch[rt][0], k, x, ch[y][0]);
    }
    pushUp(rt);
}

inline int kth(int rt, int k) {
    if(sz[ch[rt][0]] + 1 == k) {
        return rt;
    } else if(sz[ch[rt][0]] >= k) {
        return kth(ch[rt][0], k);
    } else {
        return kth(ch[rt][1], k - sz[ch[rt][0]] - 1);
    }
}

int main() {
    root[0] = merge(newNode(-2147483647), newNode(2147483647));

    int m;
    scanf("%d", &m);
    for(int i = 1; i <= m; i++) {
        int op, v, x;
        scanf("%d%d%d", &v, &op, &x);
        if(op == 1) {
            int l, r;
            split(root[v], x, l, r);
            root[i] = merge(l, merge(newNode(x), r));
        } else if(op == 2) {
            int p, q, r;
            split(root[v], x, q, r);
            split(q, x - 1, p, q);
            q = merge(ch[q][0], ch[q][1]);
            root[i] = merge(p, merge(q, r));
```

```
        }
        // ...
    }
    return 0;
}
```

4.4.4 带 pushdown 的可持久化 FHQ Treap

```
int ch[N << 7][2], w[N << 7], pri[N << 7], sz[N << 7], f[N << 7];
ll sum[N << 7];
int root[N];
int tot;

inline int mrand() {
    static int seed = 123456;
    return seed = (1LL * seed * 2333 + 1234567891) % 998244353;
}

inline int newNode(int v) {
    int x = ++tot;
    pri[x] = mrand();
    sz[x] = 1;
    sum[x] = w[x] = v;
    return x;
}

inline int copyNode(int v) {
    if(!v) {
        return 0;
    }
    int x = ++tot;
    sum[x] = sum[v];
    pri[x] = pri[v];
    sz[x] = sz[v];
    w[x] = w[v];
    ch[x][0] = ch[v][0];
    ch[x][1] = ch[v][1];
    f[x] = f[v];
    return x;
}

inline void pushUp(int x) {
    sz[x] = sz[ch[x][0]] + sz[ch[x][1]] + 1;
    sum[x] = sum[ch[x][0]] + sum[ch[x][1]] + w[x];
}

inline void pushDown(int x) {
    if(f[x]) {
```

```
        ch[x][0] = copyNode(ch[x][0]);
        ch[x][1] = copyNode(ch[x][1]);
        swap(ch[x][0], ch[x][1]);
        f[ch[x][0]] ^= 1;
        f[ch[x][1]] ^= 1;
        f[x] = 0;
    }
}

inline int merge(int x, int y) {
    if(!x || !y) {
        return x | y;
    }
    if(pri[x] < pri[y]) {
        pushDown(x);
        ch[x][1] = merge(ch[x][1], y);
        pushUp(x);
        return x;
    } else {
        pushDown(y);
        ch[y][0] = merge(x, ch[y][0]);
        pushUp(y);
        return y;
    }
}

inline void split(int rt, int k, int& x, int& y) {
    if(!rt) {
        x = y = 0;
        return;
    }
    pushDown(rt);
    rt = copyNode(rt);
    if(sz[ch[rt][0]] < k) {
        x = rt;
        split(ch[rt][1], k - sz[ch[rt][0]] - 1, ch[x][1], y);
    } else {
        y = rt;
        split(ch[rt][0], k, x, ch[y][0]);
    }
    pushUp(rt);
}
```

4.5 LCA

4.5.1 倍增法

```
void dfs(int u, int pre){
    for(int i = head[u]; ~i; i = e[i].nxt){
        int v = e[i].v;
        if(v == pre)    continue;
        dpt[v] = d[u] + 1;
        d[v] = d[u] + e[i].w;
        fa[v][0] = u;
        dfs(v, u);
    }
}

void getFa(int n){
    for(int j = 1; j <= BASE; j++){
        for(int i = 1; i <= n; i++){

            if(fa[i][j - 1] == -1)  continue;
            fa[i][j] = fa[fa[i][j - 1]][j - 1];
        }
    }
}

int getLca(int u, int v){
    if(dpt[u] > dpt[v])     swap(u, v);
    for(int j = BASE; j >= 0; j--){
        if(fa[v][j] == -1 || dpt[fa[v][j]] < dpt[u])    continue;
        v = fa[v][j];
    }
    if(u == v)  return u;
    for(int j = BASE; j >= 0; j--){
        if(fa[u][j] == -1 || fa[v][j] == -1 || fa[u][j] == fa[v][j])
         ↪ continue;
        u = fa[u][j], v = fa[v][j];
    }
    return fa[u][0];
}
```

4.5.2 Tarjan

```
void tarjan(int u, int pre, int q){
    for(int i = head[u]; ~i; i = e[i].nxt){
        int v = e[i].v;
        if(v == pre)    continue;
        d[v] = d[u] + e[i].w;
        tarjan(v, u, q);
        merge(u, v);
```

```
        used[v] = true;
    }
    for(int i = qhead[u]; ~i; i = qe[i].nxt){
        int v = qe[i].v;
        if(used[v])     pa[qe[i].w] = find(v);
    }
}
```

4.5.3 RMQ

```
const int N = 40000 + 5;
struct edge{
    int v, w, nxt;
};
edge e[N << 1];
int head[N], tot;
int dp[N << 1][21], mp[N << 1], pos[N], d[N], dfn, totDp;

inline void init(){
    memset(head, -1, sizeof(head));
    d[1] = tot = dfn = totDp = 0;
}

inline void addEdge(int u, int v, int w){
    e[tot] = edge{v, w, head[u]};
    head[u] = tot++;
}

void dfs(int u, int pre){
    int curDfn = ++dfn;
    dp[++totDp][0] = curDfn;
    mp[dfn] = u;
    pos[u] = totDp;
    for(int i = head[u]; ~i; i = e[i].nxt){
        int v = e[i].v;
        if(v == pre) {
            continue;
        }
        d[v] = d[u] + e[i].w;

        dfs(v, u);

        dp[++totDp][0] = curDfn;
    }
}

void rmqInit(){
    for(int j = 1; (1 << j) <= totDp; j++){
```

```
        for(int i = 1; i + (1 << j) - 1 <= totDp; i++){
            dp[i][j] = min(dp[i][j - 1], dp[i + (1 << (j - 1))][j -
             ↪  1]);
        }
    }
}

int getLca(int u, int v){
    int l = pos[u], r = pos[v];
    if(l > r) {
        swap(l, r);
    }

    int j = (int)log2(r - l + 1);
    return mp[min(dp[l][j], dp[r - (1 << j) + 1][j])];
}
```

4.6 带权并查集

```
int find(int x){
    if(ft[x] == x)  return x;
    int xft = ft[x];
    ft[x] = find(ft[x]);
    sum_below[x] += sum_below[xft];
    return ft[x];
}

void merge(int a, int b){
    int ra = find(a), rb = find(b);
    if(ra != rb){
        ft[rb] = ra;
        sum_below[rb] = sum_tot[ra];
        sum_tot[ra] += sum_tot[rb];
    }
}
```

4.7 DFS 序

```
void getDfsClock(int u, int pre, int n){
    l[u] = ++dfs_clock;
    for(int i = head[u]; ~i; i = e[i].nxt){
        int v = e[i].v;
        if(v == pre)    continue;
        getDfsClock(v, u, n);
    }
    r[u] = dfs_clock;
}
```

4.8 树的直径

```
void dfs1(int u, int d, int& ansu, int& ansd) {
    if(d > ansd) {
        ansd = d;
        ansu = u;
    }
    used[u] = true;
    for(int i = head[u]; ~i; i = e[i].nxt) {
        int v = e[i].v;
        if(used[v]) {
            continue;
        }

        dfs1(v, d + e[i].w, ansu, ansd);
    }
}

void dfs2(int u, int pre, int d, int& ansd) {
    ansd = max(ansd, d);
    used[u] = true;
    for(int i = head[u]; ~i; i = e[i].nxt) {
        int v = e[i].v;
        if(used[v]) {
            continue;
        }

        dfs2(v, u, d + e[i].w, ansd);
    }
}
```

4.9 树链剖分

```
inline void pushUp(int rt){
    seg[rt] = max(seg[rt << 1], seg[rt << 1 | 1]);
}

void update(int pos, int val, int l, int r, int rt){
    if(l == r){
        seg[rt] = val;
        return;
    }
    int m = (l + r) >> 1;
    if(pos <= m)    update(pos, val, lson);
    else            update(pos, val, rson);
    pushUp(rt);
}
```

```
int query(int ql, int qr, int l, int r, int rt){
    if(ql <= l && r <= qr){
        return seg[rt];
    }
    int ans = 0, m = (l + r) >> 1;
    if(ql <= m)     ans = max(ans, query(ql, qr, lson));
    if(m < qr)      ans = max(ans, query(ql, qr, rson));
    return ans;
}

void dfs(int u){
    son[u] = -1, sz[u] = 1;
    for(int i = head[u]; ~i; i = e[i].nxt){
        int v = e[i].v;
        if(v == fa[u])  continue;
        fa[v] = u;
        preew[v] = e[i].w;
        dpt[v] = dpt[u] + 1;
        dfs(v);
        sz[u] += sz[v];
        if(son[u] == -1 || sz[son[u]] < sz[v])  son[u] = v;
    }
}

void buildTree(int u, int rt, int n){
    mp[u] = ++mptot, top[u] = rt;
    update(mp[u], preew[u], 1, n, 1);
    if(~son[u])     buildTree(son[u], rt, n);
    for(int i = head[u]; ~i; i = e[i].nxt){
        int v = e[i].v;
        if(v == fa[u] || v == son[u])   continue;
        buildTree(v, v, n);
    }
}

void solveUpdate(int idx, int val){
    int v = dpt[input[idx][0]] > dpt[input[idx][1]] ? input[idx][0] :
    ↪   input[idx][1];
    update(mp[v], val, 1, mptot, 1);
}

int solveQuery(int u, int v, int n){
    int fu = top[u], fv = top[v], ret = 0;
    while(fu != fv){
        if(dpt[fu] < dpt[fv]){
            swap(fu, fv);
            swap(u, v);
        }
```

```
        ret = max(ret, query(mp[fu], mp[u], 1, n, 1));
        u = fa[fu], fu = top[u];
    }
    if(u == v)  return ret;
    if(dpt[u] > dpt[v])     swap(u, v);
    // 最后是 mp[u] + 1 而不是 mp[u] 是因为维护的是父边，故 mp[u] 不
    ↪ 在 (u,v) 路径上
    return max(ret, query(mp[u] + 1, mp[v], 1, n, 1));
}
```

4.10 ODT

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

struct Node {
    int l, r;
    mutable int val;
    inline int len() const {
        return r - l + 1;
    }
    bool operator < (const Node& b) const {
        return l < b.l;
    }
};

set<Node> st;

inline set<Node>::iterator split(int pos) {
    auto it = st.lower_bound(Node{pos, 0, 0});
    if(it != st.end() && it->l == pos) {
        return it;
    }
    --it;
    ll v = it->val;
    int l = it->l, r = it->r;
    st.erase(it);
    st.insert(Node{l, pos - 1, v});
    return st.insert(Node{pos, r, v}).first;
}

inline void assignVal(int l, int r, int v) {
    auto itr = split(r + 1), itl = split(l);
    st.erase(itl, itr);
    st.insert(Node{l, r, v});
}
```

```cpp
inline void reverse(int l, int r) {
    auto itr = split(r + 1), itl = split(l);
    for(auto it = itl; it != itr; it++) {
        it->val ^= 1;
    }
}

inline int sum(int l, int r) {
    auto itr = split(r + 1), itl = split(l);
    int res = 0;
    for(auto it = itl; it != itr; it++) {
        res += (it->val == 1) * it->len();
    }
    return res;
}


inline int calc(int l, int r) {
    auto itr = split(r + 1), itl = split(l);
    int res = 0, cnt = 0;
    for(auto it = itl; it != itr; it++) {
        if(it->val) {
            cnt += it->len();
        } else {
            cnt = 0;
        }
        res = max(res, cnt);
    }
    return res;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    int n, m;
    while(cin >> n >> m) {
        st.clear();
        for(int i = 1, val; i <= n; i++) {
            cin >> val;
            st.emplace(Node{i, i, val});
        }

        while(m--) {
            // ...
    }
```

```
}
```

4.11 点分治

4.11.1 点分治

```
void getRoot(int u, int pre, int size, int& rt){
    sum[u] = 1, maxsum[u] = 0;
    for(int i = head[u]; ~i; i = e[i].nxt){
        int v = e[i].v;
        if(used[v] || v == pre)     continue;
        getRoot(v, u, size, rt);
        sum[u] += sum[v];
        maxsum[u] = max(maxsum[u], sum[v]);
    }
    maxsum[u] = max(maxsum[u], size - sum[u]);
    if(rt == -1 || maxsum[u] < maxsum[rt])  rt = u;
}

void getDis(int u, int pre, int w){
    d[pd++] = w;
    for(int i = head[u]; ~i; i = e[i].nxt){
        int v = e[i].v;
        if(used[v] || v == pre)     continue;
        getDis(v, u, w + e[i].w);
    }
}

int calc(int u, int w, int k){
    pd = 0;
    getDis(u, -1, w);
    sort(d, d + pd);

    int ret = 0;
    int head = 0, tail = pd - 1;
    while(head < tail){
        while(d[head] + d[tail] >= k && head < tail){
            if(d[head] + d[tail] == k)  ret++;
            tail--;
        }
        head++;
    }
    return ret;
}

int dfs(int u, int n, int k){
    int rt = -1;
    getRoot(u, -1, n, rt);
```

```
    used[u] = true;
    int ans = calc(u, 0, k);
    for(int i = head[u]; ~i; i = e[i].nxt){
        int v = e[i].v;
        if(used[v])     continue;
        ans -= calc(v, e[i].w, k);
        ans += dfs(v, sum[v], k);
    }
    return ans;
}
```

4.11.2 动态点分治

```
#include <cstdio>
#include <iostream>
#include <cstring>
using namespace std;
const int N = 100100;
const int inf = 0x3f3f3f3f;
typedef long long ll;

struct edge{
    int v, w, nxt;
};

int head[N], rthead[N], tot, rttot;
int sum[N], par[N];
int dpt[N], d[N], fa[N][20], maxsum[N];
bool used[N];
edge e[N << 1];
edge rte[N << 1];

ll ans[N][2], val[N];

inline void init(){
    tot = rttot = 1;
}

inline int read(){
    char ch = getchar(); int x = 0, f = 1;
    while(ch < '0' || ch > '9') {if(ch == '-') f = -1; ch =
    ↪  getchar();}
    while('0' <= ch && ch <= '9') {x = x * 10 + ch - '0'; ch =
    ↪  getchar();}
    return x * f;
}
```

```cpp
inline void addEdge(int u, int v, int w){
    e[tot] = edge{v, w, head[u]};
    head[u] = tot++;
}

inline void addRootEdge(int u, int v, int w){
    rte[rttot] = edge{v, w, rthead[u]};
    rthead[u] = rttot++;
}

void initLCA(int u, int pre){
    fa[u][0] = pre;
    for(int j = 1; j <= 19; j++){
        if(fa[u][j - 1] == 0)   break;
        fa[u][j] = fa[fa[u][j - 1]][j - 1];
    }
    for(int i = head[u]; i; i = e[i].nxt){
        int v = e[i].v;
        if(v == pre)     continue;
        dpt[v] = dpt[u] + 1;
        d[v] = d[u] + e[i].w;
        initLCA(v, u);
    }
}

inline int lca(int u, int v) {
    if(dpt[u] < dpt[v])     swap(u, v);
    int tmp = dpt[u] - dpt[v];
    for(int k = 0, j = 1; j <= tmp; j <<= 1, k++){
        if(tmp & j)    u = fa[u][k];
    }
    while(u != v) {
        int j = 0;
        while(fa[u][j] != fa[v][j]) j++;
        if(j)   j--;
        u = fa[u][j], v = fa[v][j];
    }
    return u;
}

int getDist(int u, int v){
    return d[u] + d[v] - (d[lca(u, v)] << 1);
}

void getRoot(int u, int pre, int size, int& rt){
    sum[u] = 1, maxsum[u] = 0;
    for(int i = head[u]; i; i = e[i].nxt){
        int v = e[i].v;
```

```
        if(v == pre || used[v])     continue;
        getRoot(v, u, size, rt);
        sum[u] += sum[v];
        maxsum[u] = max(maxsum[u], sum[v]);
    }
    maxsum[u] = max(maxsum[u], size - sum[u]);
    if(rt == 0 || maxsum[u] < maxsum[rt])     rt = u;
}

void initTree(int u){
    used[u] = true;
    for(int i = head[u]; i; i = e[i].nxt){
        int v = e[i].v;
        if(used[v])     continue;
        int rt = 0;
        getRoot(v, u, sum[v], rt);
        par[rt] = u;
        addRootEdge(u, v, rt);
        initTree(rt);
    }
}

ll query(int u){
    ll ret = ans[u][0];
    for(int i = u; par[i]; i = par[i]){
        int d = getDist(u, par[i]);
        ret += ans[par[i]][0];
        ret -= ans[i][1];
        ret += (ll)d * (val[par[i]] - val[i]);
    }
    return ret;
}

void update(int u, int x){
    val[u] += x;
    for(int i = u; par[i]; i = par[i]){
        int d = getDist(u, par[i]);
        ans[par[i]][0] += (ll)x * d;
        ans[i][1] += (ll)x * d;
        val[par[i]] += x;
    }
}

ll work(int u){
    ll ret = query(u);
    for(int i = rthead[u]; i; i = rte[i].nxt){
        int v = rte[i].v;
        ll tmp = query(v);
```

```
        if(tmp < ret){
            ret = min(ret, work(rte[i].w));
        }
    }
    return ret;
}


int main(){
    int n, q;
    while(~scanf("%d%d", &n, &q)){
        init();
        for(int i = 1; i <= n - 1; i++){
            int u = read(), v = read(), w = read();
            addEdge(u, v, w);
            addEdge(v, u, w);
        }
        initLCA(1, 0);
        int rt = 0;
        getRoot(1, 0, n, rt);
        par[rt] = 0;
        initTree(rt);
        while(q--){
            int u = read(), x = read();
            update(u, x);
            printf("%lld\n", work(rt));
        }
    }
    return 0;
}
```

4.12 LCT

4.12.1 LCT

```
struct LinkCutTree{
    int fa[N], ch[N][2], sum[N], lzy[N], lzy_col[N];
    int stk[N];
    int col[N], lcol[N], rcol[N];

    inline bool nRoot(int x){
        return ch[fa[x]][0] == x || ch[fa[x]][1] == x;
    }

    void pushUp(int x){
        sum[x] = sum[lson] + sum[rson] + 1, lcol[x] = rcol[x] =
         ↪  col[x];
        if(lson){
```

```
        lcol[x] = lcol[lson];
        if(rcol[lson] == col[x])    sum[x]--;
    }
    if(rson){
        rcol[x] = rcol[rson];
        if(lcol[rson] == col[x])    sum[x]--;
    }
}

void pushR(int x){
    swap(lson, rson);
    swap(lcol[x], rcol[x]);
    lzy[x] ^= 1;
}

void updateColor(int x, int c){
    sum[x] = lzy_col[x] = 1;
    col[x] = lcol[x] = rcol[x] = c;
}

void pushDown(int x){
    if(lzy[x]){
        if(lson)    pushR(lson);
        if(rson)    pushR(rson);
        lzy[x] = 0;
    }
    if(lzy_col[x]){
        if(lson)    updateColor(lson, col[x]);
        if(rson)    updateColor(rson, col[x]);
        lzy_col[x] = 0;
    }
}

void rotate(int x){
    int y = fa[x], z = fa[y];
    int p = (ch[y][1] == x), w = ch[x][p^1];
    if(nRoot(y))    ch[z][ch[z][1] == y] = x;
    ch[x][p^1] = y, ch[y][p] = w;
    if(w)   fa[w] = y;
    fa[y] = x, fa[x] = z;
    pushUp(y);
}

void splay(int x){
    int pstk = 0, y = x;
    for(y = x; nRoot(y); y = fa[y]){
        stk[++pstk] = y;
    }
```

```
        stk[++pstk] = y;
        while(pstk)     pushDown(stk[pstk--]);

        while(nRoot(x)){
            int y = fa[x], z = fa[y];
            if(nRoot(y))     rotate((ch[y][0] == x) ^ (ch[z][0] == y)
             ↪  ? x : y);
            rotate(x);
        }
        pushUp(x);
    }

    void access(int x){
        for(int y = 0; x; y = x, x = fa[x]){
            splay(x);
            rson = y;
            pushUp(x);
        }
    }

    void makeRoot(int x){
        access(x);
        splay(x);
        pushR(x);
    }

    int findRoot(int x){
        access(x);
        splay(x);
        while(lson){
            pushDown(x);
            x = lson;
        }
        return x;
    }

    void split(int x, int y){
        makeRoot(x);
        access(y);
        splay(y);
    }

    void link(int x, int y){
        makeRoot(x);
        if(findRoot(y) != x)     fa[x] = y;
    }

    void cut(int x, int y){
```

```
        makeRoot(x);
        if(findRoot(y) == x && fa[x] == y && !rson){
            fa[x] = ch[y][0] = 0;
            pushUp(y);
        }
    }
};
```

4.12.2 可维护子树信息的 LCT

```
struct LinkCutTree{
    int fa[N], ch[N][2], sum[N], val[N], lzy[N];
    int stk[N];
    int si[N];

    inline bool nRoot(int x){
        return ch[fa[x]][0] == x || ch[fa[x]][1] == x;
    }

    void pushUp(int x){
        sum[x] = sum[lson] + sum[rson] + si[x] + 1;
    }

    void pushR(int x){
        swap(lson, rson);
        lzy[x] ^= 1;
    }

    void pushDown(int x){
        if(lzy[x]){
            if(lson)    pushR(lson);
            if(rson)    pushR(rson);
            lzy[x] = 0;
        }
    }

    void rotate(int x){
        int y = fa[x], z = fa[y];
        int p = (ch[y][1] == x), w = ch[x][p^1];
        if(nRoot(y))    ch[z][ch[z][1] == y] = x;
        ch[x][p^1] = y, ch[y][p] = w;
        if(w)   fa[w] = y;
        fa[y] = x, fa[x] = z;
        pushUp(y);
    }

    void splay(int x){
        int pstk = 0, y = x;
```

```
    for(y = x; nRoot(y); y = fa[y]){
        stk[++pstk] = y;
    }
    stk[++pstk] = y;
    while(pstk)     pushDown(stk[pstk--]);

    while(nRoot(x)){
        int y = fa[x], z = fa[y];
        if(nRoot(y))     rotate((ch[y][0] == x) ^ (ch[z][0] == y)
         ↪  ? x : y);
        rotate(x);
    }
    pushUp(x);
}

void access(int x){
    for(int y = 0; x; y = x, x = fa[x]){
        splay(x);
        si[x] += sum[rson];
        si[x] -= sum[y];
        rson = y;
        pushUp(x);
    }
}

void makeRoot(int x){
    access(x);
    splay(x);
    pushR(x);
}

int findRoot(int x){
    access(x);
    splay(x);
    while(lson){
        pushDown(x);
        x = lson;
    }
    return x;
}

void split(int x, int y){
    makeRoot(x);
    access(y);
    splay(y);
}

void link(int x, int y){
```

```
        makeRoot(x);
        if(findRoot(y) != x){
            si[y] += sum[x];
            fa[x] = y;
        }
    }

    void cut(int x, int y){
        makeRoot(x);
        if(findRoot(y) == x && fa[x] == y && !rson){
            fa[x] = ch[y][0] = 0;
            pushUp(y);
        }
    }
};
LinkCutTree lct;
```

4.13 左偏树

4.13.1 左偏树

```
const int N = 100000 + 15;

int ch[N][2], val[N], d[N], ft[N];

inline void init(int n) {
    for(int i = 1; i <= n; i++) {
        ft[i] = i;
        d[i] = 1;
        ch[i][0] = ch[i][1] = 0;
    }
}

inline int find(int x) {
    return ft[x] == x ? x : ft[x] = find(ft[x]);
}

inline int merge(int x, int y) {
    if(!x || !y) {
        return x | y;
    }
    if(val[x] < val[y]) {
        swap(x, y);
    }
    ch[x][1] = merge(ch[x][1], y);
    if(d[ch[x][0]] < d[ch[x][1]]) {
        swap(ch[x][0], ch[x][1]);
    }
```

```
    d[x] = d[ch[x][1]] + 1;
    return x;
}

int main() {
    int n, m;
    while(~scanf("%d", &n)) {
        init(n);
        for(int i = 1; i <= n; i++) {
            scanf("%d", &val[i]);
        }
        scanf("%d", &m);
        while(m--) {
            int u, v;
            scanf("%d%d", &u, &v);
            u = find(u), v = find(v);
            if(u == v) {
                puts("-1");
            } else {
                for(int k = 0; k < 2; k++) {
                    val[u] >>= 1;
                    int x = merge(ch[u][0], ch[u][1]);
                    ch[u][0] = ch[u][1] = 0;
                    int y = merge(u, x);
                    ft[u] = ft[x] = y;
                    swap(u, v);
                }
                u = find(u), v = find(v);
                int x = merge(u, v);
                ft[u] = ft[v] = x;
                printf("%d\n", val[x]);
            }
        }
    }
}
```

4.13.2 带 pushdown 的左偏树

```
inline void pushDown(int x) {
    if (lzyAdd[x] != 0 || lzyMul[x] != 1) {
        for (int k = 0; k < 2; k++) {
            val[ch[x][k]] = lzyMul[x] * val[ch[x][k]] + lzyAdd[x];
            lzyMul[ch[x][k]] *= lzyMul[x];
            lzyAdd[ch[x][k]] = lzyMul[x] * lzyAdd[ch[x][k]] +
             ↪  lzyAdd[x];
        }
        lzyAdd[x] = 0;
        lzyMul[x] = 1;
```

```
    }
}

inline int merge(int x, int y) {
    if (!x || !y) {
        return x | y;
    }
    if (val[x] > val[y]) {
        swap(x, y);
    }
    pushDown(x);
    ch[x][1] = merge(ch[x][1], y);
    if (d[ch[x][0]] < d[ch[x][1]]) {
        swap(ch[x][0], ch[x][1]);
    }
    d[x] = d[ch[x][1]] + 1;
    return x;
}

inline void dfs(int u) {
    int rt = mp[u];
    for (int v : graph[u]) {
        dpt[v] = dpt[u] + 1;
        dfs(v);
        rt = merge(rt, mp[v]);
    }
    // printf("u = %d\n", u);
    while (rt && val[rt] < h[u]) {
        // printf("rt = %d, val = %lld\n", rt, val[rt]);
        cnt[u]++;
        ed[rt] = u;
        pushDown(rt);
        rt = merge(ch[rt][0], ch[rt][1]);
    }
    // printf("rt = %d, val = %lld\n", rt, val[rt]);
    mp[u] = rt;
    if (a[u] == 0) {
        val[rt] += v[u];
        lzyAdd[rt] += v[u];
    } else {
        val[rt] *= v[u];
        lzyMul[rt] *= v[u];
        lzyAdd[rt] *= v[u];
    }
}
```

5 图论

5.1 最短路

5.1.1 SPFA（带 SLF 优化）

```
void spfa(int src){
    que.push_back(src);
    inque[src] = true;
    while(!que.empty()){
        int u = que.front();
        inque[u] = false;
        que.pop_front();
        for(int i = head[u]; ~i; i = e[i].next){
            int v = e[i].v;
            if(d[v] > d[u] + e[i].val){
                d[v] = d[u] + e[i].val;
                if(!inque[v]){
                    inque[v] = true;
                    if(!que.empty() && d[v] <= d[que.front()]) {
                        que.push_front(v);
                    } else {
                        que.push_back(v);
                    }
                }
            }
        }
    }
}
```

5.1.2 Dijkstra

```
void dijkstra(){
    que.push(node(0, 1));
    while(!que.empty()){
        int val = que.top().val;
        int u = que.top().u;
        que.pop();

        if(d[u] < val)  continue;

        for(int i = head[u]; ~i; i = e[i].next){
            int v = e[i].v;
            if(d[v] > d[u] + e[i].val){
                d[v] = d[u] + e[i].val;
                que.push(node(d[v], v));
            }
        }
```

```
    }
}
```

5.2 最小生成树

5.2.1 Prim

```
int prim(int n){
    int ans = 0;
    mincost[1] = 0;
    que.push(node(0, 1));
    while(!que.empty()){
        int u = que.top().u;
        int cost = que.top().cost;
        que.pop();

        if(used[u] || mincost[u] < cost) continue;
        used[u] = true;
        mincost[u] = cost;
        ans += cost;

        for(int v = 1; v <= n; v++){
            if(u == v)  continue;
            if(!used[v] && mincost[v] > G[u][v]){
                mincost[v] = G[u][v];
                que.push(node(G[u][v], v));
            }
        }
    }
    return ans;
}
```

5.2.2 Kruskal

```
int Kruskal(int m){
    int ans = 0;
    sort(e, e + m);
    for(int i = 0; i < m; i++){
        int p = find(e[i].u), q = find(e[i].v);
        if(p != q){
            ans += e[i].val;
            merge(p, q);
        }
    }
    return ans;
}
```

5.3 拓扑排序 - BFS

```
void topoSort(int n){
    for(int i = 1; i <= n; i++){
        if(!idg[i])    que.push(i);
    }
    while(!que.empty()){
        int u = que.front();
        que.pop();
        ans.push_back(u);
        for(int i = head[u]; ~i; i = e[i].next){
            int v = e[i].v;
            if((--idg[v]) == 0)    que.push(v);
        }
    }
}
```

5.4 极大团

```
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;
typedef long long ll;
const int N = 128 + 15;

int some[N][N], all[N][N], none[N][N];
int G[N][N];

int dfs(int d, int an, int sn, int nn, int sum) {
    if(!sn && !nn) {
        return sum;
    }

    int ans = 0;
    int u = some[d][0];
    for(int i = 0; i < sn; i++) {
        int v = some[d][i];
        if(G[u][v]) {
            continue;
        }

        for(int j = 0; j < an; j++) {
            all[d + 1][j] = all[d][j];
        }
        all[d + 1][an] = v;

        int tsn = 0, tnn = 0;
```

```cpp
        for(int j = 0; j < sn; j++) {
            if(!G[v][some[d][j]]) {
                continue;
            }
            some[d + 1][tsn++] = some[d][j];
        }
        for(int j = 0; j < nn; j++) {
            if(!G[v][none[d][j]]) {
                continue;
            }
            none[d + 1][tnn++] = none[d][j];
        }

        ans = max(ans, dfs(d + 1, an + 1, tsn, tnn, sum + 1));
        some[d][i] = 0;
        none[d][nn++] = v;
    }
    return ans;
}

inline int solve(int n) {
    for(int i = 1; i <= n; i++) {
        some[0][i - 1] = i;
    }
    return dfs(0, 0, n, 0, 0);
}

int main() {
    int n;
    while(~scanf("%d", &n) && n) {
        for(int i = 1; i <= n; i++) {
            for(int j = 1; j <= n; j++) {
                scanf("%d", &G[i][j]);
            }
        }

        printf("%d\n", solve(n));
    }
}
```

5.5 最大团

```cpp
#include <bits/stdc++.h>
const int N = 50 + 15;

int G[N][N];
int num[N];  //维护 V[i] ... v[n] 的最大团
```

```cpp
bool dfs(int adj[], int an, int cnt, int& ans) {
    if(an == 0) {
        if(cnt > ans) {
            ans = cnt;
            return true;
        }
        return false;
    }

    int tmp[N];
    for(int i = 0; i < an; i++) {
        //剪枝
        if(cnt + an - i <= ans || cnt + num[adj[i]] <= ans) {
            return false;
        }

        int k = 0;
        for(int j = i + 1; j < an; j++) {
            if(!G[adj[i]][adj[j]]) {
                continue;
            }
            tmp[k++] = adj[j];
        }

        if(dfs(tmp, k, cnt + 1, ans)) {
            return true;
        }
    }
    return false;
}

int solve(int n) {
    int adj[N];
    int ans = 0;
    //从后往前枚举, 利用 num[i]
    for(int i = n; i >= 1; i--) {
        int k = 0;
        for(int j = i + 1; j <= n; j++) {
            if(!G[i][j]) {
                continue;
            }
            adj[k++] = j;
        }
        dfs(adj, k, 1, ans);
        num[i] = ans;
    }
    return ans;
}
```

```
int main() {
    int n;
    while(~scanf("%d", &n) && n) {
        for(int i = 1; i <= n; i++) {
            for(int j = 1; j <= n; j++) {
                scanf("%d", &G[i][j]);
            }
        }

        printf("%d\n", solve(n));
    }
}
```

5.6 最大流

5.6.1 ISAP

```
struct edge{
    int v, val, nxt;
};

int  d[N], head[N], gap[N], cur[N], pre[N];
edge e[M << 1];
int  tot;

inline void init(){
    memset(head, -1, sizeof(head));
    memset(d, 0, sizeof(d));
    memset(gap, 0, sizeof(gap));
    tot = 0;
}

void addEdge(int u, int v, int val){
    e[tot].v   = v;
    e[tot].val = val;
    e[tot].nxt = head[u];
    head[u]    = tot++;
}

int ISAP(int n, int src, int des){
    memcpy(cur, head, sizeof(head));
    int sum = 0;
    int u = pre[src] = src;
    gap[0] = n;

    while(d[src] < n){
        if(u == des){
```

```cpp
            int aug = inf, v;
            for(u = pre[des], v = des; v != src; v = u, u = pre[u])
             ↪  aug = min(aug, e[cur[u]].val);
            for(u = pre[des], v = des; v != src; v = u, u = pre[u]){
                e[cur[u]].val   -= aug;
                e[cur[u]^1].val += aug;
            }
            sum += aug;
            continue;
        }

        bool flag = false;
        for(int& i = cur[u]; ~i; i = e[i].nxt){
            int v = e[i].v;
            if(d[u] == d[v] + 1 && e[i].val){
                pre[v] = u;
                u = v;
                flag = true;
                break;
            }
        }

        if(!flag){
            int mind = n;
            for(int i = head[u]; ~i; i = e[i].nxt){
                int v = e[i].v;
                if(e[i].val && d[v] < mind){
                    mind = d[v];
                    cur[u] = i;
                }
            }
            if((--gap[d[u]]) == 0)      break;
            d[u] = mind + 1;
            gap[d[u]]++;
            u = pre[u];
        }
    }
    return sum;
}
```

5.6.2 HLPP

```cpp
const int N = (int)1200 + 15;
const int M = (int)120000 + 15;
const int inf = 0x3f3f3f3f;

struct edge {
    int v, nxt, flow;
```

```
};

edge e[M * 2];
int head[N], tot, ht[N], ex[N], gap[N << 1];

struct cmp {
    inline bool operator () (int a, int b) const {
        return ht[a] < ht[b];
    }
};
priority_queue<int, vector<int>, cmp> que;
bool inq[N];

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

inline void addEdge(int u, int v, int flow) {
    e[tot] = edge{v, head[u], flow};
    head[u] = tot++;
    e[tot] = edge{u, head[v], 0};
    head[v] = tot++;
}

inline bool bfsInit(int src, int des, int n) {
    memset(ht, 0x3f, sizeof(ht));
    queue<int> que;
    que.push(des);
    ht[des] = 0;
    while(!que.empty()) {
        int u = que.front();
        que.pop();
        for(int i = head[u]; ~i; i = e[i].nxt) {
            int v = e[i].v;
            if(e[i ^ 1].flow && ht[v] > ht[u] + 1) {
                ht[v] = ht[u] + 1;
                que.push(v);
            }
        }
    }
    return ht[src] != inf;
}

inline bool push(int u, int src, int des) {
    for(int i = head[u]; ~i; i = e[i].nxt) {
        int v = e[i].v, w = e[i].flow;
        if(!w || ht[u] != ht[v] + 1) {
```

```
            continue;
        }
        int k = min(w, ex[u]);
        ex[u] -= k;
        ex[v] += k;
        e[i].flow -= k;
        e[i ^ 1].flow += k;
        if(v != src && v != des && !inq[v]) {
            que.push(v);
            inq[v] = true;
        }
        if(!ex[u]) {
            return false;
        }
    }
    return true;
}

inline void relabel(int u) {
    ht[u] = inf;
    for(int i = head[u]; ~i; i = e[i].nxt) {
        if(e[i].flow) {
            ht[u] = min(ht[u], ht[e[i].v] + 1);
        }
    }
}

inline int hlpp(int src, int des, int n) {
    if(!bfsInit(src, des, n)) {
        return 0;
    }
    ht[src] = n;
    memset(gap, 0, sizeof(gap));
    for(int i = 1; i <= n; i++) {
        if(ht[i] != inf) {
            gap[ht[i]]++;
        }
    }
    for(int i = head[src]; ~i; i = e[i].nxt) {
        int v = e[i].v, w = e[i].flow;
        if(!w) {
            continue;
        }
        ex[src] -= w;
        ex[v] += w;
        e[i].flow -= w;
        e[i ^ 1].flow += w;
        if(v != src && v != des && !inq[v]) {
```

```
            que.push(v);
            inq[v] = true;
        }
    }
    while(!que.empty()) {
        int u = que.top();
        que.pop();
        inq[u] = false;
        if(push(u, src, des)) {
            if((--gap[ht[u]]) == 0) {
                for(int v = 1; v <= n; v++) {
                    if(v != src && v != des && ht[v] > ht[u] && ht[v]
                     ↪  < n + 1) {
                        ht[v] = n + 1;
                    }
                }
            }
            relabel(u);
            gap[ht[u]]++;
            que.push(u);
            inq[u] = true;
        }
    }
    return ex[des];
}
```

5.7 费用流

5.7.1 EK+SPFA 算法

```
bool spfa(int src, int des){
    memset(d, 0x3f, sizeof(d));
    memset(inq, false, sizeof(inq));
    d[src] = 0, pre[src] = src;
    que.push(src);
    inq[src] = true;
    while(!que.empty()){
        int u = que.front();
        que.pop();
        inq[u] = false;
        for(int i = head[u]; ~i; i = e[i].next){
            int v = e[i].v;
            if(d[v] > d[u] + e[i].cost && e[i].val){
                d[v] = d[u] + e[i].cost;
                pre[v] = u;
                cur[v] = i;
                if(!inq[v]){
                    que.push(v);
```

```
                    inq[v] = true;
                }
            }
        }
    }
    return d[des] != inf;
}

int solve(int src, int des){
    int ans = 0;
    while(spfa(src, des)){
        int aug = inf;
        for(int v = des; v != src; v = pre[v])     aug = min(aug,
         ↪   e[cur[v]].val);
        for(int v = des; v != src; v = pre[v]){
            e[cur[v]].val -= aug;
            e[cur[v]^1].val += aug;
        }
        ans += d[des] * aug;
    }
    return ans;
}
```

5.7.2 Primal-dual Dijkstra 方法

```
const int N = (int)5000 + 3;
const int M = (int)50000 + 3;

struct Node {
    int u;
    ll d;
    bool operator < (const Node& b) const {
        return d > b.d;
    }
};
struct edge {
    int v, nxt, flow;
    ll cost;
};

int flow[N];
int pre[N], cur[N];
ll dis[N], h[N];
edge e[M * 4];
int head[N], tot;

inline void init() {
    memset(head, -1, sizeof(head));
```

```
    tot = 0;
}

inline void addEdge(int u, int v, int flow, ll cost) {
    e[tot] = edge{v, head[u], flow, cost};
    head[u] = tot++;
    e[tot] = edge{u, head[v], 0, -cost};
    head[v] = tot++;
}

bool dijkstra(int src, int des, int n) {
    priority_queue<Node> que;
    fill(dis + 1, dis + n + 1, 1LL << 30);
    fill(flow + 1, flow + n + 1, 1LL << 30);
    dis[src] = 0, pre[src] = src;
    que.push(Node{src, 0});
    while(!que.empty()) {
        Node ele = que.top();
        que.pop();
        int u = ele.u;
        ll d = ele.d;

        if(d > dis[u]) {
            continue;
        }

        for(int i = head[u]; ~i; i = e[i].nxt) {
            int v = e[i].v;
            if(e[i].flow && dis[v] > dis[u] + e[i].cost + h[u] -
            ↪  h[v]) {
                dis[v] = dis[u] + e[i].cost + h[u] - h[v];
                flow[v] = min(flow[u], e[i].flow);
                pre[v] = u;
                cur[v] = i;
                que.push(Node{v, dis[v]});
            }
        }
    }
    return dis[des] != (1LL << 30);
}

pair<int, ll> solve(int src, int des, int n) {
    int maxFlow = 0;
    ll minCost = 0;
    while(dijkstra(src, des, n)) {
        int aug = flow[des];
        maxFlow += aug;
        minCost += aug * (dis[des] - h[src] + h[des]);
```

```
        for(int u = pre[des], v = des; v != src; v = u, u = pre[u]) {
            e[cur[v]].flow -= aug;
            e[cur[v] ^ 1].flow += aug;
        }
        for(int i = 1; i <= n; i++) {
            h[i] += dis[i];
        }
    }
    return make_pair(maxFlow, minCost);
}
```

5.7.3 带 Primal-dual Dijkstra 的多路增广方法

```
// 时间复杂度为 O(n*m*f)
const int N = (int)5000 + 3;
const int M = (int)50000 + 3;
const int inf = 1LL << 30;

struct Node {
    int u;
    ll d;
    bool operator < (const Node& b) const {
        return d > b.d;
    }
};
struct edge {
    int v, nxt, flow;
    ll cost;
};

int G[N][N];
edge e[M * 4];
int head[N], tot;
int cur[N];
bool used[N];
ll dis[N], h[N];

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

inline void addEdge(int u, int v, int flow, ll cost) {
    e[tot] = edge{v, head[u], flow, cost};
    head[u] = tot++;
    e[tot] = edge{u, head[v], 0, -cost};
    head[v] = tot++;
}
```

```cpp
bool dijkstra(int src, int des, int n) {
    priority_queue<Node> que;
    fill(dis + 1, dis + n + 1, inf);
    memcpy(cur, head, sizeof(head));
    dis[des] = 0;
    que.push(Node{des, 0});
    while(!que.empty()) {
        Node ele = que.top();
        que.pop();
        int u = ele.u;
        ll d = ele.d;

        if(d > dis[u]) {
            continue;
        }

        for(int i = head[u]; ~i; i = e[i].nxt) {
            int v = e[i].v;
            if(e[i^1].flow && dis[v] > dis[u] - e[i].cost + h[u] -
            ↪  h[v]) {
                dis[v] = dis[u] - e[i].cost + h[u] - h[v];
                que.push(Node{v, dis[v]});
            }
        }
    }
    return dis[src] != inf;
}

int dfs(int u, int des, int low, ll& totalCost) {
    if(u == des) {
        return low;
    }
    used[u] = true;
    int sum = 0;
    for(int& i = cur[u]; ~i; i = e[i].nxt) {
        int v = e[i].v;
        ll du = (dis[u] - h[des] + h[u]);
        ll dv = (dis[v] - h[des] + h[v]);
        if(!used[v] && e[i].flow && du - e[i].cost == dv) {
            int aug = dfs(v, des, min(e[i].flow, low - sum),
            ↪  totalCost);
            if(aug) {
                e[i].flow -= aug;
                e[i ^ 1].flow += aug;
                sum += aug;
                totalCost += 1LL * aug * e[i].cost;
            }
```

```
            if(sum == low) {
                break;
            }
        }
    }
    used[u] = false;
    return sum;
}

inline void initH(int src, int des, int n) {
    static bool inq[N];
    fill(dis + 1, dis + n + 1, inf);
    dis[des] = 0;
    queue<int> que;
    que.push(des);
    inq[des] = true;
    while(!que.empty()) {
        int u = que.front();
        que.pop();
        inq[u] = false;
        for(int i = head[u]; ~i; i = e[i].nxt) {
            int v = e[i].v;
            if(e[i^1].flow && dis[v] > dis[u] - e[i].cost) {
                dis[v] = dis[u] - e[i].cost;
                if(!inq[v]) {
                    inq[v] = true;
                    que.push(v);
                }
            }
        }
    }
    for(int i = 1; i <= n; i++) {
        h[i] = dis[i];
    }
}

inline pair<int, ll> solve(int src, int des, int n) {
    initH(src, des, n);    //非负权图可不执行
    pair<int, ll> pr(0, 0LL);
    while(dijkstra(src, des, n)) {
        while(true) {
            int x = dfs(src, des, inf, pr.second);
            pr.first += x;
            if(!x) {
                break;
            }
        }
    }
```

```
    return pr;
}
```

5.8 欧拉路

5.8.1 无向图 - Fluery 算法

```
int getSrc(int n){
    for(int i = 1; i <= n; i++){
        if(dg[i]&1){
            return i;
        }
    }
    return 1;
}

void dfs(int u){
    for(int i = head[u]; ~i; i = e[i].nxt){
        int v = e[i].v;
        if(used[i])    continue;
        used[i] = used[i^1] = true;
        dfs(v);
    }
    ans[pp++] = u;
}
```

5.8.2 有向图

```
int getSrc(int n){
    for(int i = 1; i <= n; i++){
        if(dg[i]&1){
            return i;
        }
    }
    return 1;
}

void dfs(int u){
    for(int i = head[u]; ~i; i = e[i].nxt){
        int v = e[i].v;
        if(used[i])    continue;
        used[i] = used[i^1] = true;
        dfs(v);
    }
    ans[pp++] = u;
}
```

5.9 图的匹配

5.9.1 二分图最大权匹配 - KM 算法 BFS 版

```
const int N = (int)400 + 15;
const ll inf = 1LL << 60;

bool vx[N], vy[N];
ll lx[N], ly[N], slack[N], w[N][N];
int ml[N], mr[N], pre[N];
int que[N];

inline void setCrossRoad(int& v) {
    for(; v; swap(v, mr[pre[v]])) {
        ml[v] = pre[v];
    }
}

inline void bfs(int u, int n) {
    int head = 0, tail = 0;
    que[tail++] = u;
    vx[u] = true;
    while(true) {
        while(head != tail) {
            int u = que[head++];
            for(int v = 1; v <= n; v++) {
                if(vy[v]) {
                    continue;
                }
                ll d = lx[u] + ly[v] - w[u][v];
                if(d > slack[v]) {
                    continue;
                }
                pre[v] = u;
                if(d == 0) {
                    if(!ml[v]) {
                        setCrossRoad(v);
                        return;
                    }
                    vy[v] = vx[ml[v]] = true;
                    que[tail++] = ml[v];
                } else {
                    slack[v] = d;
                }
            }
        }
        ll d = inf;
        int to = 1;
        for(int v = 1; v <= n; v++) {
```

```
            if(!vy[v] && d > slack[v]) {
                d = slack[v];
                to = v;
            }
        }
        for(int i = 1; i <= n; i++) {
            if(vx[i]) {
                lx[i] -= d;
            }
            if(vy[i]) {
                ly[i] += d;
            } else {
                slack[i] -= d;
            }
        }
        if(!ml[to]) {
            setCrossRoad(to);
            return;
        }
        head = tail = 0;
        que[tail++] = ml[to];
        vx[ml[to]] = vy[to] = true;
    }
}

inline ll KM(int n) {
    for(int i = 1; i <= n; i++) {
        ly[i] = 0;
        ml[i] = mr[i] = 0;
        lx[i] = *max_element(w[i] + 1, w[i] + 1 + n);
    }
    for(int i = 1; i <= n; i++) {
        fill(vx, vx + 1 + n, false);
        fill(vy, vy + 1 + n, false);
        fill(slack, slack + 1 + n, inf);
        bfs(i, n);
    }
    return accumulate(lx + 1, lx + 1 + n, 0LL) + accumulate(ly + 1,
     ↪  ly + 1 + n, 0LL);
}

int main() {
    int n, m, k;
    scanf("%d%d%d", &n, &m, &k);
    while(k--) {
        int u, v, val;
        scanf("%d%d%d", &u, &v, &val);
        w[u][v] = val;
```

```
    }
    ll ans = KM(max(n, m));
}
```

5.9.2 一般图最大匹配 - 带花树

```
const int N = (int)500 + 3;
const int M = (int)124750 + 3;
const int inf = 1LL << 30;

struct edge {
    int v, nxt;
};

edge e[M << 1];
int head[N], tot;
int match[N], pre[N], type[N];
int que[N], qhead, qtail;
int ft[N];

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

inline void addEdge(int u, int v) {
    e[tot] = edge{v, head[u]};
    head[u] = tot++;
}

inline int find(int x) {
    return ft[x] == x ? x : ft[x] = find(ft[x]);
}

inline int getLca(int u, int v) {
    static int ss[N], tim;
    tim++;
    while(ss[u] != tim) {
        if(u) {
            ss[u] = tim;
            u = find(pre[match[u]]);
        }
        swap(u, v);
    }
    return u;
}

inline void flower(int x, int y, int p) {
```

```
    while(find(x) != p) {
        pre[x] = y;
        y = match[x];
        ft[x] = ft[y] = p;
        if(type[y] == 1) {
            que[qtail++] = y;
            type[y] = 2;
        }
        x = pre[y];
    }
}

inline bool blossom(int u, int n) {
    qhead = qtail = 0;
    for(int i = 1; i <= n; i++) {
        type[i] = 0;
        ft[i] = i;
    }
    que[qtail++] = u;
    type[u] = 2;
    while(qhead != qtail) {
        u = que[qhead++];
        for(int i = head[u]; ~i; i = e[i].nxt) {
            int v = e[i].v;
            if(type[v] == 0) {
                type[v] = 1;
                pre[v] = u;
                if(!match[v]) {
                    while(u) {
                        u = match[pre[v]];
                        match[v] = pre[v];
                        match[match[v]] = v;
                        v = u;
                    }
                    return true;
                } else {
                    que[qtail++] = match[v];
                    type[match[v]] = 2;
                }
            } else if(type[v] == 2 && find(u) != find(v)) {
                int p = getLca(u, v);
                flower(u, v, p);
                flower(v, u, p);
            }
        }
    }
    return false;
}
```

```cpp
int main() {
    init();
    int n, m, ans = 0;
    scanf("%d%d", &n, &m);
    while(m--) {
        int u, v;
        scanf("%d%d", &u, &v);
        addEdge(u, v);
        addEdge(v, u);
        if(!match[u] && !match[v]) {
            match[v] = u;
            match[u] = v;
            ans++;

        }
    }
    for(int i = 1; i <= n; i++) {
        if(!match[i] && blossom(i, n)) {
            ans++;
        }
    }
    printf("%d\n", ans);
    for(int i = 1; i <= n; i++) {
        printf("%d ", match[i]);
    }
    puts("");
}
```

5.10 连通分量

5.10.1 点双连通分量 - Tarjan

```cpp
int st[N], low[N], cnt[N], nodeChildNums[N];
int dfn;
bool isCut[N], isRoot[N], used[N];

inline void read(int& x) {
    scanf("%d", &x);
}

inline void addEdge(int u, int v) {
    e[tot] = edge{v, head[u]};
    head[u] = tot++;
}

void Tarjan(int u, int pre) {
    st[u] = low[u] = ++dfn;
```

```
    int childNums = 0;
    for(int i = head[u]; ~i; i = e[i].nxt) {
        int v = e[i].v;
        if(st[v] == 0) {
            childNums++;
            Tarjan(v, u);
            low[u] = min(low[u], low[v]);
            if(low[v] >= st[u]) {
                nodeChildNums[u]++;
                isCut[u] = true;
            }
        } else if(v != pre && st[v] < st[u]) {
            low[u] = min(low[u], st[v]);
        }
    }
    if(pre == -1 && childNums == 1) {
        isCut[u] = false;
    }
}

int main() {
    int u, v, csn = 1;
    while(true) {
        tot = dfn = 0;
        for(int i = 1; i < N; i++) {
            head[i] = -1;
            st[i] = nodeChildNums[i] = 0;
            isRoot[i] = isCut[i] = false;
        }

        v = -1;
        while(scanf("%d", &u) && u) {
            scanf("%d", &v);
            addEdge(u, v);
            addEdge(v, u);
        }
        if(v == -1) {
            break;
        }

        if(csn != 1) {
            puts("");
        }
        printf("Network #%d\n", csn++);
        for(int i = 1; i < N; i++) {
            if(st[i] == 0) {
                isRoot[i] = true;
                Tarjan(i, -1);
```

```
            }
        }

        bool flag = true;
        for(int i = 1; i <= 1000; i++) {
            if(isCut[i]) {
                flag = false;
                printf("  SPF node %d leaves %d subnets\n", i,
                 ↪  nodeChildNums[i] + (isRoot[i] == false));
            }
        }
        if(flag) {
            puts("  No SPF nodes");
        }
    }
    return 0;
}
```

5.10.2 强连通分量（有向图）

```
struct edge {
    int u, v, nxt;
};

edge e[N];
int head[N], tot;
bool instk[N];
int mp[N], out[N], cnt[N];
int low[N], st[N], dfn;
stack<int> stk;

inline void read(int& x) {
    scanf("%d", &x);
}

inline void addEdge(int u, int v) {
    e[tot] = edge{u, v, head[u]};
    head[u] = tot++;
}

void Tarjan(int u) {
    low[u] = st[u] = ++dfn;
    stk.push(u);
    instk[u] = true;
    for(int i = head[u]; ~i; i = e[i].nxt) {
        int v = e[i].v;
        if(st[v] == 0) {
            Tarjan(v);
```

```
            low[u] = min(low[u], low[v]);
        } else if(instk[v]){
            low[u] = min(low[u], st[v]);
        }
    }
    if(low[u] == st[u]) {
        while(true) {
            int v = stk.top();
            stk.pop();
            instk[v] = false;
            mp[v] = u;
            cnt[u]++;
            if(u == v) {
                break;
            }
        }
    }
}

int main(){
    int n, m;
    while(~scanf("%d%d", &n, &m)){
        memset(head, -1, sizeof(head));
        dfn = 0;
        for(int i = 1; i <= n; i++) {
            cnt[i] = out[i] = st[i] = 0;
        }

        while(m--){
            int u, v;
            read(u), read(v);
            addEdge(u, v);
        }

        for(int i = 1; i <= n; i++){
            if(st[i] == 0) {
                Tarjan(i);
            }
        }

        for(int i = 0; i < tot; i++){
            int u = e[i].u, v = e[i].v;
            if(mp[u] == mp[v]) {
                continue;   //在同一个连通分量的不统计出度
            }
            out[mp[u]]++;
        }
```

```
        int t = 0, ans = 0;
        for(int i = 1; i <= n; i++){
            if(mp[i] != i)  continue;
            if(out[i] == 0){
                t++;                    //出度为 0 的点的数量
                ans = cnt[i];
            }
        }
        printf("%d\n", t == 1 ? ans : 0);
    }
    return 0;
}
```

5.11 全局最小割

```
const ll inf = 0x3f3f3f3f3f3f3f3fLL;
const int N = 300 + 5;

bool used[N];
int v[N];
ll dis[N], G[N][N];


ll storeWagner(int n) {
    ll res = inf;
    for(int i = 0; i < n; i++) {
        v[i] = i;
    }
    while(n > 1) {
        used[v[0]] = 1 ;
        for(int i = 1; i < n; i++) {
            used[v[i]] = 0 ;
            dis[v[i]] = G[v[0]][v[i]];
        }
        int last = 0;
        for(int i = 1; i < n; i++) {
            int maxs = -1;
            for(int j = 1; j < n; j++) {
                if(used[v[j]] == false && (maxs == -1 || dis[v[j]] >
                ↪  dis[v[maxs]])) {
                    maxs = j;
                }
            }
            used[v[maxs]] = 1;
            if(i == n - 1) {
                res = min(res, dis[v[maxs]]);
                for(int j = 0 ; j < n; j++) {
                    G[v[last]][v[j]] += G[v[maxs]][v[j]];
```

```
                    G[v[j]][v[last]] += G[v[j]][v[maxs]];
                }
                v[maxs] = v[--n];
                break;
            }
            last = maxs;
            for(int j = 1; j < n; j++) {
                if(used[v[j]] == false) {
                    dis[v[j]] += G[v[maxs]][v[j]];
                }
            }
        }
    }
    return res;
}

int main() {
    int n, m, s;
    while(~scanf("%d%d%d", &n, &m, &s) && (n && m && s)) {
        memset(G, 0, sizeof(G));

        while(m--) {
            int u, v, w;
            scanf("%d%d%d", &u, &v, &w);
            G[u - 1][v - 1] += w;
            G[v - 1][u - 1] += w;
        }
        printf("%lld\n", storeWagner(n));
    }
}
```

5.12 k 短路

```
int d[N];

struct Node {
    int u, w;

    bool operator < (const Node& b) const {
        if(d[u] + w != d[b.u] + b.w) {
            return d[u] + w > d[b.u] + b.w;
        }
        return w > b.w;
    }
};

struct edge{
    int v, w, nxt;
```

```cpp
};

struct Graph {
    edge e[N];
    int head[N], tot;

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

    inline void addEdge(int u, int v, int w) {
        e[tot] = edge{v, w, head[u]};
        head[u] = tot++;
    }
};

Graph g, ginv;
bool inque[N];
int cnt[N];

void solveD(int des) {
    memset(d, 0x3f, sizeof(d));
    memset(inque, false, sizeof(inque));
    queue<int> que;

    d[des] = 0;
    que.push(des);
    inque[des] = true;
    while(!que.empty()) {
        int u = que.front();
        que.pop();
        inque[u] = false;

        for(int i = ginv.head[u]; ~i; i = ginv.e[i].nxt) {
            int v = ginv.e[i].v;
            if(d[v] > d[u] + ginv.e[i].w) {
                d[v] = d[u] + ginv.e[i].w;
                if(inque[v] == false) {
                    que.push(v);
                    inque[v] = true;
                }
            }
        }
    }
}

int solve(int src, int des, int k) {
```

```
    memset(cnt, 0, sizeof(cnt));
    priority_queue<Node> que;
    que.push(Node{src, 0});

    while(!que.empty()) {
        int u = que.top().u;
        int w = que.top().w;
        que.pop();

        cnt[u]++;
        if(cnt[u] == k && u == des) {
            return w;
        }

        for(int i = g.head[u]; ~i; i = g.e[i].nxt) {
            int v = g.e[i].v;
            que.push(Node{v, w + g.e[i].w});
        }
    }
    return false;
}
```

5.13 2-SAT - 爆搜法

```
int head[N * 2], tot;
bool mark[N * 2];
int stk[N * 2], pstk;

inline void addEdge(int u, int v) {
    e[tot] = edge{v, head[u]};
    head[u] = tot++;
}

inline void addClause(int x, int xval, int y, int yval) {
    x = x << 1 | xval;
    y = y << 1 | yval;
    addEdge(x^1, y);
    addEdge(y^1, x);
}

bool dfs(int x) {
    if(mark[x^1]) {
        return false;
    }
    if(mark[x]) {
        return true;
    }
    mark[x] = true;
```

```
    stk[pstk++] = x;
    for(int i = head[x]; ~i; i = e[i].nxt) {
        int v = e[i].v;
        if(!dfs(v)) {
            return false;
        }
    }
    return true;
}

bool solve() {
    for(int i = 0; i < 2 * N; i += 2) {
        if(!mark[i] && !mark[i + 1]) {
            pstk = 0;
            if(!dfs(i)) {
                while(pstk > 0) {
                    mark[stk[--pstk]] = false;
                }
                if(!dfs(i + 1)) {
                    return false;
                }
            }
        }
    }
    return true;
}
```

6 计算几何

6.1 半平面交

```
const int N = 1500 + 15;
const double eps = 1e-8;

struct Point{
    double x, y;
    Point() {}
    Point(double x, double y): x(x), y(y) {}
    Point operator - (const Point& b) {
        return Point(x - b.x, y - b.y);
    }
};
typedef Point Vector;

struct Line{
    Point a, b;
    double angle;
```

```
    void getAngle() {angle = atan2(b.y - a.y, b.x - a.x);}
    Line(){}
    Line(Point a, Vector b): a(a), b(b) {}
};

vector<Line> hp;
vector<Point> pt;
vector<Point> ans;
Line que[N];

int dcmp(double x) {
    return x < -eps ? -1 : x > eps;
}

double cross(Vector a, Vector b){
    return a.x * b.y - a.y * b.x;
}

double area(Point a, Point b, Point c) {
    return cross(b - a, c - a);
}

// 点是否在线的右边（不含在线上的情况)
bool isOnLineRight(Line u, Point v){
    return dcmp(cross(u.b - u.a, v - u.a)) < 0;
}

// 按极角顺时针排序
bool cmp(Line u, Line v) {
    int d = dcmp(u.angle - v.angle);
    if(d) return d > 0;
    return dcmp(cross(u.b - u.a, v.b - u.a)) < 0;
}

Point getLineIntersection(Line u, Line v){
    Point ret = u.a;
    double t = ((u.a.x-v.a.x) * (v.a.y-v.b.y)
               -(u.a.y-v.a.y) * (v.a.x-v.b.x))
             / ((u.a.x-u.b.x) * (v.a.y-v.b.y)
               -(u.a.y-u.b.y) * (v.a.x-v.b.x));
    ret.x += (u.b.x-u.a.x) * t, ret.y += (u.b.y-u.a.y) * t;
    return ret;
}

// 判断 l2,l3 的交点时候在 l1 的右边
bool judge(Line l1, Line l2, Line l3) {
    Point p = getLineIntersection(l2, l3);
    return isOnLineRight(l1, p);
```

```
}

void hpi(){//half-plane intersection
    ans.clear();
    sort(hp.begin(), hp.end(), cmp);
    int m = 0;
    // 平行的取第一个，与排序函数的写法有关，反正是取最左边的（严格的
    ↪ 说，应该是向量的左边）
    for(int i = 0; i < hp.size(); i++){
        if(i && dcmp(hp[i].angle - hp[m - 1].angle) == 0)   continue;
        hp[m++] = hp[i];
    }
    hp.erase(hp.begin() + m, hp.end());

    que[1] = hp[0], que[2] = hp[1];
    int head = 1, tail = 2;
    for(int i = 2; i < hp.size(); i++){
        while(head < tail && judge(hp[i], que[tail - 1], que[tail]))
        ↪ tail--;
        while(head < tail && judge(hp[i], que[head + 1], que[head]))
        ↪ head++;
        que[++tail] = hp[i];
    }
    while(head < tail && judge(que[head], que[tail - 1], que[tail]))
    ↪ tail--;
    while(head < tail && judge(que[tail], que[head + 1], que[head]))
    ↪ head++;

    if(tail <= head + 1){ //若半平面交退化为点或线
        return;
    }

    // 为了方便记录，直接把最后一条线放到最前面，避免最后还要保存头尾两
    ↪ 条线的交点
    que[head - 1] = que[tail];
    for(int i = head; i <= tail; i++){
        ans.push_back(getLineIntersection(que[i], que[i - 1]));
    }
}

int main(){
    int t;
    scanf("%d", &t);
    while(t--){
        hp.clear();
        pt.clear();
        int n;
        scanf("%d", &n);
```

```
        for(int i = 0; i < n; i++){
            double x, y;
            scanf("%lf%lf", &x, &y);
            pt.push_back(Point(x, y));
            if(i){
                hp.push_back(Line(pt[i], pt[i - 1]));
                hp[hp.size() - 1].getAngle();
            }
        }
        hp.push_back(Line(pt[0], pt[n - 1]));
        hp[hp.size() - 1].getAngle();
        hpi();

        double res = 0;
        for(int i = 2; i < ans.size(); i++){
            res += area(ans[0], ans[i - 1], ans[i]);
        }
        printf("%.2f\n", fabs(res / 2) + eps);
    }
    return 0;
}
```

6.2 凸包 - Andrew 算法

```
bool used[N];
Point pt[N], resPt[N];
void Andrew(int n, Point* resPt, int& m) {
    memset(used, false, sizeof(used));
    sort(pt + 1, pt + 1 + n);
    pstk = 0;
    stk[++pstk] = 1;
    for(int i = 2; i <= n; i++) {
        while(pstk > 1 && dcmp(cross(pt[stk[pstk]] - pt[stk[pstk -
        ↪ 1]], pt[i] - pt[stk[pstk]])) <= 0) {
            used[stk[pstk--]] = false;
        }
        used[i] = true;
        stk[++pstk] = i;
    }
    int tmp = pstk;
    for(int i = n - 1; i >= 1; i--) {
        if(used[i]) {
            continue;
        }
        while(pstk > tmp && dcmp(cross(pt[stk[pstk]] - pt[stk[pstk -
        ↪ 1]], pt[i] - pt[stk[pstk]])) <= 0) {
            used[stk[pstk--]] = false;
        }
```

```
        used[i] = true;
        stk[++pstk] = i;
    }
    m = pstk;
    for(int i = 1; i <= m; i++) {
        resPt[i] = pt[stk[i]];
    }
}
```

6.3 线段相交

```
struct Point {
    double x, y;
};

int dcmp(double d) {
    if(fabs(d) < eps) return 0;
    return (d > 0) ? 1 : -1;
}

double cross(const Point &A, const Point &B, const Point &C) {
    return (B.x - A.x) * (C.y - A.y) - (B.y - A.y) * (C.x - A.x);
}

int xycmp(double p, double mini, double maxi) {
    return dcmp(p - mini) * dcmp(p - maxi);
}

/*
前提条件: a 在 bc 直线上
返回:
1 表示 a 不在线段 bc 上
0 表示 a 在 b 点或者 c 点上
-1 表示 a 在线段 bc 上
*/
int betweencmp(const Point &a, const Point &b, const Point &c) {
    if(fabs(b.x - c.x) > fabs(b.y - c.y)) return xycmp(a.x, min(b.x,
     ↪ c.x), max(b.x, c.x));
    else return xycmp(a.y, min(b.y, c.y),max(b.y, c.y));
}

//判断线段 ab 是否与 cd 相交, 交点记在 p
//返回值: 0 表示不相交; 1 表示规范相交; 2 表示非规范相交
//p 为交点
int segcross(const Point &a, const Point &b, const Point &c, const
 ↪ Point &d, Point &p) {
    double s1, s2, s3, s4;
    int d1, d2, d3, d4;
```

```
    d1 = dcmp(s1 = cross(a, b, c));
    d2 = dcmp(s2 = cross(a, b, d));
    d3 = dcmp(s3 = cross(c, d, a));
    d4 = dcmp(s4 = cross(c, d, b));

    //判断规范相交：交点不会在端点上
    if((d1 ^ d2) == -2 && (d3 ^ d4) == -2)
    {
        p.x = (c.x * s2 - d.x * s1) / (s2-s1);
        p.y = (c.y * s2 - d.y * s1) / (s2-s1);
        return 1;
    }

    // 判断非规范相交：交点在端点上
    if(d1 == 0 && betweencmp(c, a, b) <= 0) {
        p = c;
        return 2;
    }
    if(d2 == 0 && betweencmp(d, a, b) <= 0) {
        p = d;
        return 2;
    }
    if(d3 == 0 && betweencmp(a, c, d) <= 0) {
        p = a;
        return 2;
    }
    if(d4 == 0 && betweencmp(b, c, d) <= 0) {
        p = d;
        return 2;
    }

    return 0;
}
```

7 数据结构与其他

7.1 单调队列求定长 RMQ

```
while(!que.empty())     que.pop_back();
for(int i = 0; i < n; i++){
    while(!que.empty() && que.back().val > a[i])    que.pop_back();
    que.push_back(Node(a[i], i));
    if(i + 1 - k >= 0){
        while(!que.empty() && que.front().idx < i + 1 - k)
         ↪ que.pop_front();
        ansmin[i + 1 - k] = que.front().val;
```

```
    }
}
```

7.2 单调栈求最小值所在区间

```
for(int i = 0; i < n; i++){
    while(stk1.top().val >= a[i])    stk1.pop();
    l[i] = stk1.top().idx + 1;
    stk1.push(Node(a[i], i));
}

for(int i = n - 1; i >= 0; i--){
    while(stk2.top().val >= a[i])    stk2.pop();
    r[i] = stk2.top().idx - 1;
    stk2.push(Node(a[i], i));
}
```

7.3 模拟退火

```
#include <cstdio>
#include <cstring>
#include <cmath>
#include <ctime>
#include <algorithm>
using namespace std;

const int N = 1000 + 15;

struct Point{
    double x, y;
    int w;
};
Point pt[N];
const int dx[] = {1, 0, -1, 0};
const int dy[] = {0, 1, 0, -1};

inline double sqr(double x){
    return x * x;
}

inline double getDis(const Point& a, const Point& b){
    return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y));
}

double getSum(Point p, int n){
    double ret = 0;
    for(int i = 1; i <= n; i++){
```

```
        ret += (getDis(p, pt[i]) * pt[i].w);
    }
    return ret;
}

void solve(Point& ansu, int n){
    const double delta = 0.998;
    const double eps = 1e-17;
    double tp = 10000;
    double ans = getSum(ansu, n);
    Point u = ansu;
    while(tp > eps){
        Point v = Point{u.x + (rand()*2-RAND_MAX)*tp, u.y +
         ↪  (rand()*2-RAND_MAX)*tp, 1};
        double tmp = getSum(v, n);
        if(tmp < ans){
            ansu = v;
            u = v;
            ans = tmp;
        }else if(exp(-(tmp - ans)/tp) * RAND_MAX > rand()){
            u = v;
        }
        tp *= delta;
    }
}

int main(){
    srand(time(0));
    int n;
    while(~scanf("%d", &n)){
        for(int i = 1; i <= n; i++){
            scanf("%lf%lf%d", &pt[i].x, &pt[i].y, &pt[i].w);
        }
        Point ansu = pt[1];
        solve(ansu, n);
        solve(ansu, n);
        solve(ansu, n);
        printf("%.3f %.3f\n", ansu.x, ansu.y);
    }
    return 0;
}
```

7.4 RMQ

```
void build(int n){
    for(int j = 1; (1 << j) <= n; j++){
        for(int i = 1; i + (1 << j) - 1 <= n; i++){
```

```
            arr_max[i][j] = max(arr_max[i][j - 1], arr_max[i + (1 <<
            ↪  (j - 1))][j - 1]);
            arr_min[i][j] = min(arr_min[i][j - 1], arr_min[i + (1 <<
            ↪  (j - 1))][j - 1]);
        }
    }
}

int query(int l, int r, int p){
    int k = log(r - l + 1)/log(2);
    if(p == 0)  return max(arr_max[l][k], arr_max[r - (1 << k) +
    ↪  1][k]);
    else        return min(arr_min[l][k], arr_min[r - (1 << k) +
    ↪  1][k]);
}
```

7.5 SG 函数

```
void solve(){
    int lim = 2;
    memset(c1, 0, sizeof(c1));
    memset(c2, 0, sizeof(c2));
    c1[0] = c1[1] = c1[2] = 1;

    for(int i = 2; i < N; i <<= 1){
        for(int k = 0; k <= (i << 1); k += i){
            for(int j = 0; j <= lim && j + k < N; j++){
                c2[j + k] += c1[j];
            }
            lim += k;
        }
        for(int j = 0; j <= lim && j < N; j++){
            c1[j] = c2[j];
            c2[j] = 0;
        }
    }
}
```

7.6 悬线法求 01 矩阵

```
int maxx[2];
char G[N][N];
int l[N][N], r[N][N], up[N][N];
bool used[N][N];

int main() {
    int n, m;
```

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

for(int i=1; i<=n; i++) {
    scanf("%s", G[i] + 1);
    for(int j = 1; j <= m; j++) {
        G[i][j] = (G[i][j] == '1');
    }
}

for(int i=1; i<=n; i++)
    for(int j=1; j<=m; j++)
        if(G[i][j])
            up[i][j]=1,r[i][j]=l[i][j]=j;
for(int i=1; i<=n; i++)
    for(int j=2; j<=m; j++)
        if(G[i][j]==1&&G[i][j-1]==1)
            l[i][j]=l[i][j-1];
for(int i=1; i<=n; i++)
    for(int j=m-1; j>0; j--)
        if(G[i][j]==1&&G[i][j+1]==1)
            r[i][j]=r[i][j+1];
for(int i=1; i<=n; i++) {
    for(int j=1; j<=m; j++) {
        if(i>1&&G[i][j]==1&&G[i-1][j]==1) {
            r[i][j]=min(r[i][j],r[i-1][j]);
            l[i][j]=max(l[i][j],l[i-1][j]);
            up[i][j]=up[i-1][j]+1;
        }
    }
}

stack<pair<int, int> > stk;
for(int i = 1; i <= n; i++) {
    for(int j = 1; j <= m; j++) {
        if(!G[i][j]) {
            continue;
        }
        if(used[l[i][j]][r[i][j]]) {
            continue;
        }
        used[l[i][j]][r[i][j]] = true;
        stk.push(make_pair(l[i][j], r[i][j]));

        int res;
        res = (r[i][j]-l[i][j]+1)*up[i][j];
        if(res > maxx[0]) {
            maxx[1] = maxx[0];
            maxx[0] = res;
```

```
            } else if(res > maxx[1]) {
                maxx[1] = res;
            }

            res = max(0, (r[i][j]-l[i][j])*up[i][j]);
            if(res > maxx[0]) {
                maxx[1] = maxx[0];
                maxx[0] = res;
            } else if(res > maxx[1]) {
                maxx[1] = res;
            }

            res = max(0, (r[i][j]-l[i][j])*(up[i][j]-1));
            if(res > maxx[0]) {
                maxx[1] = maxx[0];
                maxx[0] = res;
            } else if(res > maxx[1]) {
                maxx[1] = res;
            }
        }
        while(!stk.empty()) {
            pair<int, int> pr = stk.top();
            stk.pop();
            used[pr.first][pr.second] = false;
        }
    }
    printf("%d\n", maxx[1]);
}
```

7.7 双端队列

```
template <typename T>
struct Deque {
    int head, tail;
    T que[N << 1];

    void clear() {head = tail = N;}
    void push_back(T x) { que[tail++] = x; }
    void push_front(T x) { que[--head] = x; }
    void pop_front() { head++; }
    void pop_back() { tail--; }
    T front() { return que[head]; }
    T back() { return que[tail]; }
    bool empty() { return head == tail; }
};
```

7.8 全排列生成 - 迭代法

```
bool next_permutation(char* s, int len) {
    int x = -1, y = len - 1;
    for(int i = len - 1; i - 1 >= 0; i--) {
        if(s[i - 1] < s[i]) {
            x = i - 1;
            break;
        }
    }

    if(x == -1) {
        return false;
    }

    for(int i = x + 1; i < len; i++) {
        if(s[x] >= s[i]) {
            y = i - 1;
            break;
        }
    }

    swap(s[x], s[y]);
    reverse(s + x + 1, s + len);
    return true;
}
```

7.9 动态 DP

7.9.1 序列上的 ddp

```
#include<bits/stdc++.h>
#define lson l, m, rt << 1
#define rson m + 1, r, rt << 1 | 1
using namespace std;
typedef long long ll;
const int N = (int)5e4 + 15;
const int MOD = 1000000007;

struct Matrix {
    static int n;
    int mat[11][11];

    inline void init() {
        for(int i = 0; i < 11; i++) {
            mat[i][0] = 0;
        }
    }
```

```
};

int Matrix::n;
Matrix m0, seg[N << 2];
bool a[N][11];

inline void mulMatrix(const Matrix& a, const Matrix& b, Matrix& ret)
 ↪  {
    for(int i = 0; i < Matrix::n; i++) {
        for(int j = 0; j < Matrix::n; j++) {
            ret.mat[i][j] = 0;
            for(int k = 0; k < Matrix::n; k++) {
                ret.mat[i][j] = (ret.mat[i][j] + (ll)a.mat[i][k] *
                 ↪  b.mat[k][j] % MOD) % MOD;
            }
        }
    }
}

inline void updateMatrix(int pos, int m, int rt) {
    for(int i = 0; i < m; i++) {
        bool ok = true;
        for(int j = i; j >= 0; j--) {
            ok &= a[pos][j];
            seg[rt].mat[i][j] = ok;
        }
        ok = true;
        for(int j = i; j < m; j++) {
            ok &= a[pos][j];
            seg[rt].mat[i][j] = ok;
        }
    }
}

inline void build(int n, int mm, int l, int r, int rt) {
    if(l == r) {
        int pos = n - l + 1;
        updateMatrix(pos, mm, rt);
        return;
    }
    int m = (l + r) >> 1;
    build(n, mm, lson);
    build(n, mm, rson);
    mulMatrix(seg[rt << 1], seg[rt << 1 | 1], seg[rt]);
}

inline void update(int pos, int y, int n, int mm, int l, int r, int
 ↪  rt) {
```

```
    if(l == r) {
        pos = n - pos + 1;
        a[pos][y] = !a[pos][y];
        updateMatrix(pos, mm, rt);
        return;
    }
    int m = (l + r) >> 1;
    if(pos <= m) {
        update(pos, y, n, mm, lson);
    } else {
        update(pos, y, n, mm, rson);
    }
    mulMatrix(seg[rt << 1], seg[rt << 1 | 1], seg[rt]);
}


int main() {
    int n, m, q;
    while(~scanf("%d%d%d", &n, &m, &q)) {
        Matrix::n = m;
        for(int i = 1, tmp; i <= n; i++) {
            for(int j = 0; j < m; j++) {
                scanf("%1d", &tmp);
                a[i][j] = !tmp;
            }
        }
        Matrix res;
        if(n > 1) {
            build(n, m, 1, n - 1, 1);
        }

        while(q--) {
            int op, x, y;
            scanf("%d%d%d", &op, &x, &y);
            if(op == 1) {
                if(x == 1) {
                    a[x][y - 1] = !a[x][y - 1];
                } else {
                    update(n - x + 1, y - 1, n, m, 1, n - 1, 1);
                }
            } else {
                m0.init();
                x--;
                y--;
                for(int i = x; i >= 0; i--) {
                    if(!a[1][i]) {
                        break;
                    }
```

```
                m0.mat[i][0] = 1;
            }
            for(int i = x; i < m; i++) {
                if(!a[1][i]) {
                    break;
                }
                m0.mat[i][0] = 1;
            }
            if(n > 1) {
                mulMatrix(seg[1], m0, res);
                printf("%d\n", res.mat[y][0]);
            } else {
                printf("%d\n", m0.mat[y][0]);
            }
        }
    }
}
}
```

7.9.2 树上最大权独立集

```
#include<bits/stdc++.h>
#define lson l, m, rt << 1
#define rson m + 1, r, rt << 1 | 1
using namespace std;
const int N = (int)1e5 + 15;
const int inf = 0x3f3f3f3f;

struct Matrix {
    int mat[2][2];

    inline Matrix operator * (const Matrix& b) const {
        Matrix ret;
        for(int i = 0; i < 2; i++) {
            for(int j = 0; j < 2; j++) {
                ret.mat[i][j] = 0;
                for(int k = 0; k < 2; k++) {
                    ret.mat[i][j] = max(ret.mat[i][j], mat[i][k] +
                    ↪ b.mat[k][j]);
                }
            }
        }
        return ret;
    }
};
struct edge {
    int v, nxt;
};
```

```cpp
int val[N];
int son[N], sz[N], dpt[N], fa[N];
int mptot, mp[N << 2], top[N << 2], belong[N << 2], ed[N];
Matrix seg[N << 2], upVal[N];
int head[N], tot;
edge e[N << 1];
int dp[N][2], ldp[N][2];

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

inline void addEdge(int u, int v) {
    e[tot] = edge{v, head[u]};
    head[u] = tot++;
}

inline void dfs(int u){
    sz[u] = 1;
    for(int i = head[u]; ~i; i = e[i].nxt){
        int v = e[i].v;
        if(v == fa[u])  continue;
        fa[v] = u;
        dpt[v] = dpt[u] + 1;
        dfs(v);
        sz[u] += sz[v];
        if(son[u] == 0 || sz[son[u]] < sz[v]) {
            son[u] = v;
        }
    }
}

inline void buildTree(int u, int rt, int n){
    mp[u] = ++mptot;
    belong[mptot] = u;
    top[u] = rt;
    ed[rt] = mptot;
    ldp[u][1] = val[u];
    if(son[u]) {
        buildTree(son[u], rt, n);
        dp[u][0] += max(dp[son[u]][0], dp[son[u]][1]);
        dp[u][1] += dp[son[u]][0];
    }

    for(int i = head[u]; ~i; i = e[i].nxt){
        int v = e[i].v;
```

```
        if(v == fa[u] || v == son[u])   continue;
        buildTree(v, v, n);
        ldp[u][0] += max(dp[v][0], dp[v][1]);
        ldp[u][1] += dp[v][0];
    }
    dp[u][0] += ldp[u][0];
    dp[u][1] += ldp[u][1];
}

inline void build(int l, int r, int rt) {
    if(l == r) {
        int u = belong[l];
        upVal[u].mat[0][0] = upVal[u].mat[0][1] = ldp[u][0];
        upVal[u].mat[1][0] = ldp[u][1];
        upVal[u].mat[1][1] = -inf;
        seg[rt] = upVal[u];
        return;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    seg[rt] = seg[rt << 1] * seg[rt << 1 | 1];
}

inline void update(int pos, int l, int r, int rt) {
    if(l == r) {
        seg[rt] = upVal[belong[l]];
        return;
    }
    int m = (l + r) >> 1;
    if(pos <= m) {
        update(pos, lson);
    } else {
        update(pos, rson);
    }
    seg[rt] = seg[rt << 1] * seg[rt << 1 | 1];
}

inline Matrix query(int ql, int qr, int l, int r, int rt) {
    if(ql <= l && r <= qr) {
        return seg[rt];
    }
    int m = (l + r) >> 1;
    if(m < ql) {
        return query(ql, qr, rson);
    } else if(qr <= m) {
        return query(ql, qr, lson);
    } else {
```

```
        return query(ql, qr, lson) * query(ql, qr, rson);
    }
}

inline void change(int u, int x) {
    upVal[u].mat[1][0] += x - val[u];
    val[u] = x;
    while(u) {
        int now = top[u];
        Matrix pre = query(mp[now], ed[now], 1, mptot, 1);
        update(mp[u], 1, mptot, 1);
        Matrix cur = query(mp[now], ed[now], 1, mptot, 1);
        u = fa[now];
        upVal[u].mat[0][0] += (max(cur.mat[0][0], cur.mat[1][0]) -
         ↪ max(pre.mat[0][0], pre.mat[1][0]));
        upVal[u].mat[1][0] += (cur.mat[0][0] - pre.mat[0][0]);
        upVal[u].mat[0][1] = upVal[u].mat[0][0];
    }
}

int main() {
    int n, m;
    while(~scanf("%d%d", &n, &m)) {
        init();
        for(int i = 1; i <= n; i++) {
            scanf("%d", &val[i]);
        }
        for(int i = 1; i <= n - 1; i++) {
            int u, v;
            scanf("%d%d", &u, &v);
            addEdge(u, v);
            addEdge(v, u);
        }
        dfs(1);
        buildTree(1, 1, n);
        build(1, mptot, 1);

        while(m--) {
            int u, x;
            scanf("%d%d", &u, &x);
            change(u, x);
            Matrix mat = query(mp[1], ed[1], 1, mptot, 1);
            printf("%d\n", max(mat.mat[0][0], mat.mat[1][0]));
        }
    }
}
```

7.10 CDQ 分治

7.10.1 求解二维 LIS 问题

```
#include <bits/stdc++.h>
using namespace std;
const int N = (int)5e4 + 15;

struct info {
    int h, v, idx, dp;
    double cnt;
};
struct Tree {
    int len;
    double sum;
    Tree(): len(0), sum(0) {}
    Tree(int len, double sum): len(len), sum(sum) {}
};
info a[N], b[N];
int mph[N], mpv[N];
Tree tree[N];

inline bool cmp(const info& a, const info& b) {
    return a.idx < b.idx;
}
inline bool cmp1 (const info& a, const info& b) {
    return a.h != b.h ? a.h < b.h : a.v < b.v;
}

inline int lowbit(int x) {
    return x & -x;
}

inline void update(int x, int len, double sum) {
    for(; x < N; x += lowbit(x)) {
        if(tree[x].len < len) {
            tree[x] = Tree{len, sum};
        } else if(tree[x].len == len) {
            tree[x].sum += sum;
        }
    }
}

inline Tree getSum(int x) {
    Tree ret;
    for(; x > 0; x -= lowbit(x)) {
        if(tree[x].len > ret.len) {
            ret = tree[x];
        } else if(tree[x].len == ret.len) {
```

```
                ret.sum += tree[x].sum;
            }
        }
        return ret;
}

inline void clearTree(int x) {
    for(; x < N; x += lowbit(x)) {
        tree[x] = Tree{0, 0};
    }
}

inline void cdq(int l, int r, info* a) {
    if(r <= l) {
        return;
    }
    int mid = (l + r) >> 1;
    cdq(l, mid, a);

    sort(a + l, a + mid + 1, cmp1);
    sort(a + mid + 1, a + r + 1, cmp1);
    int st = l;
    for(int i = mid + 1; i <= r; i++) {
        while(a[st].h <= a[i].h && st <= mid) {
            update(a[st].v, a[st].dp, a[st].cnt);
            st++;
        }
        Tree res = getSum(a[i].v);
        if(res.len == 0) {
            continue;
        }
        if(res.len + 1 > a[i].dp) {
            a[i].dp = res.len + 1;
            a[i].cnt = res.sum;
        } else if(res.len + 1 == a[i].dp) {
            a[i].cnt += res.sum;
        }
    }

    for(int i = l; i < st; i++) {
        clearTree(a[i].v);
    }

    sort(a + mid + 1, a + r + 1, cmp);

    cdq(mid + 1, r, a);
}
```

```cpp
int main() {
    int n;
    while(~scanf("%d", &n)) {
        for(int i = 1; i <= n; i++) {
            scanf("%d%d", &a[i].h, &a[i].v);
            a[i].idx = i, b[i].idx = n - i + 1;
            a[i].dp = b[i].dp = 1;
            a[i].cnt = b[i].cnt = 1;
            mph[i] = a[i].h;
            mpv[i] = a[i].v;
        }
        sort(mph + 1, mph + n + 1);
        sort(mpv + 1, mpv + n + 1);
        int toth = unique(mph + 1, mph + n + 1) - mph - 1;
        int totv = unique(mpv + 1, mpv + n + 1) - mpv - 1;
        for(int i = 1; i <= n; i++) {
            a[i].h = lower_bound(mph + 1, mph + toth + 1, a[i].h) -
             ↪ mph;
            a[i].v = lower_bound(mpv + 1, mpv + totv + 1, a[i].v) -
             ↪ mpv;
        }

        for(int i = 1; i <= n; i++) {
            b[i].h = a[i].h;
            b[i].v = a[i].v;
            a[i].h = toth - a[i].h + 1;
            a[i].v = totv - a[i].v + 1;
        }
        reverse(b + 1, b + n + 1);

        cdq(1, n, a);
        sort(a + 1, a + n + 1, cmp);

        int maxLen = 0;
        for(int i = 1; i <= n; i++) {
            maxLen = max(maxLen, a[i].dp);
        }
        double sumCnt = 0;
        for(int i = 1; i <= n; i++) {
            if(a[i].dp == maxLen) {
                sumCnt += a[i].cnt;
            }
        }

        cdq(1, n, b);
        sort(b + 1, b + n + 1, cmp);

        printf("%d\n", maxLen);
```

```
        for(int i = 1; i <= n; i++) {
            if(a[i].dp + b[n - i + 1].dp - 1 == maxLen) {
                printf("%.5f ", a[i].cnt * b[n - i + 1].cnt /
                 ↪  sumCnt);
            } else {
                printf("0.00000 ");
            }
        }
        puts("");
    }
}
```

7.11 斜率 DP

7.11.1 Codeforces 1083E

```
struct Node {
    ll x, y, w;
    bool operator < (const Node& b) const {
        return x < b.x;
    }
};

Node a[N];
int que[N];
ll f[N];

inline ll calc(int i, int j) {
    return f[j] + 1LL * (a[i].x - a[j].x) * a[i].y - a[i].w;
}

inline double cmp(int i, int j) {
    ll dy = f[i] - f[j];
    ll dx = a[i].x - a[j].x;
    return (double)dy / dx;
}

int main() {
    int n;
    while(~scanf("%d", &n)) {
        for(int i = 1; i <= n; i++) {
            scanf("%lld%lld%lld", &a[i].x, &a[i].y, &a[i].w);
        }
        sort(a + 1, a + 1 + n);

        que[1] = 0;
        for(int i = 1, l = 1, r = 1; i <= n; i++) {
```

```
            while(l + 1 <= r && calc(i, que[l]) <= calc(i, que[l +
             ↪  1])) {
                l++;
            }
            f[i] = calc(i, que[l]);

            while(r >= 2 && cmp(que[r - 1], que[r]) < cmp(que[r], i))
             ↪  {
                r--;
            }
            l = min(r, l);
            que[++r] = i;
        }

        printf("%lld\n", *max_element(f + 1, f + 1 + n));
    }
}
```

7.12 莫队算法

7.12.1 不带修改

```
bool cmp(const node& a, const node& b){
    if(a.l/block_size == b.l/block_size){
        return a.r < b.r;
    }else{
        return a.l/block_size < b.l/block_size;
    }
}

int l = 0, r = -1;
int answer = 0;
for(int i = 0; i < q; i++){
    while(que[i].l < l) { add(--l, answer); }
    while(l < que[i].l) { del(l++, answer); }
    while(que[i].r < r) { del(r--, answer); }
    while(r < que[i].r) { add(++r, answer); }
    ans[que[i].id] = answer;
}
```

7.12.2 带修改

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
const int N = 10000 + 5;
```

```cpp
const int M = 1e6 + 5;

struct QQuery{
    int l, r, tim, idx;
};
struct QModify{
    int pos, val, pre;
};
int BLOCK_SIZE;
int ans[N];
int cnt[M];
int a[N];
QQuery  queq[N];
QModify queu[N];

bool cmp(const QQuery& a, const QQuery& b){
    if(a.l/BLOCK_SIZE != b.l/BLOCK_SIZE)         return a.l < b.l;
    else if(a.r/BLOCK_SIZE != b.r/BLOCK_SIZE)    return a.r < b.r;
    return a.tim/BLOCK_SIZE < b.tim/BLOCK_SIZE;
}

void moveTimeForward(int tim, int l, int r, int& ans){
    QModify& qcur = queu[tim];
    qcur.pre = a[qcur.pos];
    a[qcur.pos] = qcur.val;
    if(l <= qcur.pos && qcur.pos <= r){
        if((--cnt[qcur.pre]) == 0)  ans--;
        if((++cnt[qcur.val]) == 1)  ans++;
    }
}

void moveTimeBack(int tim, int l, int r, int& ans){
    QModify& qcur = queu[tim];
    a[qcur.pos] = qcur.pre;
    if(l <= qcur.pos && qcur.pos <= r){
        if((--cnt[qcur.val]) == 0)  ans--;
        if((++cnt[qcur.pre]) == 1)  ans++;
    }
}

void add(int pos, int& ans){
    if((++cnt[a[pos]]) == 1)    ans++;
}

void del(int pos, int& ans){
    if((--cnt[a[pos]]) == 0)    ans--;
}
```

```
int main(){
    int n, q;
    while(~scanf("%d%d", &n, &q)){
        memset(cnt, 0, sizeof(cnt));
        BLOCK_SIZE = (int)pow(n, 2.0/3);
        for(int i = 1; i <= n; i++){
            scanf("%d", &a[i]);
        }

        int ppq = 0, ppu = 0;
        while(q--){
            char s[2];
            int x, y;
            scanf("%s%d%d", s, &x, &y);
            if(s[0] == 'Q'){
                queq[ppq] = QQuery{x, y, ppu, ppq};
                ppq++;
            }else{
                queu[++ppu] = QModify{x, y, 0};
            }
        }
        sort(queq, queq + ppq, cmp);

        int r = -1, l = 0, tim = 0, curans = 0;
        for(int i = 0; i < ppq; i++){
            QQuery& q = queq[i];
            while(tim < q.tim)     moveTimeForward(++tim, l, r,
             ↪  curans);
            while(tim > q.tim)     moveTimeBack(tim--, l, r, curans);
            while(l < q.l)         del(l++, curans);
            while(l > q.l)         add(--l, curans);
            while(r < q.r)         add(++r, curans);
            while(r > q.r)         del(r--, curans);
            ans[q.idx] = curans;
        }

        for(int i = 0; i < ppq; i++){
            printf("%d\n", ans[i]);
        }
    }
}
```

7.12.3 树上莫队

```
#include <cstdio>
#include <cstring>
#include <cmath>
#include <algorithm>
```

```
using namespace std;
const int N = 1e5 + 5;
typedef long long ll;

struct squery{
    int u, v, tim, idx;
};
struct supdate{
    int u, val, preval;
};
struct edge{
    int v, next;
};
squery  query[N];
supdate update[N];
int  w[N], v[N], c[N];
edge e[N << 1];
int  head[N], tot;
int  belong[N];
int  stk[N], pstk;
int  fa[N][18];
int  dpt[N];
bool used[N];
int  cnt[N];
ll   ans[N];
int  BLOCK_SIZE;

bool cmp(const squery& a, const squery& b){
    if(belong[a.u] != belong[b.u])          return belong[a.u] <
    ↪  belong[b.u];
    else if(belong[a.v] != belong[b.v])     return belong[a.v] <
    ↪  belong[b.v];
    else                                    return a.tim/BLOCK_SIZE <
    ↪  b.tim/BLOCK_SIZE;
}

inline void init(){
    memset(head, -1, sizeof(head));
    memset(fa, -1, sizeof(fa));
    memset(cnt, 0, sizeof(cnt));
    memset(used, 0, sizeof(used));
    tot = 0;
}

inline void addEdge(int u, int v){
    e[tot] = edge{v, head[u]};
    head[u] = tot++;
}
```

```
void dfs(int u, int pre, int depth){
    int bottom = pstk;
    dpt[u] = depth;
    for(int i = head[u]; ~i; i = e[i].next){
        int v = e[i].v;
        if(v == pre)    continue;
        fa[v][0] = u;
        dfs(v, u, depth + 1);
        if(pstk - bottom >= BLOCK_SIZE){
            while(pstk != bottom){
                belong[stk[pstk--]] = u;
            }
        }
    }
    stk[++pstk] = u;
}

void initLCA(int n){
    for(int j = 1; j <= 17; j++){
        for(int u = 1; u <= n; u++){
            if(fa[u][j - 1] == -1)  continue;
            fa[u][j] = fa[fa[u][j - 1]][j - 1];
        }
    }
}

void initBlockAndLCA(int n){
    BLOCK_SIZE = (int)pow(n, 2.0/3);
    pstk = 0;
    dfs(1, -1, 0);
    while(pstk >= 1){
        belong[stk[pstk--]] = 1;
    }
    initLCA(n);
}

int getFa(int u, int v){
    if(dpt[u] > dpt[v])     swap(u, v);
    for(int j = 17; j >= 0; j--){
        if(fa[v][j] == -1 || dpt[fa[v][j]] < dpt[u])    continue;
        v = fa[v][j];
    }
    if(u == v)  return u;
    for(int j = 17; j >= 0; j--){
        if(fa[v][j] == -1 || fa[u][j] == -1 || fa[u][j] == fa[v][j])
         ↪  continue;
        u = fa[u][j], v = fa[v][j];
```

```
    }
    return fa[u][0];
}

void reverse(int u, ll& ans){
    if(used[u]){
        ans -= (ll)v[c[u]] * w[cnt[c[u]]];
        cnt[c[u]]--;
    }else{
        cnt[c[u]]++;
        ans += (ll)v[c[u]] * w[cnt[c[u]]];
    }
    used[u] ^= 1;
}

void change(int u, int val, ll& ans){
    if(used[u]){
        reverse(u, ans);
        c[u] = val;
        reverse(u, ans);
    }else{
        c[u] = val;
    }
}

void moveTimeForward(int tim, ll& ans){
    int u = update[tim].u;
    update[tim].preval = c[u];
    change(u, update[tim].val, ans);
}

void moveTimeBack(int tim, ll& ans){
    int u = update[tim].u;
    change(u, update[tim].preval, ans);
}

void moveNode(int u, int v, ll& ans){
    while(u != v){
        if(dpt[u] < dpt[v]){
            reverse(v, ans);
            v = fa[v][0];
        }else{
            reverse(u, ans);
            u = fa[u][0];
        }
    }
}
```

```
int main(){
    int n, m, q;
    while(~scanf("%d%d%d", &n, &m, &q)){
        init();
        for(int i = 1; i <= m; i++)     scanf("%d", &v[i]);
        for(int i = 1; i <= n; i++)     scanf("%d", &w[i]);
        for(int i = 1; i <= n - 1; i++){
            int u, v;
            scanf("%d%d", &u, &v);
            addEdge(u, v);
            addEdge(v, u);
        }
        for(int i = 1; i <= n; i++)     scanf("%d", &c[i]);
        initBlockAndLCA(n);

        int pp = 0, pq = 0;
        while(q--){
            int type, x, y;
            scanf("%d%d%d", &type, &x, &y);
            if(type == 0){
                update[++pq] = supdate{x, y, -1};
            }else{
                query[pp] = squery{x, y, pq, pp};
                pp++;
            }
        }
        sort(query, query + pp, cmp);

        int u = query[0].u, v = query[0].v, tim = 0;
        ll curans = 0;
        while(tim < query[0].tim)   moveTimeForward(++tim, curans);
        moveNode(u, v, curans);
        reverse(getFa(u, v), curans);
        ans[query[0].idx] = curans;
        reverse(getFa(u, v), curans);
        for(int i = 1; i < pp; i++){
            while(tim < query[i].tim)       moveTimeForward(++tim,
             ↪  curans);
            while(tim > query[i].tim)       moveTimeBack(tim--,
             ↪  curans);
            int nu = query[i].u, nv = query[i].v;
            moveNode(u, nu, curans);
            moveNode(v, nv, curans);
            int lca = getFa(nu, nv);
            reverse(lca, curans);
            ans[query[i].idx] = curans;
            reverse(lca, curans);
            u = nu, v = nv;
```

```
        }
        for(int i = 0; i < pp; i++){
            printf("%lld\n", ans[i]);
        }
    }
}
```

7.13 分块

7.13.1 区间正偶数次数的个数（动态查询）

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = (int)100000 + 3;

int belong[N], st[N], ed[N], sz[N];
int a[N], buc[N];
int f[320][320], cnt[320][N];

inline void init(int n) {
    int num = sqrt(n);

    for(int i = 1, m = n / num; i <= num; i++) {
        st[i] = m * (i - 1) + 1;
        ed[i] = m * i;
        sz[i] = m;
    }
    ed[num] = n;
    sz[num] = n - st[num] + 1;
    for(int i = 1; i <= num; i++) {
        for(int j = st[i]; j <= ed[i]; j++) {
            belong[j] = i;
        }
    }
}

int main() {
    int n, c, m;
    scanf("%d%d%d", &n, &c, &m);
    for(int i = 1; i <= n; i++) {
        scanf("%d", &a[i]);
    }
    init(n);
    for(int i = 1, m = belong[n]; i <= m; i++) {
        memset(buc + 1, 0, c * sizeof(int));
        for(int j = 1; j <= c; j++) {
            cnt[i][j] = cnt[i - 1][j];
```

```cpp
        }
        for(int k = st[i]; k <= ed[i]; k++) {
            cnt[i][a[k]]++;
        }
        for(int j = i; j <= m; j++) {
            int& res = f[i][j];
            res = f[i][j - 1];
            for(int k = st[j]; k <= ed[j]; k++) {
                res += (buc[a[k]] == 0 ? 0 : (buc[a[k]] & 1 ? 1 :
                 ↪  -1));
                buc[a[k]]++;
            }
        }
    }

    int ans = 0;
    while(m--) {
        int l, r;
        scanf("%d%d", &l, &r);
        l = (l + ans) % n + 1;
        r = (r + ans) % n + 1;
        if(l > r) {
            swap(l, r);
        }
        int x = belong[l], y = belong[r];
        ans = 0;
        if(y - x <= 1) {
            for(int i = l; i <= r; i++) {
                buc[a[i]] = 0;
            }
            for(int i = l; i <= r; i++) {
                ans += (buc[a[i]] == 0 ? 0 : (buc[a[i]] & 1 ? 1 :
                 ↪  -1));
                buc[a[i]]++;
            }
        } else {
            ans = f[x + 1][y - 1];
            for(int i = l; i <= ed[x]; i++) {
                buc[a[i]] = 0;
            }
            for(int i = st[y]; i <= r; i++) {
                buc[a[i]] = 0;
            }
            for(int i = l; i <= ed[x]; i++) {
                int tmp = buc[a[i]] + cnt[y - 1][a[i]] -
                 ↪  cnt[x][a[i]];
                ans += (tmp == 0 ? 0 : (tmp & 1 ? 1 : -1));
                buc[a[i]]++;
```

```
            }
            for(int i = st[y]; i <= r; i++) {
                int tmp = buc[a[i]] + cnt[y - 1][a[i]] -
                ↪   cnt[x][a[i]];
                ans += (tmp == 0 ? 0 : (tmp & 1 ? 1 : -1));
                buc[a[i]]++;
            }
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

8 Note

8.1 Formula And Equations

$$1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

$$1^3 + 2^3 + \cdots + n^3 = \left[\frac{n(n+1)}{2}\right]^2$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{a|i}\sum_{b|j} f(a,b) = \sum_{i=1}^{n}\sum_{j=1}^{m}\left\lfloor\frac{n}{i}\right\rfloor\left\lfloor\frac{m}{j}\right\rfloor f(i,j)$$

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{m}\left\lfloor\frac{n}{i}\right\rfloor\left\lfloor\frac{m}{j}\right\rfloor\sum_{x|(i,j)} f(x)\\
&=\sum_{i=1}^{n}\sum_{j=1}^{m}\left\lfloor\frac{n}{i}\right\rfloor\left\lfloor\frac{m}{j}\right\rfloor\sum_{d|i,d|j} f(d)\\
&=\sum_{d=1}^{\min(n,m)} f(d)\sum_{i=1}^{j}\sum_{j=1}^{m}\left\lfloor\frac{n}{id}\right\rfloor\left\lfloor\frac{m}{jd}\right\rfloor
\end{aligned}$$

$$b^2 - a^2 = \sum_{i=a}^{b-1}(2i+1)$$

8.2 Facts

- Number of possible values for $\left\lceil\frac{a}{b}\right\rceil$ is $O(\sqrt{n})$. Those numbers are $1, 2, \cdots, \sqrt{x}$ and $\left\lceil\frac{x}{1}\right\rceil, \left\lceil\frac{x}{2}\right\rceil, \cdots, \left\lceil\frac{x}{\sqrt{x}}\right\rceil$, here x can be the result or denominator.
- **Manhattan Distance**: $\sum_k |a_k - b_k| = \sum_k c_k(a_k - b_k) = \max_{c_1,c_2,\cdots,c_k} \sum_k c_k(a_k - b_k)$
- $d(n)$: $d(nm) = \sum_{i|n}\sum_{j|m}[(i,j)=1]$

- **Mobius Function** μ: $\sum\limits_{x|n} \mu(x) = [n = 1]$

- **Dirichlet Convolution**: $(f * g)(n) = \sum\limits_{d|n} f(d)g\left(\frac{n}{d}\right)$

8.3 Catalan Number

Basic

$$C_n = C_{2n}^{n} - C_{2n}^{n-1} = \frac{1}{n+1} C_{2n}^{n}$$

$$C_n = \frac{1}{n+1} \sum_{i=0}^{n} (C_n^i)^2$$

$$C_{n+1} = \frac{2(2n+1)}{n+2} C_n, \quad C_0 = 1$$

$$C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i}, \quad C_0 = 1$$

DP and Extra

We construct a sequence only contains -1 and 1. It satisfys the maximal prefix sum is less than or equal to 0, i.e.

$$\max_j \left\{ \sum_{i=1}^{j} a_i \right\} \leq 0$$

We define $f(x, y)$ is using the number of -1 is x, and the number of 1 is y. Then

$$\begin{aligned}
&f(x, y) = f(x-1, y) + f(x, y-1) && (x >= y) \\
&f(x, y) = 0 && (x < y) \\
&f(x, y) = 1 && (x > 0, y = 0)
\end{aligned}$$

It can be regarded as appending a new 1 to the ending or a new -1, and they cannot increase the maximal prefix sum because of the condition $x >= y$. After appending, the new prefix sum is $y - x$, which is less than or equal to 0.

In addtion, $f(x, y) = C_{x+y}^{x} - C_{x+y}^{x+1} \quad (x \geq y)$

8.4 Combination

Property

$$C_n^k = \frac{k}{n} C_{n-1}^{k-1}$$

$$C_n^m = C_{n-1}^{m} + C_{n-1}^{m-1}$$

$$\sum_{i=0}^{n} C_n^i = 2^n$$

$$\sum_{i=0}^{n}(-1)^i C_n^i = 0$$

$$C_{m+n}^m = \sum_{i=0}^{m} C_n^i C_m^{m-i} = \sum_{i=0}^{m} C_n^i C_m^i \quad (n \geq m)$$

$$\sum_{i=0}^{n}(C_n^i)^2 = C_{2n}^n$$

$$\sum_{i=0}^{n} iC_n^i = n2^{n-1}$$

$$\sum_{i=0}^{n} i^2 C_n^i = n(n+1)2^{n-2}$$

$$\sum_{l=0}^{r} C_l^k = C_{n+1}^{k+1}$$

$$C_n^r C_r^k = C_n^k C_{n-k}^{r-k}$$

$$\sum_{i=0}^{n} C_{n-i}^i = Fib(n+1)$$

8.5 Sum of GCD

Mobius Reversion

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{m}(i,j) \\
&= \sum_{d=1}^{n} d \cdot \sum_{i=1}^{n}\sum_{j=1}^{m}[(i,j)=d] \\
&= \sum_{d=1}^{n} d \cdot \sum_{i=1}^{n}\sum_{j=1}^{m}\left[\left(\frac{i}{d},\frac{j}{d}\right)=1\right] \\
&= \sum_{d=1}^{n} d \cdot \sum_{\frac{i}{d}=1}^{n/d}\sum_{\frac{j}{d}=1}^{m/d}\sum_{x|(\frac{i}{d},\frac{j}{d})}\mu(x) \\
&= \sum_{d=1}^{n} d \cdot \sum_{k=1}^{n/d}\mu(k)\left\lfloor\frac{n}{kd}\right\rfloor\left\lfloor\frac{m}{kd}\right\rfloor
\end{aligned}$$

Euler Function

$$
\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{m}(i,j)\\
&=\sum_{d=1}^{n} d\cdot\sum_{k=1}^{n/d}\mu(k)\left\lfloor\frac{n}{kd}\right\rfloor\left\lfloor\frac{m}{kd}\right\rfloor\\
&=\sum_{d=1}^{n} d\cdot\sum_{T|n}\left\lfloor\frac{n}{T}\right\rfloor\left\lfloor\frac{m}{T}\right\rfloor\mu(\frac{T}{d})\quad(let\ T=kd)\\
&=\sum_{T=1}^{n}\left\lfloor\frac{n}{T}\right\rfloor\left\lfloor\frac{m}{T}\right\rfloor\sum_{d|T}d\cdot\mu(\frac{T}{d})\\
&=\sum_{T=1}^{n}\left\lfloor\frac{n}{T}\right\rfloor\left\lfloor\frac{m}{T}\right\rfloor\sum_{d|T}\mu(d)\cdot\frac{T}{d}\\
&=\sum_{T=1}^{n}\left\lfloor\frac{n}{T}\right\rfloor\left\lfloor\frac{m}{T}\right\rfloor\varphi(T)
\end{aligned}
$$

Dirichlet Convolution

$$
\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{m}(i,j)\\
&=\sum_{T=1}^{n}\left\lfloor\frac{n}{T}\right\rfloor\left\lfloor\frac{m}{T}\right\rfloor\varphi(T)
\end{aligned}
$$

let $\phi(n)=\sum_{T=1}^{n}\varphi(T)$, according to

$$g(1)S(n)=\sum_{i=1}^{n}(f*g)(i)-\sum_{i=2}^{n}S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)$$

then

$$
\begin{aligned}
\phi(n)&=\sum_{i=1}^{n}\sum_{d|i}\varphi(d)-\sum_{i=2}^{n}\phi\left(\left\lfloor\frac{n}{i}\right\rfloor\right)\\
&=\sum_{i=1}^{n}i-\sum_{i=2}^{n}\phi\left(\left\lfloor\frac{n}{i}\right\rfloor\right)\\
&=\frac{n(n+1)}{2}-\sum_{i=2}^{n}\phi\left(\left\lfloor\frac{n}{i}\right\rfloor\right)
\end{aligned}
$$

8.6 Sum of LCM

Problem

$$f(n,m)=\sum_{i=1}^{n}\sum_{i=1}^{m}lcm(i,j)$$

Mobius Reversion

We assume that $n <= m$, then

$$\begin{aligned}
\sum_{i=1}^{n}\sum_{i=1}^{m} lcm(i,j) &= \sum_{i=1}^{n}\sum_{i=1}^{m} \frac{ij}{gcd(i,j)} \\
&= \sum_{i=1}^{n}\sum_{i=1}^{m}\sum_{d} \frac{ij}{[(i,j)=d]\cdot d} \\
&= \sum_{i=1}^{n}\sum_{i=1}^{m}\sum_{d} \frac{ij}{[(i,j)=d]\cdot d^2}\cdot d \\
&= \sum_{d=1}^{n} d \sum_{i=1}^{n/d}\sum_{j=1}^{m/d} [(i,j)=1]\cdot ij
\end{aligned}$$

let $g(n,m) = \sum_{i=1}^{n}\sum_{j=1}^{m} [(i,j)=1]\cdot ij$, then

$$\begin{aligned}
g(n,m) &= \sum_{i=1}^{n}\sum_{j=1}^{m} [(i,j)=1]\cdot ij \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{d|(i,j)} \mu(d)\cdot ij \\
&= \sum_{d=1}^{n}\sum_{i=1}^{n/d}\sum_{j=1}^{m/d} \mu(d)\cdot id\cdot jd \\
&= \sum_{d=1}^{n} d^2\mu(d) \sum_{i=1}^{n/d}\sum_{j=1}^{m/d} ij
\end{aligned}$$

let $h(n,m) = \sum_{i=1}^{n}\sum_{j=1}^{m} ij$, then

$$h(n,m) = \sum_{i=1}^{n}\sum_{j=1}^{m} ij = \frac{n(n+1)}{2}\cdot\frac{m(m+1)}{2}$$

Therefore,

$$\begin{aligned}
h(n,m) &= \frac{n(n+1)}{2}\cdot\frac{m(m+1)}{2} \\
g(n,m) &= \sum_{d=1}^{n} d^2\mu(d)\cdot h\left(\left\lfloor\frac{n}{d}\right\rfloor, \left\lfloor\frac{m}{d}\right\rfloor\right) \\
f(n,m) &= \sum_{d=1}^{n} d\cdot g\left(\left\lfloor\frac{n}{d}\right\rfloor, \left\lfloor\frac{m}{d}\right\rfloor\right)
\end{aligned}$$

8.7 Sum of $d(i,j)$

Problem

$$f(n,m) = \sum_{i=1}^{n}\sum_{j=1}^{m} d(ij)$$

Mobius Reversion

We assume that $n <= m$, then we have

$$\begin{aligned}
d(ij) &= \sum_{x|i} \sum_{y|j} [(x,y)=1] \\
&= \sum_{x|i} \sum_{y|j} \sum_{p|(x,y)} \mu(p) \\
&= \sum_{p=1}^{\min(i,j)} \sum_{x|i} \sum_{y|j} [p|(x,y)]\, \mu(p) \\
&= \sum_{p|i,p|j} \mu(p) \sum_{x|i} \sum_{y|j} [p|(x,y)] \\
&= \sum_{p|i,p|j} \mu(p) \sum_{px|i} \sum_{py|j} 1 \\
&= \sum_{p|i,p|j} \mu(p) \sum_{x|\frac{i}{p}} \sum_{y|\frac{j}{p}} 1 \\
&= \sum_{p|i,p|j} \mu(p) d\left(\frac{i}{p}\right) d\left(\frac{j}{p}\right)
\end{aligned}$$

Therefore,

$$\begin{aligned}
f(n,m) &= \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{p|i,p|j} \mu(p) d\left(\frac{i}{p}\right) d\left(\frac{j}{p}\right) \\
&= \sum_{p=1}^{n} \sum_{i=1}^{n/p} \sum_{j=1}^{m/p} \mu(p) d(i) d(j) \\
&= \sum_{p=1}^{n} \mu(p) \sum_{i=1}^{n/p} d(i) \sum_{j=1}^{m/p} d(j)
\end{aligned}$$

8.8 Sum of $lcm(i,n)$

Problem

$$f(n) = \sum_{i=1}^{n} lcm(i,n)$$

Mobius Reversion

$$
\begin{aligned}
f(n) &= \sum_{i=1}^{n} \frac{in}{(i,n)} \\
&= \frac{1}{2}\left(\sum_{i=1}^{n-1} \frac{in}{(i,n)} + \sum_{i=1}^{n-1} \frac{(n-i)n}{(i,n)}\right) + n \\
&= \frac{1}{2}\sum_{i=1}^{n-1} \frac{n^2}{(i,n)} + n \\
&= n + \frac{1}{2}\sum_{d=1}^{n-1}\sum_{i=1}^{n-1} \frac{n^2}{d}\left[(i,n) = d\right] \\
&= n + \frac{1}{2}\sum_{d|n,d<n}\ \sum_{d|i,i<n} \frac{n^2}{d}\left[(\frac{i}{d},\frac{n}{d}) = 1\right] \\
&= n + \frac{1}{2}\sum_{d|n,d<n} \frac{n^2}{d}\varphi(\frac{n}{d}) \\
&= n + \frac{1}{2}n\sum_{d|n,d>1} d\varphi(d)
\end{aligned}
$$

We let $g(n) = \sum_{d|n} d\varphi(d)$, and it is multiplicative obviously. Therefore, we can get it in $O(n)$, and we have

$$
f(n) = n + \frac{1}{2}n\left[g(n) - \varphi(1)\right]
$$

8.9 Lagrange polynomial

Basic

$$
f(x) = \sum_{i=0}^{k} y_i l_i(x) \tag{1}
$$

$$
l_i(x) = \prod_{j=0,i\neq j}^{k} \frac{x - x_j}{x_i - x_j} = \begin{cases} 1, & x = x_i \\ 0, & x \neq x_i \end{cases}
$$

Sum of i^k

let $f(n) = \sum\limits_{i=1}^{n} i^k$, $d = k+1$, then

$$
\begin{aligned}
f(n) &= \sum_{i=0}^{k+1} f(i) l_i(n) \\
&= \sum_{i=0}^{k+1} \left(\prod_{j=0, i \neq j}^{k+1} \frac{n - x_j}{x_i - x_j} \right) \\
&= \sum_{i=0}^{k+1} f(i) \cdot \frac{n(n-1)\cdots(n-i+1)(n-i-1)\cdots(n-d)}{i(i-1)\cdots 1 \cdot (-1)(-2)\cdots(i-d)} \\
&= \sum_{i=0}^{k+1} f(i) \cdot (-1)^{i+k+1} \cdot \frac{n(n-1)\cdots(n-i+1)\cdot(n-i-1)\cdots(n-d)}{i! \cdot (d-i)!}
\end{aligned}
$$

8.10 Min25 Sieve

Prerequisite

f is multiplicative function, and $f(p)$ is a low-degree polynomial, which can be calculated in quick time when p is a prime.

Formula

$$
\begin{aligned}
prime &= \{p_1, p_2, \cdots, p_j\} \\
sum_j &= \sum_{i=1}^{j} f(p_i) \\
g(n, j) &= \sum_{i=2}^{n} f(i) \cdot [i \in prime \ or \ min(p) > p_j, p | i, p \in prime] \\
&= g(n, j-1) - f(p_j) \cdot \left(g\left(\left\lfloor \frac{n}{p_j} \right\rfloor, j-1 \right) - sum_{j-1} \right) \\
S(n, j) &= \sum_{i=2}^{n} f(i) \cdot [min(p) > p_j, p | i, p \in prime] \\
&= g(n, \infty) - sum_j + \sum_{e} \sum_{k=j+1} f(p_k^e) \cdot \left(S\left(\left\lfloor \frac{n}{p_k^e} \right\rfloor, k \right) + [e \neq 1] \right) \\
ans &= \sum_{i=1}^{n} f(i) = S(n, 0) + f(1)
\end{aligned}
$$

8.11 Matrix Tree

Illustration

Kirchhoff Matrix Tree is used to calculate the number of the spanning tree in a graph.

BEST is used to calculate the number of the Euler circuit in a Euler graph.

Undirected Graph

Define A is the adjacent matrix, D is the degree matrix, then

$$L(G) = D(G) - A(G)$$

$$t(G) = \det L(G)\binom{1, 2, \cdots, i-1, i+1, \cdots, n}{1, 2, \cdots, i-1, i+1, \cdots, n}$$

Directed Graph

Define A is the adjacent matrix, D^{out} is the out degree matrix, D^{in} is the in degree matrix, then

$$L^{out}(G) = D^{out}(G) - A(G)$$
$$L^{in}(G) = D^{in}(G) - A(G)$$

$$t^{root}(G) = \det L^{out}(G)\binom{1, 2, \cdots, i-1, i+1, \cdots, n}{1, 2, \cdots, i-1, i+1, \cdots, n}$$
$$t^{leaf}(G) = \det L^{in}(G)\binom{1, 2, \cdots, i-1, i+1, \cdots, n}{1, 2, \cdots, i-1, i+1, \cdots, n}$$

BEST

$$ec(G) = t^{root}(G, k) \prod_{v \in V}(deg(v) - 1)!$$