

# Contents

<b>1 Basic</b>	<b>1</b>
1.1 Default Code	1
1.2 Functions	1
1.3 Pragma	1
1.4 Random Value Generator	1
1.5 set map pq	1
<b>2 Python</b>	<b>2</b>
2.1 int(str)	2
2.2 有理數 (可以直接做四則運算)	2
<b>3 Data Structure</b>	<b>2</b>
3.1 BIT	2
3.2 Distinct Value with BIT	2
3.3 BIT (Howard)	2
3.4 Segment Tree	3
3.5 Segment Tree(Howard)	4
3.6 Lazy Propagation Segment Tree	4
3.7 Binary Search on Segment Tree	5
3.8 Merging on Segment Tree	5
3.9 Segment Tree(Chung)	6
3.10 PBDS	6
3.11 Sweep Line	7
3.12 LinkedList	7
3.13 Sparse Table	8
3.14 Treap	8
<b>4 Graph</b>	<b>9</b>
4.1 DSU	9
4.2 LCA	9
4.3 HLD	9
4.4 Negative Cycle	10
4.5 Euler Tour	11
4.6 Hamiltonian Routes	11
4.7 Dijkstra	11
4.8 Floyd-Warshall	12
4.9 SCC kosaraju	12
<b>5 Flow</b>	<b>12</b>
5.1 Max Flow Dinic	12
5.2 Min Cost Max Flow	13
5.3 Min Cut	13
5.4 Maximum Bipartite Matching	14
<b>6 Math</b>	<b>14</b>
6.1 Matrix	14
6.2 Combination	14
6.3 Exponentiation	14
6.4 Unique factorization domain	15
6.5 Quick Mul	15
6.6 Miller Rabin	15
6.7 Pollard's Rho	15
6.8 Sum of Powers Formulas and Expansions	16
6.9 $ax+by=gcd(a,b)$	16
6.10 Discrete Sqrt	16
6.11 Prime	16
6.12 Gauss	16
6.13 FFT	17
6.14 Theorem	17
<b>7 Geometry</b>	<b>18</b>
7.1 Structure	18
7.2 Vector	18
7.3 Collinear	18
7.4 Inline	18
7.5 Convex Hull	18
<b>8 String</b>	<b>19</b>
8.1 Trie	19
8.2 Hash	19
8.3 Z Function	19

## 1 Basic

### 1.1 Default Code

```
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
// #pragma GCC optimize("trapv")
#include <bits/stdc++.h>
using namespace std;
```

```
int main()
```

```
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);

    return 0;
}
```

### 1.2 Functions

```
__builtin_popcount(n) // 表示將n表示成2進位有幾個
                        1
__builtin_popcountll(n) // 表示將n表示成2進位有幾個
                        1
__builtin_ctz(n) // 回傳n的二進位下最低為1後面有
                  幾個0
__builtin_ctzll(n) // 回傳n的二進位下最低為1後面
                  有幾個0
int __builtin_clz(unsigned int n) // 回傳n的二進
    位下最高為1前面有幾個0 (n不可為0)
int __builtin_clzll(unsigned long long n)
cout << fixed << setprecision(2) << 12.375019483
    << "\n"; // 12.38
```

### 1.3 Pragma

```
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
#pragma GCC optimize("trapv") // make overflow RE
    , slow
```

### 1.4 Random Value Generator

```
mt19937 rng(chrono::steady_clock::now().
    time_since_epoch().count());
shuffle(a.begin(), a.end(), rng);
uniform_int_distribution<long long>(l, r)(rng);
// [l, r]
```

### 1.5 set map pq

```
struct edge
{
    int a, b, w;
    friend istream& operator>>(istream &in, edge
        &x)
    {
        in >> x.a >> x.b >> x.w;
    }
    friend ostream& operator<<(ostream &out,
        const edge &x)
    {
        out << "(" << x.a << ", " << x.b << ", "
            << x.w << ")"; return out;
    }
};
```

```
struct cmp
{
    bool operator()(const edge &x, const edge &y) const { return x.w < y.w; } };

set<edge, cmp> st; //遞增
map<edge, long long, cmp> mp; //遞增
priority_queue<edge, vector<edge>, cmp> pq; //遞減
```

## 2 Python

### 2.1 int(str)

```
import sys
sys.set_int_max_str_digits(1000000)
int('1'*999999) # ok
int('1'*1000000) # error
# int(str) time complexity = O(n^2 / 64)
```

### 2.2 有理數 (可以直接做四則運算)

```
from fractions import Fraction
Fraction(16, -10) # Fraction(-8, 5)
Fraction(123) # Fraction(123, 1)
Fraction() # Fraction(0, 1)
Fraction('3/7') # Fraction(3, 7)
Fraction(' -3/7 ') # Fraction(-3, 7)
Fraction('1.414213 \t\n') # Fraction(1414213, 1000000)
Fraction('-.125') # Fraction(-1, 8)
Fraction('7e-6') # Fraction(7, 1000000)
Fraction(2.25) # Fraction(9, 4)
Fraction(1.1) # Fraction(2476979795053773, 2251799813685248) (直接傳小數會失真，應改成傳字串)

# method
Fraction(1,2).numerator # 1 (分子)
Fraction(1,2).denominator # 2 (分母)
Fraction(__import__(math).pi).limit_denominator(1000) # Fraction(355, 113) (113 < 1000)
Fraction(__import__(math).pi).limit_denominator(1000000) # Fraction(3126535, 995207) (995207 < 1000000)
```

## 3 Data Structure

### 3.1 BIT

```
struct BIT
{
    int n;
    long long bit[N];

    void init(int x, vector<long long> &a)
    {
        n = x;
        for(int i=1, j; i<=n; i++)
        {
            bit[i] += a[i-1], j = i + (i & -i);
            if(j <= n) bit[j] += bit[i];
        }
    }

    void update(int x, long long dif)
    {
        while(x <= n) bit[x] += dif, x += x & -x;
    }
}
```

```
}

long long query(int l, int r)
{
    if(l != 1) return query(1, r) - query(1, l-1);

    long long ret = 0;
    while(l <= r) ret += bit[r], r -= r & -r;
    return ret;
}
}bm;
```

### 3.2 Distinct Value with BIT

```
#include <bits/stdc++.h>
using namespace std;
using pii = pair<int, int>;
using pti = pair<int, pii>;

int n, q, val[200005];
pti ques[200005];
pii ans[200005];

struct BIT
{
    int bit[200005];
    map<int, int> mp;
    //color, index

    void add(int idx, int dif)
    {
        while(idx <= n)
            bit[idx] += dif, idx += idx & -idx;
    }

    void update(int idx, int color)
    {
        if(mp.find(color) == mp.end())
        {
            add(idx, 1);
            mp.insert({color, idx});
        }
        else
        {
            add(mp[color], -1);
            add(idx, 1);
            mp[color] = idx;
        }
    }

    int query(int l, int r)
    {
        if(l != 1)
            return query(1, r) - query(1, l-1);
        int ret = 0;
        while(l <= r)
            ret += bit[r], r -= r & -r;
        return ret;
    }
};

bool cmp(const pti &x, const pti &y)
{
    if(x.second.second != y.second.second)
        return x.second.second < y.second.second;
    return x.second.first < y.second.first;
}
```

```

int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);

    cin >> n >> q;
    for(int i=1; i<=n; i++)
        cin >> val[i];
    for(int i=0; i<q; i++)
    {
        cin >> ques[i].second.first >> ques[i].second
        .second;
        ques[i].first = i;
    }
    sort(ques, ques+q, cmp);

    int far = 0;
    for(int i=0; i<q; i++)
    {
        while(far < ques[i].second.second)
            a.update(far+1, val[far+1]), far++;
        ans[i] = {ques[i].first, a.query(ques[i].
            second.first, ques[i].second.second)};
    }

    sort(ans, ans+q);
    for(int i=0; i<q; i++)
        cout << ans[i].second << "\n";
}

```

### 3.3 BIT (Howard)

```

// 一維
struct BIT
{
    // 1~n
    int n;
    int bit_num;
    vector<long long> d;
    void init(int size) {
        n = size;
        bit_num = __lg(size) + 1;
        d.resize(1 << bit_num);
        memset(d.data() + 1, 0, sizeof(long long)
            * (1 << bit_num));
    }
    BIT(int size) {
        init(size);
    }
    void ins(int x, long long v) {
        for(; x <= n; x += x & -x) {
            d[x] += v;
        }
    }
    long long query(int x) {
        if(x <= 0) {
            return 0;
        }
        long long s = 0;
        for(; x > 0; x -= x & -x) {
            s += d[x];
        }
        return s;
    }
    long long query_range(int x, int y) {
        if(x > y) {
            return 0;
        }
        long long s = query(y);

```

```

        if(x) {
            s -= query(x - 1);
        }
        return s;
    }
    long long k_th(int k) {
        int now = 0;
        for(int i = bit_num - 1; i >= 0; i--) {
            if(d[now + (1 << i)] < k) {
                k -= d[now + (1 << i)];
                now += 1 << i;
            }
        }
        return now + 1;
    }
};
// 二維
struct BIT
{
    int n;
    int bit_num;
    vector<vector<int>> d;
    void init(int size) {
        n = size;
        bit_num = __lg(size) + 1;
        d.resize(1 << bit_num);
        int ln = d.size();
        for(int i = 0; i < ln; i++) {
            for(int j = 0; j < ln; j++) {
                d[i].push_back(0);
            }
        }
        // memset(d.data() + 1, 0, sizeof(long
        long) * (1 << bit_num));
    }
    BIT(int size) {
        init(size);
    }
    void ins(int x, int y, long long v) {
        for(; x <= n; x += x & -x) {
            for(int j = y; j <= n; j += j & -j) {
                d[x][j] += v;
            }
        }
    }
    long long query(int x, int y) {
        long long s = 0;
        for(; x > 0; x -= x & -x) {
            for(int j = y; j > 0; j -= j & -j){
                s += d[x][j];
            }
        }
        return s;
    }
    long long query_range(int x1, int y1, int x2,
        int y2) {
        return query(x2, y2) - query(x2, y1 - 1)
            - query(x1 - 1, y2) + query(x1 - 1,
                y1 - 1);
    }
};

```

### 3.4 Segment Tree

```

struct segtree
{
    int n, seg[1<<19];

    void init(int x)

```

```

{
    n = 1<<((__lg(x) + 1);
    for(int i=1; i<2*n; i++)
        seg[i] = inf;
}

void update(int x, int val)
{
    x += n;
    seg[x] = val, x /= 2;
    while(x)
        seg[x] = min(seg[2*x], seg[2*x+1]), x
            /= 2;
}

int query(int l, int r)
{
    l += n, r += n;
    int ret = inf;
    while(l < r)
    {
        if(l & 1)
            ret = min(ret, seg[l++]);
        if(r & 1)
            ret = min(ret, seg[--r]);
        l /= 2, r /= 2;
    }
    return ret;
}
}bm;

```

### 3.5 Segment Tree(Howard)

```

using TYPE = int;
const int SIZE = 1 << 19;
// 15:32678 16:65536 17:131072 18:2624144
// 19:524288 20:1048576
struct segment_tree
{
    // start from 1
    TYPE node[SIZE];
    TYPE merge(TYPE a, TYPE b) {
        return min(a, b);
    }
    void pull(int idx) {
        node[idx] = merge(node[idx * 2], node[idx
            * 2 + 1]);
    }
    void init(int L, int R, int idx, vector<TYPE>
        &input) {
        if(L == R) {
            node[idx] = input[L];
            return;
        }
        int now = (L + R) / 2;
        init(L, now, idx * 2, input);
        init(now + 1, R, idx * 2 + 1, input);
        pull(idx);
    }
    TYPE query(int L, int R, int idx, int l, int
        r) {
        if(L >= l && R <= r) return node[idx];
        int now = (L + R) / 2;
        if(r <= now) return query(L, now, idx *
            2, l, r);
        if(l > now) return query(now + 1, R, idx
            * 2 + 1, l, r);
        return merge(query(L, now, idx * 2, l, r)
            , query(now + 1, R, idx * 2 + 1, l, r
            ));
    }
};

```

```

}
void update(int L, int R, int idx, int p,
    TYPE v) {
    if(L == R) {
        node[idx] = v;
        return;
    }
    int now = (L + R) / 2;
    if(p <= now) update(L, now, idx * 2, p, v
        );
    else update(now + 1, R, idx * 2 + 1, p, v
        );
    pull(idx);
}
};

```

### 3.6 Lazy Propagation Segment Tree

```

struct segtree
{
    int n;
    long long tg[2][1<<19], seg[1<<19];
    //0: add, 1: set;

    void init(int x)
    {
        n = 1<<((__lg(x) + 1);
    }

    void set(bool tp, int idx, long long val)
    {
        if(tp == 0)
        {
            (tg[1][idx])? (tg[1][idx] += val): (
                tg[0][idx] += val);
            seg[idx] += val;
        }
        else
        {
            tg[0][idx] = 0, tg[1][idx] = val, seg
                [idx] = val;
        }
    }

    void push(int idx)
    {
        for(int i=__lg(idx); i>0; i--)
        {
            int now = idx>>i;
            if(tg[0][now])
            {
                set(0, 2*now, tg[0][now]/2);
                set(0, 2*now+1, tg[0][now]/2);
                tg[0][now] = 0;
            }
            if(tg[1][now])
            {
                set(1, 2*now, tg[1][now]/2);
                set(1, 2*now+1, tg[1][now]/2);
                tg[1][now] = 0;
            }
        }
    }

    void pull(int idx)
    {
        idx >>= 1;
        while(idx)
        {

```

```

        if(tg[1][idx])
            seg[idx] = tg[1][idx];
        else
            seg[idx] = tg[0][idx] + seg[2*idx] +
                seg[2*idx+1];
        idx /= 2;
    }
}

void update(bool tp, int l, int r, long long
    val)
{
    l += n, r += n;
    int tl = l, tr = r-1, len = 1;
    push(l), push(r-1);
    while(l < r)
    {
        if(l & 1)
            set(tp, l++, val*len);
        if(r & 1)
            set(tp, --r, val*len);
        l /= 2, r /= 2, len *= 2;
    }
    pull(tl), pull(tr);
}

long long query(int l, int r)
{
    l += n, r += n;
    push(l), push(r-1);
    long long ret = 0;
    while(l < r)
    {
        if(l & 1)
            ret += seg[l++];
        if(r & 1)
            ret += seg[--r];
        l /= 2, r /= 2;
    }
    return ret;
}
};

```

### 3.7 Binary Search on Segment Tree

```

struct SEG
{
    int n;
    int seg[1<<19];

    void init(int x)
    {
        n = 1<<(__lg(x) + 1);
    }

    void update(int x, int dif)
    {
        x += n;
        seg[x] += dif, x /= 2;
        while(x)
            seg[x] = max(seg[2*x], seg[2*x+1]), x
                /= 2;
    }

    int query(int g)
    {
        if(seg[1] < g)
            return -1;
        int id = 1;

```

```

        while(id < n)
        {
            if(seg[2*id] >= g)
                id = 2*id;
            else
                id = 2*id+1;
        }

        return id - n;
    }
}bm;

```

### 3.8 Merging on Segment Tree

```

struct segtree
{
    struct node
    {
        long long sum, pre;
        node()
        {
            sum = pre = 0;
        }
    };

    node mer(node x, node y)
    {
        node ret;
        ret.sum = x.sum + y.sum;
        ret.pre = max(x.pre, x.sum + y.pre);
        return ret;
    }

    int n;
    node seg[1<<19];

    void init(int x)
    {
        n = 1<<(__lg(x) + 1);
    }

    void print()
    {
        for(int i=1, j=1; i<=__lg(n)+1; i++)
        {
            for(; j<(1<<i); j++)
                cout << seg[j].sum << " ";
            cout << "\n";
        }
        cout << "\n";
    }

    void update(int x, long long val)
    {
        x += n;
        seg[x].sum = val, seg[x].pre = max(0LL,
            val);
        x /= 2;
        while(x)
            seg[x] = mer(seg[2*x], seg[2*x+1]), x
                /= 2;
    }

    long long query(int l, int r)
    {
        l += n, r += n;
        node retl, retr;
        while(l < r)
        {

```

```

        if(l & 1)
            retl = mer(retl, seg[l++]);
        if(r & 1)
            retr = mer(seg[--r], retr);
        l /= 2, r /= 2;
    }
    return mer(retl, retr).pre;
}
}bm;

```

### 3.9 Segment Tree(Chung)

```

struct seg{
    int L,R;
    int data;
    int lson, rson;
    int chg;
    int real(){
        return data + chg;
    };
};
template<typename T>
struct segtree{
    //declare
    const int maxn;
    int stptr = 1;
    vector<seg> ST;

    // constructor
    segtree(const int &sz): maxn(sz) { ST.resize
        (2*sz); }

    //operation
    void merge(seg &par, seg &lson, seg &rson){
        ;
    }
    seg merge(seg a, seg b){
        seg ret;
        ;
        return ret;
    }
    void push(int idx, int lson, int rson){
        ST[idx].data = ST[idx].real();
        ST[lson].chg += ST[idx].chg;
        ST[rson].chg += ST[idx].chg;
        ST[idx].chg = 0;
    }

    // functions
    void build(int L,int R,int idx,vector<T>& v){
        ST[idx].L = L, ST[idx].R = R;
        if(L + 1 == R){
            ST[idx].data = v[L];
            return;
        }
        int mid = (L+R) / 2;
        int lson = ST[idx].lson = stptr++;
        int rson = ST[idx].rson = stptr++;
        build(L,mid,lson,v);
        build(mid,R,rson,v);
        merge(ST[idx], ST[lson], ST[rson]);
    }
    void single_modify(int pos,int x,int idx){
        if(ST[idx].L + 1 == ST[idx].R){
            ST[idx].data = x;
            return;
        }
        int mid = (ST[idx].L+ST[idx].R) / 2;
        int lson = ST[idx].lson, rson = ST[idx].
            rson;

```

```

        if(pos < mid) single_modify(pos,x,lson);
        else single_modify(pos,x,rson);
        merge(ST[idx], ST[lson], ST[rson]);
    }
    void range_modify(int L,int R,int x,int idx){
        if(ST[idx].L == L && ST[idx].R == R){
            ST[idx].chg += x;
            return;
        }
        int mid = (ST[idx].L + ST[idx].R) / 2;
        int lson = ST[idx].lson, rson = ST[idx].
            rson;
        push(idx, lson, rson);
        if(R <= mid) range_modify(L,R,x,lson);
        else if(L >= mid) range_modify(L,R,x,rson
            );
        else range_modify(L,mid,x,lson),
            range_modify(mid,R,x,rson);
        merge(ST[idx], ST[lson], ST[rson]);
    }
    seg range_query(int L,int R,int idx){
        if(ST[idx].L == L && ST[idx].R == R){
            return ST[idx];
        }
        int mid = (ST[idx].L+ST[idx].R) / 2;
        int lson = ST[idx].lson, rson = ST[idx].
            rson;
        push(idx, lson, rson);
        if(R <= mid) return range_query(L,R,lson)
            ;
        if(L >= mid) return range_query(L,R,rson)
            ;
        return merge(range_query(L,mid,lson),
            range_query(mid,R,rson));
    }
};

int main(){
    const int maxn = 2e5+5;
    int n; cin>>n;
    vector<int> v(n);
    segtree<int> ST(maxn);
    ST.build(0,n,0,v); // [0, n)
    ST.single_modify(3,10,0); // 在 3 的位置 +10
        (0-based)
    ST.range_modify(4,8,5,0); // 在 [4, 8) 的位置
        +10 (0-based)
    cout<<ST.range_query(0,n,0).real()<<"\n"; //
        查詢 [0, n)
    return 0;
}

```

### 3.10 PBDS

```

#include <bits/extc++.h>
using namespace __gnu_pbds;

gp_hash_table<int, int> mp; // map
cc_hash_bable<int, int> mp2;

tree<int,null_type,less<int>,rb_tree_tag,
    tree_order_statistics_node_update> s; // set

int main(){
    // Insert some entries into s.
    s.insert(12); s.insert(505);
    // The order of the keys should be: 12, 505.
    assert(*s.find_by_order(0) == 12);
    assert(*s.find_by_order(3) == 505);
}

```

```
// The order of the keys should be: 12, 505.
assert(s.order_of_key(12) == 0);
assert(s.order_of_key(505) == 1);
// Erase an entry.
s.erase(12);
// The order of the keys should be: 505.
assert(*s.find_by_order(0) == 505);
// The order of the keys should be: 505.
assert(s.order_of_key(505) == 0);
cout << s.size() << endl;
}
// 一種變形寫法
tree<pair<int, int>, null_type, less<pair<int, int>>, rb_tree_tag,
    tree_order_statistics_node_update> s;
void solve()
{
    s.insert({x, i});
    s.erase(s.lower_bound({x, 0}));
    cout << s.order_of_key({x, 0}) + 1 << '\n';
    cout << s.find_by_order(x - 1) -> first << '\n';
    cout << prev(s.lower_bound({x, 0})) -> first << '\n';
    cout << s.lower_bound({x + 1, 0}) -> first << '\n';
}
}
```

### 3.11 Sweep Line

```
const int MAXN = 1e6 + 10;
int X[MAXN << 1];
struct scanline {
    int x, down, up, mark;
    bool operator<(const scanline& other) const {
        return x < other.x;
    }
};

vector<scanline> v;
int node[MAXN << 2], lazy[MAXN << 2]; // for segtree

void Push(int idx, int l, int r) {
    if(lazy[idx]) node[idx] = r - l + 1;
    else if(r != l) node[idx] = node[idx * 2] + node[idx * 2 + 1];
    else node[idx] = 0;
}

void Update(int idx, long long L, long long R, int ul, int ur, int d) { // d -> mark
    if(ul <= L && R <= ur) {
        lazy[idx] += d;
        Push(idx, L, R);
        return;
    }
    int mid = (L + R) / 2;
    if(ul <= mid) Update(idx * 2, L, mid, ul, ur, d);
    if(mid < ur) Update(idx * 2 + 1, mid + 1, R, ul, ur, d);
    Push(idx, L, R);
}

// 給定很多個四邊形，要你求總面積
void solve()
{
    int n;
```

```
cin >> n;
for(int i = 0; i < n; i++) {
    int l, r, d, u;
    cin >> l >> r >> d >> u;
    v.push_back({l, d + 1, u, 1});
    v.push_back({r, d + 1, u, -1});
}
sort(v.begin(), v.end());
long long ans = 0;
int last = 0;
for(int i = 0; i < 2 * n; i++) {
    if(v[i].x != last) {
        ans += (long long)1 * (v[i].x - last) * node[1];
        last = v[i].x;
    }
    Update(1, 1, 1000000, v[i].down, v[i].up, v[i].mark);
}
cout << ans << '\n';
return;
```

### 3.12 LinkedList

```
struct Node{
    Node *next = nullptr, *prev = nullptr;
    int data;
    Node(int x=0): data(x) {}
};

struct LinkedList{
    Node *head, *tail;
    int n = 0;
    LinkedList(){
        // 初始化頭尾偽指標
        head = new Node;
        tail = new Node;
        head -> prev = tail -> next = nullptr;
        head -> next = tail;
        tail -> prev = head;
    }
    Node *insert(Node *node, int data){
        // insert data after node a <-> c ==> a <-> b <-> c
        Node *a = node;
        Node *b = new Node(data);
        Node *c = node -> next;
        a -> next = c -> prev = b;
        b -> prev = a;
        b -> next = c;
        n++;
        return b;
    }
    Node *del(Node *node){
        // 刪除 node 並且回傳前一個指標 a <-> b <-> c ==> a <-> c
        Node *a = node -> prev;
        Node *b = node;
        Node *c = node -> next;
        a -> next = c;
        c -> prev = a;
        n--;
        return a;
    }
    Node *front(){
        return head -> next;
    }
}
```

```

    }
    Node *back(){
        return tail -> prev;
    }
    unsigned size(){
        return n;
    }
}lst;

```

### 3.13 Sparse Table

```

int n, q, a[200005], lg_table[200005],
    sparse_table[19][200005];

void init_lg()
{
    for(int i=0, lg=0; i<=n; i++)
    {
        if(i>=(2<<lg))
            lg++;
        lg_table[i] = lg;
    }
}

void init_sparse()
{
    for(int i=0; i<n; i++)
        sparse_table[0][i] = a[i];
    for(int k=1; k<=lg_table[n]; k++)
        for(int i=0; i+(1<<k)<=n; i++)
            sparse_table[k][i] = min(sparse_table
                [k-1][i], sparse_table[k-1][i
                    +(1<<(k-1))]);
}

int query_sparse(int l, int r)
{
    int k = lg_table[r-l];
    return min(sparse_table[k][l], sparse_table[k
        ][r-(1<<k)]);
}

int main()
{
    cin >> n >> q;
    for(int i=0; i<n; i++)
        cin >> a[i];
    init_lg(), init_sparse();
    while(q-->
    {
        int x, y;
        cin >> x >> y;
        x--, y--;
        cout << query_sparse(x, y+1) << "\n";
    }
}

```

### 3.14 Treap

```

// string reverse
/*
input
7 2
AYBABTU
3 4
4 7
output

```

```

AYAUTBB
*/
mt19937 gen(chrono::steady_clock::now().
    time_since_epoch().count()); // C++
    randomizer
struct Treap
{
    char key;
    int pri, sz, tag;
    Treap *l, *r;
    Treap(char _key) {
        key = _key;
        pri = gen();
        sz = 1;
        tag = 0;
        l = nullptr;
        r = nullptr;
    }
};

Treap *root = nullptr;
string s;

int Size(Treap *x)
{
    return x ? x -> sz : 0;
}

void push(Treap *x)
{
    if(x -> tag) {
        swap(x -> l, x -> r);
        if(x -> l) x -> l -> tag ^= 1;
        if(x -> r) x -> r -> tag ^= 1;
        x -> tag = 0;
    }
}

void pull(Treap *x)
{
    x -> sz = Size(x -> l) + Size(x -> r) + 1;
}

Treap *merge(Treap *a, Treap *b)
{
    if(!a || !b) return a ? a : b;
    if(a -> pri > b -> pri) {
        push(a);
        a -> r = merge(a -> r, b);
        pull(a);
        return a;
    }
    else {
        push(b);
        b -> l = merge(a, b -> l);
        pull(b);
        return b;
    }
}

void build(int n) // string size
{
    for(int i = 0; i < n; i++) {
        root = merge(root, new Treap(s[i]));
    }
}

void splitbykth(Treap *x, int k, Treap *&a, Treap
    *&b)
{
    if(!x) {

```



```

    a = b = nullptr;
    return;
}
push(x);
if(Size(x -> l) + 1 <= k) {
    a = x;
    splitbykth(a -> r, k - Size(x -> l) - 1,
        a -> r, b);
    pull(a);
}
else {
    b = x;
    splitbykth(b -> l, k, a, b -> l);
    pull(b);
}
}

void reverse_update(int a, int b)
{
    Treap *l, *m, *r;
    splitbykth(root, b, l, r);
    splitbykth(l, a - 1, l, m);
    m -> tag = 1;
    root = merge(merge(l, m), r);
}

void output(Treap *x)
{
    if(!x) return;
    push(x);
    output(x -> l);
    cout << x -> key;
    output(x -> r);
}

void solve()
{
    int n, m;
    cin >> n >> m;
    cin >> s;
    build(n);
    for(int i = 0; i < m; i++) {
        int l, r;
        cin >> l >> r;
        reverse_update(l, r);
    }
    output(root);
    cout << '\n';
    return;
}

```

## 4 Graph

### 4.1 DSU

```

struct DSU
{
    int h[N], s[N];

    void init(int n)
    { iota(h, h+n+1, 0), fill(s, s+n+1, 1); }

    int fh(int x)
    { return (h[x]==x? x: h[x]=fh(h[x])); }

    bool mer(int x, int y)
    {
        x = fh(x), y = fh(y);
        if(x == y) return 0;
        if(s[x] < s[y]) swap(x, y);
    }
}

```

```

    s[x] += s[y], s[y] = 0;
    h[y] = x;
    return 1;
}
}bm;

```

### 4.2 LCA

```

vector<int> a[N];
int dep[N], lif[N][20];

void dfs(int x, int p)
{
    dep[x] = dep[p] + 1;
    for(int i: a[x])
        if(i != p)
            dfs(i, x);
}

void init(int n)
{
    dfs(1, 0);
    for(int i=1; (1<<i)<n; i++)
        for(int j=1; j<=n; j++)
            lif[j][i] = lif[lif[j][i-1]][i-1];
}

int up(int x, int k)
{
    int j = 0;
    while(k)
    {
        if(k & 1)
            x = lif[x][j];
        k /= 2, j++;
    }
    return x;
}

int lca(int x, int y, int n)
{
    if(dep[x] > dep[y])
        swap(x, y);
    y = up(y, dep[y] - dep[x]);
    if(x == y)
        return x;
    for(int i=__lg(n); i>=0; i--)
    {
        if(lif[x][i] != lif[y][i])
            x = lif[x][i], y = lif[y][i];
    }
    return lif[x][0];
}

```

### 4.3 HLD

```

const int N = 4e4 + 10;

// add segment tree

vector<int> ed[N];
int sz[N], dep[N], son[N], fa[N];
void dfs_sz(int x, int f, int d){ //當前節點 x ,
    父節點 f, 深度 d
    sz[x] = 1; dep[x] = d; fa[x] = f;
    for(int i: ed[x]){
        if(i == f) continue;
        dfs_sz(i, x, d+1);
    }
}

```

```

        sz[x] += sz[i];
        if(sz[son[x]] < sz[i])    son[x] = i;
    }
}

int top[N]; // 維護每個節點所在的鏈的頂端節點
int dfn[N], rnk[N]; // dfn -> node to seg, rnk ->
                tree to node
int cnt = 1; // index start from 1
void dfs_hld(int x, int f){
    top[x] = (son[fa[x]] == x ? top[fa[x]] : x);
    rnk[cnt] = x;
    dfn[x] = cnt++;
    if(son[x])    dfs_hld(son[x], x);
    for(int i : ed[x]){
        if(i == f || i == son[x])    continue;
        dfs_hld(i, x);
    }
}

int getSum(int u, int v) {
    int ret = 0;
    while(top[u] != top[v]){
        if(dep[top[u]] > dep[top[v]]) {
            int a = dfn[top[u]], b = dfn[u];
            if(a > b) swap(a, b);
            ret += segsum.query(1, n, 1, a, b);
            u = fa[top[u]];
        }
        else {
            int a = dfn[top[v]], b = dfn[v];
            if(a > b) swap(a, b);
            ret += segsum.query(1, n, 1, a, b);
            v = fa[top[v]];
        }
    }
    if(dfn[u] < dfn[v]) {
        ret += segsum.query(1, n, 1, dfn[u], dfn[
            v]);
    }
    else {
        ret += segsum.query(1, n, 1, dfn[v], dfn[
            u]);
    }
    return ret;
}

int getMax(int u, int v) {
    int ret = -1e15;
    while(top[u] != top[v]){
        if(dep[top[u]] > dep[top[v]]) {
            int a = dfn[top[u]], b = dfn[u];
            if(a > b) swap(a, b);
            ret = max(ret, segmax.query(1, n, 1,
                a, b));
            u = fa[top[u]];
        }
        else {
            int a = dfn[top[v]], b = dfn[v];
            if(a > b) swap(a, b);
            ret = max(ret, segmax.query(1, n, 1,
                a, b));
            v = fa[top[v]];
        }
    }
    if(dfn[u] < dfn[v]) {
        ret = max(ret, segmax.query(1, n, 1, dfn[
            u], dfn[v]));
    }
    else {

```

```

        ret = max(ret, segmax.query(1, n, 1, dfn[
            v], dfn[u]));
    }
    return ret;
}

int getLca(int u, int v) {
    while(top[u] != top[v]){
        if(dep[top[u]] > dep[top[v]])
            u = fa[top[u]];
        else
            v = fa[top[v]];
    }
    return dep[u] > dep[v] ? v : u;
}

void solve()
{
    cin >> n;
    for(int i = 0; i < n - 1; i++) {
        int a, b;
        cin >> a >> b;
        ed[a].push_back(b);
        ed[b].push_back(a);
    }
    dfs_sz(1, 1, 0);
    dfs_hld(1, 1);
    vector<int> w(n + 1, 0);
    for(int i = 1; i <= n; i++) {
        cin >> w[dfn[i]];
    }
    segmax.init(1, n, 1, w);
    segsum.init(1, n, 1, w);
    int q;
    cin >> q;
    while(q--) {
        string op;
        cin >> op;
        if(op == "QMAX") {
            int a, b;
            cin >> a >> b;
            cout << getMax(a, b) << '\n';
        }
        else if(op == "QSUM") {
            int a, b;
            cin >> a >> b;
            cout << getSum(a, b) << '\n';
        }
        else {
            int a, t;
            cin >> a >> t;
            segmax.update(1, n, 1, dfn[a], t);
            segsum.update(1, n, 1, dfn[a], t);
        }
    }
    return;
}

```

#### 4.4 Negative Cycle

```

struct EDGE
{
    int first, second;
    long long val;
};

vector<EDGE> e;

```

```

long long dis[N];
int pre[N];
int bellman()
{
    int ret = 0;
    for(EDGE i: e)
    {
        if(dis[i.second] > dis[i.first] + i.val)
        {
            dis[i.second] = dis[i.first] + i.val;
            pre[i.second] = i.first;
            ret = i.second;
        }
    }
    return ret;
}

```

```

bool vis[N];
void f_neg(int x)
{
    vector<int> ret;
    while(!vis[x])
    {
        vis[x] = 1;
        ret.push_back(x);
        x = pre[x];
    }
    ret.push_back(x);
    reverse(ret.begin(), ret.end());
    while(ret.back() != x)
        ret.pop_back();

    cout << "YES\n";
    for(int i: ret)
        cout << i << " ";
    cout << '\n';
}

```

```

int main()
{
    for(int i=1; i<=n-1; i++)
        bellman();
    int fin = bellman();
    if(fin)
        f_neg(fin);
    else
        cout << "NO\n";
}

```

## 4.5 Euler Tour

```

#include <bits/stdc++.h>
using namespace std;
using pii = pair<int, int>;

const int N = 1e5+5, M = 2e5+5;

vector<pii> a[N], edge;
int id[N];
bool vis[M];
vector<int> seq;

void dfs(int x)
{
    for(; id[x]<(int)a[x].size(); id[x]++)
    {
        int i = a[x][id[x]].first, j = a[x][id[x]
        ].second;
        if(vis[j])

```

```

        continue;
        vis[j] = 1;
        dfs(i);
        seq.push_back(i);
    }
}

void no()
{
    cout << "IMPOSSIBLE\n";
    exit(0);
}

int main()
{
    seq.reserve(m + 1);
    dfs(1);
    seq.push_back(1);

    for(int i=1; i<=n; i++)
        if(a[i].size() % 2)
            no();
    for(int i=0; i<m; i++)
        if(!vis[i])
            no();
    if(seq.front() != 1)
        no();

    reverse(seq.begin(), seq.end());
    for(int i: seq)
        cout << i << " ";
    cout << "\n";
}

```

## 4.6 Hamiltonian Routes

```

long long dp[1<<20][20];
vector<int> a[25];
int main()
{
    dp[1][0] = 1;
    for(int i=0; i<(1<<n); i++)
    {
        for(int j=0; j<n; j++)
        {
            if(!dp[i][j])
                continue;
            for(int k: a[j])
            {
                if((i>>k) & 1)
                    continue;
                dp[i | (1<<k)][k] += dp[i][j];
                dp[i | (1<<k)][k] %= mod;
            }
        }
    }
}

```

## 4.7 Dijkstra

```

using pii = pair<int, long long>;
const long long INF = 0x3f3f3f3f3f3f3f3f;

vector<pii> a[N];

struct cmp
{
    bool operator()(pii x, pii y)

```

```

    {
        if(x.second != y.second)
            return x.second > y.second;
        return x.first > y.first;
    }
};

long long dis[N];
priority_queue<pii, vector<pii>, cmp> pq;

int main()
{
    memset(dis, 0x3f, sizeof(dis));

    for(int i=1; i<=n; i++)
        sort(a[i].begin(), a[i].end(), [](pii x, pii y){return x.second < y.second;});

    dis[1] = 0;
    for(pii i: a[1])
    {
        if(dis[i.first] <= i.second)
            continue;
        pq.push(i);
        dis[i.first] = i.second;
    }

    while(!pq.empty())
    {
        pii now = pq.top();
        pq.pop();
        if(dis[now.first] != now.second)
            continue;

        for(pii i: a[now.first])
        {
            long long dis_now = i.second + now.second;
            if(dis[i.first] <= dis_now)
                continue;
            dis[i.first] = dis_now;
            pq.push({i.first, dis[i.first]});
        }
    }

    for(int i=1; i<=n; i++)
        cout << dis[i] << " ";
    cout << "\n";
}

```

## 4.8 Floyd-Warshall

```

const int N = 505;
const long long INF = 0x3f3f3f3f3f3f3f3f;

long long dis[N][N];

int main()
{
    memset(dis, 0x3f, sizeof(dis));
    int n, m, q;
    cin >> n >> m >> q;
    for(int i=1; i<=n; i++)
        dis[i][i] = 0LL;
    for(int i=0, x, y, z; i<m; i++)
    {
        cin >> x >> y >> z;
        dis[x][y] = min(dis[x][y], 1LL * z);
        dis[y][x] = min(dis[y][x], 1LL * z);
    }
}

```

```

    }

    for(int k=1; k<=n; k++)
        for(int i=1; i<=n; i++)
            for(int j=1; j<=n; j++)
                dis[i][j] = min(dis[i][j], dis[i][k] + dis[k][j]);
}

```

## 4.9 SCC kosaraju

```

const int N = 1e5 + 10;
vector<int> ed[N], ed_b[N]; // 反邊
vector<int> SCC(N); // 最後SCC的分組
bitset<N> vis;
int SCC_cnt;
int n, m;
vector<int> pre; // 後序遍歷

void dfs(int x)
{
    vis[x] = 1;
    for(int i: ed[x]) {
        if(vis[i]) continue;
        dfs(i);
    }
    pre.push_back(x);
}

void dfs2(int x)
{
    vis[x] = 1;
    SCC[x] = SCC_cnt;
    for(int i: ed_b[x]) {
        if(vis[i]) continue;
        dfs2(i);
    }
}

void kosaraju()
{
    for(int i = 1; i <= n; i++) {
        if(!vis[i]) {
            dfs(i);
        }
    }
    SCC_cnt = 0;
    vis = 0;
    for(int i = n - 1; i >= 0; i--) {
        if(!vis[pre[i]]) {
            SCC_cnt++;
            dfs2(pre[i]);
        }
    }
}

```

## 5 Flow

### 5.1 Max Flow Dinic

```

const int MXN = 1100; // vertex
struct Dinic{
    struct Edge{ int v,f,re; };
    int n,s,t,level[MXN];
    vector<Edge> ed[MXN];
    void init(int _n, int _s, int _t){
        n = _n; s = _s; t = _t;
        for (int i = 0; i < n; i++) ed[i].clear();
    }
}

```

```

}
void add_edge(int u, int v, int f){
    ed[u].push_back({v, f, (int)ed[v].size()});
    ed[v].push_back({u, 0, (int)ed[u].size() - 1});
}
bool BFS(){
    for (int i=0; i<n; i++) level[i] = -1;
    queue<int> que;
    que.push(s);
    level[s] = 0;
    while (!que.empty()){
        int u = que.front(); que.pop();
        for (auto it : ed[u]){
            if (it.f > 0 && level[it.v] == -1){
                level[it.v] = level[u]+1;
                que.push(it.v);
            }
        }
    }
    return level[t] != -1;
}
int DFS(int u, int nf){
    if (u == t) return nf;
    int res = 0;
    for (auto &it : ed[u]){
        if (it.f > 0 && level[it.v] == level[u]+1){
            int tf = DFS(it.v, min(nf, it.f));
            res += tf; nf -= tf; it.f -= tf;
            ed[it.v][it.re].f += tf;
            if (nf == 0) return res;
        }
    }
    if (!res) level[u] = -1;
    return res;
}
int flow(int res=0){
    while (BFS())
        res += DFS(s, 2147483647);
    return res;
}
}flow;

```

## 5.2 Min Cost Max Flow

// 在最大流量的前題下最小花費

```

struct MinCostMaxFlow{
    typedef int Tcost;
    static const int MAXV = 5e3 + 10;
    static const int INFf = 1000000;
    const int INFc = 1e9;
    struct Edge{
        int v, cap;
        Tcost w;
        int rev;
        Edge(){}
        Edge(int t2, int t3, Tcost t4, int t5)
            : v(t2), cap(t3), w(t4), rev(t5) {}
    };
    int V, s, t;
    vector<Edge> g[MAXV];
    void init(int n, int _s, int _t){
        V = n; s = _s; t = _t;
        for(int i = 0; i <= V; i++) g[i].clear();
    }
    void addEdge(int a, int b, int cap, Tcost w){

```

```

        g[a].push_back(Edge(b, cap, w, (int)g[b].size()));
        g[b].push_back(Edge(a, 0, -w, (int)g[a].size()-1));
    }
    Tcost d[MAXV];
    int id[MAXV], mom[MAXV];
    bool inqu[MAXV];
    queue<int> q;
    pair<int, Tcost> solve(){ // Max Flow : Min Cost
        int mxf = 0; Tcost mnc = 0;
        while(1){
            fill(d, d+V, INFc);
            fill(inqu, inqu+V, 0);
            fill(mom, mom+V, -1);
            mom[s] = s;
            d[s] = 0;
            q.push(s); inqu[s] = 1;
            while(q.size()){
                int u = q.front(); q.pop();
                inqu[u] = 0;
                for(int i = 0; i < (int) g[u].size(); i++){
                    Edge &e = g[u][i];
                    int v = e.v;
                    if(e.cap > 0 && d[v] > d[u]+e.w){
                        d[v] = d[u]+e.w;
                        mom[v] = u;
                        id[v] = i;
                        if(!inqu[v]) q.push(v), inqu[v] = 1;
                    }
                }
            }
            if(mom[t] == -1) break;
            int df = INFf;
            for(int u = t; u != s; u = mom[u])
                df = min(df, g[mom[u]][id[u]].cap);
            for(int u = t; u != s; u = mom[u]){
                Edge &e = g[mom[u]][id[u]];
                e.cap -= df;
                g[e.v][e.rev].cap += df;
            }
            mxf += df;
            mnc += df*d[t];
        }
        return {mxf, mnc};
    }
}flow;

```

## 5.3 Min Cut

```

// add dinic
// 給定兩個點 start 和 end ， 中間會有很多條路，
// 問你說最小要刪除幾條路才可以保證 start 不能到 end
vector<pair<int, int>> ans;
vector<int> vis(MXN);

void dfs(int x)
{
    vis[x] = 1;
    for(auto i : flow.ed[x]) {
        if(vis[i.v] == 1) continue;
        if(vis[i.v] == 0 && i.f > 0) {
            dfs(i.v);

```

```

    }
}
vector<pair<int, int>> G;
void solve()
{
    int n, m;
    cin >> n >> m;
    flow.init(n + 1, 1, n);
    while(m--) {
        int a, b;
        cin >> a >> b;
        flow.add_edge(a, b, 1);
        flow.add_edge(b, a, 1);
        G.push_back({a, b});
        // flow.add_edge(b, a, 1);
    }
    flow.flow();
    dfs(1);
    for(pair<int, int> i : G) {
        if((vis[i.first] == 1 && vis[i.second] ==
            0) || (vis[i.first] == 0 && vis[i.
                second] == 1)) ans.push_back(i);
    }
    cout << ans.size() << '\n';
    for(auto i : ans) {
        cout << i.first << ' ' << i.second << '\n';
    }

    return;
}
}

```

## 5.4 Maximum Bipartite Matching

```

// add dinic
// 左邊有 n 個點，右邊有 m 個點，共有 e 條邊
void solve()
{
    int n, m, e;
    cin >> n >> m >> e;
    flow.init(n + m + 10, n + m + 1, n + m + 2);
    while(e--) {
        int a, b;
        cin >> a >> b;
        flow.add_edge(a, b + n, 1);
    }
    for(int i = 1; i <= n; i++) {
        flow.add_edge(n + m + 1, i, 1);
    }
    for(int i = n + 1; i <= n + m; i++) {
        flow.add_edge(i, n + m + 2, 1);
    }
    cout << flow.flow() << '\n';
    return;
}
}

```

## 6 Math

### 6.1 Matrix

```

const int MOD = 1e9 + 7;
typedef vector<vector<ll>> matrix;
matrix operator*(matrix A, matrix B)
{
    const int n = A.size(), m = B.size(), o = B
        [0].size();
    matrix ret(n, vector<ll>(o, 0));
    for(int i = 0; i < n; i++) {

```

```

        for(int j = 0; j < o; j++) {
            for(int k = 0; k < m; k++) {
                ret[i][j] += A[i][k] * B[k][j];
                ret[i][j] %= MOD;
            }
        }
    }
    return ret;
}

matrix power(matrix A, int n)
{
    matrix ans(A.size(), vector<ll>(A.size(), 0));
    ;
    for(int i = 0; i < A.size(); i++) ans[i][i] =
        1;

    while(n) {
        if(n & 1) ans = ans * A;
        A = A * A;
        n >>= 1;
    }
    return ans;
}

```

### 6.2 Combination

```

int c[n + 1][n + 1];
for(int i = 0; i < n; i++) {
    c[i][i] = 1;
    c[i][0] = 1;
}
for(int i = 2; i <= n; i++) {
    for(int j = 1; j <= i - 1; j++) {
        c[i][j] = c[i - 1][j] + c[i - 1][j - 1];
        c[i][i - j] = c[i - 1][j] + c[i - 1][j -
            1];
    }
}

//////////
const int N = 1e6 + 10;
const int mod = 1e9 + 7;
int fac[N], fac_inv[N];

void init()
{
    fac[0] = 1;
    for(int i = 1; i < N; i++) {
        fac[i] = fac[i - 1] * i % mod;
    }
    fac_inv[N - 1] = power(fac[N - 1], mod - 2,
        mod);
    for(int i = N - 2; i >= 0; i--) {
        fac_inv[i] = fac_inv[i + 1] * (i + 1) %
            mod;
    }
}

int C(int n, int m)
{
    return fac[n] * fac_inv[m] % mod * fac_inv[n
        - m] % mod;
}

```

### 6.3 Exponentiation

```

int power(int a, int b, int m) // (a ^ b) % m
{

```

```

    a %= m;
    int ret = 1 % m;
    for(; b; b >>= 1) {
        if(b & 1) ret = ret * a % m;
        a = a * a % m;
    }
    return ret;
}

```

## 6.4 Unique factorization domain

```

const int N = 1e4 + 10;
vector<int> prime;
vector<int> vis(N);
vector<int> prime_count(N); // 算最後的次方

void add_integer(int n, int d)
{
    for(int i = 0; i < prime.size(); i++) {
        while(n % prime[i] == 0) {
            n /= prime[i];
            prime_count[i] += d;
        }
        if(n == 1) break;
    }
}

void add_factorial(int n, int d) // n ^ d
{
    for(int i = 1; i <= n; i++) {
        add_integer(i, d);
    }
}

void solve()
{
    for(int i = 2; i <= 10000; i++) {
        if(!vis[i]) prime.push_back(i);
        for(int j = i * 2; j <= 10000; j += i)
            vis[j] = 1;
    }
    // main code

    // 對於很多數字一起相乘可使用
    // add_factorial->階層

    // 求解
    double ans = 1;
    for(int i = 0; i < prime.size(); i++) ans *=
        pow(prime[i], prime_count[i]);
    cout << fixed << setprecision(5) << ans << '\n';
    return;
}

```

## 6.5 Quick Mul

```

ll qMul(ll a, ll b, ll mod){
    ll ans = 0;
    while(b){
        if(b&1) ans = (ans+a)%mod;
        a = (a+a)%mod;
        b>>=1;
    }
    return ans;
}

// O(1)

```

```

ll qMul(ll x, ll y, ll mod){
    ll ret = x * y - (ll)((long double)x / mod *
        y) * mod;
    return ret<0?ret+mod:ret;
}

```

## 6.6 Miller Rabin

```

// n < 4,759,123,141          3 : 2, 7, 61
// n < 1,122,004,669,633     4 : 2, 13, 23,
//                          1662803
// n < 3,474,749,660,383     6 : pirmses <=
//                          13
// n < 2^64                  7 :
// 2, 325, 9375, 28178, 450775, 9780504,
// 1795265022

ll magic[] = {};
bool witness(ll a, ll n, ll u, int t){
    if(!a) return 0;
    ll x = power(a, u, n);
    for(int i = 0; i < t; i++) {
        ll nx = qMul(x, x, n);
        if(nx == 1 && x != 1 && x != n - 1) return 1;
        x = nx;
    }
    return x != 1;
}

bool miller_rabin(ll n) {
    int s = (magic size); // init
    // iterate s times of witness on n
    if(n < 2) return 0;
    if(!(n & 1)) return n == 2;
    ll u = n - 1; int t = 0;
    // n-1 = u*2^t
    while(!(u & 1)) u >>= 1, t++;
    while(s--){
        ll a = magic[s] % n;
        if(witness(a, n, u, t)) return 0;
    }
    return 1;
}

int c;
cin >> c;
if(miller_rabin(c)){
    cout << "prime\n";
}
else{
    cout << "not_prime\n";
}

```

## 6.7 Pollard's Rho

```

ll add(ll x, ll y, ll p) {
    return (x + y) % p;
}

ll f(ll x, ll mod) { return add(qMul(x, x, mod), 1,
    mod); }

ll pollard_rho(ll n) {
    if(!(n & 1)) return 2;
    while(true) {
        ll y = 2, x = rand() % (n - 1) + 1, res = 1;
        for(int sz = 2; res == 1; sz *= 2) {
            for(int i = 0; i < sz && res <= 1; i++) {

```

```

        x = f(x, n);
        res = __gcd(llabs(x - y), n);
    }
    y = x;
}
if (res != 0 && res != n) return res;
}
}

vector<ll> ret;

void fact(ll x) {
    if(miller_rabin(x)) {
        ret.push_back(x);
        return;
    }
    ll f = pollard_rho(x);
    fact(f); fact(x / f);
}

```

## 6.8 Sum of Powers Formulas and Expansions

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

$p = 1 :$

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

$p = 2 :$

$$\sum_{i=0}^n i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{2n^3 + 3n^2 + n}{6}$$

$p = 3 :$

$$\sum_{i=0}^n i^3 = \left( \frac{n(n+1)}{2} \right)^2 = \frac{n^4 + 2n^3 + n^2}{4}$$

$p = 4 :$

$$\sum_{i=0}^n i^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30} = \frac{6n^5 + 15n^4 + 10n^3 - n}{30}$$

$p = 5 :$

$$\sum_{i=0}^n i^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12} = \frac{2n^6 + 6n^5 + 5n^4 - n^2}{12}$$

## 6.9 ax+by=gcd(a,b)

```

pair<int,int> extgcd(int a, int b){
    if (b == 0) return {1, 0};
    int k = a / b;
    pair<int, int> p = extgcd(b, a - k * b);
    return {p.second, p.first - k * p.second};
}

```

## 6.10 Discrete Sqrt

```

void calcH(long long &t, long long &h, const long
    long p) {
    long long tmp=p-1;
    for(t = 0; (tmp & 1) == 0; tmp /= 2) t++;
    h = tmp;
}
// solve equation x^2 mod p = a
bool solve(long long a, long long p, long long &x
    , long long &y) {

```

```

if(p == 2) {
    x = y = 1;
    return true;
}
int p2 = p / 2, tmp = power(a, p2, p);
if (tmp == p - 1) return false;
if ((p + 1) % 4 == 0) {
    x = power(a, (p + 1) / 4, p);
    y = p - x;
    return true;
}
else {
    long long t, h, b, pb;
    calcH(t, h, p);
    if (t >= 2) {
        do {
            b = rand() % (p - 2) + 2;
        } while (power(b, p / 2, p) != p - 1);
        pb = power(b, h, p);
    }
    int s = power(a, h / 2, p);
    for (int step = 2; step <= t; step++) {
        int ss = (((long long)(s * s) % p) * a) % p;
        for(int i = 0; i < t - step; i++) ss = qMul
            (ss, ss, p);
        if(ss + 1 == p) s = (s * pb) % p;
        pb = ((long long)pb * pb) % p;
    }
    x = ((long long)s * a) % p;
    y = p - x;
}
return true;
}

```

## 6.11 Prime

```

/* 12721 13331 14341 75577 123457 222557 556679
999983 1097774749 1076767633 100102021
999997771 1001010013 1000512343 987654361
999991231 999888733 98789101 98777773
999991921 1010101333 1010102101 1000000000039
100000000000037 2305843009213693951
4611686018427387847 9223372036854775783
18446744073709551557 */

```

## 6.12 Gauss

```

// O(n^3)
const double EPS = 1e-9;
const int INF = 100; // it doesnt actually have
    to be infinity or a big number
// return meaning
// one solution : 1
// inf solution : INF
// no solution : 0

int gauss (vector < vector<double> > a, vector<
    double> & ans) {
    int n = (int) a.size();
    int m = (int) a[0].size() - 1;

    vector<int> where (m, -1);
    for (int col=0, row=0; col<m && row<n; ++col)
        {
            int sel = row;
            for (int i=row; i<n; ++i)

```



```

        if (abs (a[i][col]) > abs (a[sel][col]
            ))
            sel = i;
        if (abs (a[sel][col]) < EPS)
            continue;
        for (int i=col; i<=m; ++i)
            swap (a[sel][i], a[row][i]);
        where[col] = row;

        for (int i=0; i<n; ++i)
            if (i != row) {
                double c = a[i][col] / a[row][col]
                    ];
                for (int j=col; j<=m; ++j)
                    a[i][j] -= a[row][j] * c;
            }
        ++row;
    }

    ans.assign (m, 0);
    for (int i=0; i<m; ++i)
        if (where[i] != -1)
            ans[i] = a[where[i]][m] / a[where[i]
                ]][i];
    for (int i=0; i<n; ++i) {
        double sum = 0;
        for (int j=0; j<m; ++j)
            sum += ans[j] * a[i][j];
        if (abs (sum - a[i][m]) > EPS)
            return 0;
    }

    for (int i=0; i<m; ++i)
        if (where[i] == -1)
            return INF;
    return 1;
}

```

```

        a[k] = w * x;
    }
    theta = (theta * 2) % MAXN;
}
int i = 0;
for (int j = 1; j < n - 1; j++) {
    for (int k = n >> 1; k > (i ^ k); k >>=
        1);
    if (j < i) swap(a[i], a[j]);
}
if(inv) {
    for (i = 0; i < n; i++) a[i] /= n;
}
}
cplx arr[MAXN + 1];
inline void mul(int _n, long long a[], int _m, long
    long b[], long long ans[]){
    int n=1, sum = _n + _m - 1;
    while(n < sum) n <= 1;
    for(int i = 0; i < n; i++) {
        double x = (i < _n ? a[i] : 0), y = (i < _m
            ? b[i] : 0);
        arr[i] = complex<double>(x + y, x - y);
    }
    fft(n, arr);
    for(int i = 0; i < n; i++) arr[i] = arr[i] * arr[
        i];
    fft(n, arr, true);
    for(int i=0; i<sum; i++) ans[i] = (long long int)
        (arr[i].real() / 4 + 0.5);
}

long long a[MAXN];
long long b[MAXN];
long long ans[MAXN];
int a_length;
int b_length;

```

## 6.13 FFT

```

// 多項式乘法a*b=ans
const int MAXN = 262144;
// (must be 2^k)
// 262144, 524288, 1048576, 2097152, 4194304
// before any usage, run pre_fft() first
typedef long double ld;
typedef complex<ld> cplx; //real() ,imag()
const ld PI = acos(-1);
const cplx I(0, 1);
cplx omega[MAXN+1];
void pre_fft(){
    for(int i=0; i<=MAXN; i++) {
        omega[i] = exp(i * 2 * PI / MAXN * I);
    }
}
// n must be 2^k
void fft(int n, cplx a[], bool inv=false){
    int basic = MAXN / n;
    int theta = basic;
    for (int m = n; m >= 2; m >>= 1) {
        int mh = m >> 1;
        for (int i = 0; i < mh; i++) {
            cplx w = omega[inv ? MAXN - (i *
                theta % MAXN) : i * theta % MAXN
                ];
            for (int j = i; j < n; j += m) {
                int k = j + mh;
                cplx x = a[j] - a[k];
                a[j] += a[k];
            }
        }
    }
}

```

## 6.14 Theorem

- Lucas' Theorem :  
For  $n, m \in \mathbb{Z}^*$  and prime  $P$ ,  $C(m, n) \bmod P = \prod (C(m_i, n_i))$  where  $m_i$  is the  $i$ -th digit of  $m$  in base  $P$ .
- Stirling approximation :  
$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\frac{1}{12n}}$$
- Stirling Numbers(permutation  $|P| = n$  with  $k$  cycles):  
 $S(n, k) = \text{coefficient of } x^k \text{ in } \prod_{i=0}^{n-1} (x + i)$
- Stirling Numbers(Partition  $n$  elements into  $k$  non-empty set):  
$$S(n, k) = \frac{1}{k!} \sum_{j=0}^k (-1)^{k-j} \binom{k}{j} j^n$$
- Pick' s Theorem :  $A = i + b/2 - 1$   
 $A$ : Area,  $i$ : grid number in the inner,  $b$ : grid number on the side
- Catalan number :  $C_n = \binom{2n}{n} / (n+1)$   
$$C_n^{n+m} - C_{n+1}^{n+m} = (m+n)! \frac{n-m+1}{n+1} \quad \text{for } n \geq m$$
  
$$C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$$
  
$$C_0 = 1 \quad \text{and} \quad C_{n+1} = 2 \binom{2n+1}{n+2} C_n$$
  
$$C_0 = 1 \quad \text{and} \quad C_{n+1} = \sum_{i=0}^n C_i C_{n-i} \quad \text{for } n \geq 0$$
- Euler Characteristic:  
planar graph:  $V - E + F - C = 1$   
convex polyhedron:  $V - E + F = 2$   
 $V, E, F, C$ : number of vertices, edges, faces(regions), and components

- Kirchhoff's theorem :  
 $A_{ii} = \deg(i), A_{ij} = (i, j) \in E ? -1 : 0$ , Deleting any one row, one column, and cal the  $\det(A)$
- Polya' theorem ( $c$  is number of color,  $m$  is the number of cycle size):  
 $(\sum_{i=1}^m c^{\gcd(i,m)})/m$
- Burnside lemma:  
 $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$
- 錯排公式: ( $n$  個人中, 每個人皆不再原來位置的組合數):  
 $dp[0] = 1; dp[1] = 0;$   
 $dp[i] = (i-1) * (dp[i-1] + dp[i-2]);$
- Bell 數 (有  $n$  個人, 把他們拆組的方法總數):  
 $B_0 = 1$   
 $B_n = \sum_{k=0}^n s(n, k) \quad (\text{second-stirling})$   
 $B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k$
- Wilson's theorem :  
 $(p-1)! \equiv -1 \pmod{p}$
- Fermat's little theorem :  
 $a^p \equiv a \pmod{p}$
- Euler's totient function:  
 $A^{B^C} \pmod{p} = \text{pow}(A, \text{pow}(B, C, p-1)) \pmod{p}$
- 歐拉函數降幂公式:  
 $A^B \pmod{C} = A^{B \pmod{\phi(C)} + \phi(C)} \pmod{C}$
- 6 的倍數:  
 $(a-1)^3 + (a+1)^3 + (-a)^3 + (-a)^3 = 6a$

## 7 Geometry

### 7.1 Structure

```
using TYPE = int;
struct Pt {
    TYPE x, y;
};

struct Line{
    Pt st, ed;
};

struct Circle{
    Pt o; // center
    TYPE r; // radius
};

struct poly{
    int n; // n 邊形
    vector<Pt> pts;
};
```

### 7.2 Vector

```
using TYPE = int;
struct Pt {
    TYPE x, y;
    Pt(TYPE _x = 0, TYPE _y = 0):x(_x), y(_y) {}

    Pt operator+(const Pt &a){
        return Pt(x + a.x, y + a.y);
    }
    Pt operator-(const Pt &a){
        return Pt(x - a.x, y - a.y);
    }
    Pt operator*(const TYPE &a){
        return Pt(x * a, y * a);
    }
};
```

```
Pt operator/(const TYPE &a){
    return Pt(x / a, y / a);
}

TYPE operator*(const Pt &a){ //計算幾何程式碼
    中內積通常用*表示
    return x * a.x + y * a.y;
}

TYPE operator^(const Pt &a){ //計算幾何程式碼
    中外積通常用^表示
    return x * a.y - y * a.x;
}

bool operator<(const Pt &a) const { // 判斷兩
    點座標 先比 x 再比 y
    return x < a.x || (x == a.x && y < a.y);
}

};
```

### 7.3 Collinear

```
bool collinearity(const Pt& a, const Pt& b, const
    Pt& c){
    return (b - a) ^ (c - a) == 0;
}

bool collinearity(const Pt& a, const Pt& b, const
    Pt& c){
    return (b - a) ^ (c - a) < EPS;
}

};
```

### 7.4 InLine

```
bool inLine(const Pt& p, const Line& li){
    return collinearity(li.st, li.ed, p) && (li.
        st - p) * (li.ed - p) < 0;
}

bool inLine(const Pt& p, const Line& li){
    return collinearity(li.st, li.ed, p) && (li.
        st - p) * (li.ed - p) < EPS;
}

};
```

### 7.5 Convex Hull

```
struct Pt {
    int x, y;
    Pt(){}
    Pt(int _x, int _y) {
        x = _x, y = _y;
    }
    Pt operator-(const Pt &a) {
        return Pt(x - a.x, y - a.y);
    }
    bool operator<(const Pt &a) const {
        return x < a.x || (x == a.x && y < a.y);
    }
};

int cross(Pt& o, Pt& a, Pt& b) {
    Pt lhs = o - a, rhs = o - b;
    return lhs.x * rhs.y - lhs.y * rhs.x;
}

vector<Pt> convex_hull(vector<Pt> hull){
    sort(hull.begin(), hull.end());
    int top = 0;
    vector<Pt> stk;
    for(int i = 0; i < hull.size(); i++) {
```

```

        while(top >= 2 && cross(stk[top - 2], stk
            [top - 1], hull[i]) <= 0)
            stk.pop_back(), top--;
        stk.push_back(hull[i]);
        top++;
    }
    for(int i = hull.size() - 2, t = top + 1; i
        >= 0; i--) {
        while(top >= t && cross(stk[top - 2], stk
            [top - 1], hull[i]) <= 0)
            stk.pop_back(), top--;
        stk.push_back(hull[i]);
        top++;
    }
    stk.pop_back();
    return stk;
}

for(int i = 1, l = 0, r = 0; i < n; i++) {
    if(i <= r && z[i - 1] < r - i + 1) {
        z[i] = z[i - 1];
    }
    else {
        z[i] = max(0, r - i + 1);
        while(i + z[i] < n && s[z[i]] == s[i
            + z[i]]) z[i]++;
    }
    if(i + z[i] - 1 > r) l = i, r = i + z[i]
        - 1;
}
return z;
}

```

## 8 String

### 8.1 Trie

```

void build(string &s)
{
    int n = s.size();
    for(int i=0, v=0; i<n; i++)
    {
        pii &now = a[v][s[i]- 'a'];
        if(now.first != -1)
            v = now.first;
        else
            v = now.first = ++idx;
        if(i == n-1)
            now.second++;
    }
}

```

### 8.2 Hash

```

const int N = 1e6+5;
const long long A = 27, B = 915660423;
long long pre[N], pw[N];

long long f(int l, int r)
{
    return (pre[r] - pre[l-1]*pw[r-l+1]%B + B) %
        B;
}

int main()
{
    s = " " + s;
    pw[0] = 1;
    for(int i=1; i<=n; i++)
    {
        pre[i] = (pre[i-1]*A + (s[i]- 'a'+1)) % B;
        pw[i] = pw[i-1] * A % B;
    }
}

```

### 8.3 Z Function

```

// 一個字串的后綴可以和字串本身匹配到多長從第i項
// 開始一路到n-1
inline vector<int> z_function(string s)
{
    int n = s.size();
    vector<int> z(n, 0);
}

```