Algorithm template by Uni

1. Binary Indexed Tree (Fenwick Tree) 2
2. Interval Tree 4
3. Interval Tree 2D 14
4. KMP 16
5. Binary Search 17
6. Union Find 18
7. RMQ 18
8. Manacher 19
9. Hash 20
10. LIS 21
11. LCS 21
12. Date 22

1. Binary Indexed Tree (Fenwick Tree)

```
/// 1. Single point
// memset(c,0,sizeof(c)) before use
int c[MX];

// n -> update place, v -> update value
void U(int x, int v) {
    for (; x <= n; x += (x & -x))
        c[x] += v;
}

// get the sum from 1 to n (BIT starts from 1)
int Q(int x) {
    int r = 0;
    for (; x > 0; x -= (x & -x))
        r += c[x];
    return r;
}

// 2D
int c[MAXX][MAXY];

// update at (x,y)
void U(int x, int y, int v) {
    for (; x <= MAXX; x += (x & -x))
        for (int i = y; i <= MAXY; i += (i & -i))
            c[x][i] += v;
}

// get the sum from (1,1) to (x,y)
int Q(int x, int y) {
    int r = 0;
    for (; x > 0; x -= (x & -x))
        for (int i = y; i > 0; i -= (i & -i))
            r += c[x][i];
    return r;
}

/// 2. Update interval [l,r]
// U(l - 1, -c), U(r, c);
void U(int x, int v) {
    for (; x > 0; x -= (x & -x))
        b[x] += v;
}

// Q(x) the value of a[x]
int Q(int x) {
    int r = 0;
    for (; x <= n; x += (x & -x))
        r += b[x];
    return r;
}

/// 3. Update & query Interval
// U(r, c); if (l > 1) U(l - 1, -c);
void U(int x, int v) {
    if (x == 0) return;
    for (int i = x; i > 0; i -= (i & -i))
```

```
        b[i] += v;
    for (int i = x; i <= n; i += (i & -i))
        c[i] += x * v;
}

// Q(r) - Q(l - 1);
int QB(int x) {
    int r = 0;
    for (; x <= n; x += (x & -x))
        r += b[x];
    return r;
}

int QC(int x) {
    int r = 0;
    for (; x > 0; x -= (x & -x))
        r += c[x];
    return r;
}

int Q(int x) {
    if (x)
        return QB(x) * x + QC(x - 1);
    return 0;
}

/// 4.
// Inversion pairs with duplicate and 10^9
#include <cstdio>
#include <cstring>
#include <algorithm>
#define MX 99999
using namespace std;
typedef long long ll;
struct P {
    int v, w, i;
} p[MX];
int c[MX];
int x(P a, P b) {
    return a.v < b.v;
}
int y(P a, P b) {
    return a.i > b.i;
}
void U(int i) {
    for (; i < MX; i += i & -i)++c[i];
}
int Q(int i) {
    int r = 0;
    for (; i > 0; i -= i & -i)r += c[i];
    return r;
}
int main() {
    int n, k;
    while (~scanf("%d", &n)) {
        ll a = 0; k = 1; memset(c, 0, sizeof(c));
        for (int i = 0; i < n; ++i)scanf("%d", &p[i].v), p[i].i = i;
        sort(p, p + n, x);
```

```
        p[0].w = k;
        for (int i = 1; i < n; ++i)p[i].w = p[i].v == p[i - 1].v ? k : ++k;
        sort(p, p + n, y);
        for (int i = 0; i < n; ++i)a += Q(p[i].w - 1), U(p[i].w);
        printf("%I64d\n", a);
    }
    return 0;
}
```

2. Interval Tree

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

#define MX 1024000
#define ls l,m,n<<1     // lson
#define rs m+1,r,n<<1|1 // rson
#define lc n<<1         // lchild
#define rc n<<1|1       // rchild
using namespace std;

int num[MX], sum[MX << 2], ma[MX << 2], mi[MX << 2], add[MX << 2];
int N, L, R, V, X;

void up(int n) {
    sum[n] = sum[lc] + sum[rc];
    ma[n] = max(ma[lc], ma[rc]);
    mi[n] = min(mi[lc], mi[rc]);
}

void down(int n, int m) {
    if (add[n]) {
        add[lc] += add[n];
        add[rc] += add[n];
        sum[lc] += add[n] * (m - (m >> 1));
        sum[rc] += add[n] * (m >> 1);
        add[n] = 0;
    }
}

void B(int l = 1, int r = N, int n = 1) {
    add[n] = 0;
    if (l == r) {
        scanf("%d", &num[l]);
        sum[n] = ma[n] = mi[n] = num[l];
        return;
    }
    int m = (l + r) >> 1;
    B(ls), B(rs), up(n);
}

// Update position x
// Prepare: X, V
void U(int l = 1, int r = N, int n = 1) {
    if (l == r) {
        sum[n] = ma[n] = mi[n] = num[l] = V; // or addition
        return;
    }
```

```
    down(n, r - l + 1);
    int m = (l + r) >> 1;
    if (X <= m) U(ls);
    else U(rs);
    up(n);
}

// Update [L,R]
// Prepare: L, R, V
void U(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R) {
        add[n] += V, sum[n] += V * (r - l + 1);
        return;
    }
    down(n, r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) U(ls);
    if (m < R) U(rs);
    up(n);
}

// Query interval [L,R]
// Prepare: L, R
int Q(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R) {
        return sum[n];
    }
    down(n, r - l + 1);
    int ans = 0, m = (l + r) >> 1;
    if (L <= m) ans += Q(ls);
    // ans = max(ans, Q(ls));
    if (m < R) ans += Q(rs);
    // ans = max(ans, Q(rs));
    return ans;
}

/* LCIS */
#include <cstdio>
#include <algorithm>
#define MX 102400
#define lson l,m,n<<1
#define rson m+1,r,n<<1|1
#define lc n<<1
#define rc n<<1|1
using namespace std;
int N, X, V, num[MX], mm[MX << 2], lm[MX << 2], rm[MX << 2];
void B(int l = 1, int r = N, int n = 1) {
    if (l == r) {
        scanf("%d", &num[l]);
        lm[n] = rm[n] = mm[n] = 1;
        return;
    }
    int m = (l + r) >> 1, k = r - l + 1;
    B(lson), B(rson);
    lm[n] = lm[lc];
    if (lm[lc] == k - (k >> 1) && num[m] < num[m + 1])
        lm[n] += lm[rc];
    rm[n] = rm[rc];
```

```
    if (rm[rc] == (k >> 1) && num[m] < num[m + 1])
        rm[n] += rm[lc];
    mm[n] = max(mm[lc], mm[rc]);
    if (num[m] < num[m + 1])
        mm[n] = max(mm[n], rm[lc] + lm[rc]);
}
void U(int l = 1, int r = N, int n = 1) {
    if (l == r) {
        num[l] = V;
        return;
    }
    int m = (l + r) >> 1, k = r - l + 1;
    if (X <= m) U(lson);
    else U(rson);
    lm[n] = lm[lc];
    if (lm[lc] == k - (k >> 1) && num[m] < num[m + 1])
        lm[n] += lm[rc];
    rm[n] = rm[rc];
    if (rm[rc] == (k >> 1) && num[m] < num[m + 1])
        rm[n] += rm[lc];
    mm[n] = max(mm[lc], mm[rc]);
    if (num[m] < num[m + 1])
        mm[n] = max(mm[n], rm[lc] + lm[rc]);
}
int QL(int L, int R, int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return lm[n];
    int m = (l + r) >> 1;
    if (L > m) return QL(L, R, rson);
    if (R <= m) return QL(L, R, lson);
    int ans = QL(L, m, lson), k = m - L + 1;
    if (ans == k && num[m] < num[m + 1])
        ans += QL(m + 1, R, rson);
    return ans;
}
int QR(int L, int R, int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return rm[n];
    int m = (l + r) >> 1;
    if (L > m) return QR(L, R, rson);
    if (R <= m) return QR(L, R, lson);
    int ans = QR(m + 1, R, rson), k = R - m;
    if (ans == k && num[m] < num[m + 1])
        ans += QR(L, m, lson);
    return ans;
}
int Q(int L, int R, int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R)
        return mm[n];
    int m = (l + r) >> 1;
    if (L > m) return Q(L, R, rson);
    if (R <= m) return Q(L, R, lson);
    int ans = max(Q(L, m, lson), Q(m + 1, R, rson));
    if (num[m] < num[m + 1])
        ans = max(ans, QR(L, m, lson) + QL(m + 1, R, rson));
    return ans;
}
int main() {
```

```
    int t, m, L, R;
    char o[9];
    scanf("%d", &t);
    while (t--) {
        scanf("%d%d", &N, &m), B();
        while (m--) {
            scanf("%s", o);
            if (o[0] == 'Q') {
                scanf("%d%d", &L, &R), printf("%d\n", Q(L + 1, R + 1));
            } else {
                scanf("%d%d", &X, &V), ++X, U();
            }
        }
    }
    return 0;
}

/* update and set value */
#include <cstdio>
#include <cstring>
#include <algorithm>
#define MX 102400
#define ls l,m,n<<1
#define rs m+1,r,n<<1|1
#define lc n<<1
#define rc n<<1|1
using namespace std;
typedef long long ll;
ll sum[MX << 2], add[MX << 2], V;
bool se[MX << 2];
int L, R, N;
inline void up(int n) {
    sum[n] = sum[lc] + sum[rc];
}
void B(int l = 1, int r = N, int n = 1) {
    add[n] = 0;
    se[n] = false;
    if (l == r) {
        sum[n] = 0;
        return;
    }
    int m = (l + r) >> 1;
    B(ls), B(rs), up(n);
}
void down(int n, int m) {
    if (se[n]) {
        se[lc] = se[rc] = se[n];
        sum[lc] = sum[rc] = 0;
        se[n] = false;
        add[lc] = add[rc] = 0;
    }
    if (add[n]) {
        add[lc] += add[n];
        add[rc] += add[n];
        sum[lc] += add[n] * (m - (m >> 1));
        sum[rc] += add[n] * (m >> 1);
        add[n] = 0;
    }
```

```
}
void U(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R) {
        add[n] += V, sum[n] += V * (r - l + 1);
        return;
    }
    down(n, r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) U(ls);
    if (m < R) U(rs);
    up(n);
}
void S(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R) {
        se[n] = true, add[n] = false, sum[n] = 0;
        return;
    }
    down(n, r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) S(ls);
    if (m < R) S(rs);
    up(n);
}
ll Q(int l = 1, int r = N, int n = 1) {
    if (L <= l && r <= R) {
        return sum[n];
    }
    down(n, r - l + 1);
    ll ans = 0, m = (l + r) >> 1;
    if (L <= m) ans += Q(ls);
    if (m < R) ans += Q(rs);
    return ans;
}
int main() {
    int t, m;
    scanf("%d", &t);
    while (t--) {
        scanf("%d%d", &N, &m), B();
        ll ans = 0;
        int p = 0, q = 0;
        while (m--) {
            scanf("%d", &q);
            L = 1, R = N, V = q - p;
            U(), p = q;
            scanf("%d%d", &L, &R);
            ans += Q(), S();
        }
        printf("%lld\n", ans);
    }
    return 0;
}

/* binary reverse */
#include <bits/stdc++.h>
#define MX 1111111
#define lson l,m,n<<1
#define rson m+1,r,n<<1|1
using namespace std;
```

```
int num[MX], sum[MX << 2], add[MX << 2], inv[MX << 2];
char s[111];
inline void pushUp(int n)
{
    sum[n] = sum[n << 1] + sum[n << 1 | 1];
}
void build(int l, int r, int n)
{
    add[n] = -1, inv[n] = 0;
    if (l == r)
    {
        sum[n] = num[l];
        return;
    }
    int m = (l + r) >> 1;
    build(lson), build(rson), pushUp(n);
}
void flip(int x)
{
    if (inv[x] % 2) ++inv[x];
}
void pushDown(int n, int m)
{
    if (add[n] != -1)
    {
        add[n << 1] = add[n << 1 | 1] = add[n];
        flip(n << 1), flip(n << 1 | 1);
        sum[n << 1] = add[n] * (m - (m >> 1));
        sum[n << 1 | 1] = add[n] * (m >> 1);
        add[n] = -1;
    }
    if (inv[n] && inv[n] % 2)
    {
        ++inv[n << 1];
        ++inv[n << 1 | 1];
        sum[n << 1] = (m - (m >> 1)) - sum[n << 1];
        sum[n << 1 | 1] = (m >> 1) - sum[n << 1 | 1];
        ++inv[n];
    }
}
void update(int L, int R, int v, int l, int r, int n)
{
    if (L <= l && r <= R)
    {
        add[n] = v, flip(n);
        sum[n] = v * (r - l + 1);
        return;
    }
    pushDown(n, r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) update(L, R, v, lson);
    if (m < R) update(L, R, v, rson);
    pushUp(n);
}
void inverse(int L, int R, int l, int r, int n)
{
    if (L <= l && r <= R)
    {
```

```
        ++inv[n];
        sum[n] = r - l + 1 - sum[n];
        return;
    }
    pushDown(n, r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) inverse(L, R, lson);
    if (m < R) inverse(L, R, rson);
    pushUp(n);
}
int query(int L, int R, int l, int r, int n)
{
    if (L <= l && r <= R)
        return sum[n];
    pushDown(n, r - l + 1);
    int m = (l + r) >> 1, ans = 0;
    if (L <= m) ans += query(L, R, lson);
    if (m < R) ans += query(L, R, rson);
    return ans;
}
int main()
{
    int t, cas = 0;
    scanf("%d", &t);
    while (t--)
    {
        int n = 0, cnt = 0, k, r, len, q, a, b;
        scanf("%d", &k);
        while (k--)
        {
            scanf("%d%s", &r, s), len = strlen(s);
            for (int i = 0; i < r; ++i)
                for (int j = 1; j <= len; ++j)
                    num[n + i * len + j] = s[j - 1] - '0';
            n += r * len;
        }
        build(1, n, 1);
        printf("Case %d:\n", ++cas);
        scanf("%d", &q);
        while (q--)
        {
            scanf("%s%d%d", s, &a, &b), ++a, ++b;
            if (s[0] == 'F')
                update(a, b, 1, 1, n, 1);
            else if (s[0] == 'E')
                update(a, b, 0, 1, n, 1);
            else if (s[0] == 'I')
                inverse(a, b, 1, n, 1);
            else printf("Q%d: %d\n", ++cnt, query(a, b, 1, n, 1));
        }
    }
    return 0;
}

/* sum without duplicate (offline solution) */
#include <cstdio>
#include <algorithm>
#define MX 111111
```

```
using namespace std;
typedef long long ll;
struct S {
    int l, r, i;
} p[MX];
int x[MX], a[MX], f[MX];
ll c[MX], ans[MX];
int cmp(S a, S b) {
    return a.r < b.r;
}
void U(int x, int v) {
    for (; x < MX; x += (x & -x))
        c[x] += v;
}
ll Q(int x) {
    ll s = 0;
    for (; x > 0; x -= (x & -x))
        s += c[x];
    return s;
}
int B(int v, int u) {
    int l = 0, r = u - 1, m;
    while (l <= r) {
        m = (l + r) >> 1;
        if (x[m] == v)
            return m;
        if (x[m] < v)
            l = m + 1;
        else r = m - 1;
    }
    return -1;
}
int main() {
    int t, n, q, cnt;
    scanf("%d", &t);
    while (t--) {
        memset(c, 0, sizeof(c));
        memset(f, 0, sizeof(f));
        scanf("%d", &n), cnt = n;
        for (int i = 1; i <= n; ++i)
            scanf("%d", &a[i]), x[i - 1] = a[i];
        sort(x, x + cnt);
        cnt = unique(x, x + cnt) - x;
        scanf("%d", &q);
        for (int i = 0; i < q; ++i)
            scanf("%d%d", &p[i].l, &p[i].r), p[i].i = i;
        sort(p, p + q, cmp);
        int k = 0, d;
        for (int i = 1; i <= n; ++i) {
            d = B(a[i], cnt);
            if (f[d]) U(f[d], -a[i]);
            U(i, a[i]);
            f[d] = i;
            for (; k < q; ++k) {
                if (p[k].r == i)
                    ans[p[k].i] = Q(p[k].r) - Q(p[k].l - 1);
                else break;
            }
```

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

/* update but only keep bigger number */
// offline, sort update intervals
/* same above but huge interval */
#include <cstdio>
#include <algorithm>
#define MX 111111
#define lson l,m,n<<1
#define rson m,r,n<<1|1
using namespace std;
typedef long long ll;
int v, N, L, R, x[MX], y[MX], h[MX], p[MX], s[MX << 2];
void B(int l = 1, int r = N, int n = 1) {
    if (l == r - 1) return;
    int m = (l + r) >> 1;
    B(lson), B(rson);
}
void U(int L, int R, int l = 1, int r = N, int n = 1) {
    if (p[l] == L && p[r] == R) {
        if (s[n] < h[v]) s[n] = h[v];
        return;
    }
    int m = (l + r) >> 1;
    if (R <= p[m]) U(L, R, lson);
    else if (L >= p[m]) U(L, R, rson);
    else U(L, p[m], lson), U(p[m], R, rson);
}
ll Q(int l = 1, int r = N, int n = 1, int t = 0) {
    if (s[n] < t) s[n] = t;
    if (l == r - 1) return (ll)(p[r] - p[l]) * s[n];
    int m = (l + r) >> 1;
    return Q(lson, s[n]) + Q(rson, s[n]);
}
int main() {
    int m;
    scanf("%d", &m);
    for (int i = 1; i <= m; ++i) {
        scanf("%d%d%d", &x[i], &y[i], &h[i]);
        p[++N] = x[i], p[++N] = y[i];
    }
    sort(p + 1, p + N + 1);
    N = unique(p + 1, p + N + 1) - p - 1;
    B();
    for (v = 1; v <= m; ++v)
        U(x[v], y[v]);
    printf("%lld\n", Q());
    return 0;
}

/* UVa 1400 */
#include <bits/stdc++.h>
#define MX 500050
```

```
#define lson l,m,n<<1
#define rson m+1,r,n<<1|1
using namespace std;
typedef long long ll;
typedef pair<int, int> seg;
ll sum[MX];
int pre[MX << 2], suf[MX << 2];
seg sub[MX << 2];
ll get(int l, int r) {
    return sum[r] - sum[l - 1];
}
ll get(seg s) {
    return get(s.first, s.second);
}
seg max(seg a, seg b) {
    if (get(a) != get(b)) return get(a) > get(b) ? a : b;
    return a < b ? a : b;
}
void pushUp(int l, int r, int n) {
    ll v1 = get(l, pre[n << 1]), v2 = get(l, pre[n << 1 | 1]);
    if (v1 == v2)
        pre[n] = min(pre[n << 1], pre[n << 1 | 1]);
    else pre[n] = v1 > v2 ? pre[n << 1] : pre[n << 1 | 1];
    v1 = get(suf[n << 1], r), v2 = get(suf[n << 1 | 1], r);
    if (v1 == v2)
        suf[n] = min(suf[n << 1], suf[n << 1 | 1]);
    else suf[n] = v1 > v2 ? suf[n << 1] : suf[n << 1 | 1];
    sub[n] = max(make_pair(suf[n << 1], pre[n << 1 | 1]),
                 max(sub[n << 1], sub[n << 1 | 1]));
}
void build(int l, int r, int n) {
    if (l == r) {
        pre[n] = suf[n] = l, sub[n] = make_pair(l, l);
        return;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    pushUp(l, r, n);
}
seg prefix(int L, int R, int l, int r, int n) {
    if (pre[n] <= R) return make_pair(l, pre[n]);
    int m = (l + r) >> 1;
    if (R <= m) return prefix(L, R, lson);
    seg ans = prefix(L, R, rson);
    ans.first = l;
    return max(ans, make_pair(l, pre[n << 1]));
}
seg suffix(int L, int R, int l, int r, int n) {
    if (suf[n] >= L) return make_pair(suf[n], r);
    int m = (l + r) >> 1;
    if (L > m) return suffix(L, R, rson);
    seg ans = suffix(L, R, lson);
    ans.second = r;
    return max(ans, make_pair(suf[n << 1 | 1], r));
}
seg query(int L, int R, int l, int r, int n) {
    if (L <= l && r <= R) return sub[n];
```

```
    int m = (l + r) >> 1;
    if (R <= m) return query(L, R, lson);
    if (L > m) return query(L, R, rson);
    return max(max(query(L, R, lson), query(L, R, rson)),
               make_pair(suffix(L, R, lson).first, prefix(L, R, rson).second));
}
int main() {
    int cas = 0, n, m, a, b;
    while (~scanf("%d%d", &n, &m)) {
        sum[0] = 0;
        for (int i = 0; i < n; ++i)
            scanf("%d", &a), sum[i + 1] = sum[i] + a;
        build(1, n, 1);
        printf("Case %d:\n", ++cas);
        while (m--) {
            scanf("%d%d", &a, &b);
            seg ans = query(a, b, 1, n, 1);
            printf("%d %d\n", ans.first, ans.second);
        }
    }
    return 0;
}
```

3. Interval Tree 2D

```
#include <cstdio>
#include <algorithm>
#define maxn 510
#define it tree[p1][p2]
using namespace std;
struct Seg_Tree2D
{
    int minv, maxv;
    friend Seg_Tree2D operator + (const Seg_Tree2D &a, const Seg_Tree2D &b)
    {
        Seg_Tree2D c;
        c.minv = min(a.minv, b.minv);
        c.maxv = max(a.maxv, b.maxv);
        return c;
    }
} tree[maxn << 2][maxn << 2];
int matrix[maxn][maxn], n, m;
void Build2(int p1, int p2, int l, int r, int a, int b)
{
    if (a == b)
    {
        if (l == r)
            it.minv = it.maxv = matrix[l][a];
        else
            it = tree[p1 << 1][p2] + tree[p1 << 1 | 1][p2];
    }
    else
    {
        int mid = (a + b) >> 1;
        Build2(p1, p2 << 1, l, r, a, mid), Build2(p1, p2 << 1 | 1, l, r, mid + 1, b);
        it = tree[p1][p2 << 1] + tree[p1][p2 << 1 | 1];
    }
}
void Build1(int p1, int l, int r)
```

```
{
    if (l == r)
    {
        Build2(p1, 1, l, r, 1, m);
        return ;
    }
    int mid = (l + r) >> 1;
    Build1(p1 << 1, l, mid), Build1(p1 << 1 | 1, mid + 1, r);
    Build2(p1, 1, l, r, 1, m);
}
void Modify2(int p1, int p2, int l, int r, int a, int b, int y, int v)
{
    if (a == b)
    {
        if (l == r)
            it.minv = it.maxv = v;
        else
            it = tree[p1 << 1][p2] + tree[p1 << 1 | 1][p2];
    }
    else
    {
        int mid = (a + b) >> 1;
        if (y <= mid)
            Modify2(p1, p2 << 1, l, r, a, mid, y, v);
        else
            Modify2(p1, p2 << 1 | 1, l, r, mid + 1, b, y, v);
        it = tree[p1][p2 << 1] + tree[p1][p2 << 1 | 1];
    }
}
void Modify1(int p1, int l, int r, int x, int y, int v)
{
    if (l == r)
    {
        Modify2(p1, 1, l, r, 1, m, y, v);
        return ;
    }
    int mid = (l + r) >> 1;
    if (x <= mid)
        Modify1(p1 << 1, l, mid, x, y, v);
    else
        Modify1(p1 << 1 | 1, mid + 1, r, x, y, v);
    Modify2(p1, 1, l, r, 1, m, y, v);
}
Seg_Tree2D Query2(int p1, int p2, int l, int r, int a, int b)
{
    if (l == a && r == b)
        return it;
    int mid = (l + r) >> 1;
    if (b <= mid)
        return Query2(p1, p2 << 1, l, mid, a, b);
    else if (mid < a)
        return Query2(p1, p2 << 1 | 1, mid + 1, r, a, b);
    return Query2(p1, p2 << 1, l, mid, a, mid) + Query2(p1, p2 << 1 | 1, mid + 1, r,
mid + 1, b);
}
Seg_Tree2D Query1(int p1, int l, int r, int ax, int ay, int bx, int by)
{
    if (l == ax && r == bx)
```

```
        return Query2(p1, 1, 1, m, ay, by);
    int mid = (l + r) >> 1;
    if (bx <= mid)
        return Query1(p1 << 1, l, mid, ax, ay, bx, by);
    else if (mid < ax)
        return Query1(p1 << 1 | 1, mid + 1, r, ax, ay, bx, by);
    return Query1(p1 << 1, l, mid, ax, ay, mid, by) + Query1(p1 << 1 | 1, mid + 1, r,
mid + 1, ay, bx, by);
}
void read()
{
    scanf("%d %d", &n, &m);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            scanf("%d", &matrix[i][j]);
    Build1(1, 1, n);
}
void Query()
{
    char task[10];
    int q;
    scanf("%d", &q);
    Seg_Tree2D ans;
    for (int i = 1, a, b, c, d; i <= q; i++)
    {
        scanf("%s", task);
        if (task[0] == 'c')
        {
            scanf("%d %d %d", &a, &b, &c);
            Modify1(1, 1, n, a, b, c);
        }
        else
        {
            scanf("%d %d %d %d", &a, &b, &c, &d);
            ans = Query1(1, 1, n, a, b, c, d);
            printf("%d %d\n", ans.maxv, ans.minv);
        }
    }
}
int main()
{
    read();
    Query();
    return 0;
}
```

4. KMP

```
int m, n, p[10010];
char a[1000100], b[10010];
void init() {
    p[0] = -1;
    int i = 0, j = -1;
    while (i < n)
        if (j == -1 || b[i] == b[j])
            ++i, ++j, p[i] = j;
        else j = p[j];
}
int kmp() {
    int i = 0, j = 0, ans = 0;
```

```
    while (i < m) {
        if (j == -1 || b[j] == a[i])
            ++i, ++j;
        else j = p[j];
        if (j == n)
            ++ans;
    }
    return ans;
}
```

5. Binary Search

```
int bs(int k) {
    int l = 0, r = n - 1, p;
    while (l <= r) {
        p = (l + r) >> 1;
        if (a[p] == k)
            return p;
        if (a[p] < k)
            l = p + 1;
        else
            r = p - 1;
    }
    return -1;
}
```

```
/// algo
bool binary_search( ForwardIt first, ForwardIt last, const T &value,
                    Compare comp );
#include <iostream>
#include <algorithm>
#include <vector>
int main() {
    std::vector<int> haystack {1, 3, 4, 5, 9};
    std::vector<int> needles {1, 2, 3};

    for (auto needle : needles) {
        std::cout << "Searching for " << needle << '\n';
        if (std::binary_search(haystack.begin(), haystack.end(), needle)) {
            std::cout << "Found " << needle << '\n';
        } else {
            std::cout << "no dice!\n";
        }
    }
}
```

```
/// std
#include <cstdlib>
#include <iostream>
int compare(const void *ap, const void *bp) {
    const int *a = (int *) ap;
    const int *b = (int *) bp;
    if (*a < *b)
        return -1;
    else if (*a > *b)
        return 1;
    else
        return 0;
}
```

```
int main(int argc, char **argv) {
    const int ARR_SIZE = 8;
    int arr[ARR_SIZE] = { 1, 2, 3, 4, 5, 6, 7, 8 };

    int key1 = 4;
    int *p1 = (int *) std::bsearch(&key1, arr, ARR_SIZE, sizeof(arr[0]), compare);
    if (p1)
        std::cout << "value " << key1 << " found at position " << (p1 - arr) << '\n';
    else
        std::cout << "value " << key1 << " not found\n";

    int key2 = 9;
    int *p2 = (int *) std::bsearch(&key2, arr, ARR_SIZE, sizeof(arr[0]), compare);
    if (p2)
        std::cout << "value " << key2 << " found at position " << (p2 - arr) << '\n';
    else
        std::cout << "value " << key2 << " not found\n";
}
```

6. Union Find

```
int p[MX], q[MX], n;

void init() {
    for (int i = 0; i < MX; ++i)
        p[i] = i, q[i] = 1;
}

int F(int x) {
    return x == p[x] ? x : (p[x] = F(p[x]));
}

bool equal(int x, int y) { // whether at same group
    return F(x) == F(y);
}

void uni(int x, int y) {
    x = F(x), y = F(y);
    if (x == y) return;
    --n;
    p[x] = y;
    q[y] += q[x];
    q[x] = 0;
}

int cardinality(int x) {
    return n[F(x)];
}

bool single(int x) {
    return n[F(x)] == 1;
}
```

7. RMQ

RMQ	初始化	查询	空间
st算法	O(nlogn)	O(1)	O(nlogn)
树状数组	O(nlogn)	O(logn)	O(n)

```
// st
int p[MX], d[MX][20];
void init(int n) {
    for (int i = 0; i < n; ++i)
        d[i][0] = p[i];
    for (int j = 1; (1 << j) <= n; ++j)
        for (int i = 0; i + (1 << j) - 1 < n; ++i)
            d[i][j] = max(d[i][j - 1], d[i + (1 << (j - 1))][j - 1]);
}
int rmq(int x, int y) {
    if (x > y)
        return 0;
    int k = 0;
    while ((1 << (k + 1)) <= y - x + 1)
        ++k;
    return max(d[x][k], d[y - (1 << k) + 1][k]);
}

// bit
int p[MX], d[MX];
void init(int n) {
    for (int i = 1; i < n; ++i) {
        d[i] = p[i];
        for (int j = 1; j < (i & -i); j <<= 1)
            d[i] = max(d[i], d[i - j]);
    }
}
int rmq(int x, int y) {
    if (x > y) return 0;
    int ans = p[y];
    while (1) {
        ans = max(ans, p[y]);
        if (x == y) break;
        for (y -= 1; y - x >= (y & -y); y -= (y & -y))
            ans = max(ans, d[y]);
    }
    return ans;
}
```

8. Manacher

```
char s[MX];
int l[MX];
void palindrome(char cs[], int len[],
                int n) { //len[i] means the max palindrome length centered i/2
    for (int i = 0; i < n * 2; ++i) {
        len[i] = 0;
    }
    for (int i = 0, j = 0, k; i < n * 2; i += k, j = max(j - k, 0)) {
        while (i - j >= 0 && i + j + 1 < n * 2
                && cs[(i - j) / 2] == cs[(i + j + 1) / 2])
            j++;
        len[i] = j;
```

```
        for (k = 1; i - k >= 0 && j - k >= 0 && len[i - k] != j - k; k++) {
            len[i + k] = min(len[i - k], j - k);
        }
    }
}
int main() {
    while (~scanf("%s", s)) {
        int ans = 0, sl = strlen(s);
        palindrome(s, l, sl);
        for (int i = 0; i < sl * 2; ++i)
            ans = max(ans, l[i]);
        printf("%d\n", ans);
    }
    return 0;
}
```

9. Hash

```
const ull B = 100000007ULL; /// 哈希基数，1e8 + 7
const int mx_s_num = 105; /// 字符串个数

char s[mx_s_num][mx]; /// 注意，一定要用gets(s[i] + 1)，从下标1开始读
ull ha[mx_s_num][mx], bp[mx] = {1ULL}; /// ha[i]从1开始，一直到ha[i][n]
int len[mx_s_num]; /// len[i] = strlen(s[i] + 1); 一定要是s[i] + 1，否则n会是0

void init_hash(int s_num) { /// 请在main()中完成len的求取。
    int i, j;
    For(i, s_num) Forr(j, 1, len[i] + 1) ha[i][j] = ha[i][j - 1] * B + s[i][j];
    int n = Max(len, s_num); /// 调用#define的Max()
    Forr(i, 1, n + 1) bp[i] = bp[i - 1] * B;
}

ull get_hash(char *s) { /// 直接返回整个字符串的hash
    ull ha = 0ULL;
    for (int i = 0; s[i]; ++i) ha = ha * B + s[i];
    return ha;
}

ull get_hash(int *a, int n) { /// 返回整个int数组的hash值
    int i;
    ull ha = 0ULL;
    For(i, n) ha = ha * B + (ull)a[i];
    return ha;
}

/// 注意pos一定不能是0!!!!
inline ull get_hash(ull *Ha, int pos,
                    int l) { /// 返回Ha[pos...pos+l-1]的值，pos与l必须是正数
    return Ha[pos + l - 1] - Ha[pos - 1] * bp[l];
}

inline ull merge_hash(ull ha1, ull ha2,
                      int len2) { /// 返回s1+s2拼接后的hash值
    return ha1 * bp[len2] + ha2;
}

bool contain(int ida, int
```

```
            idb) { /// b是否为a的子串 ，ida和idb为字符串下标，若只有两个字符串，使用时传入参数(0, 1)、(1, 0)就行
    if (len[ida] < len[idb]) return false;
    ull hab = ha[idb][len[idb]];
    for (int i = 1; i + len[idb] <= len[ida]; ++i)
        if (get_hash(ha[ida], i, len[idb]) == hab) return true;
    return false;
}

int overlap(int ida, int
           idb) { /// 求a后缀与b前缀的最长公共子串，ida和idb为字符串下标，若只有两个字符串，使用时传入参数(0, 1)、(1, 0)就行
    int ans = 0, i;
    Forr(i, 1, min(len[ida], len[idb]) + 1)
    if (get_hash(ha[ida], len[ida] - i + 1, i) == get_hash(ha[idb], 1, i)) ans = i;
    // 可在if中加上 && strncmp(s[ida] + len[ida] - i + 1, s[idb] + 1, i) == 0(不过这就失去意义了，还不如双hash)
    return ans;
}
```

10. LIS

```
#include <bits/stdc++.h>
using namespace std;
int lis[MAXN];
int lis() {
    int n, i, j, x, len = 0;
    for (i = 1; i <= n; ++i) {
        scanf("%d", &x); // for existing array, use x=num[i]
        j = lower_bound(lis + 1, lis + len + 1, x) - lis;
        // LDS : j = lower_bound(lds + 1, lds + len + 1, x, greater<int>()) - lds;
        lis[j] = x;
        len = max(len, j);
    }
    return len;
}
//LIS[i] = max{1 , LIS[k] + 1}  (∀k < i, arr[i] > arr[k])
/*
Longest Not-decrease Sequence:
bool cmp(int a, int b)
{
    return a <= b;
}
j = lower_bound(lis + 1, lis + len + 1, x, cmp) - lis;
*/
```

11. LCS

```
#include <bits/stdc++.h>
using namespace std;
const int MAXSTRLEN = 1000;

char a[MAXSTRLEN], b[MAXSTRLEN];
int dp[MAXSTRLEN][MAXSTRLEN], path[MAXSTRLEN][MAXSTRLEN];

int Lcs(char x[], char y[]) {
    int i, j, len1 = strlen(x + 1), len2 = strlen(y + 1);
    memset(dp, 0, sizeof(dp));
    for (i = 1; i <= len1; ++i)
```

```
        for (j = 1; j <= len2; ++j) {
            if (x[i] == y[j])
                dp[i][j] = dp[i - 1][j - 1] + 1, path[i][j] = 1;
            else if (dp[i - 1][j] >= dp[i][j - 1])
                dp[i][j] = dp[i - 1][j], path[i][j] = 2;
            else
                dp[i][j] = dp[i][j - 1], path[i][j] = 3;
        }
    return dp[len1][len2];
}

void PrintLcs(int i, int j) {
    if (i == 0 || j == 0) return;
    if (path[i][j] == 1) {
        PrintLcs(i - 1, j - 1);
        putchar(a[i]);
    } else if (path[i][j] == 2) PrintLcs(i - 1, j);
    else PrintLcs(i, j - 1);
}

int main() {
    while (gets(a + 1)) {
        gets(b + 1);
        printf("%d\n", Lcs(a, b));
        PrintLcs(strlen(a + 1), strlen(b + 1));
        putchar(10);
    }
    return 0;
}
```

12. Date

```
//日期函数
int days[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
struct date {
    int year, month, day;
};

//判闰年
inline int leap(int year) {
    return (year % 4 == 0 && year % 100 != 0) || year % 400 == 0;
}

//判合法性
inline int legal(date a) {
    if (a.month < 0 || a.month > 12) return 0;
    if (a.month == 2)
        return a.day > 0 && a.day <= 28 + leap(a.year);
    return a.day > 0 && a.day <= days[a.month - 1];
}

//比较日期大小
inline int datecmp(date a, date b) {
    if (a.year != b.year)
        return a.year - b.year;
    if (a.month != b.month)
        return a.month - b.month;
    return a.day - b.day;
}
```

```
//返回指定日期是星期几
int weekday(date a) {
    int tm = a.month >= 3 ? (a.month - 2) : (a.month + 10);
    int ty = a.month >= 3 ? a.year : (a.year - 1);
    return (ty + ty / 4 - ty / 100 + ty / 400 + (int)(2.6 * tm - 0.2) + a.day) % 7;
}

//日期转天数偏移
int date2int(date a) {
    int ret = a.year * 365 + (a.year - 1) / 4 - (a.year - 1) / 100 +
              (a.year - 1) / 400, i; days[1] += leap(a.year);
    for (i = 0; i < a.month - 1; ret += days[i++]);
    days[1] = 28;
    return ret + a.day;
}

//天数偏移转日期
date int2date(int a) {
    date ret;
    ret.year = a / 146097 * 400;
    for (a %= 146097; a >= 365 + leap(ret.year);
            a -= 365 + leap(ret.year), ret.year++);
    days[1] += leap(ret.year);
    for (ret.month = 1; a >= days[ret.month - 1];
            a -= days[ret.month - 1], ret.month++);
    days[1] = 28; ret.day = a + 1; return ret;
}
```