Acm Code Library

By

io07

r_clover

2015.9.10

目录

头文件

```
#include <cstdio>

#include <cstring>

#include <iostream>

#include <algorithm>

#include <cmath>

#include <vector>

#include <queue>

#include <stack>

#include <set>

#include <map>

//#include <tr1/unordered_set>

//#include <tr1/unordered_map>

#include <bitset>

//#pragma comment(linker, "/STACK:1024000000,1024000000")

#define lson l, m, rt<<1

#define rson m+1, r, rt<<1|1

#define inf 1e9

#define debug(a) cout << #a" = " << (a) << endl;

#define debugarry(a, n) for (int i = 0; i < (n); i++) { cout << #a"[" << i << "] = " << (a)[i]
<< endl; }
#define clr(x, y) memset(x, y, sizeof x)
#define LL long long
#define uLL unsigned LL
using namespace std;
```

输入输出挂

```
template<class T>
inline bool read(T &n)
{
    T x = 0, tmp = 1;
    char c = getchar();
    while ((c < '0' || c > '9') && c != '-' && c != EOF) c = getchar();
    if (c == EOF) return false;
    if (c == '-') c = getchar(), tmp = -1;
    while (c >= '0' && c <= '9') x *= 10, x += (c - '0'), c = getchar();
    n = x*tmp;
    return true;
}
template <class T>
inline void write(T n)
```

```
{
        if (n < 0)
        {
                putchar('-');
                n = -n;
        }
        int len = 0, data[20];
        while (n)
        {
                data[len++] = n % 10;
                n /= 10;
        }
        if (!len) data[len++] = 0;
        while (len--) putchar(data[len] + 48);
}
```

bitset

```
#include <bitset>//正常情况 加速32倍 可能还要加速更多
bitset<n>p;
```

b.any() b 中是否存在置为 1 的二进制位？

b.none() b 中不存在置为 1 的二进制位吗？

b.count() b 中置为 1 的二进制位的个数

b.size() b 中二进制位的个数

b[pos] 访问 b 中在 pos 处的二进制位

b.test(pos) b 中在 pos 处的二进制位是否为 1？

b.set() 把 b 中所有二进制位都置为 1

b.set(pos) 把 b 中在 pos 处的二进制位置为 1

b.reset() 把 b 中所有二进制位都置为 0

b.reset(pos) 把 b 中在 pos 处的二进制位置为 0

b.flip() 把 b 中所有二进制位逐位取反

b.flip(pos) 把 b 中在 pos 处的二进制位取反

b.to_ulong() 用 b 中同样的二进制位返回一个 unsigned long 值

os << b 把 b 中的位集输出到 os 流

以及所有位操作。

```
＃include <bitset>

using std::bitset;

bitset<32> bitvec; //32 位，全为 0。
```

bitset<n> b; b 有 n 位，每位都为 0

bitset<n> b(u); b 是 unsigned long 型 u 的一个副本

bitset<n> b(s); b 是 string 对象 s 中含有的位串的副本

bitset<n> b(s, pos, n); b 是 s 中从位置 pos 开始的 n 个位的副本

hdu 5413 n 2e4

普通 n^2 111.363s

bitset 2.385s

离散化

```
int sub[maxn], len, n, A[maxn];

int main()

{

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

    {

        len = 0;

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

        {

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

            sub[len++] = A[i];

        }

        sort(sub, sub + len);

        len = uniqe(sub, sub + len) - sub;

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

            A[i] = lower_bound(sub, sub + len, A[i]) - sub + 1;

    }

    return 0;

}
```

多重背包

```
int dp[maxn];

int w[maxn], v[maxn], c[maxn];

int n, C;



void f()

{

    clr(dp, 0);

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

    {

        int cnt = c[i];

        int k = 1;

        while (cnt >= k)

        {

            for (int j = C; j >= w[i] * k; j--)

            {

                dp[j] = max(dp[j], dp[j - w[i] * k] + v[i] * k);
            }
            cnt -= k;
            k <<= 1;
        }
        if (cnt)
        {
            k = cnt;
            for (int j = C; j >= w[i] * k; j--)
            {
                dp[j] = max(dp[j], dp[j - w[i] * k] + v[i] * k);
            }
        }
    }
}
```

数位 DP

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

```
#include<iostream>
using namespace std;
typedef long long LL;
const int MAX_N = 1010;
const int MAX_K = 10;

int num[9];
int dp[9][2];

/*len：表示长度
s：当前的状态（对于此题就是上一位是否是6）
fp：表示之前的状态是否充满的（如果说充满的此处的放置就是有限制的 否则就可以随便放）
*/
int dfs(int len, int s, bool fp)
{
    if (len == 0) return 1;
    if (!fp&&dp[len][s] != -1) return dp[len][s];
    int res = 0;
    int fmax = fp ? num[len] : 9;          //根据充满状态选择限制条件
    for (int i = 0; i <= fmax; i++)
    {
        if (i == 4 || s&&i == 2) continue; //跳过62和4的情况
        res += dfs(len - 1, i == 6, fp&&i == fmax);
    }
    return fp ? res : dp[len][s] = res;
}
int solve(int n)
{
    int len = 0;
    while (n != 0)
    {
        num[++len] = n % 10;
        n /= 10;
    }
    return dfs(len, 0, 1);
}
int main()
{
    int n, m;
    memset(dp, -1, sizeof(dp));
    while (scanf("%d%d", &n, &m), n | m){
        printf("%d\n", solve(m) - solve(n - 1));
    }
    return 0;
```

```
}
```

快速幂运算

```
struct mat
{
    int m[100][100];
};
int n;
mat mul(mat a, mat b)
{
    mat c;
    clr(c.m, 0);
    for (int k = 1; k <= n; k++)
    for (int i = 1; i <= n; i++) if (a.m[i][k])
    for (int j = 1; j <= n; j++)
        c.m[i][j] += a.m[i][k] * b.m[k][j];
}

mat add(mat a, mat b)
```

```
{
    mat c;
    for (int i = 1; i <= n; i++)
    for (int j = 1; j <= n; j++)
        c.m[i][j] = a.m[i][j] + b.m[i][j];
}

mat re;

mat sum_pow_mat(mat a, int n)
{
    if (n % 2)
    {
        if (n == 1) return re = a;
        mat p = sum_pow_mat(a, n - 1);
        re = mul(re, a);
        return add(a, mul(a, p));
    }
    else
    {
        mat p = sum_pow_mat(a, n / 2);
        mat ans = add(p, mul(p, re));
        re = mul(re, re);

        return ans;

    }

}
```

KMP

```
vector<int>ans;

void getFail(char *P, int *f)

{

    int m = strlen(P);

    f[0] = 0;

    f[1] = 0;

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

    {

        int j = f[i];
        while (j&&P[i] != P[j]) j = f[j];

        f[i + 1] = P[i] == P[j] ? j + 1 : 0 ;

    }

}



void find(char *T, char *P, int *f)

{

    int n = strlen(T), m = strlen(P);

    getFail(P, f);

    int j = 0;

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

    {

        while (j&&P[j] != T[i]) j = f[j];

        if (P[j] == T[i]) j++;

        if (j == m) ans.push_back(i - m + 1);

    }

}
```

后缀数组

```
#include<string.h>
```

```
const int maxn = 1e5;

#define FOR(i,a,b) for(i=a; (a<b)?(i<=b):(i>=b) ; (a<b)?(i++):(i--) )

struct suffix_array
{
    char s[maxn];
    int sa[maxn], t[maxn], t2[maxn], c[maxn];
    int m, n;
    ///构造sa数组
    void build_sa()
    {
        int i, *x = t, *y = t2;
        FOR(i, 0, m - 1) c[i] = 0;
        FOR(i, 0, n - 1) c[x[i] = s[i]]++;
        FOR(i, 1, m - 1) c[i] += c[i - 1];
        FOR(i, n - 1, 0) sa[--c[x[i]]] = i;
        for (int k = 1; k <= n; k <<= 1)
        {
            int p = 0;
            FOR(i, n - k, n - 1) y[p++] = i;
            FOR(i, 0, n - 1) if (sa[i] >= k) y[p++] = sa[i] - k;
            FOR(i, 0, m - 1) c[i] = 0;
            FOR(i, 0, n - 1) c[x[y[i]]]++;
            FOR(i, 0, m - 1) c[i] += c[i - 1];
            FOR(i, n - 1, 0) sa[--c[x[y[i]]]] = y[i];
            swap(x, y);
            p = 1;
            x[sa[0]] = 0;
            FOR(i, 1, n - 1)
                x[sa[i]] = y[sa[i - 1]] == y[sa[i]] &&
                y[sa[i - 1] + k] == y[sa[i] + k] ? p - 1 : p++;
            if (p >= n) break;
            m = p;
        }
    }
    ///匹配模式串
    int len;
    int cmp(char *pa, int p)
    {
        return strncmp(pa, s + sa[p], len);
    }
    int find_first(char *P)
```

```
{
    len = strlen(P);
    if (cmp(P, 0)<0) return -1;
    if (cmp(P, n - 1)>0) return -1;
    int L = 0, R = n - 1, ans = n;
    while (R >= L)
    {
        int M = L + (R - L) / 2;
        int res = cmp(P, M);
        if (res <= 0)
        {
            R = M - 1;
            if (res == 0) ans = min(ans, M);
        }
        else L = M + 1;
    }
    if (ans == n) return -1;
    else return ans;
}
int find_last(char *P)
{
    len = strlen(P);
```

```
        if (cmp(P, 0)<0) return -1;
        if (cmp(P, n - 1)>0) return -1;
        int L = 0, R = n - 1, ans = -1;
        while (R >= L)
        {
            int M = L + (R - L) / 2;
            int res = cmp(P, M);
            if (res >= 0)
            {
                L = M + 1;
                if (res == 0) ans = max(ans, M);
            }
        }
        return ans;
}
///构造rank，height数组
int rank[maxn], height[maxn];
void getHeight()
{
        int i, j, k = 0;
        for (int i = 0; i<n; i++) rank[sa[i]] = i;
        for (int i = 0; i<n; i++)
```

```
    {
        if (k) k--;
        int j = sa[rank[i] - 1];
        while (s[i + k] == s[j + k]) k++;
        height[rank[i]] = k;
    }
}
int d[maxn][30], flog[maxn];
void RMQ_init()
{
    for (int i = 0; i<n; i++) d[i][0] = height[i];
    flog[0] = -1;
    for (int i = 1; i<n; i++) flog[i] = flog[i >> 1] + 1;
    for (int j = 1; (1 << j) <= n; j++)
    for (int i = 0; i + (1 << j) <= n; i++)
        d[i][j] = min(d[i][j - 1], d[i + (1 << (j - 1))][j - 1]);
}
int RMQ(int L, int R)
{
    int k = flog[R - L + 1];
    return min(d[L][k], d[R - (1 << k) + 1][k]);
}
```

```
    int query(int j, int k)

    {

        if (j == k) return n - k;

        if (rank[j]>rank[k]) swap(j, k);

        return RMQ(rank[j] + 1, rank[k]);

    }

    void init(char *ss, int mm = 200)

    {

        strcpy(s, ss);

        n = strlen(s);

        m = mm;

    }

    int all()

    {

        build_sa();

        getHeight();

        RMQ_init();

    }

}sp;
```

字符串 Hash

```
/*题意：

给出两个串A，B，让你找出B串在A串中匹配的第一个位置，匹配要求可以有最多两个位置不一样。
解法：

首先将A，B串的hash值求出来，然后就可以O(1)来求出每个子串的hash值了，判位置最裸的方法就是枚举每一个位置然后O(n)的判是否匹配，复杂度是O(n^2)，然后我们可以通过二分来加速判匹配的过程将其将成O(logn)的复杂度，具体操作如下：

二分一个求一个最长的匹配长度，然后跳过一个不匹配的位置然后再用一次二分求一个最长匹配长度，如此循环两次，如果求出来的最长匹配的长度+不匹配位置数（0 or 1 or 2）等于B串长的话就表明找到了一个符合条件的位置。*/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <cmath>
#include <cstring>
#include <vector>
```

```
#include <queue>
#include <stack>
#include <set>
#include <vector>

//#pragma comment(linker, "/STACK:1024000000,1024000000")
#define lson l, m, rt<<1
#define rson m+1, r, rt<<1|1
#define inf 1e9
#define debug(a) cout << #a" = " << (a) << endl;
#define debugarry(a, n) for (int i = 0; i < (n); i++) { cout << #a"[" << i << "] = " << (a)[i] << endl; }
#define clr(x, y) memset(x, y, sizeof x)
using namespace std;

typedef unsigned long long uLL;

const int maxn = 100005;
const uLL magic = 7;

uLL base[maxn];
uLL hash_a[maxn], hash_b[maxn];
```

```
char s1[maxn], s2[maxn];

void init_hash(int len, char cc[], uLL ha[])
{
    ha[0] = 0;
    for (int i = 1; i <= len; i++)
        ha[i] = ha[i - 1] * magic + cc[i - 1];
    base[0] = 1;
    for (int i = 1; i <= len; i++)
        base[i] = base[i - 1] * magic;
}

uLL sub_hash(uLL ha[], int L, int R)
{
    return ha[R] - ha[L] * base[R - L];
}

bool cmp(int La, int Ra, int Lb, int Rb)
{
    uLL ua = sub_hash(hash_a, La, Ra);
    uLL ub = sub_hash(hash_b, Lb, Rb);
    if (ua == ub) return true;
    else return false;
}

int get_max(int La, int Lb, int len)
{
    int st = -1, ed = len + 1;
    while (ed - st > 1)
    {
        int m = (st + ed) >> 1;
        if (cmp(La, La + m, Lb, Lb + m) == true) st = m;
        else ed = m;
    }
    return st;
}

int main()
{
    //freopen("input.txt", "r", stdin);

    int T;
    scanf("%d", &T);
```

```
for (int kk = 1; kk <= T; kk++)
{
    scanf("%s%s", s1, s2);
    int len_a = strlen(s1);
    int len_b = strlen(s2);
    init_hash(len_a, s1, hash_a);
    init_hash(len_b, s2, hash_b);
    //debugarry(hash_a, len_a+1); debugarry(hash_b, len_b+1);
    int ans = -1;
    for (int i = 0; i + len_b - 1 < len_a; i++)
    {
        int ptr_a = i, ptr_b = 0;
        int sum = 0, cnt = 0;
        while (cnt <= 2)
        {
            int maxlen = get_max(ptr_a, ptr_b, len_b - ptr_b);
            sum += maxlen;
            if (sum + cnt == len_b)
            {
                ans = i;
                break;
            }
```

```
                    else
                    {
                        cnt++;
                        ptr_a += maxlen + 1;
                        ptr_b += maxlen + 1;
                    }
                }
                //debug(sum);
                if (ans != -1) break;
            }
            printf("Case #%d: %d\n", kk, ans);
        }
        return 0;
}
```

Hash another :

```
#define ULL long long
cons ULL x = 233;

ULL pow_x[maxn];
```

```
void Hash_init(char *T, ULL *H)

{

        int len = strlen(T);

        pow_x[0] = 1ull;

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

                pow_x[i] = pow_x[i - 1] * x;

        H[len] = 0;

        for (int i = len - 1; i >= 0; i++)

                H[i] = H[i + 1] * x + (ULL)T[i];

}



void Hash(char *H, int i, int len)

{

        return H[i] - H[i + len] * pow_x[len];

}
```

AC 自动机 和 Trie 树

```
#define cls(p) clr(p,0)

const int maxn = 1e5;
const int maxsize = 30;

struct Trie
{
    int ch[maxn][maxnsize];
    int val[maxn];
    vector<int>vv[maxn];
    int sz;
    init()
    {
        sz = 1;
        cls(ch[0]);
        vv[0].clear();
    }
    int idx(int c)
    {
        return c - 'a';
    }
    void insert(char *s, int v)
    {
        int u = 0, n = strlen(s);
        for (int i = 0; i<n; i++)
        {
                int c = idx(s[i]);
                if (!ch[u][c])
                {
                        cls(ch[sz]);
                        val[sz] = 0;
                        vv[sz].clear();
                        ch[u][c] = sz++;
                }
                u = ch[u][c];
        }
        val[u] = v;
        vv.push_back(v);
}
///AC
vector<pair<int, int> >ans;
int last[maxn];
void print(int i, int j)
{
        if (j)
        {
                ans.push_back(make_pair(i, j));
        printf(i, last[j]);
    }
}
void find(char *T)
{
    ans.clear();
    int n = strlen(T);
    int j = 0;
    for (int i = 0; i<n; i++)
    {
        int c = u = idx(T[i]);
        while (j&&!ch[j][c]) j = f[j];
        j = ch[j][c];
        if (val[j]) print(i, j);
        else if (last[j]) print(i, last[j]);
    }
}
int getFail()
{
    queue<int>q;
    for (int c = 0; c<maxsize; c++)
    {
```

```
    int u = ch[0][c];

    if (u)

    {

        f[u] = 0;

        q.push(u);

        last[u] = 0;

    }

}

while (!q.empty())

{

    int r = q.front(); q.pop();

    for (int c = 0; c<maxszie; c++)

    {

        int u = ch[r][c];

        if (!u) continue;

        q.push(u);

        int v = f[r];

        while (v&&!ch[v][c]) v = f[v];

        f[u] = ch[v][c];

        last[u] = val[f[u]] ? f[u] : last[f[u]];

    }

}
```

```
        }
}
```

ST 表

```
const int maxn = 100000;
//maxn 即数组大小

int flog[(maxn << 1) + 10];
int A[maxn];
int dmax[maxn][30];
int n;

void RMQ_init(int *A) //RMQ 初始化
{
        for (int i = 0; i<n; i++) dmax[i][0] = A[i];
        for (int j = 1; (1 << j) <= n; j++)
        for (int i = 0; i + (1 << j) - 1< n; i++)
                dmax[i][j] = max(dmax[i][j - 1], dmax[i + (1 << (j - 1))][j - 1]);
        flog[0] = -1;
        for (int i = 1; i < 2 * maxn; i++) flog[i] = flog[i >> 1] + 1;
}
```

```
int RMQ(int L, int R) //RMQ 查询
{
    int k = flog[R - L + 1];
    return max(dmax[L][k], dmax[R - (1 << k) + 1][k]);
}

int dsum[maxn][30];

void st_sum_init(int *A)
{
    for (int i = 0; i<n; i++) dmax[i][0] = A[i];
    for (int j = 1; (1 << j) <= n; j++)
    for (int i = 0; i + (1 << j) - 1< n; i++)
        dmax[i][j] = dmax[i][j - 1] + dmax[i + (1 << (j - 1))][j - 1];
    flog[0] = -1;
    for (int i = 1; i < 2 * maxn; i++) flog[i] = flog[i >> 1] + 1;
}

int st_sum(int L, int R)
{
    if (L>R) return 0;
    int k = flog[R - L + 1];

    return dmax[L][k] + st_sum(L + (1 << k), R);

}
```

静态主席树

```
#include <iostream>

#include <cstdio>

#include <cstdlib>

#include <algorithm>

#include <cmath>

#include <cstring>

#include <vector>

#include <queue>

#include <stack>

#include <set>

#include <vector>

#include <deque>

#include <set>


//#pragma comment(linker, "/STACK:1024000000,1024000000")
```

```
#define lson l, (l+r>>1), ls[rt]
#define rson (l+r>>1)+1, r, rs[rt]
#define inf 1e9
#define debug(a) cout << #a" = " << (a) << endl;
#define debugarry(a, n) for (int i = 0; i < (n); i++) { cout << #a"[" << i << "] = " << (a)[i] << endl; }
#define clr(x, y) memset(x, y, sizeof x)
#define LL long long
using namespace std;

const int maxn = 1e5 + 20, maxs = maxn * 20;

#define head(p) ( p >= 0 ? h[p] : 0 )

struct __sad
{
    int ls[maxs], rs[maxs];
    int sum[maxs];
    int h[maxn];
    int si, len;
    void pushup(int rt)
    {
```

```
    sum[rt] = sum[ls[rt]] + sum[rs[rt]];

}

void build(int p, int add, int l, int r, int &rt, int rt2)

{

    if (!rt)

    {

        if (p<l || p>r)

        {

            rt = rt2;

            return;

        }

        sum[si] = ls[si] = rs[si] = 0;

        rt = si++;

    }

    if (l == r)

    {

        sum[rt] = sum[rt2] + add;

        return;

    }

    build(p, add, lson, ls[rt2]);

    build(p, add, rson, rs[rt2]);

    pushup(rt);
```

```
}

void init(int *A, int n)

{

        clr(h, 0);

        si = 1;

        ls[0] = rs[0] = sum[0] = 0;

        len = n;

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

        {

                build(A[i], 1, 0, n, head(i - 1));

        }

}

int query(int k, int l, int r, int rt, int rt2)

{

        if (l == r) return l;

        int nk = sum[ls[rt]] - sum[ls[rt2]];

        if (nk >= k) return query(k, lson, ls[rt2]);

        return query(k - nk, rson, rs[rt2]);

}

int query(int L, int R, int k)

{

        return query(k, 0, len, head(R), head(L - 1));
        }

};
```

动态主席树

用于解决动态修改某一个数，动态查询区间第k大

空间复杂度为 nlgnlgn

时间复杂度为 nlgnlgn

zoj 2112 Dynamic Rankings

裸动态主席树（其实这道题用整体二分更好）

N 50000 M 10000

输入格式

```
N M

a1 ... an

(M){

        Q l r k

                C pos t
}

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <cmath>
#include <cstring>
#include <vector>
#include <queue>
#include <stack>
#include <set>
#include <vector>
#include <map>
#include <tr1/unordered_set>
#include <tr1/unordered_map>

//#pragma comment(linker, "/STACK:1024000000,1024000000")
#define inf 1e9
#define debug(a) cout << #a" = " << (a) << endl;
#define debugarry(a, n) for (int i = 0; i < (n); i++) { cout << #a"[" << i << "] = " << (a)[i]
<< endl; }
```

```
#define clr(x, y) memset(x, y, sizeof x)
#define LL long long
using namespace std;

const int maxn = 100010; /// N×2
const int M = 8000030;    ///   ( M+N)*800
int n, q, m, tot;
int a[maxn], t[maxn];
int T[maxn], lson[M], rson[M], c[M];
int S[maxn];

struct Query
{
    int kind;
    int l, r, k;
} query[100010];

void Init_hash(int k)
{
    sort(t, t + k);
    m = unique(t, t + k) - t;
}
```

```
int get_hash(int x)
{
    return lower_bound(t, t + m, x) - t;
}
int build(int l, int r)
{
    int root = tot++;
    c[root] = 0;
    if (l != r)
    {
        int mid = (l + r) >> 1;
        lson[root] = build(l, mid);
        rson[root] = build(mid + 1, r);
    }
    return root;
}
int Insert(int root, int pos, int val)
{
    int newroot = tot++, tmp = newroot;
    int l = 0, r = m - 1;
    c[newroot] = c[root] + val;
    while (l < r)
```

```
        {
            int mid = (l + r) >> 1;
            if (pos <= mid)
            {
                lson[newroot] = tot++; rson[newroot] = rson[root];
                newroot = lson[newroot]; root = lson[root];
                r = mid;
            }
            else
            {
                rson[newroot] = tot++; lson[newroot] = lson[root];
                newroot = rson[newroot]; root = rson[root];
                l = mid + 1;
            }
            c[newroot] = c[root] + val;
        }
        return tmp;
    }
    int lowbit(int x)
    {
        return x & (-x);
    }
```

```
int use[maxn];

void add(int x, int pos, int val)
{
    while (x <= n)
    {
        S[x] = Insert(S[x], pos, val);
        x += lowbit(x);
    }
}

int sum(int x)
{
    int ret = 0;
    while (x > 0)
    {
        ret += c[lson[use[x]]];
        x -= lowbit(x);
    }
    return ret;
}
```

```
int Query(int left, int right, int k)
{
    int left_root = T[left - 1];
    int right_root = T[right];
    int l = 0, r = m - 1;
    for (int i = left - 1; i; i -= lowbit(i)) use[i] = S[i];
    for (int i = right; i; i -= lowbit(i)) use[i] = S[i];
    while (l < r)
    {
        int mid = (l + r) / 2;
        int tmp = sum(right) - sum(left - 1) + c[lson[right_root]] - c[lson[left_root]];
        if (tmp >= k)
        {
            r = mid;
            for (int i = left - 1; i; i -= lowbit(i))
                use[i] = lson[use[i]];
            for (int i = right; i; i -= lowbit(i))
                use[i] = lson[use[i]];
            left_root = lson[left_root];
            right_root = lson[right_root];
        }
        else
        {
            l = mid + 1;
            k -= tmp;
            for (int i = left - 1; i; i -= lowbit(i))
                use[i] = rson[use[i]];
            for (int i = right; i; i -= lowbit(i))
                use[i] = rson[use[i]];
            left_root = rson[left_root];
            right_root = rson[right_root];
        }
    }
    return l;
}

void Modify(int x, int p, int d)
{
    while (x <= n)
    {
        S[x] = Insert(S[x], p, d);
        x += lowbit(x);
    }
}
```

```
int main()
{
    //freopen("input.txt","r",stdin);
    while (~scanf("%d", &n))
    {
        tot = 0;
        m = 0;
        q = maxn;
        for (int i = 1; i <= n; i++)
        {
            scanf("%d", &a[i]);
            t[m++] = a[i];
        }
        scanf("%d", &q);
        int op;
        for (int i = 0; i<q; i++)
        {
            scanf("%d", &op);
            if (op == 2)
            {
                query[i].kind = 0;
                scanf("%d%d%d", &query[i].l, &query[i].r, &query[i].k);
            }
            else
            {
                query[i].kind = 1;
                scanf("%d%d", &query[i].l, &query[i].r);
                t[m++] = query[i].r;
            }
        }
        Init_hash(m);
        T[0] = build(0, m - 1);
        for (int i = 1; i <= n; i++)
            T[i] = Insert(T[i - 1], get_hash(a[i]), 1);
        for (int i = 1; i <= n; i++)
            S[i] = T[0];
        for (int i = 0; i<q; i++)
        {
            if (query[i].kind == 0)
                printf("%d\n", t[Query(query[i].l, query[i].r, query[i].k)]);
            else
            {
                Modify(query[i].l, get_hash(a[query[i].l]), -1);
```

```
                Modify(query[i].l, get_hash(query[i].r), 1);

                a[query[i].l] = query[i].r;

            }

        }

    }

    return 0;

}
```

CDQ 分治

//

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

//

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

//

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

CDQ分治，即中序遍历

可解决降维问题，和动态转静态问题

顺序一般为(l, m) -> (l, r) -> (m + 1, r)

//

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

```
//////////////////////////////////////////////////////////////////////////////////////////////////////
////////////
//////////////////////////////////////////////////////////////////////////////////////////////////////
////////////
```

HDU 5432 Boring Class

序列 L0~Ln，R0~Rn

选取一些位置，要求L递增，R递减。

输出字典序最小的答案。

解法，因为需要输出字典序最小，需要从后向前进行CDQ

```
#include<iostream>

#include<cstdio>

#include<algorithm>

#include<cmath>

#include<cstring>

#include <stdio.h>

#include <string>


#define clr(x, y) memset(x, y, sizeof x)

#define inf 1e9
```

```
using namespace std;


template<class T>
inline bool read(T &n)
{
    T x = 0, tmp = 1;
    char c = getchar();
    while ((c < '0' || c > '9') && c != '-' && c != EOF) c = getchar();
    if (c == EOF) return false;
    if (c == '-') c = getchar(), tmp = -1;
    while (c >= '0' && c <= '9') x *= 10, x += (c - '0'), c = getchar();
    n = x*tmp;
    return true;
}
template <class T>
inline void write(T n)
{
    if (n < 0)
    {
        putchar('-');
        n = -n;
    }
    int len = 0, data[20];
    while (n)
    {
        data[len++] = n % 10;
        n /= 10;
    }
    if (!len) data[len++] = 0;
    while (len--) putchar(data[len] + 48);
}

const int maxn = 55005;

struct node
{
    int Li, Ri, id;

    bool operator < (const node & A) const
    {
        return Ri > A.Ri;
    }
```

```
} q[maxn], tmp[maxn];


int Li[maxn], Ri[maxn], su[maxn];

int dp[maxn], rmax[maxn];

int n;


void upp(int x, int a)

{

    while (x <= n)

    {

        rmax[x] = max(rmax[x], a);

        x += x & -x;

    }

}

void cl(int x)   /// CDQ分治中用到树状数组要这样清空

{

    while (x <= n)

    {

        rmax[x] = 0;

        x += x & -x;

    }

}
```

```
int get_max(int x)

{

    int ans = 0;

    while (x > 0)

    {

        ans = max(ans, rmax[x]);

        x -= x & -x;

    }

    return ans;

}



void CDQ(int L, int R)

{

    if (L == R)

    {

        int &ret = dp[q[L].id];

        ret = max(ret, 1);

        return;

    }



    int mid = (L + R) >> 1;
```

```
CDQ(mid + 1, R);

int L1 = L, L2 = mid + 1;

int ptr = mid + 1;

sort(q + L, q + mid + 1);
sort(q + mid + 1, q + R + 1);

for (int i = L; i <= mid; i++)
{
    while (ptr <= R && q[i].Ri <= q[ptr].Ri)
    {
        upp(q[ptr].Li, dp[q[ptr].id]);
        ptr++;
    }
    int &ret = dp[q[i].id];
    ret = max(ret, get_max(q[i].Li) + 1);
}

for (int i = mid + 1; i <= R; i++)    ///**特殊清空
    cl(q[i].Li);
    for (int i = L; i <= mid; i++)
    {
        while (q[i].id != i)
            swap(q[i], q[q[i].id]);
    }

    CDQ(L, mid);
}

int main()
{
    //freopen("input.txt","r",stdin);
    //freopen("output.txt", "w", stdout);
    int len;
    string ans;
    int   maxlen, rr;
    while (read(n))
    {
        for (int i = 0; i < n; i++)
        {
```

```
        read(q[i].Li);
        su[i] = q[i].Li;
        q[i].id = i;
        Li[i] = q[i].Li;
}
sort(su, su + n);
len = unique(su, su + n) - su;
for (int i = 0; i < n; i++)
        q[i].Li = lower_bound(su, su + len, q[i].Li) - su + 1;

for (int i = 0; i < n; i++)
{
        read(q[i].Ri);
        su[i] = q[i].Ri;
        Ri[i] = q[i].Ri;
}

clr(dp, 0);
clr(rmax, 0);
CDQ(0, n - 1);

//for (int i = 0; i < n; i++) printf("Ri=%d i=%d dp=%d\n", q[i].Ri, q[i].id, dp[q[i].id]);
```

```
        maxlen = 0; rr = -1; ans = "";

        for (int i = 0; i < n; i++) maxlen = max(maxlen, dp[i]);

        write(maxlen); putchar('\n');

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

        {

            if (dp[i] == maxlen)

            {

                if (rr == -1 || (Li[rr] >= Li[i] && Ri[rr] <= Ri[i]))

                {

                    maxlen--;

                    if (rr != -1) putchar(' ');

                    write(i + 1);

                    rr = i;

                }

            }

        }

        putchar('\n');

        ///cout<<"maxlen:"<<maxlen<<endl;

    }



    return 0;
```

```
}
```

```
///////////////////////////////////////////////////////////////////////////////////////////////////////
////////////
///////////////////////////////////////////////////////////////////////////////////////////////////////
////////////
///////////////////////////////////////////////////////////////////////////////////////////////////////
////////////
```

hdu 5354 Bipartite Graph

给一张图 ， 要求找到一个点，删除后成为二分图

解法 ： CDQ + 可撤销(种族)并查集

保证在进入（l, r）前，（l，r）内的边没有加入，之外的边全部加入

再递归处理

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <cmath>
#include <cstring>
#include <vector>
```

```
#include <queue>
#include <stack>
#include <set>

#pragma comment(linker, "/STACK:1024000000,1024000000")
#define lson l, m, rt<<1
#define rson m+1, r, rt<<1|1
#define inf 1e9
#define clr(x, y) memset(x, y, sizeof x)
using namespace std;

const int maxn = 200005;

struct edge
{
	int to, next;
}G[maxn << 1];

struct node
{
	int u, v, hu, hv, fau, fav, colu, colv;
	node(){}
```

```
    node(int a, int b, int c, int d, int e, int f, int g, int h)

    {

        u = a; v = b; hu = c; hv = d; fau = e; fav = f;

        colu = g;

        colv = h;

    }

};

stack<node> stk;


int head[maxn], si;

int h[maxn], fa[maxn];

int ans[maxn], n, m;

int col[maxn]; //0 代表相同 1 代表不同


void init(int _n)

{

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

    {

        fa[i] = i;

        h[i] = 1;

        ///

        col[i] = 0;
        }
}
void add(int u, int v)
{
        G[si].to = v;
        G[si].next = head[u];
        head[u] = si++;
}
int find_fa(int x)
{
        int o = x;
        while (fa[o] != o) o = fa[o];
        return o;
}
int find_col(int x)
{
        if (fa[x] == x) return 1;
        ///return 1^find_col(fa[x]);
        return col[x] ^ find_col(fa[x]);
}
bool Merge(int u, int v)
{
```

```
int a = find_fa(u), b = find_fa(v);

int x = find_col(u), y = find_col(v);

if (a == b)

{

    if (x == y)

    {

        //printf("~~%d %d~fu = %d fv = %d\n", u, v, a, b);

        return false;

    }

    return true;

}

stk.push(node(a, b, h[a], h[b], fa[a], fa[b], col[a], col[b]));

if (h[a] > h[b])

{

    fa[b] = a, h[a] += h[b];

    ///

    col[b] = x ^ y ^ 1;

}

else

{

    fa[a] = b, h[b] += h[a];

    ///
```

```
        col[a] = x ^ y ^ 1;
    }
    return true;
}

bool unite(int L, int R, int a, int b)
{
    for (int u = L; u <= R; u++)
    for (int i = head[u]; i != -1; i = G[i].next)
    {
        int v = G[i].to;
        if (a <= v && v <= b) continue;
        if (!Merge(u, v)) return false;
    }
    return true;
}

void get_del(int x)
{
    node tmp;
    while (stk.size() > x)
    {
        tmp = stk.top(); stk.pop();
        int u = tmp.u, v = tmp.v;
        h[u] = tmp.hu; h[v] = tmp.hv;
        fa[u] = tmp.fau; fa[v] = tmp.fav;
        /**/col[u] = tmp.colu; col[v] = tmp.colv;
    }
}

void cdq(int l, int r)
{
    if (l == r)
    {
        ans[l] = 1;
        return;
    }
    int pre = stk.size();
    int m = (l + r) >> 1;

    if (unite(m + 1, r, l, m))    /// 加入右面的边
        cdq(l, m);
    else{
        for (int i = l; i <= m; i++) ans[i] = 0;
    }
    get_del(pre);              /// 删去右面的边
    if (unite(l, m, m + 1, r))       /// 加入左面的边
        cdq(m + 1, r);
    else{
        for (int i = m + 1; i <= r; i++) ans[i] = 0;
    }
    get_del(pre);              /// 删去左面的边
    return;
}

int main()
{
    //freopen("input.txt", "r", stdin);
    int T;
    scanf("%d", &T);
    while (T--)
    {
        scanf("%d%d", &n, &m);
        init(n);
        clr(head, -1); si = 0;
        while (stk.size()) stk.pop();
        for (int i = 0, st, ed; i < m; i++)
        {
            scanf("%d%d", &st, &ed);
            add(st, ed); add(ed, st);
        }
        cdq(1, n);
        for (int i = 1; i <= n; i++) printf("%d", ans[i]);
        printf("\n");
    }
    return 0;
}

/////////////////////////////////////////////////////////////////////////////////////////////////////
////////////
/////////////////////////////////////////////////////////////////////////////////////////////////////
////////////
/////////////////////////////////////////////////////////////////////////////////////////////////////
////////////
```

bnu 12753 Arnooks's Defensive Line

插入一些区间，并查询有多少区间包含它

解法：裸CDQ动态转静态

（对于某些CDQ问题，可能可以使用先序遍历或后序遍历，但最好用中序遍历）

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <cmath>
#include <cstring>
#include <vector>
#include <queue>
#include <stack>
#include <set>
#include <vector>

//#pragma comment(linker, "/STACK:1024000000,1024000000")
#define lson l, m, rt<<1
#define rson m+1, r, rt<<1|1
#define inf 1e9
#define debug(a) cout << #a" = " << (a) << endl;
#define debugarry(a, n) for (int i = 0; i < (n); i++) { cout << #a"[" << i << "] = " << (a)[i]
<< endl; }
```

```
#define clr(x, y) memset(x, y, sizeof x)
using namespace std;

typedef unsigned long long uLL;

const int maxn = 500000 + 30;

int ans[maxn];
struct __sad
{
    int l, r;
    char c;
    int id;
    bool operator <(const __sad &a) const{
        if (l != a.l) return l<a.l;
        else return r < a.r;
    }
}A[maxn], B[maxn], C[maxn];

int rsum[maxn * 4];
void upp(int x, int add)
{
        while (x < maxn * 4)
        {
                rsum[x] += add;
                x += x&-x;
        }
}
void clear(int x)
{
        while (x < maxn * 4)
        {
                rsum[x] = 0;
                x += x&-x;
        }
}

int get(int x)
{
        int ret = 0;
        while (x)
        {
                ret += rsum[x];
                x -= x&-x;
        }

        return ret;

}



void CDQ(int l, int r)

{

        if (l == r) return;

        int mid = l + r >> 1;

        int lb = 0, lc = 0;



        for (int i = l; i <= mid; i++)

        {

                if (A[i].c == '+') B[lb++] = A[i];

        }

        for (int i = mid + 1; i <= r; i++)

        {

                if (A[i].c == '?') C[lc++] = A[i];

        }



        sort(B, B + lb);

        sort(C, C + lc);
        int sum = 0;


        int cn = 0;

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

        {

                while (cn<lb&&B[cn].l <= C[i].l)

                {

                        upp(B[cn].r, 1);

                        sum++;

                        cn++;

                }

                int ret = sum - get(C[i].r - 1);

                ans[C[i].id] += ret;

        }


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

                clear(B[i].r);


        CDQ(l, mid);

        CDQ(mid + 1, r);

}
```

```
int sub[maxn * 4], len;

int main()
{

    //freopen("input.txt", "r", stdin);
    int n;
    while (~scanf("%d", &n))
    {
        char s[30];
        len = 0;
        for (int i = 1; i <= n; i++)
        {
            scanf("%s%d%d", s, &A[i].l, &A[i].r);
            A[i].c = *s;
            A[i].id = i;
            sub[len++] = A[i].l;
            sub[len++] = A[i].r;
        }
        sort(sub, sub + len);
        len = unique(sub, sub + len) - sub;
        for (int i = 1; i <= n; i++)
        {
            A[i].l = lower_bound(sub, sub + len, A[i].l) - sub + 1;
            A[i].r = lower_bound(sub, sub + len, A[i].r) - sub + 1;
        }
        clr(ans, 0);
        clr(rsum, 0);
        CDQ(1, n);
        for (int i = 1; i <= n; i++) if (A[i].c == '?')
            printf("%d\n", ans[i]);
    }
    return 0;
}
```

整体二分

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

```
#include <vector>
#include <queue>
#include <stack>
#include <set>
#include <vector>
#include <deque>
#include <set>

//#pragma comment(linker, "/STACK:1024000000,1024000000")
#define lson l, m, ls[rt]
#define rson m+1, r, rs[rt]
#define inf 1e9
#define debug(a) cout << #a" = " << (a) << endl;
#define debugarry(a, n) for (int i = 0; i < (n); i++) { cout << #a"[" << i << "] = " << (a)[i] << endl; }
#define clr(x, y) memset(x, y, sizeof x)
#define LL long long
using namespace std;

const int maxn = 2e6;
struct __sad
{
    int l, r, id;
    int st, k, v, add;
}p[maxn], p1[maxn], p2[maxn];
int ans[maxn];
int n;

int rsum[maxn];

void upp(int x, int add)
{
    while (x <= n)
    {
        rsum[x] += add;
        x += x&-x;
    }
}

int get(int x)
{
    int ret = 0;
    while (x)
    {
        ret += rsum[x];
        x -= x&-x;
    }
    return ret;
}

void Bin(int st, int ed, int l, int r)
{
    if (st>ed) return;
    if (l == r)
    {
        for (int i = st; i <= ed; i++) if (p[i].st == 2)
            ans[p[i].id] = l;
        return;
    }
    int mid = l + (r - l) / 2;
    int ta1 = 0, ta2 = 0;
    for (int i = st; i <= ed; i++)
    {
        if (p[i].st == 1)
        {
            if (p[i].v <= mid)
            {
                p1[ta1++] = p[i];
                upp(p[i].l, p[i].add);
            }
            else    p2[ta2++] = p[i];
        }
        else
        {
            int t = get(p[i].r) - get(p[i].l - 1);
            if (t >= p[i].k) p1[ta1++] = p[i];
            else p[i].k -= t, p2[ta2++] = p[i];
        }
    }
    for (int i = st; i <= ed; i++)
    {
        if (p[i].st == 1)
        if (p[i].v <= mid)
            upp(p[i].l, p[i].add*(-1));
    }
    for (int i = 0; i<ta1; i++)
        p[i + st] = p1[i];
    for (int i = 0; i<ta2; i++)
```

```
        p[i + st + ta1] = p2[i];

    Bin(st, st + ta1 - 1, l, mid);
    Bin(st + ta1, ed, mid + 1, r);
}

int A[maxn];
int sub[maxn], len;

int main()
{
    //freopen("1007.in", "r", stdin);
    //freopen("output.txt","w",stdout);
    int q;
    while (~scanf("%d", &n))
    {
        int cnt = 0, acnt = 0;
        int a, st, l, r, v, k;
        len = 0;
        for (int i = 1; i <= n; i++)
        {
            scanf("%d", &a);
    A[i] = a;
    sub[len++] = a;

    p[cnt].l = i; p[cnt].st = 1;
    p[cnt].v = a; p[cnt].add = 1;

    cnt++;
}
scanf("%d", &q);
for (int i = 0; i<q; i++)
{
    scanf("%d", &st);
    if (st == 1)
    {
        scanf("%d%d", &l, &v);

        sub[len++] = v;

        p[cnt].l = l; p[cnt].st = 1;
        p[cnt].v = A[l]; p[cnt].add = -1;
        cnt++;
        p[cnt].l = l; p[cnt].st = 1;
        p[cnt].v = A[l] = v; p[cnt].add = 1;
        cnt++;
    }
    else{
        scanf("%d%d%d", &l, &r, &k);
        p[cnt].l = l; p[cnt].r = r;
        p[cnt].st = 2; p[cnt].k = k;
        p[cnt].id = acnt++;
        cnt++;
    }
}

sort(sub, sub + len);
len = unique(sub, sub + len) - sub;

for (int i = 0; i<cnt; i++) if (p[i].st == 1)
{
    p[i].v = lower_bound(sub, sub + len, p[i].v) - sub + 1;
}

Bin(0, cnt - 1, 0, len + 7);
```

```
        for (int i = 0; i<acnt; i++)
            printf("%d\n", sub[ans[i] - 1]);
    }
    return 0;
}
```

2-SAT

```
/*
复杂度O(n)
*/
struct two_sat
{
    int n;
    vector<int> G[maxn << 1];
    bool mark[maxn << 1];
    int S[maxn << 1], c;

    bool dfs(int x)
    {
```

```
    if (mark[x ^ 1]) return false;
    if (mark[x]) return true;
    mark[x] = true;
    S[c++] = x;
    for (int i = 0; i < G[x].size(); i++)
    {
        if (!dfs(G[x][i])) return false;
    }
    return true;
}
void init(int n)
{
    this->n = n;
    for (int i = 0; i < (n << 1); i++) G[i].clear();
    clr(mark, 0);
}
// x = xval or y = yval
//2i + 1 means true 2i means false
void add_clause(int x, int xval, int y, int yval)
{
    x = x * 2 + xval;
    y = y * 2 + yval;
```

```
        G[x ^ 1].push_back(y);
        G[y].push_back(x ^ 1);
        G[y ^ 1].push_back(x);
        G[x].push_back(y ^ 1);
    }
    bool solve()
    {
        for (int i = 0; i < (n << 1); i += 2)
        if (!mark[i] && !mark[i + 1])
        {
            c = 0;
            if (!dfs(i))
            {
                while (c > 0) mark[S[--c]] = false;
                if (!dfs(i + 1)) return false;
            }
        }
        return true;
    }
} sat;
```

ISAP 网络流

```
const int maxn = 1050;

const int maxm = 100005;



struct edge

{

	int to, next;

	int cap;

} G[maxm];



int head[maxn], si;



void add(int st, int ed, int val)

{

	G[si].to = ed;

	G[si].cap = val;

	G[si].next = head[st];

	head[st] = si++;

	G[si].to = st;

	G[si].cap = 0;

	G[si].next = head[ed];
```

```
        head[ed] = si++;

}



int nn, n, m;

int h[maxn], gap[maxn];

int source, sink;



int dfs(int u, int cost)

{

        if (u == sink) return cost;



        int minh = nn - 1, lv = cost, d;



        for (int i = head[u]; i != -1; i = G[i].next)

        {

                int v = G[i].to;

                int val = G[i].cap;

                if (val > 0)

                {

                        if (h[v] + 1 == h[u])

                        {

                                d = min(val, lv);
                d = dfs(v, d);
                G[i].cap -= d;
                G[i ^ 1].cap += d;
                lv -= d;

                if (h[source] >= nn) return cost - lv;
                if (lv == 0) break;
            }
            if (h[v] < minh) minh = h[v];
        }
    }
    if (lv == cost)
    {
        --gap[h[u]];
        if (gap[h[u]] == 0) h[source] = nn;
        h[u] = minh + 1;
        ++gap[h[u]];
    }
    return cost - lv;
}

int sap(int st, int ed)
```

```
{
    source = st;
    sink = ed;
    int ret = 0;
    clr(gap, 0); clr(h, 0);

    gap[st] = nn;

    while (h[st] < nn)
    {
        ret += dfs(st, inf);
    }

    return ret;
}
```

最小费用最大流

```
const int maxn = 2000;
const int maxm = 2500000;

struct edge
```

```
{
        int to, next;
        int cap, flow, cc;
} G[maxm];

int head[maxn], si;
int pre[maxn], dis[maxn];
bool vis[maxn];
int nn, n, m, k;

void add(int st, int ed, int val, int cost)
{
        G[si].to = ed;
        G[si].cap = val;
        G[si].cc = cost;
        G[si].flow = 0;
        G[si].next = head[st];
        head[st] = si++;
        G[si].to = st;
        G[si].cap = 0;
        G[si].cc = -cost;
        G[si].flow = 0;
```

```
    G[si].next = head[ed];
    head[ed] = si++;
}

bool spfa(int s, int t)
{
    queue<int> que;
    for (int i = 0; i < nn; i++)
    {
        dis[i] = inf;
        vis[i] = false;
        pre[i] = -1;
    }
    dis[s] = 0;
    vis[s] = true;
    que.push(s);

    while (que.size())
    {
        int u = que.front(); que.pop();
        vis[u] = false;
        for (int i = head[u]; i != -1; i = G[i].next)
        {
            int v = G[i].to;
            if (G[i].cap > G[i].flow &&
                dis[v] > dis[u] + G[i].cc)
            {
                dis[v] = dis[u] + G[i].cc;
                pre[v] = i;
                if (!vis[v])
                {
                    vis[v] = true;
                    que.push(v);
                }
            }
        }
    }
    // if (dis[t] > 0) return false; 加上这句话可以保证只求出最小费用，不要求最大流
    if (pre[t] == -1) return false;
    return true;
}

int min_cost_maxflow(int s, int t, int &cost)
{
    int flow = 0;

    cost = 0;

    while (spfa(s, t))

    {

        int rmin = inf;

        for (int i = pre[t]; i != -1; i = pre[G[i ^ 1].to])

            rmin = min(rmin, G[i].cap - G[i].flow);

        for (int i = pre[t]; i != -1; i = pre[G[i ^ 1].to])

        {

            G[i].flow += rmin;

            G[i ^ 1].flow -= rmin;

            cost += G[i].cc * rmin;

        }

        flow += rmin;

    }

    return flow;

}
```

尺取法

```
/*

HDU 5289
```

```
求一个序列连续区间最大值-最小值不超过K的个数

*/

#include <cstdio>

#include <iostream>

#include <algorithm>

#include <cstring>

#include <string>

#include <cmath>

#include <queue>

#include <bitset>

#define clr(x, y) memset(x, y, sizeof x)

#define inf 1000000000

using namespace std;



typedef long long LL;



const int maxn = 200010;



int A[maxn];

int dmax[maxn][30];

int dmin[maxn][30];

int flog[(maxn << 1) + 10];
int n, K;

int rmax_init(int A[])
{
    for (int i = 0; i < n; i++) dmax[i][0] = A[i];
    for (int j = 1; (1 << j) <= n; j++)
    for (int i = 0; i + (1 << j) - 1 < n; i++)
        dmax[i][j] = max(dmax[i][j - 1], dmax[i + (1 << (j - 1))][j - 1]);
}
int rmax_find(int L, int R)
{
    int k = flog[R - L + 1];
    return max(dmax[L][k], dmax[R - (1 << k) + 1][k]);
}
int rmin_init(int A[])
{
    for (int i = 0; i < n; i++) dmin[i][0] = A[i];
    for (int j = 1; (1 << j) <= n; j++)
    for (int i = 0; i + (1 << j) - 1 < n; i++)
        dmin[i][j] = min(dmin[i][j - 1], dmin[i + (1 << (j - 1))][j - 1]);
}
int rmin_find(int L, int R)
```

```
{
    int k = flog[R - L + 1];
    return min(dmin[L][k], dmin[R - (1 << k) + 1][k]);
}

int main()
{

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

    flog[0] = -1;
    for (int i = 1; i < 2 * maxn; i++) flog[i] = flog[i >> 1] + 1;

    int T, he, ta;
    LL ans;
    scanf("%d", &T);
    while (T--)
    {
        scanf("%d%d", &n, &K);
        for (int i = 0; i < n; i++) scanf("%d", &A[i]);

        rmax_init(A); rmin_init(A);
        he = ta = 0;
        ans = 0;
        while (he < n)
        {
            while (ta < n && rmax_find(he, ta) - rmin_find(he, ta) < K) { ta++; }
            ans += ta - he;
            he++;
        }
        printf("%I64d\n", ans);
    }

    return 0;
}
```

倍增法求 LCA

```
/*
倍增法求LCA，复杂度O(nlogn)
*/

const int maxn = 100000;
```

```
}



const int maxk = 30;



struct edge

{

	int to, next;

} G[maxn << 1];



int head[maxn], si;

int parent[maxk][maxn]; //注意第一维为小的

int depth[maxn];



void dfs(int u, int p, int d)

{

	parent[0][u] = p;

	depth[u] = d;

	for (int i = head[u]; i != -1; i = G[i].next)

	{

		int v = G[i].to;

		if (v != p) dfs(v, u, d + 1);

	}

}

void init_lca(int _n)
```

```
{
    dfs(1, -1, 0);
    for (int k = 0; k + 1 < maxk; k++)
    {
        for (int u = 1; u <= _n; u++) //注意下标是0~n-1  还是1~n
        {
            if (parent[k][u] < 0) parent[k + 1][u] = -1;
            else parent[k + 1][u] = parent[k][parent[k][u]];
        }
    }
}
int get_lca(int u, int v)
{
    if (depth[u] > depth[v]) swap(u, v);
    for (int k = 0; k < maxk; k++)
    {
        if ((depth[v] - depth[u]) >> k & 1)
        {
            v = parent[k][v];
        }
    }
    if (u == v) return u;
    for (int k = maxk - 1; k >= 0; k--)
    {
        if (parent[k][u] != parent[k][v])
        {
            u = parent[k][u];
            v = parent[k][v];
        }
    }
    return parent[0][u];
}
```

边双连通分量

```
const int maxn = 2000;
const int maxm = 550;

struct edge
{
    int to, next;
}G[maxn << 1];

int head[maxn], si;
int pre[maxn], dfs_clock, bridge;

int par[maxn], n;



void init(int _n)

{

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

        par[i] = i;

}

int Find(int x)

{

    if (x == par[x]) return x;

    return par[x] = Find(par[x]);

}

void unite(int x, int y)

{

    x = Find(x);

    y = Find(y);

    if (x == y) return;

    par[x] = y;

}



int Tarjan(int u, int fa)
```

```
{
    int lowu = pre[u] = ++dfs_clock;
    for (int i = head[u]; i != -1; i = G[i].next)
    {
        int v = G[i].to;
        if (v == fa) continue;
        if (pre[v] == 0)
        {
            int lowv = Tarjan(v, u);
            lowu = min(lowu, lowv);
            if (lowv <= pre[u]) unite(u, v);
            // 表示是桥
            else if (lowv > pre[u]) bridge++;
        }
        else lowu = min(lowu, pre[v]);
    }
    return lowu;
}

void find_bridge(int _n)
{
    clr(pre, 0);
    dfs_clock = 0;

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

    if (!pre[i]) Tarjan(i, -1);

}
```

点双连通分量

```
#include <cstdio>

#include <iostream>

#include <algorithm>

#include <cstring>

#include <vector>

#include <queue>

#define clr(x, y) memset(x, y, sizeof x)

#define inf 1000000000

using namespace std;

typedef long long LL;



const int maxn = 1505;



int pre[maxn], iscut[maxn], bccno[maxn], dfs_clock, bcc_cnt;

vector<int> G[maxn], bcc[maxn];
```

```
int n, m;


struct edge

{

    int st, ed;

};




stack<edge> S;


int dfs(int u, int fa)

{

    int lowu = pre[u] = ++dfs_clock;

    int child = 0;

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

    {

        int v = G[u][i];

        Edge e = (Edge){ u, v };

        if (!pre[v])

        {

            S.push(e);

            child++;
```

```
int lowv = dfs(v, u);

lowu = min(lowu, lowv);

if (lowv >= pre[u])

{

        iscut[u] = true;

        bcc_cnt++; bcc[bcc_cnt].clear();

        for (;;)

        {

                edge x = S.top(); S.pop();

                if (bccno[x.u] != bcc_cnt)

                {

                        bcc[bcc_cnt].push_back(x.u);

                        bcc_cnt[x.u] = bcc_cnt;

                }

                if (bccno[x.v] != bcc_cnt)

                {

                        bcc[bcc_cnt].push_back(x.v);

                        bcc_cnt[x.v] = bcc_cnt;

                }

                if (x.u == u && x.v == v) break;

        }

}
```

```
        }
        else if (pre[v] < pre[u] && v != fa)
        {
            S.push(e);
            lowu = min(lowu, pre[v]);
        }
    }
    if (fa < 0 && child == 1) iscut[u] = 0;
    return lowu;
}
void find_bcc(int n)
{
    clr(pre, 0); clr(iscut, 0); clr(bccno, 0);
    dfs_clock = bcc_cnt = 0;
    for (int i = 0; i < n; i++)
    {
        if (!pre[i]) dfs(i, -1);
    }
}
```

LCT 动态树

```
/*

包含最基本的加入操作，删除操作，询问两个点是否联通

*/


#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;


const int MAXN = 10005;


struct Node* null;


struct Node {

    Node* c[2];

    Node* f;

    int flip;


    void newnode() {

        c[0] = c[1] = f = null;
        flip = 0;
    }

    void reverse() {
        if (this == null) return;
        swap(c[0], c[1]);
        flip ^= 1;
    }

    void link_child(Node* o, int d) {
        c[d] = o;
        o->f = this;
    }

    int is_root() {
        return f == null || f->c[0] != this && f->c[1] != this;
    }

    void push_down() {
        if (flip) {
            c[0]->reverse();
            c[1]->reverse();
            flip = 0;
        }
    }

    void sign_down() {
        if (!is_root()) f->sign_down();
        push_down();
    }

    void rotate(int d) {
        Node* p = f;
        Node* g = p->f;
        p->link_child(c[d], !d);
        if (!p->is_root()) {
            if (p == g->c[0]) g->link_child(this, 0);
            else g->link_child(this, 1);
        }
        else f = g;
        this->link_child(p, d);
    }

    void splay() {
        sign_down();
        while (!is_root()) {
            if (f->is_root()) rotate(this == f->c[0]);
            else {
                if (f == f->f->c[0]) {
                    if (this == f->c[0]) f->rotate(1), rotate(1);
                    else rotate(0), rotate(1);
                }
                else {
                    if (this == f->c[1]) f->rotate(0), rotate(0);
                    else rotate(1), rotate(0);
                }
            }
        }
    }

    void access() {
        Node* o = this;
        Node* x = null;
        while (o != null) {
            o->splay();
            o->link_child(x, 1);
            x = o;
            o = o->f;
        }
        splay();
    }

    Node* find_root() {
        access();
        Node* o = this;
        while (o->c[0] != null) o = o->c[0];
        return o;
    }

    void make_root() {
        access();
        reverse();
    }

    void cut() {
        access();
        c[0]->f = null;
        c[0] = null;
```

```
        }

        void cut(Node* o) {
                if (o->find_root() != find_root()) return;
                make_root();
                o->cut();
        }

        void link(Node* o) {
                if (o == this || o->find_root() == find_root()) return;
                make_root();
                f = o;
        }

        void query(Node* o) {
                if (o->find_root() == find_root()) printf("Yes\n");
                else printf("No\n");
        }
};

Node pool[MAXN];
Node* node[MAXN];
```

```
Node* cur;

int n, m;

void clear() {
	cur = pool;
	null = cur++;
	null->newnode();
}

void solve() {
	char s[20];
	int x, y;
	clear();
	for (int i = 1; i <= n; ++i) {
		node[i] = cur++;
		node[i]->newnode();
	}
	while (m--) {
		scanf("%s%d%d", s, &x, &y);
		if (s[0] == 'C') node[x]->link(node[y]);
		if (s[0] == 'D') node[x]->cut(node[y]);
```

```
        if (s[0] == 'Q') node[x]->query(node[y]);
    }
}

int main() {
    while (~scanf("%d%d", &n, &m)) solve();
    return 0;
}
```

强连通分量缩环

```
const int maxn = 1000;

struct edge
{
    int to, next;
}G[maxn << 1];

int head[maxn], si;
int pre[maxn], lowlink[maxn], sccno[maxn], dfs_clock, scc_cnt;
stack<int> S;

void Tarjan(int u)
```

```
{
    pre[u] = lowlink[u] = ++dfs_clock;
    S.push(u);
    for (int i = head[u]; i != -1; i = G[i].next)
    {
        int v = G[i].to;
        if (!pre[v])
        {
            Tarjan(v);
            lowlink[u] = min(lowlink[u], lowlink[v]);
        }
        else if (!sccno[v])
        {
            lowlink[u] = min(lowlink[u], pre[v]);
        }
    }
    if (lowlink[u] == pre[u])
    {
        scc_cnt++;
        while (1)
        {
            int x = S.top(); S.pop();
            sccno[x] = scc_cnt;
            if (x == u) break;
        }
    }
}
void find_scc(int _n)
{
    dfs_clock = scc_cnt = 0;
    clr(sccno, 0); clr(pre, 0);
    for (int i = 0; i < _n; i++)
    if (!pre[i]) Tarjan(i);
}
```

二分图匹配

```
/*
二分图匹配模板
跑点数较少的一侧点， 复杂度为O(VE)
对于一般图匹配，将可以将点数乘以2建成二分图，通过二分图匹配可以求出来正确的答案(ans/2)
*/
const int maxn = 3000;
```

```
vector<int> G[maxn];
int match[maxn], used[maxn];

bool dfs(int u, int mark)
{
    used[u] = mark;
    for (int i = 0; i < G[u].size(); i++)
    {
        int v = G[u][i];
        int w = match[v];
        if (w < 0 || used[w] != mark && dfs(w, mark))
        {
            match[v] = u;
            match[u] = v;
            return true;
        }
    }
    return false;
}

int bipartite_matching(int _n)
```

```
{
    int res = 0;
    clr(match, -1);
    for (int u = 0; u < _n; u++)
    {
        if (match[u] < 0)
        {
            if (dfs(u, u)) res++;
        }
    }
    return res;
}
```