ST 2D

```
include <bits/stdc++.h>
using namespace std;

/*------- Constants---- */
#define LMT            605
#define ll             long long
#define ull            unsigned long long
#define mod            1000000007
#define MEMSET_INF     63
#define MEM_VAL        1061109567
#define FOR(i,n)           for( int i=0 ; i < n ; i++ )
#define mp(i,j)        make_pair(i,j)
#define lop(i,a,b)     for( int i = (a) ; i < (b) ; i++)
#define pb(a)          push_back((a))
#define gc             getchar_unlocked
#define PI             acos(-1.0)
#define inf            1<<30
#define lc             ((n)<<1)
#define rc             ((n)<<1 |1)
#define msg(x)         cout<<x<<endl;

/*---- short Cuts ------- */
#define ms(ara_name,value) memset(ara_name,value,sizeof(ara_name))
typedef pair<int, int> ii;
typedef vector<int > vi ;
/*------ template functions ------ */
inline void sc(int &x)
{
   register int c = gc();
   x = 0;
   int neg = 0;
   for(;((c<48 | c>57) && c != '-');c = gc());
   if(c=='-') {neg=1;c=gc();}
   for(;c>47 && c<58;c = gc()) {x = (x<<1) + (x<<3) + c - 48;}
   if(neg) x=-x;
}

template <class T> inline T bigmod(T p,T e,T M){
   ll ret = 1;
   for(; e > 0; e >>= 1){
      if(e & 1) ret = (ret * p) % M;
      p = (p * p) % M;
   } return (T)ret;
}
template <class T> inline T gcd(T a,T b){if(b==0)return a;return gcd(b,a%b);}
template <class T> inline T modinverse(T a,T M){return bigmod(a,M-2,M);}

/**************************** END OF TEMPLATE ****************************/
```

```
int t[4*LMT][4*LMT];
int p[4*LMT][4*LMT];
int a[LMT][LMT];
int m , n ;
void build_y (int vx, int lx, int rx, int vy, int ly, int ry) {
   if (ly == ry)
      if (lx == rx){
         t[vx][vy] = a[lx][ly];
         p[vx][vy] = a[lx][ly];
      }

      else {
         t[vx][vy] = max(t[vx*2][vy] , t[vx*2+1][vy] );
         p[vx][vy] = min(p[vx*2][vy] , p[vx*2+1][vy] );
      }

      else {
         int my = (ly + ry) / 2;
         build_y (vx, lx, rx, vy*2, ly, my);
         build_y (vx, lx, rx, vy*2+1, my+1, ry);
         t[vx][vy] = max(t[vx][vy *2] , t[vx][vy *2 + 1]);
         p[vx][vy] = min(p[vx][vy *2] , p[vx][vy *2 + 1]);
      }
}

void build_x (int vx, int lx, int rx) {
   if (lx != rx) {
      int mx = (lx + rx) / 2;
      build_x (vx*2, lx, mx);
      build_x (vx*2+1, mx+1, rx);
   }
   build_y (vx, lx, rx, 1, 0, m-1);
}
```

```
void update_y (int vx, int lx, int rx, int vy, int ly, int ry, int x, int y, int new_val) {
   if (ly == ry) {
      if (lx == rx){
         t[vx][vy] = new_val;
         p[vx][vy] = new_val;
      }
      else {
         t[vx][vy] = max(t[vx*2][vy] , t[vx*2+1][vy] );
         p[vx][vy] = min(p[vx*2][vy] , p[vx*2+1][vy] );
      }

   }
   else {
      int my = (ly + ry) / 2;
      if (y <= my) update_y (vx, lx, rx, vy*2, ly, my, x, y, new_val);
      else update_y (vx, lx, rx, vy*2+1, my+1, ry, x, y, new_val);

      t[vx][vy] = max(t[vx][vy *2] , t[vx][vy *2 + 1]);
      p[vx][vy] = min(p[vx][vy *2] , p[vx][vy *2 + 1]);
   }
}

void update_x (int vx, int lx, int rx, int x, int y, int new_val) {
   if (lx != rx) {
      int mx = (lx + rx) / 2;
      if (x <= mx) update_x (vx*2, lx, mx, x, y, new_val);
      else update_x (vx*2+1, mx+1, rx, x, y, new_val);
   }
   update_y (vx, lx, rx, 1, 0, m-1, x, y, new_val);
}


ii query_y (int vx, int vy, int tly, int try_, int ly, int ry) {

   if (ly > ry) return ii(-1,inf );

   if (ly == tly && try_ == ry) return ii( t[vx][vy] ,p[vx][vy]);

   int tmy = (tly + try_) / 2;

   ii r1 = query_y (vx, vy*2, tly, tmy, ly, min(ry,tmy));
   ii r2 = query_y (vx, vy*2+1, tmy+1, try_, max(ly,tmy+1), ry);

   return ii( max(r1.first , r2.first) , min(r1.second, r2.second)) ;

}
```

```
ii query_x (int vx, int tlx, int trx, int lx, int rx, int ly, int ry) {
    if (lx > rx)
        return ii(-1,inf );
    if (lx == tlx && trx == rx)
        return query_y (vx, 1, 0, m-1, ly, ry);
    int tmx = (tlx + trx) / 2;

    ii r1 = query_x (vx*2, tlx, tmx, lx, min(rx,tmx), ly, ry);
    ii r2 = query_x (vx*2+1, tmx+1, trx, max(lx,tmx+1), rx, ly, ry);

    return ii( max(r1.first , r2.first) , min(r1.second, r2.second)) ;
}

int val ;

int main() {

    int  q ;
    sc(n); sc(m);
    FOR(i , n) FOR(j , m) sc(a[i][j]);
    build_x(1, 0 , n-1);
    sc(q);
    char ch ;
    int x1,  x2 , y1 , y2;
    while (q -- ) {
        do {
            ch = gc();
        } while (ch =='\n'||ch =='\0');

        if(ch == 'q') {
            sc(x1) ; sc(y1) ;sc(x2);sc(y2);
            x1 -- , y1 -- ,x2 -- , y2 --;
            ii r = query_x( 1, 0 , n -1 , x1 , x2 , y1 , y2);
            printf("%d %d\n" ,r.first , r.second);
        }
        else {
            sc(x1) ;sc(y1 );sc(val);
            x1 -- , y1--;
            update_x(1, 0, n-1 ,x1 , y1,val);
        }

    }

    return 0;
}
```

Treap Set

```
#include <bits/stdc++.h>

using namespace  std;
typedef unsigned long long ll;
unsigned next() {
	static unsigned x(987654321), y(123456789), z(521288629), w(136751267);
	unsigned t = x ^ x << 11;
	x = y, y = z, z = w;
	return w = ((t ^ t >> 8) ^ w) ^ w >> 19;
}
struct treap {
	ll key;
	int prior;
	ll V;
	int sz;
	ll sum;
	treap *l, *r;
	treap() {}
	treap(ll _, ll __) {
		key = _;
		V = __;
		sum = V;
		prior = next();
		l = r = NULL;
		sz = 1;
	}
	void resize() {
		sz = 1;
		sum = V;
		if (l) sum += l->sum;
		if (r) sum += r->sum;
		if (l)sz += l->sz;
		if (r)sz += r->sz;
	}
};
typedef treap * Node;


void split(Node t, Node &l, Node &r, ll key) {
	if (t == NULL){l = r = NULL;return;}
	if (t->key <= key) {
		split(t->r, t->r, r, key);
		l = t;
	//	if(l)l->resize();
	} else {
		split(t->l, l, t->l, key);
		r = t;
//		if(r)r->resize();
	}
	t->resize();
}
```

```
Node merge(Node l, Node r) {
	if (l == NULL)return r;
	if (r == NULL)return l;
	if (l->prior > r->prior) {
		l->r = merge(l->r, r);
		l->resize();
		return l;
	} else {
		r->l = merge(l, r->l);
		r->resize();
		return r;
	}
}

int depth;
void print(Node node, int d = 0) {
	depth = max(depth, d);
	if (node == NULL)return;
	print(node->l, d + 1);
	cout << node->key <<  " -> " << node->sz << " " << node->sum << endl;
	print(node->r, d + 1);
}
void insert(Node &root, ll key, ll v) {
	Node l, r;
	split(root, l, r, key);
	root = merge(l, merge(new treap(key, v), r));
}
ll sum(Node t) {
	if (!t)return 0LL;
	return t->sum;
}
int sz(Node t) {
	if (!t)return 0;
	return t->sz;
}
```

HLD EDGES

```
const int maxn=10010;
vector<pair<int,int> >G[maxn];
vector<int>ind[maxn];
int arcopos[maxn];//donde estara mi arco en la cadena
int n;
class SegmentTree {
  int t[2 * maxn];
  int n;
  public:
  void init(int _n) {
    n = _n;
    memset(t, 0, sizeof(t));
  }
  void set(int position, int value) {
    position += n;
    t[position] = value;
    for (position >>= 1; position > 0; position >>= 1)
      t[position] = max(t[position << 1], t[(position << 1) | 1]);
  }
  int query(int l, int r) {
    l += n;
    r += n;
    int ans = 0;
    while (l < r) {
      if (l & 1) ans = max(ans, t[l++]);
      if (r & 1) ans = max(ans, t[--r]);
      l >>= 1;
      r >>= 1;
    }
    return ans;
  }
};
int pos=0,root=0;
int ncad=0;

struct HLD{
        int head[maxn];
        int where[maxn];
        int chainIdx[maxn];
        int sz[maxn];
        int parent[maxn];
        int depth[maxn];
        int val[maxn];
        SegmentTree tree;
        void init(){
                for(int i=0;i<=n;i++)head[i]=where[i]=-1,depth[i]=sz[i]=0;
                pos=0;
                root=0;
                ncad=0;
                depth[root]=0;
                dfs(root,-1);
```

```
	descompose(root,-1);
	tree.init(n);
	for(int i=0;i<n;i++){
		tree.set(i,val[i]);
	}
}
void print(){
	cout<<"pa "<<endl;
	for(int i=0;i<n;i++)cout<<parent[i]<<" ";
	cout<<endl;
	cout<<"sizes "<<endl;
	for(int i=0;i<n;i++)cout<<sz[i]<<" ";
	cout<<endl;
	cout<<ncad<<" "<<pos<<" "<<n<<endl;
	for(int i=0;i<=n;i++)cout<<where[i]<<" ";
	cout<<endl;
}
void dfs(int u,int prev){
	sz[u]=1;
	parent[u]=prev;
	for(int i=0;i<G[u].size();i++){
		int v=G[u][i].first,w=G[u][i].second;
		if(v!=prev){
			depth[v]=depth[u]+1;
			arcopos[ind[u][i]]=G[u][i].first;
			dfs(v,u);
			sz[u]+=sz[v];
		}
	}
}
void descompose(int u,int cost){
	if(head[ncad]==-1){
		head[ncad]=u;
	}
	where[u]=ncad;
	val[pos]=cost;
	chainIdx[u]=pos++;
	int sc=-1,maxi=-1,scost=-1;
	for(int i=0;i<G[u].size();i++){
		pair<int,int> node=G[u][i];
		int v=node.first,w=node.second;
		if(where[v]==-1){
			if(sz[v]>maxi){
				sc=v;
				maxi=sz[v];
				scost=w;
			}
		}
	}
	if(sc!=-1)descompose(sc,scost);
```

```
		for(int i=0;i<G[u].size();i++){
			pair<int,int>node=G[u][i];
			int v=node.first,w=node.second;
			if(where[v]==-1){
				ncad++;
				descompose(v,w);
			}
		}
	}
	void update(int u,int val){
		//cout<<u<<" "<<arcopos[u]<<" arco "<<endl;
		tree.set(chainIdx[arcopos[u]],val);
	}
	int lca(int u,int v){
		while(where[u]!=where[v]){
			int uChain=where[u],vChain=where[v];
			int hu=head[uChain],hv=head[vChain];
			if(depth[hu]>depth[hv])u=parent[hu];
			else v=parent[hv];
		}
		return depth[u]<depth[v]?u:v;
	}
	int Q(int u,int v){
		//cout<<u<<" "<<v<<endl;
		if(u==v)return 0;
		int vChain=where[v];
		int ans=0;
		while(true){
			int uChain=where[u];
		//	cout<<"cadenas "<<uChain<<" "<<vChain<<endl;
			if(uChain==vChain){
			//	cout<<"estoy en "<<u<<" "<<v<<endl;
				if(u==v)break;
			//	cout<<"posiciones "<<chainIdx[v]<<" "<<chainIdx[u]+1<<endl;
				ans=max(ans,tree.query(chainIdx[v]+1,chainIdx[u]+1));
				break;
			}
			int hu=head[uChain];
			//cout<<"paso a la cadena "<<hu<<" "<<parent[hu]<<"
"<<where[parent[hu]]<<endl;
			//cout<<" mi head "<<hu<<endl;
			ans=max(ans,tree.query(chainIdx[hu],chainIdx[u]+1));
			u=parent[hu];
		}
		return ans;
	}
	int query(int u,int v){
		int L=lca(u,v);
	//	cout<<"LCA "<<L<<endl;
	//	cout<<"first "<<u<<" "<<L<<" ";cout<<Q(u,L)<<endl;cout<<" second "<<v<<"
"<<" "<<L;cout<<Q(v,L)<<endl;
		return max(Q(u,L),Q(v,L));}}
```

SUFFIX ARRAY

```
#define MAX_LEN 1000010
#define ALPH_SIZE 123
typedef char tipo;
tipo s[MAX_LEN+1];
int N,sa[MAX_LEN],rk[MAX_LEN];
int cont[MAX_LEN],nxt[MAX_LEN];
bool bh[MAX_LEN+1],b2h[MAX_LEN+1];
int idx[MAX_LEN];
int cover[MAX_LEN];
void build_suffix_array(){
    N = strlen(s);
    memset(cont,0,sizeof(cont));
    for(int i = 0;i<N;++i) ++cont[s[i]];
    for(int i = 1;i<ALPH_SIZE;++i) cont[i] += cont[i-1];
    for(int i = 0;i<N;++i)sa[--cont[s[i]]] = i;
    for(int i = 0;i<N;++i){
        bh[i] = (i==0 || s[sa[i]] != s[sa[i-1]]);
        b2h[i] = false;
    }
    for(int H = 1;H<N;H <<= 1){
        int buckets = 0;
        for(int i = 0,j;i<N;i = j){
            j = i+1;
            while(j<N && !bh[j]) ++j;
            nxt[i] = j;
            ++buckets;
        }
        if(buckets==N) break;
        for(int i = 0;i<N;i = nxt[i]){
            cont[i] = 0;
            for(int j = i;j<nxt[i];++j)
                rk[sa[j]] = i;
        }
        ++cont[rk[N-H]];
        b2h[rk[N-H]] = true;
        for(int i = 0;i<N;i = nxt[i]){
            for(int j = i;j<nxt[i];++j){
                int s = sa[j]-H;
                if(s>=0){
                    int head = rk[s];
                    rk[s] = head+cont[head];
                    ++cont[head];
                    b2h[rk[s]] = true;
                }
            }
            for(int j = i;j<nxt[i];++j){
                int s = sa[j]-H;
                if(s>=0 && b2h[rk[s]]){
                    for(int k = rk[s]+1;!bh[k] && b2h[k];++k)
                        b2h[k] = false;}}}
```

```
        for(int i = 0;i<N;++i){
            sa[rk[i]] = i;
            bh[i] |= b2h[i];
        }
    }
    for (int i = 0; i < N;i++)idx[sa[i]] = i;
}
```

TREAP IMP

```
const int N = 100010;
unsigned next() {
        static unsigned x(987654321), y(123456789), z(521288629), w(136751267);
        unsigned t = x ^ x << 11;
        x = y, y = z, z = w;
        return w = ((t ^ t >> 8) ^ w) ^ w >> 19;
}
struct treap{
        treap  *l,*r;
        int sz;
        bool reverse;
        char val;
        unsigned h1 = 0,h2 = 0;
        unsigned y;
        treap(char p){
                reverse = false;
                l = r =  NULL;
                y = next();
                sz = 1, val = p;
        }
        void recalc(){
                sz = 1;
                if(l)sz += l->sz;
                if(r)sz += r->sz;
        }
        /*treap* root;
        treap(): root(EMPTY){}
        ~treap(){delete root;}
        * */
};
typedef treap * Node;
int cnt(Node t){
        if(t)return t->sz;
        return 0;
}
void push(Node t){
        if(t){
                if(t->reverse){
                        swap(t->l,t->r);
                        t->reverse = 0;
                        if(t->l)t->l->reverse = !t->l->reverse;
                        if(t->r)t->r->reverse = !t->r->reverse;
                }}}
```

```
void split(Node t,Node &l,Node &r,int k){
	push(t);
	if(!t){
		l = r = NULL;
		return;
	}
	int pos = cnt(t->l) + 1;
	if(pos <= k){
		split(t->r,t->r,r,k - pos);
		l = t;
	}
	else{
		split(t->l,l,t->l,k);
		r = t;
	}
   t->recalc();
}

Node merge(Node l,Node r){
	push(l);push(r);
	if(!l)return r;
	if(!r)return l;
	if(l->y > r->y){
		l->r = merge(l->r,r);
		l->recalc();
		return l;
	}
	else{
		r->l = merge(l,r->l);
		r->recalc();
		return r;
	}
}

void print(Node a){
	Node cpy = a;
	Node my = NULL;
	while(cnt(cpy)){
		Node l, r, trash;
		split(cpy,l,r,0);//in the treap r we can find pos
		split(r,trash,r,1);
		//cout << (trash?1:0);
		cout << trash->val;
		cpy = merge(l,r);
		my = merge(my,trash);
	}
	a = my;
	cout << endl;
}
void insert(Node &root,int pos,char car){
	Node l,r;
	split(root,l,r,pos);//in the treap R we can find pos
```

```
        root = merge(merge(l,new treap(car)),r);
}
void reverse(Node &root, int le, int ri) {
        Node l, m, r;
        split(root, m, r, ri + 1);
        split(m, l, m, le);
        //print(m);
        if (m)m->reverse = !m->reverse;
        root = merge(l, merge(m, r));
}
char query(Node &root,int pos){
        Node L, M, R;
        split(root, L, R, pos);
//      print(L);
//      print(R);
//      cout << "MY POS " << pos << endl;
        split(R, M, R, 1);
//      print(M);
//      print(R);
        char c;
        if (M)c = M->val;
        else assert(false);
        root = merge(L, merge(M, R));
        //print(root);
        return c;
}
```

```
#define L(n)((n)<<(1))
#define R(n)(L(n)+(1))

struct event {
  int x, ya, yb, t;
  event( int _x, int _ya, int _yb, int _t ) {
    x = _x; ya = _ya; yb = _yb; t = _t;
  }
  friend bool operator<( const event &A, const event &B ) {
    return ( A.x == B.x ) ? ( A.t > B.t ) : ( A.x < B.x );
  }
};
struct node {
  int cnt, lzy;
  node() : cnt( 0 ), lzy( 0 ) {}
};

vector< event > E;
vector< node > Tree;
int lo, hi, tlen;

void Construct( int n ) {
  for( tlen = 1; tlen < n; tlen <<= 1 );
  Tree.resize( tlen << 1 );
}
void update( int x, int y, int v, int n ) {
  if( x >= hi || y <= lo ) return;
  if( x >= lo && y <= hi ) {
    Tree[ n ].lzy += v;
    if( Tree[ n ].lzy ) Tree[ n ].cnt = ( y - x );
    if( Tree[ n ].lzy < 1 ) {
      if( n >= tlen ) Tree[ n ].cnt = 0;
      else Tree[ n ].cnt = Tree[ L( n ) ].cnt + Tree[ R( n ) ].cnt;
    }
    return;
  }

  update( x, ( x + y )>>1, v, L( n ) );
  update( ( x + y )>>1, y, v, R( n ) );

  if( Tree[ n ].lzy < 1 )
    Tree[ n ].cnt = Tree[ L( n ) ].cnt + Tree[ R( n ) ].cnt;
}

void Update( int x, int y, int v ) {
  lo = x; hi = y;
  update( 0, tlen, v, 1 );
}
int query( int x, int y, int n ) {
  if( x >= hi || y <= lo ) return 0;
  if( x >= lo && y <= hi ) return Tree[ n ].cnt;
```

```
  if( Tree[ n ].lzy ) return min( y, hi ) - max( x, lo );

  return query( x, ( x + y )>>1, L( n ) ) +
         query( ( x + y )>>1, y, R( n ) );
}
int Query() {
  lo = 0; hi = 30005;
  return query( 0, tlen, 1 );
}

int main( void )
{
  int N; scanf( "%d", &N );
  for( int i = 0; i < N; ++i ) {
    int xa, ya, xb, yb;
    scanf( "%d%d%d%d", &xa, &ya, &xb, &yb );
    E.push_back( event( xa, ya, yb, +1 ) );
    E.push_back( event( xb, ya, yb, -1 ) );
  }

  Construct( 30005 );

  sort( E.begin(), E.end() );
  int last = 0, sol = 0;

  for( int i = 0; i < E.size(); ++i ) {
    sol += Query()*( E[i].x - last );
    Update( E[i].ya, E[i].yb, E[i].t );
    last = E[i].x;
  }

  printf( "%d\n", sol );

  return 0;
}
```