数据结构

- 单调队列（求区间最大值）
- 单调栈（求之后第一个大于自己的元素下标）
  - 求子矩阵个数
- 树状数组
- 线段树
  - 扫描线
- 我也不知道叫什么的东西
  - O(n)内排除一个二元组数据类型数组中，严格小于数组内另一元素的所有较小元素
- ST表

数据结构

单调队列（求区间最大值）

```
#include<iostream>
#include<cstdio>
using namespace std;
const int maxn=2e6+10;
int a[maxn],q[maxn];  //q队列记录a中对应数据的下标
int main()
{
  int n,k;
  cin>>n>>k;
  for(int i=1;i<=n;i++)
  {
    scanf("%d",&a[i]);
  }
  int l=1,r=0;
  for(int i=1;i<=n;i++) //注意扫的方向
  {
    while(l<=r&&q[l]<=i-k)  l++;    //维护区间长度
    while(l<=r&&a[i]>a[q[r]]) r--;    //维护单调队列的单调性
    q[++r]=i;   //记录下标
    if(i>=k)
      cout<<a[q[l]]<<'\n';
  }
}
```

> 逆序

```
for(int i=n;i>=1;--i) //注意扫的方向
  {
    while(l<=r&&q[l]>=i+m)  l++;    //维护区间长度
    while(l<=r&&a[i]>=a[q[r]]) r--;   //维护单调队列的单调性
    q[++r]=i;   //记录下标
    if(i<=n-m+1)
      cout<<a[q[l]]<<'\n';
  }
```

单调栈（求之后第一个大于自己的元素下标）

```
#include<iostream>
#include<algorithm>
#include<stack>
#include<cstdio>
using namespace std;
#define ll long long
const int maxn=3e6+10;
int a[maxn];
int ans[maxn];
stack<int> s;
int main()
{
    int n;
    cin>>n;
    for(int i=1;i<=n;i++)
        scanf("%d",&a[i]);
    //s.push(a[0]);
    for(int i=n;i>=1;i--) //注意扫的方向
    {
        while(!s.empty()&&a[s.top()]<=a[i])
            s.pop();
        ans[i]=s.empty()?0:s.top();
        s.push(i);
    }
    for(int i=1;i<=n;i++)
    {
        printf("%d ",ans[i]);
    }
    return 0;
}
```

> 维护左边第一个小于自己的数的下标

```
  for(int j=1;j<=m;j++){
    while(top!=0 && h[stk[top]]>=h[j]) top--;
    l[j]=top==0?0:stk[top];
    stk[++top]=j;
  }
```

> 维护左边第一个小于等于自己的数的下标

```
for(int j=1;j<=m;j++){
  while(top!=0 && h[stk[top]]>h[j]) top--;
  l[j]=top==0?0:stk[top];
  stk[++top]=j;
}
```

求子矩阵个数

P1950 长方形

大意：子矩阵N*M，求有多少个长方形子矩阵不包含 '*'

单调栈复杂度 O(m) ,因为有n行，所以复杂度为 O(nm)

1. 定义 hi为当前行第i 列可向上延伸多少（即有多少为图画的块，如果当前块被图画那么值为00）
2. 使用单调栈算出li和ri，分别是h中左边第一个（从hi开始）不大于hi的数和右边第一个（从hi开始）小于hi的数
3. 对每一列求出被这一列的高度限制的长方形数，即为(i-li)×(ri-i)×hi，将所有列的答案相加就是以当前行为底边构造的长方形的方案数了

```
#include<iostream>
#include<cstdio>
#include<string>
#include<vector>
#include<cmath>
#include<stack>
#include<queue>
#include<map>
#include<unordered_map>
#include<set>
#include<cstring>
#include<algorithm>
#include<climits>
#include<numeric>
#define int long long
#define pii pair<int,int>
#define forn(i,t) for(int i=1;i<=t;i++)
#define IOFast() ios::sync_with_stdio(0),cin.tie(0),cout.tie(0)
using namespace std;
const int maxn=1e3+10;
int h[maxn],l[maxn],r[maxn];
char mp[maxn][maxn];
int n,m,ans;
int stk[maxn];
int top=0;
void init(){
  cin>>n>>m;
  for(int i=1;i<=n;i++)
    for(int j=1;j<=m;j++)
```

```
      cin>>mp[i][j];
}
void calc(){
  top=0;
  for(int j=1;j<=m;j++){
    while(top!=0 && h[stk[top]]>h[j]) top--;
    l[j]=top==0?0:stk[top];
    stk[++top]=j;
  }

  top=0;
  for(int j=m;j>=1;j--){
    while(top!=0 && h[stk[top]]>=h[j]) top--;
    r[j]=top==0?m+1:stk[top];
    stk[++top]=j;
  }

  for(int j=1;j<=m;j++)
    ans+=h[j]*(j-l[j])*(r[j]-j);
}
void solve(){
  for(int i=1;i<=n;i++)
  {
    for(int j=1;j<=m;j++)
    {
      h[j]++;
      if(mp[i][j]=='*')
        h[j]=0;
    }
    calc();
  }
  cout<<ans<<endl;
}
signed main(){
  IOFast();
  init();
  solve();
}
```

树状数组

```
#include<iostream>
using namespace std;
const int maxn=5e5+10;
int a[maxn];
int n,m;
```

```
int lowbit(int x)
{
  return x&(-x);
}
void add(int x,int k)
{
  while(x<=n)
  {
    a[x]+=k;
    x+=lowbit(x);
  }
}
int search(int x)
{
  int ans=0;
  while(x!=0)
  {
    ans+=a[x];
    x-=lowbit(x);
  }
  return ans;
}
int main()
{
  cin>>n>>m;
  for(int i=1;i<=n;i++)
  {
    int num;
    cin>>num;
    add(i,num);
  }
  for(int i=0;i<m;i++)
  {
    int tri,num1,num2;
    cin>>tri>>num1>>num2;
    if(tri==1)
    {
      add(num1,num2);
    }
    else {
      cout<<search(num2)-search(num1-1)<<endl;
    }
  }
}
```

线段树

```
#include<iostream>
```

```cpp
#include<cstdio>
#include<string>
#include<vector>
#include<cmath>
#include<cstring>
#include<algorithm>
#include<iomanip>
using namespace std;
using ll=long long;
#define int ll
#define ls u<<1
#define rs u<<1|1
#define forn(i,j) for(int i=1;i<=j;i++)
#define IOFast() ios::sync_with_stdio(0),cin.tie(0),cout.tie(0)
int n,m;
const int maxn=1e5+10;
int a[maxn];
int mod;
struct node{
  int l,r;
  int sum,add,mul;
}tr[maxn<<2];

void pushup(int u)
{
  tr[u].sum=(tr[ls].sum+tr[rs].sum)%mod;
}
void eval(node & root ,int add ,int mul)
{
  root.sum=(root.sum*mul+add*(root.r-root.l+1))%mod;
  root.mul=(root.mul*mul)%mod;
  root.add=(root.add*mul+add)%mod;
}
void pushdown(int u)
{
  eval(tr[ls],tr[u].add,tr[u].mul);
  eval(tr[rs],tr[u].add,tr[u].mul);
  tr[u].add=0,tr[u].mul=1;
}
void build(int u,int l,int r)
{
  if(l==r) tr[u]={l,r,a[l],0,1};
  else{
    tr[u]={l,r,0,0,1};
    int mid=l+r>>1;
    build(ls,l,mid);
    build(rs,mid+1,r);
    pushup(u);
  }
```

```
}
void modify(int u,int l,int r,int add,int mul)
{
  if(tr[u].l>=l&&tr[u].r<=r)  eval(tr[u],add,mul);
  else{
    pushdown(u);
    int mid=tr[u].l+tr[u].r>>1;
    if(l<=mid) modify(ls,l,r,add,mul);
    if(r>mid) modify(rs,l,r,add,mul);
    pushup(u);
  }
}
int query(int u,int l,int r)
{

  if(tr[u].l>=l&&tr[u].r<=r) return tr[u].sum;
  pushdown(u);
  int mid=tr[u].l+tr[u].r>>1;
  int sum=0;
  if(l<=mid) sum=query(ls,l,r);
  if(r>mid) sum=(sum+query(rs,l,r))%mod;
  return sum;
}
signed main()
{
  IOFast();
  cin>>n>>m>>mod;
  for(int i=1;i<=n;i++) cin>>a[i];
  build(1,1,n);
  forn(_,m)
  {
    int op;
    cin>>op;
    if(op==1)
    {
      int l,r,k;
      cin>>l>>r>>k;
      modify(1,l,r,0,k);
      continue;
    }
    if(op==2)
    {
      int l,r,k;
      cin>>l>>r>>k;
      modify(1,l,r,k,1);
      continue;
    }
    int l,r;
    cin>>l>>r;
```

```
    cout<<query(1,l,r)<<'\n';
  }
}
```

扫描线

```
#include<iostream>
#include<cstdio>
#include<string>
#include<vector>
#include<cmath>
#include<cstring>
#include<algorithm>
#include<iomanip>
using namespace std;
//using ll=long long;
#define int long long
#define ls u<<1
#define rs u<<1|1
#define forn(i,j) for(int i=1;i<=j;i++)
#define IOFast() ios::sync_with_stdio(0),cin.tie(0),cout.tie(0)
const int maxn=1e3+10;
struct segment{
  double x,y1,y2;
  int k;
  segment(){x=0,y1=0,y2=0,k=0;}
  segment(double x_,double y1_,double y2_,int k_):x(x_),y1(y1_),y2(y2_),k(k_){}
  bool operator < (const segment &t) const{
    return x<t.x;
  }
}seg[maxn<<1];
struct node{
  int l,r;
  int cnt;
  double len;
  node(){cnt=0,len=0,l=0,r=0;}
  node(int l_,int r_,int cnt_,double len_):l(l_),r(r_),cnt(cnt_),len(len_){}
}tr[maxn<<3];
int n;
vector<double> ys;
int find(double y)
{
  return lower_bound(ys.begin(),ys.end(),y)-ys.begin();
}
void pushup(int u)
{
  if(tr[u].cnt) tr[u].len=(ys[tr[u].r+1]-ys[tr[u].l]);
  else if(tr[u].l!=tr[u].r)
```

```
  {
    tr[u].len=tr[ls].len+tr[rs].len;
  }
  else tr[u].len=0;
}
void modify(int u,int l,int r,int k)
{
  if(tr[u].l>=l&&tr[u].r<=r)
  {
    tr[u].cnt+=k;
    pushup(u);
  }
  else {
    int mid=tr[u].l+tr[u].r>>1;
    if(l<=mid) modify(ls,l,r,k);
    if(r>mid) modify(rs,l,r,k);
    pushup(u);
  }
}
void build(int u,int l,int r)
{
  tr[u]=node(l,r,0,0);
  if(l!=r)
  {
    int mid=l+r>>1;
    build(ls,l,mid);
    build(rs,mid+1,r);
  }
}

int T=1;
void solve(){
  ys.clear();
  for(int i=0,j=0;i<n;i++)
  {
    double x1,y1,x2,y2;
    scanf("%lf%lf%lf%lf",&x1,&y1,&x2,&y2);
    seg[j++]=segment(x1,y1,y2,1);
    seg[j++]=segment(x2,y1,y2,-1);
    ys.push_back(y1),ys.push_back(y2);
  }
  sort(ys.begin(),ys.end());
  ys.erase(unique(ys.begin(),ys.end()),ys.end());
  build(1,0,ys.size()-2);
  sort(seg,seg+2*n);
  double ans=0;
  for(int i=0;i<2*n;i++)
  {
    if(i>0) ans+=tr[1].len*(seg[i].x-seg[i-1].x);
```

```
    modify(1,find(seg[i].y1),find(seg[i].y2)-1,seg[i].k);
  }
  printf("Test cast #%lld\n",T++);
  printf("Total explored area: %.2lf\n\n",ans);
}
signed main(){
  IOFast();
  while(scanf("%lld",&n)){
    if(n==0) break;
    solve();
  }
}
```

我也不知道叫什么的东西

O(n)内排除一个二元组数据类型数组中，严格小于数组内另一元素的所有较小元素

```
int m;
pii pool[1000005];
void ins(const pii & t){
    while(m&&pool[m].second<=t.second) m--;
    if(!m||pool[m].first<t.first) pool[++m]=t;
}
void merge(const vector<pii> & a,const vector<pii> &b,vector<pii> & c)
{
    int lena=a.size(),lenb=b.size();
    m=0;
    int i=0,j=0;
    while(i<lena && j<lenb) ins(a[i].first<b[j].first?a[i++]:b[j++]);
    while(i<lena) ins(a[i++]);
    while(j<lenb) ins(b[j++]);
    c.resize(m);
    forn(k,m) c[k-1]=pool[k];
}
```

假设a与b中所有的元素已经按照第一维数据升序排列，那么merge()中的三个while能够保证：后塞入元素的第一维数据一定比前塞入元素的第一维数据大。

ins()中的while能够保证在出了while之后，要么栈pool空了，要么栈顶元素的第二维数据比当前正要塞入的元素t的第二维数据大。

那么若栈内还有元素，则若栈顶元素的第一维数据大于等于正要塞入的数据，那么正要塞入的数据一定严格小于栈顶元素，则不需要入栈，反之则需要塞入。

最终能够保证c中的元素一定又是按照第一维数据升序排列，且已经筛去较小元素。

ST表

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<fstream>
#include<vector>
#include<bitset>
#include<string>
#include<cmath>
#include<map>
#define ll long long
using namespace std;
const int maxn=2e5+10;
ll a[maxn];
inline ll read()
{
    ll x=0,f=1;char ch=getchar();
    while (!isdigit(ch)){if (ch=='-') f=-1;ch=getchar();}
    while (isdigit(ch)){x=x*10+ch-48;ch=getchar();}
    return x*f;
}
ll Max[maxn][21];
int query(int l,int r)
{
    int k=log2(r-l+1);
    return max(Max[l][k],Max[r-(1<<k)+1][k]);
}
int main()
{
    int n,m;
    cin>>n>>m;
    for(int i=1;i<=n;i++)
    {
        Max[i][0]=read();
    }
    for(int j=1;j<21;j++)
    {
        for(int i=1;i+(1<<j)-1<=n;i++)
        {
            Max[i][j]=max(Max[i][j-1],Max[i+(1<<(j-1))][j-1]);
        }
    }
    for(int i=1;i<=m;i++)
    {
        int l=read(),r=read();
        printf("%d\n",query(l, r));
```

```
    }
    return 0;
}
```