PROB D GARDEN REPORT

Write by Gestalt Lur
2012-06-27

题目大意

給出 n 個數字和 m 個詢問，對于每個詢問(p,x,y),p=0 代表將 x 位置上的數字賦值為 y;p=1 表示交換 x 和 y 位置上的值；p=2,表示輸出[x,y]之間的連續 k 個數字和中的最大值。

首先输入 T 代表數據的組數，对于每組數據，第一行給出 n,m,k(1<=k<=n<=200000,0<=m<=200000)三个整数，第二行有 n 个[-100,100]的整數，接下來 m 行每行三個整數表示(p,x,y)，對于每個 p=2 的詢問輸出[x,y]之間的連續 k 個數字和中的最大值，(保証 y-x+1>=k,1<=x<=y<=n)

算法分析

因為是區間問題，可以想到線段樹標記的方法來做。首先考慮到詢問的對象是連續的 k 個數字和的最大值，所以線段樹節點中存儲的對象應當是這個最大值，否則的話就要每次再去計算區間裏的這個值而導致 TLE。那麼下面的問題就是如何維護這個存儲連續 k 個數字和的最大值的線段樹。

首先要建立這樣一個線段樹。因為輸入數據保証 y-x+1>=k,那麼其葉子節點必然是一段連續 k 個數字的和，因此線段樹的範圍就是[1,n-k+1]。因此在初始建立線段樹的時候可以用 O(n)的時間復雜度計算出每一個連續 k 個數字的和并將其加入線段樹相應的位置上。這樣建立完成之後，對于線段樹上的區間[x,y],實際對應的區間是[x,y+k-1]，反之亦然。

所以對于詢問[x,y]的操作,只要查詢[x,y-k+1]即可，而對于更改節點值的操作，另外記錄一個數組 a[n]，每次更改的時候先求出更改位置上與更改後值的改變量，給線段樹相應的位置增加這個改變量，在訪問的時候如果出現做標記的節點就加上這個改變量，這是一個線段樹常用的優化方法，故不贅述。

参考代碼

pascal:

```
{
ZJNU D GARDEN
gestapolur
2012-06-12
2012-06-15 ACCEPTED
}
program ZJNU_INVITED_2012_D_GARDEN;
const
  MAXT = 8000006;
  MAXN = 300005;
  INF  = 2141483647;
var
```

```
  n , m , wide : longint;
  a            : array[ 1..MAXN ] of longint;
  maxv , del   : array[ 1..MAXT ] of longint;
  sign         : array[ 1..MAXT ] of boolean;
  tt           : integer;

function min(a , b : longint ) : longint;
begin
  if a < b then
    exit( a );
  exit( b );
end; { min }

function max(a , b : longint ) : longint;
begin
  if a > b then
    exit( a );
  exit( b );
end; { max }

procedure update_child(pos : longint );
begin
  if sign[ pos ] then begin
    sign[ pos ] := FALSE;

    sign[ pos shl 1 ] := TRUE;
    sign[ pos shl 1 or 1 ] := TRUE;
    { apply marked change and update it's children's mark & value }
    inc( del[ pos shl 1 ] , del[ pos ] );
    inc( del[ pos shl 1 or 1 ] , del[ pos ] );


    inc( maxv[ pos shl 1 ] , del[ pos ]);
    inc( maxv[ pos shl 1 or 1 ] , del[ pos ] );

    del[ pos ] := 0;
  end;
end; { update_child }

procedure init_update( pos , l , r , pt , val : longint );
var
  mid : longint;
begin
  if ( l = pt ) and ( pt = r ) then begin
    maxv[ pos ] := val;
    del[ pos ] := val;
    exit;
  end;

  update_child( pos );

  mid := ( l + r ) shr 1;
  if pt <= mid then
    init_update( pos shl 1 , l , mid , pt , val )
  else
    init_update( pos shl 1 or 1 , mid + 1 , r , pt , val );

  if l <> r then
    maxv[ pos ] := max ( maxv[ pos shl 1 ] , maxv[ pos shl 1 or 1 ] );
end;

procedure update( pos , ql , qr , l , r , val : longint );
var
  mid : longint;
begin
  if ( ql <= l ) and ( r <= qr ) then begin
    inc( del[ pos ] , val );
    sign[ pos ] := TRUE;
    inc( maxv[ pos ] , val );
    exit;
  end;

  update_child( pos );

  mid := ( l + r ) shr 1;
  if ql <= mid then
    update( pos shl 1 , ql , qr , l , mid , val );
  if qr > mid then
    update( pos shl 1 or 1 , ql , qr , mid + 1 , r , val );

  if l <> r then
    maxv[ pos ] := max( maxv[ pos shl 1 ] , maxv[ pos shl 1 or 1 ] );

end; { update }

function query(pos , ql , qr , l , r : longint ) : longint;
var
  mid , rmax : longint;
```

```
begin
  update_child( pos );

  if ( ql <= l ) and ( r <= qr ) then
    exit( maxv[ pos ] );

  mid := ( l + r ) shr 1;
  rmax := -INF;

  if ql <= mid then
    rmax := query( pos shl 1 , ql , qr , l , mid );
  if qr > mid then
    rmax := max ( rmax , query( pos shl 1 or 1 , ql , qr , mid + 1 , r ) );
  exit( rmax );
end; { query }

procedure build(l , r , pos : longint  );
var
  mid : longint;
begin
  sign[ pos ] := false;
  maxv[ pos ] := 0;
  del[ pos ] := 0;
  if l <> r then begin
    mid := ( l + r ) shr 1;
    build( l , mid , pos shl 1 );
    build( mid + 1 , r , pos shl 1 or 1 );
  end;
end;

procedure init();
var
  tmp , i : longint;
begin
  readln( n , m , wide );

  build( 1 , n - wide + 1 , 1 );


  for i := 1 to n do
    read( a[ i ] );
  readln;
  { first node }
  tmp := 0;
  for i := 1 to wide do
    inc( tmp , a[ i ] );

  init_update( 1 , 1 , n - wide + 1 , 1 , tmp );
  { node 2 to n - wide + 1 }
  for i := wide + 1 to n do begin
    tmp := tmp - a[ i - wide ] + a[ i ];
    init_update( 1 , 1 , n - wide + 1 , i - wide + 1 , tmp );
  end;
end; { init }

procedure work();
var
  i , ins , sx , sy , tmp1 , tmp2 : longint;
begin
  for i := 1 to m do begin
    readln( ins , sx , sy );
    if ins = 0 then begin
      tmp1 := sy - a[ sx ];
      a[ sx ] := sy;
      update( 1 , max( sx - wide + 1 , 1 ) , sx , 1 , n - wide + 1 , tmp1 );
    end
    else if ins = 1 then begin

      tmp1 := a[ sy ] - a[ sx ];
      tmp2 := a[ sx ] - a[ sy ];
      {change a node i in original may refer [ i - K + 1 , i ] in interval tree.}
      update( 1 , max( sx - wide + 1 , 1 ) , sx , 1 , n - wide + 1 , tmp1 );
      update( 1 , max( sy - wide + 1 , 1 ) , sy , 1 , n - wide + 1 , tmp2 );

      tmp1 := a[ sx ];
      a[ sx ] := a[ sy ];
      a[ sy ] := tmp1;

    end
    else begin
      writeln( query( 1 , sx , max( sy - wide + 1 , 1 ) , 1 , n - wide + 1 ) );
    end;
  end;
end;

begin
```

```
  readln( tt );
  while tt <> 0 do begin
    init;
    work;
    dec( tt );
  end;
end.
```

JAVA:

```
/*
  ZJNU INVITED 2012 D GARDEN
  2012-06-15 ACCEPTED
  write by gestapolur
*/
import java.io.*;
import java.util.*;

public class zjnu_d {
   public static final int MAXN = 200005;
   public static final int MAXT = 3000002;
   public static final int INF = 2141483647;
   public static int cnt , n , m , wide;
   public static int []a = new int[ MAXN ];
   public static boolean []sign = new boolean [ MAXT ];
   public static int []maxv = new int [ MAXT ];
   public static int []del = new int [ MAXT ];

   public static Scanner in = new Scanner( System.in );

   public static int max( int a , int b ){
      return a > b ? a : b;
   }

   public static int min( int a , int b ){
      return a < b ? a : b;
   }

   public static void build( int pos , int l , int r ){
      sign[ pos ] = false;
      del[ pos ] = 0;
      maxv[ pos ] = 0;
      if( l != r )
         {
```

```
         int mid = l + r >> 1;
         build( pos << 1 , l , mid );
         build( pos << 1 | 1 , mid + 1 , r );
      }
}

public static void update_child( int pos ){
   if( sign[ pos ] )
      {
         sign[ pos ] = false;

         sign[ pos << 1 ] = true;
         sign[ pos << 1 | 1 ] = true;

         del[ pos << 1 ] += del[ pos ];
         del[ pos << 1 | 1 ] += del[ pos ];

         maxv[ pos << 1 ] += del[ pos ];
         maxv[ pos << 1 | 1 ] += del[ pos ];

         del[ pos ] = 0;
      }
}

public static int query( int pos , int l , int r , int L , int R ){

   if( L <= l && r <= R )
      return maxv[ pos ];

   update_child( pos );


   int mid = l + r >> 1;
   int res = -INF;
   if( L <= mid )
      res = max ( res , query( pos << 1 , l , mid , L , R ) );
   if( mid < R )
      res = max ( res , query( pos << 1 | 1 , mid + 1 , r , L , R ) );
   return res;
}

public static void update( int pos , int l , int r , int L , int R , int val ){
    if( L <= l && r <= R )
      {
        sign[ pos ] = true;
        maxv[ pos ] += val;
        del[ pos ] += val;
        return ;
      }

    update_child( pos );

    int mid = l + r >> 1;
    if( L <= mid )
      update( pos << 1 , l , mid , L , R , val );
    if( mid < R )
      update( pos << 1 | 1 , mid + 1 , r , L , R , val );

    maxv[ pos ] = max( maxv[ pos << 1 ] , maxv[ pos << 1 | 1 ] );
  }

  public static void work(){
    int i;
    int ins , sx , sy , tmp1 , tmp2;
    for( i = 1 ; i <= m ; ++ i )
      {
        ins = in.nextInt();
        sx = in.nextInt();
        sy = in.nextInt();
        if( ins == 0 )
          {
            tmp1 = sy - a[ sx ];
            a[ sx ] = sy;
            update( 1 , 1 , n - wide + 1 , max( sx - wide + 1 , 1 ) , sx , tmp1 );
          }
        else if( ins == 1 )
          {
            tmp1 = a[ sy ] - a[ sx ];
            tmp2 = a[ sx ] - a[ sy ];

            update( 1 , 1 , n - wide + 1 , max( sx - wide + 1 , 1 ) , sx , tmp1 );
            update( 1 , 1 , n - wide + 1 , max( sy - wide + 1 , 1 ) , sy , tmp2 );
            tmp1 = a[ sx ];
            a[ sx ] = a[ sy ];
            a[ sy ] = tmp1;
          }
        else
          System.out.printf( "%d\n" , query( 1 , 1 , n - wide + 1 , sx , sy - wide + 1 ));

    }
}

public static void init(){
  int i , tmp;
  n = in.nextInt();
  m = in.nextInt();
  wide = in.nextInt();

  for( i = 1 ; i <= n ; ++ i )
    a[ i ] = in.nextInt();

  tmp = 0;
  for( i = 1 ; i <= wide ; ++ i )
    tmp += a[ i ];
  update( 1 , 1 , n - wide + 1 , 1 , 1 , tmp );
  for( i = wide + 1 ; i <= n ; ++ i )
    {
      tmp = tmp - a[ i - wide ] + a[ i ];
      update( 1 , 1 , n - wide + 1 , i - wide + 1 , i - wide + 1 , tmp );
    }
}

public static void main( String[] args ) throws Exception {

  cnt = in.nextInt();
  while( cnt > 0 )
    {
      build( 1 , 1 , n - wide + 1 );
      init();
      work();
      -- cnt;
    }
}
```

```
}
```