**Sequence operation**

**Time Limit: 10000/5000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 2131    Accepted Submission(s): 608**

Problem Description

lxhgww got a sequence contains n characters which are all '0's or '1's.  
We have five operations here:  
Change operations:  
0 a b change all characters into '0's in [a , b]  
1 a b change all characters into '1's in [a , b]  
2 a b change all '0's into '1's and change all '1's into '0's in [a, b]  
Output operations:  
3 a b output the number of '1's in [a, b]  
4 a b output the length of the longest continuous '1' string in [a , b]

Input

T(T<=10) in the first line is the case number.  
Each case has two integers in the first line: n and m (1 <= n , m <= 100000).  
The next line contains n characters, '0' or '1' separated by spaces.  
Then m lines are the operations:  
op a b: 0 <= op <= 4 , 0 <= a <= b < n.

Output

For each output operation , output the result.

Sample Input

1

10 10

0 0 0 1 1 0 1 0 1 1

1 0 2

3 0 5

2 2 2

4 0 4

0 3 6

2 3 7

4 2 8

1 0 5

0 5 6

3 3 9

Sample Output

5

2

6

5

Author

lxhgww&&NotOnlySuccess

/\*

\* File: main.cpp

\* Author: etaf

\* 题意：给定一个0,1序列，有0,1,2,3,4,种操作：

\* 0：[a,b] 全变0

\* 1：[a,b] 全变1

\* 2：[a,b] 取反

\* 3: [a,b] 求和

\* 4：[a,b] 求最长连续1个数。

\*/

#include<cstdio>

#include <cstdlib>

#include<cstring>

#include<cmath>

#include<iostream>

#include<algorithm>

#include<vector>

#include<queue>

#include<stack>

#include<map>

#include<set>

#include<bitset>

using namespace std;

const int N= 100010;

#define L(i) (i<<1)

#define R(i) (i<<1|1)

inline int Mid(int x,int y)

{return (x+y)>>1;}

inline int Max(int x,int y)

{return x>y?x:y;}

inline void Swap(int&x,int& y)

{x^=y;y^=x;x^=y;}

typedef struct seg

{

int l,r,lv0,lv1,rv0,rv1,max1,max0,sum;

bool lazy[3];

}seg;

seg a[3\*N];

int data[N];

void Upbyson(seg& parent,seg& lc,seg& rc)

{

parent.sum=lc.sum+rc.sum;

parent.lv0=lc.lv0+(lc.max0==lc.r-lc.l+1?rc.lv0:0);

parent.lv1=lc.lv1+(lc.max1==lc.r-lc.l+1?rc.lv1:0);

parent.rv0=rc.rv0+(rc.max0==rc.r-rc.l+1?lc.rv0:0);

parent.rv1=rc.rv1+(rc.max1==rc.r-rc.l+1?lc.rv1:0);

parent.max0=Max(Max(lc.max0,rc.max0),lc.rv0+rc.lv0);

parent.max1=Max(Max(lc.max1,rc.max1),lc.rv1+rc.lv1);

}

inline void process(seg &parent,int type)

{

if(type==0)

{

parent.sum=0;

parent.max0=parent.lv0=parent.rv0=parent.r-parent.l+1;

parent.max1=parent.lv1=parent.rv1=0;

}

if(type==1)

{

parent.sum=parent.max1=parent.lv1=parent.rv1=parent.r-parent.l+1;

parent.max0=parent.lv0=parent.rv0=0;

}

if(type==2)

{

Swap(parent.lv0,parent.lv1);

Swap(parent.rv0,parent.rv1);

Swap(parent.max0,parent.max1);

parent.sum=parent.r-parent.l+1-parent.sum;

}

}

void pushdown(int i)

{

if (a[i].l == a[i].r) {

a[i].lazy[0] = a[i].lazy[1] = 0;

a[i].lazy[2] ^= 1;

return;

}

if (a[i].lazy[0]) {

a[L(i)].lazy[0] = a[R(i)].lazy[0] = 1;

a[L(i)].lazy[1]=a[L(i)].lazy[2]=a[R(i)].lazy[1]=a[R(i)].lazy[2]=0;

process(a[L(i)], 0);

process(a[R(i)], 0);

a[i].lazy[0] = 0;

}

if (a[i].lazy[1]) {

a[L(i)].lazy[1] = a[R(i)].lazy[1] = 1;

a[L(i)].lazy[0]=a[L(i)].lazy[2]=a[R(i)].lazy[0]=a[R(i)].lazy[2]=0;

process(a[L(i)], 1);

process(a[R(i)], 1);

a[i].lazy[1] = 0;

}

if(a[i].lazy[2])

{

a[i].lazy[2]=0;

a[L(i)].lazy[2]^=1;

a[R(i)].lazy[2]^=1;

process(a[L(i)],2);

process(a[R(i)],2);

}

}

void build(int i,int left,int right )

{

a[i].l=left;

a[i].r=right;

a[i].lazy[0]=a[i].lazy[1]=a[i].lazy[2]=0;

if(left==right)

{

a[i].sum=a[i].max1=a[i].lv1=a[i].rv1=data[left];

a[i].max0=a[i].lv0=a[i].rv0=1-data[left];

return;

}

int mid=Mid(left,right);

build(L(i),left,mid);

build(R(i),mid+1,right);

Upbyson(a[i],a[L(i)],a[R(i)]);

}

void Update(int i,int left,int right,int type)

{

/\* if(type==0 && a[i].lazy[0])

return;

if(type==1 && a[i].lazy[1])

return;

\*/

pushdown(i);

if(left<=a[i].l && a[i].r<=right)

{

a[i].lazy[type]=1;

if(type==0)

{a[i].lazy[1]=a[i].lazy[2]=0;}

if(type==1)

{a[i].lazy[0]=a[i].lazy[2]=0;}

process(a[i],type);

return;

}

int mid=Mid(a[i].l,a[i].r);

if(left<=mid)

Update(L(i),left,right,type);

if(right>mid)

Update(R(i),left,right,type);

Upbyson(a[i],a[L(i)],a[R(i)]);

}

seg Query(int i,int left,int right,int type)

{

pushdown(i);

if(left==a[i].l && right==a[i].r)

{

return a[i];

}

int mid=Mid(a[i].l,a[i].r);

if(right<=mid)

return Query(L(i),left,right,type);

else if(left>mid)

return Query(R(i),left,right,type);

else

{

seg t0,t1,t2;

t1=Query(L(i),left,mid,type);

t2=Query(R(i),mid+1,right,type);

Upbyson(t0,t1,t2);

return t0;

}

Upbyson(a[i],a[L(i)],a[R(i)]);

}

//----------------------------

/\*void randomData() {

srand((unsigned)time(NULL));

freopen("in", "w", stdout);

int dataset=100,x;

cout<<dataset<<endl;

while(dataset--) {

int n=rand()%10+1, m=rand()%(n\*n)+1;

cout<<n<<" "<<m<<endl;

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

x=rand()%2;

cout<<x<<" ";

}

cout<<endl;

while(m--) {

int op=rand()%5,p=rand()%n, q=rand()%n;

if(p>q)

swap(p,q);

cout<<op<<" "<<p<<" "<<q<<endl;

}

}

}

\*/

//------------------------------------------------------

int main(int argc, char\*\* argv)

{

// randomData();

// return 0;

// freopen("in","r",stdin);

// freopen("out","w",stdout);

int n,m,i,p,q,op,T;

seg ans;

scanf("%d",&T);

while(T--)

{

scanf("%d%d",&n,&m);

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

scanf("%d",data+i);

build(1,1,n);

while(m--)

{

scanf("%d%d%d",&op,&p,&q);

++p;++q;

if(p>q)Swap(p,q);

if(op<3)

{

Update(1,p,q,op);

}

else

{

ans=Query(1,p,q,op);

if(op==3)

printf("%d\n",ans.sum);

else

printf("%d\n",ans.max1);

}

}

}

return 0;

}