线段树总模板

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <vector>
using namespace std;
typedef long long LL;
#define ls (rt<<1)
#define rs (rt<<1|1)
#define lson ls,l,m
#define rson rs,m+1,r
#define root 1,1,n
const int N=1e5+2;
const int inf=0x3f3f3f3f;
int SUM[4*N],MAX[4*N],MIN[4*N],add[4*N],color[4*N],XOR[4*N];
/*****************************************************************
                                向上更新
 *****************************************************************/
void PushUp(int rt,int l,int r) {
    SUM[rt]=SUM[rt<<1]+SUM[rt<<1 | 1];
    MAX[rt]=max( MAX[rt<<1] , MAX[rt<<1 | 1]);
    MIN[rt]=min( MIN[rt<<1] , MIN[rt<<1 | 1]);
}
/*****************************************************************
                                反转
 *****************************************************************/
void Reserval(int rt,int l,int r) {
    if (color[rt]!=-1) color[rt]^=1;
    else    XOR[rt]^=1;
}
/*****************************************************************
                                向下传递
 *****************************************************************/
void PushDown(int rt,int l,int r) {
    int m=(l+r)>>1;
    if (add[rt]) {
        add[ls]        +=add[rt];
        add[rs]        +=add[rt];
        SUM[ls]        +=add[rt]*(m-l+1);
        SUM[rs]        +=add[rt]*(r-m);
        MAX[ls]        +=add[rt];
        MAX[rs]        +=add[rt];
        MIN[ls]        +=add[rt];
        MIN[rs]        +=add[rt];
        add[rt]=0;
    }
    if(color[rt]!=-1) {
        color[ls]    =color[rt];
        color[rs]    =color[rt];
        SUM[ls]        =(m-l+1)*color[rt];
        SUM[rs]        =(r-m)*color[rt];
        MAX[ls]        =color[rt];
        MAX[rs]        =color[rt];
        MIN[ls]        =color[rt];
        MIN[rs]        =color[rt];
        color[rt]=-1;
    }
```

```
    if (XOR[rt]){
        Reserval(lson);
        Reserval(rson);
        XOR[rt]=0;
    }
}
/*****************************************************************
                                build
 *****************************************************************/
void build(int rt,int l,int r) {
    add[rt]=0;
    color[rt]=0;
    if (l==r) {
        cin>>SUM[rt];
        MAX[rt]=SUM[rt];
        MIN[rt]=SUM[rt];
        return ;
    }
    int m=(l+r)>>1;
    build(lson);
    build(rson);
    PushUp(rt,l,r);
}
/*****************************************************************
                              单点增减
 *****************************************************************/

void point_add(int p,int addv,int rt,int l,int r) {
    if (l==r) {
        SUM[rt]+=addv;
        MAX[rt]+=addv;
        MIN[rt]+=addv;
        return ;
    }
    int m=(l+r)>>1;
    if (p<=m) point_add(p,addv,lson);
    else      point_add(p,addv,rson);
    PushUp(rt,l,r);
}

/*****************************************************************
                              单点赋值
 *****************************************************************/

void point_assign(int p,int val,int rt,int l,int r) {
    if (l==r) {
        SUM[rt]=val;
        MAX[rt]=val;
        MIN[rt]=val;
        return ;
    }
    int m=(l+r)>>1;
    if (p<=m) point_assign(p,val,lson);
    else      point_assign(p,val,rson);
    PushUp(rt,l,r);
}
```

```
/*****************************************************************************
                                区间增减
 *****************************************************************************/

void interval_add(int L,int R,int addv,int rt,int l,int r) {
    if (L<=l && r<=R) {
        add[rt]+=addv;
        SUM[rt]+=addv*(r-l+1);
        MAX[rt]+=addv;
        MIN[rt]+=addv;
        return ;
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    if(L<=m) interval_add(L,R,addv,lson);
    if(R>m) interval_add(L,R,addv,rson);
    PushUp(rt,l,r);
}

/*****************************************************************************
                                区间赋值
 *****************************************************************************/

void interval_assign(int L,int R,int val,int rt,int l,int r) {
    if (L<=l && r<=R) {
        color[rt]=val;
        SUM[rt]=val*(r-l+1);
        MAX[rt]=val;
        MIN[rt]=val;
        return ;
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    if(L<=m) interval_assign(L,R,val,lson);
    if(R>m) interval_assign(L,R,val,rson);
    PushUp(rt,l,r);
}

/*****************************************************************************
                                异或
 *****************************************************************************/

void FXOR(int L,int R,int rt,int l,int r) {
    if (L<=l && r<=R) {
        Reserval(rt,l,r);
        return ;
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    if(L<=m) FXOR(L,R,lson);
    if(R>m)  FXOR(L,R,rson);
    PushUp(rt,l,r);
}


/*****************************************************************************
                                区间求和
```

```
 ************************************************************/

int query_SUM(int L,int R,int rt,int l,int r) {
    if (L<=l && r<=R) {
        return SUM[rt];
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    int res=0;
    if (L<=m) res+=query_SUM(L,R,lson);
    if (R>m)  res+=query_SUM(L,R,rson);
    return res;
}

/************************************************************
                          区间最大值
 ************************************************************/

int query_MAX(int L,int R,int rt,int l,int r) {
    if (L<=l && r<=R) {
        return MAX[rt];
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    int res=-inf;
    if (L<=m) res=max( res , query_MAX(L,R,lson) );
    if (R>m)  res=max( res , query_MAX(L,R,rson) );
    return res;
}

/************************************************************
                          区间最小值
 ************************************************************/

int query_MIN(int L,int R,int rt,int l,int r) {
    if (L<=l && r<=R) {
        return MIN[rt];
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    int res=inf;
    if (L<=m) res=min( res , query_MIN(L,R,lson) );
    if (R>m)  res=min( res , query_MIN(L,R,rson) );
    return res;
}

/************************************************************
                            主程序
 ************************************************************/
void solve() {
    int n;
    cin>>n;
    build(root);
}

int main() {
    int T;
```

```
    cin>>T;
    while(T--) {
        solve();
    }
    return 0;
}
```