**Segment Tree [with lazy], insert O(nlogn), query O(nlogn)**

**Tested on Codeforces 52C, Circular RMQ**

int mn[MX\*4], lazy\_value[4\*MX];

void update\_node(int idx, int st, int ed, int val)

{

lazy\_value[idx]+=val, mn[idx]+=val;

return;

}

void update\_lazy(int idx, int st, int ed)

{

if(st==ed) return;

int mid=(st+ed)>>1, lft=idx<<1, rgt=lft | 1;

update\_node(lft, st, mid, lazy\_value[idx]);

update\_node(rgt, mid+1, ed, lazy\_value[idx]);

lazy\_value[idx]=0;

}

void insert(int idx, int st, int ed, int s, int e, int val)

{

if(st==s && ed==e)

{

update\_node(idx, st, ed, val);

return;

}

if(lazy\_value[idx]) update\_lazy(idx, st, ed);

int mid=(st+ed)>>1, lft=idx<<1, rgt=lft | 1;

if(e<=mid) insert(lft, st, mid, s, e, val);

else if(s>mid) insert(rgt, mid+1, ed, s, e, val);

else

{

insert(lft, st, mid, s, mid, val);

insert(rgt, mid+1, ed, mid+1, e, val);

}

mn[idx]=min(mn[lft], mn[rgt]);

return;

}

int query(int idx, int st , int ed, int s, int e)

{

if(st==s && ed==e) return mn[idx];

if(lazy\_value[idx]) update\_lazy(idx, st, ed);

int mid=(st+ed)>>1, lft=idx<<1, rgt=lft | 1;

if(mid>=e) return query(lft, st, mid, s, e);

else if(s>mid) return query(rgt, mid+1, ed, s, e);

else return min(query(lft, st, mid, s, mid), query(rgt, mid+1, ed, mid+1, e));

}