```
ll BIT[MAX],cnt;
void update(int inx)
    {
    while(inx<=cnt)
        {
        BIT[inx]++;
        inx+=(inx&(-inx));
        }
    }
ll get(int inx)
    {
    ll ans=0;
    while(inx>0)
        {
        ans += BIT[inx];
        inx -= (inx&(-inx));
        }
    return ans;
    }
```

To build: just call update n times.
inx is the index in original array

```
void build(int st,int en,int i){
    if(st==en){
        seg[i] = a[st];
        return;
        }
    int mid=(st+en)>>1;
    build(st,mid,2*i+1);
    build(mid+1,en,2*i+2);
    seg[i] = min(seg[2*i+1],seg[2*i+2]);
    }
ll get(int st,int en,int l,int r,int i){
    if(st>r || en<l)
        return inf;
    if(l<=st && en<=r)
        return seg[i];
    int mid = (st+en)>>1;
    ll le = get(st,mid,l,r,2*i+1),ri = get(mid+1,en,l,r,2*i+2);
    return min(le,ri);
    }
void update(int st,int en,int i,int pos,ll val){
    seg[i] = min(seg[i],val);
    if(st==en)
        return;

    int mid = (st+en)>>1;
    if(pos<=mid && pos>=st)
        update(st,mid,2*i+1,pos,val);
    else if(pos>mid && pos<=en)
        update(mid+1,en,2*i+2,pos,val);
}
```

basically st en denote the range of values
l and r denote the range query
pos is the actual position you want to update

```
build(0,n,0);
get(0,n,l,r,0);
update(0,n,0,pos,val);
```