First element at least X

Problem

You are given an array of size n. You will be given m range queries and point updates on the array. Queries and updates on the array will be of the type given below

1 i v: Update a[i] to v

2 x: Find the minimum index j such that a[j] >= x. If there is no such index, print -1.

Constraints

$$1 <= n, m <= 10^5$$

$$0 <= x <= 10^9$$

Example input

```
5 7
1 3 2 4 6
2 2
2 5
1 2 5
2 4
2 8
1 3 7
2 6
```

Output

```
1
4
2
-1
3
```

Approach

Main idea: Segment Tree + Binary Search.

1. Make a segment tree of max queries and update.
2. Binary Search on the interval [0,n-1], starting from lo=0 and hi=n-1 and keep updating the ans index (i.e. the minimum mid such that a[mid] >= x).

Code

```
#include "bits/stdc++.h"
using namespace std;
#define int long long
const int N = 1e5+2, MOD = 1e9+7;

int tree[4*N], a[N];

void build(int node, int st, int en)
{
    if(st == en){
        tree[node] = a[st];
        return;
    }

    int mid = (st + en)/2;
    build(2*node, st, mid);
    build(2*node+1, mid+1, en);

    tree[node] = max(tree[2*node], tree[2*node+1]);
}

int query(int node, int st, int en, int l, int r){
    if(st>r || en<l)
        return -MOD;

    if(l<=st && en<=r)
        return tree[node];

    int mid = (st + en)/2;
    int q1 = query(2*node, st, mid, l, r);
    int q2 = query(2*node+1, mid+1, en, l, r);

    return max(q1, q2);
}

void update(int node, int st, int en, int idx, int val){
    if(st == en){
        a[st] = val;
        tree[node] = val;
```

```
        return;
    }

    int mid = (st+en)/2;
    if(idx <= mid){
        update(2*node, st, mid, idx, val);
    }
    else
    {
        update(2*node+1, mid+1, en, idx, val);
    }

    tree[node] = max(tree[2*node], tree[2*node+1]);
}

signed main()
{
    int n,m; cin >> n >> m;
    for(int i=0; i<n; i++){
        cin >> a[i];
    }
    build(1,0,n-1);
    while(m--){
        int type;
        cin >> type;
        if(type == 1){
            int idx,val;
            cin >> idx >>val;
            update(1,0,n-1,idx,val);
        }
        else if(type == 2){
            int x;
            cin >> x;
            int lo = 0, hi = n-1;
            int ans = n;
            while(lo<=hi){
                int mid = (lo+hi)/2;
                if(query(1,0,n-1,lo,mid) < x){
                    lo = mid+1;
                }
```

```
                else {
                    hi = mid-1;
                    ans = min(ans, mid);
                }
            }

            if(ans == n){
                cout << "-1" << endl;
            }
            else{
                cout << ans << endl;
            }
        }
    }
    return 0;
}
```