数组版

```
#include<bits/stdc++.h>
//Luogu 3919
using namespace std;
const int MAXN = 1e6 + 10;
struct node
{
        int ls, rs;
        int data;
}tree[MAXN * 40];
int root[MAXN << 1], tot;
int a[MAXN], n, m;
inline int build(int l, int r)
{
        int p = ++tot;
        if(l == r)
        {
                tree[p].data = a[l];
                return p;
        }
        int mid = (l + r) >> 1;
        tree[p].ls = build(l, mid);
        tree[p].rs = build(mid + 1, r);
        return p;
}
inline int insert(int tr, int l, int r, int pos, int d)
{
        int p = ++tot;
        tree[p] = tree[tr];
        if(l == r)
        {
                tree[p].data = d;
                return p;
        }
        int mid = (l + r) >> 1;
        if(pos <= mid)
                tree[p].ls = insert(tree[tr].ls, l, mid, pos, d);
        else
                tree[p].rs = insert(tree[tr].rs, mid + 1, r, pos, d);
        return p;
}
inline int query(int tr, int l, int r, int pos)
{
        if(l == r)
        {
                return tree[tr].data;
        }
        int mid = (l + r) >> 1;
        if(pos <= mid)
                return query(tree[tr].ls, l, mid, pos);
        else
                return query(tree[tr].rs, mid + 1, r, pos);
}
int main()
{
        ios::sync_with_stdio(false);
        cin >> n >> m;
```

```
        for(int i = 1; i <= n; ++i)
                cin >> a[i];
        root[0] = build(1, n);
        for(int i = 1; i <= m; ++i)
        {
                int v, opt, id, d;
                cin >> v >> opt >> id;
                if(opt == 1)
                {
                        cin >> d;
                        root[i] = insert(root[v], 1, n, id, d);
                }
                else
                {
                        cout << query(root[v], 1, n, id) << endl;
                        root[i] = root[v];
                }
        }
}
```

指针版

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 1e6 + 10;
class segment_tree
{
private:
        struct Tree
        {
                int data;
                Tree *ls, *rs;
        };
        vector<Tree*> root;
public:
        segment_tree(int l, int r, int a[])
        {
                Tree *p = build(l, r, a);
                root.push_back(p);
        }
        Tree* build(int l, int r, int a[])
        {
                Tree *p = new Tree;
                if(l == r)
                {
                        p->data = a[l];
                        return p;
                }
                int mid = (l + r) >> 1;
                p->ls = build(l, mid, a);
                p->rs = build(mid + 1, r, a);
                return p;
        }
        Tree *insert(Tree *tr, int l, int r, int pos, int d)
        {
                Tree *p = new Tree;
                (*p) = (*tr);
                if(l == r)
                {
                        p->data = d;
                        return p;
                }
                int mid = (l + r) >> 1;
                if(pos <= mid)
                        p->ls = insert(tr->ls, l, mid, pos, d);
                else
                        p->rs = insert(tr->rs, mid + 1, r, pos, d);
                return p;
        }
        int query(Tree *tr, int l, int r, int pos)
        {
                if(l == r)
                {
                        return tr->data;
                }
                int mid = (l + r) >> 1;
                if(pos <= mid)
                        return query(tr->ls, l, mid, pos);
                else
                        return query(tr->rs, mid + 1, r, pos);
```

```
        }
        void change(int tr, int l, int r, int pos, int d)
        {
                Tree *p = insert(root[tr], l, r, pos, d);
                root.push_back(p);
        }
        int ask(int tr, int l, int r, int pos)
        {
                return query(root[tr], l, r, pos);
        }
        void copy(int id)
        {
                root.push_back(root[id]);
        }
};
int n, m;
int a[MAXN];
int main()
{
        scanf("%d %d", &n, &m);
        for(int i = 1; i <= n; ++i)
        {
                scanf("%d", &a[i]);
        }
        segment_tree TREE(1, n, a);
        for(int i = 1; i <= m; ++i)
        {
                int v, opt, id, d;
                scanf("%d %d %d", &v, &opt, &id);
                if(opt == 1)
                {
                        scanf("%d", &d);
                        TREE.change(v, 1, n, id, d);
                }
                else
                {
                        printf("%d\n", TREE.ask(v, 1, n, id));
                        TREE.copy(v);
                }
        }
}
```