可持久化线段树

Persistent Segment Tree

Idea：将线段树可持久化，每一棵新的线段树都在前一棵线段树上做扩充。

ATT：空间一般开 20 倍。

Complexity：$O(n\lg n)$

Code（以单点修改，单点查询为例）：

```cpp
struct segTree{
    int l, r, lson, rson, val;
}tr[20000005];
int cnt, root[N];
void build(int id, int l, int r){
    tr[id].l = l; tr[id].r = r;
    if(tr[id].l == tr[id].r){
        tr[id].val = a[l];
        return;
    }
    tr[id].lson = ++cnt;
    tr[id].rson = ++cnt;
    int mid = (tr[id].l + tr[id].r) >> 1;
    build(tr[id].lson, l, mid);
    build(tr[id].rson, mid+1, r);
}
void modify(int cur, int pre, int pos, int val){ // modify value of pos in current tree to val based on previous tree
    tr[cur] = tr[pre];
    if(tr[cur].l == tr[cur].r){
        tr[cur].val = val;
        return;
    }
    int mid = (tr[cur].l + tr[cur].r) >> 1;
    if(pos <= mid){
        tr[cur].lson = ++cnt;
        modify(tr[cur].lson, tr[pre].lson, pos, val);
    }
    else{
        tr[cur].rson = ++cnt;
        modify(tr[cur].rson, tr[pre].rson, pos, val);
    }
}
int queryVal(int id, int pos){ // query value stored in pos of tr[id]
    if(tr[id].l == tr[id].r)    return tr[id].val;
    int mid = (tr[id].l + tr[id].r) >> 1;
    if(pos <= mid)  return queryVal(tr[id].lson, pos);
    else    return queryVal(tr[id].rson, pos);
}

int main(){
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i++)
        scanf("%d", &a[i]);
    root[0] = ++cnt;
    build(root[0], 1, n);
    for(int i = 1; i <= m; i++){
        scanf("%d%d%d", &ver, &opt, &loc);
        if(opt == 1){
```

```c
            scanf("%d", &value);
            root[i] = ++cnt;
            modify(root[i], root[ver], loc, value);
        }
        else if(opt == 2){
            printf("%d\n", queryVal(root[ver], loc));
            root[i] = ++cnt;
            tr[root[i]] = tr[root[ver]];
        }
    }
    return 0;
}
```