Team notebook

January 1, 2019

Contents

1 segment tree

1.1 Segment tree

```
#include <bits/stdc++.h>
using namespace std;
#define vi vector<long long>


class segmenTree{
    private: vi st, A;
    int n;
    int left(int p){return p<<1;}
    int right(int p){return (p<<1)+1;}

    void build(int p, int L, int R){
        if(L==R)
            st[p] = L;
        else{
            build(left(p) , L, (L+R)/2);
            build(right(p), (L+R)/2+1, R);
            int p1=st[left(p)], p2 = st[right(p)];
            st[p] = (A[p1] >= A[p2])? p1 : p2;
        }
    }
    int rmq(int p, int L, int R, int i, int j){
        if(i > R || j < L)return -1;
        if(L >= i && R <= j) return st[p];
        int p1 = rmq(left(p), L, (L+R)/2, i, j);
        int p2 = rmq(right(p), (L+R)/2+1, R, i, j);

        if(p1 == -1)return p2;
        if(p2 == -1)return p1;
        return (A[p1] >= A[p2])? p1 : p2;
    }

    void update(int p, int L, int R, int i){
        if(L==R)st[p]=L;
        else if(L<=i && i<=R){
            update(left(p), L, (R+L)/2, i);
            update(right(p), (R+L)/2+1, R, i);
            int p1 = st[left(p)], p2 = st[right(p)];
            st[p] = (A[p1] >= A[p2])? p1 : p2;
        }
```

```
    }
    public:
    segmenTree(const vi &_A){
        A = _A; n = (int)A.size();
        st.assign(4*n, 0);
        build(1, 0, n-1);
    }

    long long rmq(int i, int j){
        return A[rmq(1, 0, n-1, i, j)];
    }

    void update(int i, int w){A[i]+=w; update(1, 0, n-1, i);}
};

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    int N, Q, t, i, j;
    cin >> N >> Q;
    vector<long long> v(N);
    for(int i =0; i <N; i++)cin >> v[i];
    segmenTree st(v);
    while(Q--){
        cin >> t >> i >> j;
        if(t==2)cout << st.rmq(i-1,j-1)<< '\n';
        else st.update(i-1, j);
    }
    return 0;
}
```

2 UnionFind

```
#include <bits/stdc++.h>
using namespace std;
typedef vector<int> vi;

class UnionFind{
       private: vi rank, p;
       public:
              UnionFind(int N){ rank.assign(N,0);
                     p.assign(N,0); for(int i=0; i<N; i++)p[i]=i;}
              int findSet(int i){ return (p[i]==i)? i : (p[i]=findSet(p[i]));}
              bool isSameSet(int i, int j){return findSet(i) == findSet(j);}
              void unionSet(int i, int j){
                if(!isSameSet(i,j)){
                     int x = findSet(i), y = findSet(j);
                     if(rank[x] > rank[y]) p[y] = x;
                     else{
                            p[x] = y;
                            if(rank[x] == rank[y]) rank[y]++;
              }}}
};


int main(){
       ios_base::sync_with_stdio(false);cin.tie(NULL);

       return 0;
}
```