```cpp
// https://toph.co/p/rio-and-inversion
#include<bits/stdc++.h>
using namespace std;
#define N tree[node] #define L tree[lson] #define R tree[rson]
int a[10002],dp[10002][10002];
vector<int>tree[4*10002];
void build(int node, int b, int e){
    if(b==e){ tree[node].push_back(a[b]); return; }
    int lson=(node<<1), rson=lson+1, m=(b+e)/2;
    build(lson,b,m);
    build(rson,m+1,e);
    merge(L.begin(),L.end(),R.begin(),R.end(),back_inserter(N));
}
int query(int node,int b,int e,int x,int y,int v,int c){
    if(x>y)return 0;
    if(b==x && e==y){
        if(c==0){
            int cnt = lower_bound(N.begin(),N.end(),v)-N.begin();
            return cnt;
        }else{
            int cnt = upper_bound(N.begin(),N.end(),v)-N.begin();
            return (e-b+1)-cnt;
        }
    }
    int lson=(node<<1), rson=lson+1, m=(b+e)/2;
    if(y<=m) return query(lson,b,m,x,y,v,c);
    else if(x>m) return query(rson,m+1,e,x,y,v,c);
    else return query(lson,b,m,x,m,v,c) + query(rson,m+1,e,m+1,y,v,c);
}
int main(){
    int n; scanf("%d",&n);
    for(int i=1; i<=n; i++)scanf("%d",&a[i]);

    build(1,1,n);

    int ans = 0;
    for(int i=1; i<=n; i++){
        ans += query(1,1,n,i+1,n,a[i],0);
    }

    dp[0][0] = dp[1][0] = dp[n+1][n] = ans;
    for(int j=1; j<=n; j++){
        dp[1][j] = dp[1][j-1] - query(1,1,n,j,n,a[j],0);
    }
    for(int i=n; i>=1; i--){
        dp[i][n] = dp[i+1][n] - query(1,1,n,1,i,a[i],1);
    }
    for(int i=2; i<n; i++){
        for(int j=n-1; j>=i; j--){
            dp[i][j] = dp[i-1][j] + dp[i][j+1] - dp[i-1][j+1];
            if(a[i-1]>a[j+1]) dp[i][j]++;
        }
    }
    int q; scanf("%d",&q);
    while(q--){
        int type; scanf("%d",&type);
        if(type==0){
            int x,y; scanf("%d%d",&x,&y); x++, y++; if(x>y)swap(x,y);
            int res = ans;
            if(a[x]!=a[y]){
                res -= query(1,1,n,x+1,n,a[x],0);
                res -= query(1,1,n,1,x-1,a[x],1);
                res -= query(1,1,n,y+1,n,a[y],0);
                res -= query(1,1,n,1,y-1,a[y],1);
                if(a[x]>a[y])res++;

                res += query(1,1,n,x+1,n,a[y],0);
                res += query(1,1,n,1,x-1,a[y],1);
                res += query(1,1,n,y+1,n,a[x],0);
                res += query(1,1,n,1,y-1,a[x],1);
                if(a[x]<a[y])res++;
            }
            printf("%d\n",res);
        }else{
            int x,y; scanf("%d%d",&x,&y); x++, y++; if(x>y)swap(x,y);
            int res = dp[x][y];
            printf("%d\n",res);
        }
    }
    return 0;
}
```