## GSS1 - Can you answer these queries I & GSS3 - Can you answer these queries III:

**Problem:** You are given a sequence A[1], A[2],...,A[N]. (|A[i]|≤15007, 1≤N≤50000).
A query is defined as follows:
Query(x,y) = Max { a[i]+a[i+1]+...+a[j] ; x ≤ i ≤ j ≤ y }.
Given M queries, your program must output the results of these queries.

**Solution:** Lets analyze this query.

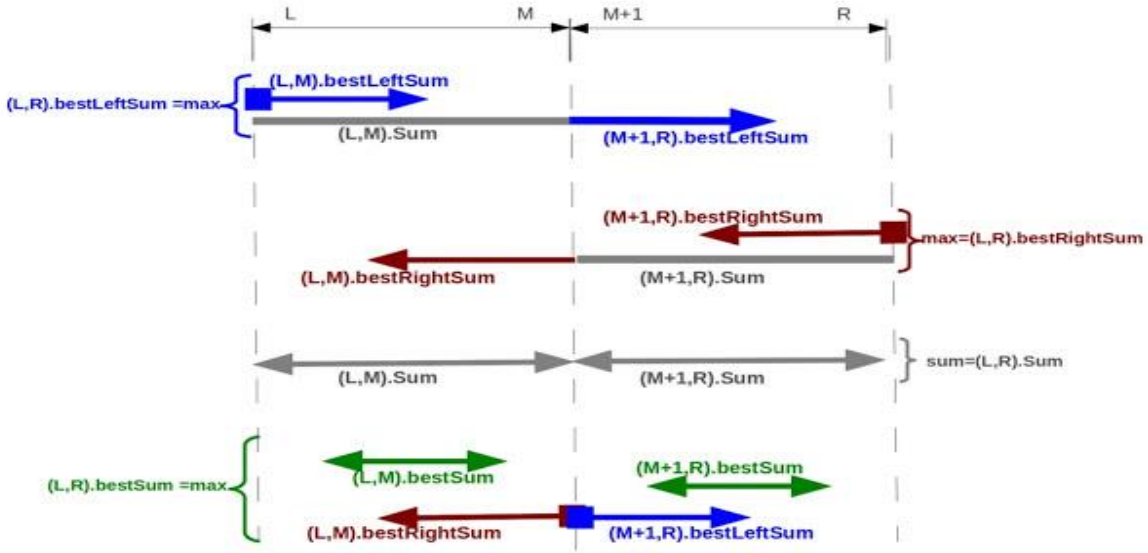
Query(x, y) is the maximum of below values where sum(i, j) = a[i]+a[i+1]+.....+a[j].

**Left Sum:** Where the range starts at 'x' but ends at anywhere in [x, y].
**Right Sum:** Where the range starts at somewhere in [x, y] and ends at 'y'.
**Sum:** Simply sum of all elements in interval [x, y].
**Best Sum:** Maximum possible sum in range [x, y].

Lets maintain 4 values.
**bestLeftSum** - Best of all the Left Sums, **bestRightSum** - Best of all the Right Sums
**Sum** - Sum of all the elements, **bestSum** - need to store resultSum of each query.
Logically, Query(x, y).resultSum = max (bestLeftSum, bestRightSum, sum, Query(x+1,y-1).bestSum);

Lets call the above info {bestLeftSum, bestRightSum, Sum,bestSum} as QueryNode. and given information about QueryNode(L, M) and QueryNode(M+1, R).
QueryNode(L, M) -> l and QueryNode(M+1, R) -> r. Then For QueryNode(L, R),

**bestLeftSum = max (l.bestLeftSum, l.Sum + r.bestLeftSum);**
**bestRightSum = max (l.bestRightSum + r.Sum, r.bestRightSum);**
**Sum = l.Sum + r.Sum;**
**bestSum = max(l.bestSum, r.bestSum, l.bestRightSum + r.bestLeftSum)**



```cpp
/* SPOJ - GSS3 - Can you answer these queries III
Given a sequence A of N(N<=50000) integers between -10000 and 10000.
On this sequence you have to apply M (M <= 50000) operations:
Modify the i-th element in the sequence or
For given x y print max{Ai + Ai+1 + .. + Aj | x<=i<=j<=y }.
*/
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define MAXN 50005
#define lf nd<<1
#define rg (nd<<1)+1
#define m (int)((b+e)>>1)
#define N tree[nd]
#define L tree[lf]
#define R tree[rg]

int a[MAXN];
struct data{
    ll prefix,suffix,bestSum,sum;
}tree[4*MAXN];

data maxData(data x, data y){
    data z;
    z.prefix = max(x.prefix, x.sum + y.prefix);
    z.suffix = max(y.suffix, y.sum + x.suffix);
    z.bestSum = max(max(x.bestSum, y.bestSum), x.suffix + y.prefix);
    z.sum    = x.sum + y.sum;
    return z;
}
void build(int nd,int b,int e){
    if(b==e){ N.prefix = N.suffix = N.bestSum = N.sum = a[b]; return; }
    build(lf,b,m);
    build(rg,m+1,e);
    N = maxData(L,R);
}
void update(int nd,int b,int e,int x,int v){
    if(b==e && b==x){ N.prefix = N.suffix = N.bestSum = N.sum = a[b] = v; return; }
    if(x<=m)update(lf,b,m,x,v);
    else update(rg,m+1,e,x,v);
    N = maxData(L,R);
}
data query(int nd,int b,int e,int x,int y){
    if(b>y || e<x){ data INF; INF.prefix=INF.suffix=INF.bestSum=INF.sum=-1000000000; return INF; }
    if(b>=x && e<=y) return tree[nd];
    return maxData(query(lf,b,m,x,y), query(rg,m+1,e,x,y));
}
int main(){
    int n; scanf("%d",&n);
    for(int i=1; i<=n; i++) scanf("%d",&a[i]);

    build(1,1,n);

    int q; scanf("%d",&q);
    while(q--){
        int t,x,y; scanf("%d%d%d",&t,&x,&y);
        if(t==0){
            update(1,1,n,x,y);
        }else{
            data ans = query(1,1,n,x,y);
            ll res = max(max(ans.prefix,ans.suffix), max(ans.bestSum,ans.sum));
            printf("%lld\n",res);
        }
    }
}
```