SPOJ - GSS5 - Can you answer these queries V:

Problem: Given a array of numbers a[1...n] , and a Query(x1,y1,x2,y2).

Query(x1, y1, x2, y2) = Max { A[i]+A[i+1]+...+A[j] ; x1 <= i <= y1 , x2 <= j <= y2 and x1 <= x2 , y1 <= y2 }.

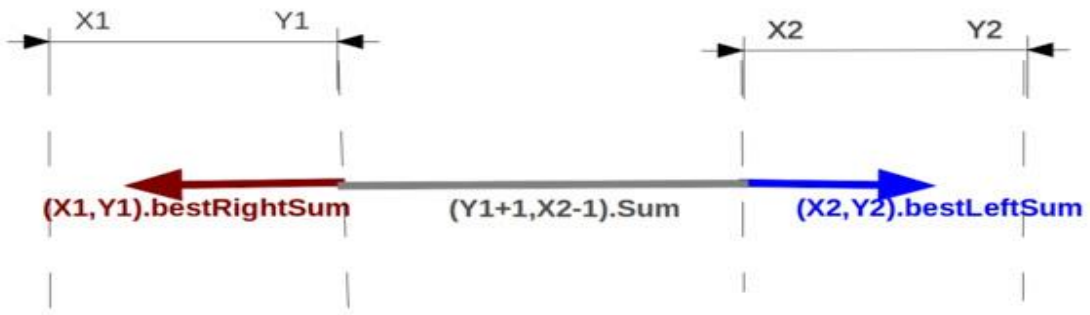
Solution: Here two cases arise based on conditions,

Case-1: (x1, y1) and (x2, y2) doesn't overlap.

Case-2: (x1, y1) and (x2, y2) overlaps.

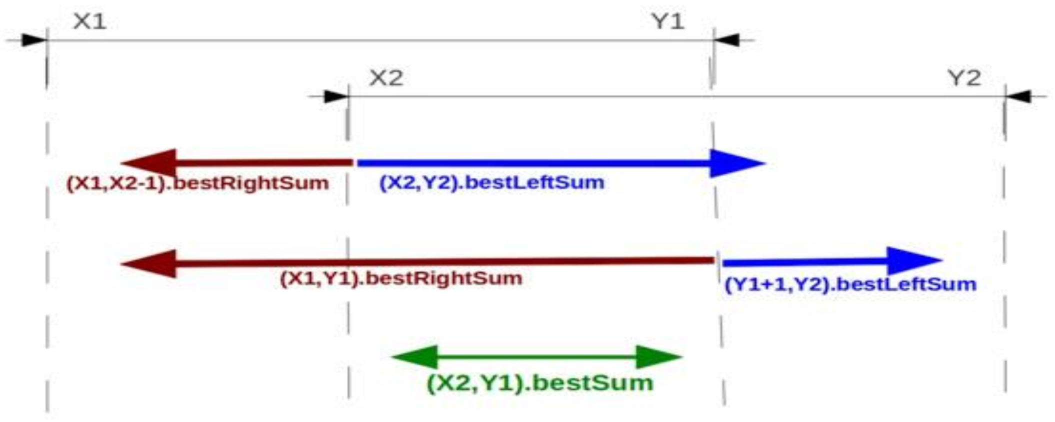
- **Case-1: No Overlapping**

  Result would be (x1, y1).bestRightSum + (y1+1, x2-1).sum + (x2, y2).bestLeftSum



- **Case-2: Overlapping**

  Result would be max of
  {
  (x1, x2-1).bestRightSum + (x2, y2).bestLeftSum,
  (x1, y1).bestRightSum + (y1+1, y2).bestLeftSum,
  (x2, y1).bestSum
  }



```cpp
// https://www.spoj.com/problems/GSS5/en/ GSS5 - Can you answer these queries V
#define ll long long
#define INF 10000000000000000LL
#define MAXN 10005
#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 n,a[MAXN];

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

data maxData(data l, data r){
    data z;
    z.prefix  = max(l.prefix, l.sum + r.prefix);
    z.suffix  = max(r.suffix, r.sum + l.suffix);
    z.bestSum = max(max(l.bestSum, r.bestSum), l.suffix + r.prefix);
    z.sum     = l.sum + r.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);
}
data query(int nd,int b,int e,int x,int y){
    if(x>y) { data d; d.prefix=d.suffix=d.bestSum=d.sum=0; return d; }
    if(b==x && e==y) return tree[nd];
    if(y<=m) return query(lf,b,m,x,y);
    else if(x>m) return query(rg,m+1,e,x,y);
    else return maxData(query(lf,b,m,x,m), query(rg,m+1,e,m+1,y));
}
ll solve(int x1,int y1,int x2,int y2){
    ll ans = -INF;
    if(y1 < x2){
        ans = query(1,1,n,x1,y1).suffix + query(1,1,n,y1+1,x2-1).sum + query(1,1,n,x2,y2).prefix;
    }else {
        ans = query(1,1,n,x1,x2-1).suffix + query(1,1,n,x2,y2).prefix;
        ans = max(ans, query(1,1,n,x1,y1).suffix + query(1,1,n,y1+1,y2).prefix);
        ans = max(ans, query(1,1,n,x2, y1).bestSum);
    }
    return ans;
}
int main(){
    int tt; scanf("%d",&tt);
    while(tt--){
        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 x1,y1,x2,y2; scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
            ll ans = solve(x1,y1,x2,y2);
            printf("%lld\n",ans);
        }
    }
}
```