p6-ex4

OBS: this exercise is not-trivial only if some of the integers are negative, otherwise the best sum will be tri vially given by the sum of all the n integers.

My solution exploit the hint: we will use a tournment tree where each node will be

```
node{

    sum
    max_prefix
    max_suffix
    max_subinterval

}
```

Let A be the input sequence.
We initialize the tree as we saw during lecture:

init(A):

```
num_leafs = first power of 2 larger than n    //where n is len(A)
tree = array of size (2 * num_leafs) - 1 initialized to 0

//initialize leafs
for i = 0...n-1
    tree[num_leafs + i].sum = A[i]
    tree[num_leafs + i].max_prefix = A[i]
    tree[num_leafs + i].max_suffix = A[i]
    tree[num_leafs + i].max_subinterval = A[i]

//initialize upper nodes
for i = num_leafs - 1....1

    tree[i].sum = tree[2i].sum + tree[2i+1].sum
    //the total sum of an interval is simply the sum of its left half plus the sum of its right half

    tree[i].max_prefix = max(tree[2i].max_prefix, tree[2i].sum + tree[2i+1].max_prefix)
    //this must necesserly involve the first integers of the interval,
    //and the best prefix might exceed the first half, i.e.involving
    //the whole left subinterval and the best right prefix,
    //or not, i.e. being the left prefix itself

    tree[i].max_suffix = max(tree[2i+1].max_suffix, tree[2i+1].sum + tree[2i].max_suffix)
    //as above

    tree[i].max_subinterval = max(tree[2i].max_subinterval, tree[2i+1].max_subinterval, tree[2i].max_suffi
x + tree[2i+1].max_prefix)
    //the optimal subinterval might be all inside the left half or the right half,
    //or it might involve both last integers in the left half
    //and first integers in the right half
```

set(i,v) is almost identical to the one saw during lecture (just updating nodes as seen above), so updating parents up to the root.

```
set(i,v):

    tree[num_leafs + i] = v

    x = floor(num_leafs/2)

    while x >= 1
        tree[x].sum = tree[2x].sum + tree[2x+1].sum
        tree[x].max_prefix = ...
        tree[x].max_suffix = ...
        tree[x].max_subinterval = ...

max():

    return tree[1].max_subinterval
```