## Maximum Contiguous Subarray Sum

We saw an  $O(n \log n)$  divide and conquer algorithm.

Can we do better with DP?

Given: Array A[]

Output: i, j such that  $A[i] + A[i + 1] + \cdots + A[j]$  is maximized.

For today: just output the value $A[i] + A[i+1] + \cdots + A[j]$ .

*OPT*(*i*) is....

## Two Values

Pollev.com/robbie

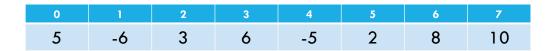
Need two recursive values:

INCLUDE(i): sum of the maximum sum subarray among elements from 0 to i that includes index i in the sum

 $\mathit{OPT}(i)$ : sum of the maximum sum subarray among elements 0 to i (that might or might not include i)

How can you calculate these values? Try to write recurrence(s), then think about memoization and running time.

## Longest Increasing Subsequence



Longest set of (not necessarily consecutive) elements that are increasing

5 is optimal for the array above (indices 1,2,3,6,7; elements -6,3,6,8,10)

For simplicity – assume all array elements are distinct.

## Longest Increasing Subsequence

LIS(i,j) is "Number of elements of the maximum increasing subsequence from 0, ..., i where every element of the sequence is at most A[j]"

Need a recurrence

$$LIS(i,j) = \begin{cases} ? & \text{if } i < 0 \\ ? & \text{if } i = 0 \\ ? & \text{if } A[i] > A[j] \\ ? & \text{otherwise} \end{cases}$$