Steps for designing dynamic programming algorithms

1. Define the subproblems – typically by defining an array of values you want to compute

2. Give a recurrence relation that describes how a subproblem can be solved using solutions of smaller subproblems
   - Imagine smaller subproblems have already been solved and the solutions can be looked up from your array
   - Consider different ways one can reduce the problem to a smaller subproblem from the current problem
   - Among these options pick the optimal one

3. Filling in the base cases of the array, and then use recurrence in 2 to incrementally fill up all values.

4. Return the final solution based on the filled array

Revisit: Maximum subarray

Problem Definition:

Given an array A of numbers, find the contiguous subarray that has the largest sum

Example: for input A=(4,-5,6,7,8,-10,5,2)
Solution: 6, 7, 8, sum=21

Two possible subproblems

L(i) = maximum subarray for A[1,…,i]

L(i) = maximum subarray ending at position i

Q: which one will allow us to compute L(i) given the values of L(1), L(2), …., L(i-1)?

Recurrence relation for L(i)

Fill in the algorithm

- Base case:
- Iteratively fill in the array
- Return the final solution