Lab 8 Recursion – II

(Note: This lab is a continuation of Lab 7. To get the credit of this lab, please submit the work of Lab 7 AND the work of this lab.)

Goal

In this lab you will continue and turn-in the work of Lab 7 and design and implement one extra recursive algorithm. The focus is on double recursion. The doubly recursive algorithm of this lab computes the maximum of an array.

Resources

- Chapter 7: Recursion

Java Files

- *BadArgumentsForMaxException.java*
- *RecursiveMaxOfArray.java*
- *TestMax.java*

Introduction

The application computes the maximum value in an array. It will split the array into halves and thus avoid an exponential performance cost. The pattern of this recursion is similar to what will be seen later for the advanced sorting algorithms.

As with the other recursive algorithms that work on an array, the recursion will be on a range of values that are being examined. This range will be split in half to get the subproblems.

To start, consider how the split will be made. Suppose the recursive algorithm is asked to look at the portion of an array that ranges from 3 to 9.

INDEX	...	3	4	5	6	7	8	9	...
VALUE		20	40	10	90	50	70	80	

How many values are there in the range?

What is the index of the middle value in the range?

If the limits of the range are *first* and *last*, give a formula to compute the index of the middle value.

The task now is to split up the array, but where should the middle value go? Potentially, it can go in the first half, the second half, or neither half. To answer this question, consider a portion of the array with just two entries. This is the smallest range that can be split up and is a useful test case. If this case does not work, the recursive algorithm is doomed to failure.

Using the preceding formula, what will be the index of the middle value?

In the following arrays, box the left and right halves for the given way of handling the middle value.

Middle is in first half:

INDEX	...	3	4	...
VALUE		20	40	

Middle is in second half

INDEX	...	3	4	...
VALUE		20	40	

Middle is in neither half

INDEX	...	3	4	...
VALUE		20	40	

What is the maximum of an empty range? This is problematic. The maximum is not defined in that situation. Only one of the three cases above will have elements in both of the halves.

If the limits of the range are *first* and *last* and middle is the middle index, what are the ranges for the first and second halves so that both halves are nonempty?

First half range:

Second half range:

Having thought about how to split the range in half, continue on with the recursive design.

Identify the problem:

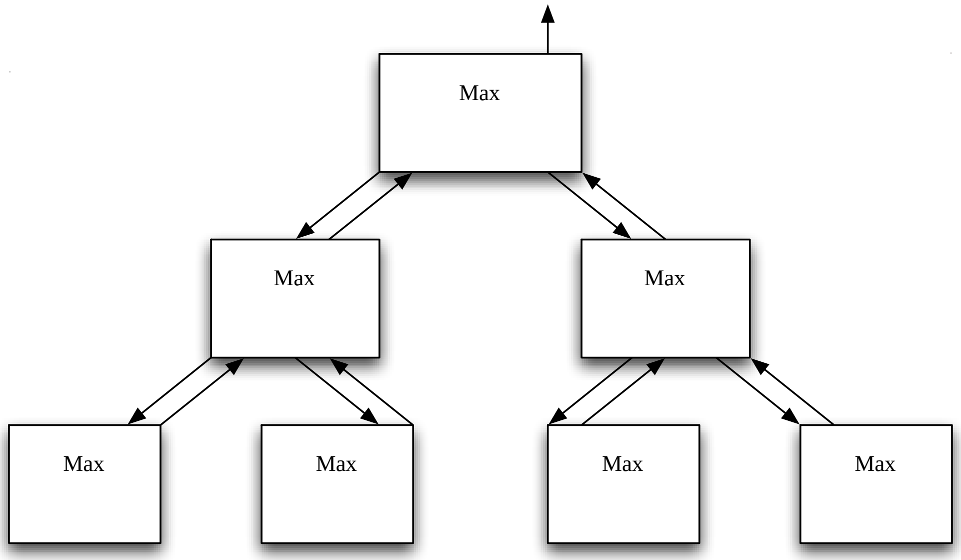
Identify the smaller problems:

Identify how the answers are composed:

Identify the base cases:

Compose the recursive definition:

Show the operation of your definition on the array [9 3 4 5 7] with the range of 0 to 4 on the following diagram. Inside the boxes, show the values of the arguments passed into the method. On the left-hand side, show the operations done before the recursive call by the method. On the right-hand side, show operations done after the recursive call. There are two more recursive calls that will be made but are not shown in the picture. Their location will depend on how the split was formulated. Add them in.



Directed Lab Work

Step 1. Look at the skeleton in `RecursiveMaxOfArray.java`. Compile and run the `main` method in `TestMax`.

Checkpoint: The program should run and fail all tests.

Refer to the recursive design from the pre-lab exercises. Complete the recursive method `max()`. Don't forget to throw an exception if there is not at least one value in the range.

Final checkpoint: Run TestMax. All tests should pass.

Post-Lab Follow-Ups

1. Develop code to time the performance of the recursive maximum. Plot the performance and identify the time complexity of the solution.

2. Develop and implement a doubly recursive algorithm for computing the product of all the values in an array.

3. Develop and implement a doubly recursive algorithm that moves the minimum value to the beginning of an array.