# Lab 3 Recursion—Part II

## Points: 100pts

## Goal

The exploration of recursion is continued in this lab with a focus on double recursion. One application will be developed. It computes the maximum of an array.

## Resources

* Chapter 6: Recursion

## Java Files

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

## Pre-Lab Visualization

### Maximum

This application will compute 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?

images:bighand.jpg 7 values

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

images:bighand.jpg 6

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

images:bighand.jpg

(Last – first) / 2

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?

images:bighand.jpg Nothing it will just compare the two numbers.

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

images:bighand.jpg **Middle is in first half:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | ... | 3 | 4 | ... |
| VALUE |  | 20 | 40 |  |

images:bighand.jpg Middle is in second half

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | ... | 3 | 4 | ... |
| VALUE |  | 20 | 40 |  |

images:bighand.jpg 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?

images:bighand.jpg First half range: first to mid

images:bighand.jpg Second half range: mid + 1 to last

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

images:bighand.jpg **Identify the problem**: With given values from and to find a way to compute the biggest value withing that range.

images:bighand.jpg **Identify the smaller problems**:

If the array is empty or if the from is negative and the to is greater than the length of the list.

images:bighand.jpg **Identify how the answers are composed**:

If any of these cases were to happen then throw an exception.

images:bighand.jpg **Identify the base cases**:

Find a way to split the list in half then find the biggest within the first half and the second half then compare it.

images:bighand.jpg **Compose the recursive definition**:

## The recursive will constantly divide the range into halfs and compare the results and find the biggest value and return it, as it returns it will compare with the others until only one result is left.

## Directed Lab Work

### Maximum

*Complete the maximum application.*

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.*

1. 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.*

1. Please submit the java files along with this file.