Recursive Functions with Arrays

Iterative version

- Have an array/ArrayList called array. Want to find the sum of all the elements:

```
int sum = 0;
for (int i = 0; i < array.size(); i++)
   sum += array[i];
```

Recursive version

- Base case: What is the smallest size of an array for which we would ever want to add up all the elements?
- Recursive case:
  - Suppose you have an array with >1 element.
  - How can I describe finding the sum of all the elements as involving ***finding the sum of the elements of a smaller sized array?***
  - Hint: Suppose my array has 5 elements. My best friend knows how to find the largest value in an array, but only for 4 elements. How can I use him to solve my problem?

Recursive version

- sum(array)
- Base case: If array.size() == 1, return array[0]
- Recursive case: If array.size() > 1:
  - Compute the sum of all the elements in the sub-array from index 1 to the end ➔ sum(array[1:]) in Python syntax.
  - Add array[0] to this sum from above.
  - Return this value.

array = [7, 9, 8]

- sum(array)
- Base case: If array.size() == 1, return array[0]
- Recursive case: If array.size() > 1:
  - Find the sum of elements in array[1:] (whole array except A[0])
    - call it smallerSum
  - Return smallerSum + array[0]

Call sum([7, 9, 8])

array = [7, 9, 8]
smallerSum = sum([9, 8])

Call sum([9, 8])

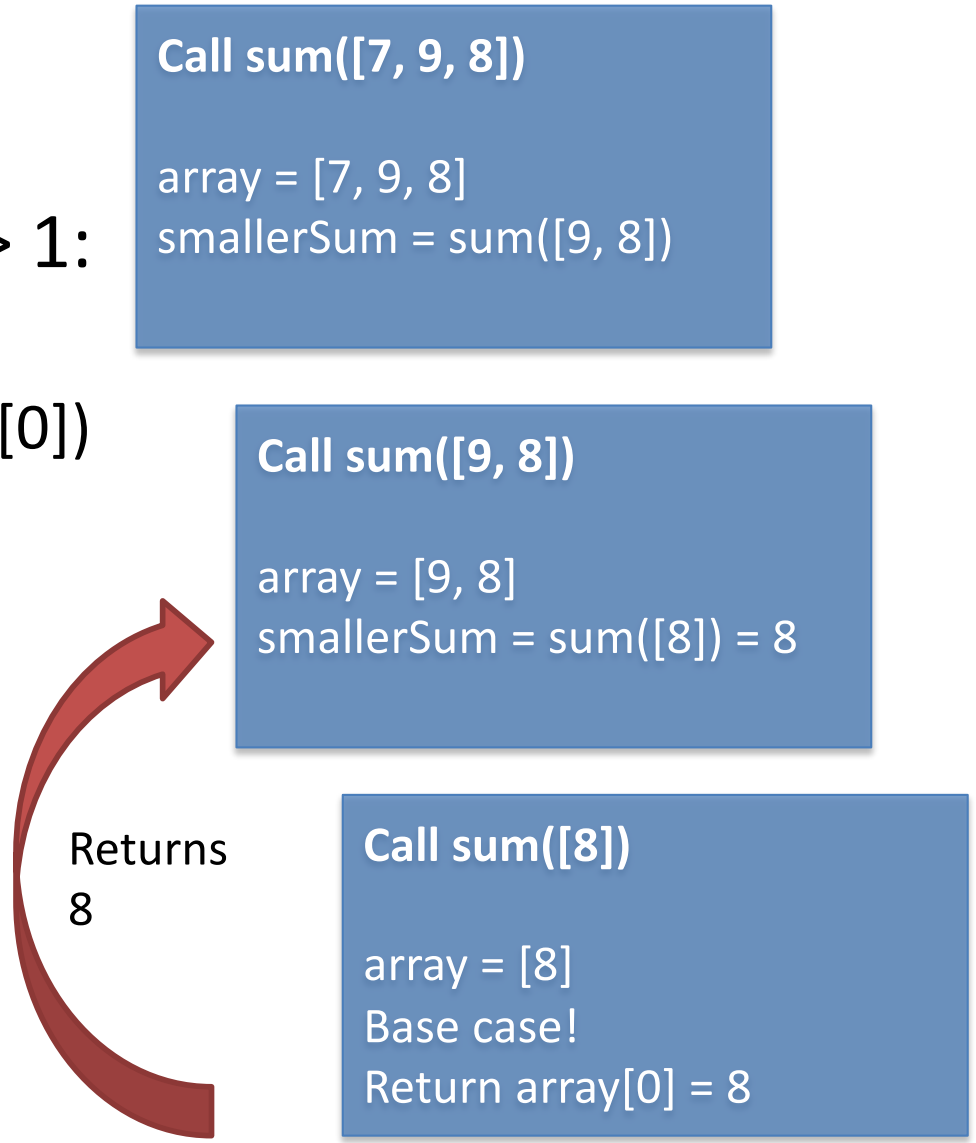
array = [9, 8]
smallerSum = sum([8])

Call sum([8])

array = [8]
Base case!

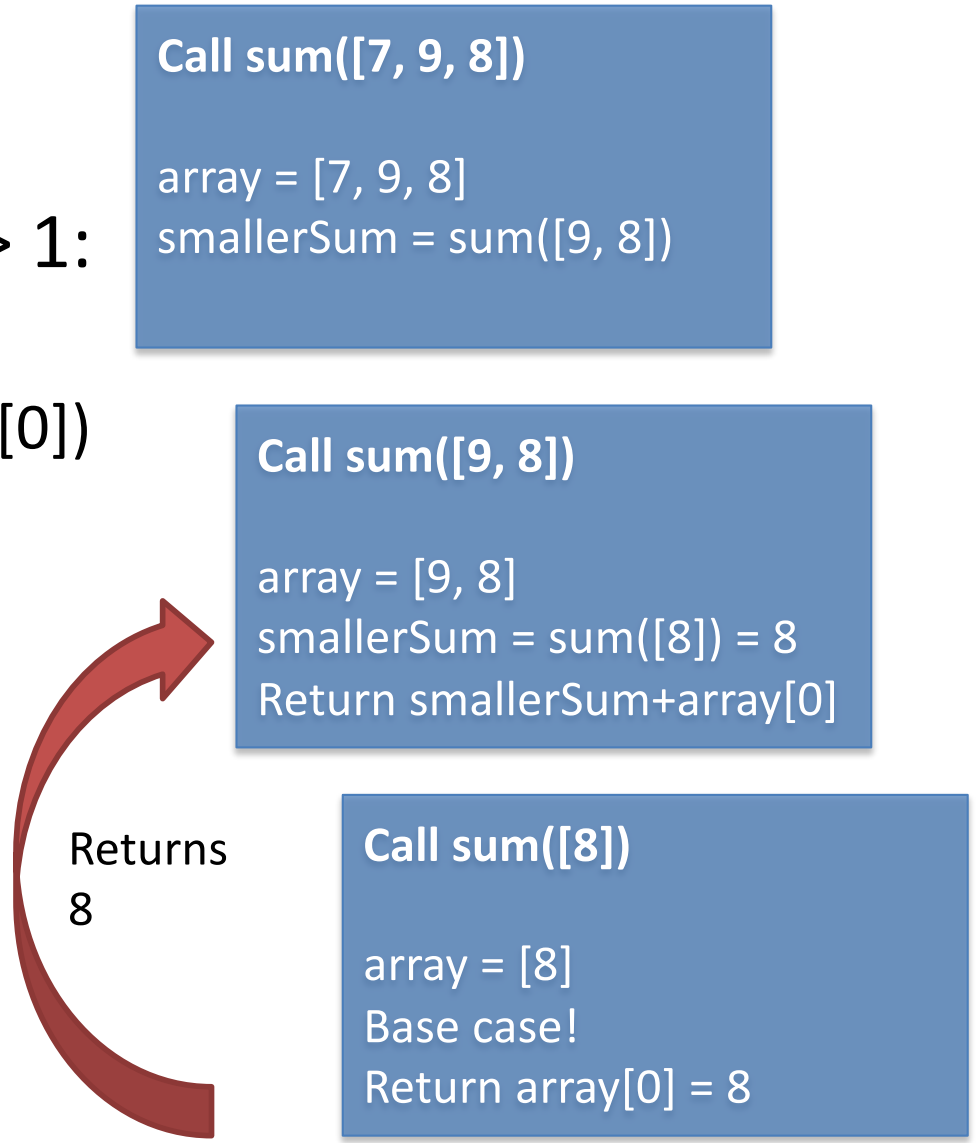
array = [7, 9, 8]

- sum(array)
- Base case: If array.size() == 1, return array[0]
- Recursive case: If array.size() > 1:
  - Find the sum of elements in array[1:] (whole array except A[0])
    - call it smallerSum
  - Return smallerSum + array[0]



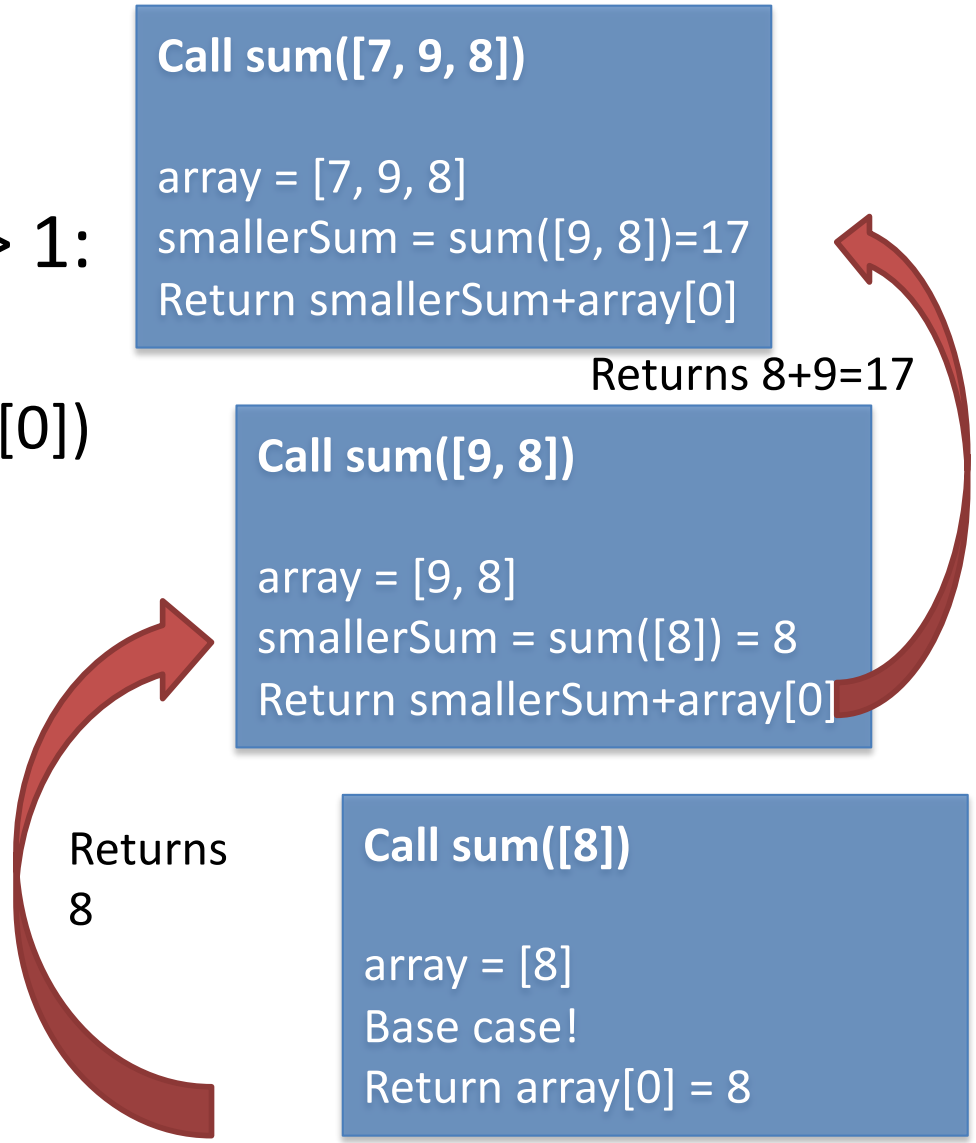
array = [7, 9, 8]

- sum(array)
- Base case: If array.size() == 1, return array[0]
- Recursive case: If array.size() > 1:
  - Find the sum of elements in array[1:] (whole array except A[0])
    - call it smallerSum
  - Return smallerSum + array[0]

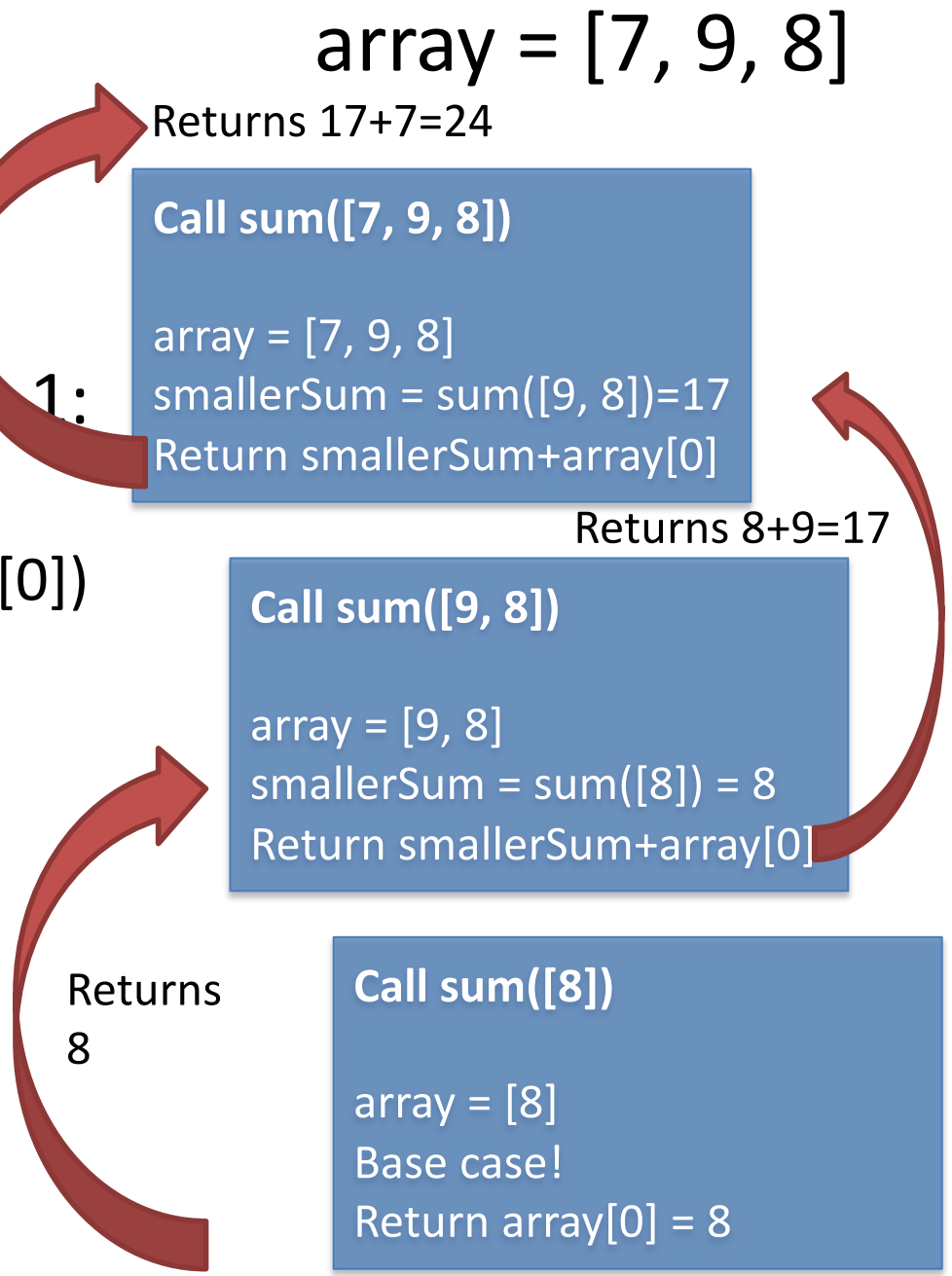


- sum(array)
- Base case: If array.size() == 1, return array[0]
- Recursive case: If array.size() > 1:
  - Find the sum of elements in array[1:] (whole array except A[0])
    - call it smallerSum
  - Return smallerSum + array[0]

array = [7, 9, 8]



- sum(array)
- Base case: If array.size() == 1 return array[0]
- Recursive case: If array.size() > 1:
  - Find the sum of elements in array[1:] (whole array except A[0])
    - call it smallerSum
  - Return smallerSum + array[0]



Java recursive version

- Java does let you take slices of arrays like Python, it involves using some techniques we haven't learned yet, so we're going to see a different way.
- Notice that our slices always involving chopping off the first element in the array; i.e, A[0]
  - [7, 9, 8] -> [9, 8] -> [8]
- How can we simulate an array slice without actually doing the slicing?
  - Hint: In the olden days, people used these things called "bookmarks" to hold their spot in a book while they were reading. We can use the same idea to mark the section of the array that we are interested in recursing on.

Java recursive version

- Use an integer variable "bookmark" to save your spot in the array.
- When we make a recursive call, instead of passing an updated array (like the Python version), we will pass an updated bookmark.
- Our function will now be sum(array, leftIdx)
  - leftIdx ("left index") represents the index of the bookmark in the array: everything before the bookmark is already read, everything afterwards is unread. So it is the leftmost index of the portion of the array we have still left to read.

Recursive Java version

- sum(A, leftIdx)
- Base case: ???
- Recursive case:
  - Find the sum of elements in ???
    - call it smallerSum
  - return ???

- Where does the bookmark start?

Recursive Java version

- sum(A, leftIdx)
- Base case: if leftIdx == array.size() - 1
- Recursive case:
  - Find the sum of elements in everything after array[leftIdx]
    - smallerSum = sum(array, leftIdx + 1)
  - return smallerSum + array[0]

- Initial call should be sum(array, 0)

- How can we use this idea to find the largest element in an array/arraylist? (the max element)?
- On paper, write out a recursive formulation for this.
  - What is the base case? (What is the smallest size of an array we would want to take the max of?)
  - What is the recursive case? For a bigger array, how do we find the max element by reducing the problem to a smaller version of the same problem?
  - Hint: For an array of size >= 2, suppose a friend tells you they have already computed the largest element in the sub-array from index 1 to the end. How can you use this information to help you find the overall largest element?
- Code this in Java!