Lecture 9.5:

Recursion

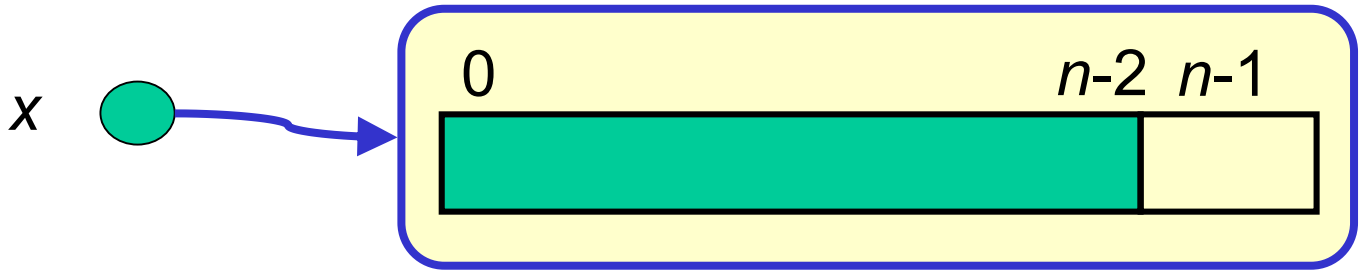
Prof. Shervin Shirmohammadi
University of Ottawa

Recursion

- Recursion is a problem-solving technique which uses smaller sub-problems of problem P
- In recursion the sub-problems are similar to problem P, but are simpler versions of it.
- When the sub-problem is small enough (the base case), the sub-problem is solved directly.
- If the sub-problem is large, the problem is reformulated in terms of a sub-problem with a smaller parameter.
  - The solution to the larger problem is found using the solution of the smaller sub-problem.
  - If the parameter of the sub-problem is still to large, then it must be reduced until we reach the base case.
- The problem and sub-problems are solved using multiple executions of a single subprogram (algorithm/method) which calls itself
  - Each execution of the subprogram can be viewed as an instance of its execution.

Example 1: basic idea

- What is the maximum element in array *x*?

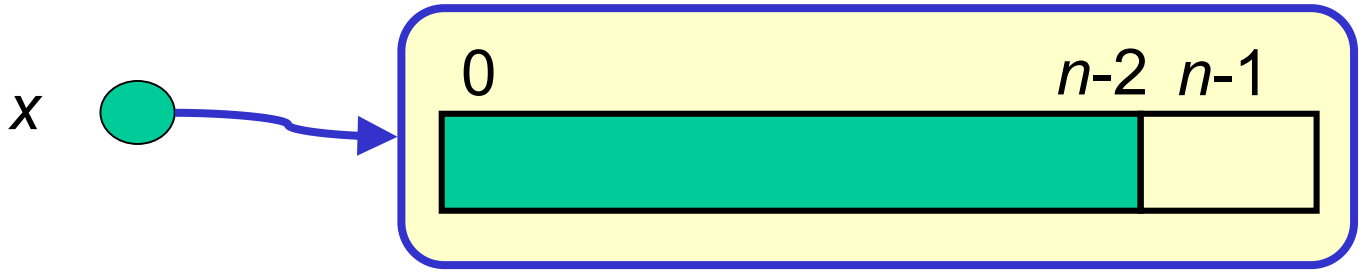


- Let's assume the maximum element in the shaded area is *m*.

- Answer: the maximum of *m* and the value at index *n*-1

Example 2 (basic idea)

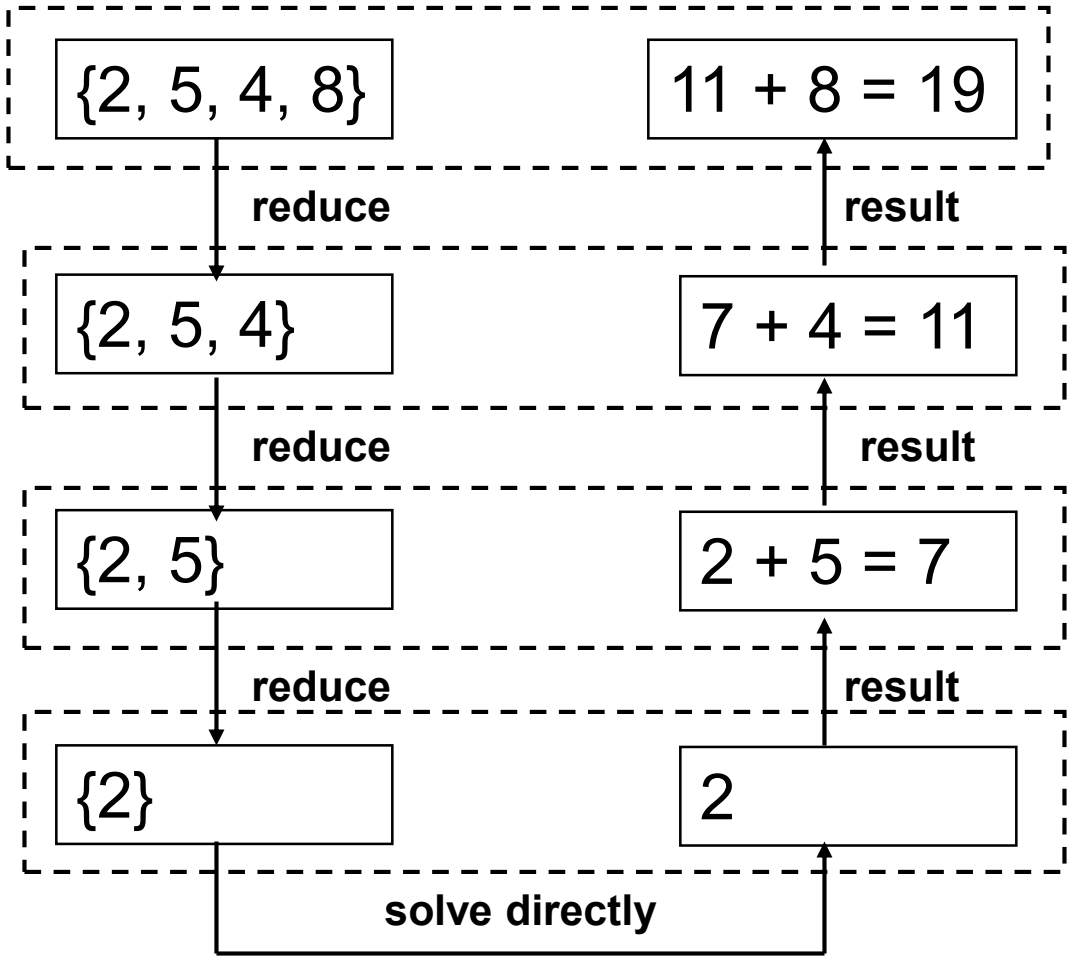
- What is the sum of all elements in array *x*?



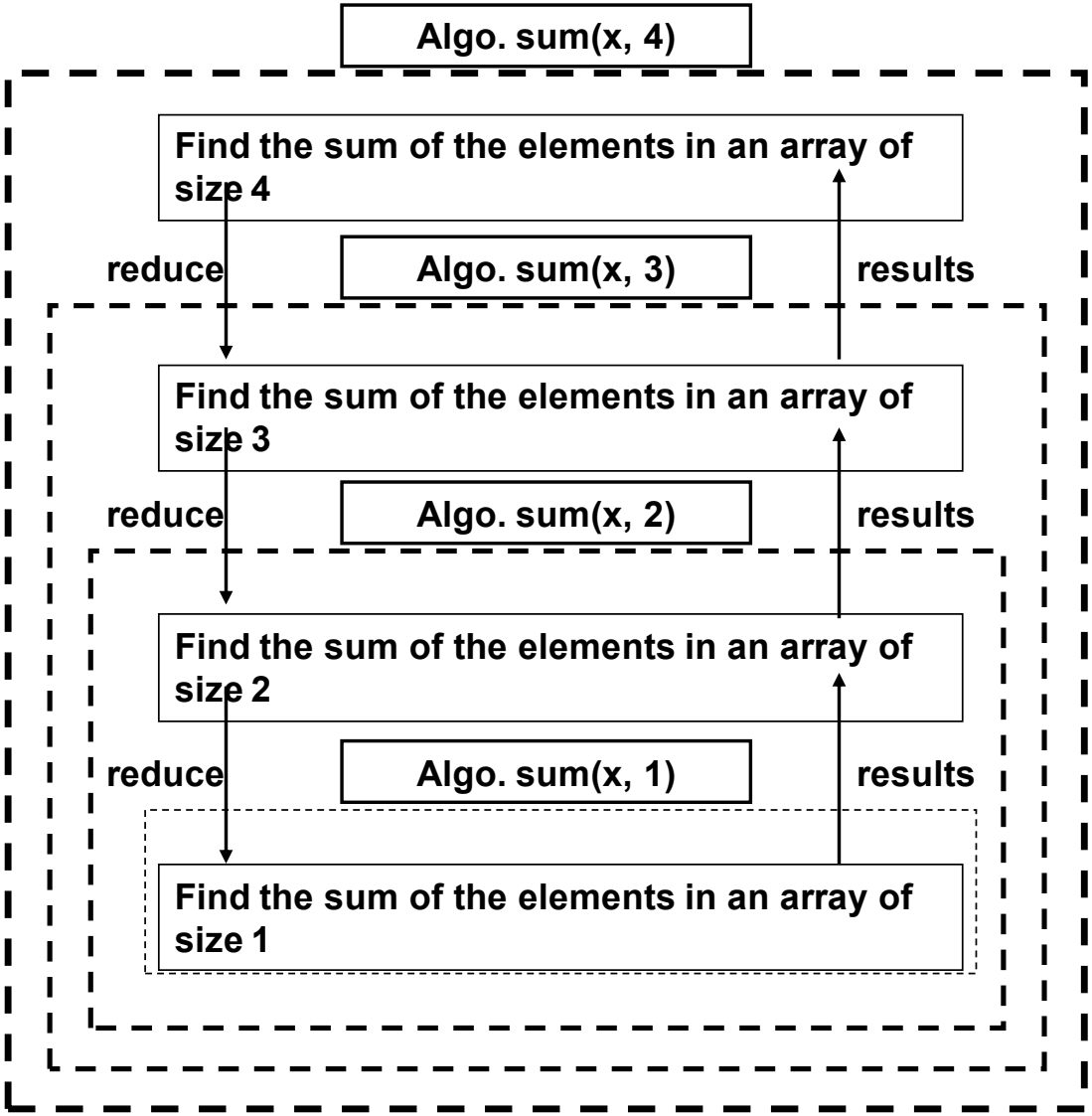
- Assume the sum of all values in the shaded area is *s*.

- Answer: The sum is *s* + the value at index *n*-1

Example for x = {2, 5, 4, 8}



Recursive Calls

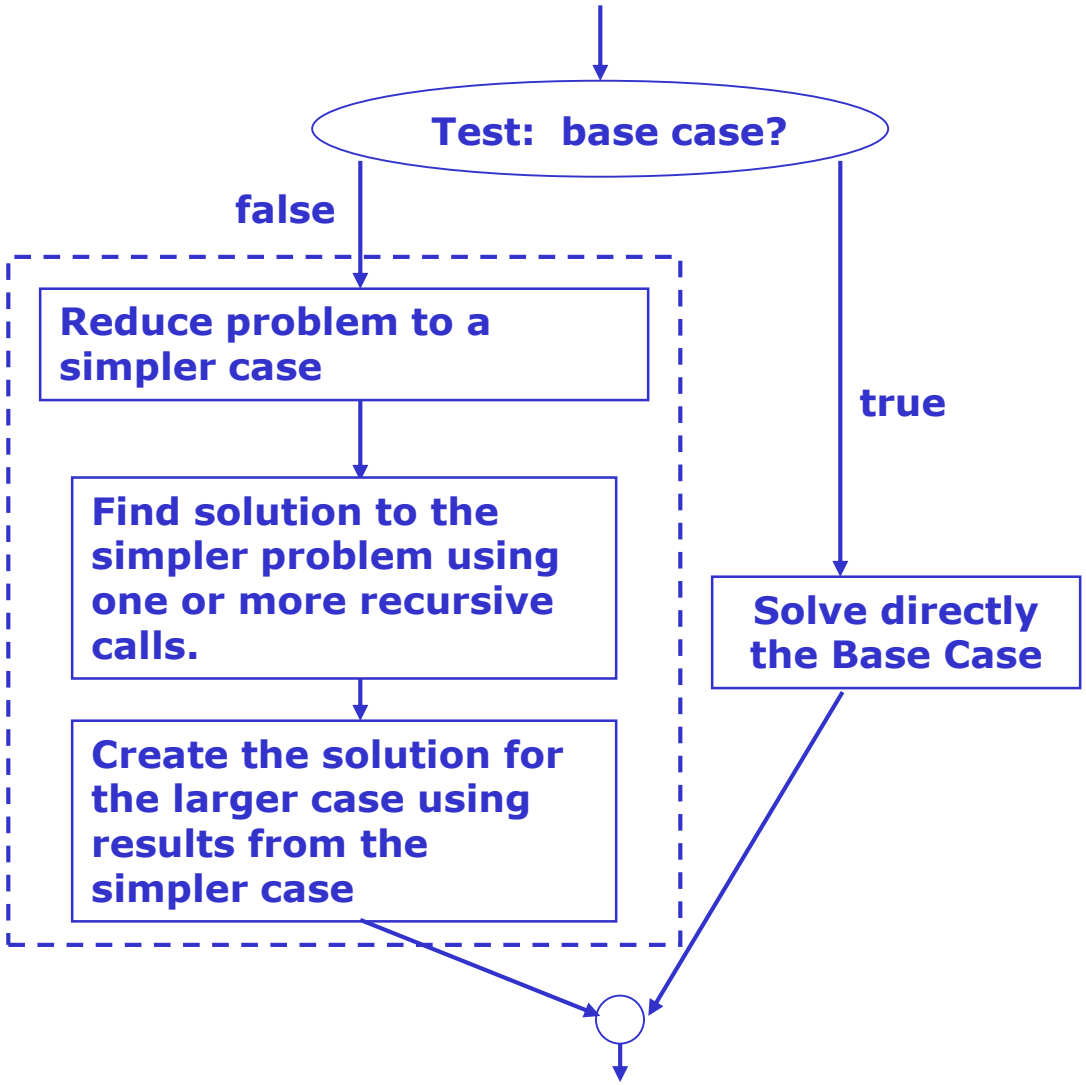


Components of Recursion

There are 3 components to recursion:

1. A test to see if the problem is simple enough to solve directly (i.e. non-recursively): the "base case"
2. The solution for the base case.
3. A solution to the problem which involves solving one (or more) smaller versions of the same problem.

Template for recursive algorithms



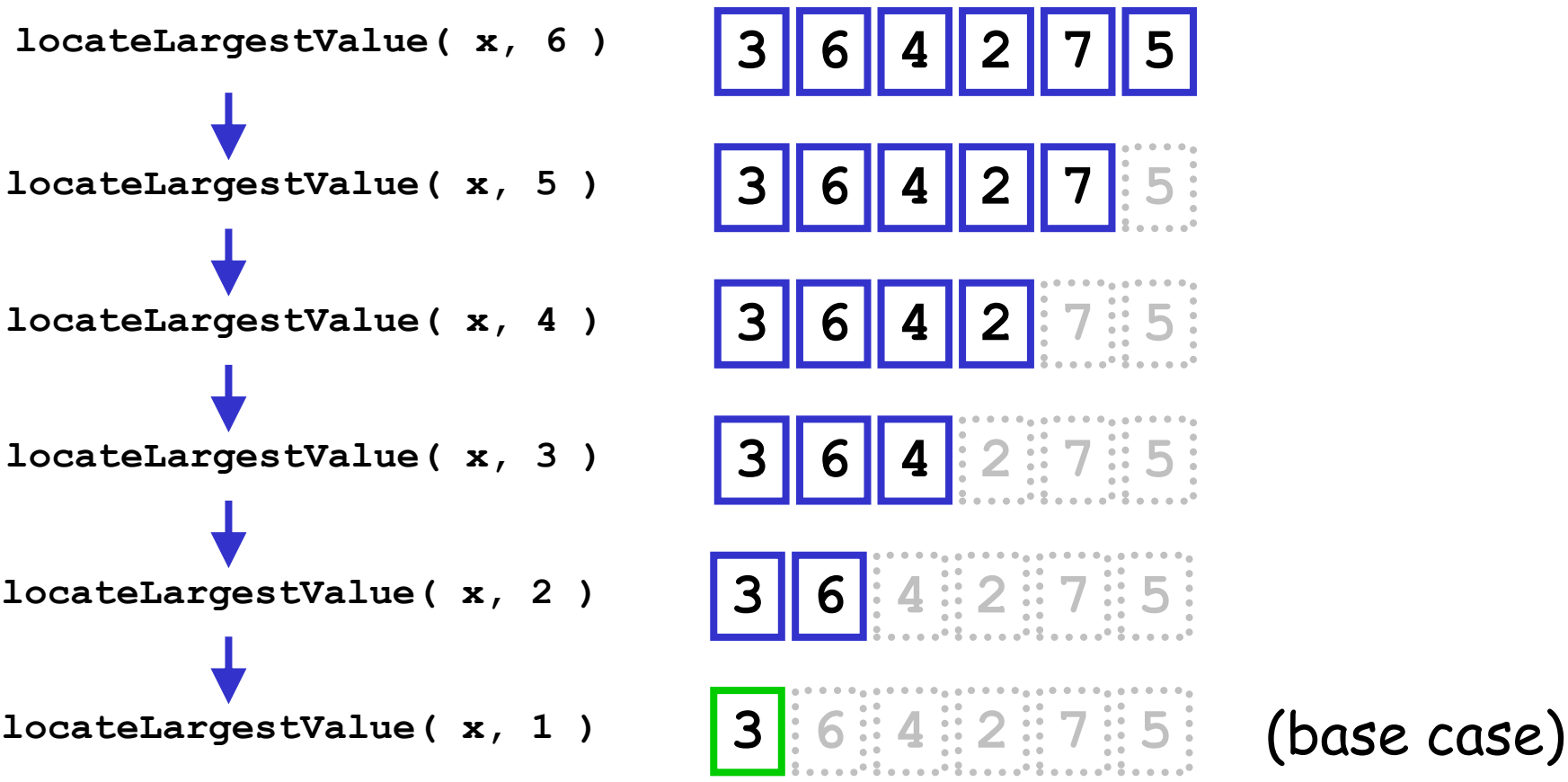
Exercise 1

- Write a recursive algorithm to find the factorial of integer x.

Exercise 2

- Write a recursive algorithm to find the largest element in array x of size n.

Recursive location of largest value



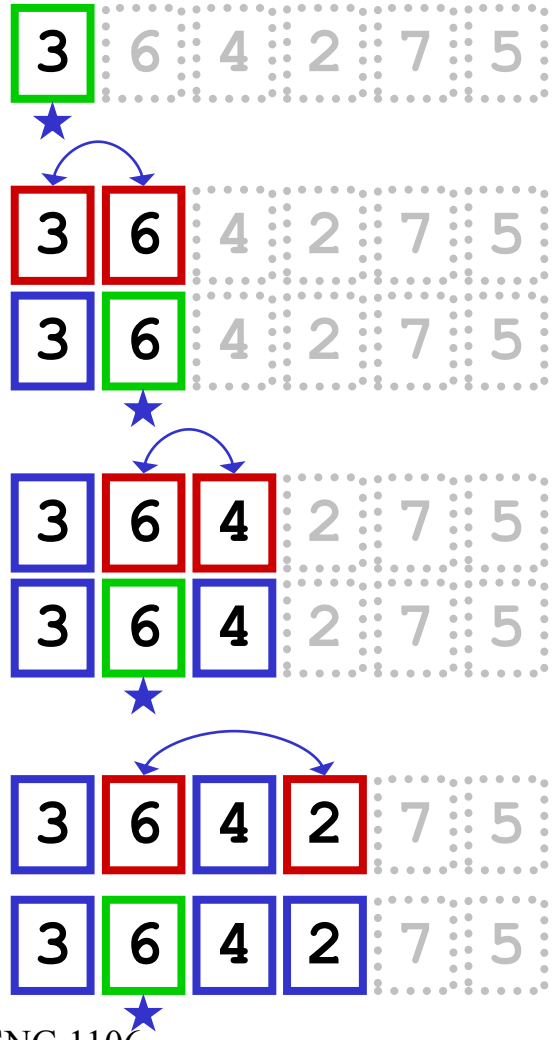
Exercise 8-8: Recursive location of largest value

repeated from previous page
`locateLargestValue( x, 1 )`

return to
`locateLargestValue( x, 2 )`

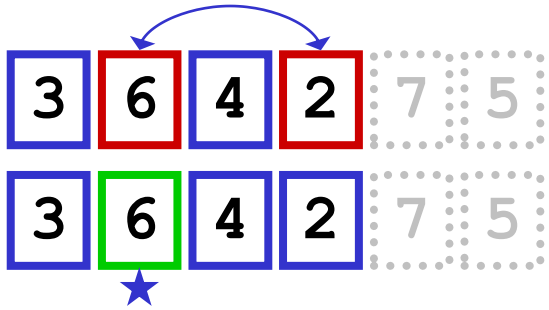
return to
`locateLargestValue( x, 3 )`

return to
`locateLargestValue( x, 4 )`



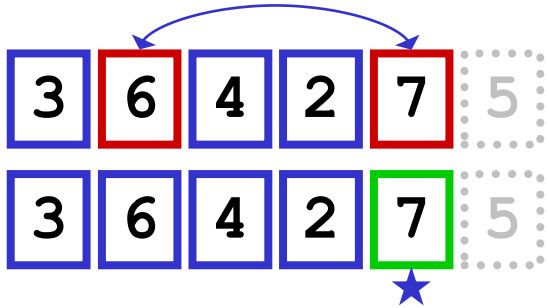
Exercise 8-8: Recursive location of largest value

repeated from previous page
`locateLargestValue( x, 4 )`



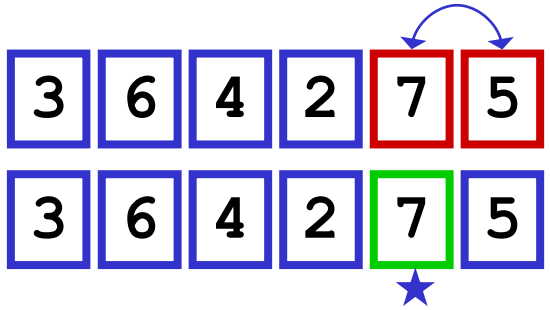
(compare)
(choose)

return to
`locateLargestValue( x, 5 )`



(compare)
(choose)

return to
`locateLargestValue( x, 6 )`



(compare)
(choose)

done!

When does the problem actually get solved?

- In the previous example, we needed the result of a recursive call first.
  - Therefore, recursive calls were made without doing any "work" until the base case was reached.
  - Only after the recursive calls start returning, the actual comparisons/operations are done, and the problem is solved.
    - This is not always the case. Sometimes some "work" is done before the recursive call, and when we reach the base case, the problem is solved.
    - e.g., certain sort algorithms.

Static Variables

- A static variable inside a function keeps its value between invocations; e.g.

```
#include <stdio.h>
int main()
{
    int i;
    for (i = 0; i < 5; ++i)
        doStatic();
}

void doStatic()
{
    int x = 1;
    static int y = 1;
    printf("\nX is %d and Y is %d", x, y);
    x++; y++;
}
```