Lab 09: Divide and Conquer

CS 0445: Data Structures

TAs: Jon Rutkauskas
Brian Nixon
http://db.cs.pitt.edu/courses/cs0445/current.term/

November 11, 2019
University of Pittsburgh, Pittsburgh, PA

Review – requirements for recursion

- Recursive case(s)
  - Calling the algorithm again with a structurally similar subproblem.
- Base case(s)
  - A case for which an answer is known and is returned without a recursive call.
- Termination
  - All recursive cases must eventually lead to base cases.

Example problem: searching a sorted array

`int[] array =`

15	25	25	59	72	89	203	576

array is already *sorted*

```
public boolean searchArray(int[] array, int value) {
       // ?
}

searchArray(array, 203);
```

A “Recursive” solution – Tail Recursion

```
int[] array =
```

15	25	25	59	72	89	203	576

```
public boolean searchArray(int[] array, int value)
{
   return searchArray(array, value, 0);
}
private boolean searchArray(int[] array, int value, int index)
{
      if(index >= array.length)
            return false;
      if(array[index] == value)
            return true;
      return searchArray(array, value, index + 1);
}

searchArray(array, 203);
```

A “Recursive” solution – Tail Recursion

```
int[] array =
```

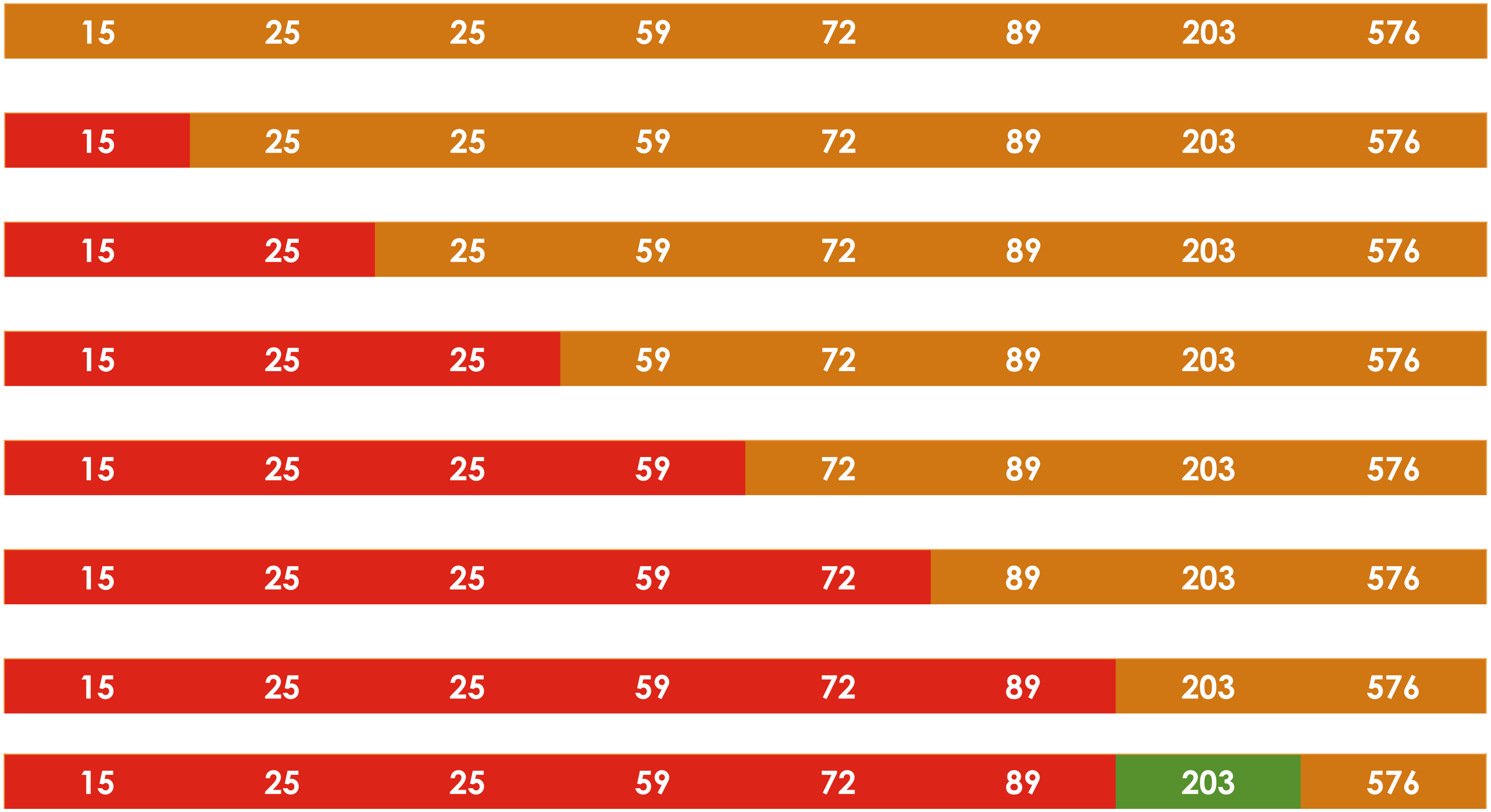
15	25	25	59	72	89	203	576

```
public boolean searchArray(int[] array, int value)
{
   return searchArray(array, value, 0);
}
private boolean searchArray(int[] array, int value, int index)
{
      if(index >= array.length)
            return false;
      if(array[index] == value)
            return true;
      return searchArray(array, value, index + 1);
}

searchArray(array, 203);
```

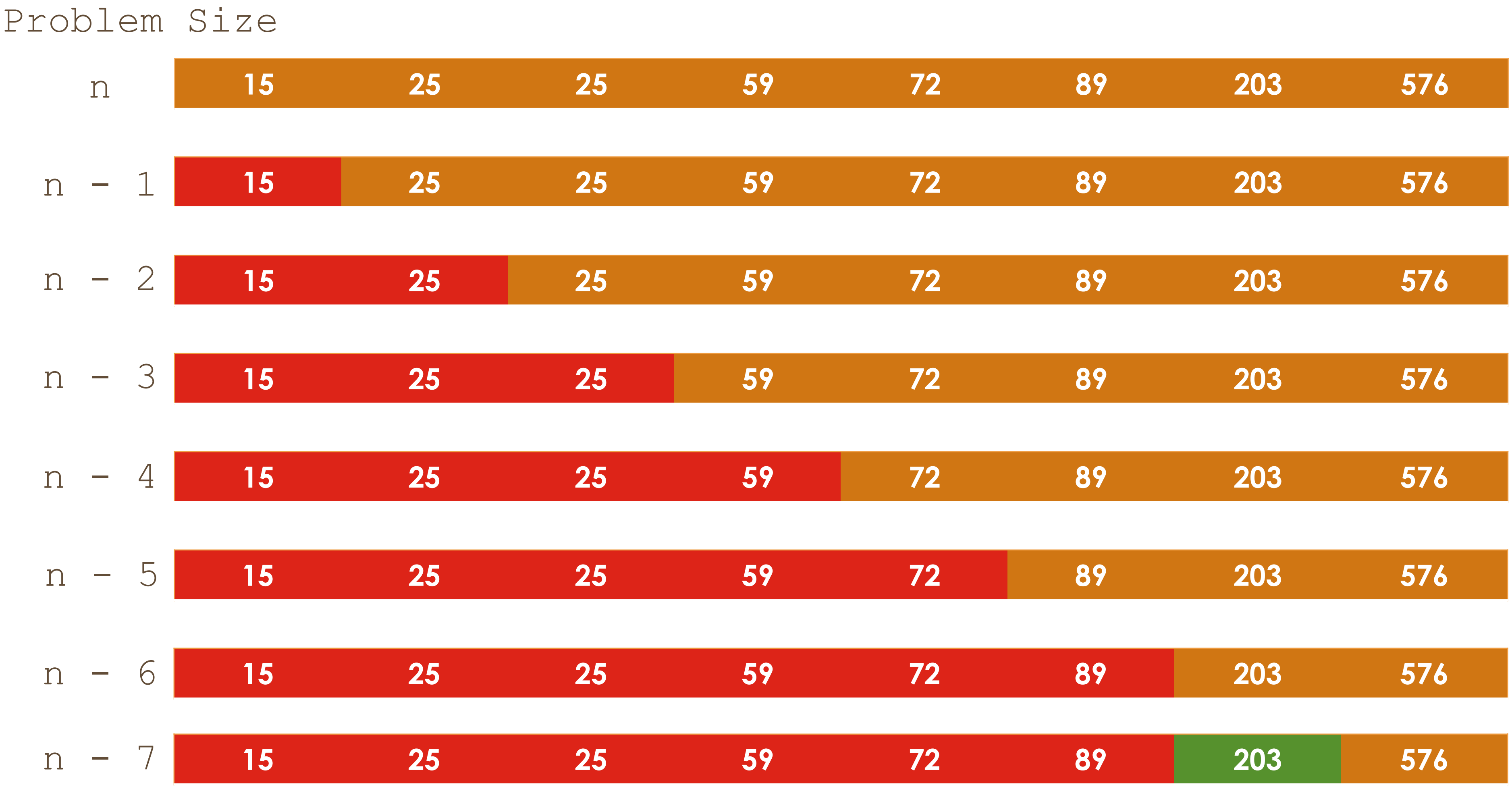
Look for the recursive case, base cases, and termination

A “Recursive” solution



15	25	25	59	72	89	203	576
15	25	25	59	72	89	203	576
15	25	25	59	72	89	203	576
15	25	25	59	72	89	203	576
15	25	25	59	72	89	203	576
15	25	25	59	72	89	203	576
15	25	25	59	72	89	203	576
15	25	25	59	72	89	203	576

A “Recursive” solution



Problem Size								
n	15	25	25	59	72	89	203	576
n – 1	15	25	25	59	72	89	203	576
n – 2	15	25	25	59	72	89	203	576
n – 3	15	25	25	59	72	89	203	576
n – 4	15	25	25	59	72	89	203	576
n – 5	15	25	25	59	72	89	203	576
n – 6	15	25	25	59	72	89	203	576
n – 7	15	25	25	59	72	89	203	576

Tail Recursion

- That was not a divide and conquer solution
- Only reduced our problem size by **one** each time
- There is a more efficient way of solving this
- Think: because the array is sorted, if we find a number smaller than the value we are searching for, we know that value cannot be at any lower index
  - This has the potential to shrink our problem size by more than just one.

New requirement for ‘divide and conquer’ recursion

- Recursive case(s)
  - Calling the algorithm again with a structurally similar subproblem.
  - For divide and conquer, the recursive subproblem must be a **fraction of the size of the original**.
- Base case(s)
  - A case for which an answer is known and is returned without a recursive call.
- Termination
  - All recursive cases must eventually lead to base cases.

Divide and Conquer Solution

```
int[] array =
```

15	25	25	59	72	89	203	576

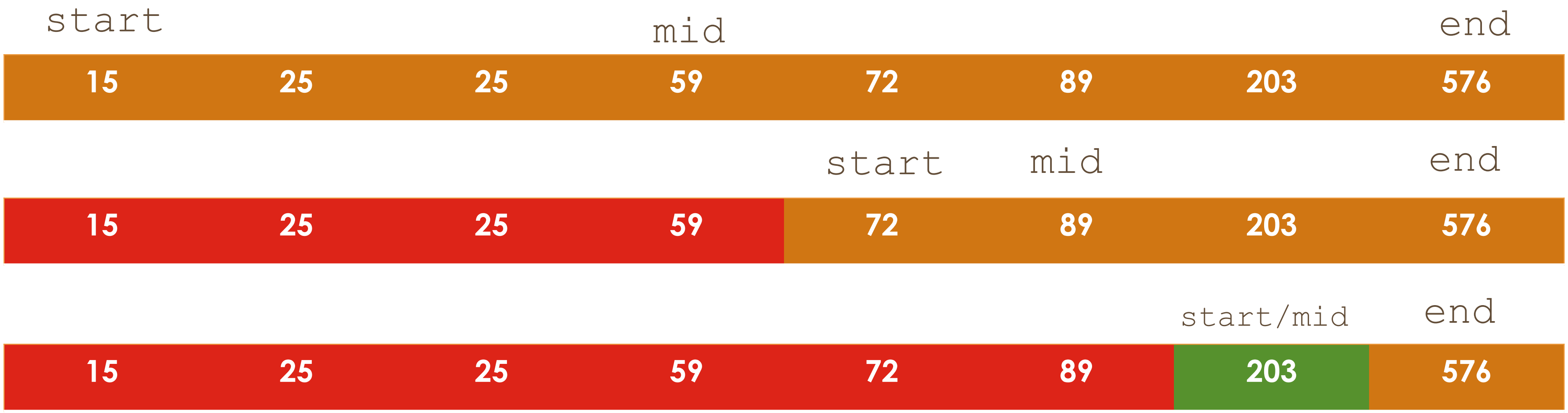
```java
public boolean binarySearch(int[] array, int value) {
    return binarySearch(array, value, 0, array.length - 1);
}

private boolean binarySearch(int[] array, int value, int start, int end) {
    if(end < start)
        return false;

    int mid = (start + end) / 2;

    if(value == array[mid])
        return true;
    else if(value < array[mid])
        return binarySearch(array, value, start, mid - 1);
    else //value > array[mid]
        return binarySearch(array, value, mid + 1, end);
}
```

Divide and Conquer Solution



start mid end

15 25 25 59 72 89 203 576

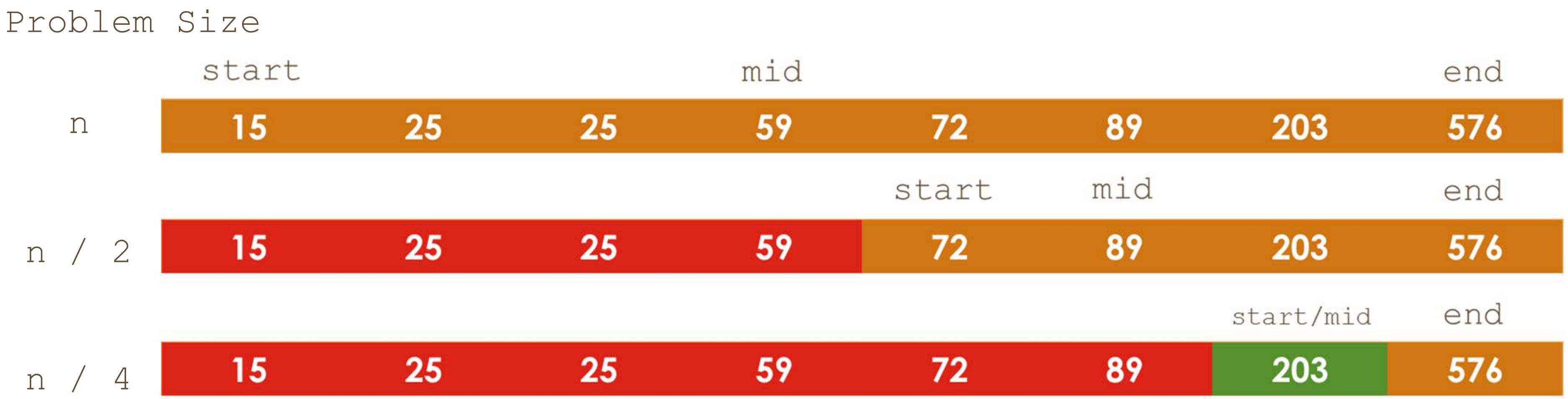
start mid end

15 25 25 59 72 89 203 576

start/mid end

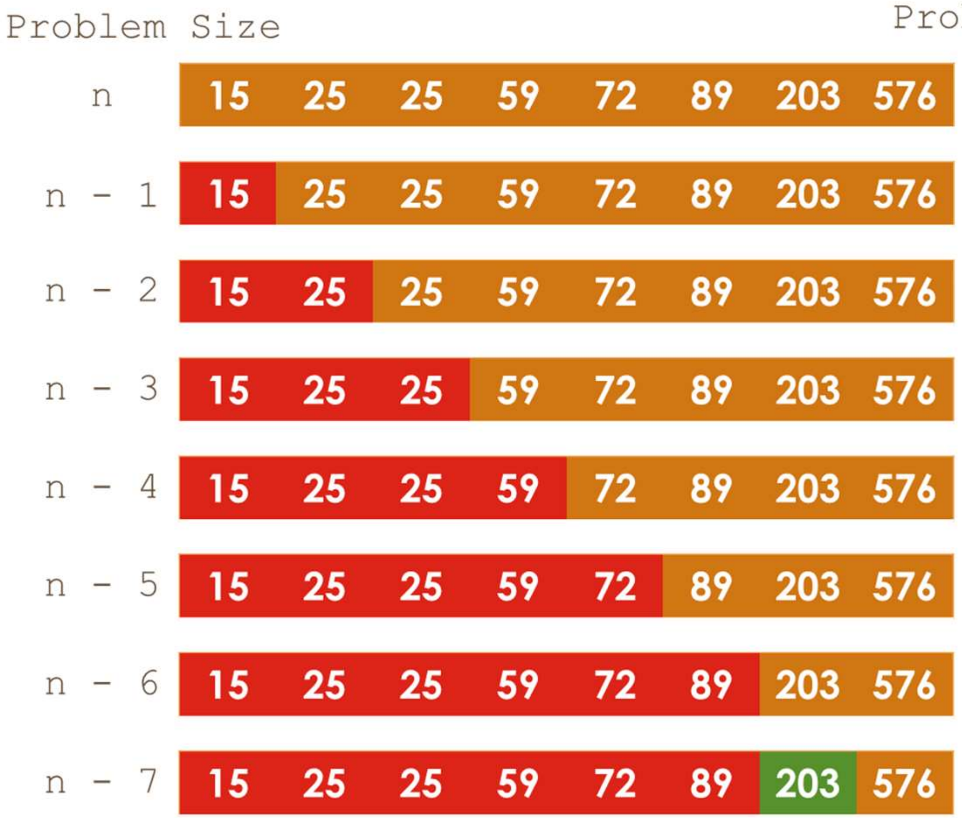
15 25 25 59 72 89 203 576

Divide and Conquer Solution



Problem Size								
	start			mid				end
n	15	25	25	59	72	89	203	576
					start	mid		end
n / 2	15	25	25	59	72	89	203	576
							start/mid	end
n / 4	15	25	25	59	72	89	203	576

Tail Recursion



Problem Size

n 15 25 25 59 72 89 203 576

n - 1 15 25 25 59 72 89 203 576

n - 2 15 25 25 59 72 89 203 576

n - 3 15 25 25 59 72 89 203 576

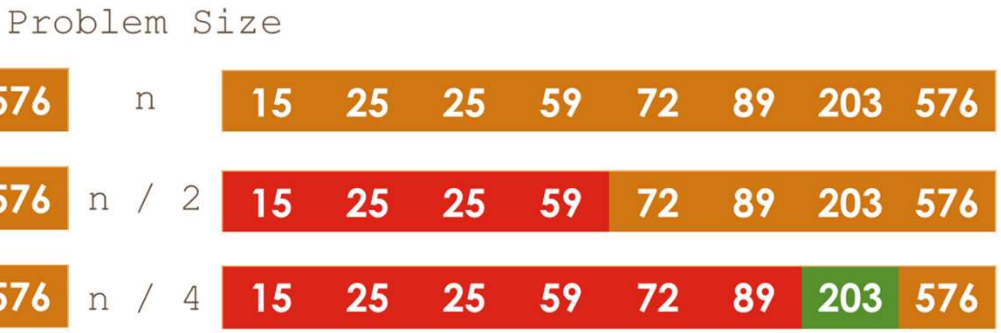
n - 4 15 25 25 59 72 89 203 576

n - 5 15 25 25 59 72 89 203 576

n - 6 15 25 25 59 72 89 203 576

n - 7 15 25 25 59 72 89 203 576

Divide and Conquer



Problem Size

n 15 25 25 59 72 89 203 576

n / 2 15 25 25 59 72 89 203 576

n / 4 15 25 25 59 72 89 203 576

Example problem #2 – sorting an array

```
int[] array =
```

576	72	25	203	25	59	89	15

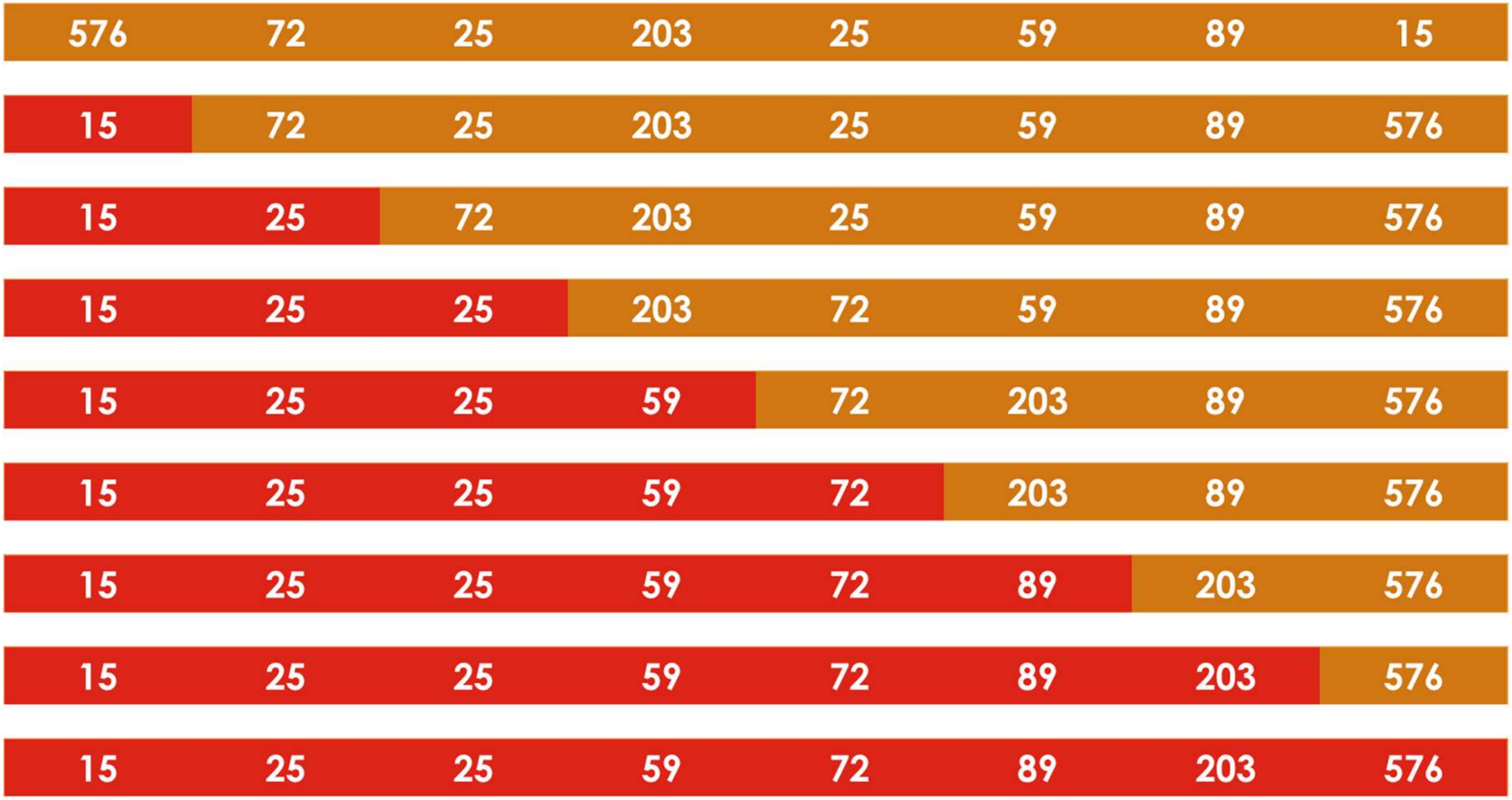
Recursive Selection Sort

`int[] array =`

576	72	25	203	25	59	89	15

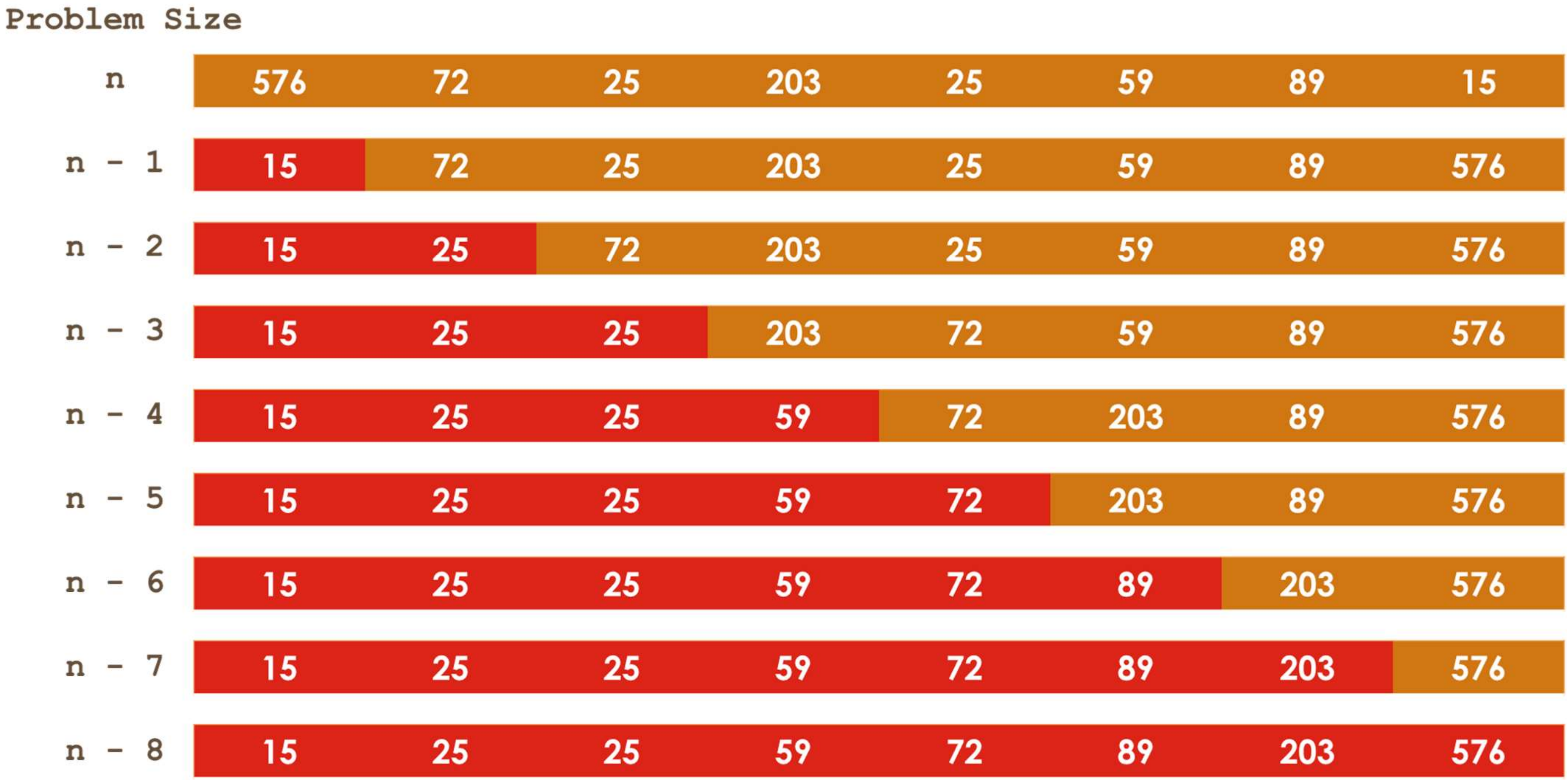
```
public void selectionSort(int[] array) {
    selectionSort(array, 0, array.length - 1);
}
public void selectionSort(int[] array, int first, in last) {
    if(first < last) {
        int indexOfSmallest = getIndexOfSmallest(array, first, last);
        swap(array, first, indexOfSmallest);
        selectionSort(array, fist + 1, last);
    }
}
```

Recursive Selection Sort



576	72	25	203	25	59	89	15
15	72	25	203	25	59	89	576
15	25	72	203	25	59	89	576
15	25	25	203	72	59	89	576
15	25	25	59	72	203	89	576
15	25	25	59	72	203	89	576
15	25	25	59	72	89	203	576
15	25	25	59	72	89	203	576
15	25	25	59	72	89	203	576

Recursive Selection Sort



Problem Size								
n	576	72	25	203	25	59	89	15
n - 1	15	72	25	203	25	59	89	576
n - 2	15	25	72	203	25	59	89	576
n - 3	15	25	25	203	72	59	89	576
n - 4	15	25	25	59	72	203	89	576
n - 5	15	25	25	59	72	203	89	576
n - 6	15	25	25	59	72	89	203	576
n - 7	15	25	25	59	72	89	203	576
n - 8	15	25	25	59	72	89	203	576

Merge Sort – Another Divide and Conquer Approach

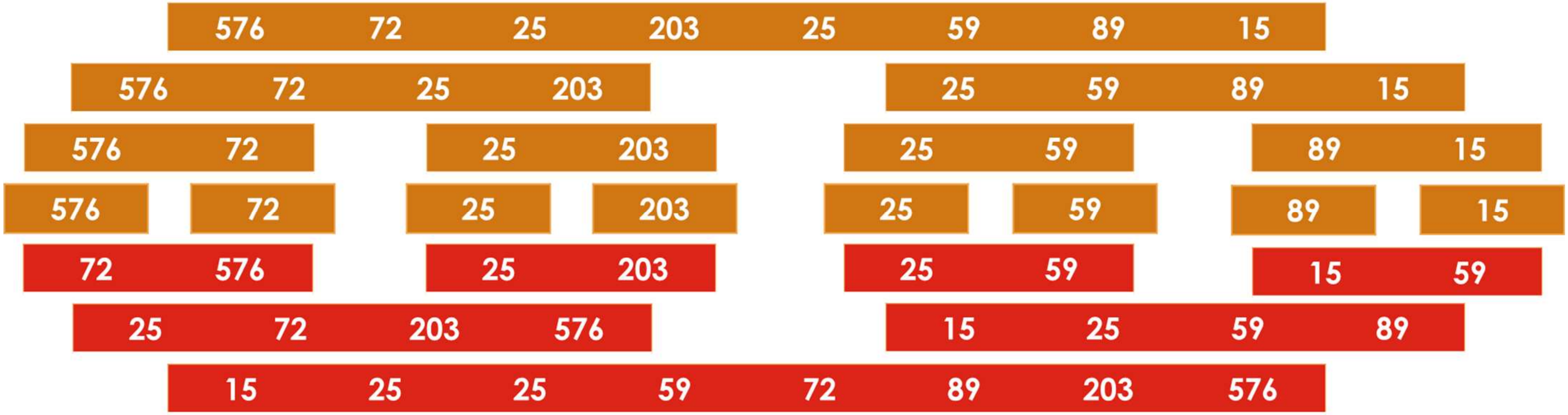
int[] array =

576	72	25	203	25	59	89	15

```
public void mergeSort(int[] array) {
    int[] tempArray = new int[array.length];
    mergeSort(array, tempArray, 0, array.length – 1);
}

public void mergeSort(int[] array, int[] tempArray, int first, in last) {
    if(first < last) {
        int mid = (first + last) / 2
        mergeSort(array, tempArray, first, mid);     //Merge sort left
        mergeSort(array, tempArray, mid + 1, last);  //Merge sort right
        merge(a, tempArray, first, mid, last);       //Combine the solutions
                                                  // (the actual work)
    }
}
```

Merge Sort



576 72 25 203 25 59 89 15

576 72 25 203 | 25 59 89 15

576 72 | 25 203 | 25 59 | 89 15

576 | 72 | 25 | 203 | 25 | 59 | 89 | 15

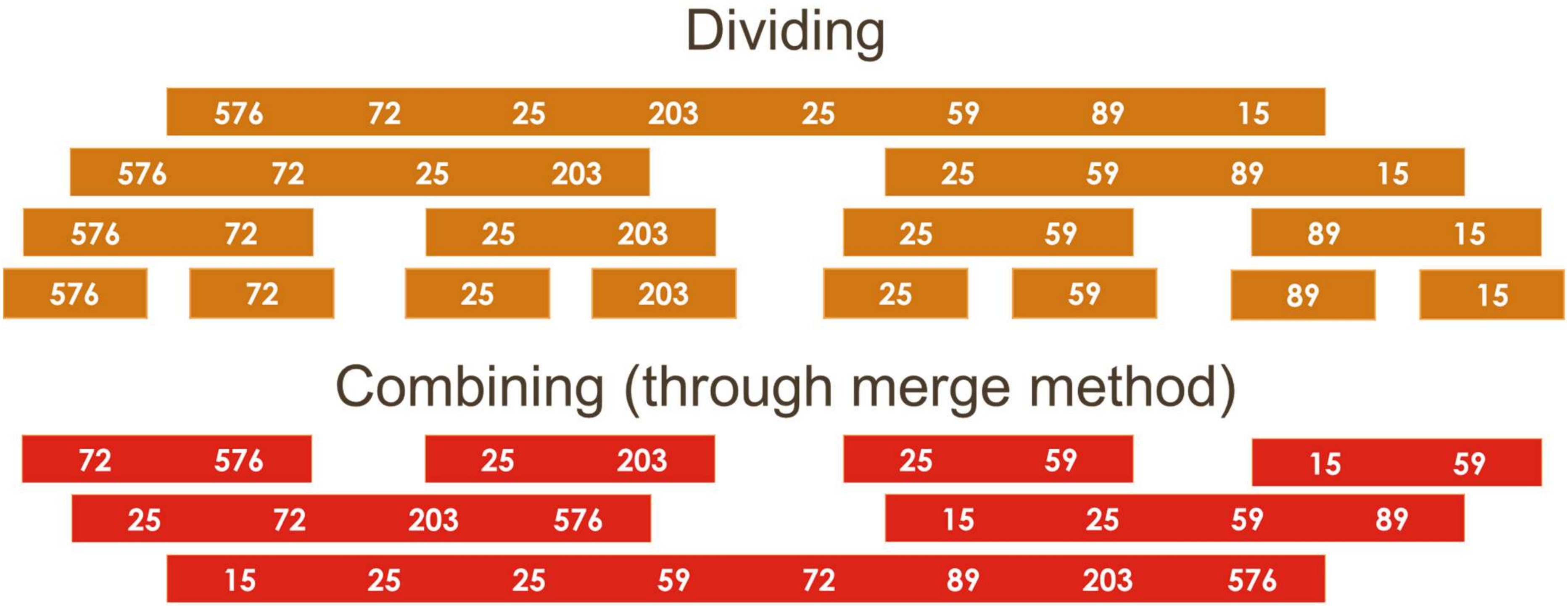
72 576 | 25 203 | 25 59 | 15 59

25 72 203 576 | 15 25 59 89

15 25 25 59 72 89 203 576

- Each recursive call, the problem is broken in half
- Once a base case is reached, the halves are recombined

Merge Sort



Dividing

576 72 25 203 25 59 89 15

576 72 25 203 | 25 59 89 15

576 72 | 25 203 | 25 59 | 89 15

576 | 72 | 25 | 203 | 25 | 59 | 89 | 15

Combining (through merge method)

72 576 | 25 203 | 25 59 | 15 59

25 72 203 576 | 15 25 59 89

15 25 25 59 72 89 203 576

- In this solution, the work solving the problem is done when combining the solutions to the subproblems

Your Tasks

- Download the Lab 9 instructions and Provided Code from the course website
  - http://db.cs.pitt.edu/courses/cs0445/current.term/
- Devise a divide and conquer solution to the methods you implemented in the tail recursion lab:
  - `static <T> void reverse(T[] a)`
  - `static String replace(String str, char before, char after)`
- Test your work!