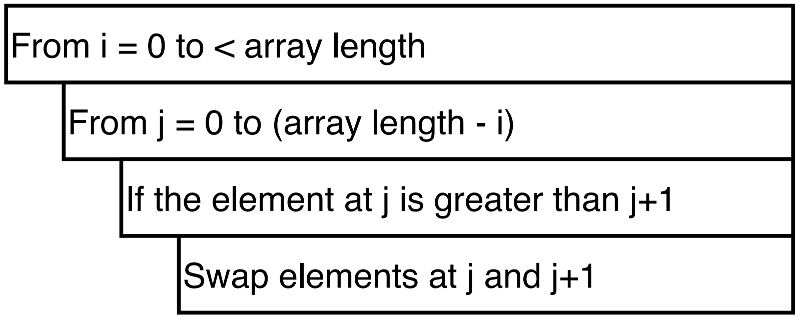
Name	Worst Case Runtime	Difficulty
BubbleSort	n^2	easiest
SelectionSort	n^2	easier
MergeSort	n*log(n)	medium

-30	0	5	10	97



```
From i = 0 to < array length
    From j = 0 to (array length - i)
        If the element at j is greater than j+1
            Swap elements at j and j+1
```

I = 1

J = 2

indexofMin = 1

-30	0	5	10	97

```
From i = 0 to < array length
    Assume the element at 'i' is the least in the array,
    assign i to 'indexOfMin'
    For j from i + 1 to end of array
        See if there is an element with lower value
            If there is, record its index
    If the index of the current element and the index
    of the 'lowest' element is not the same, swap em
```

5

Splitter

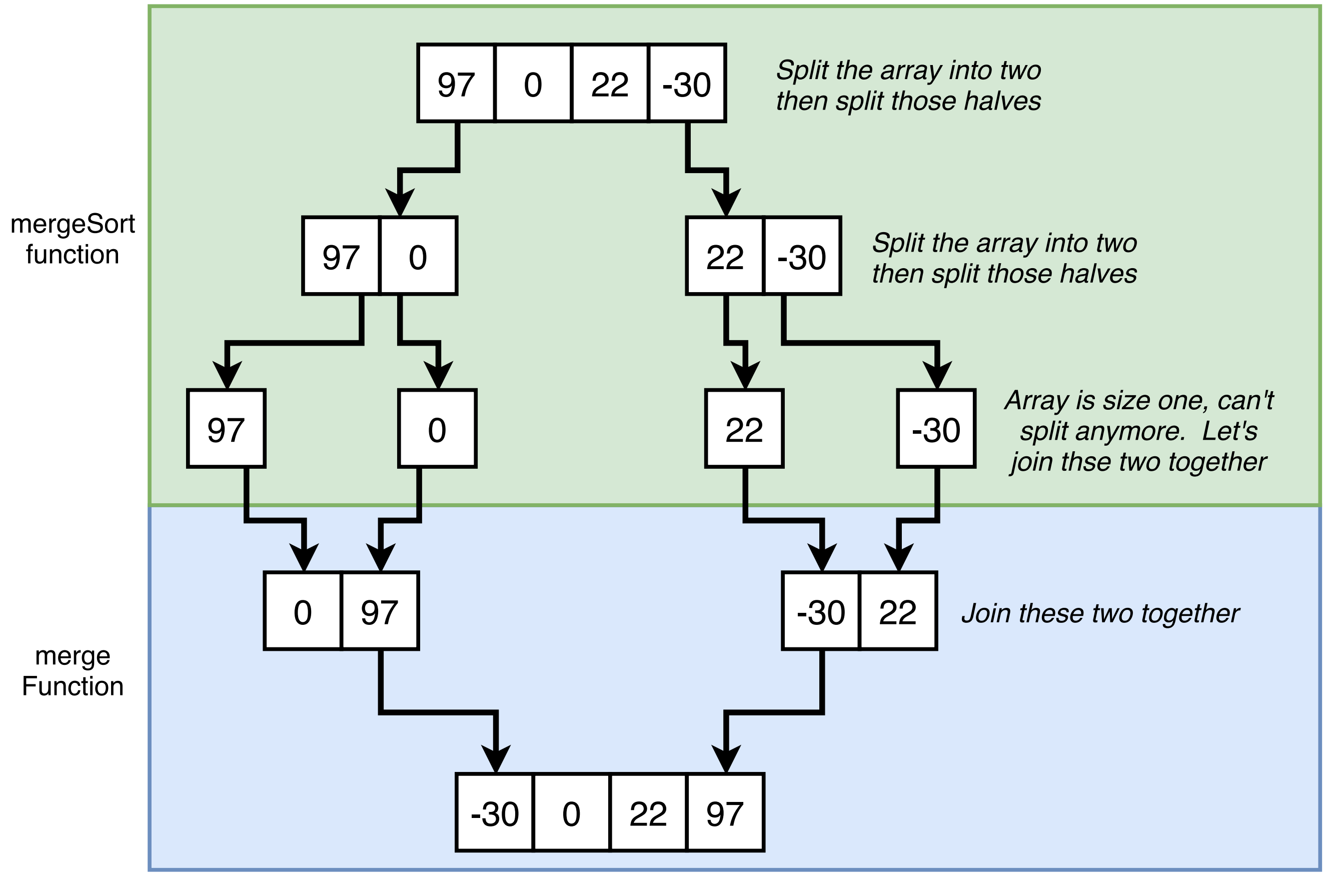
```
If the incoming array *doesn't* have length 1
    Divide the incoming array into two halves
    Call the 'splitter' again with each half
    Call the 'merger' with each half
```

Merger

```
Create 'results' array
While there are still elements in *both* arrays
    If the first element the left half is less than first in right half
        'shift' the element from left into a 'result' arr
    else
        'shift' the element from right into a 'result' arr
Take everything from the array that still has stuff in it and put it in results
```

-30	0	22	97

Results

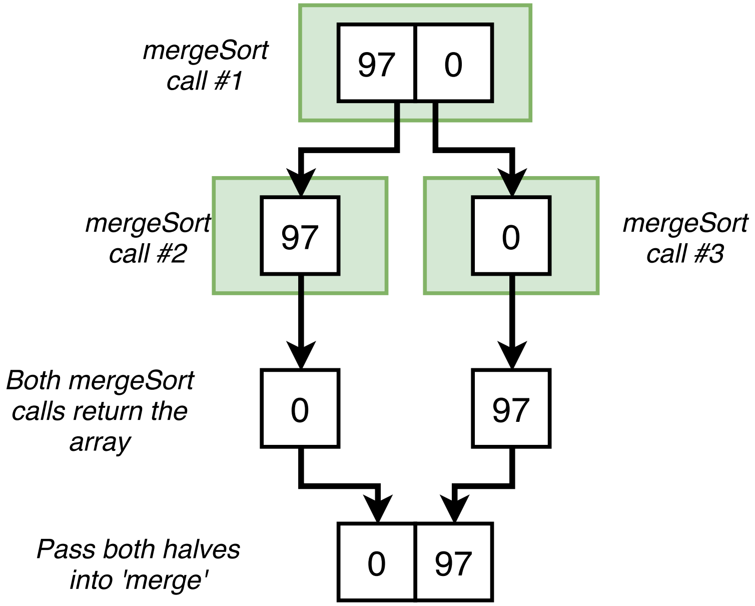


Split the array into two then split those halves

Split the array into two then split those halves

Array is size one, can't split anymore. Let's join thse two together

Join these two together



mergeSort call #1
97
0
mergeSort call #2
97
0
mergeSort call #3
Both mergeSort calls return the array
0
97
Pass both halves into 'merge'
0
97