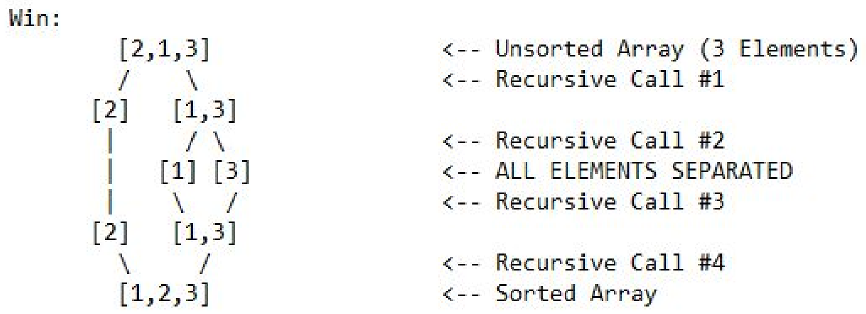
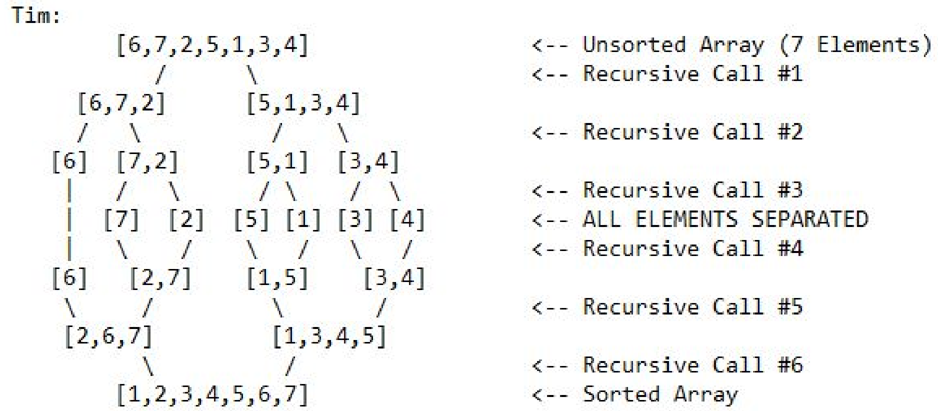
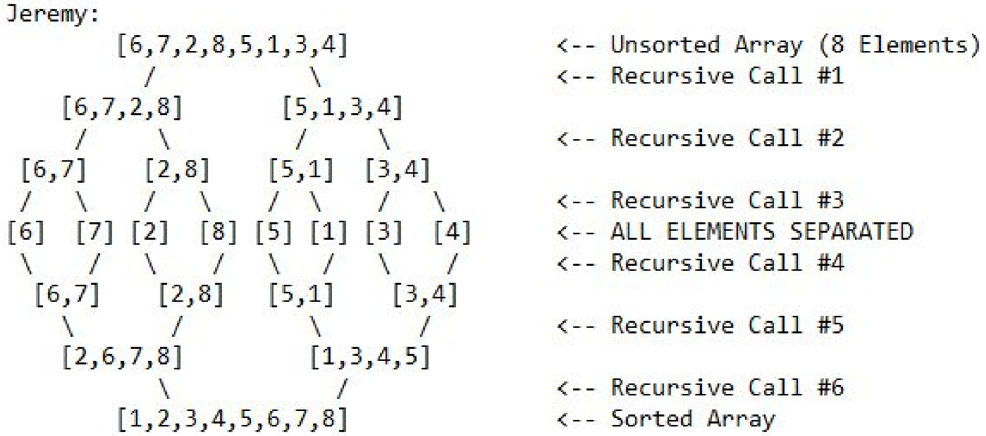
Kayli Matsuyoshi
APCS2 pd1
HW06 -- How Fast Are Your Turtles?
2018-0213t



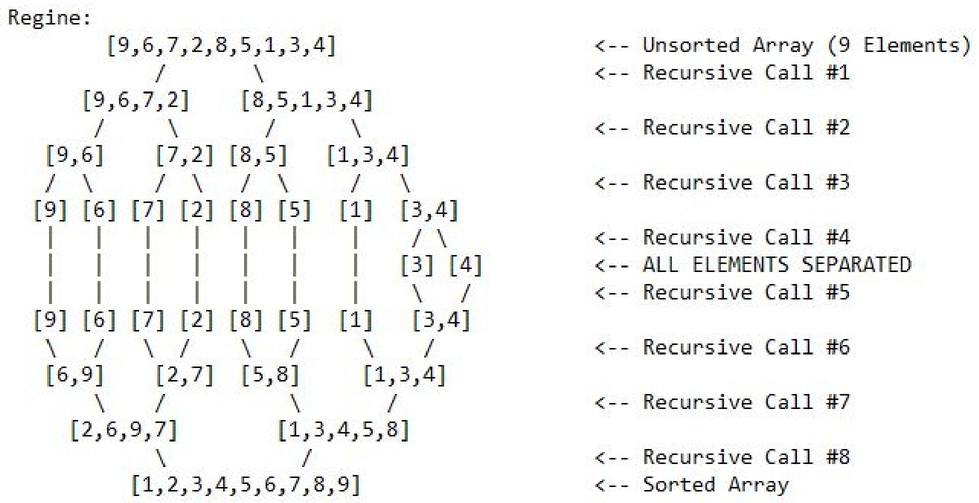
```
Win:
        [2,1,3]                  <-- Unsorted Array (3 Elements)
         /    \                  <-- Recursive Call #1
      [2]    [1,3]
        |    / \                 <-- Recursive Call #2
        |  [1] [3]               <-- ALL ELEMENTS SEPARATED
        |   \   /                <-- Recursive Call #3
      [2]    [1,3]
         \     /                 <-- Recursive Call #4
        [1,2,3]                  <-- Sorted Array
```



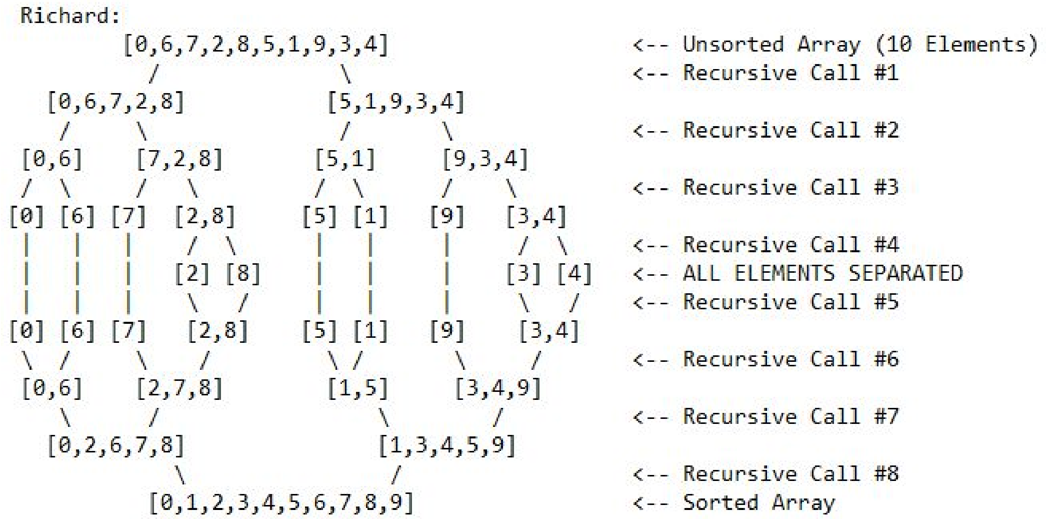
```
Tim:
        [6,7,2,5,1,3,4]                <-- Unsorted Array (7 Elements)
           /      \                    <-- Recursive Call #1
     [6,7,2]     [5,1,3,4]
     /   \          /    \             <-- Recursive Call #2
   [6]  [7,2]     [5,1]  [3,4]
    |   /   \      / \    /  \         <-- Recursive Call #3
    |  [7]  [2]  [5] [1] [3] [4]       <-- ALL ELEMENTS SEPARATED
    |   \    /    \   /   \   /        <-- Recursive Call #4
   [6]   [2,7]    [1,5]    [3,4]
    \     /         \       /          <-- Recursive Call #5
    [2,6,7]        [1,3,4,5]
          \          /                 <-- Recursive Call #6
        [1,2,3,4,5,6,7]                <-- Sorted Array
```



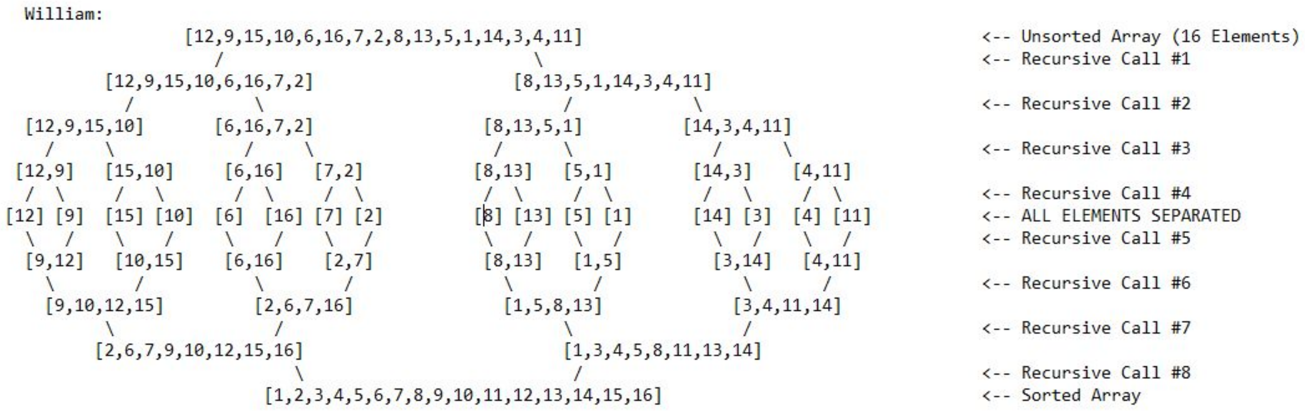
```
Jeremy:
        [6,7,2,8,5,1,3,4]               <-- Unsorted Array (8 Elements)
          /            \                <-- Recursive Call #1
    [6,7,2,8]       [5,1,3,4]
     /     \         /      \           <-- Recursive Call #2
 [6,7]    [2,8]    [5,1]  [3,4]
 /   \    /   \    /  \    /   \        <-- Recursive Call #3
[6]  [7] [2]  [8] [5] [1] [3]  [4]      <-- ALL ELEMENTS SEPARATED
 \    /   \    /   \   /   \    /       <-- Recursive Call #4
  [6,7]    [2,8]   [5,1]    [3,4]
    \       /          \       /        <-- Recursive Call #5
    [2,6,7,8]         [1,3,4,5]
           \            /               <-- Recursive Call #6
        [1,2,3,4,5,6,7,8]               <-- Sorted Array
```



```
Regine:
        [9,6,7,2,8,5,1,3,4]                     <-- Unsorted Array (9 Elements)
            /       \                           <-- Recursive Call #1
      [9,6,7,2]    [8,5,1,3,4]
       /     \       /      \                   <-- Recursive Call #2
   [9,6]    [7,2] [8,5]   [1,3,4]
   /  \     /  \   /  \     /   \               <-- Recursive Call #3
  [9] [6] [7] [2] [8] [5]  [1]  [3,4]
   |   |   |   |   |   |    |    / \            <-- Recursive Call #4
   |   |   |   |   |   |    |   [3] [4]         <-- ALL ELEMENTS SEPARATED
   |   |   |   |   |   |    |    \   /          <-- Recursive Call #5
  [9] [6] [7] [2] [8] [5]  [1]   [3,4]
   \   /   \  /    \   /     \    /             <-- Recursive Call #6
   [6,9]    [2,7]  [5,8]     [1,3,4]
       \    /          \       /                <-- Recursive Call #7
     [2,6,9,7]        [1,3,4,5,8]
            \           /                       <-- Recursive Call #8
          [1,2,3,4,5,6,7,8,9]                   <-- Sorted Array
```



```
 Richard:
         [0,6,7,2,8,5,1,9,3,4]                   <-- Unsorted Array (10 Elements)
           /              \                      <-- Recursive Call #1
   [0,6,7,2,8]           [5,1,9,3,4]
    /     \               /       \              <-- Recursive Call #2
 [0,6]    [7,2,8]       [5,1]     [9,3,4]
 /  \     /   \         /  \      /     \        <-- Recursive Call #3
[0] [6] [7]  [2,8]     [5] [1]   [9]   [3,4]
 |   |   |    /  \      |   |     |     /  \     <-- Recursive Call #4
 |   |   |   [2] [8]    |   |     |    [3] [4]   <-- ALL ELEMENTS SEPARATED
 |   |   |    \   /     |   |     |     \   /    <-- Recursive Call #5
[0] [6] [7]   [2,8]    [5] [1]   [9]    [3,4]
 \  /     \     /         \ /       \     /      <-- Recursive Call #6
 [0,6]    [2,7,8]        [1,5]     [3,4,9]
    \      /                 \        /          <-- Recursive Call #7
   [0,2,6,7,8]               [1,3,4,5,9]
             \                /                  <-- Recursive Call #8
           [0,1,2,3,4,5,6,7,8,9]                 <-- Sorted Array
```



```
  William:
                  [12,9,15,10,6,16,7,2,8,13,5,1,14,3,4,11]                                        <-- Unsorted Array (16 Elements)
                     /                               \                                            <-- Recursive Call #1
          [12,9,15,10,6,16,7,2]                    [8,13,5,1,14,3,4,11]
            /            \                              /            \                            <-- Recursive Call #2
  [12,9,15,10]       [6,16,7,2]                 [8,13,5,1]          [14,3,4,11]
    /     \             /     \                  /      \              /      \                   <-- Recursive Call #3
 [12,9]   [15,10]     [6,16]   [7,2]           [8,13]   [5,1]        [14,3]    [4,11]
  /  \     /   \       /  \     /  \            /  \     /  \         /   \     /  \              <-- Recursive Call #4
[12] [9]  [15] [10]  [6]  [16] [7] [2]         [8] [13] [5] [1]      [14] [3]  [4] [11]           <-- ALL ELEMENTS SEPARATED
  \   /    \    /     \    /    \   /           \   /    \   /         \   /    \   /             <-- Recursive Call #5
  [9,12]   [10,15]    [6,16]    [2,7]           [8,13]   [1,5]         [3,14]   [4,11]
    \        /           \        /               \        /              \       /               <-- Recursive Call #6
    [9,10,12,15]         [2,6,7,16]               [1,5,8,13]             [3,4,11,14]
          \                /                            \                 /                       <-- Recursive Call #7
         [2,6,7,9,10,12,15,16]                          [1,3,4,5,8,11,13,14]
                             \                           /                                        <-- Recursive Call #8
                          [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]                                <-- Sorted Array
```

The number of recursive calls once all the elements are divided can be doubled to obtain the total number of recursive calls for the sort.
I think the runtime is affected by the divisibility of the array. 2^x arrays can be sorted efficiently without "holding elements".