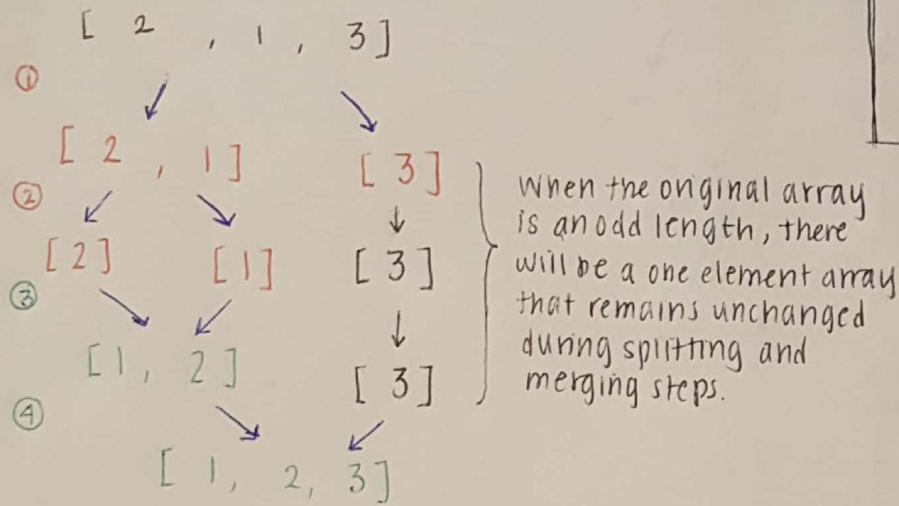
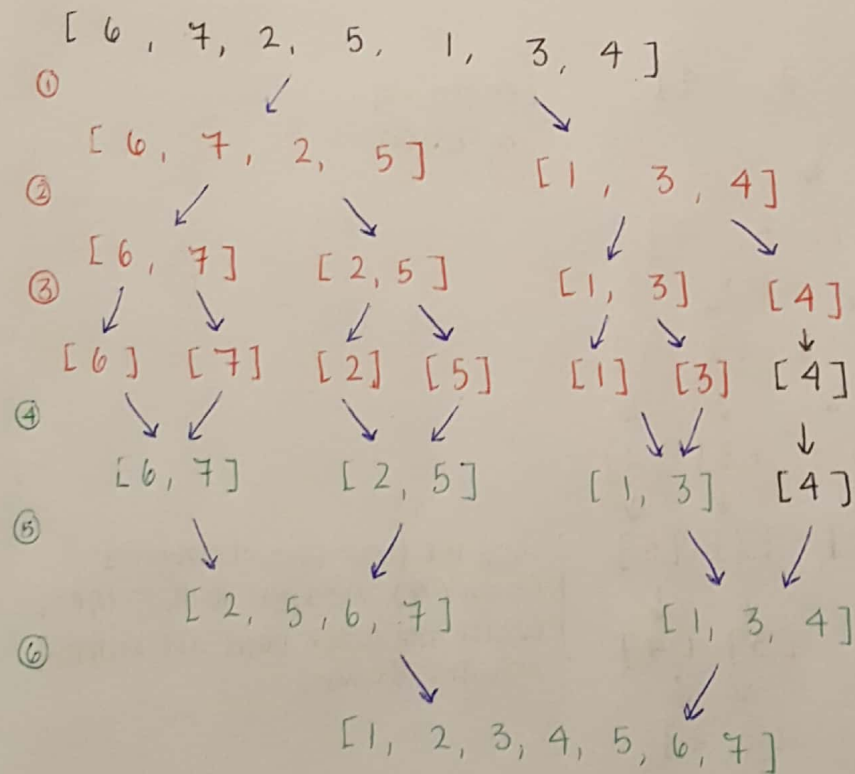


Maggie Zhao
 APCS 2 pd01
 HW06-- How Fast Are Your Turtles?
 2018-02-12

Win:



Tim:



Key:

- \rightarrow = either a split or a merge is occurring in this step
- \rightarrow = no action occurred (the element has been brought down to the next layer)

$[x, \dots]$ = the array has been split

$[x, \dots]$ = the array has been merged.

\otimes = splitting step

\otimes = merging step

$[x, \dots]$ = no change occurred

Length = 3

4 splits

4 merges

4 layers

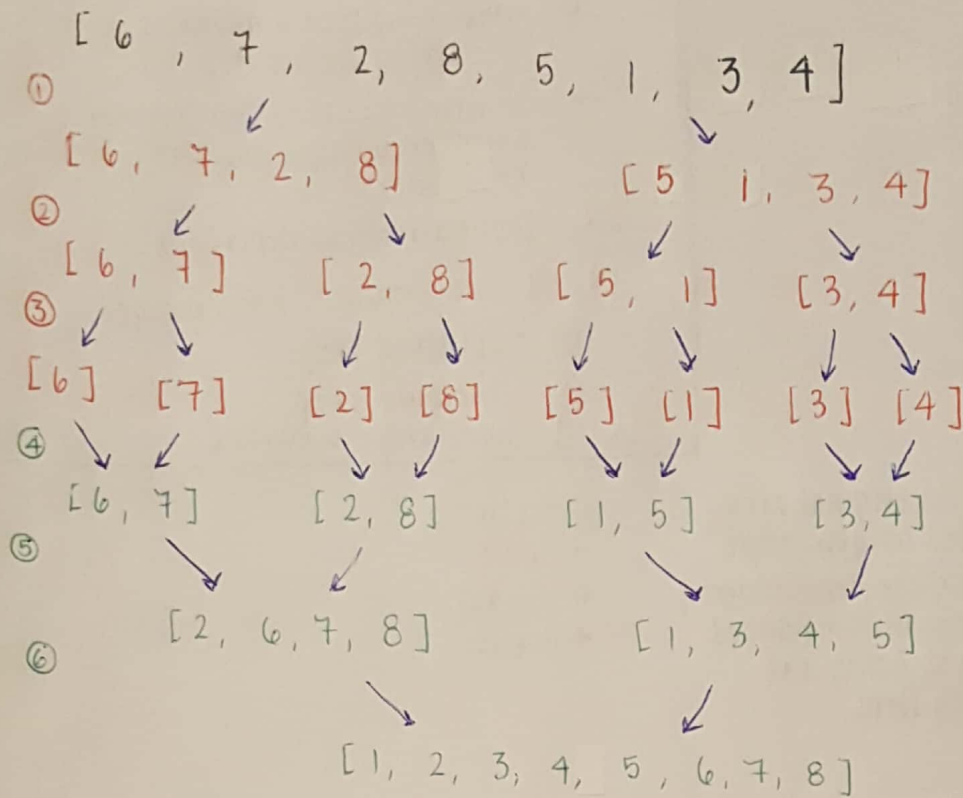
Length = 7

12 splits

12 merges

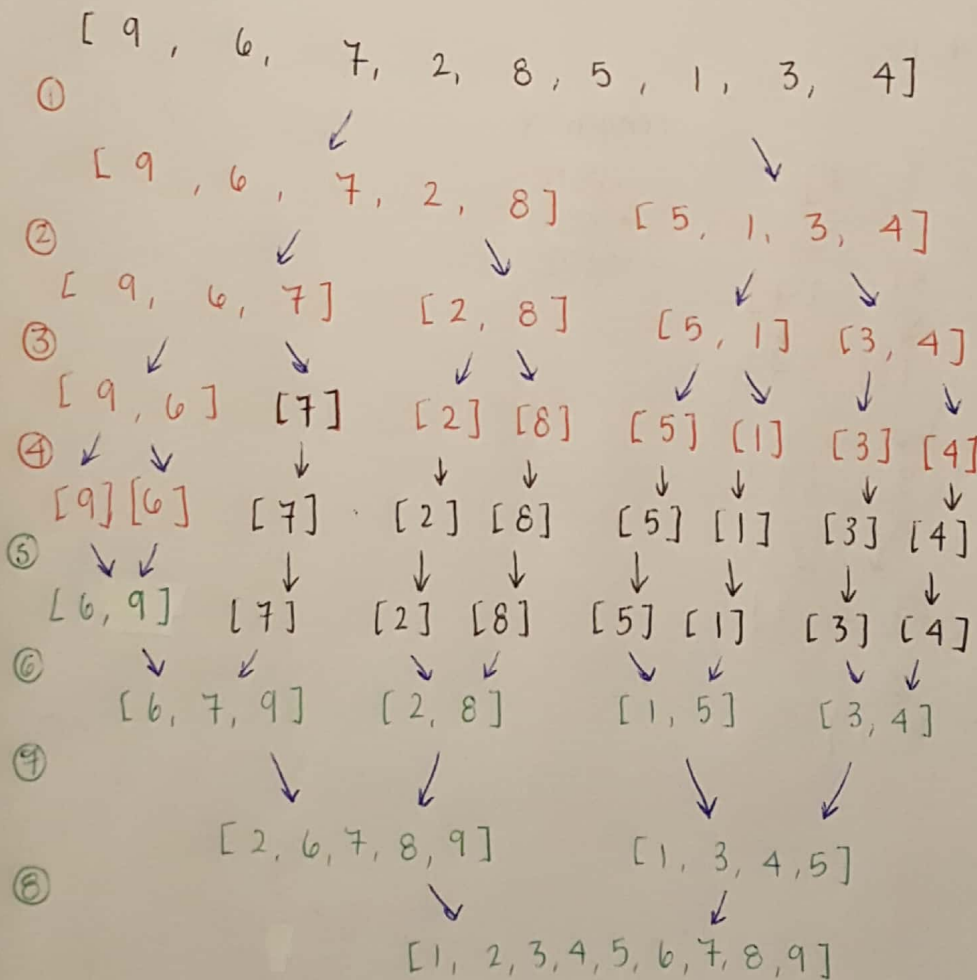
6 layers

Jeremy:



Length = 8
14 splits
14 merges
6 layers

Régine:



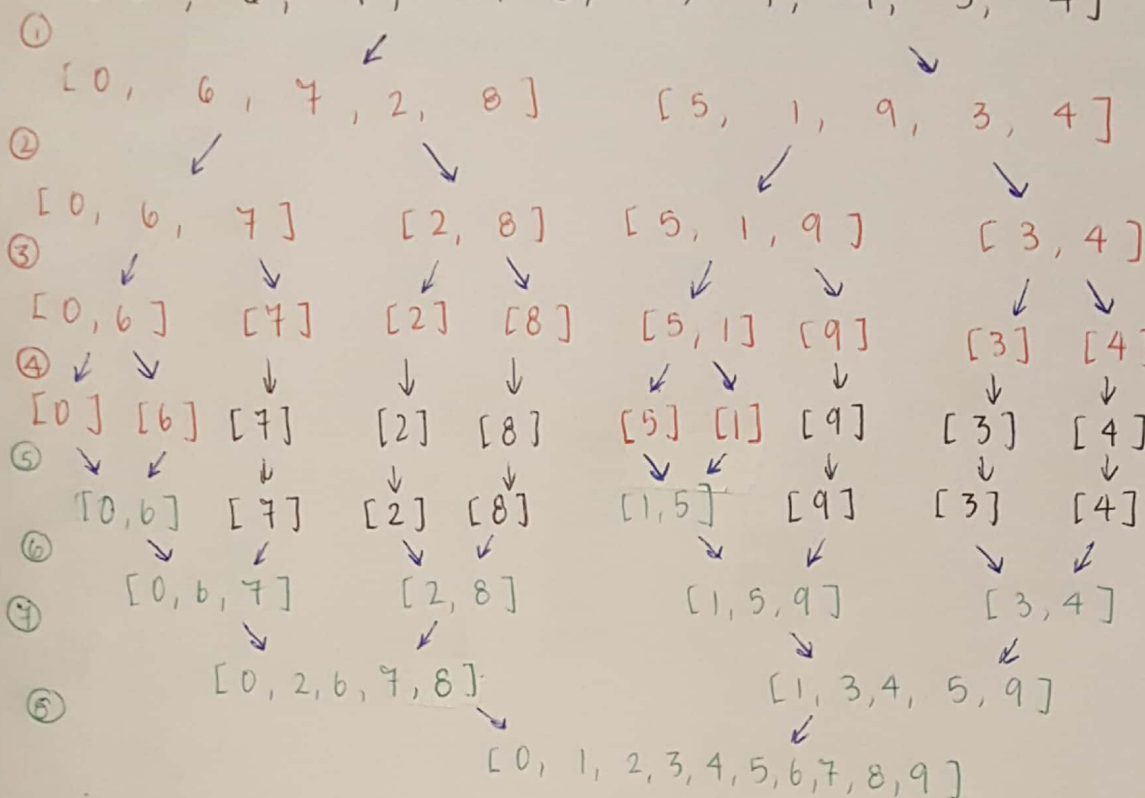
Length = 9
8 layers

} Only the first two elements (6 and 9) are split and merged, while the other pairs are simply brought down.

Richard

[0, 6, 7, 2, 8, 5, 1, 9, 3, 4]

Length: 10
8 layers

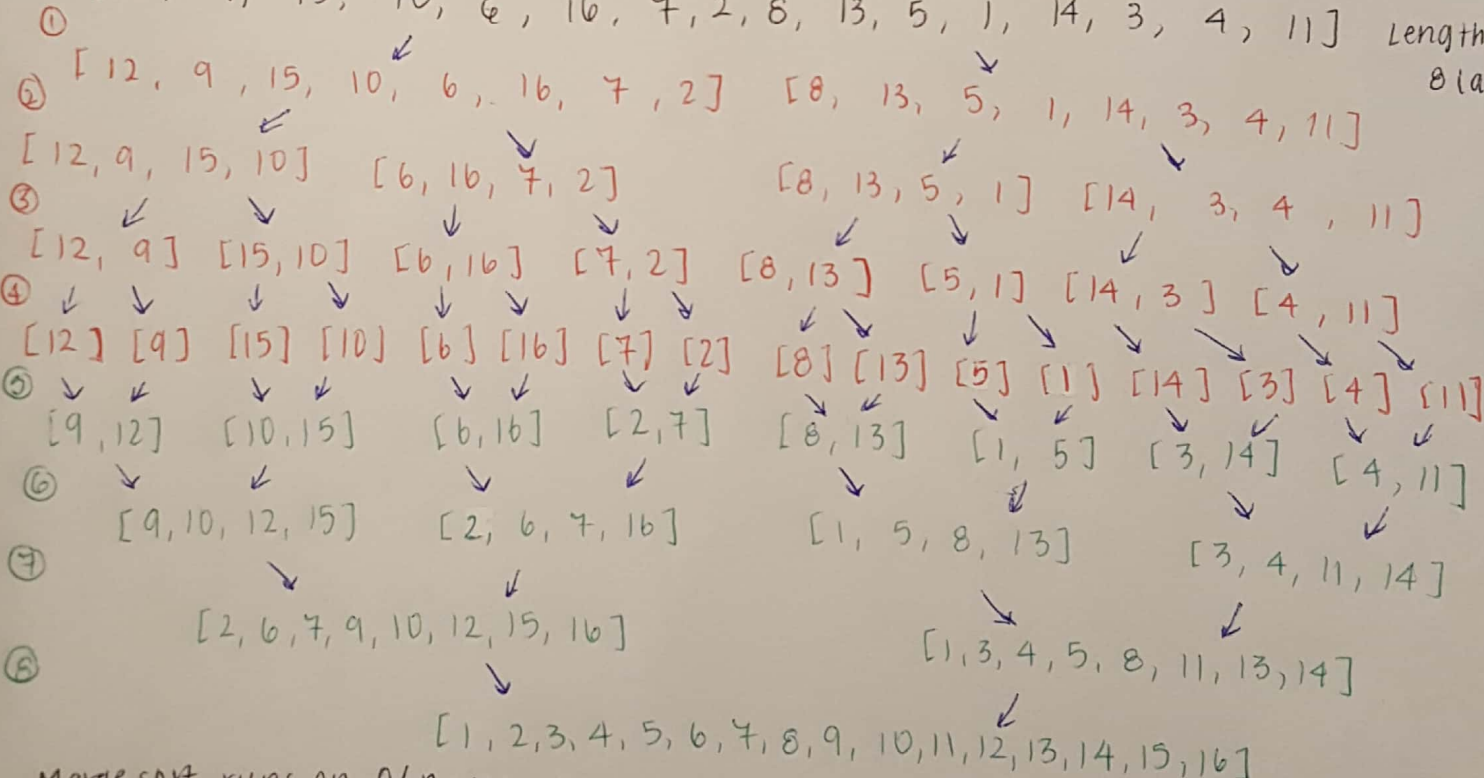


The elements are merged along the same lines they are split. If the element was not part of an x-element array originally, it cannot be merged into one.

William

[12, 9, 15, 10, 6, 16, 7, 2, 8, 13, 5, 1, 14, 3, 4, 11]

Length=16
8 layers



Merge sort runs on $O(n \cdot \log_2 n)$ time. When each array was being sorted, it took (the ceiling of) $\log_2 n$ passes for the entire array to be split up into individual one element arrays. This is because, like binary search, it reduces the field of vision by 2 with each pass, but in this sort algorithm, all of the different views are stored. The merge algorithm is $O(n)$, occurs at every level of the dividing part, and n items iterated $\log_2(n)$ times yields a total time of $O(n \log_2 n)$.