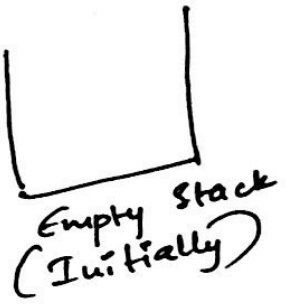
Merge Sort:

Base State a = | 38 | 10 | 43 | 3 | 9 | 82 | 27 |

left: | 38 | 10 | 43 |

right: | 3 | 9 | 82 | 27 |

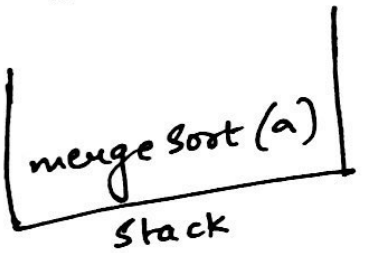


First Recursive Call –

| 38 | 10 | 43 |

left: | 38 |

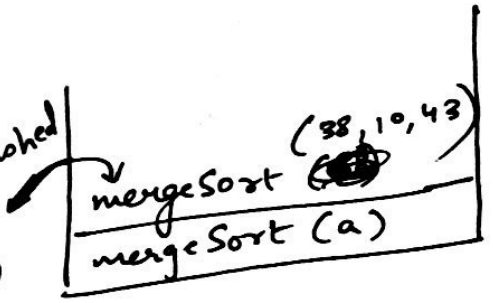
right: | 10 | 43 |



2nd Recursive Call –

| 38 |

↳ exits as only single element is present; pops mergeSort (38, 10, 43)

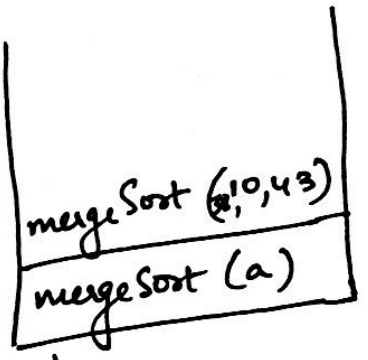


3rd Recursive Call –

| 10 | 43 |

left: | 10 |

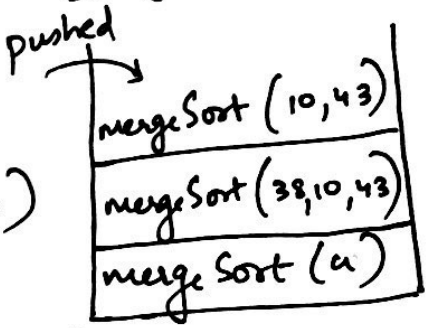
right: | 43 |



4th, 5th Recursive Call –

| 10 | | 43 |

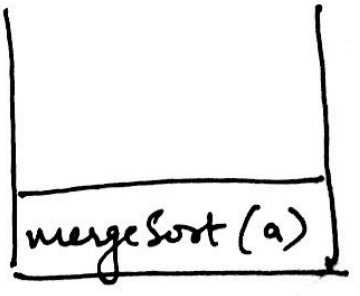
↳ exits as only single values are present pops mergeSort (10, 43)



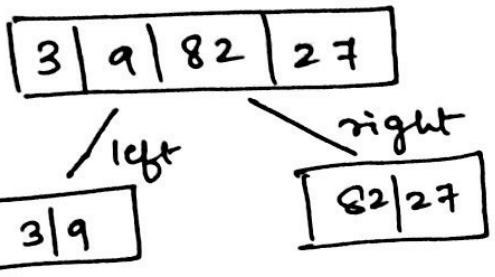
End of 3rd recursive call, pops elements from stack.

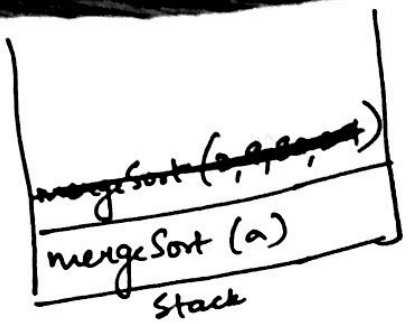
Merges left & right splits | 10 | 38 | 43 |

Returns to 1st recursive call

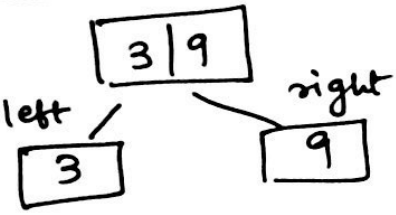


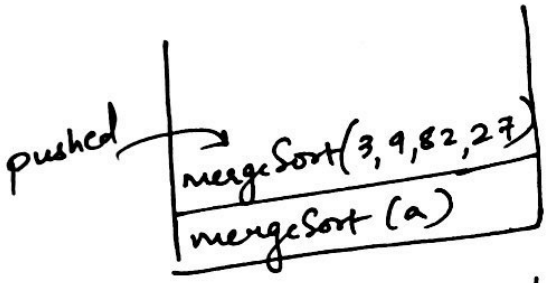
6th Recursive Call –





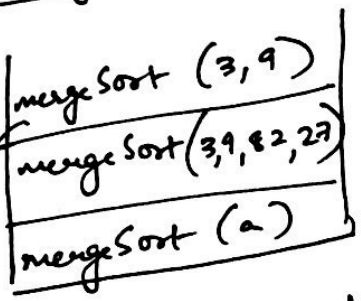
7th Recursive Call –





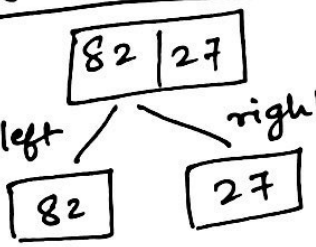
8th, 9th Recursive Call –

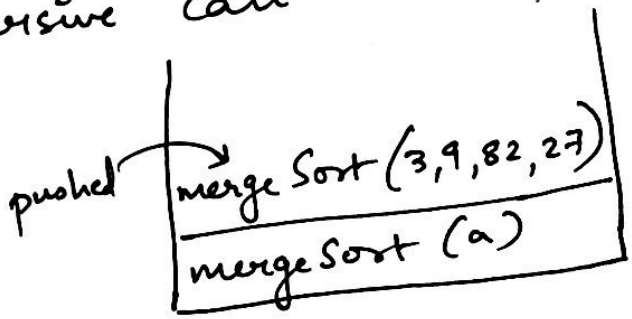
↳ exits as only single values are present pop pops mergeSort(3,9)



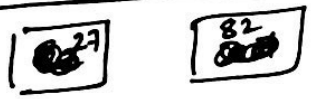
End of 7th recursive call, pops element from stack
Merges left & right halves 3 | 9
Returns to the 6th Recursive Call.

10th Recursive Call –

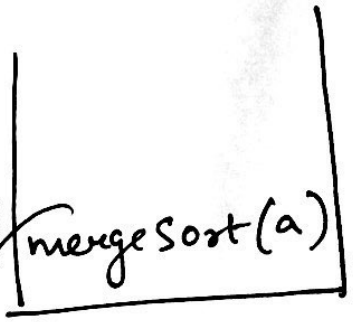


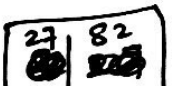


11th Recursive Call –



↳ exits as only single values are present pops mergeSort(3,9,82,27) pop



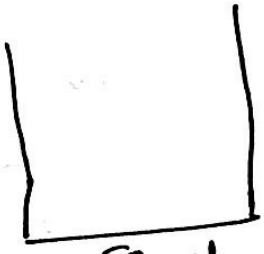
End of 10th Recursive Call
Merges left & right halves 
Returns to 6th Recursive Call
Merges left & right halves 3 | 9 | 27 | 82

Now,

Return to the base state

left sorted array | 10 | 38 | 43 |

right sorted array | 3 | 9 | 27 | 82 |



Final Empty Stack.

Merge left and right sorted array

| 3 | 9 | 10 | 27 | 38 | 43 | 82 |

↳ Final Sorted Array