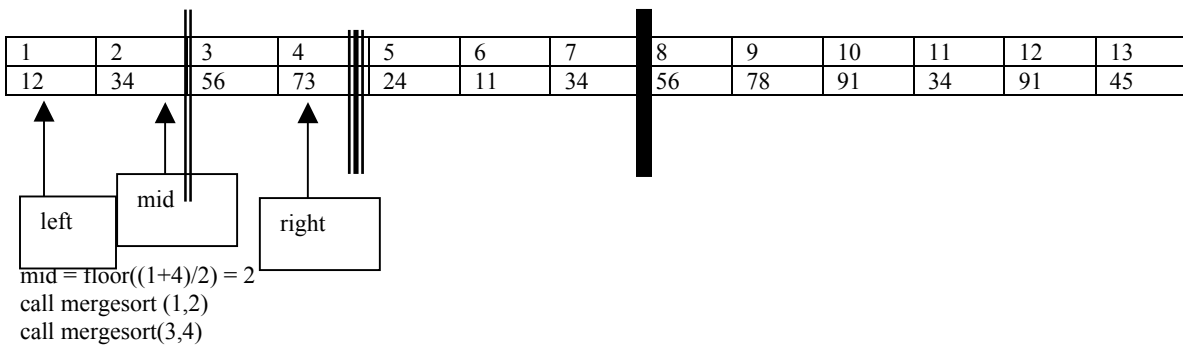
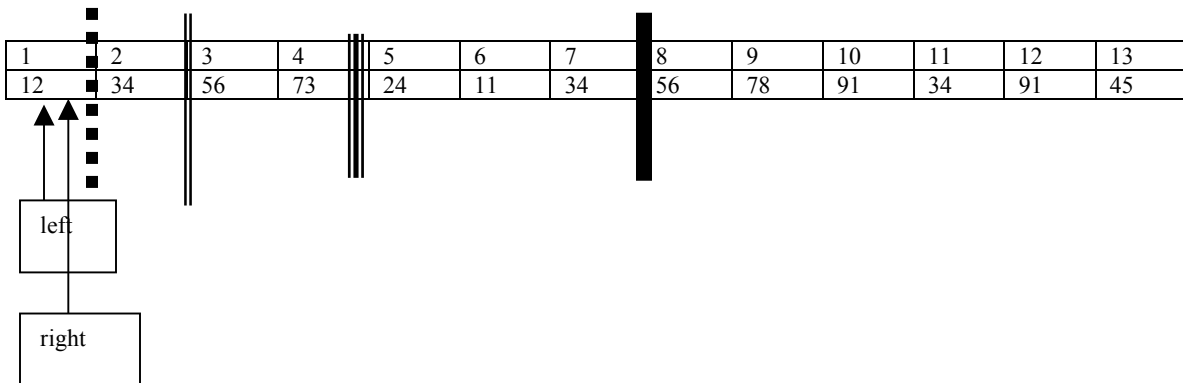
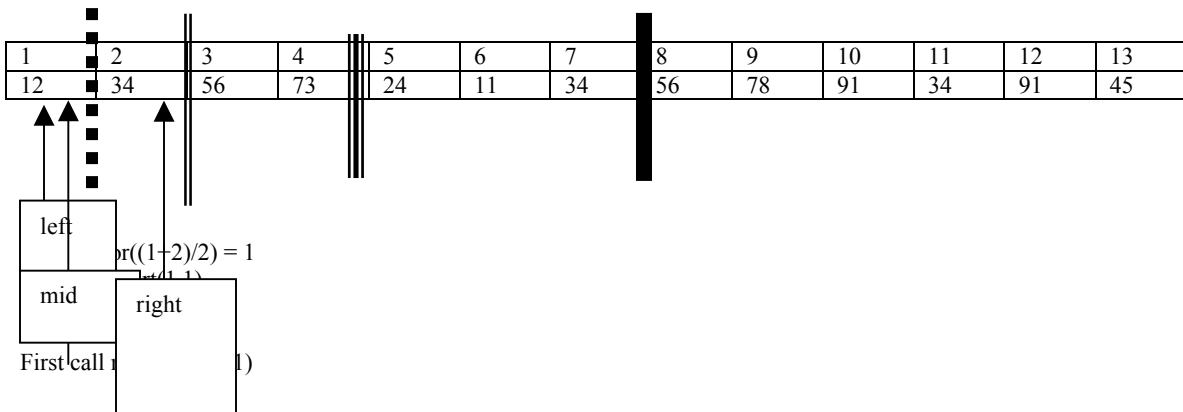


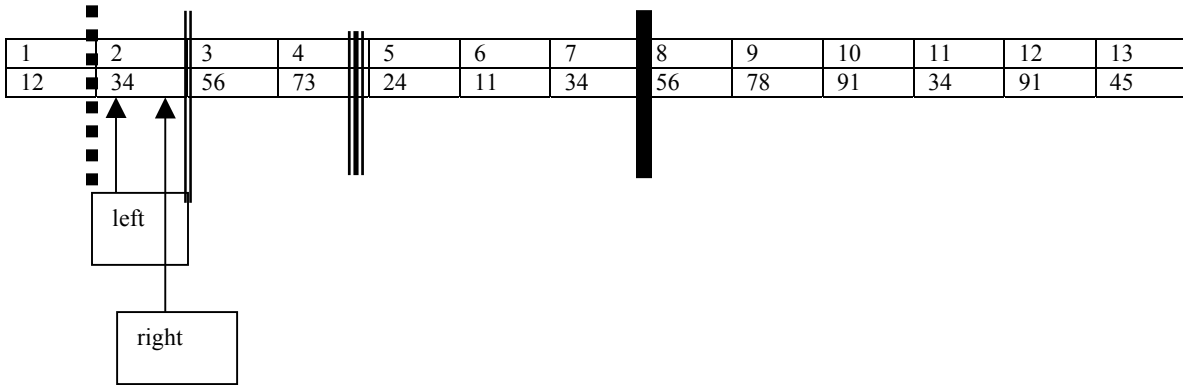
First call mergesort(1,4)



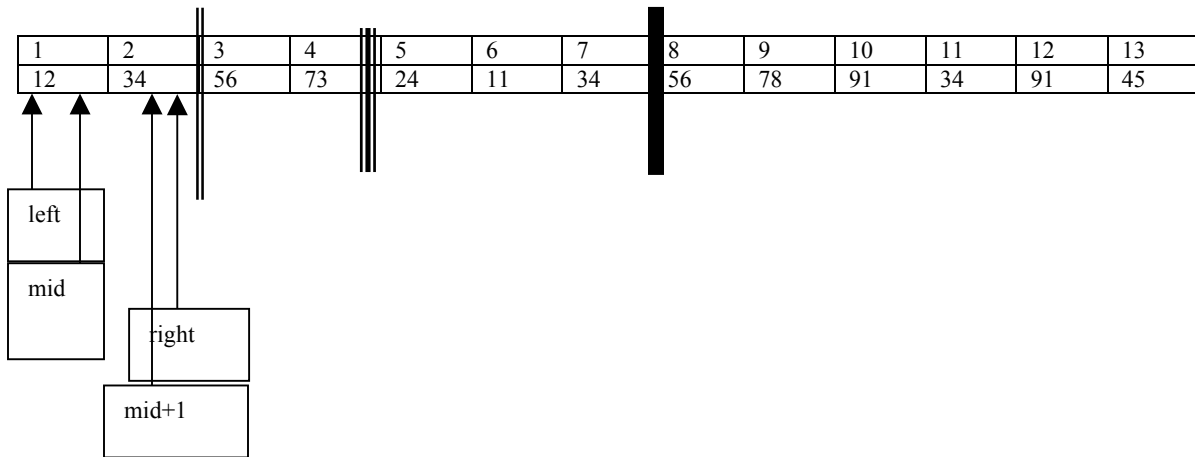
First call mergesort(1,2)



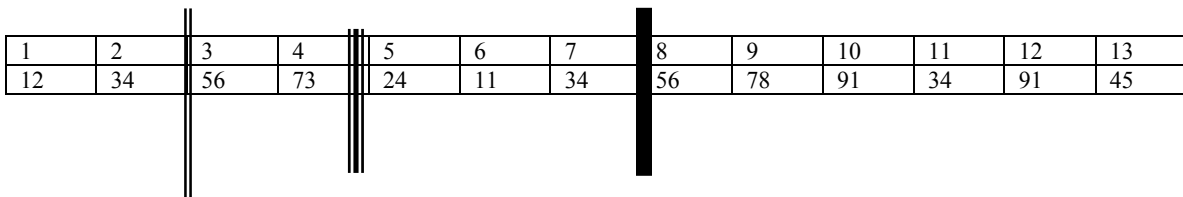
As there is only one element to be sorted control goes back to mergesort.  
 Call mergesort(2,2).



As there is only one element to be sorted control goes back to mergesort.  
 NOW we call merge(left, mid, mid+1, right)=mid(1, 1, 2, 2)

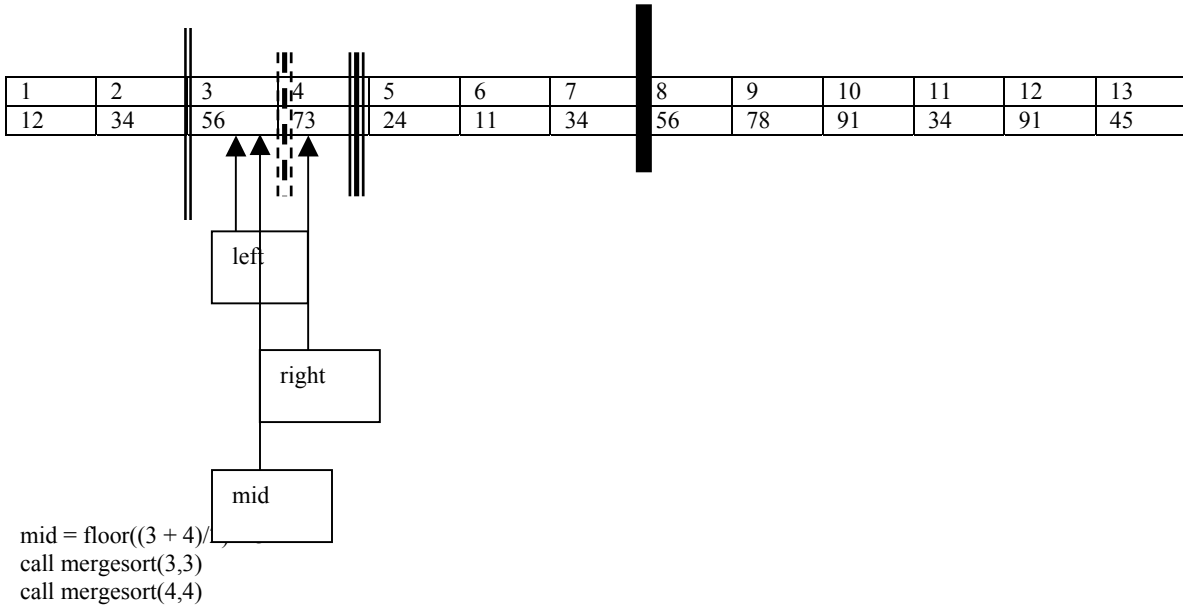


This sort the elements in 1 to 2.

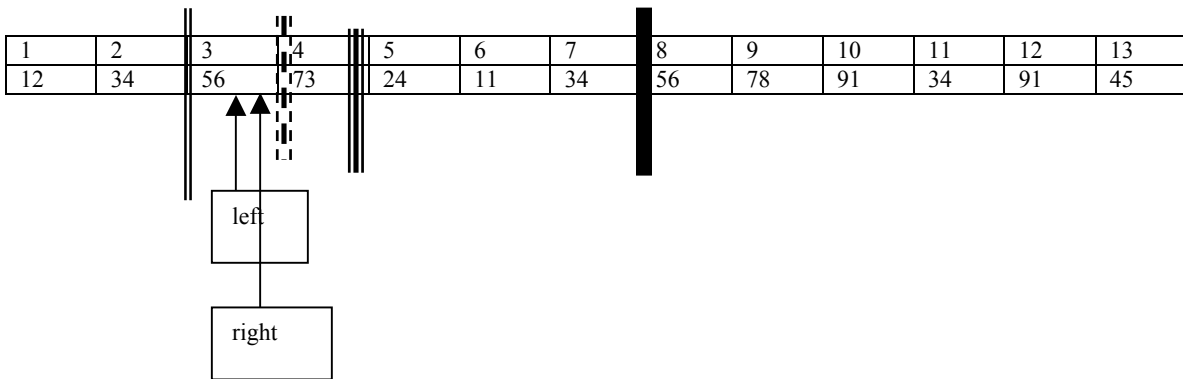


Control goes back to mergesort.

Call mergesort(3,4)

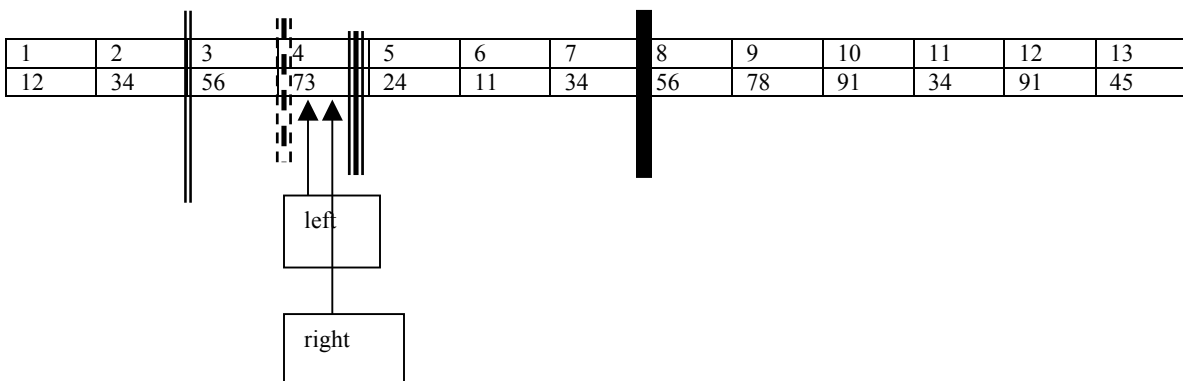


First call mergesort(3,3)



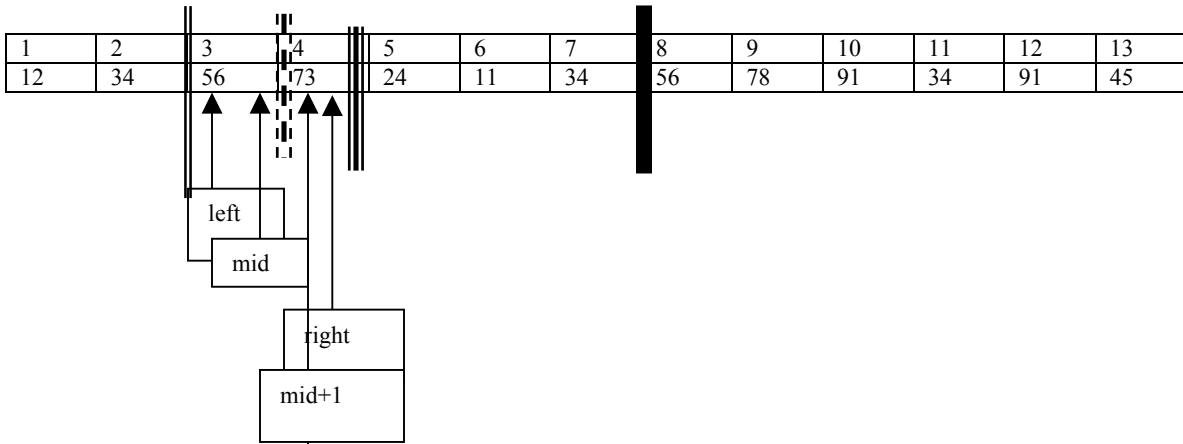
As there is only one element to be sorted this returns control to mergesort.

Call mergesort(4,4)

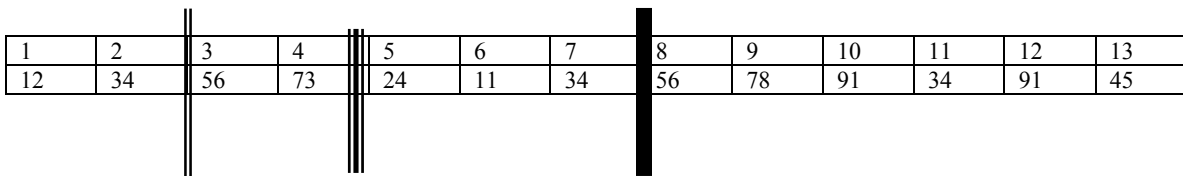


As there is only one element to be sorted this returns control to mergesort

NOW CALL  $\text{merge}(\text{left}, \text{mid}, \text{mid}+1, \text{right}) = \text{merge}(3, 3, 4, 4)$

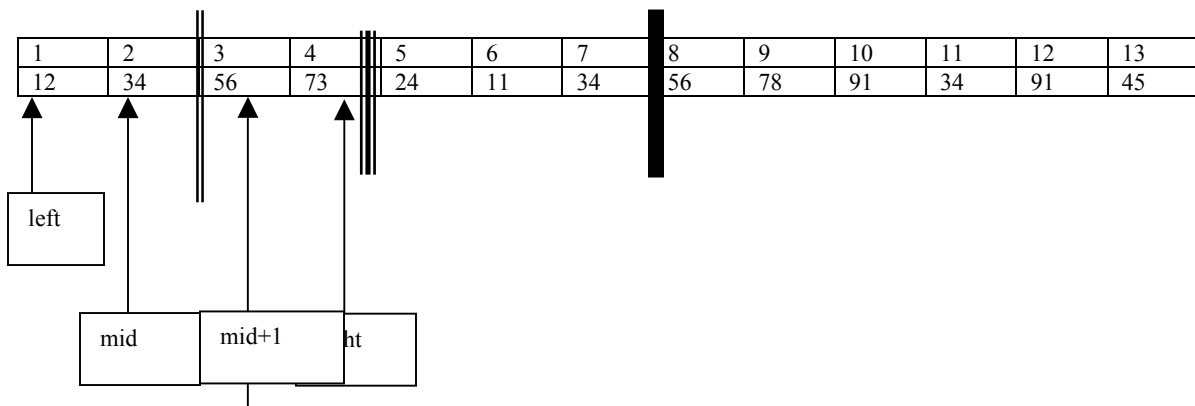


This merges and results in a sorted list from 3 to 4

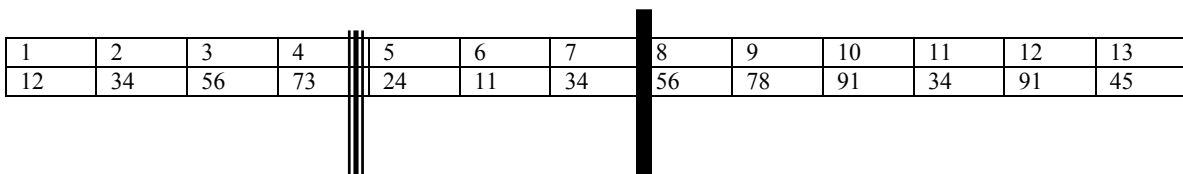


Control returns to mergesort.

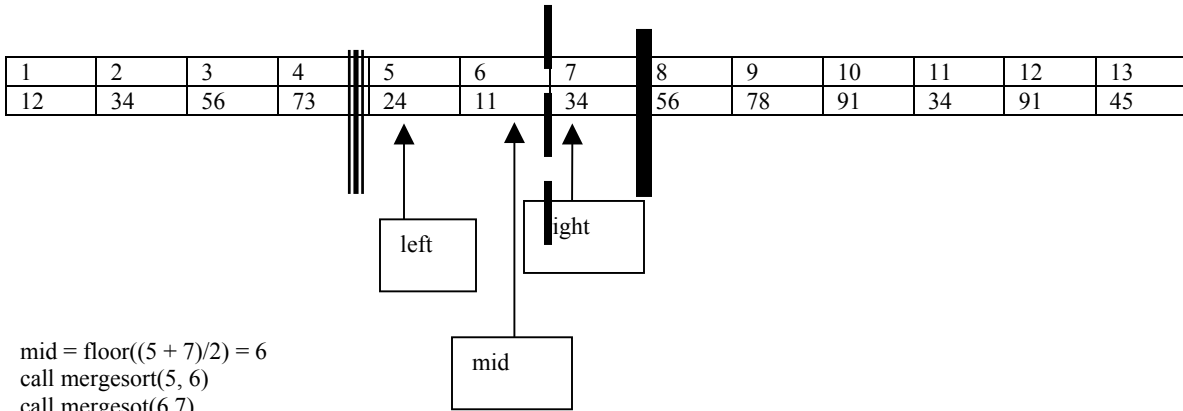
NOW CALL  $\text{merge}(\text{left}, \text{mid}, \text{mid}+1, \text{right}) = \text{merge}(1, 2, 3, 4)$



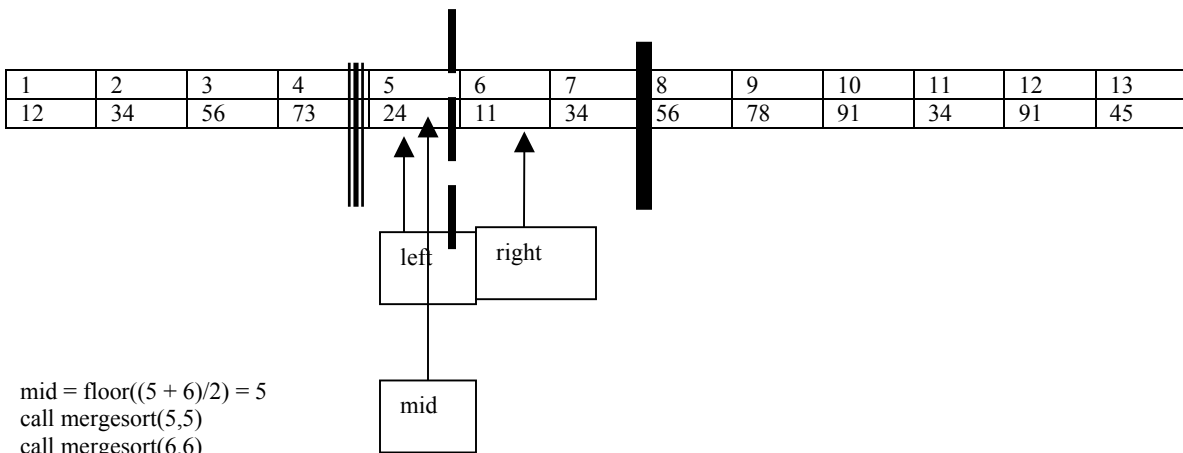
This results in a sorted list from 1 to 4



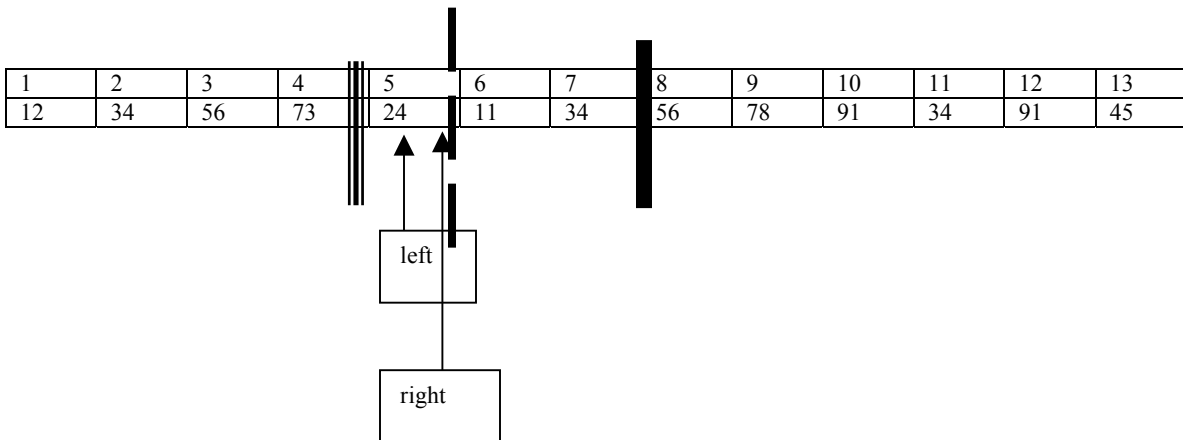
Call mergesort(5,7)



$mid = \text{floor}((5 + 7)/2) = 6$   
 call mergesort(5, 6)  
 call mergesort(6, 7)  
 First call mergesort(5,6)

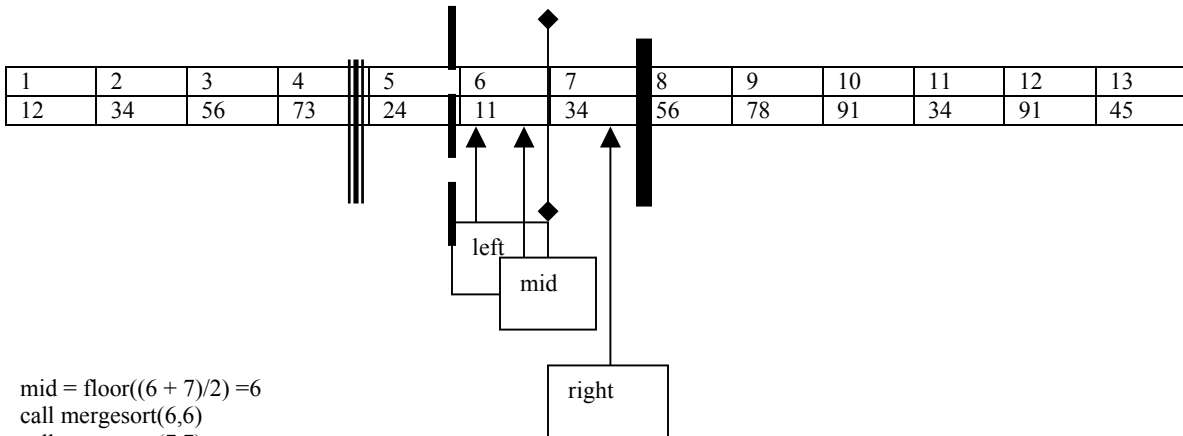


$mid = \text{floor}((5 + 6)/2) = 5$   
 call mergesort(5,5)  
 call mergesort(6,6)  
 call mergesort(5,5)



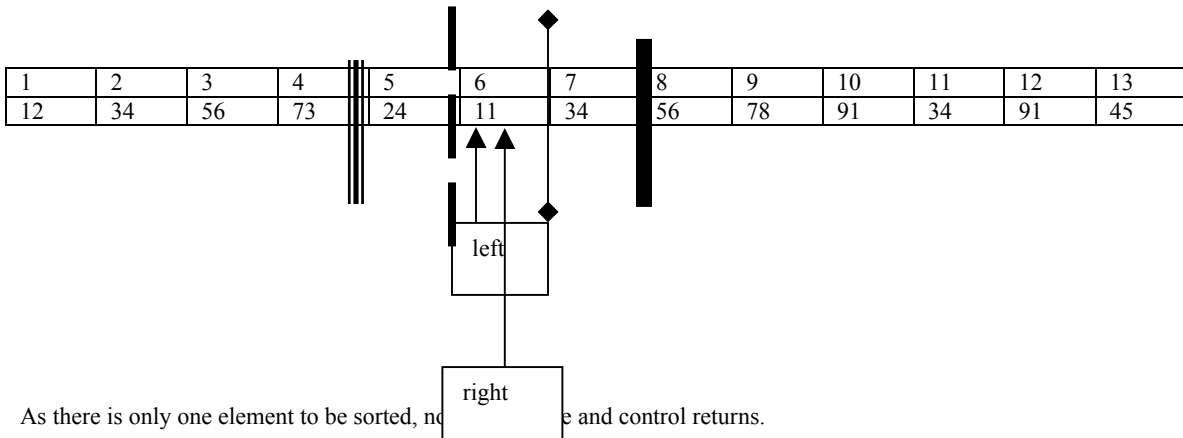
This returns back as one element is always sorted.

Call mergesort(6,7)



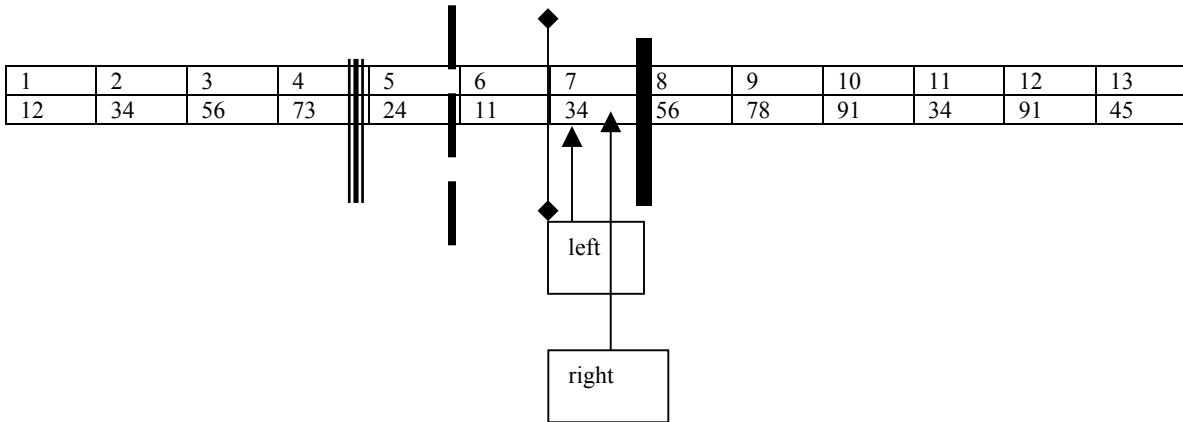
$mid = \text{floor}((6 + 7)/2) = 6$   
 call mergesort(6,6)  
 call mergesort(7,7)

First call mergesort(6, 6)



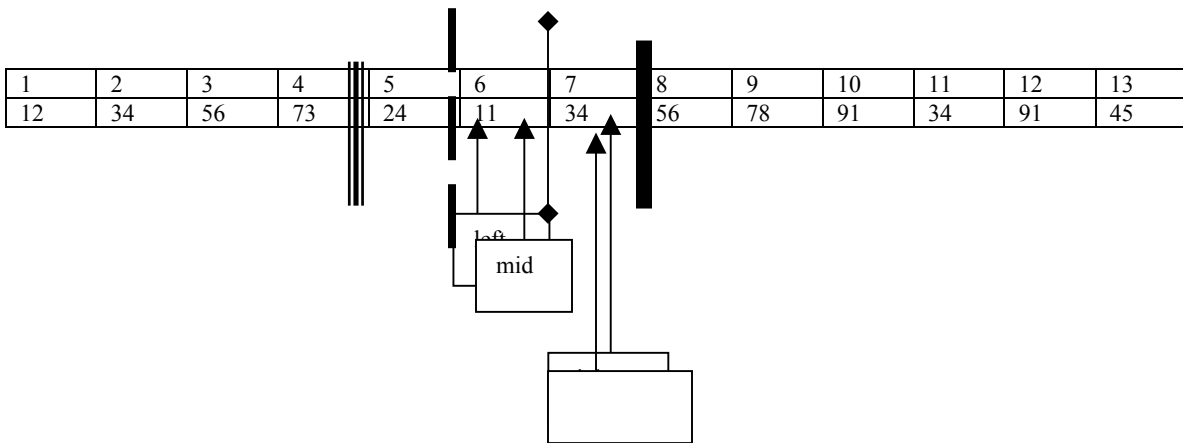
As there is only one element to be sorted, no further recursive calls are made and control returns.

Call mergesort(7,7)

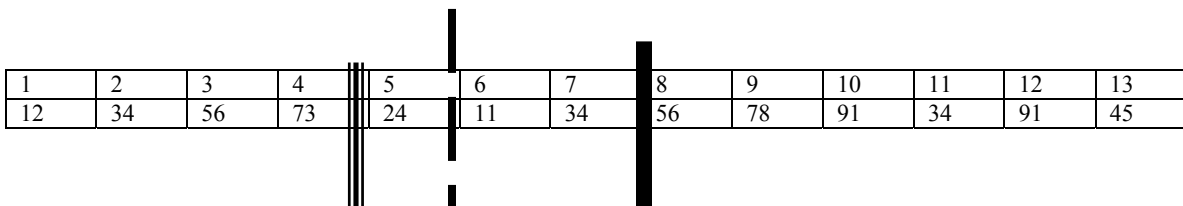


As there is only one element to be sorted, control returns without doing anything.

NOW CALL merge(left, mid, mid+1, right) = merge(6, 6, 7, 7)

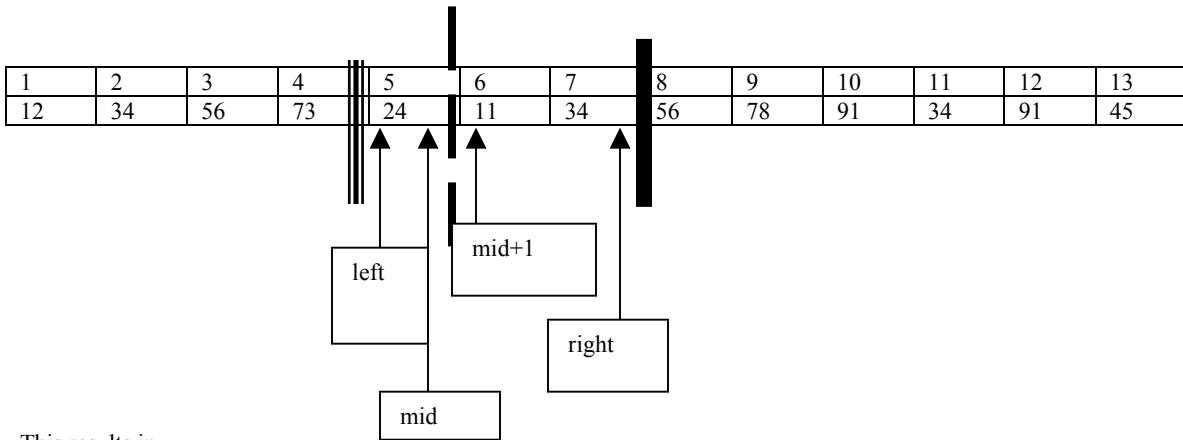


This results in elements 6 to 7 being merged, and a sorted list results from element 6 to element 7.

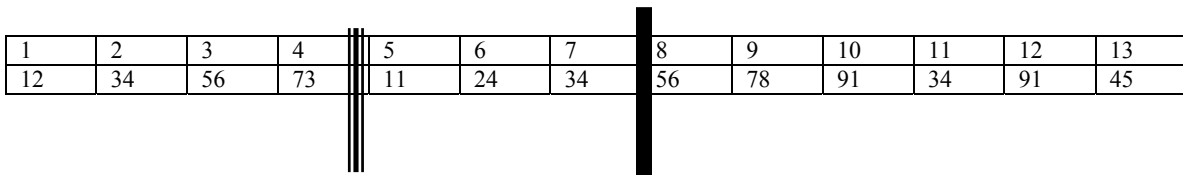


Control returns to mergesort.

NOW CALL merge(left, mid, mid+1, right) = merge(5, 5, 6, 7)

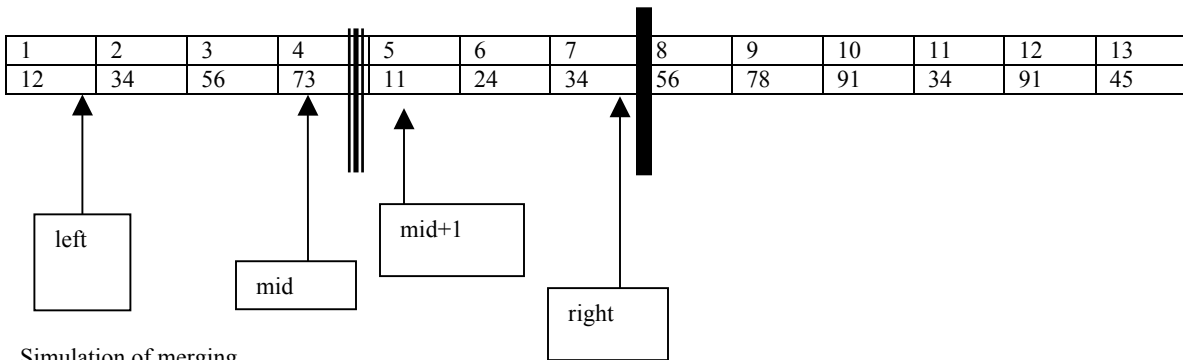


This results in



Now control returns to mergesort.

Call  $\text{merge}(\text{left}, \text{mid}, \text{mid}+1, \text{right}) = \text{merge}(1, 4, 5, 7)$



Simulation of merging.

Initial

Sorted list a[1..4]	Sorted list a[5..7]	Sorted list[1..7]
12	11	
34	24	
56	34	
73		

Compare(12,11) and move 11

Sorted list a[1..4]	Sorted list a[5..7]	Sorted list[1..7]
12		
34	24	
56	34	
73		
		11

Compare(12,24) and move 12

Sorted list a[1..4]	Sorted list a[5..7]	Sorted list[1..7]
34	24	
56	34	
73		
		12
		11

Compare(34,24) and move 24

Sorted list a[1..4]	Sorted list a[5..7]	Sorted list[1..7]
34		
56	34	
73		
		24
		12
		11

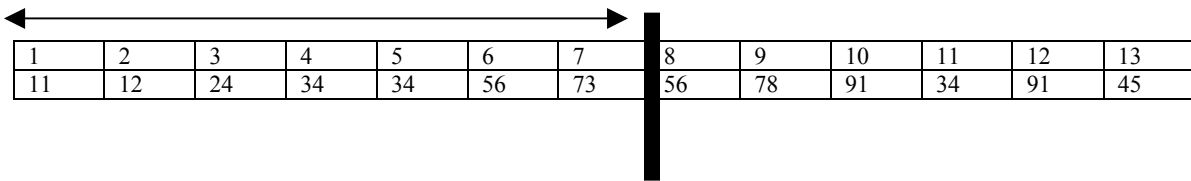
Compare(34,34) and move 34

Sorted list a[1..4]	Sorted list a[5..7]	Sorted list[1..7]
34		
56		
73		
		34
		24
		12
		11

Empty out list on the left to the sorted list at the right.

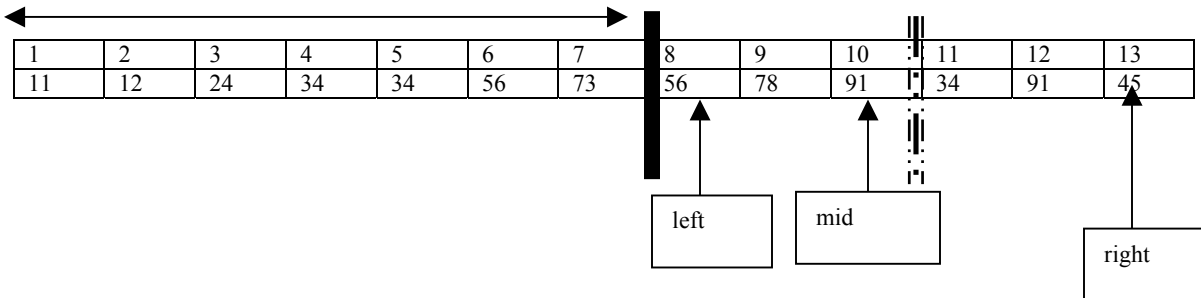
Sorted list a[1..4]	Sorted list a[5..7]	Sorted list[1..7]
		73
		56
		34
		34
		24
		12
		11

Sorted list 1..7



Call mergesort(8,13)

Sorted list 1..7



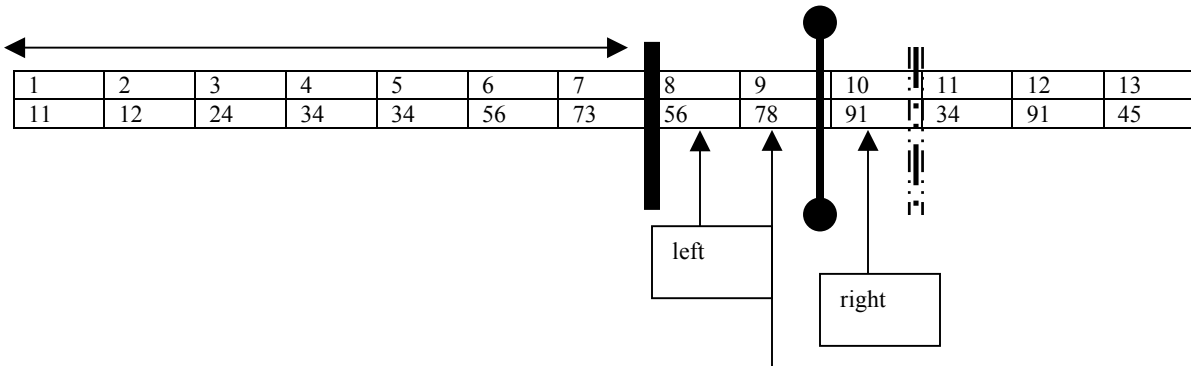
$mid = \text{floor}((8, 13)/2) = 10$

call mergesort(8, 10)

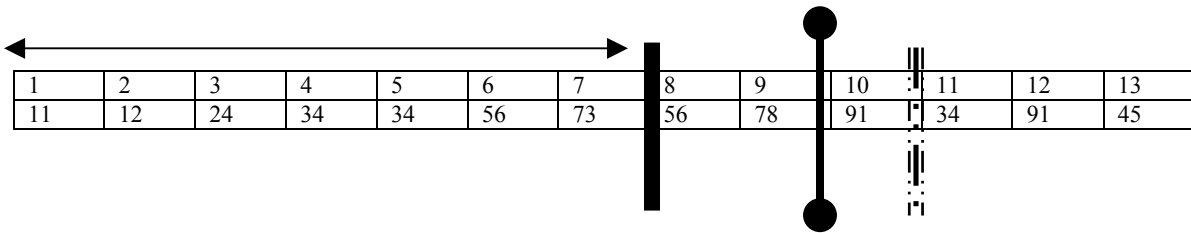
call mergesort(11, 13)

First call mergesort(8,10)

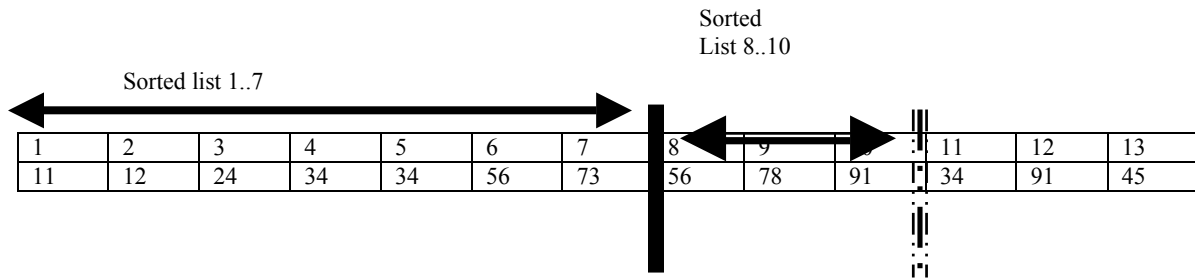
Sorted list 1..7



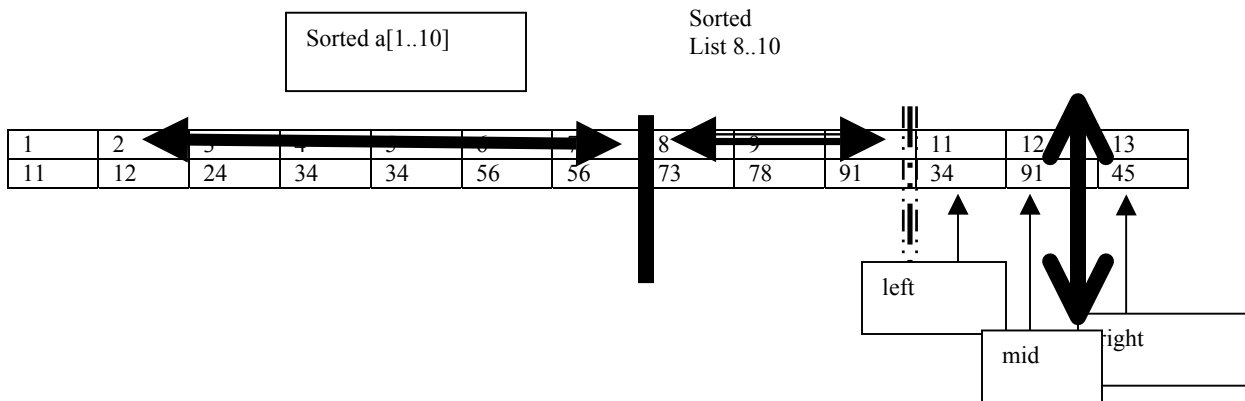




Call on mergesort(10,10) returns without doing anything as only one element is to be sorted.  
 NOW CALL merge(8, 9, 10) which results in

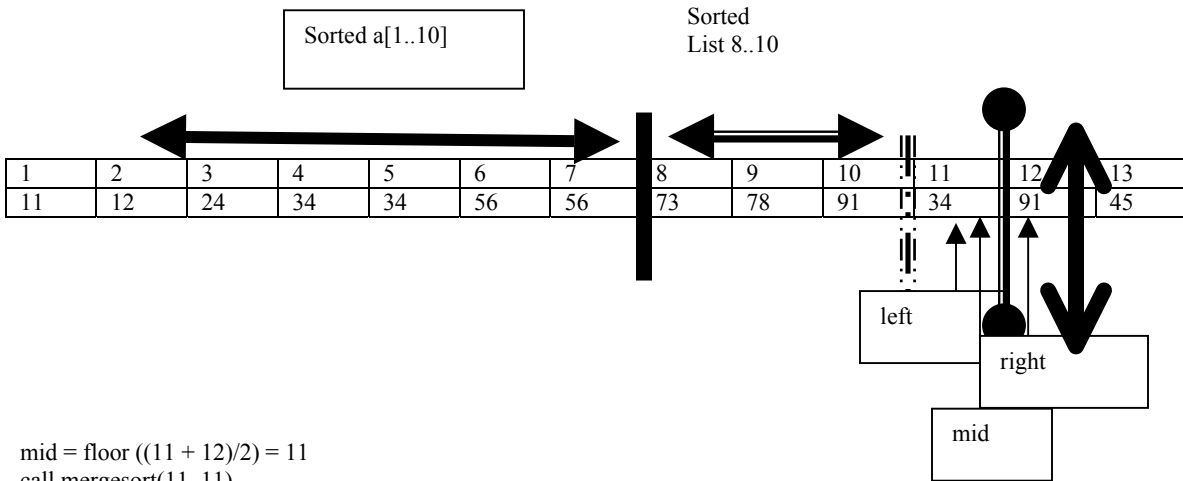


Now control returns to mergesort  
 Call mergesort(11,13)



$mid = \text{floor}((11+13)/2) = 12$   
 call mergesort(11, 12)  
 call mergesort(13, 13)

First call mergesort(11, 12)



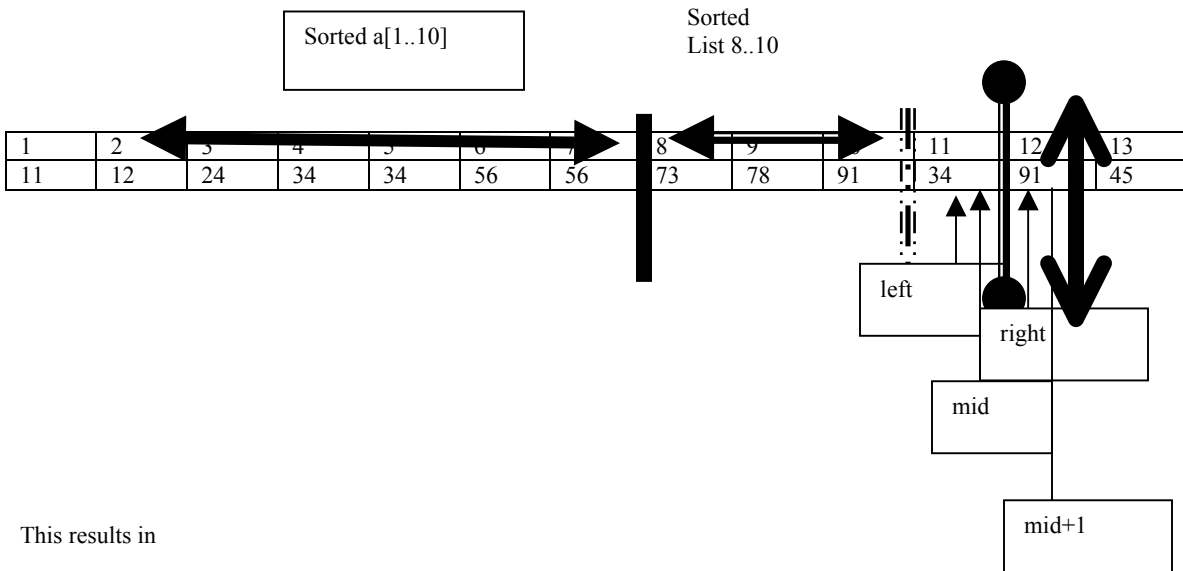
mid = floor ((11 + 12)/2) = 11

call mergesort(11, 11)

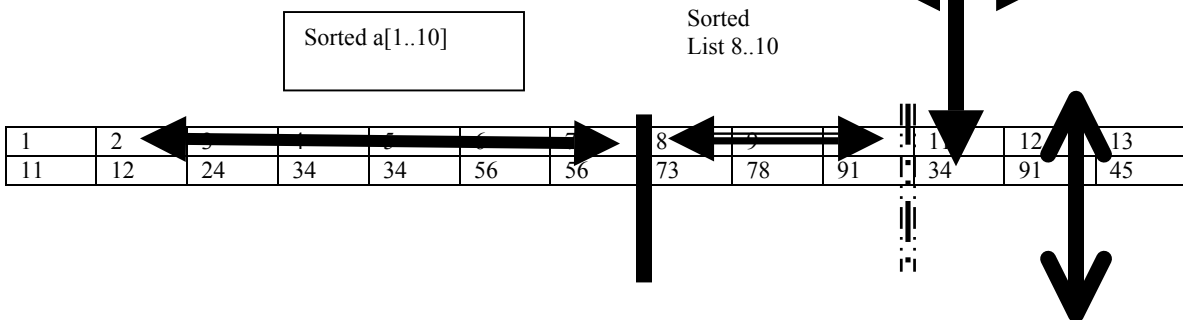
call mergesort(12, 12)

Both the above calls return doing nothing as only one element is to be sorted.

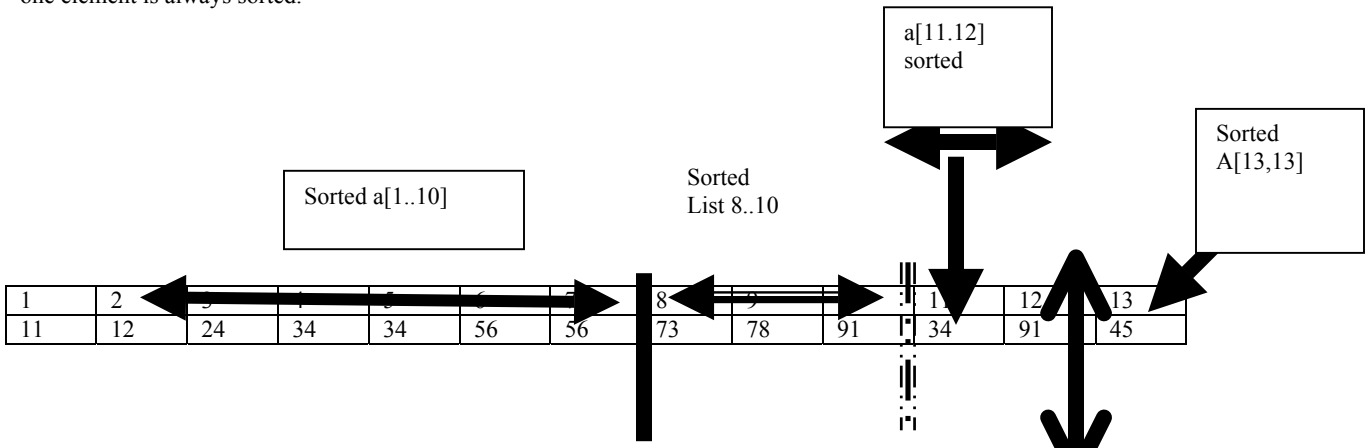
NOW call merge(11,11,12,12) = merge(left, mid, mid+1, right)



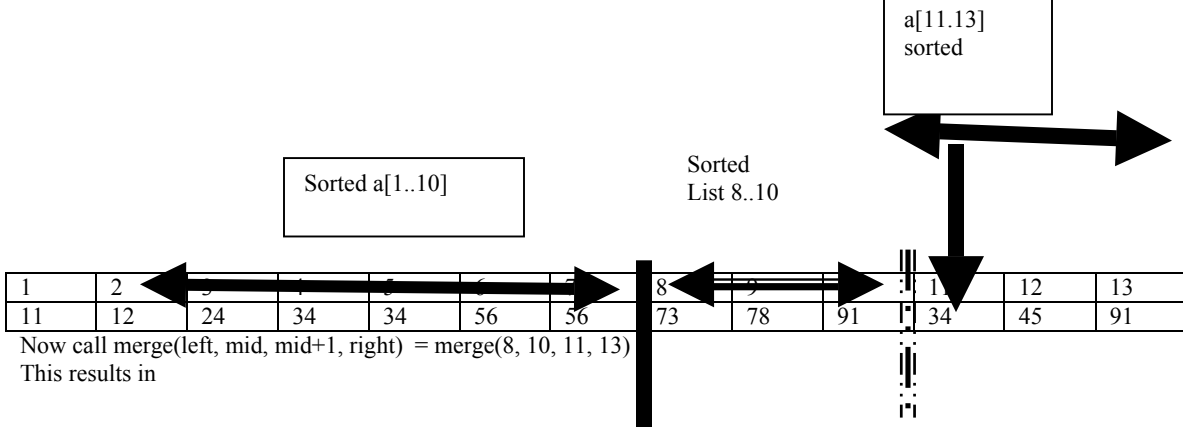
This results in



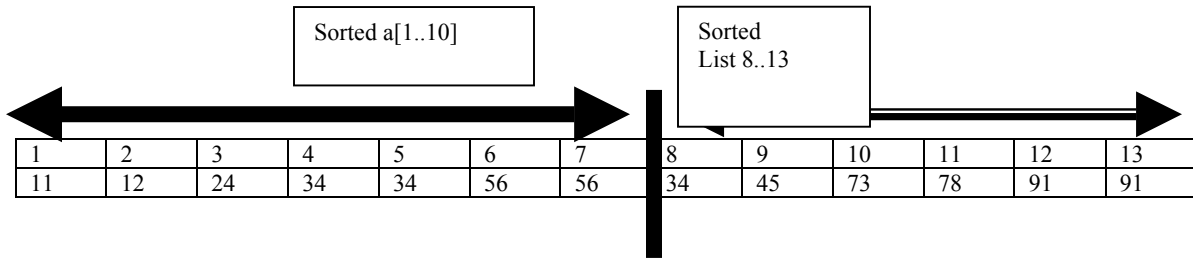
Control returns to mergesort which now calls mergesort(13,13), which call returns control without doing anything as one element is always sorted.



NOW CALL merge(left, mid, mid+1, right) = merge(11, 12,13,13) which results in the situation shown below



Now call merge(left, mid, mid+1, right) = merge(8, 10, 11, 13)  
This results in





		11

Compare (12,34) and send 11 to the output

Sorted input a[1..7]	Sorted inptua[8..13]	Sorted output a[1..13]
	34	
	45	
24	73	
34	78	
34	91	
56	91	
56		
		12
		11

Compare (24,34) and send 24 to the output

Sorted input a[1..7]	Sorted inptua[8..13]	Sorted output a[1..13]
	34	
	45	
	73	
34	78	
34	91	
56	91	
56		
		24
		12
		11

Compare (34,34) and send 34 to the output

Sorted input a[1..7]	Sorted inptua[8..13]	Sorted output a[1..13]
	34	
	45	
	73	
	78	
34	91	
56	91	
56		
		34
		24
		12
		11

Compare (56,34) and send 34 to the output

Sorted input a[1..7]	Sorted inptua[8..13]	Sorted output a[1..13]
	45	
	73	
	78	
	91	
56	91	
56		
		34
		34
		24
		12
		11

Compare (56,45) and send 45 to the output

Sorted input a[1..7]	Sorted inptua[8..13]	Sorted output a[1..13]
	73	
	78	
	91	
56	91	
56		
		45
		34
		34
		24
		12
		11

Compare (56,73) and send 56 to the output

Sorted input a[1..7]	Sorted inptua[8..13]	Sorted output a[1..13]
	73	
	78	
	91	
	91	56
56		45
		34
		34
		34
		24
		12
		11

Compare (56,73) and send 56 to the output

Sorted input a[1..7]	Sorted inptua[8..13]	Sorted output a[1..13]
	78	

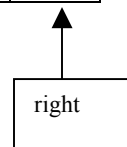
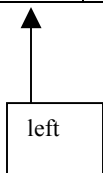
	91	73
	91	56
		45
		34
		34
		34
		24
		12
		11

Empty out the second file to the output to obtain the sorted file.

Sorted input a[1..7]	Sorted inptua[8..13]	Sorted output a[1..13]
		91
		91
		78
		73
		56
		45
		34
		34
		34
		24
		12
		11

THE FINAL SORTED FILE IS GIVEN BELOW

1	2	3	4	5	6	7	8	9	10	11	12	13
11	12	24	34	34	34	45	56	56	73	78	91	91



THE CALLING SEQUENCE

