WIA1002/WIB1002 Data Structures

Tutorial: Sorting and searching

1. Compare between linear search and binary search algorithms by searching for the numbers 45 and 54 in the following list :

3	**8**	**12**	**34**	**54**	**85**	**61**	**110**

Linear Search

- To search for the number 45, the linear search algorithm must compare every element in the list sequentially. The number of iterations is 8 because the number 45 is not in the list which later return -1 to indicate the element is not found in the list.
- To search for the number 54, the linear search algorithm needs to compare 5 elements where the array index 4 is returned. The number of iterations is 5.

Binary Search

Prerequisites: The list must be sorted either ascending or descending order.

The sorted list should be **3 8 12 34 54 61 85 110**

- To search for the number 45, The binary search algorithm begins by comparing the key with the middle element of the list which is 34 in this case where **list[3] = 34 (*list.length = 8, mid = (0 + 7) / 2 = 3, list[mid] = 34*)**. The specified number (key), 45 is greater than 34 so low pointer is updated to 4 where 3 (*mid*) + 1 so **list[4] = 54**. In the second loop, the mid index is recalculated **as *(4 + 7) / 2 = 5*** to obtain **list[5] = 61**. The key 45 is less than 61 so the high pointer is updated to mid - 1= 5 - 1 = 4. The third loop recalculates **mid = *(4 + 4) / 2 = 4*** to obtain **list[4] = 54**. Again, since 45 is less than 54, the high pointer is updated to 4 - 1 = 3. At this point, **high is 3** and **low is 4** terminates the loop and returns **-5 (*-insertion point-1*)** since the element is not found and the insertion point would be **index 4**. The number of iterations is 3.
- To search for the number 54, the binary search algorithm will take the middle element 34 as pivot point. Since the element 34 is less than the key 54, the low pointer is calculated as **3 (mid) + 1 = 4**. In the second loop, the middle element is recalculated, ***(4 + 7)/2 = 5*** so **list[5] = 61** compares with the key. Since 61 is greater than the key, the high pointer is adjusted to **3 - 1 = 4**. In the third loop, mid index is calculated using ***(4 + 4) / 2 = 4*** so **list[4] = 54** and compares with the key. The search ends with a match by returning the **index 4.** The number of iterations is 3.

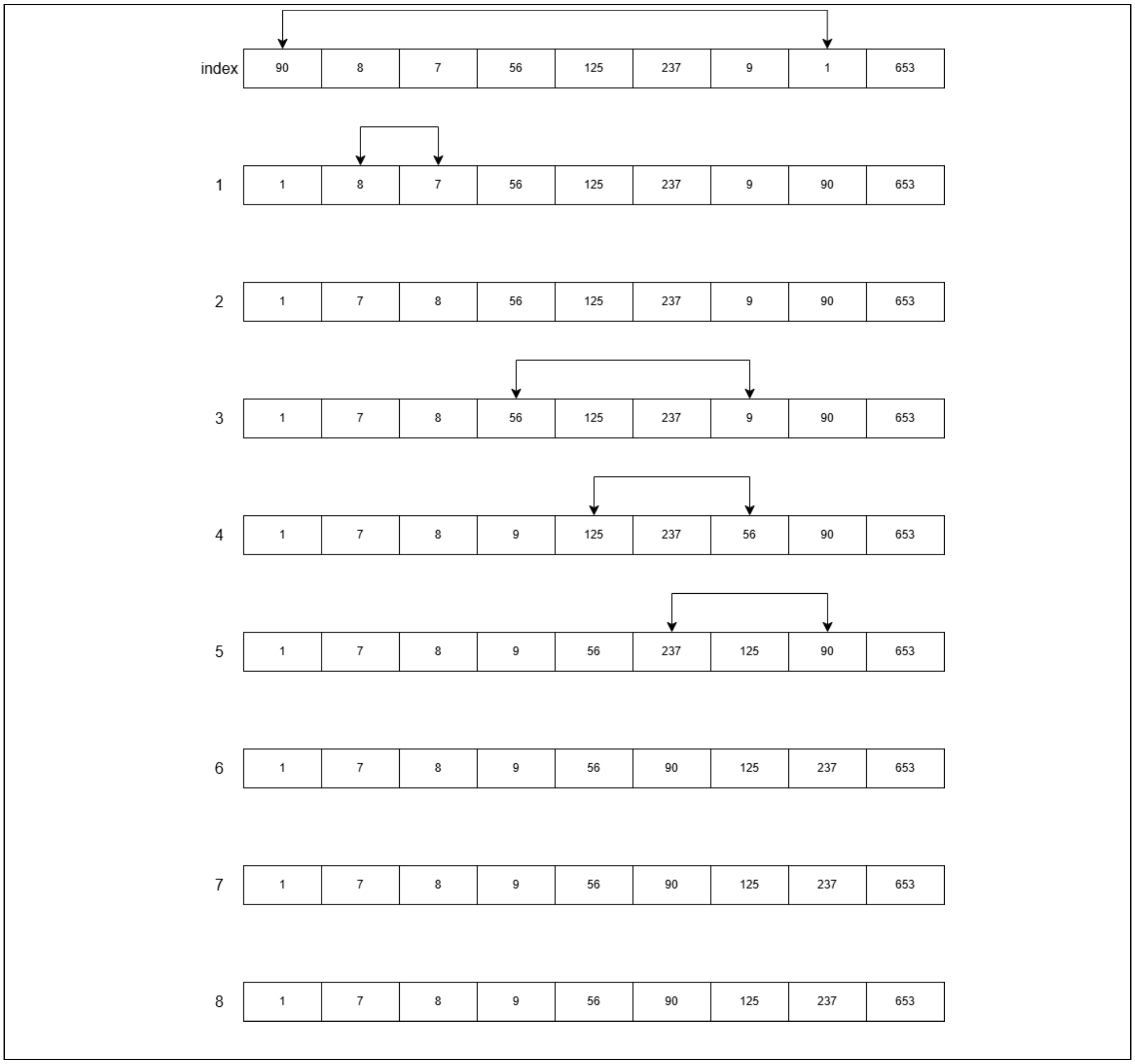
2. Describe the technique for each sort algorithm below. Given the following list:

90 8 7 56 125 237 9 1 653

Show a trace of execution for:

a. Selection sort
b. Insertion sort
c. Bubble sort
d. Merge sort

a. Selection sort



index									
	90	8	7	56	125	237	9	1	653
1	1	8	7	56	125	237	9	90	653
2	1	7	8	56	125	237	9	90	653
3	1	7	8	56	125	237	9	90	653
4	1	7	8	9	125	237	56	90	653
5	1	7	8	9	56	237	125	90	653
6	1	7	8	9	56	90	125	237	653
7	1	7	8	9	56	90	125	237	653
8	1	7	8	9	56	90	125	237	653

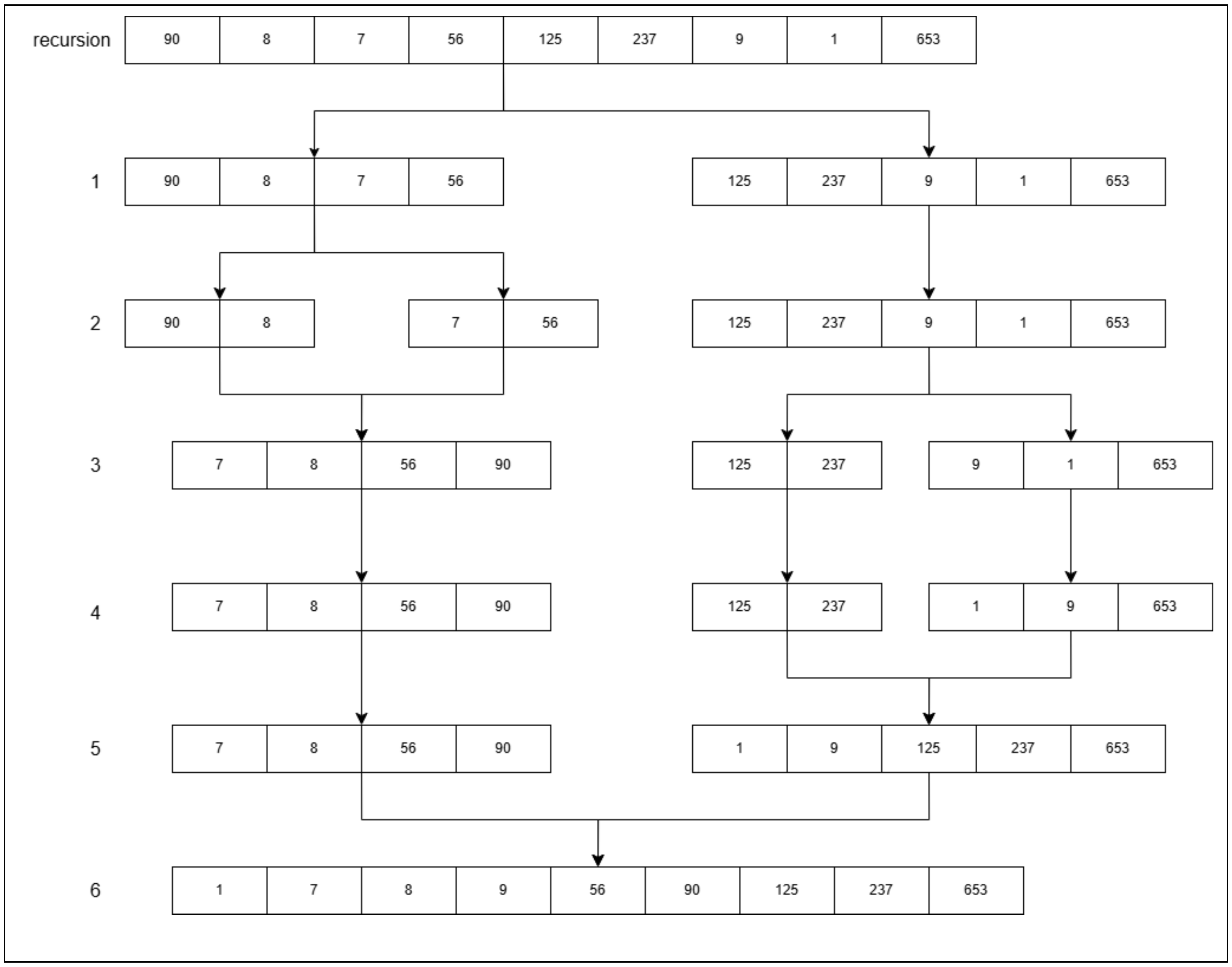
b. Insertion sort

iteration	90	8	7	56	125	237	9	1	653
1	8	90	7	56	125	237	9	1	653
2	7	8	90	56	125	237	9	1	653
3	7	8	56	90	125	237	9	1	653
4	7	8	56	90	125	237	9	1	653
5	7	8	56	90	125	237	9	1	653
6	7	8	9	56	90	125	237	1	653
7	1	7	8	9	56	90	125	237	653
8	1	7	8	9	56	90	125	237	653

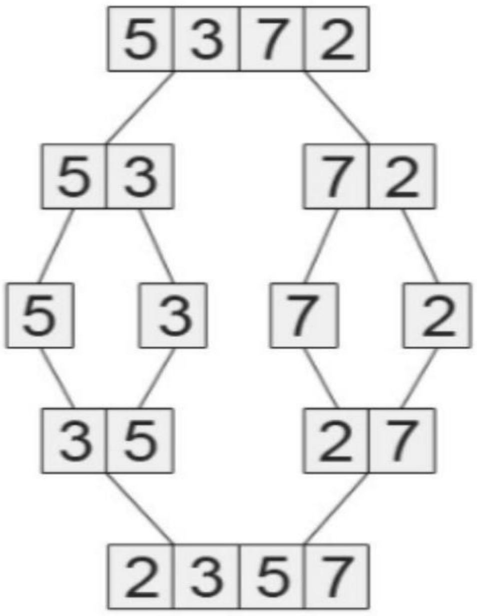
c. Bubble sort

Pass	90	8	7	56	125	237	9	1	653
1	8	7	56	90	125	9	1	237	653
2	7	8	56	90	9	1	125	237	653
3	7	8	56	9	1	90	125	237	653
4	7	8	9	1	56	90	125	237	653
5	7	8	1	9	56	90	125	237	653
6	1	7	8	9	56	90	125	237	653
7	1	7	8	9	56	90	125	237	653
8	1	7	8	9	56	90	125	237	653

d. Merge sort

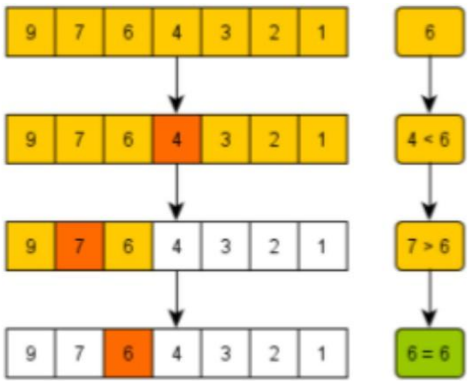

recursion
90 8 7 56 125 237 9 1 653
1
90 8 7 56
125 237 9 1 653
2
90 8
7 56
125 237 9 1 653
3
7 8 56 90
125 237
9 1 653
4
7 8 56 90
125 237
1 9 653
5
7 8 56 90
1 9 125 237 653
6
1 7 8 9 56 90 125 237 653

3. Which type of sort algorithm is this?



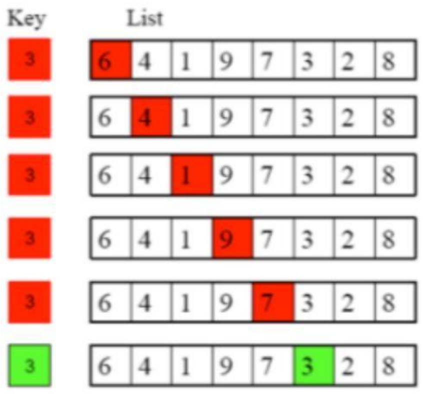
Merge sort

4. Which type of search algorithm is this?



Binary Search

5. Which type of search algorithm is this?



Linear Search