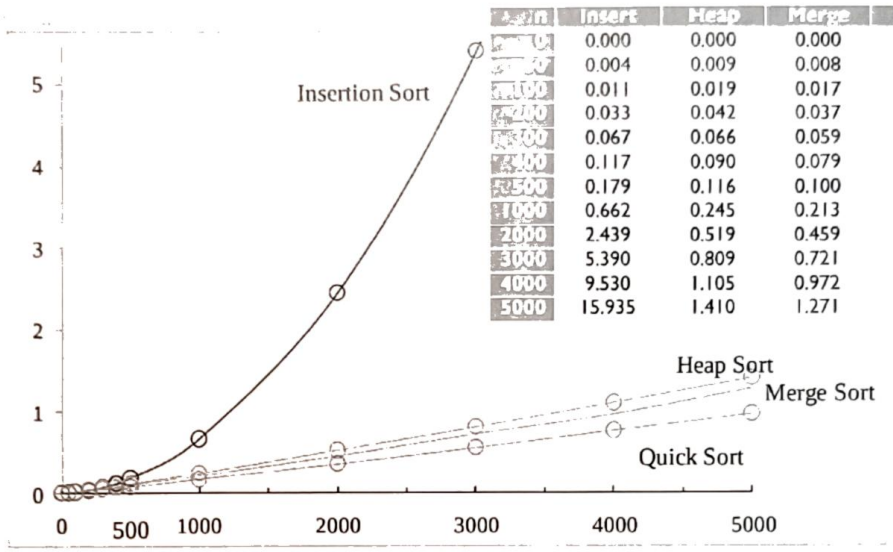
True or False

> For each question, simply answer T (True) or F (False). No explanation is required. (2 points for each question)

T 1. Based on the runtime comparison presented in the textbook, quicksort demonstrates the fastest performance for large datasets. Therefore, it is advisable to utilize quicksort consistently to enhance efficiency when dealing with extensive data sizes.



n	Insert	Heap	Merge	Quick
0	0.000	0.000	0.000	0.000
50	0.004	0.009	0.008	0.006
100	0.011	0.019	0.017	0.013
200	0.033	0.042	0.037	0.029
300	0.067	0.066	0.059	0.045
400	0.117	0.090	0.079	0.061
500	0.179	0.116	0.100	0.079
1000	0.662	0.245	0.213	0.169
2000	2.439	0.519	0.459	0.358
3000	5.390	0.809	0.721	0.560
4000	9.530	1.105	0.972	0.761
5000	15.935	1.410	1.271	0.970

T 2. For sorted arrays, the most efficient method to search for an element requires $\Omega(\log_2 N)$ time.

F 3. Insertion sort and quick sort are both stable sort, while merge sort and heap sort are not.

F 4. SHA-1 is the abbreviation of Secure Hash Function. It is "secure" since it's impossible to find collisions.

T 5. Dynamic hashing helps minimize the overhead associated with rebuilding hash buckets in the event of an overflow.

F 6. Searching for an element in an AVL tree guarantees an $\Theta(N\log N)$ time complexity, where N is the total elements currently in the AVL tree.

T 7. There are total 4 possible unbalanced AVL tree cases, which are LL, RR, LR, RL.

T 8. If **T** is a B-tree of order 128, then each node can have 64 ~ 128 children. $\lceil \frac{128}{2} \rceil = 64$

F 9. Suppose a Bloom filter employs *k* hash functions; as *k* increases, the false positive rate of the Bloom filter decreases.

T 10. A balanced tree should have at least one node, which serves as the root.

Basic Questions

> For each question, please give your answer with a short but concise explanation. Answers without explanations will get 0 points. (6 points for each question)

11. Apply merge sort to arrange the elements of the array [3, 9, 1, 2, 7, 13, 0] in asending order. You may use either iterative merge sort or recursive merge sort. Provide the updated sublists after each step to receive full marks.

12. Apply quicksort to organize the elements of the array [3, 9, 1, 2, 7, 13, 0] in asending order. Provide the updated array after each step to receive full marks.
13. Provide an example input array demonstrating that heap sort is not stable. The input array should contain at least 5 unique elements. Display the heap structure after each step to earn full marks.
14. Assuming a chained hash table is utilized in a hashing scheme H with a total of **N** elements distributed across **M** buckets, what is the worst-case time complexity of searching for an element in **H**? Provide an example of **H** and **N** to support your answer.
15. Insert the elements [3, 9, 2, 6, 7, 10, 9] sequentially into an initially empty 2-3 tree. Provide the updated tree structure after each insertion to receive full marks.

Application Questions

For each question, please give your answer with a short but concise explanation. Answers without explanations will get 0 points. (10 points for each question)

16. Assuming we have a large array with N (N >> 1000) integers to sort, and we know beforehand that these N integers only fall within the range of 0 to 999. Please devise an efficient method to sort this array with O(N) time complexity and O(1) extra space.
17. Please design a method for stable radix sort. To receive full marks, ensure a clear explanation of the method and comprehensive coverage of all corner cases.
18. Sequentially insert the elements [6, 18, 17, 12, 6, 14, 20, 16] into an initially empty AVL tree. After completing all insertions, remove the element [12] from the tree. Display the tree's updated structure after each step to receive full marks.
19. Consider the hash function **H(x) = x mod 10**, where there are 10 buckets. Assume each time *x* is a randomly chosen 16-bit unsigned integer. Identify the bucket(s) with the highest probability of overflowing. If multiple buckets have the same probability, please list all of them.
20. Assuming an array *Arr* of size N containing N unique integers in the range [1, N+1], determine the missing integer in [1, N+1] using O(1) extra space with the smallest time complexity for the following cases: (a) If *Arr* **is** sorted. (b) If *Arr* **is not** sorted. To receive full marks, provide your method for each constraint along with their respective time complexities.