

**Symbiosis Institute of Technology**

**Faculty of Engineering**

**CSE- Academic Year 2023-24**

**Data Structures – Lab Batch 2022-26**

|  |  |
| --- | --- |
| **Lab Assignment No:- 1,2,3** | |
|  | |
| **Name of Student** | Prabhat Shankar |
| **PRN No.** | 22070122147 |
| **Batch** | 2022-2026 |
| **Class** | CSE-B3 |
| **Academic Year & Semester** | 2023 and 2nd semester |
| **Date of Submission** | 28th august,2023 |
|  | |
| **Title of Assignment:** | A. Implement following searching algorithm: Linear search with multiple occurrences  B. Implement following searching algorithms in menu:  1. Binary search with iteration  2. Binary search with recursion |
| **Theory:** | 1. Prepare table for following searching and sorting algorithms for their best case, average case and worst case time complexities.   Linear search, binary search, bubble sort, Insertion sort, selection sort, merge sort, quick sort.   1. Discuss on Best case and Worst case time complexities of   Linear search, binary search, bubble sort, Insertion sort, selection sort, merge sort, quick sort.    linear search: best case element found at first position  worst case:element found at end or not found  binary search:best case when element found at middle position  worst case: when element not found.  Bubble Sort:  Best Case: Think of sorting numbers like arranging toys by size from small to big. If they're already in the right order, you don't need to move them around much. That's the best case of bubble sort – when things are already sorted, it's easy and quick.  Worst Case: But if you have to sort toys from biggest to smallest and can only swap adjacent toys, you might have to do a lot of swapping to get them in order. That's the worst case of bubble sort, where you have to do lots of swaps and it takes more time.  Insertion Sort:  Best Case: Imagine you're playing cards and your hand is almost sorted. You just need to insert one card in the right spot. That's like the best case of insertion sort – it's quick when things are already close to being sorted.  Worst Case: But if you're playing cards and your hand is a big mess, you might have to keep moving cards around to get them in order. That's the worst case of insertion sort – it takes more time when things are all jumbled up.  Selection Sort:  No Best Case: Selection sort works the same no matter what. You're always picking the smallest thing and putting it in the right place, no matter how things are to start with.  Worst Case: It's like if you have a pile of numbered cards, and you keep picking the smallest number and putting it in a new pile. You keep doing that until all the cards are in order. It takes a bit more time, no matter what.  Merge Sort:  Best Case: Think of sorting like folding clothes. If you have a bunch of clothes and you keep folding them in half and half, it doesn't take too long. That's the best case of merge sort – things get neatly divided, and it's not too slow.  Worst Case: Even if you have a lot of clothes, if you keep folding them in half and then combining them, it's still not too bad. That's the worst case of merge sort – it's pretty good at organizing things even when there's a lot.  Quick Sort:  Best Case: Imagine you're sorting numbered cards by picking a middle number as your "pivot." You put smaller numbers on one side and bigger numbers on the other. If you keep doing that, it's pretty fast. That's the best case of quick sort.  Worst Case: But if you're unlucky and keep picking bad pivots, you might have to do more work to get things sorted. It's like sorting cards where each time your pivot isn't helping much. That's the worst case of quick sort, when it takes more time. |
| **Source Code/Algorithm/Flow Chart:** |  |
| **Output Screenshots (if applicable)** |  |
| **Conclusion** | Thus we have studied different sorting algorithms and their time complexities. |