DSALG – Data Structures and Algorithms

Worksheet 2: Searching and Sorting

Introduction

This worksheet is designed to give you practice in implementing iterative sorting algorithms, and iterative/recursive searching algorithms particularly on a sorted array where we have O(1) complexity for accessing an element with its index known.

The implementation of each algorithm is not unique. Feel free to add your own solutions to the exercises. Please do have an evaluation of the BigO (time complexity and space complexity) of your solution.

Hint: Don’t get stuck in any particular exercise for too long. Please set a maximum time spent on each exercise. You are encouraged to attempt most exercises and leave the tricky ones for a discussion with me during the practical sessions or the drop-in sessions. A short discussion will help sort the question and clear up your concerns quickly.

Programming exercises

1. Implement the Selection Sort:

   Recall that for each loop, the Selection Sort will find the minimum/maximum of the unsorted part and put it to the end of the sorted part, until the whole array is sorted.

2. Implement the Insertion Sort:

   Recall that for each loop, the Insertion Sort will include one element to the sorted part from the unsorted part and insert it to its correct position, until the whole array is sorted.

3. Implement the Bubble Sort:

   Recall that for each loop, the Bubble Sort will step through the unsorted part from the first element, compare the adjacent elements, and swap them if they are in the wrong order (at the end of each loop, the last element of the unsorted part is in its final correct position and excluded from the unsorted part), until the whole array is sorted.

4. Implement the Linear Search for an Array/List.

5. Implement the Binary Search for a Sorted Array using Iterative definition.

6. Implement the Binary Search for a Sorted Array using Recursive definition and compare the performance with the Iterative implementation on an arbitrary sorted array.

7. (Advanced) Implement the Ternary Search (different from dividing the elements into two equal halves by Binary Search, Ternary Search divides the elements into 3 equal parts) for a Sorted Array. Evaluate the performance of ternary search on an arbitrary sorted array, in comparison with binary search.

8. (Advanced) Further to Ex 7, implement the N-ary Search for a Sorted Array, where N can be any positive integer. Justify why binary search is better on average with either evaluation on arbitrary sorted arrays or written analysis.