ADS 2023 spring Week 4 Exercises

Exercises for Algorithms and Data Structures at ITU. The exercises are from *Algorithms, 4th Edition* by Robert Sedgewick and Kevin Wayne unless otherwise specified. Color-coding of difficulty level and alterations to the exercises (if any) are made by the teachers of the ADS course at ITU.

2.1.1 Selection Sort - Green Show in the style of the example trace with Algorithm 2.1 from the book [SW 2.1], how selection sort sorts the array `E A S Y Q U E S T I O N`.

2.1.4 Insertion Sort - Green Show in the style of the example trace with Algorithm 2.2 from the book [SW 2.1], how insertion sort sorts the array `E A S Y Q U E S T I O N`.

2.2.2 Merge Sort - Green Show in the style of the example trace with Algorithm 2.4 from the book [SW 2.2], how top-down mergesort sorts the keys `E A S Y Q U E S T I O N`.

2.1.6 Identical Keys - Green Which method runs faster for an array with all keys identical, selection sort or insertion sort?

2.1.7 Reverse Order - Green Which method runs faster for an array in reverse order, selection sort or insertion sort?

2.1.8 Random Order - Yellow Suppose we use insertion sort on a randomly ordered array where items have only one of three values. Is the running time linear, quadratic or something in between?

2.2.15 Bottom-up Queue Mergesort - Yellow Explain how to take two queues of sorted items and merge them into a single queue which is in sorted order.

2.1.13 Deck Sort - Yellow Explain how you would put a deck of cards in order by suit (in the order spades, hearts, clubs, diamonds) and by rank within each suit, with the restriction that cards must be laid out face down in a row, and the only allowed operations are to check the values of two cards and to exchange two cards (keeping them face down).

2.2.17 Linked-list sort - Yellow Implement a natural mergesort for linked lists. Use only constant amount of extra space and linearithmic time.

2.1.14 Dequeue Sort - Red Explain how you would sort a deck of cards, with the restriction that the only allowed operations are to look at the values of the top two cards, to exchange the top two cards, and to move the top card to the bottom of the deck.

2.2.16 Natural merge sort - Red Write a version of bottom-up mergesort that takes advantage of order in the input array by using the following idea: Use one additional array of the same length as the input. Work in rounds, in each round repeatedly identify two sorted subarrays in one array by scanning (incrementing a pointer as long as the pointed at cell and its neighbor are sorted), then merge these into the other array. Analyze the running time in terms of the array length and the number of maximal increasing sequences in the array.

2.2.21 (variant) Triplicates - Red Given three lexicographically sorted lists of n strings with a total of m characters. Devise an algorithm that finds the lexicographically smallest string that is contained in all lists. How many string comparisons does your algorithm perform? How many character comparisons does your algorithm perform?