Tutorial (week 8)

All the answers can be given with pseudocode or with Python. Remember that most of the time, many solutions are possible.

1. Consider the following list $L = [54, 26, 93, 17, 77, 31, 44, 55, 20]$. when applying a recursive merge-sort algorithm on L, explain with a binary tree the successive recursive calls and with another binary tree the successive merge processes.

2. Give a Python or pseudocode description of the merge-sort algorithm assuming the input is given as a linked list.

3. Describe a variation of the merge-sort algorithm that is given a single array, S, as input, and uses only an additional array, T, as a workspace. No other memory should be used other than a constant number of variables.

4. Let A be a collection of objects. Describe an efficient method for converting A into a set. That is, remove all duplicates from A. What is the running time of this method ?

5. Suppose we are given two n-element sorted sequences A and B that should not be viewed as sets (that is, A and B may contain duplicate entries). Describe an $O(n)$-time method for computing a sequence representing the set $A \cup B$ (with no duplicates).

6. Suppose we are given a sequence S of n elements, each of which is colored red or blue. Assuming S is represented as an array, give an in-place method for ordering S so that all the blue elements are listed before all the red elements. Can you extend your approach to three colors?

7. Suppose we are given an n-element sequence S such that each element in S represents a different vote in an election, where each vote is given as an integer representing the ID of the chosen candidate. Without making any assumptions about who is running or even how many candidates there are, design an $O(n \log n)$-time algorithm to see who wins the election S represents, assuming the candidate with the most votes wins.

Hints

- **Question 3: Avoid the use of recursion, by doing things bottom-up by levels, and use the auxiliary array as a "buffer" so as to swap S and T with each level.**
- **Question 4: Sort first**
- **Question 7: Sort first, choosing the appropriate key for comparisons.**