Self-Practice Week 4 - Sorting (part 2)

Applications of Sort

The goal of this assignment is to use sorting algorisms to solve a variety of problems.

Exercise 1 – Permutations

Given two integer arrays of size `n`, design an algorithm to determine whether one is a permutation of the other (i.e., they contain exactly the same entries but, possibly, in a different order). Your algorithm should have guaranteed sub-quadratic performance in the worst-case scenario.

Exercise 2 – Triplicates

Given 3 arrays of `n` strings each, design a guaranteed linearithmic (i.e., O(`n`log`n`)) algorithm to determine if there is any string that is common to all three. Return such string.

Exercise3 – Set Intersection

Given two arrays `a[]` and `b[]`, each containing `n` distinct 2D points in the plane, design a subquadratic algorithm to count the number of points that are contained both in array `a[]` and `b[]`.

Exercise 4 – Idle Times

Consider a machine that needs to process `n` jobs. Design and implement an algorithm that, given the list of `n` jobs with their start and end times, determines the largest interval where the machine is idle, and the largest interval where the machine is not idle.