TA Lab Week 3 – Sort

The goal of this lab is to tackle the sorting problem (and analyse its complexity) in different variations, using some of the most common sorting algorithms.

Exercise 1 – Anagrams

Given a list of words, design and implement an efficient algorithm to group all anagrams together. Words w1 and w2 are said to be anagrams if by rearranging the letters of w1, we can get w2 using all the original letters of w1 exactly once.

Example:
Given the list of words [eat, tea, part, ate, trap, pass], your algorithm should output:
[eat, tea, ate]
[part, trap]
[pass]

Hint
First sort each individual word (rearrange its characters alphabetically), then sort the list of rearranged words.

Exercise 2 – Separate positive and negative numbers

Given a list of both positive and negative numbers in random order, design and implement an efficient algorithm to rearrange the array elements so that positive and negative numbers are placed alternatively, starting from a positive number, and so that positives are sorted and negatives are sorted. If there are more negative or positive numbers, they should be placed at the end of the rearranged list (for this latter part, the order in which numbers appear is not important).

Example:
Given [-8, 1, 2, -4, 6, 12, 5, -10, 16, 7, 11], the algorithm should output:
[1, -4, 6, -8, 12, -10, 5, 2, 16, 7, 11]

Hint
You can do this using a modified version of insertion sort in $O(N^2)$, and of mergesort in O(N log N). If you only care about alternating +/- without sorting them, try using Quicksort' pivoting method (see next week lectures) to elegantly do it in O(N).

Exercise 3 – Sort distances for K-nearest-neighbours classifier

K-nearest-neighbours (KNN) is a very common machine-learning technique used to classify a new data point x, based on the classes that the K data points nearest to x belong to. The algorithm works by first determining the K (already classified) data points nearest (i.e., most similar) to x; then taking a "majority vote" from these K points; and, finally assigning to x the class that the majority of its K nearest neighbours belong to. Implement some of the sorting algorithms seen at lectures, to efficiently find the most similar data points (i.e., the nearest neighbours), out of a dataset of potentially millions of datapoints. Analyse the performance of the sorting algorithms on two different applications: classifying apples based on size and weight, and classifying images of hand-written digits. Note: the classification framework, in which the sorting algorithms will be applied, is already provided in the notebook.