German International University
Informatics and Computer Science
Prof. Dr. Slim Abdennadher

Data Structures and Algorithms, Winter semester 2021
Practice Assignment 2

Exercise 2-1

Sorting Algorithms

Given the following array of integers:

11	5	14	10	2

Apply each of the following methods to sort this array:

- Bubble Sort
- Selection Sort
- Insertion Sort

In each step, you are required to:

- clearly describe the current step being performed
- show the resulting contents of the array

Please note that you are **not** required to write a program.

Exercise 2-2 Recursive Bubble Sort

Re-implement bubble sort algorithm such that you compensate for the outer loop with recursion.

Exercise 2-3 Modified Selection Sort

In the lecture you were introduced to the *Selection Sort* algorithm. For this exercise, you need to implement a modified version of the *Selection Sort* algorithm, where:

- instead of finding the smallest element in the array and changing its position with an element at the beginning of the array,
- you need to find **both** the smallest and the largest elements and change their positions with elements at the beginning and end of the array respectively.

You are required to select the smallest and largest numbers in one pass; in other words, your implementation should only include a *single loop* for finding both of the smallest and largest numbers.
Remark: There are some special cases that you need to figure out and handle in your implementation. Make sure that your solution works for the arrays:

`{6, 4, 5}` and `{2, 6, 8, 1, 7, 9, 4, 3, 5}`

What is the invariant of this sorting algorithm?

To be solved in tutorial

Exercise 2-4 Index Sort

a) Write a method `indexSort` that uses the following algorithm to sort an array `x` of integers. Loop through the integers in `x`. For each integer, calculate what index it will be stored at in the output array `y` of the same length as `x`.

   For example, for the following array:

4	3	8	2

   by comparing `4` with the rest of the array, we can deduce that `4` should be stored at index `2`. Thus, after the first pass the resulting array should be of the following form:

		4	

   Make sure that your algorithm handles arrays with **duplicate keys**.

b) What is the time complexity of your algorithm?

c) What is the invariant of your algorithm?

   To be solved in tutorial

Exercise 2-5 Shaker Sort

Shaker Sort is a simple optimisation of the `Bubble Sort`, that performs passes in both directions (simultaneous-ly), allowing out-of-place items to move fast across the whole array.

`Shaker Sort` performs the following steps:

1. Examines pairs of items, starting from the beginning of the array and performs swapping if necessary.
2. Examines pairs of items, starting from the end of the array and performs swapping if necessary.
3. Repeats the previous two steps until the array is sorted.

Consider the following array:

7	5	11	10	8

Example:

a) From the beginning:

   - Examine a[0] and a[1].
   - Compare the contents of this pair (7 and 5): swap contents. Resulting array:

5	7	11	10	8

   - Examine a[1] and a[2], compare the contents of this pair (7 and 11): do not swap contents.
   - Examine a[2] and a[3], compare the contents of this pair (11 and 10): swap contents. Resulting array:

5	7	10	11	8

   - Examine a[3] and a[4], compare the contents of this pair (11 and 8): swap contents. Resulting array:

5	7	10	8	11

b) From the end:

   - Examine a[4] and a[3], compare the contents of this pair (11 and 8): do not swap contents.
   - Examine a[3] and a[2], compare the contents of this pair (8 and 10): swap contents.

   Repeat until array is sorted.

Write a method `shakerSort` to implement this algorithm.

To be solved in lab

Exercise 2-6 Counting Sort

a) Write a method `booleanSort` that uses the following algorithm to sort an array `x` of **positive** integers. It should use another `boolean flag` array of as many elements as the maximum value in array `x` plus `1`. This `boolean` array will be used to denote the presence of the elements of array `x` if found. Thus the algorithm, should loop through the integers in `x`. For each integer `i` of the array, the flag of index `i` of the `boolean` array should be set to `true`. Then the algorithm should reconstruct the sorted array from the `true` values of `flag` depending on their locations.

b) How can you modify your algorithm to handle arrays with duplicate values? (Hint: Call it `countingSort`!)

c) What is the time complexity of your algorithm?

d) What are the drawbacks of the algorithm described above?

e) What is the invariant of your algorithm?

To be solved in lab

Exercise 2-7 Student Records

You are required to implement a sorting algorithm and use it to sort some student records.

Each record is composed of:

- Name
- Student ID
- GPA

Your program should sort a list of students, according to their names.

Exercise 2-8

Midterm Exam 2013

Suppose that while your computer is sorting an array of objects, its memory is struck by a cosmic ray that changes exactly one of the keys to something completely different. For each of the sorting algorithms discussed in the lectures, what is the worst-case possibility in terms of the ordering of the final array?

For each sorting algorithm the final array could

x) not even close to sorted,

y) have just one or two keys out of place, or

z) consist of two separate sorted subsets, one following the other, plus perhaps one or two additional keys out of place.

Choose x), y), or z) for the following algorithms. Justify your answers by examples.

a) Insertion sort

b) Selection sort

c) Bubble sort

Exercise 2-9

Midterm Exam 2015

Suppose we have an array a containing n integers. Initially the integers are not sorted. We would like to rearrange the array in ascending order using the following **Bogosort** algorithm.

Bogosort simply shuffles elements of the array randomly until it is sorted. The Bogosort is only a theoretical concept, which has no use in practical applications. The Bogosort is also know as the Slow Sort or Stupid Sort.

Algorithm in Pseudocode:

```
bogoSort(array)
   while !isSorted(array)
      shuffle(array)
```

a) Write a Java method `void bogoSort(int[] a)` that takes an array a and sorts the array based on the above described algorithm. Please note that the shuffling operation should be done in every pass for all numbers of the array randomly. You are requested to implement the `shuffle` and the `isSorted` methods.

b) Given an intially unordered array

1. What is the best case complexity of the algorithm. Justify your answer.
2. What is the worst case complexity of the algorithm. Justify your answer.
3. What is the invariant of the algorithm? Justify your answer.