Data Structures
Homework 5

Subjects:
Sorts

Honor code:
1. Do not copy the answers from any source
2. You may work in small groups but write your own solution (mention this in the homework)
3. Cheating students will face Committee on Discipline (COD)

Submission date - 17/5 - through moodle

In this homework you'll implement 3.5 sorts (what is half a sort you ask yourself? well you'll implement two versions of the Merge sort…)
You'll deal with natural numbers (positive integers) including 0.
The sorts are
1. Merge sort
2. Merge sort with modification - you'll split the array to 3 equal parts instead of 2
3. Counting sort -
    a. You should calculate the range for the input array yourself
4. Bucket sort -
    a. You should calculate the range for the input array yourself.
    b. Number of buckets will be as the size of the input array (each bucket will hold a linked list as we learned in the class)

After you'll implement the the sorts you'll need to test them and answer the theoretical questions with the help of the

You are given a very simple interface to implement:
public int[] sort(int [] input)

You will fill 5 files:

1. MergeSort.java
2. MergeSortModified.java
3. CountingSort.java
4. BucketSort.java
5. Theoretical.txt

These files will already contain:

1. Frame - e.g. public class MergeSort implements Sort…
2. Default constructor

You'll get another 4 files -

1. SortTest.java - simple test for the sorts
2. SortTestOutput.txt - output of SortTest.java
3. SortsTimingTest.java - will provide you the data needed to answer the theoretical part
4. ArrayFactory.java - helper class

Theoretical part:

For each of the sorting algorithms we define $f(n)$ to be the average running time in seconds of the sorting algorithm.

We define $n_1 = 500000$ and $n_2 = 1000000 = 2n_1$.

The supplied main function (in SortsTimingTest.java) Calculates an estimate of $f(n_1)$ and $f(n_2)$ for all the sorts above and outputs the results. Use these results to answer the following questions. (The estimate calculation of $f(n)$ is based on 100 random arrays as you'll see in the code)

1. In class we learned that the average time complexity of Merge Sort (regular) is $\Theta(n\log n)$. We can strengthen this result and show that average time complexity of Merge Sort is approximately $cn\log n$ for some constant $c > 0$. Assume there exists a constant d that represents the time it takes in seconds for your computer to complete a single instruction. We get that for large values of n we have: $f(n) \approx dcn\log(n)$.
    a. Using the theoretical estimation $f(n) \approx dcn\log(n)$, What is $f(n_1) / f(n_2)$?
    b. What is the value of $f(n_1) / f(n_2)$ based on the values of $f(n_1)$ and $f(n_2)$ calculated by our java program?
    c. What is the relative error between the theoretical estimation of $f(n_1) / f(n_2)$ and the calculated estimation of $f(n_1) / f(n_2)$?

       Remark: The relative error between two values $a, b > 0$ is $\frac{|a-b|}{\min\{a,b\}}$.
2. You can easily calculate that the complexity of the Modified Merge Sort in $\Theta(n\log n)$ - to be more precise it's - log of base 3 (not 2). We can strengthen this result and show that average time complexity of is approximately $c * nlog_3 n$ for some constant $c > 0$. Assume there exists a constant d that represents the time it takes in seconds for your computer to complete a single instruction. We get that for large values of n we have: $f(n) \approx d * c * nlog_3 n$
    a. Using the theoretical estimation $f(n) \approx d * c * nlog_3 n$, What is $f(n_1) / f(n_2)$?
    b. What is the value of $f(n_1) / f(n_2)$ based on the values of $f(n_1)$ and $f(n_2)$ calculated by our java program?

   c. What is the relative error between the theoretical estimation of $f(n_1)\,/\,f(n_2)$ and the calculated estimation of $f(n_1)\,/\,f(n_2)$?

3. In class we learned that the average time complexity of Bucket Sort is $\Theta(n)$ (when the distribution is uniform - and we will assume that this is the case as we generate random input). We can strengthen this result and show that average time complexity of Bucket Sort is approximately cn for some constant $c > 0$. Assume there exists a constant d that represents the time it takes in seconds for your computer to complete a single instruction. We get that for large values of n we have: $f(n) \approx dcn$.
   a. Using the theoretical estimation $f(n) \approx dcn$, What is $f(n_1)\,/\,f(n_2)$?
   b. What is the value of $f(n_1)\,/\,f(n_2)$ based on the values of $f(n_1)$ and $f(n_2)$ calculated by our java program?
   c. What is the relative error between the theoretical estimation of $f(n_1)\,/\,f(n_2)$ and the calculated estimation of $f(n_1)\,/\,f(n_2)$?
4. In class we learned that the average time complexity of Counting Sort is $\Theta(n)$ (well actually it's $\Theta(n+k)$ where k is the range but in our case the range will be O(n). We can strengthen this result and show that average time complexity of Counting Sort is approximately cn for some constant $c > 0$. Assume there exists a constant d that represents the time it takes in seconds for your computer to complete a single instruction. We get that for large values of n we have: $f(n) \approx dcn$.
   a. Using the theoretical estimation $f(n) \approx dcn$, What is $f(n_1)\,/\,f(n_2)$?
   b. What is the value of $f(n_1)\,/\,f(n_2)$ based on the values of $f(n_1)$ and $f(n_2)$ calculated by our java program?
   c. What is the relative error between the theoretical estimation of $f(n_1)\,/\,f(n_2)$ and the calculated estimation of $f(n_1)\,/\,f(n_2)$?
5. In order to estimate the average running time of Merge Sort on arrays of size n we averaged the running time of Merge Sort on many random arrays of size n. Could we get a good estimation of the average time complexity of Merge Sort just by taking the running time of Merge Sort on a single fixed array of size n ? Explain.
   Hint: Read the pseudo-code of merge-sort given in the lecture (the functions MergeSort and Merge). Consider the following question: Let n be a fixed positive integer. When merge-sort is given as input an array of size n, does the run time on the input depend on the values of the keys?

Implementation instructions:

1. **You shouldn't use any java ready DS (Vector, Set, …). Do not import unnecessary packages. If you have question about this one - approach the TA. You can make one exception for the bucket sort - will be explained further**
2. You may add classes and methods that were not defined in the given API, as long as they conform to proper Object Oriented Design.
3. Do not change the given method signatures.
4. All the sorts **must meet the complexity and space requirements** as learned in the class
5. You can assume that the input is valid - valid arrays with natural numbers.
6. You can use ArrayList for the implementation of the Bucket Sort, only to spare you an implementation of a Linked List. Though you can of course implement Linked List yourself

Submission instructions:

1. The submission is in pairs (not obligatory but highly recommended).
2. If you submit in pari the submission file will be ID_ID.zip (e.g. 123456789_987654321.zip). If you submit by yourself the file will be named ID.zip (e.g. 123456789.zip)
3. Don't create any directories in the **zip** - it should include just the java files of the solution
4. If your code won't compile - it won't be checked
5. In the check process we will unzip your file and replace the SortTest.java with another set of tests.

Grading

● Correctness:	50%
● Efficiency:	10%
● Robustness:	10%
● Architecture & Design:	10%
● Documentation & Readability:	10%
● Submission instructions	10%