Seoul National University

Data Structure

Fall 2017, Kang

Programming Assignment 4: Internal Sorting (Chapter 7)

Due: November 28, 02:00 pm, submit at eTL

Reminders

- The points of this homework add up to 100.
- Like all homeworks, this has to be done individually.
- Lead T.A.: Jun-gi Jang (elnino9158@gmail.com)
- Write a program in Java.
- Do not use Java Collection Framework and the third-party implementation from the Internet.

1. How to submit the programming assignment

1) Create a ***JAR*** file including 'src' folder that contains your sources files, but without 'release' folder. (Refer to '1 – Introduction.pptx' in the first lab session)
   - We will run your ***Main*** class in the JAR file to grade your programming assignments. Before submitting the JAR file, please check if your Main class in the JAR file works correctly.
   - The submitted JAR should contain your sources files. If you don't use Eclipse, please see https://www.clear.rice.edu/comp201/05-spring/info/jar.shtml to create a proper JAR file.
   - You **MUST** obey the I/O specification of the programming assignment, and rules for the submission of the programming assignment.
   - Before submitting, check if your JAR file runs properly in your terminal with the following commands: "`java -jar ./PA_01_2015-12345.jar`".

2) Compress **the JAR file, and the project directory.** The readme file needs to include your student id, name, how to execute your program, and any other information TA needs to know.
   - The format of the JAR file is "*PA_##_(StudentID).jar*", where '##' is the programming assignment ID, and *(StudentID)* is your student ID.
     - ex) PA_01_2016-12345.jar
   - All documents should be written in English.

3) The name of the compressed file (zip file) should be '*PA_##_(StudentID).zip*' where '##' is the programming assignment ID, and (StudentID) is your student ID.
   - ex) PA_01_2016-12345.zip
     - ➢ *01: the first programming assignment*
     - ➢ *2016-12345: your student ID*

4) Submit the compress file to the eTL (http://etl.snu.ac.kr/) .

2. How to grade your programming assignment

1) Grade your programming assignment consists of three parts: ‘source code’ (50 pts), ‘program execution’ (50 pts).
2) We made a grading machine for the ‘program execution’ part. The machine will run your program, and compare answers and outputs that your program generates for given inputs. If your program cannot generate correct answers for an input file, it will not give you the point corresponding to the input. Our machine will consider the following scenarios:
   - (***Accept***) When your program generates exact outputs for an input file, the machine will give you the point of the input.
   - (***Wrong Answer***) When your program runs normally, but generates incorrect outputs for an input file, including typos, the machine will not give you the point of the input.
   - (***Run Error***) When your program does not run, or is terminated suddenly for some reasons, the machine will not give you the point of an input file because it cannot generate any outputs.
   - (***Time Limit***) When your program runs over a predefined execution time for an input file, our machine will stop your program, and it will not give you the point of the input. The time limit of the execution is ***7 seconds***.

3) We will generate 10 input files, and assign 5 points for each input file. For example, if your program gets 9 accepts, and 1 wrong answer by the machine, the points for ‘program execution’ part will be 45 points. Hence, before submitting your programming assignment, please be sure that your program makes correct answers in reasonable time for any input case.

3. Problem

How can we make the mergesort algorithm have better performance? One of the possible solutions is a hybrid sorting algorithm which combines the mergesort with insertion sorting algorithm. The hybrid sorting algorithm begins with the mergesort and switches to the insertion sorting algorithm when the size of subarray is equal to 32 (The size of the last subarray can be less than 32). This algorithm takes the advantages of both algorithms achieving practical performance on typical data sets. Especially, the hybrid sorting algorithm is better performance than the basic mergesort in the best case.

Our goal is to implement a program that sorts the given key-value pairs in lexicographic order using the hybrid sorting algorithm. Fill your code in 'HybridSorter.java' and 'InsertionSorter.java'. Several rules that you **must** follow are as follows:

- Make your program run through Main class. The main class should handle inputs and outputs using standard I/O in JAVA. (input from a keyboard, output to a monitor)
- The number of given key-value pair doesn't exceed 1,000,000.
- All keys are distinct.
- Your program should be finished within 7 seconds on the PC used for lab session for any test case.
- You should implement all functions listed in Section 4.

The hybrid sorting algorithm and the level to switch the sorting algorithm are important to ensure $O(n \log n)$ time complexity in the worst case, and better performance in the best case.

4. Interface of Algorithms

1) HybridSorter

Function
void sort(Pair<K, ?>[] array, int left, int right)

Description
- Sorts the elements in given array from left to right in lexicographic order using the hybrid sorting algorithm. - 'left' and 'right' are inclusive.

Function
Pair<K,?> search(Pair<K, ?>[] array, int k)
Description
- Find the pair which has lexicographically **k**-th smallest element in given array. - (k) is a parameter as integer.

2) InsertionSorter

Function
void sort(Pair<K, ? >[] array, int left, int right)
Description
- Sorts the elements in given array from left to right in lexicographic order using the insertion sort algorithm. - 'left' and 'right' are inclusive.

5. Specification of I/O

The program should accept only the inputs listed below and print the listed outputs. You **must** use standard I/O operations in Java, not file operations.

1) n

Input form	Output form
`n (#elements)`	

Description

- Creates the first array of size (#elements).
- 'n' command appears at the first line of input.
- 'n' command doesn't appear multiple times.

Example Input	Example Output
`n 3`	

2) append

Input form	Output form
`append (key) (value)`	

Description

- Appends a new pair of which key and values are (key), and (value), respectively.
- (key) is a string which doesn't contain any whitespace.
- (value) is a string.

Example Input	Example Output
`append data 2`	

3) sort

Input form	Output form
`sort`	

Description

- Sorts the pairs in the array in lexicographic order using the hybrid sorting algorithm.

Example Input	Example Output
`sort`	

4) search

Input form	Output form
`search (k)`	`Search: (k) (key) (value)`

Description
- Prints the (k)-th smallest key-value pair to the console.   - (k) is between 0 and length of the array – 1.

Example Input	Example Output
`search 0`	`Search: 0 data 2`

6. **Sample Input**

```
n 3

append hello 1

append data 2

append structure 3

sort

search 1

search 0

search 2
```

7. **Sample Output**

```
Search: 1 hello 1

Search: 0 data 2

Search: 2 structure 3
```