Seoul National University

Data Structure

Fall 2015, Kang

Programming Assignment 4 : Internal Sorting (Chapter 7)

Due: Dec. 1, 11:00 am, submit at eTL

Reminders

- The points of this homework add up to 100.
- Like all homeworks, this has to be done individually.
- Lead TA: Jinhong Jung (montecast9@gmail.com)
- Write a program in Java.
- Do not use Java Collection Framework.

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.
   - 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 command line: java -classpath ./PA_04_2015-12345.jar ds.test.Main.

2) Compress **the JAR file, a readme file and the project folder.** 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_04_2015-12345.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_04_2015-12345.zip
     - *04: the fourth programming assignment*
     - *2015-12345: your student ID*

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

2. How to grade your programming assignment

1) We made a grading machine to automatically grade your programming assignment. The machine will run your program, and compare answers and outputs that your program generates for given inputs. If your program can not 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 can not 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 ***5 seconds***.

2) We will generate 10 input files, and assign 10 points for each input file. For example, if your program gets 9 accepts, and 1 wrong answer by the machine, the total point will be 90 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

In this assignment, you should implement “**Quicksort**” algorithm. You will be given distinct integers, and you have to sort them in either descending or ascending order using the “**Quicksort**” algorithm. The specific operations you need to implement are as follows:

- Add or remove: You need to add an integer into an array. Also, you need to remove an integer from the array.
- Sort: You need to sort the array containing integers in either descending or ascending order.
- Find top-k: For an integer k, you need to find top-k largest or smallest integers in the array.

Here are several assumptions for clarity.

- All given integers are *distinct*. The maximum number of integers is 100,000.
- The range of an integer is between -2^{31} and $2^{31}-1$.
- The sorting and finding top-k operations are performed only when the ar ray is not empty.

When you decide a pivot in the “Quicksort” algorithm, ***randomly select an element*** from the array as the pivot. (Do not choose the middle element.)

Complete “Sorter” class supporting the above requirements. You need to fill in “Sorter.java” and “Main.java” of “PA_04” java project. You can modify the skeleton codes, or add classes if you want.

4. ADT of Data structure

1) add

Function

void add(int value)

Description

- This function adds (value) into the array.
- All given values are distinct.

3) remove

Function

boolean remove(int value)

Description

- This function removes (value) from the array.
- If the function removes the (value) successfully, then return true.
- If the array does not contain the (value), then return false.

2) sort

Function

void sort(String type)

Description

- This function prints the integers on the array in the given ordering type.
- If (type) is “descend”, then print the integers in descending order.
- If (type) is “ascend”, then print the integers in ascending order.

3) top

Function

void top(int k, String type)

Description

- This function prints the top-**k** largest or smallest integers in the array.
- (k) is a top-**k** parameter as integer. The (k) is always less than or equal to

the number of integers in the array.

- If (type) is “largest”, then print top-k largest integers in descending order.
- If (type) is “smallest”, then print top-k smallest integers in ascending order.

5. Specification of I/O

1) add

Input form	Output form
`add (value)`	`ADD: (value)`

Description

- (value) is an integer to be added into the array.

Example Input	Example Output
`add 1`	`ADD: 1`

2) remove

Input form	Output form
`remove (value)`	`REMOVE: (value)`

Description

- (value) is an integer to be removed from the array.
- If the array does not contain (value), then print “The value does not exist”. (Note that you should NOT print a dot in the message.)

Example Input	Example Output
`remove 2`	`REMOVE: 2`
	`The value does not exist`

3) sort

Input form	Output form
`sort (type)`	`SORT: (sorted integers)`

Description

- (type) indicates the ordering types of the sort: ‘descend’ or ‘ascend’.
- If (type) is ‘descend’, then (sorted integers) should be shown in descending

order.

- If (type) is ‘ascend’, then (sorted integers) should be shown in ascending order.
- Suppose that 4, 5, 1, 3, and 2 are added into the array in this example.

Example Input	**Example Output**
`sort descend`	`SORT: 5 4 3 2 1`
`sort ascend`	`SORT: 1 2 3 4 5`

4) top

Input form	**Output form**
`top (k) (type)`	`TOP: (top-k integers)`

Description

- (k) is a top-**k** parameter as integer. (type) is ‘largest’ or ‘smallest’.
- If (type) is ‘largest’, then (top-k integers) are the top-k largest integers in *descending order*.
- If (type) is ‘smallest’, then (top-k integers) are the top-k smallest integers in *ascending order.*
- Suppose that 4, 5, 1, 3, and 2 are added into the array in this example.

Example Input	**Example Output**
`top 2 largest`	`TOP: 5 4`
`top 2 smallest`	`TOP: 1 2`

6. Sample Input

```
add 4

add 5

add 1

add 3

add 2

sort descend
sort ascend

top 3 largest

top 3 smallest

remove 1

add 10

top 3 largest

top 3 smallest

remove 9
```

7. Sample Output

```
ADD: 4

ADD: 5

ADD: 1

ADD: 3

ADD: 2

SORT: 5 4 3 2 1

SORT: 1 2 3 4 5

TOP: 5 4 3

TOP: 1 2 3

REMOVE: 1

ADD: 10

TOP: 10 5 4

TOP: 2 3 4

The value does not exist
```