CS300 – Summer 2018-2019 - Sabancı University

Homework #4 – Median of Streaming Integers

Due May 6 Wednesday at 23:55

Brief Description

In this homework, you will write a program that finds and maintains the median of streaming integers. Your program must store all integers as well as updating the median value as new inputs arrive. To achieve both goals, your program must make use of the Heap data structure in the background. The program will utilize two heaps: a **MaxHeap** and a **MinHeap**. Each streamed integer will be stored in the appropriate Heap in **O(logN).** Using the capabilities of the heap data structure, the program tells the median in **O(1)** time complexity.

Therefore, you have to provide a Heap implementation which is used as both **MaxHeap** and **MinHeap** in the program flow.

Note that this an implementation of Question 2 of Recitation 9.

Program Flow

Streaming integers will be read from **.txt** files. Each integer at each file will be considered as the next integer in the stream. Therefore you are going to get a name of a file from the console until **Ctrl + Z** is pressed, which ends the input (HINT: You may use **getline(cin, fileName)**). After getting the name of each file, you have to open the input file and process all the integers in it. The files will contain only signed integers and you do not have to do input checks. Please check sample runs and sample input files. After processing each file, you have to prompt the current median.

Insertions

As mentioned before, your program has to run a **MaxHeap** and a **MinHeap.** Depending on the current value of **median**, you will insert the next integer either to the **MaxHeap** or the **MinHeap.** Particularly, **MaxHeap** is responsible for storing the integers less than the **median**, and **MinHeap** stores the integers greater than the **median.** Therefore, new insertions will be made in this fashion.

Every time you insert, compare the new element with those at the top of the heaps in order to decide where to insert it. If the new element is greater than the current median, it goes to the min-heap. If it is less than the current median, it goes to the max heap. Then you might need to rebalance. If the sizes of the heaps differ by more than one element, extract the min/max from the heap with more elements and insert it into the other heap.

Telling the Median

If the number of elements in both Heaps are equal, the median is the average of the top elements of the **MaxHeap** and **MinHeap.** Otherwise, it is equal to the top element of the one which has more elements. Note that the difference in the number of elements between the heaps is either 1 or 0. Also note that the top element refers to the maximum element for **MaxHeap** and the minimum element for **MinHeap,** which can be decided at O(1).

Rules

- You have to use Heaps, no other data structure or sorting methods are allowed.
- You can't give separate implementations for **MaxHeap** and **MinHeap**. Code duplications will lose points. You have to create a class hierarchy and use polymorphism.
- Heap implementation must be template-based.

Input

The file names will be received from the console until the input is ended by the user.

Output

After processing each file, you need to print out the current median.

Sample Run 1

Please enter the next filename that contains integer stream: ***input1.txt***
current median: 520
Please enter the next filename that contains integer stream: ***input2.txt***
current median: 487
Please enter the next filename that contains integer stream: ***input3.txt***
current median: 481
Please enter the next filename that contains integer stream: ***input4.txt***
current median: 434.5
Please enter the next filename that contains integer stream: ***Ctrl-Z***
Press any key to continue . . .

General Rules and Guidelines about Homeworks

The following rules and guidelines will be applicable to all homeworks, unless otherwise noted.

How to get help?

You may ask questions to TAs (Teaching Assistants) of CS300. Office hours of TAs can be found here. Recitations will partially be dedicated to clarify the issues related to homework, so it is to your benefit to attend recitations.

What and Where to Submit

Please see the detailed instructions below/in the next page. The submission steps will get natural/easy for later homeworks.

Grading

Careful about the semi-automatic grading: Your programs will be graded using a semi-automated system. Therefore, you should follow the guidelines about input and output order; moreover, you should also use the exact same prompts as given in the Sample Runs. Otherwise semi-automated grading process will fail for your homework, and you may get a zero, or in the best scenario you will lose points.

Grading:

- **We will hold a demo session for your homework grading. Each one of you must show that your code is running as expected and explain several parts of your code if necessary. Wait for an announcement for the demo scheduling.**
- Late penalty is 10% off the full grade and only one late day is allowed.
- **Having a correct program is necessary, but not sufficient to get the full grade. Comments, indentation, meaningful and understandable identifier names, informative introduction and prompts, and especially proper use of required functions, unnecessarily long program (which is bad) and unnecessary code duplications will also affect your grade.**
- Please submit your own work only (even if it is not working). It is really easy to find out "similar" programs!
- For detailed rules and course policy on plagiarism, please check out http://myweb.sabanciuniv.edu/gulsend/courses/cs201/plagiarism/

Plagiarism will not be tolerated!

Grade announcements: Grades will be posted in SUCourse, and you will get an Announcement at the same time. You will find the grading policy and test cases in that announcement.

Grade objections: Since we will grade your homeworks with a demo session, there will be very likely no further objection to your grade once determined during the demo.

What and where to submit (IMPORTANT)

Submission guidelines are below. Most parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

Add your name to the program: It is a good practice to write your name and last name somewhere in the beginning program (as a comment line of course).

Name your submission file:

- Use only English alphabet letters, digits or underscore in the file names. Do not use blank, Turkish characters or any other special symbols or characters.
- Name your cpp file that contains your program as follows.
  "SUCourseUserName_yourLastname_yourName_HWnumber.cpp"
- Your SUCourse user name is actually your SUNet username which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:
  cago_ozbugsizkodyazaroglu_caglayan_hw4.cpp
- Do not add any other character or phrase to the file name.

- Make sure that this file is the latest version of your homework program.
- You need to submit ALL .cpp and .h files including the data structure files in addition to your main.cpp in your VS solution.
- The name of the main cpp file should be as follows.

  "SUCourseUserName_yourLastname_yourName_HWnumber.cpp"

  For example zubosman_Osmanoglu_Zubeyir_hw4.cpp is a valid name, but hw4_hoz_HasanOz.cpp, HasanOzHoz.cpp are NOT valid names.

Submission:

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
Gülşen Demiröz