Comp 202 - Spring 2022
Homework #4
Due date - 23:59 31/05/2022

In this homework assignment, you will implement the two of the popular sorting algorithms in computer science, namely **insertion sort, heapsort** and custom schemes **TopK** and **Fast-TopK** which both require the sorting algorithm that you have implemented.

For this assignment, you will implement and analyze the performance of the both sorting algorithms then prepare a pdf report for your findings. In your report, we expect you to sort random arrays with lengths of [10, 100, 1,000, 10,000, 100,000, 1,000,000] and show your results using graphical tools such as charts and plots etc and explain which sorting method is better suited for a specific length. **Asymptotically analyze** both methods then prove your analysis with empirical results. Sort algorithms must work in both increasing and decreasing ways with the same computational complexity. To sort different sized arrays, you need to set the **test** variable to **false.**

Answer the following questions in your own words:

1-) Does heapsort work inplace ? Explain your answer.

2-) What is the running time of heapsort for presorted input ?

3-) Give an example input that requires heap-sort to take O(n log n) time to sort, but insertion-sort runs in O(n) time. What if you reverse this list ?

TopK:

TopK method takes an unsorted **array**, integer **K**, boolean **largest** and string **method**. Algorithm returns the sorted largest or smallest K **values** and their respective **indices** using the given sorting method. To ease your implementation, Your sorting algorithms should also sort the indices by default.

Example for TopK,

arrr = [1, 5 , 6 , 2 ,4 ,3 ,9]

TopK(arr, K=3, largest=true, ”insertionsort”) returns = > [9, 6, 5] , [6, 2, 1]

TopK(arr, K=3,largest=false, ”heapsort”) returns = > [1, 2, 3] , [0, 3 ,5]

For more information about TopK, you can check out the documentation on torch.topk for Pytorch.

Fast-Topk:

Now we want to speed up the TopK process with an approximate TopK, to this end we want to minimize unintended sorting where the given **K** value is equivalent to the very small portion of the array (1-2 % at most).

Your objective is to implement a method that returns the largest or smallest **K** elements and their respective indices for the given very large array (500k to 1M). To ease the implementation, the random generator we have provided to you generates signed integers in normal (gaussian) distribution manner.
You must select an optimal threshold to minimize sorting, however statistically, your threshold may not work every time but this is an extremely small probability if the given array is large enough.

In your report, you must **asymptotically analyze your implementation.** Show why your method has better performance than TopK, both empirically and theoretically. You should also include an adequate amount of comments in your code. **You may use java heap for heap sort but you are not allowed to use additional sorting libraries**. All of the implementations must be your **OWN WORK.**

Submission Details:

Usage of github classroom and submission details:
1) Accept the invitation using this link
2) A personal repository with the starter code will be created. You can directly edit the code on github’s page, clone it to your local storage and edit it there. **Ensure that your repositories are private.**
3) Make sure your changes are committed and pushed to the repository.

4) Check if you have successfully passed the tests in the “Actions” tab of your repository. Autograder checks all implementations in one run therefore partial implementations will not pass autograder.
5) To make sure your code is received and avoid any potential problems on github’s side, submit a copy of java files on Blackboard as well. Submit all files including your report as a single .zip file, named as: “ID_NAME_SURNAME.zip”