BLG 335E - Analysis of Algorithms Assignment #1 Quicksort

Alperen Kantarcı
kantarcia@itu.edu.tr

Due: November, 25th 23:55

Important Notes

- Please write your own codes, copying code parts from **books**, **websites** or any other sources **including your friends** is considered as **plagiarism** and will be treated seriously. You are expected to act according to the Code of Honor.
- Do not upload your codes to any public platform (e.g. Github) until the deadline of homework passes.
- Use C++ language and do not forget to compile your codes on Linux using g++ command before sending them.
- You may use STL but do not use built-in sorting functions.
- Submit your source codes and report files on Ninova before the deadline. Late submissions and submissions via e-mail may **not be accepted**.
- For any issues regarding the assignment, please contact Alperen Kantarcı (**kantarcia@itu.edu.tr**). Do not hesitate to send an e-mail if you have any confusion.

I. Implementation (40 points)

You are given a dataset where the music listening histories of the Twitter users between 2012-2013 are reported. If you are curious about the original dataset you can take a look from here. There are the following attributes in the dataset:

tweet_id: Enumeration of the Tweet
tweet_unixtime: Tweet time in Unix format
track_title: Name of the song

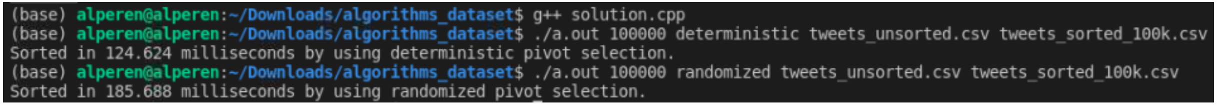
tweet_datetime: Tweet time in a structured format
artist_name: Artist name of the song
country: Tweet location

You are supposed to sort the tweets in case-insensitive alphabetical order in terms of the country. For the tweets with the same country name, you should sort them in in case-insensitive alphabetical order in terms of the artist name. If both country name and the artist name are the same then you should sort them in ascending order of *tweet_unixtime*. As a sorting algorithm, you must use In-Place (without extra array for left and right partitions) Quicksort with both **deterministic** and **randomized** pivot selection.

Your code must run with the following command: ./a.out N A I O
N: number of the tweets to be sorted (Take the first N entries from the file. N will be always smaller than the given number of tweets)
A: name of the pivot selection algorithm (either randomized or deterministic)
I: input file path (e.g., unsorted.csv)
O: output file path for saving the result (e.g., output.csv)

After the execution, a message including the elapsed execution time should be printed out. You should write the tweets in sorted order into the given output file with the same format. You can see an example usage of the program in Figure I.. Please also note that you should have comments in your source code.

You **will also be graded** according to the comment quality. If your implementation is correct without any explanatory comments on algorithm's steps, you will not get full points.



```
(base) alperen@alperen:~/Downloads/algorithms_dataset$ g++ solution.cpp
(base) alperen@alperen:~/Downloads/algorithms_dataset$ ./a.out 100000 deterministic tweets_unsorted.csv tweets_sorted_100k.csv
Sorted in 124.624 milliseconds by using deterministic pivot selection.
(base) alperen@alperen:~/Downloads/algorithms_dataset$ ./a.out 100000 randomized tweets_unsorted.csv tweets_sorted_100k.csv
Sorted in 185.688 milliseconds by using randomized pivot selection.
```

> **Notice** For calculating running time you can use clock() function under the ctime library. You should only calculate the running time of the sorting algorithm. Therefore, I/O operations shouldn't be included. You can read more about it from here.

II. Report (60 points)

1. **(5 points)** Write down the asymptotic upper bound for the Quicksort with deterministic pivot selection for best case and worst case by solving the recurrence equations.

2. **(10 points)** Write down the asymptotic upper bound for the Quicksort with randomized pivot selection by doing probabilistic analysis.

3. **(20 points)** Run both algorithms (Quicksort with randomized and deterministic pivot selection) for different N values (1000, 10K, 100K, 500K, 1M) by using **tweets_unsorted.csv** data and calculate the average time of multiple executions (at least 5 times). Save each sorting result into a csv file (e.g. tweets_sorted_100k.csv, tweets_sorted_1M.csv). Report the average execution times in a table and prepare an Excel plot which shows the N – runtime relation of both algorithms in one plot. Comment on the results considering the asymptotic bound that you have found in Question 1, and 2.

4. **(10 points)** Run both algorithms (Quicksort with randomized or deterministic pivot selection) for the sorted data (1000, 10K 100K, 500K, 1M) that you saved in Question 3. Prepare a table and plot as you did in Question 3. Comment on the running time differences between sorted and unsorted data by considering asymptotic upper bounds that you have found in Question 1, and 2. Indicate in which cases (sorted vs unsorted) you would choose which algorithm. Why?

5. **(15 points)** Consider the given dual pivot Quicksort algorithm. You can consider either random or regular pivot selection method for the dual pivot Quicksort algorithm. Write down the asymptotic upper bound for the algorithm. Is the running time different than the Quicksort? Comment on the algorithm's running time and possible use cases of the algorithm. You don't have to implement the algorithm.

Algorithm 1 Dual Pivot Quicksort(*array*, *left*, *right*)

```
if left < right then
    Choose two pivot indexes $L$ and $R$
    if array[$L$] > array[$R$] then
        swap pivots
    end if
    partition the array elements into
        (i) those less than the value of the $L$
        (ii) those between the value of the pivots
        (iii) those greater than value of the $R$

    update $L$ and $R$ indexes according to the new indexes of pivots

    DualPivotQuickSort(array, left, $L$ - 1)
    DualPivotQuickSort(array, $L$ + 1, $R$ - 1)
    DualPivotQuickSort(array, $R$ + 1, right)
end if
```