Algorithms Homework Assignment #3

"3-way Quick Sort"

Due date: Wednesday, April 18th (Wed) 2018
- Submit softcopy on our class server

In this homework assignment, you will first implement a basic randomized quick sort algorithm which sorts an array of integers given in an input file. By randomized, it means that you choose a random element as the pivot. You will then improve this quick sort by implementing 3-way quick sort. The goal of this homework assignment is to redesign the basic randomized quick sort algorithm so that it works fast even on sequences containing many equal elements. You will also compare the running times of the two algorithms.

Here are the steps and requirements on what you'll need to do.

Download the sample input text files "hw-3wq-input-*.txt" where * = {1, 2, …, 7}. These files contain 'K' integers, one integer in each row, and you'll use these numbers for your input array.

- "hw-3wq-input-1.txt" file contains all the integers between 1 and 1,000,000 (inclusive, with no repeats) in random order, similar to your first homework.
- "hw-3wq-input-2.txt" file contains all integers from 1 to 1,000,000 in sorted order.
- "hw-3wq-input-3.txt" file contains 1M integers between 1 and 100,000 in random order with repeats.
- "hw-3wq-input-4.txt" file contains 1M integers between 1 and 10,000 in random order with repeats.
- "hw-3wq-input-5.txt" file contains 1M integers between 1 and 1,000 in random order with repeats.
- "hw-3wq-input-6.txt" file contains 1M integers between 1 and 100 in random order with repeats.
- "hw-3wq-input-7.txt" file contains 1M integers between 1 and 10 in random order with repeats.

You should use these files as inputs to test your program. However, your program should work with other input files with different number and different order of integers also. You may assume that all integers are below 10^9.

Your main task is to

1. Implement 'quick sort' algorithm and '3-way quick sort' algorithm.
2. Sort the first 'N' integers in an input file, in non-decreasing order, using quick sort and 3-way quick sort.
3. Measure their running time for all 7 sample input files with various N values (up to N = 1,000,000), and
4. Plot a graph that compares the actual running time of the two algorithms.

Choosing the pivot:
As you know, the performance of quick sort algorithm depends heavily on which elements are chosen as pivots. For this homework, you will always choose a random element from the given array and use that as the pivot.

Hint: To improve the basic quick sort algorithm to efficiently process sequences with few unique elements and many duplicates, your 3-way quick sort algorithm should replace the 2-way partition function in basic quick sort with a 3-way partition function. That is, your new partition procedure in 3-way quick sort should partition the array into three parts: $< x$ part, $= x$ part, and $> x$ part.

You must implement two programs using **C**, and you source code MUST have the filename "basic_quick_sort.c" and "3way_quick_sort.c". The file names must be exactly as given, otherwise it won't be graded and you will get zero points.

Your programs must take three command line arguments: <input filename>, <N>

- Ex> `./basic_quick_sort <input_file> <N>`
- Ex> `./3way_quick_sort <input_file> <N>`

Your programs should do the following:

- For a given N value, read the first N integers from the input file, put them into an array of integers, and sort them using quick sort algorithm.
  - ✓ if N > K, then your program should sort all K, and exactly K, numbers in the file correctly.
- Your program **MUST output on the sorted result** on the screen.
- Your program MUST ALSO output the **running time** of the program in **milliseconds**. Make sure that your sorted result is correct!!

Plot **two graphs** that compares the running time of the two sorting algorithms.

- First graph, with input file name on the x-axis, running time on the y-axis (in milliseconds), and
- Second graph, with N on the x-axis, running time on the y-axis (in milliseconds).
  - ✓ For this graph, use "hw-3wq-input-7.txt" as the input file, and
  - ✓ N values must include "N = 10000, 100000, 200000, 500000, <student ID>%1000000, 1000000", and you may use more N values for smoother and better looking plot.
  - ✓ Note that x-axis values are not equally spaced. Make sure you have correct scale on the x-axis!
- Both graph must plot both "Quick Sort", "3-way Quick Sort"

You program must work with other input files as well, not just 'hw-3wq-input-*.txt'. We will test and grade your program with other input files with different number of integers. Also, your program must gracefully handle all exception/error cases such as N > K, N < K, no input file, incorrect command line arguments, Windows/Linux line ending, etc.

What and how to submit

- You must submit both a hardcopy report and softcopies of all the files;
- Hardcopy report should include the graphs that compare the running time of the three sorting algorithms, and your conclusion. Hardcopy report should be submitted during class.
- Here is the instruction on how to submit the softcopy files :
  1. Login to your server account at nsl2.cau.ac.kr.
  2. In your home directory, create a directory "submit_alg/submit_<student ID>_hw3" (ex> "/student/20149999/submit_alg/submit_20149999_hw3")
  3. Put your "basic_quick_sort.c" and "3way_quick_sort.c" file in that directory, together with the given Makefile.
- Your submission will not be graded if you do not follow the instructions exactly.

Other requirements:

- Your program must run on our class Linux server at nsl2.cau.ac.kr.
- Your code must include your name and student ID at the beginning of the code as a comment.
- Your code should be easily readable and include sufficient comments for easy understanding.
- Your code must be properly indented. Bad indentation/formatting will result in score deduction.
- Your code should not include any Korean characters.
- Your program should work regardless of whether the input file format is Windows/Linux/MacOS based.
  - ✓ That is, input file may have Windows or Linux or MacOS line ending format (\n, \r\n, \r).

Grading criteria:

- You get 3 points
  - ✓ if all your programs work as requested, AND if you meet all the requirements, AND
  - ✓ if your hardcopy report is well written.
- Otherwise, partial deduction may apply.
- No delayed submissions are accepted.
- Copying other student's work will result in negative points.
- Code that does not compile or code that does not run will result in negative points.