1 Assignment 2

Submit source code and running instructions to EAS[1]. Do it in Java. Do not use java's search methods! Do not use Java's Collections or Subclasses, code your own. Place all textual responses in a PDF report submitted with the code.

Please put all files in the default pacakge. This isn't good design, but it *is* easier to mark.

This assignment should use Command-Line. Not Scanner, not anything else. Command-Line input. Learn what String[] args are for in the main method! Similar, but a bit smarter, to what you may have used in C++.

Posted: Monday, July 10^{th}

Due: Sunday, July 23^{rd}

Grade: 5%

1. Quicksort
   (a) Implement a generic Quicksort algorithm that takes an array as input
      i. This file should be called QSNormal.java
      ii. This class should have a sort method:
      public static void sort(int[] input)
   (b) Implement a Quicksort algorithm that uses diversion to Insertion Sort.
      i. This file should be called QSInsertion.java
      ii. The class should have a sort method:
      public static void sort(int[] input)
   (c) Write classes that are able to generate test inputs of size 10, 100, 10000, 1000000.
      i. One file should be RandomGen.java
      ii. One file should be FixedGen.java
      iii. RandomGen should generate random integers of a uniform distribution.
      iv. FixedGen should always generate a fixed ascending input.
   (d) Make a driver that sorts values from your input
      i. This file should be called QSDriver.java
      ii. This file should output the run-time in either ns or μs
      iii. it should accept command-line as follows:
      java QSDriver <sort> <gen> <length> <seed>
         A. <sort> is either QSNormal or QSInsertion
         B. <gen> is either RandomGen or FixedGen

[1] https://fis.encs.concordia.ca/eas/

C. <length> is the number of ints to be sorted in the input array

D. <seed> is an optional argument (it might not be passed) that lets you repeat the random seed for RandomGen (but is ignored by FixedGen)

(e) Record performance times of runs for each input size specified in 1c for Quicksorts implemented in 1a and 1b using RandomGen.

2. In clear, natural language, describe the performance differences between the two sorts. Try to correlate this with the underlying mechanism. Identify how you picked your diversion threshold.

   (a) This textual response should be no more than 8 lines / 80 words.

3. In clear, natural language, describe a pathological input (one that yields a worst-case) for your Quicksorts defined in 1a. If FixedGen is not a pathological case, describe how pivot selection in your Quicksorts would lead to it becoming a pathological case.

   (a) This textual response should be no more than 8 lines / 80 words.