CS1102S Data Structures and Algorithms

Assignment 04:
Sorting

Version of Wednesday 17th March, 2010, 21:47

For this assignment, use the following submission procedure:

- Solve your assignment.
- Copy the file

  ```
  SortingData.java
  IterativeMergeSort.java
  SortingFour.java
  TwoSum.java
  ```

  to your account on `sunfire` to a folder (let's say `cs1102s/assignment6`).
- Go to that folder and execute the `submit` program:

  ```
  % cd cs1102s/assignment6
  % ls *.java
  SortingData.java
  IterativeMergeSort.java
  SortingFour.java
  TwoSum.java
  % /home/course/cs1102s/bin/submit
  ```

  You will get a message that tells you whether the file has been submitted successfully. The script `submit` can be run several times, and you can remove submitted files later. For details on how to use the submit script, type

  ```
  /home/course/cs1102s/bin/submit -h
  ```

Please note all file names are case sensitive and have to conform to the assignment questions.
Submission is activated 5 days before the submission deadline, which is on 19/3, 6:00pm and will be de-activated at that time.

1. (8 marks) Download the assignment project from
   `http://www.comp.nus.edu.sg/~cs1102s/java/assignment_06.zip`.

   Extact the zip file and create a Java project in Eclipse, as usual.

   Look at the class SortingData in the package sorting. The class contains the declaration of four arrays, whose elements are not given.

   Consider the following two sorting algorithms:

   - Mergesort. Note that in the best case, fewer comparisons are done during the execution of the function merge (page 261) than in the worst case, because many elements can be copied to tmpArray using the second and third while loops.
   - Quicksort using the following pivot selection:

     ```
     int half = (left + right) / 2;
     ```

     Use the quicksort implementation given in sorting.QuickSort to make sure that your partition algorithm is exactly the same as the one used by the other students in the class.

   Assign the array variables using literal arrays containing all integers from 0 to 31, following the example, such that

   - the array SortingData.mergeSortBest uses the smallest possible number of comparisons using mergesort,
   - the array SortingData.mergeSortWorst uses the largest possible number of comparisons using mergesort,
   - the array SortingData.quickSortBest uses the smallest possible number of comparisons using quicksort with the pivot selection given above,
   - the array SortingData.quickSortWorst uses the largest possible number of comparisons using quicksort with the pivot selection given above.

   By "comparison", we mean object comparisons using compareTo(...); index comparisons are not counted.

2. (6 marks) Implement in the class sorting.IterativeMergeSort a version of mergesort that does not use recursion. Your class should not contain any (static or non-static) functions except for the function mergeSort itself. The program should not create any objects (using new) while the sorting is done; you may create objects before the first comparison.

   Hint: Use arrays to simulate the runtime stack of the JVM.

3. (Question 7.41, page 288; 5 marks)

   - (no submission) Prove that any comparison-based algorithm to sort four elements requires at least five comparisons.

- Implement in the class sorting.SortingFour an algorithm for sorting four elements, which requires at most five comparisons.

4. (Question 7.48 b., page 289; 5 marks) We are given an array that contains N numbers. We want to determine if there are two numbers whose sum equals a given number K. For instance, if the input is 8, 4, 1, and 6, and K is 10, then the answer is yes (4 plus 6 is 10). A number n may appear more than once in the input array; in that case and only in that case the sum may have the form $n + n$.

   Give in the class sorting.TwoSum an algorithm to solve this problem in $O(N \log N)$ time.

   Hint: Sort the items first!