CSC 211—Data Structures and Algorithms

Compare the timings to find the median and its co-median by (1) sorting with `heapsort`, (2) sorting with `quicksort`, (3) and using `partition` to determine the median directly, and (4) by using a heap built bottom-up to determine the median without sorting.

Practical 5 Term 2 — 16 May 2016

Submit by 22 May 2016

Use the first $N = 2^{20} + 1$ elements of the file `data.median` in the `notes` file. Do NOT copy this file to your own directory. You will be penalized by 20% if you do not comply with this instruction. For a list of odd length the *median* is the element lying in the middle of the sorted list. For a list of of odd length the *co-median* is the average of the three elements lying nearest the middle of the sorted list. When the list has an even number of elements the median and the co-median coincide. They are formed by the average of the two elements nearest the middle of the list.

1. Create a directory that contains only todays's work called `52practical`.
2. Use the same dataset of $N = 2^{20} + 1$ shuffled integer elements in all four experiments.
3. Find the median and co-median by first sorting the data using `heapsort`.
4. Find the median and co-median by first sorting the data using `quicksort`.
5. Find the median and co-median without sorting by using `partition`.
6. Find the median and co-median without sorting by using a `heap` built from the bottom up.
7. Each of the methods must be timed by running it using the first $N = 2^{20} + 1$ shuffled integer elements stored in the notes file in `data.heap`.
8. Your code must print `N`, the median, the co-median and the time taken for each of the of the methods.
9. It should be needless to say that you ought to test your code with small a set of data of about $7 - -11$ elements.
