Department of Electrical & Computer Engineering (ECE)
Concordia University

Data Structure

COEN 352: Summer '24

Assignment 2: (due date: June 6th @ 23hr55, via Moodle)

Questions

1) Shell sort question:
   - Generate a set of lists, each contains N = 500, 1000, 2000, 4000 and 8000 random order integers.
   - Use two gaps to implement Shell sort: Gap 1: h = 4x + 2.
     Gap 2: find your favorite one from literature.
   - Time two programs with different gaps and fill out the table 1.

2) Merge sort question:
   - Generate 3 sets of lists (best, worst, average), each set contains 500, 1000, 2000, 4000 and 8000 integers – with distribution of elements representing **best**, **worst**, and **average** case for merge sort.
   - Implement Merge sort.
   - Time each case and fill out the table 1.

Table 1

Algorithm Name	**Chosen gap (h)**	**Run time for every N value** **0.5K 1K 2K 4K 8K** **Nano Seconds**				
Shell sort_gap1	4x+2	**1342654**	**1664766**	**1239789**	**2837257**	**3617725**
Shell sort_gap2	2^k+1	**418985**	**631896**	**1604246**	**1171971**	**2392317**
Merge sort (best)	---	**398136**	**169581**	**356135**	**762073**	**55832669**
Merge sort (worst)	---	**54428**	**159238**	**350522**	**734726**	**1609038**
Merge sort (average)	---	**102328**	**250575**	**478707**	**1022284**	**1464263**

Recommended practice problems in textbook (do not need to submit)

2.2.4 Does the abstract in-place merge produce proper output if and only if the two input subarrays are in sorted order? Prove your answer, or provide a counterexample.

2.3.5 Give a code fragment that sorts an array that is known to consist of items having just two distinct keys.

2.2.14 Merging sorted queues. Develop a static method that takes two queues of sorted items as arguments and returns a queue that results from merging the queues into sorted order.

2.2.15 Bottom-up queue mergesort. Develop a bottom-up mergesort implementation based on the following approach: Given N items, create N queues, each containing one of the items. Create a queue of the N queues. Then repeatedly apply the merging operation of Exercise 2.2.14 to the first two queues and reinsert the merged queue at the end. Repeat until the queue of queues contains only one queue.

2.2.18 Shuffling a linked list. Develop and implement a divide-and-conquer algorithm that randomly shuffles a linked list in linearithmic time and logarithmic extra space.

2.3.2 Show, in the style of the quicksort trace given in this section, how quicksort sorts the array E A S Y Q U E S T I O N (for the purposes of this exercise, ignore the initial shuffle).

2.3.11 Suppose that we scan over items with keys equal to the partitioning item's key instead of stopping the scans when we encounter them. Show that the running time of this version of quicksort is quadratic for all arrays with just a constant number of distinct keys.

Submission

You must submit a ZIP file that includes all your programs and related files. Name you ZIP file exactly “Assignment2”. ONLY SUBMIT ONE ZIP FILE PER TEAM.

The programs must be in Java (.java file), any other type of file will not be marked. The programs need to be titled exactly as “Ass2_Q1” and “Ass2_Q2”. I will open your Assignment2 folder as a project, so name your main class differently for two questions (avoid duplicate main.java). I will test your code using a random list containing 10000 integers. Also, place the **names and IDs** of all team members (maximum 2 people per team) on the first line of each file (commented). A team only needs one submission and team members will share the same grades. If you want to be marked separately, put only your name and ID in your program and submit separately. Comment your code.

For the table 1, fill that out in Assignment2.pdf and include the pdf in your ZIP file.

Include all your .java files in a folder called “Ass2Java”.