TCP2101 Algorithm Design & Analysis

Semester 2, 2019/2020

Assignment (20%)

Instruction

- This is a group assignment with **exactly 4 persons per group**. Any request for more members or less members will not be entertained unless it is a special case.
- Each group will be randomly assigned with **ONE (1)** of the questions below. Implement the assignment using **C++ language**. The code should be able to handle the sample set of data suggested in the topic description.
- Write a report to describe the algorithms implemented, results of the tests, discussions/reasoning for the results obtain and a conclusion on the algorithms (e.g. suitability, possible improvements etc.). Include all the citations and references in the report.
- Group leader should zip the submission (report + source codes) and email to the respective tutor. If the file size is too large, upload the zip file into Google drive and email the downloadable-link to the respective tutor. The tutor will reply to confirm the receipt of the assignment within a week of time. If there is no reply from the tutor, please contact the tutor immediately. The submission deadline is on **5th February 2020 (Wednesday, before 5pm)**. Start the assignment as soon as possible. No extension of submission deadline will be made.
- There will be an **interview** for the assignment. All members must present to explain the algorithms, demo the program, present the experimental results, conclude the findings, and Q&A. No mark will be given to the sleeping member. The interview date, time and venue will be announced by the respective tutor.
- Include the checklist table (**ONLY the checklist related to the Question assigned to the group**) below in the report and fill in the rightmost column.

Question 1

No.	Tasks	Mark	Checklist (Yes/no) and person(s) to work on this task
1	Correct implementation of dataset generation of input files with different integers in random (500k, 1m, 5m, 10m, etc.). Write a function to generate input data with unique integers randomly in the file. The filenames are: - bst_avl_hash_00500000.txt, bst_avl_hash_01000000.txt, bst_avl_hash_05000000.txt, bst_avl_hash_10000000.txt, etc.	2	
2	BST tree (Binary Search Tree) implementation for search algorithm - Correct implementation of the algorithm. - Input of 10 unique integers from a file. The filename is bst_avl_hash_10.txt. - Output the result for tutor to inspect the correctness of the algorithm. - Provide step-by-step illustrations of the algorithm with the input data above and draw them in report	4	
3	AVL tree (balanced BST) implementation for search algorithm - Correct implementation of the algorithm. - Input of 10 unique integers from a file. Filename is bst_avl_hash_10.txt - Output the result for tutor to inspect the correctness of the algorithm - Provide step-by-step illustrations of the algorithm with the input data above and draw them in report	4	
4	Hash table with separate chaining by hashing implementation for search algorithm - Correct implementation of the algorithm - Input of 10 unique integers from a file. Filename is bst_avl_hash_10.txt - Output the result for tutor to inspect the correctness of the algorithm - Provide step-by-step illustrations of the algorithm with the input data above and draw them in report - Prove that a good hash function (use prime number and not closed to the power of two) can reduce collision problem.	5	
5	Your report of input files of different unique integers (500k, 1m, 5m, 10m, etc.) must include: - perform numerous random searches (i.e. 10 random integers to	5	

	be searched) using the three algorithms, and get the average times for one integer for the algorithms - the above experiment results that can be used to perform a comparative analysis between the three implementations - conclude your findings in the report		

Question 2

No.	Tasks	Mark	Checklist (Yes/no) and person(s) to work on this task
1	Kruskal's minimum spanning tree (MST) algorithm - Correct implementation of the algorithm on adjacency matrix - Input of MST on a graph of 10 nodes from a file. The filename is am_graph10.txt - Output the result of MST for tutor to inspect the correctness of the algorithm - Provide step-by-step illustrations of the algorithm with the same graph as above and draw them in report	4	
2	Kruskal's minimum spanning tree (MST) algorithm - Correct implementation of the algorithm on adjacency list. - Input of MST on a graph of 10 nodes from a file. The filename is al_graph10.txt - Output the result of MST for tutor to inspect the correctness of the algorithm - Provide step-by-step illustrations of the algorithm with the same graph as above and draw them in report	4	
3	Correct implementation of input files of random graphs of different number of vertices (10000, 50000, 100000, 500000, etc.) and of different patterns (dense and sparse). The filenames are: - am_dense10000.txt, am_dense50000.txt, am_dense100000.txt, am_dense500000.txt, etc - am_sparse10000.txt, am_sparse50000.txt, am_sparse100000.txt, am_sparse500000.txt, etc	2	
4	Kruskal's minimum spanning tree (MST) algorithm Your report must include: - the above experiment results that can be used to perform a comparative analysis between the two implementations - conclude your findings in the report	2	
5	Prim's minimum spanning tree (MST) algorithm - Correct implementation of the algorithm on adjacency matrix	4	

	- Input of MST on a graph of 10 nodes from a file. The filename is am_graph10.txt - Output the result of MST for tutor to inspect the correctness of algorithm - Provide step-by-step illustrations of the algorithm with the same graph as above and draw them in report		
6	Prim's minimum spanning tree (MST) algorithm - Correct implementation of the algorithm on adjacency list - Input of MST on a graph of 10 nodes from a file. The filename is al_graph10.txt. - Output the result of MST for tutor to inspect the correctness of algorithm. - Provide step-by-step illustrations of the algorithm with the same graph as above and draw them in report.	2	
7	Prim's minimum spanning tree (MST) algorithm Your report of input files of random graphs of different number of vertices (10000, 50000, 100000, 500000, etc.) and of different patterns (dense and sparse) must include: - the above experiment results that can be used to perform a comparative analysis between the two implementations - conclude your findings in the report	2	

Question 3

No.	Tasks	Mark	Checklist (Yes/no) and person(s) to work on this task
1	Quick-select algorithm for searching the *k*-th element. - Correct implementation of searching *k*-th element using quick-select algorithm. - Output the intermediate results for a random array of 10 elements for tutor to inspect the correctness of the algorithms.	4	
2	Test the Quick-select algorithm with - Different array sizes (10,000, 100,000, 1,000,000, etc.) that can show significant results. - Different pivot (random pivot vs fixed pivot). - Different cases (e.g. best, average, and worst).	4	
3	Merge-sort algorithm for searching the *k*-th element. - Correct implementation of searching *k*-th element using merge-sort algorithm. - Output the intermediate results for a random array of 10 elements for tutor to inspect the correctness of algorithms.	4	

4	Test the Merge-sort algorithm with - Different array sizes (10,000, 100,000, 1,000,000, etc.) that can show significant results. - Test the Merge-sort algorithm with different cases (e.g. best, average, and worst).	4	
5	- Include the above experiment results that can be used to perform a comparative analysis (such as drawing the graphs for comparison) between the two algorithms (Quick-select & Merge-sort) in the report. - Conclude your findings in the report.	4	

Question 4

No.	Tasks	Mark	Checklist (Yes/no) and person(s) to work on this task
1	Quick-sort algorithm: - Correct implementation of Quick-sort algorithm. - Output the intermediate results for a random array of 10 elements for tutor to inspect the correctness of algorithms.	4	
2	Test the Quick-sort algorithm with: - Different array sizes (10,000, 100,000, 1,000,000, etc.) that can show significant results. - Different pivot (random pivot vs fixed pivot). - Different cases (best, average, and worst).	4	
3	Radix-sort algorithm: - Correct implementation of Radix-sort algorithm. - Output the intermediate results for a random array of 10 elements for tutor to inspect the correctness of algorithms.	4	
4	Test the Radix-sort algorithm with: - Different array sizes (10,000, 100,000, 1,000,000, etc.) that can show significant results. - Different cases (best, average, and worst).	4	
5	- Include the above experiment results that can be used to perform a comparative analysis (such as drawing the graphs for comparison) between the two algorithms (Quick-sort & Radix-sort) in the report. - Conclude your findings in the report.	4	