TCP2101 Algorithm Design & Analysis
- Trimester 2 Session 2017/2018 -
ASSIGNMENT

Assignment Mark: **20%**
Type: **2 members per group (every member must handle at least one algorithm)**
Submission Due: **11 Feb 2018 (Sun)**
Submission Format: One group submits **one zip** file named *ID1_ID2*.zip (*ID1* and *ID2* are the IDs of the members). Upload the zip to your lab section at **MMLS**. The zip file should contain working source code and report.
Programming Language: C++
Presentation: **20 minutes. All members** must present to explain the algorithms, demo the program, present experimental results, conclude your findings, and Q&A.

Registration of Assignment Question

1. Each Assignment Question is limited to a fixed number of groups as shown below (depending on the class size of the tutorial section):

Class size	Max groups per question (Quota)
< 13	1
13-24	2
>24	3

2. Registration your assignment question with your lab lecturer during Week 6 lab, primarily on first-come-first-serve basis.

Question 1

Search for an element in a dataset can be performed using sequential search, binary search or search by hashing. Perform a comparative analysis between these search algorithms. Apply the algorithms to search for a URL in a dataset with at least 100,000 unique URLs (example: www.google.com.my). The experiments should cover the following aspects:

1. Generate a dataset with at least 100,000 unique URLs. You may write a function to generate URLs randomly (2m)
2. Implement sequential search by using an array (2m)
3. Implement AVL (balanced BST) for binary search (2m)
4. Implement hash table with chaining for search by hashing (2m)
5. Prove that a good hash function (use prime number and not closed to the power of 2) can reduce collisions (2m)
6. Correct implementation. Output result for the small size example. This is for tutor to inspect the correctness of algorithms [2m]
7. Perform numerous random searches (multiple URLs) using the three algorithms, and get the average times for the algorithms [2m]
8. In the report, include the experiment results that can be used to perform a comparative analysis between the two algorithms. Conclude your findings in the report [2m]
9. Presentation. [4m]

Question 2

Perform a comparative analysis to evaluate the effectiveness of polynomial hashing (over non-polynomial hashing) on a dataset that contains many email addresses. Prove that the polynomial hashing can be used to reduce collision significantly in the dataset. You may refer to online materials on how to convert string to numerical values, as well as how to do polynomial hashing. The experiments should cover the following aspects:

1. Generate a dataset with at least 100,000 variable length email addresses. You many write a function to generate email addresses randomly (2m)
2. Implement the conversion of string to numerical value (2m)
3. Implement non-polynomial hashing. Must use prime number and not close to the power of 2 (3m)
4. Implement polynomial hashing (3m)
5. Correct implementation. Output result for the small size example. This is for lecturer/tutor to inspect the correctness of algorithms [2m]
6. Perform insertions using the two hash functions, and get the collision rate for both hash functions [2m]

7. In the report, include the experiment results that can be used to perform a comparative analysis between the two hashing functions. Conclude your findings in the report [2m]
8. Presentation. [4m]

Question 3
Perform a comparative analysis for searching the *k*-th element between two algorithms: quick-select and merge-sort. The comparative analysis should cover the following aspects:

1. Correct implementation of searching *k*-th element using quick-select. Output the intermediate results for a random array of 20 elements for lecturer to inspect the correctness of algorithms. [3m]
2. Correct implementation of searching *k*-th element using merge-sort. Output the intermediate results for a random array of 20 elements for lecturer to inspect the correctness of algorithms. [3m]
3. Different array sizes (10,000, 100,000, 1,000,000, etc.) that can show significant results. [2m]
4. Different pivot for quick-select (random pivot vs fixed pivot). [2m]
5. Different cases for both algortihms (best, average, and worst). [5m]
6. Your report must include the above experiment results that can be used to perform a comparative analysis between the two algorithms. Conclude your findings in the report [2m]
7. Presentation. [4m]

Question 4
Perform a comparative analysis between two sorting algorithms: quick-sort and radix-sort. The experiments should cover the following aspects:

1. Correct implementation of quick-sort. Output the intermediate results for a random array of 20 elements for lecturer to inspect the correctness of algorithms. [3m]
2. Correct implementation of radix-sort. Output the intermediate results for a random array of 20 elements for lecturer to inspect the correctness of algorithms. [3m]
3. Different array sizes (10,000, 100,000, 1,000,000, etc.) that can show significant results. [2m]
4. Different pivot for quick-sort (random pivot vs fixed pivot). [2m]
5. Different cases for both algortihms (best, average, and worst). [4m]
6. Your report must include the above experiment results that can be used to perform a comparative analysis between the two algorithms. Conclude your findings in the report [2m]
7. Presentation. [4m]

Question 5
Perform a comparative analysis of Kruskal's minimum spanning tree (MST) algorithm between two implementations of Graph ADT: adjacency matrix and adjacency list. The experiments should cover the following aspects:

1. Correct implementation of the algorithm on adjacency matrix. Output the result of MST on a graph of 8 nodes for lecturer to inspect the correctness of algorithm. Provide 2 demo graphs and drawed them in report. [4m]
2. Correct implementation of the algorithm on adjacency list. Output the result of MST on the same graphs as above. [4m]
3. Random graphs of different number of vertices (10,000, 50,000, 100,000, etc.) [4m]
4. Random graphs of different patterns (dense and sparse). [4m]
5. 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 [2m]
6. Presentation. [4m]

Question 6
Perform a comparative analysis of Prim's minimum spanning tree (MST) algorithm between two implementations of Graph ADT: adjacency matrix and adjacency list. The experiments should cover the following aspects:

1. Correct implementation of the algorithm on adjacency matrix. Output the result of MST on a graph of 8 nodes for lecturer to inspect the correctness of algorithm. Provide 2 demo graphs and drawed them in report. [4m]
2. Correct implementation of the algorithm on adjacency list. Output the result of MST on the same graphs as above. [4m]
3. Random graphs of different number of vertices (10,000, 50,000, 100,000, etc.) [2m]
4. Random graphs of different patterns (dense and sparse). [4m]
5. 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 [2m]
6. Presentation. [4m]