Department Computer Science and Software Engineering
Concordia University

COMP 352: Data Structures and Algorithms
Winter 2024 – Theory Assignment 3

Due date and time: Saturday Nov. 30th, 2024 by 11:59 PM
The due date is strict. NO EXTENSION WILL BE ALLOWED BEYOND THIS TIME!

Please read carefully: You must submit the answers to all the questions below. However, only one or more questions, possibly chosen at random, will be corrected and will be evaluated to the full mark of the assignment.

Question 1

Is it possible to have a single tree that satisfies the following traversals:

Inorder: **X H C K Y A S M W Q D R L**
Postorder: **X C H Y S A W M D Q L R K**

Question 2

Assume that *linear probing* is used for hash-tables. To improve the time complexity of the operations performed on the table, a special *AVAILABLE* object is used to mark a location when an item is removed from the location. Assuming that all keys are positive integers, the following two techniques were suggested instead of marking the location as AVAILABLE:

i) When an entry is removed, instead of marking its location in the table as AVAILABLE, indicate the key in the location as the negative value of the removed key (e.g., if the removed key was 16, indicate the key as -16). Searching for an entry with the removed key would then terminate once a negative value of the key is found (instead of continuing to search if AVAILABLE is used).

ii) Instead of using AVAILABLE, find a key in the table that should have been placed in the location of the removed entry, then place that key (the entire entry of course) in that location (instead of setting the location as AVAILABLE). The motive is to find the key faster since it is now in its hashed location. This would also avoid the dependence on the AVAILABLE object.

Will either of these approaches have an advantage? You should analyze in terms of both time and space complexities. Additionally, will any of these approaches result in misbehaviors (in terms of functionalities)? If so, explain clearly through illustrative examples.

Question 3

i) Draw the min-heap that results from the **bottom-up** heap construction algorithm on the following list of values:

32, 12, 47, 31, 20, 22, 38, 36, 29, 52, 5, 31, 33, 3, 39.

Starting from the bottom layer, use the values from left to right as specified above. Show immediate steps and the final tree representing the min-heap. Afterwards perform the operation `removeMin` 6 times and show the resulting min-heap after each step.

ii) Create again a min-heap using the list of values from the above part (i) of this question but this time you have to insert these values step by step (i.e. one by one) using the order from left to right (i.e. insert 32, then insert 12, then 47, etc.) as shown in the above question. Show the tree after each step and the final tree representing the min-heap.

Question 4

Assume a hash table utilizes an array of 13 elements and that collisions are handled by separate chaining. Considering the hash function is defined as: $h(k)=k \bmod 13$.

i) Draw the contents of the table after inserting elements with the following keys:

56, 472, 352, 140, 217, 120, 18, 21, 182, 204, 91, 93, 178, 78, 70, 33, 51, 90.

ii) What is the maximum number of collisions caused by the above insertions?

Question 5

Bubble sort is a sorting algorithm that works as follows: Go through the input list/array element by element, comparing the current element with the one that follows it (i.e. compare A[i] with A[i+1]), then swap the two values if they are not sorted. These process is repeated through the list until no swaps can be performed during a pass, meaning that the list is already sorted. Given the following elements:

23 32 72 76 22 73 40 30 20 60 16 74 28 14

i) Show the needed steps for sorting these values into ascending order using Bubble Sort.
ii) Does Bubble Sort have better time complexity compared to Selection Sort? Explain.

Question 6

Show the steps that a radix sort takes when sorting the following array of 3-tuple Integer keys (notice that each digit in the following values represent a key):

5,4,3 3,5,6 2,9,5 6,9,2 4,9,1 9,4,7 7,8,3 9,9,2 4,7,2 1,8,2 2,6,4

Question 7

Assume an *open addressing* hash table implementation, where the size of the array is $N = 19$, and that *double hashing* is performed for collision handling. The second hash function is defined as: $d(k) = q - k \bmod q$, where k is the key being inserted in the table and the prime number q is = 7. Use simple modular operation ($k \bmod N$) for the first hash function.

i) Show the content of the table after performing the following operations, in order:

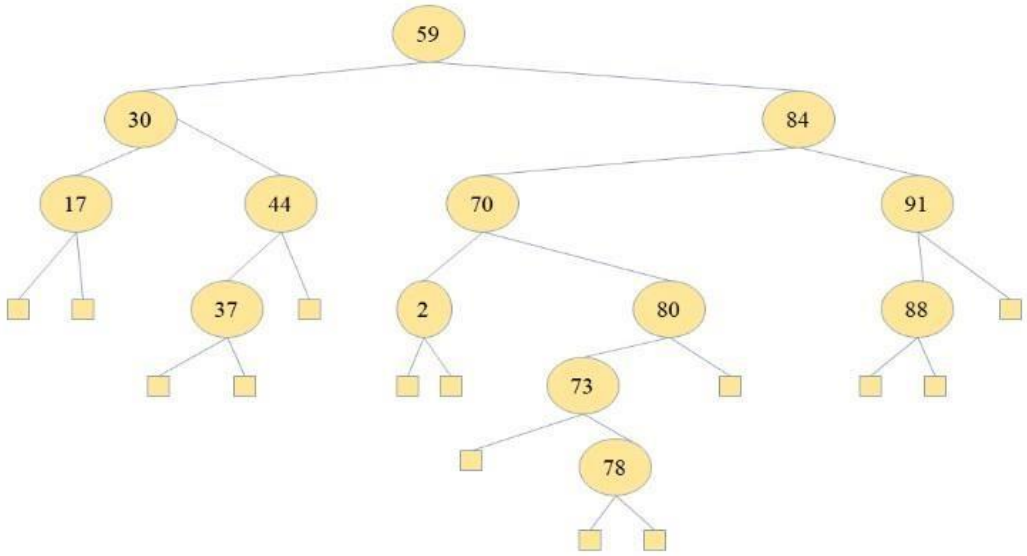
put(45), put(25), put(12), put(61), put(38), put(88), remove(12), put(39), remove(61), put(18), put(29), put(29), put(35).

i) What is the size of the longest cluster caused by the above insertions? ii) What is the number of occurred collisions as a result of the above operations?

iii) What is the current value of the table's *load factor*?

Question 8

Given the following tree, which is assumed to be an AVL tree!



i) Are there any errors with the tree as shown? If so, indicate what the error(s) are, correct these error(s), show the corrected AVL tree, then proceed to the following questions (Questions ii to iv) and start with the tree that you have just corrected. If no errors are there in the above tree, indicate why the tree is correctly an AVL tree, then proceed to the following questions (Questions ii to iv) and continue working on the tree as shown above.

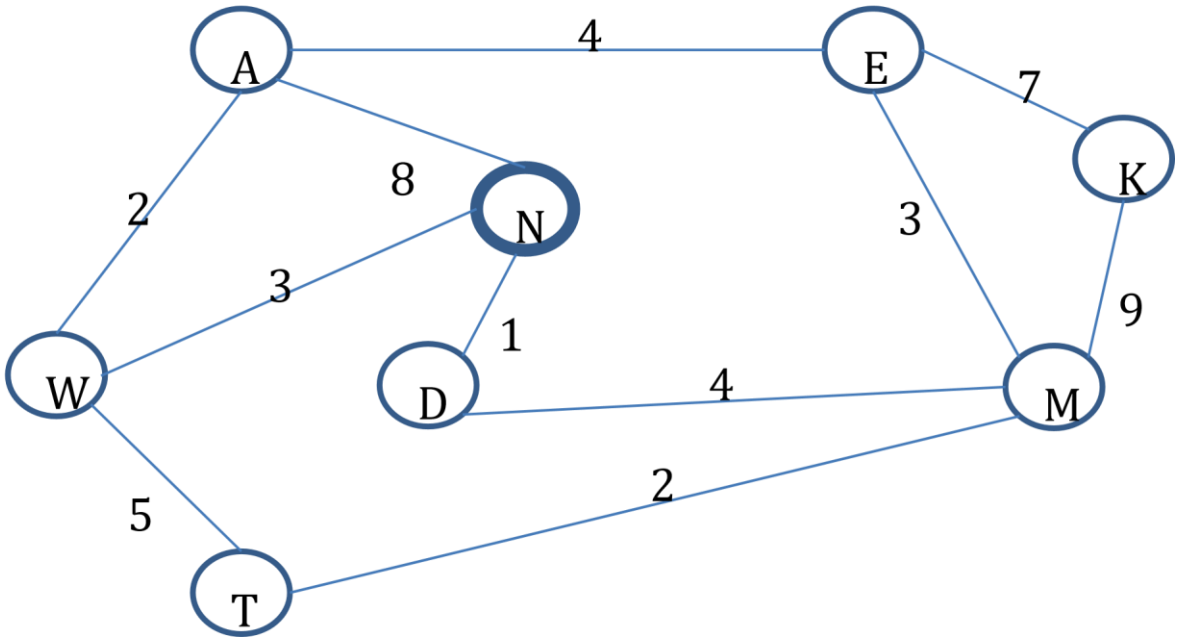
ii) Show the AVL tree after ***put(74)*** operation is performed. What is the complexity of this operation?

iii) Show the AVL tree after ***remove(70)*** is performed. What is the complexity of this operation?

iv) Show the AVL tree after ***remove(91)*** is performed. Show the progress of your work step-by-step. What is the complexity of this operation?

Question 9

Using ***Dijkstra's Algorithm***, find the shortest paths in the following graph from node N to each of the other nodes.



Submission

- **Assignment must be submitted individually (No groups are allowed).**
- **Submit your assignment under the submission folder: Theory Assignment 3.**
- **Submit all your answers as PDF file.**
- **All you text must be typed (no scans or hand-writings). However, you can submit handwritten/scanned drawings. While this will not cost you any marks, a small bonus mark will be given for quality submissions where everything is typeset.**
- **Your text should be concise and brief (¼ of a page for each question) in your answers.**

Submission format: All assignment-related submissions that have more than one document must be adequately archived in a ZIP file using your ID and last name as file name. The submission itself must also contain your name and student ID. Use your "official" name only - no abbreviations or nick names; capitalize the usual "last" name. Inappropriate submissions will be heavily penalized.