BRAINWARE UNIVERSITY

Term End Examination 2023
Programme – M.Tech.(CSE)-2021
Course Name – Advanced Data Structures
Course Code - PCC-MCS101
(Semester I)

Full Marks : 60 **Time : 2:30 Hours**

[The figure in the margin indicates full marks. Candidates are required to give their answers in their own words as far as practicable.]

Group-A
(Multiple Choice Type Question) 1 x 15=15

1. *Choose the correct alternative from the following :*

(i) Recall the key operations that can be performed on skip lists and explain their significance in data organization.

a) Addition and subtraction
b) Search and update
c) Sorting and merging
d) Encryption and decryption

(ii) Explain the significance of suffix tries in text processing and how they assist in solving various string-related problems.

a) Suffix tries are used for character encoding.
b) Suffix tries enable efficient substring searching and pattern matching.
c) Suffix tries are exclusively used for numerical calculations.
d) Suffix tries have no relevance in text processing.

(iii) Apply the Huffman coding algorithm to a set of characters with given frequencies to construct an optimal binary code for data compression.

a) The Huffman coding algorithm is used for data encryption.
b) The Huffman coding algorithm generates random binary codes.
c) The Huffman coding algorithm constructs variable-length binary codes to represent characters based on their frequencies.
d) The Huffman coding algorithm requires fixed-length codes for all characters.

(iv) Recall the problem addressed by the Longest Common Subsequence (LCS) and explain its relevance in string processing.

a) LCS solves the problem of sorting strings.
b) LCS addresses the issue of identifying common characters between two strings.
c) LCS finds the longest sequence of characters that are common to two given strings.
d) LCS eliminates the need for string concatenation.

(v) Apply the concept of constructing a priority search tree to a two-dimensional range searching problem involving points on a plane.

a) Construct a binary search tree for the given points.
b) Construct a priority search tree using the x-coordinates of the points.
c) Construct a priority search tree using the y-coordinates of the points.
d) Construct a priority search tree by randomly selecting points from the plane.

(vi) Recall the purpose of Quadtrees in computational geometry and explain how they facilitate efficient spatial data storage and retrieval.

a) Quadtrees are used for sorting elements in descending order.
b) Quadtrees improve random access to data points in a two-dimensional space.
c) Quadtrees are used exclusively for linear data structures.
d) Quadtrees partition space into regions and enable efficient data representation.

(vii) Explain the concept of k-D Trees and how they extend the idea of binary search trees to higher-dimensional spaces for efficient point queries.

a) k-D Trees only work for one-dimensional data.
b) k-D Trees are used for sorting data in descending order.
c) k-D Trees partition space into regions and allow efficient point queries in higher-dimensional spaces.
d) k-D Trees are limited to two-dimensional data representation.

(viii) Recall the primary focus of recent trends in advanced data structures, such as hashing, trees, and computational geometry methods.

a) Recent trends focus on reinventing basic data structures.
b) Recent trends aim to replace traditional data structures with newer alternatives.
c) Recent trends seek to develop more efficient and effective methods for solving evolving problems using hashing, trees, and computational geometry.
d) Recent trends prioritize removing all data structures in favor of direct memory access.

(ix) Recall the key characteristics of Compressed Tries and how they differ from Standard Tries in terms of space efficiency.

a) Compressed Tries store data in uncompressed form.
b) Compressed Tries use more memory than Standard Tries.
c) Compressed Tries optimize space usage by compressing common prefixes.
d) Compressed Tries have no impact on memory utilization.

(x) Explain the process of constructing a suffix trie for a given string and how it can be utilized for efficient substring search.

a) Suffix Tries only store individual characters.
b) Suffix Tries store all possible substrings of a given string and facilitate fast substring search.
c) Suffix Tries require a separate data structure for substring search.
d) Suffix Tries are only relevant for numerical data.

(xi) Evaluate the trade-offs between deterministic skip lists and probabilistic skip lists in terms of performance, implementation complexity, and their suitability for dynamic data structures.

a) Deterministic skip lists have better performance but are harder to implement.
b) Probabilistic skip lists offer predictable time complexity.
c) Deterministic skip lists are better suited for dynamic data structures.
d) Probabilistic skip lists are only useful for static data.

(xii) Devise a plan for balancing a binary search tree as items are inserted, ensuring that the tree remains balanced for efficient search and update operations.

a) Implement a depth-first traversal for balancing.
b) Use a hash function to maintain tree balance.
c) Perform rotations to ensure tree balance after insertions.
d) Avoid balancing the tree to minimize overhead.

(xiii) Evaluate the benefits of using B-trees over binary search trees for applications that require efficient insertion, deletion, and search operations on large datasets, considering their node structure and balancing mechanisms.

a) B-trees have better performance on small datasets.
b) B-trees store more keys in each node, reducing the height of the tree.
c) Binary search trees offer faster insertion and deletion operations.
d) B-trees are only suitable for read-only applications.

(xiv) Analyze the time complexity of the Knuth-Morris-Pratt algorithm for pattern matching, considering its reliance on preprocessed information and how it efficiently avoids unnecessary character comparisons.

a) Knuth-Morris-Pratt algorithm always requires O(n^2) time complexity.
b) Knuth-Morris-Pratt algorithm relies on hashing for pattern matching.
c) Knuth-Morris-Pratt algorithm achieves O(n) time complexity due to preprocessing.
d) Knuth-Morris-Pratt algorithm uses brute force for character comparisons.

(xv) Analyze the benefits of using priority search trees in two-dimensional range searching applications, considering their structure and ability to efficiently retrieve points that fall within a given range.

a) Priority search trees are inefficient for range searching.
b) Priority search trees use a balanced tree structure for range searching.
c) Priority search trees offer constant time complexity for all range queries.
d) Priority search trees can only search within one-dimensional ranges.

Group-B

(Short Answer Type Questions) 3 x 5=15

2. Explain Brute-force pattern matching algorithm. (3)
3. What do you mean by hash table? Explain. (3)
4. Write down the algorithm for finding the Huffman coding (3)
5. Circular queue is needed to overcome the problem of linear queue – Justify (3)
6. Write a pseudocode to delete a node from the beginning of a singly linked list (3)

OR

Write a pseudocode to delete a node from the end of a singly linked list. (3)

Group-C

(Long Answer Type Questions) 5 x 6=30

7. Show that the hash table that results when the letters in COMPUTERSCIENCE are stored in the given order using the linear probe collision resolution method. Assume a hash table of size 19 and use the hash function H(K)=K MOD 19 for the kth letter of the alphabet. (5)
8. Briefly explain the types of Binary Tree. (5)
9. What is the difference between linear probing and quadratic probing. (5)
10. Compare single, Double and Circular Linked List with logical structure. (5)
11. Construct Tree from given Inorder and Preorder traversals Inorder sequence: DB E A F C Preorder sequence: A B D E C F (5)
12. Explain about Boyer-Moore algorithm with example (5)

OR

Let obtain a set of Huffman code for the message (m1.....m7) with relative frequencies (q1.....q7) = (4,5,7,8,10,12,20). Draw the Huffman tree for the given set of codes. (5)

****