Multiple Choice Q&A

1. What is an AVL Tree?

   a) A type of self-balancing binary search tree

   b) A data structure used for sorting elements

   c) A form of hash table

   d) A variant of linked list

   Answer: a) A type of self-balancing binary search tree

2. How is the height of an AVL Tree defined?

   a) The number of elements in the tree

   b) The maximum depth of the tree

   c) The difference in heights between left and right subtrees

   d) The sum of heights of all nodes in the tree

   Answer: c) The difference in heights between left and right subtrees

3. What operations can be performed on an AVL Tree?

   a) Insertion, merging, and sorting

   b) Addition, subtraction, and multiplication

   c) Insertion, deletion, and searching

   d) Traversal, rotation, and rebalancing

   Answer: c) Insertion, deletion, and searching

4. How is insertion done in an AVL Tree?

   a) Elements are randomly added to the tree

   b) Elements are inserted at the root

   c) Elements are inserted and then the tree is balanced if needed

   d) Elements are always inserted as leaves

   Answer: c) Elements are inserted and then the tree is balanced if needed

5. What is the process of deletion in an AVL Tree?

   a) Deleting the root element

   b) Removing elements randomly

   c) Deleting elements and then balancing the tree if necessary

   d) Deleting only leaf nodes

   Answer: c) Deleting elements and then balancing the tree if necessary

6. How does searching work in an AVL Tree?

   a) Using linear search

   b) Employing binary search

   c) Employing depth-first search

   d) Using breadth-first search

   Answer: b) Employing binary search

7. What is a Red-Black Tree?

   a) A type of graph

   b) A self-balancing binary search tree

   c) A hashing technique

   d) A sorting algorithm

   Answer: b) A self-balancing binary search tree

8. How does a Red-Black Tree differ from an AVL Tree?

   a) AVL Trees guarantee balance more strictly

   b) Red-Black Trees have a more complex structure

   c) Red-Black Trees cannot handle insertion and deletion

   d) AVL Trees use red and black colors for nodes

   Answer: a) AVL Trees guarantee balance more strictly

9. What is the height of a Red-Black Tree?

   a) It is always logarithmic

   b) It is always constant

   c) It varies depending on the number of nodes

   d) It is always linear

   Answer: a) It is always logarithmic

10. What are the main properties of a Red-Black Tree?

a) All leaf nodes are red

b) The root node is always red

c) No two red nodes can be adjacent

d) All nodes must have two children

Answer: c) No two red nodes can be adjacent

11. How does insertion work in a Red-Black Tree?

a) Elements are inserted at random positions

b) Elements are inserted and then the tree is balanced if needed

c) Elements are inserted as leaves

d) Elements are inserted at the root

Answer: b) Elements are inserted and then the tree is balanced if needed

12. What is the rebalancing process in a Red-Black Tree called?

a) Rotation

b) Recoloring

c) Rebalancing

d) Restructuring

Answer: b) Recoloring

13. What is a Splay Tree?

a) A type of self-adjusting binary search tree

b) A tree with no balance constraints

c) A tree structure used in graphics

d) A variant of a heap data structure

Answer: a) A type of self-adjusting binary search tree

14. How does a Splay Tree differ from AVL and Red-Black Trees?

a) It does not maintain any balance properties

b) It has a fixed structure

c) It is based on a hash table

d) It is used only for sorting

Answer: a) It does not maintain any balance properties

15. What is the purpose of a Splay Tree?

a) To ensure logarithmic time for search, insert, and delete operations on frequently accessed elements

b) To maintain balance among nodes

c) To guarantee constant time complexity for all operations

d) To achieve linear time complexity for all operations

Answer: a) To ensure logarithmic time for search, insert, and delete operations on frequently accessed elements

16. How does the splaying operation work in a Splay Tree?

a) It rotates the tree nodes

b) It brings the most recently accessed node to the root

c) It swaps the root with the deepest leaf node

d) It randomly rearranges the tree structure

Answer: b) It brings the most recently accessed node to the root

17. What is the time complexity of splaying in a Splay Tree?

a) O(1)

b) O(log n)

c) O(n)

d) O(n log n)

Answer: b) O(log n)

18. Can a Splay Tree be used for range queries?

a) Yes, it supports efficient range queries

b) No, it does not support range queries

c) Only with additional auxiliary data structures

d) Only if the tree is perfectly balanced

Answer: a) Yes, it supports efficient range queries

19. What is the primary advantage of using Red-Black Trees over AVL Trees?

a) Red-Black Trees have a simpler balancing mechanism

b) Red-Black Trees guarantee faster search operations

c) Red-Black Trees have a lower memory overhead

d) Red-Black Trees allow for faster insertion and deletion operations

Answer: d) Red-Black Trees allow for faster insertion and deletion operations

20. In which scenario would you prefer to use an AVL Tree over a Red-Black Tree?

a) When memory usage is a critical concern

b) When search operations are more frequent than insertions and deletions

c) When insertions and deletions need to be performed faster

d) When the tree needs to be rebalanced less frequently

Answer: b) When search operations are more frequent than insertions and deletions

21. What is the main drawback of Splay Trees?

a) They require a lot of memory

b) They have higher time complexity for search operations compared to AVL Trees

c) They can become unbalanced in certain scenarios

d) They cannot handle dynamic data well

Answer: c) They can become unbalanced in certain scenarios

22. Which operation is more costly in terms of time complexity in a Splay Tree?

a) Insertion

b) Deletion

c) Searching

d) Splaying

Answer: d) Splaying

23. What is the minimum number of children a node can have in a Red-Black Tree?

a) 0

b) 1

c) 2

d) 3

Answer: b) 1

24. What does the term "color flip" refer to in the context of Red-Black Trees?

a) Changing the color of a node from black to red

b) Changing the color of a node from red to black

c) Changing the color of a node and its children

d) Changing the color of the entire tree

Answer: c) Changing the color of a node and its children

25. Which balancing operation is commonly used in AVL Trees?

a) Splitting

b) Reversing

c) Rotating

d) Flipping

Answer: c) Rotating

26. What is a common data structure used for the adjacency list representation of a graph?

a) Array

b) Linked List

c) Hash Table

d) Queue

Answer: b) Linked List

27. In the adjacency matrix representation of a graph, what does a non-zero value at matrix[i][j] indicate?

a) The weight of the edge from vertex i to j

b) No edge exists between vertex i and j

c) An edge exists between vertex i and j

d) The number of edges between vertex i and j

Answer: c) An edge exists between vertex i and j

28. Which graph implementation is generally more space-efficient for sparse graphs?

a) Adjacency matrix

b) Adjacency list

c) Edge list

d) Both a and b

Answer: b) Adjacency list

29. Which of the following is NOT an advantage of using an adjacency list over an adjacency matrix for graph representation?

a) Lower memory usage for sparse graphs

b) Easier to add or remove edges

c) Faster to iterate over all edges

d) Better for dense graphs

Answer: d) Better for dense graphs

30. In an adjacency matrix, how is a weighted graph represented?

a) By using 0 for no edge and 1 for an edge

b) By storing the weight of the edge at matrix[i][j]

c) By using a list of weights for each edge

d) By using a separate matrix for weights

Answer: b) By storing the weight of the edge at matrix[i][j]

31. What data structure can be used to efficiently check if there is an edge between two vertices in a graph?

a) Queue

b) Stack

c) Adjacency matrix

d) Binary search tree

Answer: c) Adjacency matrix

32. How does the space complexity of storing a graph as an adjacency list compare to using an adjacency matrix?

a) It is always lower

b) It is always higher

c) It is the same

d) It depends on the graph's sparsity

Answer: d) It depends on the graph's sparsity

33. What is the time complexity of adding a new edge in the adjacency list representation?

a) O(1)

b) O(log n)

c) O(n)

d) O(n^2)

Answer: a) O(1)

34. For which operation is the adjacency matrix representation more efficient compared to the adjacency list?

a) Adding a new vertex

b) Checking the existence of an edge

c) Adding a new edge

d) Removing an edge

Answer: b) Checking the existence of an edge

35. In graph implementation, what is a typical way to represent a graph with weighted edges using adjacency lists?

a) Lists of vertex indices

b) Arrays of edge weights

c) Lists of (vertex, weight) pairs

d) Separate lists for vertices and weights

Answer: c) Lists of (vertex, weight) pairs

36. Which graph traversal algorithm uses a queue to keep track of the next vertex to visit?

a) Depth-First Search (DFS)

b) Breadth-First Search (BFS)

c) Greedy Best-First Search

d) A* Search

Answer: b) Breadth-First Search (BFS)

37. In Depth-First Search (DFS), if a vertex has no unvisited adjacent vertices, what does the algorithm do next?

a) Visits the next vertex in the queue

b) Revisits the previous vertex

c) Terminates the search

d) Selects a random vertex to continue

Answer: b) Revisits the previous vertex

38. What is the primary difference between recursive and iterative implementations of Depth-First Search (DFS)?

a) The order in which vertices are visited

b) The data structure used to store vertices

c) The space complexity

d) The time complexity

Answer: b) The data structure used to store vertices

39. Which graph traversal method is particularly useful for finding the shortest path in unweighted graphs?

a) Depth-First Search (DFS)

b) Breadth-First Search (BFS)

c) Dijkstra's algorithm

d) Bellman-Ford algorithm

Answer: b) Breadth-First Search (BFS)

40. What type of graph traversal would likely be most efficient for solving a maze?

a) Breadth-First Search (BFS)

b) Depth-First Search (DFS)

c) Greedy Best-First Search

d) A* Search

Answer: b) Depth-First Search (DFS)

41. Which of the following is a characteristic of Breadth-First Search (BFS)?

a) It explores all paths simultaneously

b) It always uses a stack for storing vertices

c) It can get trapped in cycles without special precautions

d) It is guaranteed to find the deepest node first

Answer: a) It explores all paths simultaneously

42. How does Depth-First Search (DFS) determine which vertex to visit next?

a) By choosing the vertex that is closest to the start vertex

b) By choosing the next unvisited vertex in the adjacency list

c) By backtracking when no adjacent unvisited vertices are found

d) By randomly selecting an unvisited adjacent vertex

Answer: b) By choosing the next unvisited vertex in the adjacency list

43. What is a common application of graph traversal algorithms?

a) Sorting numbers

b) Pattern matching in strings

c) Finding connected components in a graph

d) Calculating the factorial of a number

Answer: c) Finding connected components in a graph

44. Which of the following statements is true about graph traversal algorithms?

a) DFS is generally faster than BFS for all types of graphs

b) BFS can be implemented using recursion

c) DFS can be implemented using either a stack or recursion

d) BFS is more memory-efficient than DFS in all cases

Answer: c) DFS can be implemented using either a stack or recursion

45. In which scenario is it more advantageous to use Breadth-First Search (BFS) over Depth-First Search (DFS)?

a) When the graph is very deep and solutions are rare

b) When the solution is known to be not far from the root of the graph

c) When the graph has very few branches

d) When a comprehensive search of the graph is required

Answer: b) When the solution is known to be not far from the root of the graph

46. What is the first step in the heap sort algorithm?

a) Sorting the array in ascending order

b) Building a max heap from the unsorted array

c) Swapping the first and last elements of the array

d) Dividing the array into sorted and unsorted sections

Answer: b) Building a max heap from the unsorted array

47. During heap sort, when is the heap property restored?

a) After every insertion

b) After removing the root of the heap

c) Before adding a new element to the heap

d) Once at the beginning of the sort

Answer: b) After removing the root of the heap

48. What is the time complexity of heap sort in the worst case?

a) O(n log n)

b) O(n^2)

c) O(n)

d) O(log n)

Answer: a) O(n log n)

49. In heap sort, which part of the array is considered the "heap"?

a) The entire array

b) The first half of the array

c) The unsorted portion of the array

d) The last element of the array

Answer: c) The unsorted portion of the array

50. What happens to the max heap after the maximum element is removed during the heap sort process?

a) It is converted into a min heap

b) It is rebuilt from scratch

c) The heap size is decreased, and the heap property is restored

d) The algorithm terminates

Answer: c) The heap size is decreased, and the heap property is restored

51. What is a key characteristic of heap sort compared to other sorting algorithms?

a) It uses a comparison-based sorting technique

b) It does not require any extra space for sorting

c) It is a stable sort

d) It can sort linked lists efficiently

Answer: b) It does not require any extra space for sorting

52. Which data structure is primarily used in the implementation of heap sort?

a) Array

b) Linked List

c) Binary Tree

d) Graph

Answer: a) Array

53. What is adjusted in the heap during the "heapify" process of heap sort?

a) The positions of the root elements only

b) The depth of the heap

c) The structure of the heap to maintain the heap property

d) The order of elements in the sorted portion of the array

Answer: c) The structure of the heap to maintain the heap property

54. How does heap sort compare to quicksort in terms of average performance?

a) Heap sort is faster on average

b) Quicksort is faster on average

c) They have the same average performance

d) It depends on the type of data

Answer: b) Quicksort is faster on average

55. Which of the following is true about the space complexity of heap sort?

a) It has O(n) space complexity due to the heap storage

b) It has O(log n) space complexity due to recursion

c) It is not space-efficient and requires additional memory for the heap

d) It has O(1) space complexity and is considered an in-place sorting algorithm

Answer: d) It has O(1) space complexity and is considered an in-place sorting algorithm

56. What is the primary goal of external sorting algorithms?

a) To sort data that fits entirely in memory

b) To sort large datasets that do not fit in memory

c) To improve cache performance during sorting

d) To reduce the number of comparisons in sorting

Answer: b) To sort large datasets that do not fit in memory

57. Which technique is commonly used in external sorting to manage data?

a) Quick sort

b) Merge sort

c) Radix sort

d) Heap sort

Answer: b) Merge sort

58. In the context of external sorting, what is a "run"?

a) A sequence of instructions in the sorting algorithm

b) A sorted sequence of data from the dataset

c) The total time taken to sort the dataset

d) A random subset of data used for sorting

Answer: b) A sorted sequence of data from the dataset

59. What is the first step in a typical external merge sort algorithm?

a) Merging sorted runs from disk

b) Creating initial runs on disk

c) Sorting the entire dataset in memory

d) Dividing the data into equal parts

Answer: b) Creating initial runs on disk

60. How does external sorting typically handle the sorted runs?

a) By combining them using a binary tree

b) By merging them into larger sorted sequences

c) By sorting each run again in memory

d) By randomly shuffling them to improve efficiency

Answer: b) By merging them into larger sorted sequences

61. Which of the following is crucial for reducing the number of disk accesses in external sorting?

a) Decreasing the size of each run

b) Increasing the number of runs

c) Minimizing the number of passes over the data

d) Using a faster sorting algorithm for initial runs

Answer: c) Minimizing the number of passes over the data

62. In external sorting, what role does a buffer play?

a) It temporarily stores data to be sorted

b) It holds the final sorted data before writing to disk

c) It is used to cache the next set of data to be read from disk

d) It stores intermediate results during the merge process

Answer: c) It is used to cache the next set of data to be read from disk

63. What is a primary challenge in external sorting not present in internal sorting?

a) Determining the pivot in quicksort

b) Managing memory usage efficiently

c) Minimizing the computational complexity

d) Optimizing disk I/O operations

Answer: d) Optimizing disk I/O operations

64. How does the external merge sort algorithm differ from traditional merge sort?

a) It uses a different merging technique

b) It performs sorting in parallel

c) It sorts data in chunks that fit into memory

d) It relies solely on disk storage for sorting

Answer: c) It sorts data in chunks that fit into memory

65. Which factor most influences the efficiency of external sorting?

a) The speed of the CPU

b) The amount of RAM available

c) The speed of disk access

d) The complexity of the sorting algorithm

Answer: c) The speed of disk access

66. What is the basic principle behind merge sort?

a) Dividing the array into halves and sorting each half

b) Selecting a pivot and organizing elements around it

c) Building a heap and then sorting the array

d) Inserting elements into their correct position

Answer: a) Dividing the array into halves and sorting each half

67. What is the time complexity of merge sort in the worst case?

a) O(n log n)

b) O(n^2)

c) O(n)

d) O(log n)

Answer: a) O(n log n)

68. In merge sort, what happens during the "merge" step?

a) The array is divided into two halves

b) Two sorted halves are combined into a single sorted sequence

c) The largest element is found and placed at the end of the array

d) Elements are swapped until the array is sorted

Answer: b) Two sorted halves are combined into a single sorted sequence

69. What is the space complexity of the merge sort algorithm?

a) O(n)

b) O(log n)

c) O(1)

d) O(n log n)

Answer: a) O(n)

70. Which of the following is a disadvantage of merge sort compared to other sorting algorithms?

a) It has a higher time complexity

b) It cannot sort linked lists

c) It requires additional space for merging

d) It is not a comparison-based sort

Answer: c) It requires additional space for merging

71. How does merge sort perform on nearly sorted arrays?

a) It is slower than on randomly ordered arrays

b) It performs significantly faster

c) Its performance is unaffected by the order of elements

d) It requires more space

Answer: c) Its performance is unaffected by the order of elements

72. What is the best-case time complexity of merge sort?

a) O(n)

b) O(n^2)

c) O(n log n)

d) O(log n)

Answer: c) O(n log n)

73. How does merge sort ensure stability in sorting?

a) By not changing the order of equal elements

b) By using a heap to manage elements

c) By selecting a pivot for partitioning

d) By reversing the array before sorting

Answer: a) By not changing the order of equal elements

74. Can merge sort be used for external sorting?

a) Yes, it is well-suited for handling large datasets

b) No, it can only sort data that fits in memory

c) Yes, but only with modifications to its basic algorithm

d) No, because it requires too much additional space

Answer: a) Yes, it is well-suited for handling large datasets

75. What modification can improve the space efficiency of merge sort in practice?

a) Using a linked list instead of an array

b) Implementing the sort in-place without extra arrays

c) Reducing the number of recursive calls

d) Sorting small subarrays with an alternative method and then merging

Answer: b) Implementing the sort in-place without extra arrays

76. Which algorithm is known for its simple but inefficient approach in pattern matching?

a) Boyer-Moore algorithm

b) Knuth-Morris-Pratt algorithm

c) Brute force algorithm

d) Suffix tries algorithm

Answer: c) Brute force algorithm

77. Which algorithm among the listed ones is specifically efficient in handling pattern matching with preprocessing?

a) Boyer-Moore algorithm

b) Brute force algorithm

c) Standard Tries algorithm

d) Compressed Tries algorithm

Answer: a) Boyer-Moore algorithm

78. In which scenario is the Brute force algorithm most suitable for pattern matching?

a) Small text and small pattern

b) Small text and large pattern

c) Large text and small pattern

d) Large text and large pattern

Answer: a) Small text and small pattern

79. Which algorithm is primarily known for its "right-to-left" approach in pattern matching?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Standard Tries algorithm

Answer: b) Boyer-Moore algorithm

80. The Knuth-Morris-Pratt algorithm is efficient due to its utilization of which technique?

a) Preprocessing

b) Dynamic programming

c) Greedy approach

d) Backtracking

Answer: a) Preprocessing

81. Which data structure is typically used in the implementation of the Knuth-Morris-Pratt algorithm?

a) Array

b) Linked list

c) Stack

d) Trie

Answer: a) Array

82. What distinguishes Standard Tries from Compressed Tries?

a) Standard Tries are more memory-efficient.

b) Standard Tries are faster in lookup operations.

c) Compressed Tries are more memory-efficient.

d) Compressed Tries support fewer characters.

Answer: a) Standard Tries are more memory-efficient.

83. Which algorithm is particularly suitable for dealing with large collections of strings?

a) Brute force algorithm

b) Suffix tries algorithm

c) Boyer-Moore algorithm

d) Knuth-Morris-Pratt algorithm

Answer: b) Suffix tries algorithm

84. What is the primary advantage of using Compressed Tries over Standard Tries?

a) Faster lookup operations

b) Reduced memory consumption

c) Easier implementation

d) Better support for dynamic resizing

Answer: b) Reduced memory consumption

85. Which algorithm is not suitable for handling patterns with repeating characters efficiently?

a) Boyer-Moore algorithm

b) Brute force algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: c) Knuth-Morris-Pratt algorithm

86. Which among the listed algorithms provides an efficient approach for handling multiple pattern searches in a given text?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Suffix tries algorithm

d) Knuth-Morris-Pratt algorithm

Answer: c) Suffix tries algorithm

87. What is the main limitation of using Brute force algorithm in pattern matching?

a) High memory consumption

b) Inefficient for large patterns

c) Requires complex preprocessing

d) Limited character set support

Answer: b) Inefficient for large patterns

88. The Boyer-Moore algorithm uses which strategy for character comparison during pattern matching?

a) Left-to-right

b) Right-to-left

c) Random

d) Bidirectional

Answer: b) Right-to-left

89. Which algorithm's efficiency heavily depends on the length of the pattern?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: c) Knuth-Morris-Pratt algorithm

90. In which type of pattern matching scenario does the Boyer-Moore algorithm excel?

a) Patterns with unique characters

b) Patterns with repeating characters

c) Patterns with large alphabets

d) Patterns with small alphabets

Answer: b) Patterns with repeating characters

91. What is a common application of Suffix tries?

a) Text compression

b) Pattern matching

c) Sorting algorithms

d) Cryptography

Answer: b) Pattern matching

92. Which algorithm utilizes the concept of "skipping" characters during pattern matching?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: b) Boyer-Moore algorithm

93. Which algorithm is considered the most efficient for searching for a single pattern in a text?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: b) Boyer-Moore algorithm

94. What makes the Knuth-Morris-Pratt algorithm more efficient than the Brute force algorithm?

a) Dynamic programming

b) Greedy approach

c) Preprocessing

d) Backtracking

Answer: c) Preprocessing

95. Which algorithm is particularly useful for applications where memory usage is a concern?

a) Boyer-Moore algorithm

b) Suffix tries algorithm

c) Brute force algorithm

d) Knuth-Morris-Pratt algorithm

Answer: b) Suffix tries algorithm

96. What aspect of the Boyer-Moore algorithm contributes to its efficiency in pattern matching?

a) Dynamic programming

b) Greedy approach

c) Preprocessing

d) Backtracking

Answer: c) Preprocessing

97. Which algorithm's efficiency relies on the concept of "failure function"?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: c) Knuth-Morris-Pratt algorithm

98. In which type of data retrieval application are Standard Tries commonly used?

a) Web search engines

b) Spell checkers

c) Image processing

d) Network routing

Answer: b) Spell checkers

99. Which algorithm is particularly advantageous for pattern matching in DNA sequencing?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: d) Suffix tries algorithm

100. What makes Suffix tries suitable for DNA sequencing applications?

a) High memory efficiency

b) Fast lookup operations

c) Support for variable-length patterns

d) Ability to handle large alphabets

Answer: c) Support for variable-length patterns

101. Which algorithm is known for its ability to efficiently handle patterns with wildcards or mismatches?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: b) Boyer-Moore algorithm

102. Which algorithm typically requires the least amount of preprocessing before pattern matching?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: a) Brute force algorithm

103. What is the main drawback of using Suffix tries in terms of memory usage?

a) It requires excessive preprocessing.

b) It consumes a large amount of memory.

c) It cannot handle variable-length patterns.

d) It does not support dynamic resizing.

Answer: b) It consumes a large amount of memory.

104. Which algorithm is best suited for scenarios where the text remains constant, but patterns are frequently changed?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: d) Suffix tries algorithm

105. What is a primary advantage of the Boyer-Moore algorithm over the Knuth-Morris-Pratt algorithm?

a) It has better worst-case time complexity.

b) It requires less preprocessing time.

c) It can handle larger patterns efficiently.

d) It is more memory-efficient.

Answer: c) It can handle larger patterns efficiently.

106. Which algorithm is particularly efficient in scenarios where the alphabet size is relatively small?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: d) Suffix tries algorithm

107. In which scenario would one prefer using Standard Tries over Compressed Tries?

a) When memory usage is a concern

b) When patterns are highly repetitive

c) When patterns are of variable length

d) When patterns are relatively short

Answer: c) When patterns are of variable length

108. What is a significant limitation of Standard Tries compared to Compressed Tries?

a) Inefficient lookup operations

b) High memory consumption

c) Limited character set support

d) Inability to handle dynamic resizing

Answer: b) High memory consumption

109. Which algorithm is least affected by the presence of repeating characters in patterns?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: d) Suffix tries algorithm

110. Which algorithm's efficiency depends heavily on the length of the text being searched?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: b) Boyer-Moore algorithm

111. What is a common application of Standard Tries?

a) DNA sequencing

b) Text compression

c) Spell checking

d) Image processing

Answer: c) Spell checking

112. Which algorithm is particularly suitable for pattern matching in streaming data?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: d) Suffix tries algorithm

113. In which scenario does the Knuth-Morris-Pratt algorithm exhibit superior performance over the Boyer-Moore algorithm?

a) Large alphabet size

b) Small pattern size

c) Repeating pattern characters

d) Sparse pattern occurrences

Answer: d) Sparse pattern occurrences

114. Which algorithm's efficiency can be improved by utilizing multiple pattern searches simultaneously?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: d) Suffix tries algorithm

115. What is a significant disadvantage of using Brute force algorithm in real-world applications?

a) It requires extensive preprocessing.

b) It has poor average-case time complexity.

c) It cannot handle variable-length patterns.

d) It consumes excessive memory.

Answer: b) It has poor average-case time complexity.

116. Which algorithm is known for its ability to efficiently handle patterns with overlapping occurrences?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: d) Suffix tries algorithm

117. What type of patterns does the Boyer-Moore algorithm struggle with compared to other algorithms?

a) Patterns with few unique characters

b) Patterns with many repeating characters

c) Patterns with variable lengths

d) Patterns with overlapping occurrences

Answer: a) Patterns with few unique characters

118. Which algorithm is advantageous for pattern matching in scenarios where text preprocessing time is critical?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: a) Brute force algorithm

119. What is a primary disadvantage of using Suffix tries for pattern matching?

a) High preprocessing time

b) Inefficient lookup operations

c) Limited pattern support

d) Excessive memory consumption

Answer: d) Excessive memory consumption

120. Which algorithm exhibits the highest worst-case time complexity among the listed ones?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: c) Knuth-Morris-Pratt algorithm

121. In which scenario is the Brute force algorithm the most practical choice for pattern matching?

a) Sparse pattern occurrences

b) Large alphabet size

c) Short text length

d) Variable-length patterns

Answer: c) Short text length

122. Which algorithm is particularly suitable for handling patterns with a large alphabet size?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: b) Boyer-Moore algorithm

123. What is a common application of Compressed Tries?

a) Spell checking

b) DNA sequencing

c) Image processing

d) Network routing

Answer: d) Network routing

124. Which algorithm's efficiency can be significantly affected by the presence of repeating patterns in the text?

a) Brute force algorithm

b) Boyer-Moore algorithm

c) Knuth-Morris-Pratt algorithm

d) Suffix tries algorithm

Answer: b) Boyer-Moore algorithm