Western Governors University (WGU) ICSC2100 C949 Data Structures and Algorithms I Practice Exam (Sample)

Study Guide

Everything you need from our exam experts!

Questions

1. How does bubble sort function in sorting?

   A. By continuously comparing and swapping adjacent elements

   B. By selecting pivot elements for partitioning

   C. By allocating nodes using a stack

   D. By recursively dividing the array

2. How many children can each node in a tree have?

   A. One

   B. Two

   C. Zero or more

   D. At most four

3. What type of collections allow for duplicate values?

   A. Sets

   B. Dictionaries

   C. Lists

   D. Tuples

4. How can a node with one child be removed from a tree?

   A. By deleting both nodes

   B. By promoting the child

   C. By replacing it with a leaf

   D. By relinking parent directly to grandchild

5. In graph theory, what represents an item within a graph?

   A. Edge

   B. Vertex

   C. Node

   D. List

6. Which sorting algorithm is known for selecting a 'pivot' element?

A. Bubble sort

B. Merge sort

C. Quicksort

D. Insertion sort

7. What happens when a collision occurs in a hash table?

A. A new hash function must be applied

B. The existing key-value pair is overwritten

C. The data is lost

D. A method to resolve the collision is applied

8. Which operation is commonly associated with sets in data structures?

A. Slicing

B. Union

C. Filtering

D. Sorting

9. Iterating over a collection of data once typically indicates which type of algorithm?

A. O(log n)

B. O(n)

C. O(n^2)

D. O(nm)

10. What is recursion?

A. A programming technique where a function calls itself

B. A method for sorting data structures

C. An algorithm for searching data

D. A way to manage memory usage

Answers

1. A
2. C
3. C
4. B
5. B
6. C
7. D
8. B
9. B
10. A

Explanations

1. How does bubble sort function in sorting?

A. By continuously comparing and swapping adjacent elements

B. By selecting pivot elements for partitioning

C. By allocating nodes using a stack

D. By recursively dividing the array

Bubble sort functions by continuously comparing and swapping adjacent elements to sort a list of values. This sorting algorithm works by iterating through the array multiple times and checking pairs of adjacent items. If the first item in the pair is greater than the second, they are swapped. This process is repeated, gradually "bubbling" the largest unsorted element to its correct position at the end of the array with each pass. The core concept relies on the idea that with each complete iteration of the array, the next largest element is placed in its final position, allowing subsequent iterations to focus on a smaller subset of the array. This algorithm is simple and intuitive, making it easy to implement, but it is not the most efficient for large datasets due to its average and worst-case time complexity of $O(n^2)$. In contrast, other sorting methods mentioned, such as those involving pivot selection or recursion, are based on fundamentally different principles, such as quicksort or mergesort, where elements are partitioned or divided into subarrays. These methods are typically more efficient than bubble sort for larger datasets.

2. How many children can each node in a tree have?

A. One

B. Two

C. Zero or more

D. At most four

In a tree data structure, each node can have zero or more children, which means that there is no upper limit on the number of children a single node can possess. This flexibility allows for the representation of various types of hierarchical relationships. For instance, in a binary tree, each node is restricted to a maximum of two children (known as left and right children). However, in more generalized tree structures such as n-ary trees or multi-way trees, a node can have any number of children, which aligns with the concept presented in the correct response. This characteristic of trees makes them versatile for different applications, such as representing file systems, organizational structures, and other hierarchical data representations. Therefore, understanding that a node can indeed have zero or more children provides a broader perspective on how trees can be utilized in programming and data organization.

3. What type of collections allow for duplicate values?

A. Sets

B. Dictionaries

C. Lists

D. Tuples

The correct answer is that lists allow for duplicate values. In a list, elements can be added multiple times, which means the same value can appear more than once. This characteristic is essential in many programming scenarios, such as when the order of elements matters, or when a specific count of a particular value is required in the collection. For example, in a list of student names, the name "John" can appear multiple times for different students. This feature is useful when you need to maintain an indexed order and allow for repeated entries. While dictionaries, sets, and tuples have their own unique properties, they either do not support duplicates (like sets) or have constraints on immutability or key uniqueness (like dictionaries). Therefore, lists are the quintessential data structure for scenarios where duplicate values are needed.

4. How can a node with one child be removed from a tree?

A. By deleting both nodes

B. By promoting the child

C. By replacing it with a leaf

D. By relinking parent directly to grandchild

When removing a node that has one child from a tree, promoting the child is the appropriate action. This process involves updating the parent of the node to point directly to its only child, thereby effectively removing the node while still preserving the tree's structure. Promoting the child means that the single child of the node takes the place of the node being deleted. This action maintains the binary search tree properties if the tree is organized as such, as the promotion of the child maintains all necessary ordering. The parent now references the child directly, allowing it to continue functioning within the tree without introducing any inconsistencies. The other methods of removal, such as deleting both nodes or replacing the node with a leaf, do not accurately represent valid or efficient techniques in maintaining tree structure. Relinking a parent directly to a grandchild is only applicable in scenarios where a node has two children and not one, as it would require careful consideration of the subtree to ensure that the binary search tree properties are preserved. Thus, promoting the child accurately addresses the removal process when a node has only one child.

5. In graph theory, what represents an item within a graph?

A. Edge

B. Vertex

C. Node

D. List

In graph theory, a vertex represents an item within a graph. A vertex (also known as a node) is a fundamental unit by which graphs are formed, typically representing entities such as points, locations, or objects in a model. Vertices are connected by edges, which represent the relationships or connections between those items. Understanding vertices is crucial in various applications, including social networks, transportation systems, and computer networks, where each vertex might represent a person, a city, or a computer, respectively. Each vertex may hold data and has connections (edges) to other vertices that describe how they relate to each other. While a node is often used interchangeably with the term "vertex" in certain contexts, in the strict sense of graph theory, the term vertex is preferred in most formal definitions. Lists are data structures used to store collections of items but do not represent items within a graph itself. Edges represent the connections between vertices but are not standalone items within the graph.

6. Which sorting algorithm is known for selecting a 'pivot' element?

A. Bubble sort

B. Merge sort

C. Quicksort

D. Insertion sort

Quicksort is a highly efficient sorting algorithm that utilizes the concept of a 'pivot' element as a key aspect of its sorting process. In this algorithm, the pivot is chosen from the array elements, and the other elements are rearranged into two partitions: those less than the pivot and those greater than the pivot. This process is recursive, meaning that quicksort will repeatedly select new pivots and partition the subarrays until the entire array is sorted. The selection of a pivot is crucial as it significantly impacts the performance of the algorithm; ideally, a good pivot will lead to evenly sized partitions, resulting in optimal efficiency. Quicksort has an average time complexity of O(n log n), making it faster than various other sorting algorithms for large datasets in most practical applications.

7. What happens when a collision occurs in a hash table?

A. A new hash function must be applied

B. The existing key-value pair is overwritten

C. The data is lost

D. A method to resolve the collision is applied

When a collision occurs in a hash table, it indicates that two different keys hash to the same index in the underlying array. To properly handle this situation, a collision resolution method is applied. This is essential because the integrity of the hash table must be maintained, ensuring that all key-value pairs are accessible. Common methods for resolving collisions include: 1. **Chaining**: This allows multiple key-value pairs to be stored at the same index using a linked list or another data structure. If a collision occurs, the new key-value pair is simply added to the list at that index. 2. **Open Addressing**: This technique involves finding another empty slot within the hash table based on a probing sequence when a collision occurs. Techniques such as linear probing, quadratic probing, or double hashing can be used to find the next available slot. By implementing one of these resolution strategies, the hash table can effectively manage collisions while still maintaining fast access to stored elements. Therefore, applying a method to resolve the collision is the correct and essential response when a collision occurs in a hash table.

8. Which operation is commonly associated with sets in data structures?

A. Slicing

B. Union

C. Filtering

D. Sorting

The correct choice is associated with sets in data structures and refers to the operation that combines two distinct sets to create a new set that contains all the unique elements from both sets. This operation, known as union, is fundamental to set theory and plays a critical role in many algorithms that require the aggregation of unique items. In a set data structure, the union operation is particularly important because one of the defining characteristics of a set is that it inherently prevents duplicate values. When performing a union, the elements are combined while automatically discarding any duplicates, which is efficient in maintaining the integrity of the set. Visualizing this, if you have two sets—let's say Set A containing {1, 2, 3} and Set B containing {3, 4, 5}—the union of these two sets would result in a new set that reflects all unique elements: {1, 2, 3, 4, 5}. This operation demonstrates how sets can be used effectively to manage collections of items in programming and algorithm design. The other operations mentioned, like slicing, filtering, and sorting, are more typically associated with lists or arrays. Slicing refers to accessing a subset of elements from data structures, filtering is about selecting

9. Iterating over a collection of data once typically indicates which type of algorithm?

A. O(log n)

B. O(n)

C. O(n^2)

D. O(nm)

The scenario described—iterating over a collection of data once—demonstrates a linear time complexity, which is denoted as O(n). In this case, 'n' represents the number of elements in the collection. When you execute an operation that requires visiting each element exactly one time, the time taken grows linearly with the increase in the number of elements. Thus, if you double the size of the dataset, the time required to complete the iteration also approximately doubles. This characteristic aligns with the definition of O(n): the execution time is directly proportional to the size of the input data set. Other complexities, such as O(log n), O(n^2), and O(nm), represent more complex relationships where the time taken does not increase linearly with the number of elements. For example, O(log n) indicates logarithmic growth, which occurs in algorithms like binary search that cut the dataset size significantly with each step. Meanwhile, O(n^2) indicates a quadratic relationship, often seen in nested loops over the same dataset, meaning the time grows exponentially compared to the input size. O(nm) implies a situation where two different collections are involved and results in a more elaborate calculation based on the sizes of both collections

10. What is recursion?

A. A programming technique where a function calls itself

B. A method for sorting data structures

C. An algorithm for searching data

D. A way to manage memory usage

Recursion is a programming technique where a function calls itself in order to solve a problem. It typically involves breaking down a problem into smaller, more manageable subproblems, allowing the function to tackle each subproblem individually. This approach often uses a base case to terminate the recursive calls and prevent infinite loops. Recursive solutions can be particularly elegant for processes that have a natural recursive structure, such as traversing trees or calculating factorials. The other options describe different concepts in computer science. For example, sorting data structures refers to organizing data in a specific order, searching algorithms focus on finding elements in collections, and managing memory usage involves techniques to handle how data is stored or retrieved in a program's memory. These concepts are distinct from recursion, which specifically pertains to self-referential function calls.