30 Questions and Answers Based on PDF Content

1. What is mathematical induction?

Answer: Mathematical induction is a proof technique used to prove that a statement holds for all natural numbers by proving it for a base case and then showing that if it holds for an arbitrary case, it also holds for the next case.

2. What are the steps of mathematical induction?

Answer: The steps are: 1. Base case: Prove the statement for the smallest value (usually n = 0 or n = 1). 2. Inductive hypothesis: Assume the statement holds for n = k. 3. Inductive step: Prove the statement holds for n = k + 1.

3. What is recursion in computer science?

Answer: Recursion is a technique where a function calls itself to solve smaller instances of a problem until reaching a base case.

4. What is an example of a recursive algorithm in the document?

Answer: An example is the recursive algorithm for calculating the factorial of a number: n! = n × (n-1)!

5. What is the time complexity of the recursive factorial algorithm?

Answer: The time complexity of the recursive factorial algorithm is O(n).

6. What is a stack used for in recursion?

Answer: A stack is used to store the state of each function call, including parameters and return addresses, so that the program can resume correctly after each recursive call.

7. What is an iterative version of the factorial algorithm?

Answer: The iterative version uses a loop to calculate the factorial by multiplying a variable by

decreasing values of n until n = 0.

8. What is the Tower of Hanoi problem?

Answer: The Tower of Hanoi is a classic problem where n disks need to be moved from one peg to another, using a third peg as an auxiliary, without placing a larger disk on a smaller one.

9. What is the time complexity of the Tower of Hanoi problem?

Answer: The time complexity is T(n) = 2T(n-1) + a, which grows exponentially as O(2^n).

10. What is a recurrence relation?

Answer: A recurrence relation is an equation that describes a function in terms of its values at smaller inputs, often used to express the time complexity of recursive algorithms.

11. What is the purpose of a base case in recursion?

Answer: The base case is the simplest instance of the problem that can be solved directly, preventing infinite recursion by providing a termination point.

12. What is tail recursion?

Answer: Tail recursion is a special form of recursion where the recursive call is the last operation in the function, allowing for optimizations that reduce stack usage.

13. What is dynamic memory allocation?

Answer: Dynamic memory allocation is the process of allocating memory at runtime, often used in recursion when the number of function calls cannot be determined at compile time.

14. How can recursion be converted to iteration?

Answer: Recursion can be converted to iteration by using an explicit stack to manage the state that would otherwise be handled by recursive function calls.

15. What is the space complexity of recursive algorithms?

Answer: The space complexity is usually O(n), where n is the depth of the recursive calls.

16. Why might recursion be more readable than iteration?

Answer: Recursion can be more readable because it often directly mirrors the problem's structure, making it easier to understand and implement for problems like tree traversal.

17. What is a recurrence tree?

Answer: A recurrence tree is a tool used to solve recurrence relations by visualizing how the cost of recursive calls accumulates across the levels of recursion.

18. What is an example of an algorithm with exponential time complexity?

Answer: The recursive Fibonacci algorithm has exponential time complexity, specifically O(2^n), due to the repeated calculations of the same subproblems.

19. What is the iterative version of the Fibonacci algorithm?

Answer: The iterative version calculates Fibonacci numbers using a loop that stores the last two values and computes the next one in the sequence.

20. What is the time complexity of the iterative Fibonacci algorithm?

Answer: The time complexity of the iterative Fibonacci algorithm is O(n).

21. What is the principle of recursion in algorithm design?

Answer: The principle of recursion in algorithm design is to break a problem into smaller subproblems, solve those recursively, and combine their solutions to solve the original problem.

22. What is tail-call optimization?

Answer: Tail-call optimization is an optimization technique where tail-recursive functions are optimized to avoid adding new frames to the stack, effectively turning recursion into iteration.

23. What are the two main types of memory allocation in recursion?

Answer: The two main types are static allocation, which occurs at compile time, and dynamic allocation, which occurs at runtime and is managed by the system.

24. What is the difference between static and dynamic variables?

Answer: Static variables are allocated at compile time and exist for the lifetime of the program, while dynamic variables are allocated at runtime and are released when no longer needed.

25. What is the purpose of an explicit stack in converting recursion to iteration?

Answer: An explicit stack is used to manually simulate the recursive function calls, managing the variables and return addresses that would otherwise be handled by the system's call stack.

26. What is an example of an efficient iterative algorithm?

Answer: An efficient iterative algorithm for finding the factorial of a number uses a loop, reducing both time and space complexity compared to the recursive version.

27. What is the space complexity of the Tower of Hanoi algorithm?

Answer: The space complexity of the Tower of Hanoi algorithm is O(n), where n is the number of disks.

28. How does the complexity of recursive functions differ from iterative functions?

Answer: Recursive functions may have higher time and space complexity due to the overhead of maintaining the call stack, while iterative functions tend to be more efficient in terms of both.

29. What is the difference between top-down and bottom-up approaches in recursion?

Answer: In the top-down approach, recursion solves the problem by breaking it into smaller subproblems, while in the bottom-up approach, iteration builds up the solution from the simplest cases.

30. What is an example of a recursive problem involving binary trees?

Answer: A common recursive problem involving binary trees is tree traversal, where the algorithm

visits each node of the tree in a specific order, such as in-order, pre-order, or post-order traversal.