Class 11 Computer Chapter 3: Algorithms and Problem Solving

SHORT QUESTIONS ANSWERS (ADDITIONAL)

Understanding Computational Problems

Q1 **What is a computational problem?**

Ans. A computational problem is one that can be solved using an algorithm, a clear set of steps that a computer can follow to turn input into output.

Q2 **What are the three parts of solving a computational problem?**

Ans. The three parts are: input (what you start with), process (steps to solve it), and output (the result after processing).

Q3 **What is an algorithm?**

Ans. An algorithm is a step-by-step plan to solve a problem, like a recipe that tells a computer exactly what to do.

Q4 **Why are algorithms important in computing?**

Ans. Algorithms help computers solve problems quickly and correctly. They are the core of all software and apps we use every day.

Q5 **What is input in a computational problem?**

Ans. Input is the data or information given to the algorithm at the start, like a list of numbers or a word to search for.

Q6 **What is process in a computational problem?**

Ans. Process means the steps taken to solve the problem, like sorting a list or finding a specific item.

taleemzone.com

Q7 What is output in a computational problem?

Ans. Output is the final result produced by the algorithm, like a sorted list or a "yes/no" answer.

Q8 What is a well-defined problem?

Ans. A well-defined problem has clear goals, known inputs, a defined process, and expected outputs, making it easier to solve with algorithms.

Q9 What is an ill-defined problem?

Ans. An ill-defined problem lacks clarity in one or more areas, such as unclear goals or unknown inputs, making it harder to model and solve.

Algorithms for Problem Solving

Q10 What is the Generate and Test algorithm?

Ans. It generates possible solutions and tests them one by one until the correct or best one is found.

Q11 When is the Generate and Test method useful?

Ans. It's useful when there's no clear shortcut to solving a problem, and you have to try multiple options.

Q12 When should you avoid Generate and Test?

Ans. Avoid it when the solution space is very large, as it can take too long and waste time checking many wrong options.

Problem Solvability and Complexity

Q13 What is a decision problem?

Ans. A decision problem is one where the answer is either "yes" or "no," like checking if a number is even.

Q14 What is a search problem?

Ans. A search problem involves finding a specific item or solution from a large amount of data, like looking up a name in a phone book.

Q15 What is an optimization problem?

Ans. An optimization problem aims to find the best possible solution, such as the fastest route or the cheapest way to build something.

Q16 What is a counting problem?

Ans. A counting problem asks how many ways something can be done, like how many different paths exist between two points.

Q17 What is a solvable problem?

Ans. A solvable problem is one where an algorithm exists that will always give an answer in finite time.

Q18 What is an unsolvable problem?

Ans. An unsolvable problem is one where no algorithm can solve it for all possible inputs, like the Halting Problem.

Q19 What is the Halting Problem?

Ans. The Halting Problem asks whether a program will stop or run forever. It's unsolvable because no general algorithm can predict this for all programs.

Q20 Why study unsolvable problems?

Ans. Studying unsolvable problems helps us understand the limits of computing and avoid wasting time trying to solve what cannot be solved.

Q21 What is a tractable problem?

Ans. A tractable problem can be solved efficiently, usually in polynomial time, meaning it stays manageable even with large inputs.

Q22 What is an intractable problem?

Ans. An intractable problem takes too long to solve as the input size grows, often requiring exponential time.

Q23 What does the complexity class P mean?

Ans. P includes problems that can be solved quickly (in polynomial time), like sorting or searching.

Q24 What does complexity class NP mean?

Ans. NP includes problems where solutions can be checked quickly, but finding them may take a long time, like solving a Sudoku puzzle.

Q25 What is the P vs NP question?

Ans. It asks whether all problems whose solutions can be checked quickly can also be solved quickly, one of the biggest open questions in computer science.

Algorithm Analysis

Q26 Why is algorithm analysis important?

Ans. It helps us understand how efficient an algorithm is in terms of time and memory, so we can choose the best one for the job.

Q27 What is time complexity?

Ans. Time complexity measures how long an algorithm takes to run based on the size of its input.

Q28 What is space complexity?

Ans. Space complexity measures how much memory an algorithm uses as the input size increases.

Q29 What is Big O notation used for?

Ans. Big O notation describes how fast or slow an algorithm gets as the input becomes larger, helping compare different algorithms.

© taleemzone.com

Algorithm Design Techniques

Q30 What is Divide and Conquer strategy?

Ans. Divide and Conquer breaks a big problem into smaller ones, solves them separately, and combines their results, used in Merge Sort and Quick Sort.

Q31 What is a Greedy Algorithm?

Ans. A Greedy Algorithm makes the best choice at each step hoping it leads to the best overall solution, like in coin change.

Q32 When might a Greedy Algorithm fail?

Ans. It can fail when choosing the best local option doesn't lead to the best global solution, like in some path finding problems.

Q33 What is Dynamic Programming?

Ans. Dynamic Programming solves complex problems by breaking them into overlapping sub-problems and storing results to avoid recomputing.

Q34 What kind of problems suit Dynamic Programming?

Ans. Problems with overlapping sub-problems and optimal substructure, such as Fibonacci numbers or the Knapsack Problem.

Q35 What is Backtracking?

Ans. Backtracking builds solutions step by step and goes back if a path doesn't work, often used in puzzles and combinatorial problems.

Commonly Used Algorithms

Q36 What is Bubble Sort good for?

Ans. Bubble Sort is simple and easy to learn, but it's slow for large datasets due to its $O(n^2)$ time complexity.

Q37 Why is Binary Search faster than Linear Search?

Ans. Binary Search cuts the search area in half each time, making it much faster, especially for large sorted lists.

Q38 Why must data be sorted before Binary Search?

Ans. Because Binary Search works by dividing the list in half, which only works if the data is in order.

Q39 What are Sorting Algorithms used for?

Ans. Sorting Algorithms arrange data in a specific order, making it easier and faster to search, analyze, and process.

Q40 What are Searching Algorithms used for?

Ans. Searching Algorithms help find specific items in a dataset quickly and efficiently, like finding a name in a phone book.

Q41 What is Linear Search?

Ans. Linear Search checks each item in a list one by one until it finds the target, it works on both sorted and unsorted data.

Q42 What is Binary Search?

Ans. Binary Search finds an item in a sorted list by repeatedly dividing the search interval in half, it's fast with O(log n) time.

Q43 What is Breadth-First Search (BFS)?

Ans. BFS explores a graph level by level using a queue, it's good for finding the shortest path in unweighted graphs.

Q44 What is Depth-First Search (DFS)?

Ans. DFS explores as far down a branch as possible before backtracking, it uses less memory than BFS and is good for exploring all paths.

Q45 What is the difference between BFS and DFS?

Ans. BFS is better for shortest paths; DFS is better for deep exploration. BFS uses a queue; DFS uses a stack.

Q46 What is the purpose of Graph Algorithms?

Ans. Graph algorithms help analyze relationships between nodes, like social networks, maps, or web links.

Q47 What is Selection Sort?

Ans. Selection Sort finds the smallest element and moves it to the front, it has $O(n^2)$ time and is easy to implement.

Q48 How does Bubble Sort work?

Ans. Bubble Sort compares adjacent elements and swaps them if they're in the wrong order, it repeats until the list is sorted.

Q49 Why is Bubble Sort inefficient for large data?

Ans. Because it has $O(n^2)$ time complexity, which means it gets very slow as the data size increases.

Q50 What is the importance of understanding algorithms?

Ans. Understanding algorithms helps you write better code, solve problems efficiently, and make smart decisions about which tools to use.

End of Chapter 3 Short Questions (Additional) – 50 Questions Covered