SSLC, HSE, DIPLOMA, B.E/B.TECH, M.E/M.TECH, MBA, MCA

Notes *Available @*
Syllabus
Question Papers www.AllAbtEngg.com
Results and Many more…

CS8451 DESIGN AND ANALYSIS OF ALGORITHM

2Mark Questions with Answers

UNIT-V

COPING WITH THE LIMITATIONS OF ALGORITHM POWER

1. What are the limitations of algorithm power?

There are many algorithms for solving a variety of different problems. They are very powerful instruments, especially when they are executed by modern computers.
The power of algorithms is because of the following reasons:

- There are some problems cannot be solved by any algorithm.
- There are some problems can be solved algorithmically but not in polynomial time.
- There are some problems can be solved in polynomial time by some algorithms, but there are usually lower bounds on their efficiency.

Algorithms limits are identified by the following:

- Lower-Bound Arguments
- Decision Trees
- P, NP and NP-Complete Problems

2. What are lower-bound arguments?

We can look at the efficiency of an algorithm two ways. We can establish its asymptotic efficiency class (say, for the worst case) and see where this class stands with respect to the hierarchy of efficiency classes.

Lower bounds means estimating the minimum amount of work needed to solve the problem. We present several methods for establishing lower bounds and illustrate them with specific examples.

1. Trivial Lower Bounds
2. Information-Theoretic Arguments
3. Adversary Arguments
4. Problem Reduction

In analyzing the efficiency of specific algorithms in the preceding, we should distinguish between a lower-bound class and a minimum number of times a particular operation needs to be executed.

3. Define Trivial Lower Bounds.

The simplest method of obtaining a lower-bound class is based on counting the number of items in the problem's input that must be processed and the number of output items that need to be produced. Since any algorithm must at least "read" all the items it needs to process and "write" all its outputs, such a count yields a trivial lower bound.

Available in /AllAbtEngg Android App too, Check *www.SmartPoet.Net* & *www.PhotoShip.Net*

SSLC, HSE, DIPLOMA, B.E/B.TECH, M.E/M.TECH, MBA, MCA
Notes *Available @*
Syllabus www.AllAbtEngg.com
Question Papers
Results and Many more…

4. Define Information-Theoretic Arguments.

The information-theoretical approach seeks to establish a lower bound based on the amount of information it has to produce by algorithm.

5. Define Adversary Arguments.

Adversary Argument is a method of proving by playing a role of adversary in which algorithm has to work more for adjusting input consistently.

Consider the Game of guessing number between positive integer 1 and n by asking a person (Adversary) with yes/no type answers for questions. After each question at least onehalf of the numbers reduced. If an algorithm stops before the size of the set is reduced to 1, the adversary can exhibit a number.

Any algorithm needs [log2 n] iterations to shrink an n-element set to a one-element set by halving and rounding up the size of the remaining set. Hence, at least [log2 n] questions need to be asked by any algorithm in the worst case. This example illustrates the adversary method for establishing lower bounds.

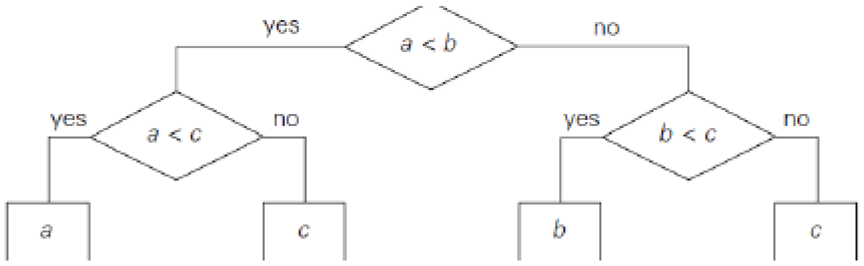
6. Define Problem Reduction.

Problem reduction is a method in which a difficult unsolvable problem P is reduced to another solvable problem B which can be solved by a known algorithm.

A similar reduction idea can be used for finding a lower bound. To show that problem P is at least as hard as another problem Q with a known lower bound, we need to reduce Q to P (not P to Q!). In other words, we should show that an arbitrary instance of problem Q can be transformed to an instance of problem P, so any algorithm solving P would solve Q as well. Then a lower bound for Q will be a lower bound for P.

7. Define decision trees.

Important algorithms like sorting and searching are based on comparing items of their inputs. The study of the performance of such algorithm is called a decision tree. As an example, Figure presents a decision tree of an algorithm for finding a minimum of three numbers. Each internal node of a binary decision tree represents a key comparison indicated in the node.



Decision tree for finding a minimum of three numbers.

8. Define tractable and intractable.

Problems that can be solved in polynomial time are called **tractable**, and problems that cannot be solved in polynomial time are called **intractable**.

Available in /AllAbtEngg Android App too, Check *www.SmartPoet.Net* & *www.PhotoShip.Net*

SSLC, HSE, DIPLOMA, B.E/B.TECH, M.E/M.TECH, MBA, MCA

Notes *Available @*
Syllabus www.AllAbtEngg.com
Question Papers
Results and Many more…

9. Define Hamiltonian circuit problem.

Determine whether a given graph has a Hamiltonian circuit—a path that starts and ends at the same vertex and passes through all the other vertices exactly once.

10. Define Traveling salesman problem.

Find the shortest tour through n cities with known positive integer distances between them (find the shortest Hamiltonian circuit in a complete graph with positive integer weights).

Applications

- Vehicle routing.
- Discrete optimization
- Computer network problem
- Airport tour.
- Sonnet ring
- Power cable

11. Define Knapsack problem.

Find the most valuable subset of n items of given positive integer weights and values that fit into a knapsack of a given positive integer capacity.

12. Define Partition problem.

Given n positive integers, determine whether it is possible to partition them into two disjoint subsets with the same sum.

13. Define Bin-packing problem.

Given n items whose sizes are positive rational numbers not larger than 1, put them into the smallest number of bins of size 1.

14. Define Graph-coloring problem.

For a given graph, find its chromatic number, which is the smallest number of colors that need to be assigned to the graph's vertices so that no two adjacent vertices are assigned the same color. Every Planner graph is 4 colorable.

15. Define Integer linear programming problem.

Find the maximum (or minimum) value of a linear function of several integer-valued variables subject to a finite set of constraints in the form of linear equalities and inequalities.

16. Define deterministic and nondeterministic algorithm.

A nondeterministic algorithm is a two-stage procedure that takes as its input an instance I of a decision problem and does the following.

1. **Nondeterministic ("guessing") stage:** An arbitrary string S is generated that can be thought of as a candidate solution to the given instance.
2. **Deterministic ("verification") stage:** A deterministic algorithm takes both I and S as its input and outputs yes if S represents a solution to instance I. (If S is not a solution to instance I , the algorithm either returns no or is allowed not to halt at all.)

Available in /AllAbtEngg Android App too, Check *www.SmartPoet.Net* & *www.PhotoShip.Net*

SSLC, HSE, DIPLOMA, B.E/B.TECH, M.E/M.TECH, MBA, MCA
Notes *Available @*
Syllabus www.AllAbtEngg.com
Question Papers
Results and Many more…

Finally, a nondeterministic algorithm is said to be **nondeterministic polynomial** if the time efficiency of its verification stage is polynomial.

17. Define Class P.

Class P is a class of decision problems that can be solved in polynomial time by deterministic algorithms. This class of problems is called polynomial class.

Examples:

- Searching
- Element uniqueness
- Graph connectivity
- Graph acyclicity
- Primality testing

18. Define Class NP.

Class NP is the class of decision problems that can be solved by nondeterministic polynomial algorithms. This class of problems is called nondeterministic polynomial.

Examples: Integer factorization problem, graph isomorphism problem,

All NP-complete problem (travelling salesman problem, Boolean satisfiability problem).

19. Define Class NP-Hard. / List out the properties of NP-Hard Problems.

A problem is NP-hard if an algorithm for solving it can be translated into one for solving any NP-problem (nondeterministic polynomial time) problem. Therefore NP-hard means "at least as hard as any NP-problem," although it might, in fact, be harder.

There are no polynomial-time algorithms for NP-hard problems.

Traveling salesman and knapsack problems are NP-hard problems.

20. Define NP-complete.

A decision problem D is said to be NP-complete if it is hard as any problem in NP.

1. It belongs to class NP
2. Every problem in NP is polynomially reducible to D

Examples:

Capacitated minimum spanning tree, Route inspection problem (also called Chinese postman problem), Clique problem, Maximum independent set, Minimum spanning tree, Complete coloring, Bandwidth problem, Clique cover problem, Graph homomorphism problem, Graph coloring, Graph partition into subgraphs of specific types (triangles, isomorphic subgraphs, Hamiltonian subgraphs, forests, perfect matchings) are known NP-complete.

21. Compare class P and NP problems.

Class P	**Class NP**
Class P is a class of decision problems that can be solved in polynomial time by deterministic algorithms.	Class NP is the class of decision problems that can be solved by nondeterministic polynomial algorithms.

Available in /AllAbtEngg Android App too, Check *www.SmartPoet.Net & www.PhotoShip.Net*

SSLC, HSE, DIPLOMA, B.E/B.TECH, M.E/M.TECH, MBA, MCA
Notes *Available @*
Syllabus www.AllAbtEngg.com
Question Papers
Results and Many more…

This class of problems is called polynomial class.	This class of problems is called nondeterministic polynomial class.
Subset of Class NP	Super set of Class P
Examples: • Searching • Element uniqueness • Graph connectivity • Graph acyclicity • Primality testing	**Examples:** • Travelling salesman problem • Boolean satisfiability problem • Knapsack problem

22. Compare class NP hard and NP complete problems.

Class NP Hard	Class NP Complete
Class of problems hard to solve	A decision problem *D* is said to be NP-complete if it is hard as any problem in NP. **1.** It belongs to class *NP* **2.** Every problem in *NP* is polynomially reducible to *D*
We can not solve all Class NP Complete problems in polynomial time	We can solve all Class NP Complete problems in polynomial time
Super set of Class NP Complete	Subset of Class NP hard
Examples: • Travelling Sales Person • Halting problem • SAT problem	**Examples:** • Travelling Sales Person • 3- SAT problem

23. Compare class Deterministic and non deterministic algorithms.

A deterministic algorithm is an algorithm which, given a particular input, will always produce the same output, with the underlying machine always passing through the same sequence of states.

A nondeterministic algorithm is a two-stage procedure that takes as its input an instance I of a decision problem and does the following.

1. **Nondeterministic (“guessing”) stage:** An arbitrary string S is generated that can be thought of as a candidate solution to the given instance.
2. **Deterministic (“verification”) stage:** A deterministic algorithm takes both I and S as its input and outputs yes if S represents a solution to instance I. (If S is not a solution to instance I , the algorithm either returns no or is allowed not to halt at all.)

Finally, a nondeterministic algorithm is said to be nondeterministic polynomial if the time efficiency of its verification stage is polynomial.

Available in /AllAbtEngg Android App too, Check *www.SmartPoet.Net* & *www.PhotoShip.Net*

SSLC, HSE, DIPLOMA, B.E/B.TECH, M.E/M.TECH, MBA, MCA

Notes *Available @*
Syllabus www.AllAbtEngg.com
Question Papers
Results and Many more…

24. Define Cook's theorem.

In computational complexity theory, the Cook–Levin theorem, also known as Cook's theorem, states that the Boolean satisfiability problem is NP-complete. That is, any problem in NP can be reduced in polynomial time by a deterministic algorithm (Turing machine) to the problem of determining whether a Boolean formula is satisfiable. E.g SAT-3 problem.

25. Define backtracking.

Backtracking is a general algorithmic technique that considers searching every possible combination in order to solve an optimization problem. Backtracking is also known as depthfirst search or branch and bound. By inserting more knowledge of the problem, the search tree can be pruned to avoid considering cases that don't look promising. While backtracking is useful for hard problems to which we do not know more efficient solutions.

Backtracking is a general algorithm for finding all (or some) solutions to some computational problems, notably constraint satisfaction problems, that incrementally builds candidates to the solutions, and abandons each partial candidate c ("backtracks") as soon as it determines that c cannot possibly be completed to a valid solution.

Backtracking examples.

- *n*-Queens Problem
- Hamiltonian Circuit Problem
- Subset-Sum Problem

26. Define Branch and bound. ®

Branch and bound (BB or B&B) is an algorithm design paradigm for discrete and combinatorial optimization problems, as well as general real valued problems. A branch-and bound algorithm consists of a systematic enumeration of candidate solutions by means of state space search: the set of candidate solutions is thought of as forming a rooted tree with the full set at the root. The algorithm explores branches of this tree, which represent subsets of the solution set. Before enumerating the candidate solutions of a branch, the branch is checked against upper and lower estimated bounds on the optimal solution, and is discarded if it cannot produce a better solution than the best one found so far by the algorithm.

The algorithm depends on the efficient estimation of the lower and upper bounds of a region/branch of the search space and approaches exhaustive enumeration as the size (ndimensional volume) of the region tends to zero.

Branch-and-Bound Examples:

- Assignment Problem
- Knapsack Problem
- Traveling Salesman Problem
-

27. List out Exact Solution technique.

Exhaustive search (brute force)-

• useful only for small instances

Dynamic programming

• applicable to some problems (e.g., the knapsack problem)

Backtracking

• eliminates some unnecessary cases from consideration

Available in /AllAbtEngg Android App too, Check *www.SmartPoet.Net* & *www.PhotoShip.Net*

SSLC, HSE, DIPLOMA, B.E/B.TECH, M.E/M.TECH, MBA, MCA

Notes *Available @*
Syllabus www.AllAbtEngg.com
Question Papers
Results and Many more…

• yields solutions in reasonable time for many instances but worst case is still exponential

Branch-and-bound

• further refines the backtracking idea for optimization problems

28. List out coping techniques with the Limitations of Algorithm Power.

Backtracking

- *n*-Queens Problem
- Hamiltonian Circuit Problem
- Subset-Sum Problem

Branch-and-Bound

- Assignment Problem
- Knapsack Problem
- Traveling Salesman Problem

Approximation Algorithms for *NP*-Hard Problems

- Approximation Algorithms for the Traveling Salesman Problem
- Approximation Algorithms for the Knapsack Problem

Algorithms for Solving Nonlinear Equations

- Bisection Method
- False Position Method
- Newton's Method

29. Define Backtracking

- Backtracking is a more intelligent variation approach.
- The principal idea is to construct solutions one component at a time and evaluate such partially constructed candidates as follows.
- If a partially constructed solution can be developed further without violating the problem's constraints, it is done by taking the first remaining legitimate option for the next component.
- If there is no legitimate option for the next component, no alternatives for any remaining component need to be considered. In this case, the algorithm backtracks to replace the last component of the partially constructed solution with its next option.
- It is convenient to implement this kind of processing by constructing a tree of choices being made, called the state-space tree.
- Depth first node generation with bounding function is called backtracking. The backtracking algorithm has its virtue the ability to yield the answer with far fewer than m trials
- Backtracking techniques are applied to solve the following problems
    - Hamiltonian Circuit Problem
    - Subset-Sum Problem
    - n-Queens Problem

Available in /AllAbtEngg Android App too, Check www.SmartPoet.Net & www.PhotoShip.Net

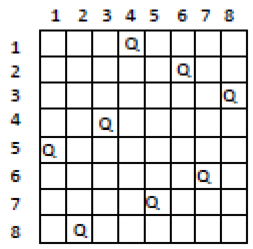
SSLC, HSE, DIPLOMA, B.E/B.TECH, M.E/M.TECH, MBA, MCA
Notes *Available @*
Syllabus www.AllAbtEngg.com
Question Papers
Results and Many more…

30. Define N-Queens Problem.

The problem is to place n queens on an n × n chessboard so that no two queens attack each other by being in the same row or in the same column or on the same diagonal.

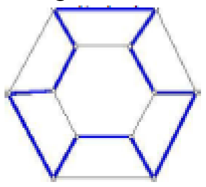
31. Find a solution 8 queens problem.

There is solution to place 8 queens in 8 × 8 chessboard.



32. Define Hamiltonian circuit problem.

A Hamiltonian circuit (also called a Hamiltonian cycle, Hamilton cycle, or Hamilton circuit) is a graph cycle (i.e., closed loop) through a graph that visits each node exactly once. A graph possessing a Hamiltonian cycle is said to be a Hamiltonian graph.



33. Given an application for knapsack problem?

The knapsack problem is problem combinatorial optimization. It derives its name from the maximum problem of choosing possible essential that can fit too bag to be carried on trip. A similar problem very often appears business, combinatory, complexity theory, cryptography and applied mathematics.

34. Define subset sum problem.

Subset sum problem is problem, which is used to find a subset of a given set S={S1,S2,S3,…….Sn} of positive integers whose sum is equal to given positive integer d. The subset-sum problem finds a subset of a given set *A* = {*a*1, . . . , *an*} of *n* positive integers whose sum is equal to a given positive integer *d.* For example, for *A* = {1, 2, 5, 6, 8} and *d* = 9, there are two solutions: {1, 2, 6} and {1, 8}. Of course, some instances of this problem may have no solutions.

35. What is heuristic?

A heuristic is common sense rule drawn from experience rather than from mathematically proved assertion. For example, going to the nearest unvisited city in the travelling salesman problem is good example for heuristic.

36. State the concept of branch and bound method?

The branch and bound method refers to all state space search methods in which all children of the E-Node are generated before any other live node can become the E-node.

Available in /AllAbtEngg Android App too, *Check www.SmartPoet.Net & www.PhotoShip.Net*

SSLC, HSE, DIPLOMA, B.E/B.TECH, M.E/M.TECH, MBA, MCA

Notes
Syllabus
Question Papers
Results and Many more…

Available @
www.AllAbtEngg.com

Some problems can be solved by Branch-and-Bound are:

1. Assignment Problem
2. Knapsack Problem
3. Traveling Salesman Problem

37. Give the upper bound and lower bound of matrix multiplication algorithm?

Upper bound: The given algorithm does n*n*n multiplication hence at most n*n*n multiplication are necessary. Lower bound: It has been proved in the literature that at least n*n multiplication are necessary.

38. What is state space tree?

Backtracking and branch bound are based on the construction of a state space tree, whose nodes reflect specific choices made for a solution's component. Its root represents an initial state before the search for a solution begins. The nodes of the first level the tree represent the made for the first component of solution, the nodes of the second level represent the Choices for the second components & so on

39. What is promising and non promising node?

A node state space tree is said to be promising, if it corresponds to a partially constructed solution that may still lead to complete solution. Otherwise, node is called non-promising.

40. What are the additional items are required for branch and bound compare to backtracking technique?

Compared to backtracking, branch and bound requires 2 additional items.

1) A way to prove, for every node of node of state space tree, a bound on the best value of the objective function on any solution that can be obtain d by adding further components to the partial solution represented by the node.
2) The value of the best solution seen so far.

41. Differentiate backtracking and branch bound techniques.

• Backtracking is applicable only to non-optimization problems.
• Backtracking generates state space tree depth first manner.
• Branch and bound is applicable only to optimization problem.
• Branch and bound generated a node of state space tree using best first rule.

42. What is called all pair shortest path problem?

Given a weighted connected graph, the all pairs shortest paths problem is to find the distances from each vertex to all other vertices.

43. When do you say a tree as minimum spanning tree?

A spanning tree is said to be minimum spanning tree when the weight of the spanning tree is smallest, where the weight of a tree is defined as the sum of the weight of all its edges.

Available in /AllAbtEngg Android App too, Check *www.SmartPoet.Net* & *www.PhotoShip.Net*

SSLC, HSE, DIPLOMA, B.E/B.TECH, M.E/M.TECH, MBA, MCA

Notes *Available @*
Syllabus www.AllAbtEngg.com
Question Papers
Results and Many more…

44. How will you construct an optimal binary search tree?

A binary search tree is one of the most important data structures in computer sciences. Its principal applications are to implement a dictionary, a set of elements with the operations of searching, insertion and deletion. If probabilities of searching for elements of a set are known as optimal binary search tree, for which the average number of comparisons in a search is the smallest possible.

45. What is the runtime of shortest path algorithm?

The runtime of shortest path algorithm is $\Theta((n+|E|) \log n)$

46. Define Nearest-Neighbor Algorithm

The following well-known greedy algorithm is based on the nearest-neighbor heuristic: always go next to the nearest unvisited city.

Step 1 Choose an arbitrary city as the start.

Step 2 Repeat the following operation until all the cities have been visited: go to the unvisited city nearest the one visited last (ties can be broken arbitrarily).

Step 3 Return to the starting city.

47. Define Knapsack Problem (Approximation Algorithm)

The knapsack problem is one well-known *NP*-hard problem. Given *n* items of known weights $w_1, \ldots, w_n$ and values $v_1, \ldots, v_n$ and a knapsack of weight capacity *W,* find the most valuable subset of the items that fits into the knapsack.

48. Define Greedy algorithm for the discrete knapsack problem

Step 1 Compute the value-to-weight ratios $r_i = v_i/w_i$, $i = 1, \ldots, n$, for the items given.

Step 2 Sort the items in nonincreasing order of the ratios computed in Step 1.(Ties can be broken arbitrarily.)

Step 3 Repeat the following operation until no item is left in the sorted list: if the current item on the list fits into the knapsack, place it in the knapsack and proceed to the next item; otherwise, just proceed to the next item.

49. Define Greedy Algorithm for the Continuous Knapsack Problem

Step 1 Compute the value-to-weight ratios *vi/wi*, i = 1, . . . , n, for the items given.

Step 2 Sort the items in nonincreasing order of the ratios computed in Step 1. (Ties can be broken arbitrarily.)

Step 3 Repeat the following operation until the knapsack is filled to its full capacity or no item is left in the sorted list: if the current item on the list fits into the knapsack in its entirety, take it and proceed to the next item; otherwise, take its largest fraction to fill the knapsack to its full capacity and stop.

50. What do you mean by accuracy ratio and performance ratio of approximation algorithm?

We can quantify the accuracy of an approximate solution S_a to a problem of minimizing some function f by the size of the relative error (re) of this approximation,

$$\mathrm{Re}(S_a) = \frac{f(Sa) - f(S^*)}{f(S')}$$

Available in /AllAbtEngg Android App too, Check *www.SmartPoet.Net & www.PhotoShip.Net*

SSLC, HSE, DIPLOMA, B.E/B.TECH, M.E/M.TECH, MBA, MCA

Notes *Available @*
Syllabus www.AllAbtEngg.com
Question Papers
Results and Many more…

where *s** is an exact solution to the problem. Alternatively, *re*(s_a) = *f* (s_a)/*f* (*s**) − 1, we can simply use the accuracy ratio

$$r(S_a) = \frac{f(Sa)}{f(S^*)}$$

as a measure of accuracy of sa. Note that for the sake of scale uniformity, the accuracy ratio of approximate solutions to maximization problems is usually computed as

$$r(s_a) = \frac{f(S^*)}{f(Sa)}$$

to make this ratio greater than or equal to 1, as it is for minimization problems.

Available in /AllAbtEngg Android App too, Check *www.SmartPoet.Net* & *www.PhotoShip.Net*