CS6402-DESIGN AND ANALYSIS OF ALGORITHM

TWO MARK QUESTION WITH ANSWERS

Unit-I

Introduction

Why is the need of studying algorithms?
From a practical standpoint, a standard set of algorithms from different areas of computing must be known, in addition to be able to design them and analyze their efficiencies. From a theoretical standpoint the study of algorithms is the cornerstone of computer science.

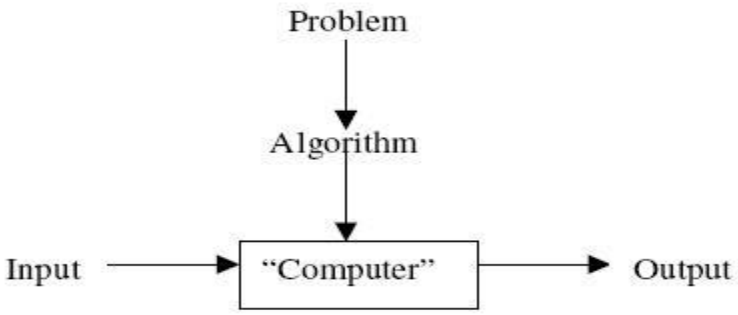
2. What is algorithmic?

The study of algorithms is called algorithmic. It is more than a branch of computer science. It is the core of computer science and is said to be relevant to most of science, business and technology.

3. What is an algorithm?

An algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in finite amount of time. An algorithm is step by step procedure to solve a problem.

4. Give the diagram representation of Notion of algorithm.



5. What is the formula used in Euclid's algorithm for finding the greatest common divisor of two numbers?

Euclid's algorithm is based on repeatedly applying the equality Gcd(m,n)=gcd(n,m mod n) until m mod n is equal to 0, since gcd(m,0)=m.

6. What are the three different algorithms used to find the gcd of two numbers?

The three algorithms used to find the gcd of two numbers are

- Euclid‘s algorithm
- Consecutive integer checking algorithm
- Middle school procedure

7. What are the fundamental steps involved in algorithmic problem solving?

The fundamental steps are

Understanding the problem

Ascertain the capabilities of computational device

- choose between exact and approximate problem solving
- Decide on appropriate data structures
- Algorithm design techniques
- Methods for specifying the algorithm
- Proving an algorithms correctness
- Analyzing an algorithm
- Coding an algorithm

8. What is an algorithm design technique?

An algorithm design technique is a general approach to solving problems algorithmically that is applicable to a variety of problems from different areas of computing.

9. What is pseudocode?

A pseudocode is a mixture of a natural language and programming language constructs to specify an algorithm. A pseudocode is more precisethan a natural language and its usage often yields more concise algorithm descriptions.

10. What are the types of algorithm efficiencies?

The two types of algorithm efficiencies are

- *Time efficiency:* indicates how fast the algorithm runs
- *Space efficiency:* indicates how much extra memory the algorithm needs

11. Mention some of the important problem types?

Some of the important problem types are as follows

- Sorting
- Searching
- String processing
- Graph problems
- Combinatorial problems
- Geometric problems
- Numerical problems

12. What are the classical geometric problems?

The two classic geometric problems are

- *The closest pair problem:* given n points in a plane find the closest pair among them
- *The convex hull problem:* find the smallest convex polygon that would include all the points of a given set.

13. What are the steps involved in the analysis framework?

The various steps are as follows

- Measuring the input‘s size
- Units for measuring running time
- Orders of growth
- Worst case, best case and average case efficiencies

14. What is the basic operation of an algorithm and how is it identified?

- The most important operation of the algorithm is called the basic operation of the algorithm, the operation that contributes the most to the total running time.
- It can be identified easily because it is usually the most time consuming operation in the algorithms innermost loop.

15. What is the running time of a program implementing the algorithm?

The running time T(n) is given by the following formula

$$T(n) \approx copC(n)$$

cop is the time of execution of an algorithm‘s basic operation on a particular computer and C(n) is the number of times this operation needs to be executed for the particular algorithm.

What are exponential growth functions?

The functions 2n and n! are exponential growth functions, because these two functions grow so fast that their values become astronomically large even for rather smaller values of n.

17. What is worst-case efficiency?

The worst-case efficiency of an algorithm is its efficiency for the worst-case input of size n, which is an input or inputs of size n for which the algorithm runs the longest among all possible inputs of that size.

18. What is best-case efficiency?

The best-case efficiency of an algorithm is its efficiency for the best-case input of size n, which is an input or inputs for which the algorithm runs the fastest among all possible inputs of that size.

19. What is average case efficiency?

The average case efficiency of an algorithm is its efficiency for an average case input of size n. It provides information about an algorithm behavior on a ―typical‖ or ―random‖ input.

20. What is amortized efficiency?

In some situations a single operation can be expensive, but the total time for the entire sequence of n such operations is always significantly better that the worst case efficiency of that single operation multiplied by n. this is called amortized efficiency.

21. Define O-notation?

A function t(n) is said to be in O(g(n)), denoted by t(n) ε O(g(n)), if t(n) is bounded above by some constant multiple of g(n) for all large n, i.e., if there exists some positive constant c and some non-negative integer n_0 such that

$$T(n) <= cg(n) \text{ for all } n >= n_0$$

22. Define Ω-notation?

A function t(n) is said to be in Ω (g(n)), denoted by t(n) ε Ω (g(n)), if t(n) is bounded below by some constant multiple of g(n) for all large n, i.e., if there exists some positive constant c and some non-negative integer n_0 such that

$$T(n) >= cg(n) \text{ for all } n >= n_0$$

23. Define θ-notation?

A function t(n) is said to be in θ (g(n)), denoted by t(n) ε θ (g(n)), if t(n) is bounded both above & below by some constant multiple of g(n) for all large n, i.e., if there exists some positive constants c1 & c2 and some nonnegative integer n0 such that

$$c_2 g(n) <= t(n) <= c_1 g(n) \text{ for all } n >= n0$$

24. Mention the useful property, which can be applied to the asymptotic notations and its use?

If $t_1(n)$ ε $O(g_1(n))$ and $t_2(n)$ ε $O(g_2(n))$ then $t_1(n)+t_2(n)$ ε max $\{g_1(n),g_2(n)\}$ this property is also true for Ω and θ notations. This property will be useful in analyzing algorithms that comprise of two consecutive executable parts.

25. What are the basic asymptotic efficiency classes?

The various basic efficiency classes are

- Constant : 1
- Logarithmic : log n
- Linear : n
- N-log-n : nlog n
- Quadratic : n2
- Cubic : n3
- Exponential : 2n
- Factorial : n!

26.What is algorithm visualization?

Algorithm visualization is a way to study algorithms. It is defined as the use of images to convey some useful information about algorithms. That information can be a visual illustration of algorithm‘s operation, of its performance on different kinds of inputs, or of its execution speed versus that of other
algorithms for the same problem.

27. What are the two variations of algorithm visualization?

- The two principal variations of algorithm visualization‖ ._*Static algorithm visualization*: It shows the algorithm‘s progress
- through a series of still images ._*Dynamic algorithm visualization*: Algorithm animation shows a
- continuous movie like presentation of algorithms operations

28. What is order of growth?

Measuring the performance of an algorithm based on the input size n is called order of growth

UNIT-2

BRUTE FORCE AND DIVIDE-AND-CONQUER

1.What is brute force algorithm?

A straightforward approach, usually based directly on the problem‘s statement and definitions of the concepts involved.

2. List the strength and weakness of brute force algorithm. Strengths

a. wide applicability,
b. simplicity
c. yields reasonable algorithms for some important problems
(e.g., matrix multiplication, sorting, searching, string matching)

Weaknesses

a. rarely yields efficient algorithms
b. some brute-force algorithms are unacceptably slow not as constructive as some other design techniques

3. What is exhaustive search?

A brute force solution to a problem involving search for an element with a special property, usually among combinatorial objects such as permutations, combinations, or subsets of a set.

4. Give the general plan of exhaustive search. Method:

- generate a list of all potential solutions to the problem in a systematic manner
- evaluate potential solutions one by one, disqualifying infeasible ones and, for an optimization problem, keeping track of the best one found so far
- when search ends, announce the solution(s) found

5. Give the general plan for divide-and-conquer algorithms.

The general plan is as follows

- A problems instance is divided into several smaller instances of the same problem, ideally about the same size
- The smaller instances are solved, typically recursively
- If necessary the solutions obtained are combined to get the solution of the original problem

Given a function to compute on ‗n‘ inputs the divide-and-comquer strategy suggests splitting the inputs in to‘k‘ distinct susbsets, 1<k <n, yielding ‗k‘ subproblems. The subproblems must be solved, and then a method must be found to combine subsolutions into a solution of the whole. If the subproblems are still relatively large, then the divide-and conquer strategy can possibly be reapplied.

6. List the advantages of Divide and Conquer Algorithm

Solving difficult problems, Algorithm efficiency, Parallelism, Memory access, Round off control.

7. Define of feasibility

A feasible set (of candidates) is promising if it can be extended to produce not merely a solution, but an optimal solution to the problem.

8. Define Hamiltonian circuit.

A Hamiltonian circuit is defined as a cycle that passes through all the vertices of the graph exactly

9. State the Master theorem and its use.

If f(n) □ θ(n^d) where d ³ 0 in recurrence equation T(n) = aT(n/b)+f(n), then

θ (n^d) if a<b^d T(n) □ θ

(n^dlog n) if a=b^d

θ (nlog b^a) if a>b^d

The efficiency analysis of many divide-and-conquer algorithms is greatly simplified by the use of Master theorem.

10. What is the general divide-and-conquer recurrence relation?

An instance of size _n' can be divided into several instances of size n/b, with _a' of them needing to be solved. Assuming that size _n' is a power of _b', to simplify the analysis, the following recurrence for the running time is obtained:

$$T(n) = aT(n/b)+f(n)$$

Where f(n) is a function that accounts for the time spent on dividing the problem into smaller ones and on combining their solutions.

11. Define mergesort.

Mergesort sorts a given array A[0..n-1] by dividing it into two halves a[0..(n/2)-1] and A[n/2..n-1] sorting each of them recursively and then merging the two smaller sorted arrays into a single sorted one.

12. List the Steps in Merge Sort

1. Divide Step: If given array A has zero or one element, return S; it is already sorted. Otherwise, divide A into two arrays, A1 and A2, each containing about half of the elements of A.

2. Recursion Step: Recursively sort array A1 and A2.

3. Conquer Step: Combine the elements back in A by merging the sorted arrays A1 and A2 into a sorted sequence

13. List out Disadvantages of Divide and Conquer Algorithm

- Conceptual difficulty
- Recursion overhead
- Repeated subproblems

14. Define Quick Sort

Quick sort is an algorithm of choice in many situations because it is not difficult to implement, it is a good \"general purpose\" sort and it consumes relatively fewer resources during execution.

15. List out the Advantages in Quick Sort

- It is in-place since it uses only a small auxiliary stack.
- It requires only n log(n) time to sort n items.
- It has an extremely short inner loop
- This algorithm has been subjected to a thorough mathematical analysis, a very precise statement can be made about performance issues.

16. List out the Disadvantages in Quick Sort

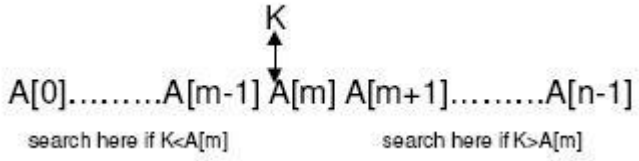
- It is recursive. Especially if recursion is not available, the implementation is extremely complicated.
- It requires quadratic (i.e., n2) time in the worst-case.
- It is fragile i.e., a simple mistake in the implementation can go unnoticed and cause it to perform badly.

17. What is the difference between quicksort and mergesort?

Both quicksort and mergesort use the divide-and-conquer technique in which the given array is partitioned into subarrays and solved. The difference lies in the technique that the arrays are partitioned. For mergesort the arrays are partitioned according to their position and in quicksort they are partitioned according to the element values.

.18. **What is binary search?**

Binary search is a remarkably efficient algorithm for searching in a sorted array. It works by comparing a search key K with the arrays middle element A[m]. If they match the algorithm stops; otherwise the same operation is repeated recursively for the first half of the array if K < A[m] and the second half if K > A[m].



19. List out the 4 steps in Strassen's Method?

1. Divide the input matrices A and B into n/2 * n/2 submatrices, as in equation (1).
2. Using Θ(n2) scalar additions and subtractions, compute 14 n/2 * n/2 matrices A1, B1, A2, B2, …, A7, B7.
3. Recursively compute the seven matrix products Pi =AiBi for i =1, 2, 7.
4. Compute the desired submatrices r, s, t, u of the result matrix C by adding and/or subtracting various combinations of the Pi matrices, using only Θ(n2) scalar additions and subtractions.

UNIT – 3

DYNAMIC PROGRAMMING AND GREEDY

1. **Define dynamic programming.**

Dynamic programming is an algorithm design method that can be used when a solution to the problem is viewed as the result of sequence of decisions.

Dynamic programming is a technique for solving problems with overlapping subproblems. These sub problems arise from a recurrence relating a solution to a given problem with solutions to its smaller sub problems only once and recording the results in a table from which the solution to the original problem is obtained. It was invented by a prominent U.S Mathematician, Richard Bellman in the 1950s.

2. **What are the features of dynamic programming?**
   - Optimal solutions to sub problems are retained so as to avoid recomputing their values.
   - Decision sequences containing subsequences that are sub optimal are not considered.
   - It definitely gives the optimal solution always.
3. **What are the drawbacks of dynamic programming?**
   - Time and space requirements are high, since storage is needed for all level.
   - Optimality should be checked at all levels.

4. **Write the general procedure of dynamic programming.**

The development of dynamic programming algorithm can be broken into a sequence of 4 steps.

- Characterize the structure of an optimal solution.
- Recursively define the value of the optimal solution.
- Compute the value of an optimal solution in the bottom-up fashion.
- Construct an optimal solution from the computed information.

5. **Define principle of optimality.**

It states that an optimal sequence of decisions has the property that whenever the initial stage or decisions must constitute an optimal sequence with regard to stage resulting from the first decision.

6. **Write the difference between the Greedy method and Dynamic programming.**

- Greedy method

1. Only one sequence of decision is generated.
2. It does not guarantee to give an optimal solution always.

- Dynamic programming

1. Many number of decisions are generated.
2. It definitely gives an optimal solution always.

7. **What is greedy technique?**

Greedy technique suggests a greedy grab of the best alternative available in the hope that a sequence of locally optimal choices will yield a globally optimal solution to the entire problem. The choice must be made as follows

- *Feasible : It has to satisfy the problem's constraints*
- *Locally optimal : It has to be the best local choice among all feasible choices available on that step.*
- *Irrevocable : Once made, it cannot be changed on a subsequent step of the algorithm*

8.Write any two characteristics of Greedy Algorithm?

- To solve a problem in an optimal way construct the solution from given set of candidates.As the algorithm proceeds, two other sets get accumulated among this one set contains the candidates that have been already considered and chosen while the other set contains the candidates that have been considered but rejected.

9. **What is the Greedy choice property?**

- The first component is greedy choice property (i.e.) a globally optimal solution can arrive at by making a locally optimal choice.
- The choice made by greedy algorithm depends on choices made so far but it cannot depend on any future choices or on solution to the sub problem.
- It progresses in top down fashion.

10. **What is greedy method?**

Greedy method is the most important design technique, which makes a choice that looks best at that moment. A given _n' inputs are required us to obtain a subset that satisfies some constraints that is the feasible solution. A greedy method suggests that one can device an algorithm that works in stages considering one input at a time.

11. What are the steps required to develop a greedy algorithm?

- Determine the optimal substructure of the problem.
- Develop a recursive solution.
- Prove that at any stage of recursion one of the optimal choices is greedy choice. Thus it is always safe to make greedy choice.
- Show that all but one of the sub problems induced by having made the greedy choice are empty.
- Develop a recursive algorithm and convert into iterative algorithm.

12. What is greedy technique?

- Greedy technique suggests a greedy grab of the best alternative available in the hope that a sequence of locally optimal choices will yield a globally optimal solution to the entire problem. The choice must be made as follows.
- Feasible: It has to satisfy the problem‘s constraints.
- Locally optimal: It has to be the best local choice among all feasible choices available on that step.
- Irrevocable : Once made, it cannot be changed on a subsequent step of the algorithm

13. What are the labels in Prim’s algorithm used for?

Prim‘s algorithm makes it necessary to provide each vertex not in the current tree with the information about the shortest edge connecting the vertex to a tree vertex. The information is provided by attaching two labels to a vertex.

- The name of the nearest tree vertex.
- The length of the corresponding edge

14. How are the vertices not in the tree split into?

The vertices that are not in the tree are split into two sets

- Fringe : It contains the vertices that are not in the tree but are adjacent to atleast one tree vertex.
- Unseen : All other vertices of the graph are called unseen because they are yet to be affected by the algorithm.

15. What are the operations to be done after identifying a vertex u* to be added to the tree?

After identifying a vertex u* to be added to the tree, the following two operations need to be performed

- Move u* from the set $V-V_T$ to the set of tree vertices V_T.
- For each remaining vertex u in $V-V_T$ that is connected to u* by a shorter edge than the u‘s current distance label, update its labels by u* and the weight of the edge between u* and u, respectively.

16.What is the use of Dijksra’s algorithm?

Dijkstra‘s algorithm is used to solve the single-source shortest-paths problem: for a given vertex called the source in a weighted connected graph, find the shortest path to all its other vertices. The single-source shortest-paths problem asks for a family of paths, each leading from the source to a different vertex in the graph, though some paths may have edges in common.

17. Define Spanning tree.

Spanning tree of a connected graph *G*: a connected acyclic subgraph of *G* that includes all of *G*‘s vertices

18. What is minimum spanning tree.

Minimum spanning tree of a weighted, connected graph *G*: a spanning tree of *G* of the minimum total weight

UNIT -4

ITERATIVE IMPROVEMENT

PART - A

1.Define linear programming.

Every LP problem can be represented in such form

maximize $3x + 5y$ subject to $x + y \leq 4$, $x \geq 0$, $y \geq 0$

maximize $3x + 5y + 0u + 0v$ subject to $x + y + u = 4$, $x + 3y \leq 6$, $x + 3y + v = 6$, $x \geq 0$, $y \geq 0$, $u \geq 0$, $v \geq 0$

Variables *u* and *v*, transforming inequality constraints into equality constrains, are called *slack variables*

2. What is basic solution?

A *basic solution* to a system of *m* linear equations in *n* unknowns ($n \geq m$) is obtained by setting $n - m$ variables to 0 and solving the resulting system to get the values of the other *m* variables. The variables set to 0 are called *nonbasic*; the variables obtained by solving the system are called *basic*.

A basic solution is called *feasible* if all its (basic) variables are nonnegative.

3. Define flow and flow conservation requirement.

A *flow* is an assignment of real numbers x_{ij} to edges (i,j) of a given network that satisfy the following:

- ***flow-conservation requirements*:** The total amount of material entering an intermediate vertex must be equal to the total amount of the material leaving the vertex
- ***capacity constraints***

  $0 \leq x_{ij} \leq u_{ij}$ for every edge $(i,j) \in E$

4. What is cut and min cut?

Let X be a set of vertices in a network that includes its source but does not include its sink, and let X, the complement of X, be the rest of the vertices including the sink. The *cut* induced by this partition of the vertices is the set of all the edges with a tail in X and a head in X.

Capacity of a cut is defined as the sum of capacities of the edges that compose the cut.

- We'll denote a cut and its capacity by C(X,X) and c(X,X)
- Note that if all the edges of a cut were deleted from the network, there would be no directed path from source to sink
- *Minimum cut* is a cut of the smallest capacity in a given network

5. State max – flow – min – cut theorem.

The value of maximum flow in a network is equal to the capacity of its minimum cut

6.**Define Bipartite Graphs.**

Bipartite graph: a graph whose vertices can be partitioned into two disjoint sets V and U, not necessarily of the same size, so that every edge connects a vertex in V to a vertex in U. A graph is bipartite if and only if it does not have a cycle of an odd length

7. What is augmentation and augmentation path?

An *augmenting path* for a matching M is a path from a free vertex in V to a free vertex in U whose edges alternate between edges not in M and edges in M

- The length of an augmenting path is always odd
- Adding to M the odd numbered path edges and deleting from it the even numbered path edges increases the matching size by 1 (*augmentation*)

One-edge path between two free vertices is special case of augmenting path.

UNIT V
COPING WITH THE LIMITATIONS OF A LGORITHM POWER

PART - A

1. What is meant by n-queen Problem?

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

2. Define Backtracking

Backtracking is used to solve problems with tree structures. Even problems seemingly remote to trees such as a walking a maze are actually trees when the decision \'back-left-straight-right\' is considered a node in a tree. The principle idea is to construct solutions one component at a time and evaluate such partially constructed candidates

3. What is the Aim of Backtracking?

Backtracking is the approach to find a path in a tree. There are several different aims to be achieved :

- just a path
- all paths
- the shortest path

4. Define the Implementation considerations of Backtracking?

The implementation bases on recursion. Each step has to be reversible; hence the state has to be saved somehow. There are two approaches to save the state:

- As full state on the stack
- As reversible action on the stack

5. List out the implementation procedure of Backtracking

As usual in a recursion, the recursive function has to contain all the knowledge. The standard implementaion is :

1. check if the goal is achieved REPEAT
2. check if the next step is possible at all
3. check if the next step leads to a known position - prevent circles
4. do this next step UNTIL (the goal is achieved) or (this position failed)

6. Define Approximation Algorithm

Approximation algorithms are often used to find approximate solutions to difficult problems of combinatorial optimization.

7. Define Promising Node?

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

8. Define Non-Promising Node?

A node in a state-space tree is said to be nonpromising if it backtracks to the node‘s parent to consider the nest possible solution for its last component.

9. Why the search path in a state-space tree of a branch and bound algorithm is terminated?

- The value of the node‘s bound is not better than the value of the best solution.
- The node represents no feasible solutions because the constraints of the problem are already violated
- The subset of feasible solutions represented by the node consists of a single point

10.Define Subset-Sum Problem?

This problem find a subset of a given set S={s1,s2,……,sn} of n positive integers whose sum is equal to a given positive integer d.

11. Define Traveling Salesman Problem?

Given a complete undirected graph G=(V, E) that has nonnegative integer cost c(u, v) associated with each edge (u, v) in E, the problem is to find a hamiltonian cycle (tour) of G with minimum cost.

12. Define Knapsack Problem

Given n items of known weight wi and values vi=1,2,..,n and a knapsack of capacity w, find the most valuable subset of the items that fit in the knapsack.

13. Define Branch and Bound?

A counter-part of the backtracking search algorithm which, in the absence of a cost criteria, the algorithm traverses a spanning tree of the solution space using the breadth-first approach. That is, a queue is used, and the nodes are processed in first-in-first-out order.

14. What is a state space tree?

The processing of backtracking is implemented by constructing a tree of choices being made. This is called the state-space tree. Its root represents a initial state before the search for a solution begins. The nodes of the first level in the tree represent the choices made for the first component of the solution, the nodes in the second level represent the choices for the second component and so on.

15. What is a promising node in the state-space tree?

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

16. What is a non-promising node in the state-space tree?

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

17. What do leaves in the state space tree represent?

Leaves in the state-space tree represent either non-promising dead ends or complete solutions found by the algorithm.

18. What is the manner in which the state-space tree for a backtracking algorithm is constructed?

In the majority of cases, a state-space tree for backtracking algorithm is constructed in the manner of depth-first search. If the current node is promising, its child is generated by adding the first remaining legitimate option for the next component of a solution, and the processing moves to this child.

If the current node turns out to be non-promising, the algorithm backtracks to the node‘s parent to consider the next possible solution to the problem, it either stops or backtracks to continue searching for other possible solutions.

19. Define the Hamiltonian circuit.

The Hamiltonian is defined as a cycle that passes through all the vertices of the graph exactly once. It is named after the Irish mathematician Sir William Rowan Hamilton (1805-1865).It is a sequence of n+1 adjacent vertices vi0, vi1,……, vin-1, vi0 where the first vertex of the sequence is same as the last one while all the other n-1 vertices are distinct.

20. What are the tricks used to reduce the size of the state-space tree?

The various tricks are

- Exploit the symmetry often present in combinatorial problems. So some solutions can be obtained by the reflection of others. This cuts the size of the tree by about half.
- Pre assign values to one or more components of a solution
- Rearranging the data of a given instance.

21. What are the additional features required in branch-and-bound when compared to backtracking?

Compared to backtracking, branch-and-bound requires:

- A way to provide, for every node of a state space tree, a bound on the best value of the objective function on any solution that can be obtained by adding further components to the partial solution represented by the node.
- The value of the best solution seen so far

22. What is a feasible solution and what is an optimal solution?

In optimization problems, a feasible solution is a point in the problem‘s search space that satisfies all the problem‘s constraints, while an optimal solution is a feasible solution with the best value of the objective function.

23. When can a search path be terminated in a branch-and-bound algorithm?

A search path at the current node in a state-space tree of a branch and- bound algorithm can be terminated if o The value of the node‘s bound is not better than the value of the best solution seen so far

- The node represents no feasible solution because the constraints of the problem are already violated.
- The subset of feasible solutions represented by the node consists of a single point in this case compare the value of the objective function for this feasible solution with that of the best solution seen so far and update the latter with the former if the new solution is better.

24.What is the assignment problem?

Assigning _n‘ people to _n‘ jobs so that the total cost of the assignment is as small as possible. The instance of the problem is specified as a n-by-n cost matrix C so that the problem can be stated as: select one element in each row of the matrix so that no two selected items are in the same column and the sum is the smallest possible.

25. What is best-first branch-and-bound?

It is sensible to consider a node with the best bound as the most promising, although this does not preclude the possibility that an optimal solution will ultimately belong to a different branch of the state-space tree. This strategy is called best-first branch-and-bound.

26. What is knapsack problem?

Given n items of known weights wi and values vi, i=1,2,…,n, and a knapsack of capacity W, find the most valuable subset of the items that fit the knapsack. It is convenient to order the items of a

given instance in descending order by their value-to-weight ratios. Then the first item gives the best payoff per weight unit and the last one gives the worst payoff per weight unit.

27.Give the formula used to find the upper bound for knapsack problem.

A simple way to find the upper bound _ub' is to add _v', the total value of the items already selected, the product of the remaining capacity of the knapsack W-w and the best per unit payoff among the remaining items, which is vi+1/wi+1

ub = v + (W-w)(vi+1/wi+1)

28. What is the traveling salesman problem?

The problem can be modeled as a weighted graph, with the graph's vertices representing the cities and the edge weights specifying the distances. Then the problem can be stated as finding the shortest Hamiltonian circuit of the graph, where the Hamiltonian is defined as a cycle that passes through all the vertices of the graph exactly once.

29.List out the steps of Greedy algorithms for the Knapsack Problem

a) Compare the value to weight ratio

b) Sort the items in nonincreasing order of the ratios

c) If the current item on the list fits into the knapsack place it in the knapsack; otherwise proceed to the next.

30.Define Christofides Algorithm

Christofides Algorithm is an algorithm exploits a relationship with a minimum spanning tree but it in

a more sophisticated way than the twice around the tree algorithm. It has the performance ratio 1:5

31.List out the steps of Nearest-neighbor Algorithm

a. Choose an arbitrary city as the start

b. Repeat the following operations until all the cities have been visited: go to the unvisited city nearest the one visited last

c. Return to the starting city.

32.What is meant by c-approximation Algorithm?

We can also says that a polynomial-time approximation algorithm is a c-approximation algorithm if

its performance ratio is at most c, that is, for any instance of the problem in

question F (sa) <c f(s*)

33.Define Performance ratio

The best upper bound of possible r(Ss)values taken over all instances of the problem is called the

performance ratio of the algorithm and denoted RA

34.List out the steps of Twice-around tree algorithm.

(A) Construct a minimum spanning tree the graph corresponding to a given instance of the traveling salesman problem.

(B) Starting at an arbitrary vertex, perform a walk around the minimum spanning tree recording the vertices passed by.

(C) Scan The List of Vertices Obtained in Step2 and Eliminate from it all repeated occurrences of the same vertices except the starting one at the end of the list

35.Define Integer Linear Programming

It is a Programming to find the 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/or in equalities.

36.Define nondeterministic Polynomial

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

37. Define NP-Complete

An NP-Complete problem is a problem in NP that is as difficult as any other problem in this class because any other problem in NP can be reduced to it in Polynomial time.

38.Define Polynomial reducible

A Decision problem D1 is said to be polynomial reducible to a decision problem D2 if there exists a function t that transforms instances of D2 such that

- T maps all yes instances of D1 to yes instances of D2 and all noninstances of D1 to no instance of D2
- T is computable by a Polynomial-time algorithm

39.What is the difference between tractable and intractable?

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

40.Define undecidable Problem

Some decision problem that cannot be solved at all by any algorithm is called undecidable algorithm.

41.Define Heuristic

Generally speaking, a heuristic is a "rule of thumb," or a good guide to follow when making decisions. In computer science, a heuristic has a similar meaning, but refers specifically to algorithms.

42. What are the strengths of backtracking and branch-and-bound?

The strengths are as follows

- It is typically applied to difficult combinatorial problems for which no efficient algorithm for finding exact solution possibly exist
- It holds hope for solving some instances of nontrivial sizes in an acceptable amount of time

Even if it does not eliminate any elements of a problem‘s state space and ends up generating all its elements, it provides a specific technique for doing so, which can be of some value