SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)
COIMBATORE – 35

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (UG & PG)

Second Year Computer Science and Engineering, 4th Semester

2 Marks Question and Answer

Subject Code & Name: 23ITB201/Design and Analysis of Algorithm

Prepared by: Dr.M.Shobana, ASP/CSE , Ms.M.Lavanya AP/CSE, Ms.Priyadharshini AP/CSE

UNIT I - INTRODUCTION

1. **What is performance measurement?**
   Performance measurement is concerned with obtaining the space and the time requirements of a particular algorithm.

2. **What is an algorithm?**
   An algorithm is a finite set of instructions that, if followed, accomplishes a particular task.

3. **What are the characteristics of an algorithm?**
   1) Input
   2) Output
   3) Definiteness
   4) Finiteness
   5) Effectiveness

4. **Define Program.**
   A program is the expression of an algorithm in a programming language. Sometimes works such as procedure, function and subroutine are used synonymously program.

5. **Write the For LOOP general format.**
   The general form of a for Loop is
   For variable : = value 1 to value 2 step
   Step do
   {
   <statement 1>
   <statement n >
   }

6. **What is recursive algorithm?**
   An algorithm is said to be recursive if the same algorithm is invoked in the body. An algorithm that calls itself is Direct recursive. Algorithm A is said to be indeed recursive if it calls another algorithm, which in turn calls A.

7. What is space complexity?

The space complexity of an algorithm is the amount of memory it needs to run to completion.

8. What is time complexity?

The time complexity of an algorithm is the amount of computer time it needs to run to completion.

9. Give the two major phases of performance evaluation.

Performance evaluation can be loosely divided into two major phases:

(i) A prior estimates (performance analysis)
(ii) A Posterior testing(performance measurement)

10. Define input size.

The input size of any instance of a problem is defined to be the number of words(or the number of elements) needed to describe that instance.

11. Define best-case step count.

The best-case step count is the minimum number of steps that can be executed for the given parameters.

12. Define worst-case step count.

The worst-case step count is the maximum number of steps that can be executed for the given parameters.

13. Define average step count.

The average step count is the average number of steps executed an instances with the given parameters.

14. Define the asymptotic notation "Big oh" (O).

The function f(n) = O(g(n)) iff there exist positive constants C and no such that f(n)£ C * g(n) for all n, n ³n0.

15. Define the asymptotic notation "Omega" (Ω).

The function f(n) =W (g(n)) iff there exist positive constant C and no such that f(n) C * g(n) for all n, n ³ n0.

16. Define the asymptotic notation "theta" (Θ).

The function f(n) =q (g(n)) iff there exist positive constant C1, C2, and no such that C1 g(n)£ f(n) £ C2 g(n) for all n, n ³ n0.

17. Define Little "oh".

The function f(n) = 0(g(n))

Iff Lim f(n) = 0

n - µ g(n)

18. Define Little Omega.

The function f(n) = w (g(n))
iff
Lim f(n) = 0
n - µ g(n)

19. Write algorithm using iterative function to fine sum of n numbers.

```
Algorithm sum(a,n)
{
        S : = 0.0
        For i=1 to n do
        S : - S + a[i];
        Return S;
}
```

20. Write an algorithm using Recursive function to fine sum of n numbers.

```
Algorithm Rsum (a,n)
{
        If(n_ 0) then
        Return 0.0;
        Else Return Rsum(a, n- 1) + a(n);
}
```

21. What are the basic asymptotic efficiency classes?

The various basic efficiency classes are
Constant:1
Logarithmic: log n
Linear: n
N-log-n: n log n
Quadratic: n^2
Cubic: n^3
Exponential:2^n
Factorial:n!

22. What is the substitution method?

One of the methods for solving any such recurrence relation is called the substitution method.

23. Define Time Space Tradeoff.

Time and space complexity can be reduced only to certain levels, as later on reduction of time increases the space and vice-versa. This is known as Time-space trade-off.

24. Why do we need algorithm analysis?

- Writing a working program is not good enough.
- The program may be in-efficient.
- If the program is run on a large data set, then the running time becomes an issue.

25. List the factors which affects the running time of the algorithm.

A. Computer
B. Compiler

C. Algorithm used
D. Input to the algorithm
   i. The content of the input affects the running time
   ii. Typically, the input size is the main consideration.

26. What is recurrence equation?

A recurrence equation is an equation or inequality that describes a function in terms of its value on smaller inputs.

27. What is classification of the recurrence equation?

Recurrence Equations can be classified into

1. Homogeneous
2. Inhomogeneous

UNIT II - BRUTE FORCE AND DIVIDE-AND-CONQUER

1. Define the divide and conquer method.

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 tocombine subsolutions into a solution of the whole. If the subproblems are still relatively large, then the divide-and conquer strategy can possibly be reapplied.

2. Define Control abstraction.

A control abstraction we mean a procedure whose flow of control is clear but whose primary operations are by other procedures whose precise meanings are left undefined.

3. Write the Control abstraction for Divide-and conquer.

```
Algorithm D And(r)
{
        if small(p) then return S(r);
        else
        {
                divide P into smaller instance _ 1, _ 2, _ k, k ³ 1;
                Apply D and C to each of these subproblems
                Return combine (DAnd C(_1) DAnd C(_2),----, DAnd ((_k));
        }
}
```

4. What is the Binary search?

If 'q' is always chosen such that 'aq' is the middle element(that is, q=[(n+1)/2), then the resulting search algorithm is known as binary search.

5. Give computing time for Binary search.

The computing time of binary search by giving formulas that describe the best, average and worst cases. Successful searches q(1) q(logn) q(Logn) best average worst unsuccessful searches q(log n) best, average and worst.

6. Define external path length.

The external path length E, is defines analogously as sum of the distance of all external nodes from the root.

7. Define internal path length.

The internal path length 'I' is the sum of the distances of all internal nodes from the root.

8. Write about Traveling salesperson problem.

Let g = (V, E) be a directed. The tour of G is a directed simple cycle that includes every vertex in V. The cost of a tour is the sum of the cost of the edges on the tour. The traveling salesperson problem to find a tour of minimum cost.

9. Write some applications of Traveling salesperson problem.

- Routing a postal van to pick up mail from boxes located at n different sites.
- Using a robot arm to tighten the nuts on some piece of machinery on an assembly line.
- Production environment in which several commodities are manufactured on the same set of machines.

10. Give the time complexity and space complexity of Traveling salesperson problem.

- Time complexity is O (n2 2n).
- Space complexity is O (n 2n).

11. What is the maximum and minimum problem?

The problem is to find the maximum and minimum items in a set of 'n' elements. Though this problem may look so simple as to be contrived, it allows us to demonstrate divide and conquer in simple setting.

12. What is the Quick sort?

N quick sort, the division into sub arrays is made so that the sorted sub arrays do not need to be merged later.

13. Write the analysis for the Quick sort.

In analyzing QUICKSORT, we can only make the number of element comparisons C(n). It is easy to see that the frequency count of other operations is of the same order as C(n).

14. Is insertion sort better than the merge sort?

Insertion sort works exceedingly fast on arrays of less then 16 elements, though for large 'n' its computing time is O(n2).

15. Write a algorithm for straightforward maximum and minimum.

```
algorithm straight MaxMin(a,n,max,min)
//set max to the maximum and min to the minimum of a[1:n]
{
```

```
        max := min: = a[i];
        for i = 2 to n do
        {
                if(a[i] >max) then max: = a[i];
                if(a[i] >min) then min: = a[i];
        }
}
```

16. Give the recurrence relation of divide-and-conquer.

The recurrence relation is

T(n) = g(n)

T(n1) + T(n2) + ----+ T(nk) + f(n)

17. Write the algorithm for Iterative binary search.

```
Algorithm BinSearch(a,n,x)
//Given an array a[1:n] of elements in non decreasing
// order, n>0, determine whether x is present
{
        low : = 1;
        high : = n;
        while (low < high) do
        {
                mid : = [(low+high)/2];
                if(x < a[mid]) then high:= mid-1;
                else if (x >a[mid]) then low:=mid + 1;
                else return mid;
        }
        return 0;
}
```

18. What are internal nodes?

The circular node is called the internal nodes.

19. Describe the recurrence relation of merge sort.

If the time for the merging operation is proportional to n, then the computing time of merge sort is described by the recurrence relation

n = 1, a a constant

T(n) = a

2T (n/2) + n n >1, c a constant

20. What is meant by feasible solution?

Given n inputs and we are required to form a subset such that it satisfies some given constraints then such a subset is called feasible solution.

21. Define optimal solution.

A feasible solution either maximizes or minimizes the given objective function is called as optimal solution.

22. What is Knapsack problem?

A bag or sack is given capacity n and n objects are given. Each object has weight wi and profit pi .Fraction of object is considered as xi (i.e) 0<=xi<=1 .If fraction is 1 then entire object is put into sack. When we place this fraction into the sack we get wixi and pixi.

23. Define weighted tree.

A directed binary tree for which each edge is labeled with a real number (weight) is called as weighted tree.

24. What is the use of TVSP?

In places where the loss exceeds the tolerance level boosters have to the placed. Given a network and loss tolerance level the tree vertex splitting problems is to determine an optimal placement of boosters.

25. What is activity selection problem?

The 'n' task will be given with starting time si and finishing time fi. Feasible Solution is that the task should not overlap and optimal solution is that the task should be completed in minimum number of machine set.

26. Write the specification of TVSP.

Let T= (V, E, W) be a weighted directed binary tree where

V_ vertex set

E_ edge set

W_ weight function for the edge.

W is more commonly denoted as w (i,j) which is the weight of the edge <i,j> _ E.

27. Define Forest.

Collection of sub trees that are obtained when root node is eliminated is known as forest

28. Define post order traversal.

The order in which the TVSP visits the nodes of the tree is called the post order traversal.

29. Write the formula to calculate delay and also write the condition in which the node gets splitted.

To calculate delay

d(u)=max{d(v)+w(u, v)}

v _ c(u)

The condition in which the node gets splitted are:

d(u)+w(u ,v)>_ and also d(u) is set to zero.

30. What is meant by tolerance level?

The network can tolerate the losses only up to a certain limit tolerance limit.

31. When is a task said to be late in a schedule?

A task is said to be late in any schedule if it finishes after its deadline else a task is early in a schedule.

32. Define feasible solution for TVSP.

Given a weighted tree T(V,E,W) and a tolerance limit _ any subset X of V is a feasible solution if d(T/X).

33. Define optimal solution for TVSP.

An optimal solution is one in which the number of nodes in X is minimized.

UNIT III - DYNAMIC PROGRAMMING AND GREEDY TECHNIQUE

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.

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.

1. Characterize the structure of an optimal solution.
2. Recursively define the value of the optimal solution.
3. Compute the value of an optimal solution in the bottom-up fashion.
4. 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. Give an example of dynamic programming and explain.

An example of dynamic programming is knapsack problem. The solution to the knapsack problem can be viewed as a result of sequence of decisions. We have to decide the value of xi for 1<i<n. First we make a decision on x1 and then on x2 and so on. An optimal sequence of decisions maximizes the object function Spi xi.

7. **Write about optimal merge pattern problem.**

Two files x1 and x2 containing m & n records could be merged together to obtain one merged file. When more than 2 files are to be merged together. The merge

can be accomplished by repeatedly merging the files in pairs. An optimal merge pattern tells which pair of files should be merged at each step. An optimal sequence is a least cost sequence.

8. What does Job sequencing with deadlines mean?

i. Given a set of 'n' jobs each job 'i' has a deadline di such that di>=0 and a

- profit pi such that pi>=0.
- For job 'i' profit pi is earned iff it is completed within deadline.
- Processing the job on the machine is for 1unit of time. Only one machine is available.

9. Write the general algorithm for Greedy method control abstraction.

```
Algorithm Greedy (a, n)
{       solution=0;
        for i=1 to n do  {
                x= select(a);
                if feasible(solution ,x) then
                solution=Union(solution ,x);   }
        return solution;
}
```

10. Define optimal solution for Job sequencing with deadlines.

Feasible solution with maximum profit is optimal solution for Job sequencing with deadlines.

11. Write the difference between the Greedy method and Dynamic programming.

Greedy method	Dynamic Programming
▪ Only one sequence of decision is generated1.	Many number of decisions are generated.
▪ It does not guarantee to give an always.	It definitely gives an optimal solution optimal solution always.

12. 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.

13. What is the Greedy choice property?

i. The first component is greedy choice property (i.e.) a globally optimal solution can arrive at by making a locally optimal choice.

ii. 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.

iii. It progresses in top down fashion.

14. 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.

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

i. Determine the optimal substructure of the problem.
ii. Develop a recursive solution.
iii. Prove that at any stage of recursion one of the optimal choices is greedy choice.

Thus it is always safe to make greedy choice.

iv. Show that all but one of the sub problems induced by having made the greedy

choice are empty.

v. Develop a recursive algorithm and convert into iterative algorithm.

16. Explain any one method of finding the shortest path.

One way of finding a shortest path from vertex i to j in a directed graph G is to decide which vertex should be the second, which is the third, which is the fourth, and so on, until vertex j is reached. An optimal sequence of decisions is one that results in a path of least length.

17. Define 0/1 knapsack problem.

The solution to the knapsack problem can be viewed as a result of sequence of decisions. We have to decide the value of xi. xi is restricted to have the value 0 or 1 and by using the function knap(l, j, y) we can represent the problem as maximum Spi xi subject to Swi xi < y where l -iteration, j - number of objects, y – capacity.

18. What is the formula to calculate optimal solution in 0/1 knapsack problem?

The formula to calculate optimal solution is

g0(m)=max{g1, g1(m-w1)+p1}.

19. Define flow shop scheduling.

The processing of jobs requires the performance of several distinct job. In flow shop we have n jobs each requiring n tasks i.e. T1i, T2i,…...Tmi, 1<i<n.

20. What are the conditions of flow shop scheduling?

- Let Tji denote jth task of the ith job. Task Tij is to be performed on Pj number of processors where 1<j<m i.e. number of processors will be equal to number of task
- Any task Tji must be assigned to the processor Pj.
- No processor can have more than one task assigned to it at any time. For any jobI
- processing the task for j>1 cannot be started until T(j-i),i has been completed.

21. Define non preemptive schedule.

A non preemptive schedule is a schedule in which the processing of a task on any processor is not terminated until the task is complete.

22. Define preemptive schedule.

A preemptive schedule is a schedule in which the processing of a task on any processor can be terminated before the task is completed.

23. Define finish time.

The finish time fi (S) of job i is the time at which all tasks of job i have been completed in schedule S. The finish time F(S) of schedule S is given by F(S)=max{ fi (S)} 1<i<n.

24. Define mean flow time.

The mean flow time MFT (S) is defined to be]

MFT (S) = 1 S fi(S)

n 1<i<n

25. Define optimal finish time.

Optimal finish time scheduling for a given set of tasks is a non preemptive schedule S for which F (S) is minimum over all non preemptive schedules S.

26. Define preemptive optimal finish time.

Preemptive optimal finish time scheduling for a given set of tasks is a preemptive schedule S for which F (S) is minimum over all preemptive schedules S.

27. What are static trees?

The tree organizations that are independent of the problem instance being solved are called as static tree.

28. What are dynamic trees?

The tree organizations those are independent of the problem instance being solved are called as static tree.

29. Define a live node.

A node which has been generated and all of whose children have not yet been generated is called as a live node.

30. Define a E – node.

E – node (or) node being expanded. Any live node whose children are currently being generated is called as a E – node.

31. Define a dead node.

Dead node is defined as a generated node, which is to be expanded further all of whose children have been generated.

32. What is spanning tree?

A spanning tree of a graph G is a sub graph which is basically a tree and it contains all the vertices of G containing no circuit.

33. What is minimum spanning tree?

A minimum spanning tree of a weighted connected graph G is a spanning tree with minimum or smallest weight.

34. What is the difference between Prim's and Kruskal's algorithm?

S.No	Prim's algorithm	Kruskal's algorithm
1	This algorithm is for obtaining minimum spanning tree by selecting the adjacent vertices of already selected vertices.	This algorithm is for obtaining minimum spanning tree but it is not necessary to choose adjacent vertices of already selected vertices

UNIT IV - ITERATIVE IMPROVEMENT AND STRING MATCHING

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 xij 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

$0 \leq xij \leq uij$ for every edge (i,j) 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 - BACKTRACKING AND BRANCH & BOUND

1. What is Bi-connected component?

A bi-connected graph G = (V, E) is a connected graph which has no articulation point. A bi-connected component of a graph G is maximal bi-connected sub graph.

2. Define Articulation point.

The Graph G = (V, E_0) be a connected undirected graph, then an articulation point of a graph G is a vertex whose removal disconnects graph G. This articulation point is a kind of cut-vertex.

3. What is Backtracking?

Find a solution by trying one of several choices. If the choice proves incorrect, computation backtracks or restarts at the point of choice and tries another choice. It is often convenient to maintain choice points and alternate choices using recursion.

4. What is a branch and bound technique?

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

5. What is the difference between backtracking and branch and bound?

S.No	Backtracking	Branch and bound
1	State space tree is constructed using depth first search	State space tree is constructed using best first search
2.	Finds solutions for combinatorial Non-optimization problems.	Finds solutions for combinatorial optimization problems.
3.	No bounds are associated with the nodes in the state space tree.	Bounds are associated with the each and every node in the state space tree.

6. What are the types of branch and bound?

- BFS- like state space search (or) FIFO search
- D search- like state space search (or) LIFO search

7. What is Least Cost (LC) search?

The Least cost search is a kind of search in which least cost is involved for reaching to answer node. At each E-node the probability of being an answer node is checked.

8. Define P and NP.

- P is the class of decision problems that can be solved by a deterministic polynomial algorithm.
- NP is the class of decision problem that can be solved by a non-deterministic polynomial algorithm.

9. Define NP hard and NP complete.

If an NP hard problem can be solved in polynomial time then all NP complete problems can also be solved in polynomial time. An optimization problem may be hard, cannot be complete.

A problem that is NP complete can be solved in polynomial time then all NP complete problems can also be solved in polynomial time. Only decision problem can be NP complete.

10. Differentiate between Deterministic and Non Deterministic algorithms.

S.No	Deterministic algorithm	Non-Deterministic algorithm
1	Deterministic algorithms are the algorithms with uniquely defined results and predictable in terms of output for certain output.	Non-Deterministic algorithms are the algorithms that allowed containing operations whose outcomes are limited to a given a set possibilities instead of being uniquely defined.

10. What is called a decision problem?

Any problem for which the answer is either zero or one is called a decision problem.

11. What is called an optimization problem?

Any problem that involves the identification of an optimal value of a given cost function is known as an optimization problem.

12. What are the different types of constraints?

- Explicit constraints
- Implicit constraints

13. List out some problems which are solved using backtracking algorithm.

- 8- Queens problem
- Hamiltonian problem
- Knapsack problem

14. What do you mean by explicit constraint?

Explicit constraints are rules that restrict each x_i to take values only from a given set.

e.g :

$x_i > 0$ or $S_i =$ {all nonnegative real numbers}

$x_i = 0$ or 1 or $S_i =$ {0,1}

$l_i \leq x_i \leq u_i$ or $S_i =$ {a: $l_i \leq a \leq u_i$}

All tuples satisfying the explicit constraints define a possible solution space for i (i-problem instance).

15. Define implicit constraint.

The implicit constraints are rules that determine which of the tuples in the solution space of I satisfy the criterion function. Thus implicit constraints describe the way in which the x_i must relate to each other. E.g: 8-queens problem.

16. What is 8-Queens problem?

It is a classic combinatorial problem that place eight queens on an 8×8 chessboard so that no two attack that is no two of them are on the same row, column, or diagonal.

17. What is state space tree?

A state space tree is a rooted tree whose nodes represent partially constructed solutions to given problem. In backtracking the state space tree is built for finding the solution. This tree is built using depth first search fashion.

18. Define static tree?

Static trees are ones for which tree organizations are independent of the problem instance being solved

- Fixed tuple size formulation.
- Tree organization is independent of the problem instance being solved.

19. What is live node?

Live node is a generated node for which all of the children have not yet been generated.

20. What is dead node?

Dead node is a generated node that is not to be expanded any further or for which all of its children have been generated.

- All the children of a dead node are already generated.
- Live nodes are killed using a bounding function to make them dead nodes.
- These states satisfy implicit constraints.

21. What is bounding function?

Bounding function is a function used to kill live nodes without generating all their children.

22. What is solution space?

Tuples that satisfy the explicit constraints define a solution space. The solution space can be organized into a tree.

23. State if backtracking always produces optimal solution.

Backtracking always produces optimal solution since backtracking is a systematic way to go through all the possible configurations of a solution space for the problem instance

24. Define Sum of subsets.

There are n distinct positive numbers and desired to find all combinations of these numbers, whose sums are m. This is called sum of subset problem.

25. Define Hamiltonian cycle.

G = (V, E) be a connected graph with n vertices. A Hamiltonian cycle is a round trip path along n edges of G which visits every vertex once and returns to its starting position. The tour of a traveling salesperson problem is a Hamiltonian cycle.

26. What are the two types of solution space tree organization?

- Variable tuple size formulation
- Fixed tuple size formulation

27. What is E-node?

E-node is a live node whose children are currently being generated or expanded.

28. Define answer state.

Answer states are that solution state s, for which the path from root node to s defines a tuple that is a member of set of solutions. These states satisfy implicit constraints.

29. What are the requirements that are needed for performing backtracking?

To solve any problem using backtracking, it requires that all the solutions satisfy a complex set of constraints. They are:

i. Explicit constraints.
ii. Implicit constraints.

30. Define explicit constraint.

They are rules that restrict each xi to take on values only from a give set. They depend on the particular instance I of the problem being solved. All tuples that satisfy the explicit constraints define a possible solution space.

31. Define implicit constraint.

They are rules that determine which of the tuples in the solution space of I satisfy the criteria function. It describes the way in which the xi must relate to each other.

32. Define state space tree.

The tree organization of the solution space is referred to as state space tree.

33. Define state space of the problem.

All the paths from the root of the organization tree to all the nodes is called as state space of the problem

34. Define answer states.

Answer states are those solution states s for which the path from the root to s defines a tuple that is a member of the set of solutions of the problem.