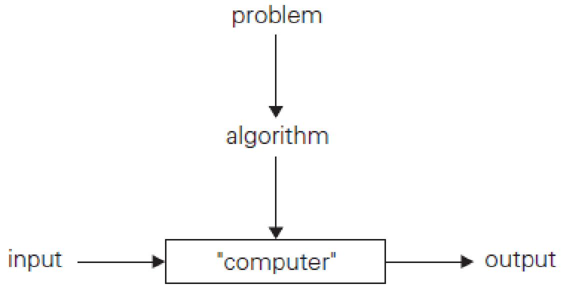
PART A
UNIT – I
INTRODUCTION

1. What is an algorithm? [Nov Dec 2013]

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



2. What do you mean by Amortized Analysis?

Amortized analysis finds the average running time per operation over a worst case sequence of operations. Amortized analysis differs from average-case performance in that probability is not involved; amortized analysis guarantees the time per operation over worst-case performance.

3. What are important problem types? (or) Enumerate some important types of problems?

(i) Sorting **(ii)** Searching **(iii)** Numerical problems **(iv)** Geometric problems **(v)** Combinatorial Problems **(vi)** Graph Problems **(vii)** String processing Problems

4. Name some basic Efficiency classes.

(i) Constant **(ii)** Logarithmic **(iii)** Linear **(iv)** Quadratic **(v)** Cubic **(vi)** Exponential **(vii)** Factorial

5. What are algorithm design techniques?

Algorithm design techniques (or strategies or paradigms) are general approaches to solving problems algorithmatically, applicable to a variety of problems from different areas of computing. General design techniques are: **(i)** Brute force **(ii)** divide and conquer **(iii)** decrease and conquer **(iv)** transform and conquer **(v)** Greedy technique **(vi)** dynamic programming **(vii)** backtracking **(viii)** branch and bound

6. How is an algorithm's time efficiency measured?

Time efficiency indicates how fast the algorithm runs. An algorithm's time efficiency is measured as a function of its input size by counting the number of times its basic operation (running time) is executed. Basic operation is the most time consuming operation in the algorithm's innermost loop.

7. What is Big 'Oh' notation? [May/June 2012]

A function t(n) is said to be in O(g(n)), denoted t(n) O(g(n)) , if t(n) is bounded above by some constant multiple of g(n) for all large n, i.e., if there exist some positive constant c and some nonnegative integers no such that

$$t(n) \leq cg(n) \text{ for all } n \geq n0$$

8. What is an Activation frame?

It is a storage area for an invocation of recursive program (parameters, local variables, return address/value etc.). Activation frame allocated from frame stack pointed by frame pointer.

9. Define order of an algorithm.

Measuring the performance of an algorithm in relation with the input size n is known as order of growth

10. What is recursive call?

Recursive algorithm is an algorithm where a function calls itself and it is known as recursive call. An algorithm that calls itself is direct recursive. An algorithm 'A' is said to be indirect recursive if it calls another algorithm which in turn calls 'A'.

11. Compare the order of growth n(n-1)/2 and n^2. [May/June 2016]

n	**n(n-1)/2**	***n^2***
Polynomial	**Quadratic**	**Quadratic**
1	0	1
2	1	4
4	6	16
8	28	64
10	45	10^2
10^2	4950	10^4
Complexity	Low	High
Growth	Low	high

n(n-1)/2 is lesser than the half of n^2

12. Define recurrence relation. [Nov/Dec 2016]

A recurrence relation is an equation that defines a sequence based on a rule that gives the next term as a function of the previous term(s). The simplest form of a recurrence relation is the case where the next term depends only on the immediately previous term.

13. How is the efficiency of the algorithm defined?

The efficiency of an algorithm is defined with the components.

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

14. Write down the properties of asymptotic notations. [April/May 2015]

Asymptotic notation is a notation, which is used to take meaningful statement about the efficiency of a program. To compare and rank such orders of growth, computer scientists use three notations, they are:

- O - Big oh notation
- Ω - Big omega notation
- Θ - Big theta notation

15. What do you mean by time complexity and space complexity of an algorithm? [Nov/Dec 2012]

Time complexity indicates how fast the algorithm runs. Space complexity deals with extra memory it require. Time efficiency is analyzed by determining the number of repetitions of the basic operation as a function of input size. Basic operation: the operation that contributes most towards the running time of the algorithm The running time of an algorithm is the function defined by the number of steps (or amount of memory) required to solve input instances of size n.

16. Define Big Omega Notations. [May/June 2013]

A function t(n) is said to be in Ω(g(n)) , denoted t(n) Є Ω((g(n)) , if t(n) is bounded below by some positive constant multiple of g(n) for all large n, i.e., if there exist some positive constant c and some nonnegative integer n0 such that

$$t(n) \geq cg(n) \text{ for all for all } n \geq n0$$

17. What are the different criteria used to improve the effectiveness of algorithm?

- The effectiveness of algorithm is improved, when the design, satisfies the following constraints to be minimum.
  - Time efficiency - how fast an algorithm in question runs.
  - Space efficiency – an extra space the algorithm requires
- The algorithm has to provide result for all valid inputs.

***18.* Give the Euclid's algorithm for computing *gcd(m, n)* [May/June 2016]**

```
Algorithm Euclid_gcd(m, n)
        //Computes gcd(m, n) by Euclid's algorithm
        //Input: Two nonnegative, not-both-zero integers m and n
        //Output: Greatest common divisor of m and n
                while n ≠ 0 do
                        r ←m mod n
                        m←n
                        n←r
                return m
```

Example: gcd(60, 24) = gcd(24, 12) = gcd(12, 0) = 12.

19. Write general plan for analyzing non-recursive algorithms.

- Decide on parameter indicating an input's size.
- Identify the algorithm's basic operation
- Checking the no. of times basic operation executed depends on size of input. If it depends on some additional property, then best, worst, avg. cases need to be investigated
- Set up sum expressing the no. of times the basic operation is executed. (establishing order of growth)

20. How will you measure input size of algorithms?

The time taken by an algorithm grows with the size of the input. So the running time of the program depends on the size of its input. The input size is measured as the number of items in the input that is a parameter n is indicating the algorithm's input size.

21. Write general plan for analyzing recursive algorithms.

- Decide on parameter indicating an input's size.
- Identify the algorithm's basic operation
- Checking the no. of times basic operation executed depends on size of input. If it depends on some additional property, then best, worst, avg. cases need to be investigated
- Set up the recurrence relation, with an appropriate initial condition, for the number of times the basic operation is executed
- Solve recurrence (establishing order of growth)

22. What do you mean by Combinatorial Problem?

Combinatorial Problems are problems that ask to find a combinatorial object-such as permutation, a combination, or a subset--that satisfies certain constraints and has some desired property.

23. Define Big Theta Notations [Nov/Dec 2014]

A function t(n) is said to be in $\Theta(g(n))$, denoted $t(n) \in \Theta(g(n))$, if t(n) is bounded both above and below by some positive constant multiple of g(n) for all large n, i.e., if there exist some positive constants c1 and c2 and some nonnegative integer n0 such that $c1\ g(n) \leq t(n) \leq c2g(n)$ for all $n \geq n0$

24. What is performance measurement?

Performance measurement is concerned with obtaining the space and the time requirements of a particular algorithm.

25. Define Program.

A program is the expression of an algorithm in a programming language. The procedure, function and subroutine are used synonymously program.

26. What is recursive algorithm?

An algorithm is said to be recursive if the same algorithm is invoked in the body.

27. What is space complexity and time complexity?

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

30. Give the two major phases of performance evaluation.

Performance evaluation can be loosely divided into two major phases:

- a prior estimates (performance analysis)
- a Posterior testing (performance measurement)

31. 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.

32. Define average step, best-case and worst-case step count.

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

33. Define Little "oh". [April/May 2014]

The function f(n) = 0(g(n)) iff
Lim f(n) =0
n →∞ g(n)

34. Define Little Omega. [April/May 2014]

The function f(n) = ω (g(n)) iff
Lim f(n) =0
n →∞ g(n)

35. Write an algorithm to find the number of binary digits in the binary representation of a positive decimal integer. [April/May 2015]

```
Algorithm Binary(n)
    //Input: A positive decimal integer n
    //Output: The number of binary digits in n's binary representation
        count ←1
        while n > 1 do
            count ←count + 1
            n←[n/2]
        return count
```

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

```
    Algorithm Rsum (a,n)
     If($n \leq 0$) then
    Return 0.0;
    Else Return Rsum(a, n- 1) + a(n);
```

37. What are six steps processes in algorithmic problem solving? [Nov/Dec 2009]

- Understanding the problem
- Decision making on - Capabilities of computational devices, Choice of exact or approximate problem solving, Data structures
- Algorithmic strategies
- Specification of algorithm
- Algorithmic verification
- Analysis of algorithms

39. Define algorithm validation.

The process of measuring the effectiveness of the algorithm before actually making program or code from it, in order to know whether the algorithm is correct for valid input is known as algorithm validation. Algorithm validation can be done with the help of mathematical and empirical method

40. What are the components of fixed and variable part in space complexity? The space requirements S(p) is given by following formula

$$S(p)= C + Sp$$

C - fixed part that denotes the space of inputs and outputs, this space is the amount of space taken by instructions, variables and identification
Sp - denotes variable part of the space. It denotes the space requirements based on particular problem instance

41. Define Program proving and program verification. [April/May 2014]

Program proving means proving each and every instruction of the program with the help of mathematical theorems. Program verification means checking the correctness of the program

42. What is recurrence equation? [April/May 2010, Nov/Dec 2016]

The recurrence is an equation that defines a sequence recursively It is normally in the following form

$$T(n) = T(n-1) + n \quad n>0 \qquad T(0) = 0$$

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

The function $f(n) = O(g(n))$ iff there exist positive constants C and no such that $f(n) \leq C * g(n)$ for all n, $n \geq n_0$.

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

The function $f(n) = \Omega(g(n))$ iff there exist positive constant C and no such that f(n) C * g(n) for all n, $n \geq n_0$.

45. Define the asymptotic t\notation "theta" (θ)

The function $f(n) = \theta(g(n))$ iff there exist positive constant C_1, C_2, and no such that $C_1 g(n) \leq f(n) \leq C_2 g(n)$ for all n, $n \geq n_0$.

46. 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;
}
```

47. 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);
}
```

48. 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: n2
- Cubic: n3
- Exponential:2n
- Factorial:n!

49. 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.

50. 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.

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

A. Computer
B. Compiler
C. Input to the algorithm
  i. The content of the input affects the running time
  ii. Typically, the input size is the main consideration.

52. 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

1. Euclid's algorithm

2. Consecutive integer checking algorithm
3. Middle school procedure

53. 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

54. 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

55. 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 and Coding an algorithm

56. Give an non-recursive algorithm to find out the largest element in a list of n numbers.

```
ALGORITHM MaxElement(A[0..n-1])
//Determines the value of the largest element in a given array
Input:An array A[0..n-1] of real numbers
//Output: The value of the largest element in A maxval ïƒŸ a[0]
for I ïƒŸ 1 to n-1 do
 if A[I] > maxval return maxval
ïƒŸ A[I] return maxval
```

57. Mention the two summation formulas.

The two summation formulas are

1 = u-l+1 where l u are some lower and upper integer limits = = 1+2+3+...+n = n(n+1)/2 n2/2 (n2)

59. Write a recursive algorithm for computing the nth fibonacci number.

```
ALGORITHM F(n)
// Computes the nth Fibonacci number recursively by using the definition
// Input A non-negative integer n
// Output The nth Fibonacci number
```

```
if n 1 return n
else return F(n-1)+F(n-2)
```

60. 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

$$\begin{cases} T(n) = a & n=1 \text{ , a is a constant} \\ 2T(n/2) + n & n=1\text{, c is a constant} \end{cases}$$

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

The recurrence relation is

$$T(n) = \begin{cases} g(n) \\ T(n_1) + T(n_2) + \text{---} + T(n_k) + f(n) \end{cases}$$

PART- B
UNIT-I

1. What are the important problem types focused by the researchers? Explain all the types with example
2. What is empirical analysis of an algorithm? Discuss its strength and weakness.
3. Discuss the fundamentals of analysis framework.
4. Explain the various asymptotic notations used in algorithm techniques.
5. Describe briefly the notions of complexity of algorithm.
6. What is pseudo code? Explain with an example.
7. Explain various criteria used for analyzing algorithms.
8. Discuss briefly the sequence of steps in designing and analyzing algorithm
9. Develop the general framework for analyzing the efficiency of algorithm
10. Explain the fundamentals of algorithmic problem solving with algorithm design and analysis process diagram
11. Elaborate the simple factoring algorithm with example .
12. Discuss the Euclid’s algorithm with example.
13. Outline the steps in analyzing & coding an algorithm.
14. Explain some of the problem types used in the design of algorithm.
15. Discuss the fundamentals of analysis framework.
16. Elaborate the various asymptotic notations used in algorithm design.
17. Explain the general framework for analyzing the efficiency of algorithm.
18. Outline the various Asymptotic efficiencies of an algorithm.
19. Discuss the basic efficiency classes.
20. Explain briefly the concept of algorithmic strategies.
21. Describe briefly the notions of complexity of an algorithm.
22. What is Pseudo-code? Explain with an example.
23. Find the complexity C(n) of the algorithm for the worst case, best case and average case. (Evaluate average case complexity for n=3, Where n is the number of inputs)
24. Set up & solve a recurrence relation for the number of key comparisons made by above pseudo code.

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-conquer strategy suggests splitting the inputs in to'k' distinct susbsets, 1<k <n, yielding 'k' sub problems. The sub problems must be solved, and then a method must be found to combine sub solutions into a solution of the whole. If the sub problems 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 DandC(P)
{
if small(p) then return S(P);
else
{
divide P into smaller instance p1,p2,…pk, k ≥1;
apply DandC to each of these subproblems
return combine (Dand C(p1) Dand C(p2),--- , Dand (pk));
}
}
```

4. What is the substitution method? [Nov/Dec 2014]

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

5. What is the binary search? [Nov/Dec 2014]

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.

6. Derive the complexity of Binary Search algorithm. [April/May 2015, Nov/Dec 2016]

In conclusion we are now able completely describe the computing time of binary search by giving formulas that describe the best, average and worst cases.

Successful searches	**Unsuccessful searches**
Best case - $\Theta(1)$ Average case - $\Theta(\log_2 n)$ Worst case - $\Theta(\log_2 n)$	Best case, Average case, Worst case - $\Theta(\log_2 n)$

7. Design a brute-force algorithm for computing the value of a polynomial $p(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x ... a_0$ at a given point x_0 and determine its worst case efficiency class. [April/May 2015]

```
Algorithm BetterBruteForcePolynomialEvaluation(P[0..n], x)
//The algorithm computes the value of polynomial P at a given point x by the "lowest-to-highest
//term" algorithm
//Input: Array P[0..n] of the coefficients of a polynomial of degree n, from the lowest to the
//highest, and a number x
//Output: The value of the polynomial at the point x
        p ← P[0]; power ← 1
        for i ← 1 to n do
                power ← power ∗ x
                p ← p + P[i] ∗ power
        return p
```

8. Write an algorithm for brute force closest-pair problem. [Nov/Dec 2016]

```
Algorithm BruteForceClosestPair(P )
//Finds distance between two closest points in the plane by brute force
//Input: A list P of n (n ≥ 2) points p1(x1, y1), . . . , pn(xn, yn)
//Output: The distance between the closest pair of points
    d←∞
    for i ←1 to n − 1 do
        for j ←i + 1 to n do
            d ←min(d, sqrt((xi − xj )^2 + (yi − yj )^2)) //sqrt is square root
    return d
```

9. Define internal path length and external path length.

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

10. Give the General strategy divide and conquer method. [May/June 2016]

A **divide and conquer algorithm** works by recursively breaking down a problem into two or more sub-problems of the same (or related) type (**divide**), until these become simple enough to be solved directly (**conquer**).

Divide-and-conquer algorithms work according to the following general plan:

- A problem is divided into several subproblems of the same type, ideally of about equal size.
- The subproblems are solved (typically recursively, though sometimes a different algorithm is employed, especially when subproblems become small enough).
- If necessary, the solutions to the subproblems are combined to get a solution to the original problem.

Example: Merge sort, Quick sort, Binary search, Multiplication of Large Integers and Strassen's Matrix Multiplication.

11. What is the Quick sort? Write the Analysis for the Quick sort.

Quicksort is dividing and conquer strategy that works by partitioning its input elements according to their value relative to some preselected element (pivot). It uses recursion and the method is also called partition –exchange sort. O(nlogn) in average and best cases, O(n2) in worst case

12. What is Merge sort and is insertion sort better than the merge sort?

Merge sort is divide and conquer strategy that works by dividing an input array in to two halves, sorting them recursively and then merging the two sorted halves to get the original array sorted Insertion sort works exceedingly fast on arrays of less than 16 elements, though for large 'n' its computing time is O(n2).

13. 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];
} }
```

14. What is general divide and conquer recurrence?

Time efficiency T(n)of many divide and conquer algorithms satisfies the equation. This is the general recurrence relation.

T(n)=a.T(n/b)+f(n)

15. What is Master's theorem?

Let T(n) be a monotonically increasing function that satisfies

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

T (1) = c

Where $a \geq 1, b \geq 2, c > 0$. *If* $f(n) \in \Theta(n^d)$ *where* $d \geq 0$, *then*

$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d, \\ \Theta(n^d \log n) & \text{if } a = b^d, \\ \Theta(n^{\log_b a}) & \text{if } a > b^d. \end{cases}$$

16. Write the algorithm for Iterative binary search.

```
Algorithm BinSearch(a,n,x)
//Given an array a[1:n] of elements in nondecreasing
// 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;
}
```

17. What is closest pair problem? [May/June 2016]

The closest-pair problem finds the two closest points in a set of n points. It is the simplest of a variety of problems in computational geometry that deals with proximity of points in the plane or higher-dimensional spaces. The distance between two Cartesian coordinates is calculated by Euclidean distance formula

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

18. What is convex hull?

Convex Hull is defined as: If S is a set of points then the Convex Hull of S is the smallest convex set containing S

19. Define Quick hull.

The divide-and conquer algorithm is called quick hull because of its resemblance to quick sort

20. What is travelling salesman problem?

Travelling salesman problem can be stated as follows – Consider that there are n cities and travelling salesman has to visit each city exactly once and has to return to the city from where the salesman has started.

21. What is Knapsack problem? [Nov/Dec 2014]

A bag or sack is given capacity 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.

22. Define feasible and optimal solution. [Nov/Dec 2013]

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

23. Define the single source shortest path problem. [May/June 2016]

Dijkstra's algorithm solves the single source shortest path problem of finding shortest paths from a given vertex (the source), to all the other vertices of a weighted graph or digraph. Dijkstra's algorithm provides a correct solution for a graph with non negative weights.

24. State Assignment problem. [May/June 2016]

There are n people who need to be assigned to execute n jobs, one person per job. (That is, each person is assigned to exactly one job and each job is assigned to exactly one person.) The cost that would accrue if the ith person is assigned to the jth job is a known quantity $C[i, j]$ for each pair i, j = 1, 2, . . . , n. The problem is to find an assignment with the minimum total cost.

25. Define Brute force approach.

Brute force is a straightforward approach to solving a problem, usually directly based on the problem's statement and definitions of the concepts involved. The brute force approach is one that is easiest to apply.

26. What are the advantages of brute force technique?

The various advantages of brute force technique are
1. Brute force applicable to a very wide variety of problems. It is used for many elementary but important algorithmic tasks
2. For some important problems this approach yields reasonable algorithms of at least some practical value with no limitation on instance size
3. The expense to design a more efficient algorithm may be unjustifiable if only a few instances of problems need to be solved and a brute force algorithm can solve those instances with acceptable speed
4. Even if inefficient in general it can still be used for solving small-size instances of a problem

It can serve as a yardstick with which to judge more efficient alternatives for solving a problem

27. Give the benefit of application of brute force technique to solve a problem.

With the application of brute force strategy, the first version of an algorithm obtained can often be improved with a modest amount of effort. So a first application of the brute force approach often results in an algorithm that can be improved with a modest amount of effort.

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

The general plan is as follows
1. A problems instance is divided into several smaller instances of the same problem, ideally about the same size
2. The smaller instances are solved, typically recursively
3. If necessary the solutions obtained are combined to get the solution of the original problem

29. Define merge sort.

Merge sort 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.

30. What is the difference between quick sort and merge sort?

Both quicksort and mergesort use the divide-and-conquer technique in which the given array is partitioned into sub arrays 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.

31. 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].

K
A[0].....A[m-1] A[m] A[m+1]....A[n-1]
search here if KA[m]

32. 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.

33. 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.

UNIT - II
PART -B

1. Discuss the general plan for analyzing the efficiency of non recursive algorithms.
2. Write an algorithm for a given numbers to n generate the nth number of the Fibonacci sequences.
3. Explain the necessary steps for analyzing the efficiency of recursive algorithms.
4. Write short notes on algorithm visualization.
5. Design a recursive algorithm to compute the factorial function F(n) =n! for an arbitrary non negative integer n also derive the recurrence relation.
6. Explain the general plan for mathematical analysis of recursive algorithms with example.
7. Explain algorithm visualization with examples.
8. Write an algorithm to find the number of binary digits in the binary representation of a positive decimal integer and analyze its efficiency
9. Write an algorithm for finding of the largest element in a list of n numbers.
10. Differentiate mathematical analysis of algorithm and empirical analysis of algorithm.
11. Explain static algorithm visualization and dynamic algorithm visualization with example.
12. Explain algorithm animation with example
13. Write a pseudo code for divide & conquer algorithm for merging two sorted arrays in to a single sorted one. Explain with example.
14. Construct a minimum spanning tree using Kruskal’s algorithm with your own example.
15. Explain about Knapsack Problem with example
16. Explain Dijikstra algorithm
17. Define Spanning tree. Discuss design steps in Prim’s algorithm to construct minimum spanning tree with an example.
18. Explain Kruskal’s algorithm.
19. Explain about binary search with example.

UNIT – III
DYNAMMIC PROGRAMMING AND GREEDY TECHNIQUE

1. Define 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.

2. Write down the optimization technique used for Warshall's algorithm. State the rules and assumptions which are implied behind that. [April/May 2015]

Warshall's algorithm constructs the transitive closure of the given diagraph with n vertices through a series of n by n Boolean matrices. This computation is done as follows

$$R^{(0)},....,R^{(k-1)},....,R^{(k)},...,R^{(n)}$$

The optimization technique used for Warshalls's algorithm is based on construction of Boolean matrices. The elements of this matrix can be generated using the formula

$$r_{ij}^{(k)} = r_{ij}^{(k-1)} \text{ or } r_{ik}^{(k\text{-}1)} \text{ and } r_{kj}^{(k-1)}$$

where $r_{ij}^{(k)}$ is the element in i^{th} row and j^{th} column of matrix $R^{(k)}$

3. Write the control abstraction for greedy method.

```
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;
}
```

4. What are the constraints of knapsack problem?

To maximize ∑pixi
1≤i≤n
The constraint is : ∑wixi ≥ m and 0 ≤ xi ≤ 1 1≤ i ≤ n
1≤i≤n
where m is the bag capacity, n is the number of objects and for each object i wi
and pi are the weight and profit of object respectively.

5. Specify the algorithms used for constructing Minimum cost spanning tree.

- Prim's Algorithm
- Kruskal's Algorithm

6. State single source shortest path algorithm (Dijkstra's algorithm).

For a given vertex called the source in a weighted connected graph, find shortest paths to all its other vertices. Dijikstra's algorithm applies to graph with non-negative weights only.

7. 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.

8. What is the Greedy approach?

The method suggests constructing solution through sequence of steps, each expanding partially constructed solution obtained so far, until a complete solution is reached. On each step, the choice must be

- Feasible (satisfy problem constraints)
- Locally optimal (best local choice among all feasible choices available on that step)
- Irrevocable(once made it can't be changed

9. 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.

10. What is meant by principle of optimality? [May/June 2014, Nov/Dec 2016]

The principle of optimality is the basic principle of dynamic programming. It states that an optimal sequence of decisions or choices, each subsequence must also be optimal.

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

Greedy Method	Dynamic Programming
• One sequence of decision is generated. • It guarantee to give an optimal does not solution always	• Many number of decisions are generated. • It definitely gives an optimal solution always.

12. List out the memory functions used under Dynamic Programming. [April/May 2015]

Memory function for binomial coefficient is

$$C(n, k) = C(n-1, k-1) + C(n-1, k)$$

and

$$C(n,0) = C(n, n) = 1$$

Memory function for Warshall algorithm is

$$r_{ij}^{(k)} = r_{ij}^{(k-1)} \text{ or } r_{ik}^{(k\text{-}1)} \text{ and } r_{kj}^{(k-1)}$$

Memory function for Knapsack problem is

$$table\,[i,j\,] \;=\; \max\left\{table\,[i\text{-}1,j], V_i + table\,[i-1, j-W_i]\ if\ j \geq W_i\right\}$$

or

$$table\,[i-1,j]\ if\ j < W$$

13. State the requirement in optimal storage problem in tapes.

Finding a permutation for the n programs so that when they are stored on the tape in this order the MRT is minimized. This problem fits the ordering paradigm.

14. State efficiency of prim's algorithm.

O(|v|2) (weight matrix and priority queue as unordered)

ARRAY) O(|E| LOG|V|) (adjacency list and priority queue as minheap)

15. State Kruskal Algorithm.
The algorithm looks at a MST for a weighted connected graph as an acyclic subgraph with |v|-1 edges for which the sum of edge weights is the smallest.

16. How to calculate the efficiency of Dijkstra's Algorithm? [Nov/Dec 2016]
Time complexity O(|E| log |V |) (Adjacency list and Priority Queue as Minheap)
Space complexity Θ (|V |2) (Weight Matrix and Priority Queue as Unordered array)

16. Differentiate subset paradigm and ordering paradigm

Subset Paradigm	Ordering Paradigm
• At each stage a decision is made regarding whether a particular input is in an optimal solution (generating sub optimal solutions)	• For problems that do not call for selection of optimal subset, in the greedy manner we make decision by considering inputs in some order

17. 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. It is technique for solving problems with overlapping subproblems.

18. What are the features of dynamic programming?
- Optimal solutions to sub problems are retained so as to avoid re-computing of their values.
- Decision sequences containing subsequences that are sub optimal are not considered.
- It definitely gives the optimal solution always.

19. 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.

20. 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.

21. Define principle of optimality.
It states that an optimal solution to any of its instances must be made up of optimal solutions to its sub instances.

22. Define multistage graph
A multistage graph G =(V,E) is a directed graph in which the vertices are partitioned into disjoint sets Vi, where k>=2 and 1<=i<=k. The multi stage graph problem is to find a minimum cost paths from s(source) to t(sink)
Two approach (forward and backward)

23. Define all pair shortest path problem

Given a weighted connected graph, all pair shortest path problem asks to find the lengths of shortest paths from each vertex to all other vertices.

24. Define Distance matrix
Recording the lengths of shortest path in n x n matrix is called Distance matrix (D)

25. Define floyd's algorithm
- To find all pair shortest path
- The algorithm computes the distance matrix of a weighted graph with n vertices through series of n by n matrices :D(0)…D(k-1),D(k)…..D(n)

26. State the time efficiency of floyd's algorithm
Time efficiency of Floyd's algorithm is O(n3). It is cubic

27 Define catalan number.
The total number of binary search trees with n keys is equal to nth catalan
c (n)=(2n to n) 1/(n+1) for n>0,c(0)=1

28. State time and space efficiency of OBST(Optimal Binary Search Tree).
Space efficiency is quadratic and Time efficiency is cubic.

29. What is state space tree? [May/June 2016]
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

30. State Extreme point theorem. [May/June 2016]
Any linear programming problem with a nonempty bounded feasible region has an optimal solution; moreover, an optimal solution can always be found at an extreme point of the problem's feasible region. Extreme point theorem states that if S is convex and compact in a locally convex space, then S is the closed convex hull of its extreme points: In particular, such a set has extreme points. Convex set has its extreme points at the boundary. Extreme points should be the end points of the line connecting any two points of convex set.

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

32. 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.

33. 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.

34. 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.

35. 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;
}
```

36. Define transitive closure.

The transitive closure of a directed graph with 'n' vertices is defined as the n-by-n Boolean matrix T={tij}, in which the elements in the ith row (1 i n) and the jth column (1 j n) is 1 if there exists a non trivial directed path from the ith vertex to the jth vertex otherwise, tij is 0 .

UNIT-III

PART- B

1. Discuss Brute force method with proper example .
2. Develop algorithm for selection sort and bubble sort with proper example .
3. Build a sequential searching algorithm with example .
4. Explain brute force string matching algorithm with example
5. Develop the divide and conquer algorithms with example .
6. Outline merge sort algorithm with example .
7. Plan the quick sort algorithm with example .
8. Explain binary search algorithm with example .
9. Discuss the binary tree traversals and related properties with example.
10. Explain insertion sort with example .
11. Explain Depth First Search and Breadth First Search .
12. Write an algorithm to sort a set of N numbers using insertion sort .
13. Trace the algorithm for the following set of number 20, 35, 18, 8 14, 41, 3, 39.
14. Mention any three search algorithms which is preferred in general? why?
15. Develop algorithm for Warshall’s & Floyd’s to find the optimal solution.
16. Explain about Multistage graphs with example.
17. Develop optimal binary search trees with example.
18. Explain 0/1 knapsack problem with example.
19. Discuss the solution for Travelling salesman problem using branch & bound technique.

UNIT – IV
ITERATIVE IMPROVEMENT

1. Define the iterative improvement technique. [Nov/Dec 2016]

An iterative improvement is a mathematical procedure that generates a sequence of improving approximate solution for a class of problem. Each subsequent solution in such a sequence typically involves a small, localized change in the previous feasible solution. When no such change improves the value of the objective function, the algorithm returns the last feasible solution as optimal and stops.

2. What are the disadvantages of iterative improvement?

- No small change makes an improvement, but some sequence of changes can bring the improvement
- Finding an initial feasible solution is difficult
- It is not always clear what changes should be allowed in a feasible solution

3. What is simplex method?

- It is a technique to solve linear programming problems
- The general problem of optimizing a linear function of several variables subject to a set of linear constrains is linear programming

4. Define slack variable.

Variables transforming inequality constraints into equality constrains

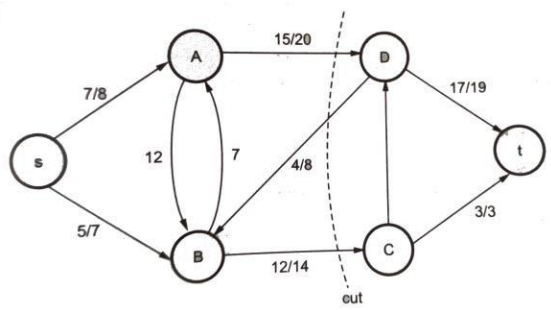
5. What is maximum flow problem?

It involve finding a feasible flow through a single source, single sink flow network that is maximum

6. Define Network Flow and Cut. [April/May 2015, Nov/Dec 2015]

A network flow graph G=(V,E) is a directed graph with two special vertices: the source vertex s, and the sink (destination) vertex t. a flow network (also known as a transportation network) is a directed graph where each edge has a capacity and each edge receives a flow. The amount of flow on an edge cannot exceed the capacity of the edge.

A cut is a collection of arcs such that if they are removed there is no path from *s* to *t*



A cut is said to be minimum in a network whose capacity is minimum over all cuts of the network.

7. Define preflow.

A preflow is a flow that satisfies the capacity constraints but not the flow conservation requirement

8. What is matching in bipartite graph?

A maximum matching or maximum cardinality of matching is a matching with the largest possible number of edges; it is globally optimal.

9. What do you mean by ‘perfect matching’ in bipartite graphs? [April/May 2015]

In a bipartite graph, a perfect matching is a matching in which each node has exactly one edge incident on it. A perfect matching is a graph which matches all vertices of the graph. That is every vertex of the graph is incident to exactly one edge of the matching

10. What is maximum cardinality matching? [Nov/Dec 2016]

A maximum cardinality matching is the largest subset of edges in a graph such that no two edges share the same vertex. For a bipartite graph, it can be found by a sequence of augmentations of previously obtained matchings.

11. Define stable marriage problem

SMP (Stable Marriage Problem) is the problem of finding a stable matching between two sets of elements given a set of preferences for each element

12. How can you tell that a marriage is unstable?

A marriage is unstable if

a. Some given element A of the matched set prefers some given element B of the second matched Set over the element to which A is already matched
b. B also prefers A over the element to which B is already matched

13. State the extreme points

Any LP Problem with a nonempty bounded feasible region has an optimal solution; moreover an optimal solution can always be founded at an extreme point of the problem's feasible region.
This theorem implies that to solve a linear programming problem. at least in the case of a bounded feasible region . We can ignore all but a finite number of points in its feasible region

14. What is linear programming?

The general problem of optimizing a linear function of several variables subject to a set of linear constraints is linear programming

15. Give the Warshall's algorithm.

```
ALGORITHM Warshall(A[1..n,1..n])
//Implements Warshall's algorithm for computing the transitive closure
//Input The adjacency matrix A of a digraph with 'n' vertices
//Output The transitive closure of the digraph
R(0) ⇓ A
for k ⇓ 1 to n do
for i ⇓ 1 to n do
for j ⇓ 1 to n do
R(k)[I,j] ⇓ R(k-1)[I,j] or R(k-1)[I,k] and R(k-1)[k,j]
return R(n)
```

16. Give the Floyd's algorithm

```
ALGORITHM Floyd(W[1..n,1..n])
//Implements Floyd's algorithm for the all-pair shortest–path problem
//Input The weight matrix W of a graph
//Output The distance matrix of the shortest paths' lengths
D ⇓ W
for k ⇓ 1 to n do
for i ⇓ 1 to n do
for j ⇓ 1 to n do
```

```
D[I,j] ⇓ min{D[I,j], D[I,k] + D[k,j]}
return D
```

17. How many binary search trees can be formed with 'n' keys?

The total number of binary search trees with 'n' keys is equal to the nth Catalan number

c(n) = for n >0, c(0) = 1, which grows to infinity as fast as 4n/n1.5.

18. Give the algorithm used to find the optimal binary search tree.

```
ALGORITHM OptimalBST(P[1..n])
//Finds an optimal binary search tree by dynamic programming
//Input An array P[1..n] of search probabilities for a sorted list of 'n' keys
//Output Average number of comparisons in successful searches in the optimal //BST and table R of
subtrees' roots in the optimal BST
for I ⇓ 1 to n do
C[I,I-1] ⇓ 0
C[I,I] ⇓ P[I]
R[I,I] ⇓ I
C[n+1,n] ⇓ 0
for d ⇓ 1 to n-1 do
for i ⇓ 1 to n-d do
j ⇓ i +d
minval ⇓
for k ⇓ I to j do
if C[I,k-1]+C[k+1,j] < minval
minval ⇓ C[I,k-1]+C[k+1,j]; kmin ⇓ k
R[I,j] ⇓ k
Sum ⇓P[I]; for s ⇓ I+1 to j do sum ⇓ sum + P[s]
C[I,j] ⇓ minval+sum
Return C[1,n], R
```

19. List out three possible outcomes in 3 possible outcomes in solving an LP problem.

3 possible outcomes in solving an LP problem are

1. has a finite optimal solution, which may no be unique
2. *unbounded*: the objective function of maximization (minimization) LP problem is unbounded from above (below) on its feasible region
3. *infeasible*: there are no points satisfying all the constraints, i.e. the constraints are contradictory

20. Define flow.

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

21. Define Ford – Fulkerson Method.

- Start with the zero flow ($x_{ij} = 0$ for every edge)

- On each iteration, try to find a *flow-augmenting path* from source to sink, which a path along which some additional flow can be sent
- If a flow-augmenting path is found, adjust the flow along the edges of this path to get a flow of increased value and try again
- If no flow-augmenting path is found, the current flow is maximum

1. How to find flow augumenting path in Network flow problem

To find a flow-augmenting path for a flow x, consider paths from source to sink in the underlying <u>undirected</u> graph in which any two consecutive vertices *i,j* are either:

- connected by a directed edge (*i* to *j*) with some positive unused capacity $r_{ij} = u_{ij} - x_{ij}$
  - known as *forward edge* (→)

OR

- connected by a directed edge (*j* to *i*) with positive flow x_{ji}
  - known as *backward edge* (←)

If a flow-augmenting path is found, the current flow can be increased by *r* units by increasing x_{ij} by *r* on each forward edge and decreasing x_{ji} by *r* on each backward edge, where

r = min {r_{ij} on all forward edges, x_{ji} on all backward edges}

22. Give performance degradation of Ford Fulkerson method.

- The augmenting-path method doesn't prescribe a specific way for generating flow-augmenting paths
- Selecting a bad sequence of augmenting paths could impact the method's efficiency

23. Give Shortest path augmenting algorithm.

- Generate augmenting path with the least number of edges by BFS as follows.
- Starting at the source, perform BFS traversal by marking new (unlabeled) vertices with two labels:
- first label – indicates the amount of additional flow that can be brought from the source to the vertex being labeled
- second label – indicates the vertex from which the vertex being labeled was reached, with "+" or "–" added to the second label to indicate whether the vertex was reached via a forward or backward edge

24. Give the steps for labeling vertices in Shortest path augmenting algorithm

The source is always labeled with ∞,-

All other vertices are labeled as follows:

a. If unlabeled vertex *j* is connected to the front vertex *i* of the traversal queue by a directed edge from *i* to *j* with positive unused capacity $r_{ij} = u_{ij} - x_{ij}$ (forward edge), vertex *j* is labeled with l_j, i^+, where $l_j = \min\{l_i, r_{ij}\}$

b. If unlabeled vertex *j* is connected to the front vertex *i* of the traversal queue by a directed edge from *j* to *i* with positive flow x_{ji} (backward edge), vertex *j* is labeled l_j, i^-, where $l_j = \min\{l_i, x_{ji}\}$

25. Define Cut and its capacity.

- 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.

UNIT-IV

PART- B

1. Discuss Transform and conquer techniques with example .
2. Explain pre sorting and balanced search tree with example .
3. Develop Binary Search Tree with example .
4. Outline the AVL Trees with example .
5. Explain Warshall’s and Floyd’salgorithm with example.
6. Explain Optimal Binary search trees with example .
7. Discuss the Prim’s algorithm with example .
8. Explain Kruskal’s and Dijkstra’s Algorithm with example .
9. Develop an algorithm for Huffman Trees and find the maximum number of bits/char.
10. Explain the method of finding the minimum spanning tree for connected graph using prim’;s algorithm.
11. How will you find the shortest path between two given vertices using Dijkkstra’s Algorithm? Explain.
12. Explain the 8-Queen’s problem & discuss the possible solutions.
13. Solve the following instance of the knapsack problem by the branch & bound algorithm.
14. Apply backtracking technique to solve the following instance of subset sum problem: S={1,3,4,5} and d=11 .
15. Explain subset sum problem & discuss the possible solution strategies using backtracking
16. Explain Graph coloring with example.
17. Explain about Knapsack Problem using back tracking with example.

UNIT – V
COPING WITH THE LIMITATIONS OF ALGORITHM POWER

1. What is a Biconnected Graph? [April /May 2010]

A **biconnected** undirected graph is a connected graph that is not broken into disconnected pieces by deleting any single vertex (and its incident edges). A **biconnected** directed graph is one such that for any two vertices *v* and *w* there are two directed paths from *v* to *w* which have no vertices in common other than *v* and *w*.

2. Define Branch-and-Bound method. What are the searching techniques that are commonly used in Branch-and-Bound?

The term Branch-and-Bound refers to all the state space methods in which all children of the E-node are generated before any other live node can become the E- node. The searching techniques that are commonly used in Branch-and-Bound method are:

- FIFO
- LIFO
- LC
- Heuristic search

3. What are NP- hard and Np-complete problems?

The problems whose solutions have computing times are bounded by polynomials of small degree.

4. What is Hamiltonian cycle in an undirected graph?

A Hamiltonian cycle is round trip along n edges of G that visits every vertex once and returns to its starting position.

5. What is a decision problem?

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

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

7. Define the theory of computational complexity.
A problem's intractability remains the same for all principal models of computations and all reasonable input encoding schemes for the problem under consideration.

8. How NP-Hard problems are different form NP-Complete? [April/May 2015]

- The NP complete has a property that it can be solved in polynomial time if and only if all other NP complete problems can also be solved in polynomial time.
- If an NP hard problem can be solved in polynomial time then all the NP complete problems can be solved in polynomial time.
- All the NP complete problems are NP hard but there are some NP hard problems that are not known to be NP complete.

9. Define Hamiltonian circuit problem. [April/May 2015]
Hamiltonian circuit problem is a problem of finding a Hamiltonian circuit. Hamiltonian circuit is a circuit that visits every vertex exactly once and return to the starting vertex.

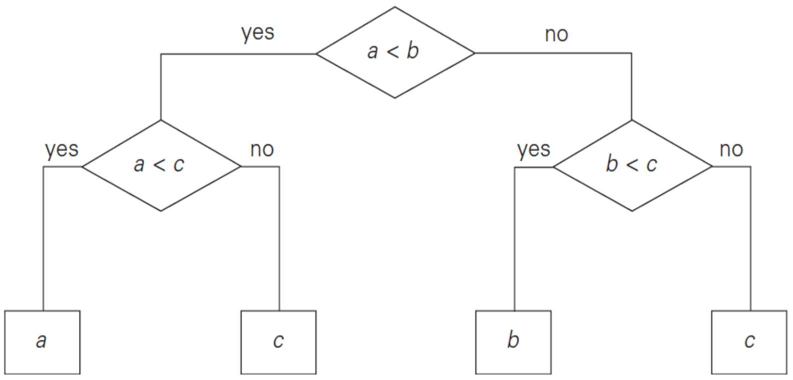
10. Given an application for knapsack problem.
The knapsack problem is problem in combinatorial optimization. It derives its name from the maximum problem of choosing possible essential that can fit into one bag to be carried on a trip. A similar problem very often appears in business, combinatory, complexity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

11. Give the purpose of lower bound. [May/June 2016]

- The elements are compared using operator < to make selection.
- Branch and bound is an algorithm design technique that uses lower bound comparisons.
- Main purpose is to select the best lower bound.
- Example: Assignment problem and transportation problem.

12. What is The Euclidean minimum spanning tree problem? [May/June 2016]
The Euclidean minimum spanning tree or EMST is a minimum spanning tree of a set of n points in the plane where the weight of the edge between each pair of points is the Euclidean distance between those two points. In simpler terms, an EMST connects a set of dots using lines such that the total length of all the lines is minimized and any dot can be reached from any other by following the lines.

13. Draw the decision tree for comparison of three values. [Nov/Dec 2015]



14. Write the formula for decision tree for searching a sorted array. [Nov/Dec 2016]

$$C^{bs}_{worst}(n) = \lfloor \log_2 n \rfloor + 1 = \lceil \log_2(n+1) \rceil$$

15. State the reason for terminating search path at the current node in branch and bound algorithm. [Nov/Dec 2016]

- The value of the nodes bound is not better than the value of the best solution seen so far.
- 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 – here we 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.

16. Define Traversals.

When the search necessarily involves the examination of every vertex in the object being searched it is called a traversal.

17. List out the techniques for traversals in graph.

- Breadth first search
- Depth first search

18. What is articulation point?

A vertex v in a connected graph G is an articulation point if and only if the deletion of vertex v together with all edged incident to v disconnects the graph in to two or more nonempty components.

19. Define Heap sort.

Heap sort is a sorting algorithm. Heap sort is a 2 stage algorithm. The two stages are
Stage 1- Heap Construction- Construct a heap for a given array of elements
Stage 2- Maximum Deletion- Apply the root deletion operation n-1 times to the remaining heap.

20. What is backtracking?

Backtracking constructs solutions one component at a time and such partially constructed solutions are evaluated as follows.
1) 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.
2) If there is no legitimate option for the next component, no alternatives for the 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. What is n-queens 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 column or in the same diagonal.

27. 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.

28. What is the subset-sum problem?

Find a subset of a given set S={s1,. ....,sn} of 'n' positive integers whose sum is equal to a given positive integer 'd'.

29. When can a node be terminated in the subset-sum problem?

The sum of the numbers included are added and given as the value for the root as s'. The node can be terminated as a non-promising node if either of the two equalities holds:
s'+si+1d (the sum s' is too large)
s'+ sjd (the sum s' is too small)

30. How does the output of a backtracking algorithm appear?

The output of a backtracking algorithm can be thought of as an n-tuple (x1,..xn) where each coordinate xi is an element of some finite linearly ordered set Si. If such a tuple (x1,. xi) is not a solution, the algorithm finds the next element in Si+1 that is consistent with the values of (x1,. . xi) and the problem's constraints and adds it to the tuple as its (I+1)st coordinate. If such an element does not exist, the algorithm backtracks to consider the next value of xi, and so on.

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

The various tricks are

a) 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.

b) Reassign values to one or more components of a solution

c) rearranging the data of a given instance.

32. What is the method used to find the solution in n-queen problem by symmetry?

The board of the n-queens problem has several symmetries so that some solutions can be obtained by other reflections. Placements in the last n/2 columns need not be considered, because any solution with the first queen in square (1,i), n/2 i n can be obtained by reflection from a solution with the first queen in square (1,n-i+1)

33. 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.

34. 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.

35. Compare backtracking and branch-and-bound.

Backtracking Branch-and-bound: State-space tree is constructed using depth-first search State-space tree is constructed using best-first search. Finds solutions for combinatorial non-optimization problems Finds solutions for combinatorial optimization problems. 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

36. 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.

37. 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.

38. What is knapsack problem?

Given n items of known weights wi and values vi, i=1,2,â€¦,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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. 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.

48. What are the factors that influence the efficiency of the backtracking algorithm?

The efficiency of the backtracking algorithm depends on the following four factors. They are:

i. The time needed to generate the next xk
ii. The number of xk satisfying the explicit constraints.
iii. The time for the bounding functions Bk
iv. The number of xk satisfying the Bk.

UNIT-V

1. Discuss backtracking with example.

2. Solve n-Queen’s problem with example .
3. Build Hamiltonian circuit problem with proper example .
4. Develop subset problem with proper example.
5. Explain Branch and bound Techniques with proper example .
6. Solve Knapsack problem with example .
7. Explain travelling salesman problem with example .
8. Explain strassen’s matrix multiplication problem with example .
9. Discuss the features of travelling salesman problem .
10. Given a graph G={V,E,W} shown in the figure where vertices refer to cities, edges refer to connection between cities, weight associated with each edge represents the cost. Find out the minimum cost for the salesman .
11. Discuss NP completeness in Knapsack problem with justification .
12. Apply backtracking technique to solve the following instance of the subset sum problem S={1,3,4,5} and d=11 .
13. Explain subset problem and discuss the possible solution strategies using backtracking.
14. Give a suitable example & explain the Breadth first search & Depth first search. (16)
15. Explain about biconnected components with example.
16. Briefly explain NP-Hard and NP-Completeness with examples.
17. Explain about 0/1 Knapsack Problem using branch and bound with example.

Staff Incharge **Dean(CSE)**