UNIT I
INTRODUCTION

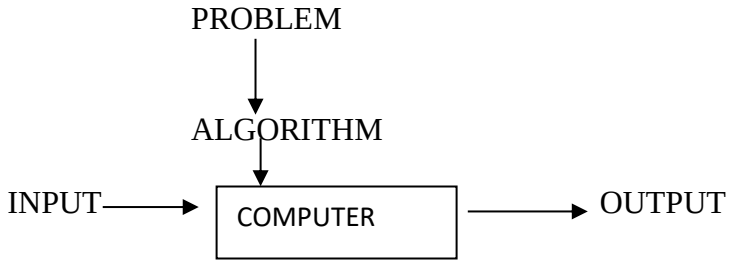
Algorithm analysis: Time and space complexity - Asymptotic Notations and its properties Best case, Worst case and average case analysis – Recurrence relation: substitution method - Lower bounds – **searching:** linear search, binary search and Interpolation Search, **Pattern search:** The naïve string- matching algorithm - Rabin-Karp algorithm - Knuth-Morris-Pratt algorithm. **Sorting:** Insertion sort – heap sort

PART - A

1. **What do you mean by algorithm? (Nov/Dec 2008) (May/June 2013) (Apr/May 17) (U)**
   An algorithm is a sequence of unambiguous for solving a problem i.e., for obtaining a required output for any legitimate input in a finite amount of time. In addition, all algorithms must satisfy the following criteria:
   1) Input
   2) Output
   3) Definiteness
   4) Finiteness
   5) Effectiveness.

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

3. **Give the diagram representation of Notion of algorithm. (C)**



4. **What are the types of algorithm efficiencies? (R)**
   The two types of algorithm efficiencies are
   Time efficiency: indicates how fast the algorithm runs.
   Space efficiency: indicates how much extra memory the algorithm needs.

5. **What is space complexity? (Nov/Dec 2012) (R)**
   Space Complexity indicates how much extra memory the algorithm needs. Basicallyit has three components. They are instruction space, data space and environment space.

6. **What is time complexity? (Nov/Dec 2012) (R)**
   Time Complexity indicates how fast an algorithm runs. T (P) =Compile Time + Run Time.(Tp) ,Where Tp is no of add, sub, mul...

7. **What is an algorithm design technique? (R)**
   An algorithm design technique is a general approach to solving problems algorithmically that is applicable to a variety of problems from different areas of computing.

8. **What is pseudo code? (R)**
   A pseudo code is a mixture of a natural language and programming language constructs to specify an algorithm. A pseudo code is more precise than a natural language and its usage often yields more concise algorithm descriptions.

9. **What are the types of algorithm efficiencies? (R)**
   The two types of algorithm efficiencies are
   Time efficiency: indicates how fast the algorithm runs
   Space efficiency: indicates how much extra memory the algorithm needs.

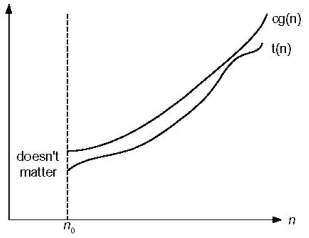
10. **What do you mean by "worst-case efficiency" of and algorithm? (R) (Nov 17)**
    The worst case efficiency of an algorithm, its efficiency for the worst-case input of size n, which is an input or inputs of size n for which the algorithm runs the longest among all possible inputs of that size.

11. **What is best-case efficiency? (R)**
    The best-case efficiency of an algorithm is its efficiency for the best-case input of size n, which is an input or inputs for which the algorithm runs the fastest among all possible inputs of that size.

12. **What is average case efficiency? (May 2008) (R)**
    The average case efficiency of an algorithm is its efficiency for an averagecase input of size n. It provides information about an algorithm behavior on a "typical" or "random" input.

13. **Define asymptotic notations. (R)**
    To choose best algorithm, we need to check efficiency of each algorithm the efficiency can be measured by computing time complexity of each algorithm. Asymptotic notation is shorthand way to represent the time complexity
    The various notations are Big "Oh", Big Omega, and Theta.

14. **Define the asymptotic notation "Big oh" (0)(May 2008)(April/May 2012&2013) (R)**
    Let, f(n) and g(n) be two non-negative functions. Let, n0 and constant c are two integers such that no denotes some value of input similarly c is constant such that c > 0. We can write



F(n) <= c*g(n) For all n ≥ n0

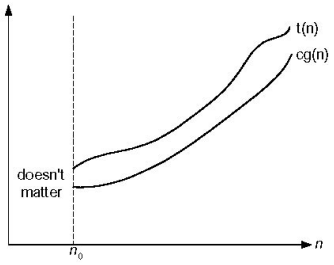
A function t(n) is said to be in O(g(n)) denoted t(n) ε O (g(n)), if t(n) is bounded above by some constant multiple of g(n) for all large n, i.e., if there exist some positive constant c and some non-negative integer n0 such that

T (n) < c g (n) for n > n0

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

A function f(n) is said to be in Ω (g(n)) if f(n) is bounded below by some positive constant multiple of g(n) such that

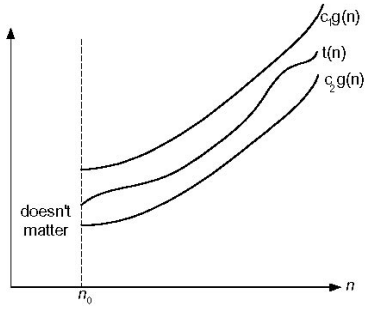
f(n) ≥ c * g(n) For all n ≥ n0



16. Define the asymptotic notation "theta" (θ) (R)

Let, f(n) and g(n) be two non negative functions. There are two positive constants namely c1 and c2 such that c1 ≤ g(n) ≤ c2 g(n)

Then we can say that, f(n) ε Θ (g(n))



17. What is recursive algorithm? (R)

An algorithm is said to be recursive if the same algorithm is invoked in the body. An algorithm that calls itself is direct recursive.

18. Define recurrence. (May2010) (R)

A recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs. The complexity of many interesting algorithms is easily expressed as a recurrence, especially divide and conquer algorithms. The complexity of recursive algorithms is readily expressed as a recurrence.

Example :(i)Linear search of a list

$$T(n) = \begin{cases} 1 & \text{for } n \le 1 \\ T(n-1) + 1 & \text{otherwise} \end{cases}$$

(ii) Binary search

$$T(n) = \begin{cases} 1 & \text{for } n \le 1 \\ T(n/2) + 1 & \text{otherwise} \end{cases}$$

19. Define Linear Search. (Nov/Dec 2011) (April/May 2010&2012) (R)

In computer science, linear search or sequential search is a method for finding aparticular value in a list, which consists in checking every one of its elements, one at atime and in sequence, until the desired one is found.

20. What are the Best, Worst and Average Case complexity of Linear Search? (R)

- ✓ Best Case – O(1)
- ✓ Average Case – O(N)
- ✓ Worst Case – O(N)

21. Difference between Best Case and Worst Case Complexities.(AN)

The best case complexity of an algorithm is its efficiency for the best case input ofsize N, which is an input of size N for which the algorithm runs fastest among all possible inputs of that size.

The worst case complexity of an algorithm is its efficiency for the worst case inputof size N, which is an input of size N for which the algorithm runs longest among all possible inputs of that size.

22. What is binary search? (R)

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].

a. A[0]……A[m-1] A[m] A[m+1]……A[n-1]

23. Give computing time for Binary search?(APRIL/MAY 2012) (E)

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

Best-(1) average-(logn) worst -(Logn)

unsuccessful searches best, average, worst- (logn)

24. Write the algorithm for Iterative binary search? (A)

```
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;
}
```

25. Give the general plan for analyzing non recursive algorithm. (R)

- Decide a parameter indicating an Input's Size.
- Identify the algorithms Basic Operation
- Check whether the number of times the basic operation is executed only on thesize of an input. If it also depends on some additional property, the worst case, average case, and if necessary, best case efficiencies have to be investigated separately.
- Using standard formulas and rules of sum manipulation either find a closedformula for the count or, at the very least, establish its order of growth.

26. What is validation of algorithm? (Nov/Dec 2012) (R)

The process of measuring the effectiveness of an algorithm before it is coded to know the algorithm is correct for every possible input. This process is called validation.

27. What are all the methods available for solving recurrence relations? (R)

- ✓ Forward Substitution
- ✓ Backward Substitution
- ✓ Smoothness Rule
- ✓ Master Theorem

28. Write down the problem types.(April/May 2008) (R)

- ✓ Sorting
- ✓ Searching
- ✓ String Processing
- ✓ Graph Problem
- ✓ Combinational Problem
- ✓ Geometric Problem
- ✓ Numerical Problem

29. What are the types of recurrence relations? (R)

- ✓ Homogeneous recurrence relation.
- ✓ Non homogeneous recurrence relation.

30. Define Substitution Method. (April /May 2010) (R)

Substitution method is a method of solving a system of equations wherein one of theequations is solved for one variable in terms of the other variables.

31. Give the general plan for analyzing recursive algorithm. (R)

- ✓ Decide a parameter indicating an Input's Size.
- ✓ Identify the algorithms Basic Operation

- ✓ Check whether the number of times the basic operation is executed only on thesize of an input. If it also depends on some additional property, the worst case,average case, and if necessary, best case efficiencies have to be investigatedseparately.
- ✓ Set up a recurrence relation, with an appropriate initial condition, for thenumber of times basic operation is executed.

32. Define Recurrence Relation. (APRIL/MAY 2010) (Nov/Dec 2016) (R)

The equation defines M (N) not explicitly, ie.., is a function of n, but implicitly as afunction of its value at another point, namely N-1. Such equations are calledrecurrence relation.

33. Give the General Plan for the Empirical Analysis of Algorithm Time Efficiency? (C)

1. Understand the experiment's purpose.
2. Decide on the efficiency metric M to be measured and the measurement unit (an operation count vs. a time unit).
3. Decide on characteristics of the input sample (its range, size, and so on).
4. Prepare a program implementing the algorithm (or algorithms) for the experimentation.
5. Generate a sample of inputs.
6. Run the algorithm (or algorithms) on the sample's inputs and record the data observed.
7. Analyze the data obtained.

34. Write down the properties of asymptotic notations?(MAY 2015) (R)

If t1(n)Ɛ O(g1(n)) and t2(n)Ɛ O(g2(n)), then
t1(n)+t2(n)Ɛ O(max{g1(n),g2(n)})

35. Give the Euclid algorithm for computing gcd(m, n) (MAY/JUN 2016) (Apr/May 17) (APR 17) (C)

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 $n \neq 0$ **do**
$r \leftarrow m \bmod n$
$m \leftarrow n$
$n \leftarrow r$
return *m*
Example: gcd*(*60, 24*)* = gcd*(*24, 12*)* = gcd*(*12, 0*)* = 12.

36. Design an algorithm for computing area and circumference of the circle. (NOV/DEC 2016) (C)

//Computes area and circumference of the circle
//Input: One non-negative integer radius
//Output: Area and Circumference of the circle

```
Area = PI * rad * rad;
Ci = 2 * PI * rad;
return area, ci
```

37. How to measure an algorithm running time? (NOV/DEC 2017) (R)

One possible approach is to count the number of times each of the algorithm's operation is executed. The thing to do is identify the most important operation of the algorithm called as basic operation and compute the number of times the basic operation is executed.

$$T(n) \cong Cop\ C(n)$$

38. What is a basic operation? (APR/MAY 2018) (R)

The process of identify the most important operation of the algorithm called as basic operation

39. Define best,worst,average case time complexity. (NOV/DEC 2018)

Best case: In the best case analysis, we calculate lower bound on running time of an algorithm. We must know the case that causes minimum number of operations to be executed.

Worst case: In the worst case analysis, we calculate upper bound on running time of an algorithm. We must know the case that causes maximum number of operations to be executed.

Averge case: In average case analysis, we take all possible inputs and calculate computing time for all of the inputs. Sum all the calculated values and divide the sum by total number of inputs.

40. How do you measure the efficiency of an algorithm. (APR/MAY 2019)

Time **efficiency** - a **measure** of amount of time for an **algorithm** to execute.

Space **efficiency** - a **measure** of the amount of memory needed for an **algorithm** to execute.

41. Prove that the if f(n)=O(g(n)) and g(n)=O(f(n)),then f(n)= θg (n) (APR/MAY 2019)

42. What do you mean bt interpolation search?

Interpolation search finds a particular item by computing the probe position. Initially, the probe position is the position of the middle most item of the collection.

If a match occurs, then the index of the item is returned. To split the list into two parts, we use the following method −

mid = Lo + ((Hi - Lo) / (A[Hi] - A[Lo])) * (X - A[Lo])

where −

A = list
Lo = Lowest index of the list
Hi = Highest index of the list

A[n] = Value stored at index n in the list

43. Define naïve string matching alogorithm.

The naïve approach tests all the possible placement of Pattern P [1.....m] relative to text T [1.....n]. We try shift s = 0, 1.....n-m, successively and for each shift s. Compare T [s+1.....s+m] to P [1.....m].

The naïve algorithm finds all valid shifts using a loop that checks the condition P [1.....m] = T [s+1.....s+m] for each of the n - m +1 possible value of s.

NAIVE-STRING-MATCHER (T, P)

```
1. n ← length [T]
2. m ← length [P]
3. for s ← 0 to n -m
4. do if P [1.....m] = T [s + 1....s + m]
5. then print "Pattern occurs with shift" s
```

44. Define Rabin-Karp algorithm.

The Rabin-Karp string matching algorithm calculates a hash value for the pattern, as well as for each M-character subsequences of text to be compared. If the hash values are unequal, the algorithm will determine the hash value for next M-character sequence. If the hash values are equal, the algorithm will analyze the pattern and the M-character sequence. In this way, there is only one comparison per text subsequence, and character matching is only required when the hash values match.

RABIN-KARP-MATCHER (T, P, d, q)

1. n ← length [T]
2. m ← length [P]
3. h ← d^{m-1} mod q
4. p ← 0
5. t_0 ← 0
6. for i ← 1 to m
7. do p ← (dp + P[i]) mod q
8. t_0 ← (dt_0+T [i]) mod q
9. for s ← 0 to n-m
10. do if p = t_s
11. then if P [1.....m] = T [s+1.....s + m]
12. then "Pattern occurs with shift" s
13. If s < n-m
14. then t_{s+1} ← (d (t_s-T [s+1]h)+T [s+m+1])mod q

45. Define Knuth-Morris-Pratt algorithm.

Knuth-Morris and Pratt introduce a linear time algorithm for the string matching problem. A matching time of O (n) is achieved by avoiding comparison with an element of 'S' that have previously been involved in comparison with some element of the pattern 'p' to be matched. i.e., backtracking on the string 'S' never occurs

46. Write the advantage of insertion sort? (NOV/DEC 2017)

1.Simple implementation
2.Efficient for small data sets
3.Adaptive
4.More efficient in practice than most other simple quadratic, i.e. O(n2) algorithms such as Selection sort or bubble sort; the best case (nearly sorted input) is O(n)
5.Stable - does not change the relative order of elements with equal keys

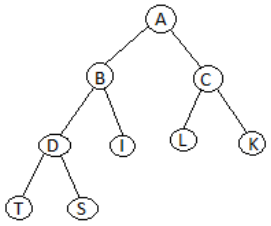
47. Define Heap sort?

Heap Sort is one of the best sorting methods being in-place and with no quadratic worst-case running time. Heap sort involves building a **Heap** data structure from the given array and then utilizing the Heap to sort the array.

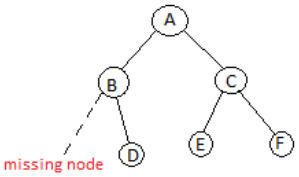
48. What are the properties of heap structure?

The special heap properties given below:

Shape Property: Heap data structure is always a Complete Binary Tree, which means all levels of the tree are fully filled.

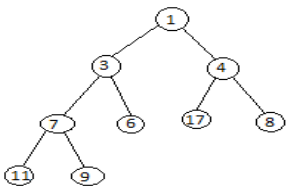


Complete Binary Tree



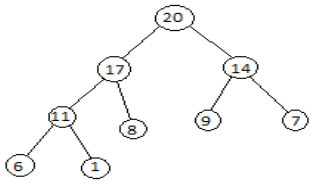
In-Complete Binary Tree

Heap Property: All nodes are either **greater than or equal to** or **less than or equal to** each of its children. If the parent nodes are greater than their child nodes, heap is called a **Max-Heap**, and if the parent nodes are smaller than their child nodes, heap is called **Min-Heap**.



Min-Heap

In min-heap, first element is the smallest. So when we want to sort a list in ascending order, we create a Min-heap from that list, and picks the first element, as it is the smallest, then we repeat the process with remaining elements.



Max-Heap

In max-heap, the first element is the largest, hence it is used when we need to sort a list in descending order.

49. What are the two basic parts of Heap sort ?

- Creating a Heap of the unsorted list/array.
- Then a sorted array is created by repeatedly removing the largest/smallest element from the heap, and inserting it into the array. The heap is reconstructed after each removal.

50. What is the Complexity Analysis of Heap Sort?

Worst Case Time Complexity: **O(n*log n)**
Best Case Time Complexity: **O(n*log n)**
Average Time Complexity: **O(n*log n)**
Space Complexity : **O(1)**

51. What are the advantages of Heap sort ?

Heap sort is not a Stable sort, and requires a constant space for sorting a list.
Heap Sort is very fast and is widely used for sorting

PART B

1. Define the asymptotic notations used for best case average case and worst case analysis? **(APIRL/MAY2009) (APRIL/MAY-2008)(R) (Apr 18)**
2. How do you evaluate the performance of the algorithms? **(E)**
3. Explain properties of BIG (oh) Notation.(8) **(MAY 2016) (R) (Nov 17)**
4. What is meant by recurrence? Give one example to solve recurrence equations. **(APRIL/MAY 2012) (r)**
5. (i) Distinguish between Big Oh, Theta and Omega natation. **(NOV/DEC 2012) (AN)**
(ii) Analyze the best, worst and average case analysis for linear search.

6. Find complexity of algorithm C (n) of the algorithm for the best, worst, average case,(evaluate average case complexity of n=3 n mean number of inputs.) **(E)**
7. (i) Define Asymptotic notations. Distinguish between Asymptotic notation and Conditional asymptotic notation. (10)**(APRIL/MAY 2010)(APRIL/MAY 2011) (Apr/May 17)**
ii) Explain how the removing condition is done from the conditional asymptotic notation with an example. (6)**(NOV/DEC 2011) (R)**

8. (i) Explain how analysis of linear search is done with a suitable illustration. (10)
(ii) Define recurrence equation and explain how solving recurrence equations are done.(6) **(NOV/DEC 2011) (R)**

9. Explain how Time Complexity is calculated. Give an example. **(APRIL/MAY 2010) (E)**

10. Discuss all the asymptotic notations in detail. **(APRIL/MAY 2012)(R)**

11. Write an algorithm for finding maximum element of an array, perform best, worst and average case complexity with appropriate order notations. **(APRIL/MAY 2008) (R)**

12. Write an algorithm to find mean and variance of an array perform best, worst and average case complexity, defining the notations used for each type of analysis. **(APRIL/MAY 2008) (AN)**

13. (i)Briefly explain the time complexity, space complexity estimation.(6) **(MAY/JUNE 2013)**

(ii) Write the linear search algorithm and analyze its time complexity.(10) **(Nov/Dec 2016)**

14. (i)Find the time complexity and space complexity of the following problems. Factorial using recursion and Compute n^{th} Fibonacci number using Iterative statements. **(C)**

(ii) Solve the following recurrence relations: **(NOV/DEC 2012) (A)**

$$1).T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + 3n & n > 2 \\ 2 & n = 2 \end{cases}$$

$$2)T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + cn & n > 1 \\ a & n = 1 \end{cases}$$ Where a and c are constants.

15. Solve the following inequalities are correct **(MAY/JUNE 2013) (A)**

(i)$5n^2 - 6n = \emptyset(n^2)$
(ii) $n!=O(n^n)$
(iii)$n^3 + 10n^2 = \Theta(n^3)$
(iv)$2n^2 2^n + nlogn = \Theta(n^2 2^n)$

16. If you have to solve the searching problem for a list of n numbers, how can you take advantage of the fact that the list is known to be sorted? Give separate answer for

(i) List represented as arrays
(ii) List represented as linked lists
Compare the time complexities involved in the analysis of both the algorithms.**(MAY 2015) (AN)**

17. (i) Derive the worst case analysis of merge sort using suitable illustrations.(8)**(MAY 2015)**

(ii) Derive a loose bound on the following equation (8) **(A)**

$F(x) = 35x^8 - 22x^7 + 14x^5 - 2x^4 - 4x^2 + x - 15$

18. Give an algorithm to check whether all the Elements in a given array of n elements are distinct. Find the worst case complexity of the same.(8) **(MAY / JUNE 2016) (A)**

19. Give the recursive algorithm which finds the number of binary digits in the binary representation of a positive decimal integer. Find the recurrence relation and complexity. **(MAY / JUNE 2016) (A)**

20. State the general plan for analyzing the time efficiency of non-recursive algorithm and explain with an example. (8) **(Nov/Dec 2016) (AN)**

21. Solve the following recurrence relation. **(A)**

- $x(n)=x(n-1)+5$ for $n>1$ $x(1)=0$
- $x(n)=3x(n-1)$ for $n>1$ $x(1)=4$
- $x(n)=x(n-1)+n$ for $n>1$ $x(0)=0$
- $x(n)=x(n/2)+n$ for $n>1$ $x(1)=1$ (solve for $n=2^k$)
- $x(n)=x(n/3)+1$ for $n>1$ $x(1)=1$ (solve for $n=3^k$) (16) **(Nov/Dec 2016)**

22. Briefly explain the mathematical analysis of recursive and non-recursive algorithm. (8) **(Apr/May 2017) (R)**

23. Discuss the steps in mathematical analysis for recursive algorithms. Do the same for finding the factorial of the number. **(NOV/DEC 2017)**

24. Give the general plan for analyzing the time efficiency of recursive algorithm and use recurrence to find number of moves Towers of Hanoi problem. **(APR/MAY 18)**

25. Consider the problem of finding the smallest and largest elements in an array of n numbers. **(APR/MAY 18)**

(i) Design a presorting- based algorithm for solving this problem and determine its efficiency class. (7)

(ii) Compare the efficiency of the three algorithms: (8)

(A) the brute force algorithm (B) this presorting based algorithm (C) the divide and conquer algorithm.

26. (i)Prove that if g(n) is $\Omega(f(n))$ then f(n) is O(g(n)). (5) **(NOV/DEC 2018)**

(ii) Discuss vrious methods used for mathematical analysis of recursive algorithms.(8) **(NOV/DEC 2018)**

27. Write the asymptotic notations used for best case,average case and worst case analysi of lgorithms. Writa an algorithm for finding maximum elements in an array. Give best,worst and average case complexities. **(NOV/DEC 2018)**

28. Solve the following recurrence relation: **(APR/MAY 2019)**

(1) $T(n)=T(n/2)+1$, where $n=2^k$ for all $k \geq 0$ (4)

(2) $T(n)=T(n/3)+T(2n/3)+cn$, where 'c' is a constant and 'n' is the input size.

29. Explains the steps involved in problem solving. **(APR/MAY 2019)**

30. What do you mean bt interpolation search?

31. Explain in detail about naïve string matching alogorithm.
32. Explain in detail about Rabin-Karp algorithm.
33. Explain in detail about Knuth-Morris-Pratt algorithm.
34. Explain in detail about insertion sort? (NOV/DEC 2017)

UNIT II

GRAPH ALGORITHMS

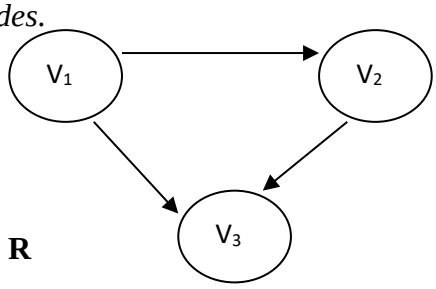
Graph algorithms: Representations of graphs - Graph traversal: DFS – BFS - applications - Connectivity, strong connectivity, bi-connectivity - Minimum spanning tree: Kruskal's and Prim's algorithm- Shortest path: Bellman-Ford algorithm - Dijkstra's algorithm - Floyd-Warshall algorithm Network flow: Flow networks - Ford-Fulkerson method – Matching: Maximum bipartite matching

1. Define a graph. (April/May 2007) - U

A graph G=(V, E) consists of a set of vertices(V) and set of edges(E).

In general, A *graph* is a collection of *nodes* (also called *vertices*) and *edges* (also called *arcs* or *links*) each connecting a pair of *nodes*.

Example:

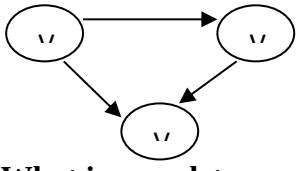


2. What are the different types of Graph? - R

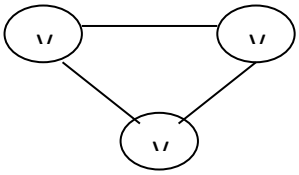
There are two types of Graphs. They are:

1. Directed Graphs.
2. Undirected Graphs.

Example for Directed Graph:



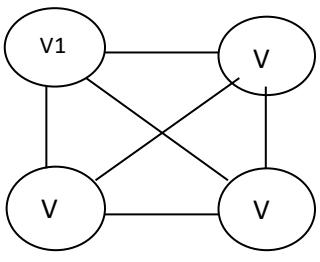
Example for undirected Graph:



3. What is complete graph? - U

If an undirected graph of n vertices consists of n(n-1)/2 number of edges it is called as complete graph.

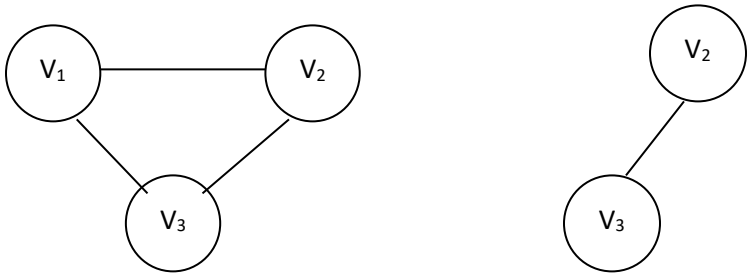
Example:



4. What is sub graph? - U

A subgraph G' of graph G is a graph such that the set of vertices and set of edges of G' are proper subset of the set of edges of G.

Example:

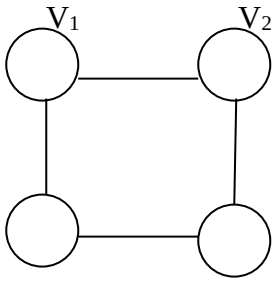


G Grapth **G' Sub Graph**

5. What is connected graph? - U

An directed graph is said to be connected if for every pair of distinct vertices V_i and V_j in V(G) there is a graph from V_i to Vj in G.

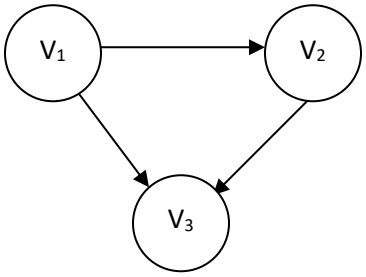
Example:



V3 V_4

6. What are directed graphs? - U

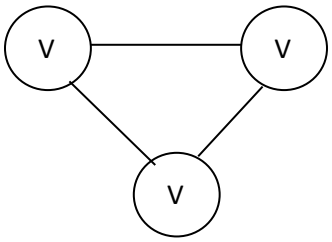
Directed graph is a graph which consists of directed edges, where each edge in E is unidirectional. It is also referred as Digraph. If (v,w) is a directed edge then (v,w) ≠ (w,v



$(v_1,v_2) \neq (v_2,v_1)$

7. What are undirected graphs? - U

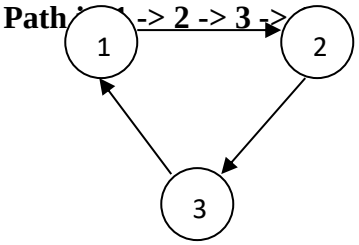
An undirected graph is a graph, which consists of undirected edges. If (v,w) is an undirected edge then (v,w) = (w,v)



$(v_1,v_2) = (v_2,v_1)$

8. Define cycle. - U

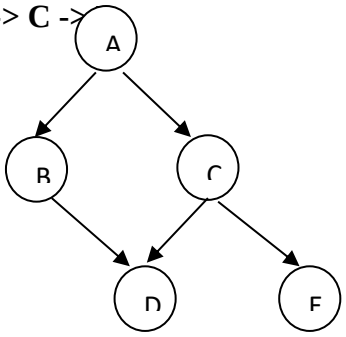
A cycle in a graph is a path in which first and last vertex are the same.



A graph which has a cycle is referred to as cyclic graph.

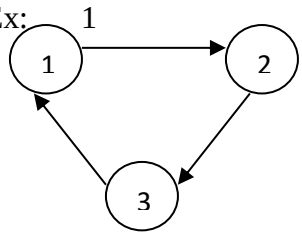
9. Define Acyclic graph. - U

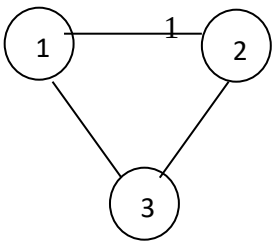
A directed graph is said to be acyclic when there is no cycle path in it. It is also called DAG(Directed Acyclic Graph).



10. Define weighted graph. - U

A graph is said to be weighted graph if every edge in the graph is assigned a weight or value. It can be directed or undirected graph.



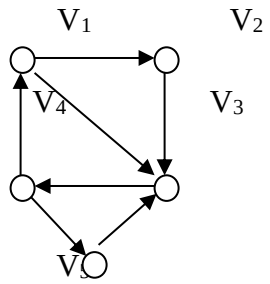


2 1 2 1

11. Define strongly connected graph. - U

A directed graph is said to be strongly connected, if for every pair of vertex, there exists a path

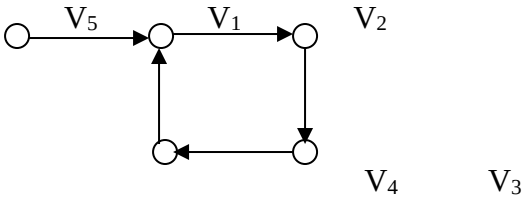
Diagram:



12. What is weakly connected graph. (May/June 2012) - U

A digraph is weakly connected if all the vertices are connected to each other.

Diagram:



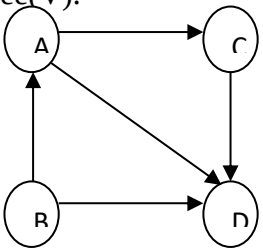
13. Define path in a graph. - U

A **path** in a graph is a sequence of vertices such that from each of its vertices there is an edge to the next vertex in the sequence. A path may be infinite, but a finite path always has a first vertex, called its *start vertex*, and a last vertex, called its *end vertex*. Both of them are called *end or terminal vertices* of the path. The other vertices in the path are *internal vertices*. A **cycle** is a path such that the start vertex and end vertex are the same.

14. What is the degree of a graph? - U

The number of edges incident on a vertex determines its degree. The degree of the vertex V is written as degree(V).

Example:



Degree (A) = 0

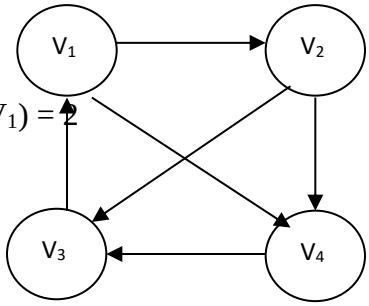
Degree (C) = 1

Degree (D) = 3

15. Define indegree, outdegree in a graph. (April/May 2010) - U

The **indegree** is the numbers of edges entering in to the vertex V.

The **out degree** is the numbers of edges that are exiting or leaving from the vertex V.



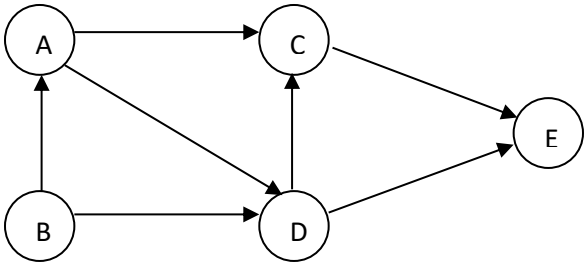
Outdegree (V_1) = 2

Indegree(V_1) = 1

16. What is adjacent node? - U

Adjacency node is the node which is connected to the other nodes in the graph.

Example:



In the above diagram, C is an adjacent node to E

17. What are the applications of graphs? - R

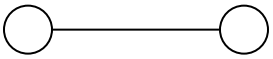
The various applications of graphs are:

- In computer networking such LAN, WAN etc., internetworking of computer systems is done using graph concept.
- In telephone cabling, graph theory is effectively used.
- In job scheduling algorithms, the graph is used.

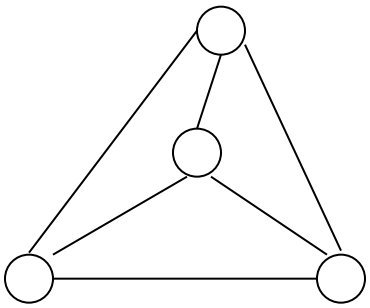
Solution: a b c d

18. Prove that the number of odd degree vertices in a connected graph should be even. (May/June 2007) - An

Consider a graph with odd degree vertices.



Degree =1



No. of vertices = 2 Degree = 3 No. of vertices = 4

This implies that the no. of vertices is even for odd degree graphs.

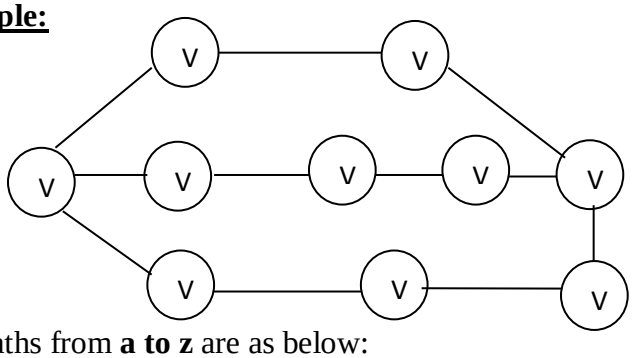
19. What is a single source shortest path problem? - U

The single source shortest path problem finds the minimum cost from single source vertex to all other vertices. Dijkstra's algorithm is used to solve this problem which follows the greedy technique.

20. What is unweighted shortest path? -- U

The unweighted shortest path is the path in unweighted graph which is equal to number of edges traveled from source to destination.

Example:



The paths from **a to z** are as below:

S.No	Path	Number of edges
1	V_1-V_2-V_3-	3
2	V_1-V_4-V_5-	4
3	V_1-V_7-V_8-	4

V_1-V_2-V_3-V_{10} is the shortest path.

21. What are the different kinds of graph traversals? -R

- Depth-first traversal
- Breadth-first traversal

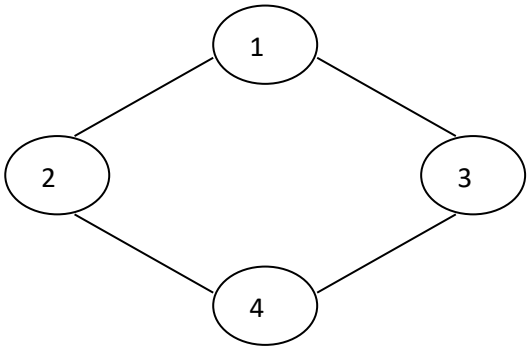
22. What is depth-first traversal? Give an example. -- U

Depth-first search (**DFS**) is a graph search algorithm that begins at the root and explores as far as possible along each branch before backtracking.

Note: Backtracking is finding 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.

Example:

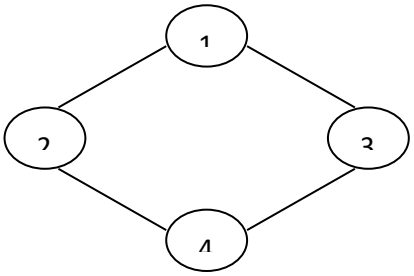
The Depth-first traversal is **1 – 2 – 4 – 3**



23. What is breadth-first traversal? Give an example. - U

Breadth-first search (**BFS**) is a graph search algorithm that begins at the root node and explores all the neighboring nodes. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal.

Example:

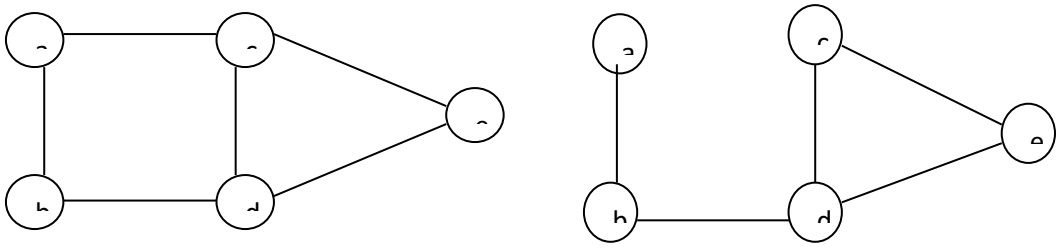


The breadth-first Traversal is **1 – 2 – 3 – 4**

24. What is biconnectivity? – U

Biconnected graphs are the graphs which cannot be broken into two disoconnected graphs by disconnecting single edge.

Example



In the given example, after removing edge E_1 the graph does not become disconnected.

25. What is a Spanning Tree?

Given an undirected and connected graph G=(V,E), a spanning tree of the graph G is a tree that spans G (that is, it includes every vertex of G) and is a subgraph of G (every edge in the tree belongs to G)

26. Define prims algorithm.

Prim's Algorithm is a greedy algorithm that is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized.

Prim's algorithm starts with the single node and explores all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected.

27. Define Kruskal algorithm.

Kruskal's Algorithm is used to find the minimum spanning tree for a connected weighted graph. The main target of the algorithm is to find the subset of edges by using which we can traverse every vertex of the graph. It follows the greedy approach that finds an optimum solution at every stage instead of focusing on a global optimum.

28. Define Bellman Ford algorithm.

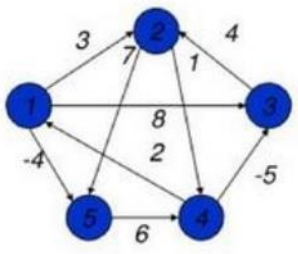
Bellman Ford algorithm helps us find the shortest path from a vertex to all other vertices of a weighted graph.It can work with graphs in which edges can have negative weights.

29. Define Dijkstra's algorithm.

Dijkstra's Algorithm finds the shortest path between a given node (which is called the "source node") and all other nodes in a graph.This algorithm uses the weights of the edges to find the path that minimizes the total distance (weight) between the source node and all other nodes.

30. Define Floyd-Warshall algorithm.

The all pair shortest path algorithm is also known as Floyd-Warshall algorithm is used to find all pair shortest path problem from a given weighted graph. As a result of this algorithm, it will generate a matrix, which will represent the minimum distance from any node to all other nodes in the graph.



$$\begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

At first the output matrix is same as given cost matrix of the graph. After that the output matrix will be updated with all vertices k as the intermediate vertex.

31. Give the Floyd's algorithm (C) (Nov 17)

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]

32. Define Prims Algorithm. (R)

Prim's algorithm constructs a minimum spanning tree through a sequenceof expanding subtrees. The initial subtree in such a sequence consists of a singlevertex selected arbitrarily from the set V of the graph's vertices. On each iteration,the algorithm expands the current tree in the greedy manner by simply attaching toit the nearest vertex not in that tree.

33. Write down the optimization technique used for Warshalls algorithm. State the rules and assumptions which are implied behind that. (MAY 2015) (R)

Warshalls algorithm constructs the transitive closure of given digraph with n vertices through a series of n-by-n Boolean matrices. The computations in warshalls algorithm are given by following sequences,

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

Rules:

1.Start with computation of $R^{(0)}$. In $R^{(0)}$ any path with intermediate veritices is not allowed. That means only direct edges towards the vertices are considered. In other words the path length of one edge is allowed in $R^{(0)}$. Thus $R^{(0)}$ is adjacency matrix for the digraph.

2. Construct $R^{(1)}$ in which first vertex is used as intermediate vertex and a path length of two edge is allowed. Note that $R^{(1)}$ is build using $R^{(0)}$which is already computed.

3. Go on building $R^{(k)}$ by adding one intermediate vertex each time and with more path length. Each $R^{(k)}$ has to be built from $R^{(k-1)}$

4. The last matrix in this series is $R^{(1)}$, in this $R^{(n)}$ all the n vertices are used as intermediate vertices. And the $R^{(n)}$ which is obtained is nothing but the transitive closure of given digraph.

34. Define the single source shortest path problem. (MAY\JUNE 2016) (R)

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.

35. How to calculate the efficiency of dijkstra's algorithm. (NOV/DEC 16) (R)

The time_efficiency depends on the data structure used for implementing the priority queue and for representing the input graph. It is in $\theta\ (|V|)^2$ for graphs represented by their weight matrix and the priority queue implemented as an unordered array. For graphs represented by their adjacency linked lists and the priority queue implemented as a min_heap it is in O(|E|log|V|).

36. Define transitive closure of a directed graph. (APR/MAY 2018)

The transitive closure of a directed graph with n vertices can be defined as the "n-by-n" Boolean matrix T={tij} in which,the element in the ith row (1<i<n) and the jth column (1<j<n) exists a non-trival directed path from the Ith vertex to the jth vertex,otherwise tij is 0.

37. Write short notes on the Maximum-Flow problem. (R)

Maximum-flow algorithms require processing edges in both directions, it is convenient to modify the adjacency matrix representation of a network as follows. If there is a directed edge from vertex i to vertex j of capacity uij ,then the element in the ith row and the jth column is set to uij , and the element in the jth row and the ith column is

set to −uij; if there is no edge between vertices i and j , both these elements are set to zero. Outline a simple algorithm for identifying a source and a sink in a network presented by sucha matrix and indicate its time efficiency.

38. Define maximum matching. (R)

In many situations we are faced with a problem of pairing elements of two sets.The traditional example is boys and girls for a dance, but you can easily thinkof more serious applications. It is convenient to represent elements of two givensets by vertices of a graph, with edges between vertices that can be paired. Amatching in a graph is a subset of its edges with the property that no two edgesshare a vertex. A maximum matching—more precisely, a maximum cardinalitymatching—is a matching with the largest number of edges.

39. Define maximum cardinality. (R) (NOV/DEC 2016) (R) (Apr 18)

Maximum cardinality matching—is a matching with the largest number of edges. The maximum-matching problem is the problem offinding a maximum matching in a given graph. For an arbitrary graph, this is arather difficult problem. It was solved in 1965 by Jack Edmonds.

40. Define maximum flow problem. (R)

The maximum flow problem can be seen as a special case of more complex network flow problems, such as the circulation problem. The maximum value of an s-t flow (i.e., flow from source s to sink t) is equal to the minimum capacity of an s-t cut (i.e., cut severing s from t) in the network, as stated in the max-flow min-cut theorem.

41. Define Bipartite Graphs? (R) (Nov 17)

A bipartite graph, also called a bigraph, is a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent. A bipartite graph is a special case of a k-partite graphwith $k = 2$. The illustration above shows some bipartite graphs, with vertices in each graph colored based on to which of the two disjoint sets they belong.

42. What do you meant by perfect matching in bipartite graph? (APR/MAY 2017) (R)

When there are an equal number of nodes on each side of a bipartite graph, a perfect matching is an assignment of nodes on the left to nodes on the right, in such a way that

i)each node is connected by an edge to the node it is assigned to, and

ii)no two nodes on the left are assigned to the same node on the right

43. Define flow cut. (R)

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.

44. How is a transportation network represented? (R)(APR/MAY 18)

The transportation network can be represented by a connected weighted digraph with n vertices numbered from 1 to n and a set of edges E, with the following properties:

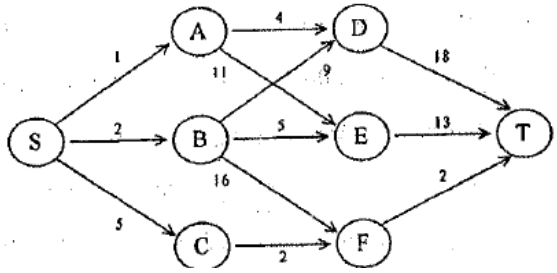
- It contains only one vertex with no entering edges called as **source** and assumed to be 1.
- It contains only one vertex at end with no leaving edges called as **sink** and assumed as n.
- The weight u_{ij} of each directed edge (i, j) is a positive integer, called as **edge capacity**.

45. Define the constraint in the context of maximum flow problem. (APR/MAY 2019)

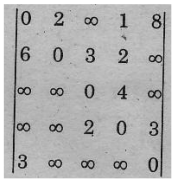
It represents the maximum amount of flow that can pass through an edge. ... , for each (capacity constraint: the flow of an edge cannot exceed its capacity); , for each (conservation of flows: the sum of the flows entering a node must equal the sum of the flows exiting a node, except for the source and the sink nodes).

PART –B

1. Write an algorithm for all pairs shortest path algorithm and what are the time and space complexity of the algorithm. **(APIRL/MAY2009) (APRIL/MAY 2012) (R)**
2. Explain Floyd algorithm with example. Write down and explain the algorithm to solve all pairs shortest paths problem. **(APRIL/MAY 2010)(MAY/JUNE 2013). (R)**

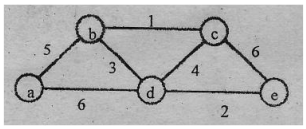


3. With a suitable example, explain all-pair shortest paths problem. **(R)**
4. How do you construct a minimum spanning tree using Kruskals algorithm? Explain?(4) **(MAY 2015) (R)**
5. Discuss about the algorithm and pseudocode to find the Minimum Spanning Tree using Prim's Algorithm. Find the Minimum Spanning Tree for the graph. Discuss about the efficiency of the algorithm**.(MAY\JUNE 2016) (C) (Apr 18)**
6. Solve the all-Pairs shortest-path problem for the diagraph with the following weight

matrix: 

0	2	∞	1	8
6	0	3	2	∞
∞	∞	0	4	∞
∞	∞	2	0	3
3	∞	∞	∞	0

(NOV/DEC 2016) (A)

7. Apply Kruskal's algorithm to find a minimum spanning tree of the following graph. **(NOV/DEC 2016) (A)** 

8. Explain the Dijkstra's shortest path algorithm and its efficiency.**(R) (NOV/DEC 2017)**

9. Write down the Dijkstra's algorithm and explain it with an example (8)

(APR/MAY 11) – Ap

10. Explain Dijkstra's algorithm with example.**(May/June 2012, Nov/Dec 2012) – Ap**

11. Write suitable ADT operation for shortest path problem. Show the simulation of shortest path with an example graph. **(April/May 2008) – Ap**

12. What is single source shortest path problem? Discuss Dijkstra's single source shortest path
algorithm with an example. (8) **(April/May 2007) –Ap**

13. Explain Depth – first & Breadth – First Traversal algorithms. – U

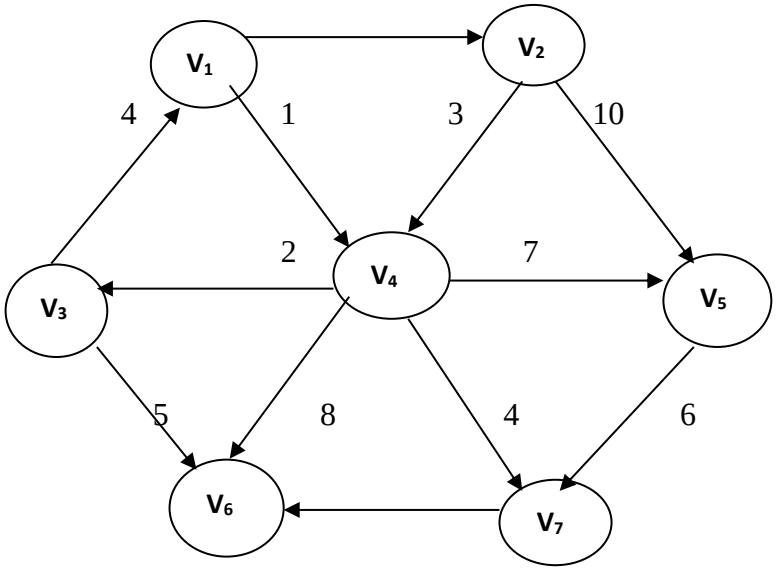
14. Explain depth first search on a graph with necessary data structures. (8)

(April/May 2008) – U

15. Write short notes on Biconnectivity? **– U**

16. Explain Dijkstra's algorithm using the following graph. Find the shortest path between v1, v2, v3, v4, v5, v6 & v7. 2

(May/June 2007) - Ap

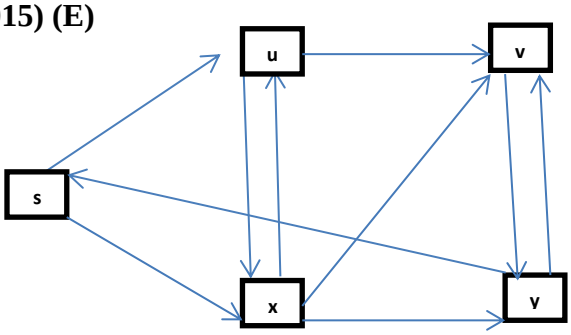


1

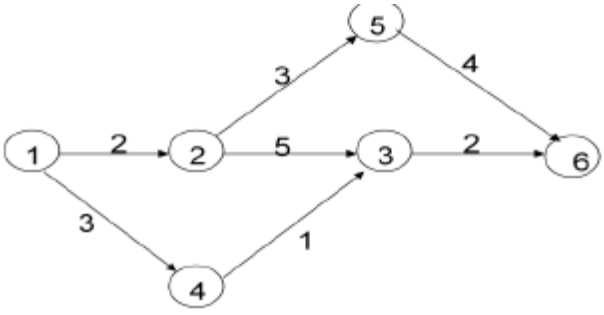
17. Distinguish between breadth first search and depth first search with example.**(13)**

(NOV/DEC 2018)

18. Explain the algorithm for Maximum Bipartite Matching. **(R)**
19. How do you compute maximum flow for the following graph using Ford-Fulkerson method? **(MAY 2015) (E)**



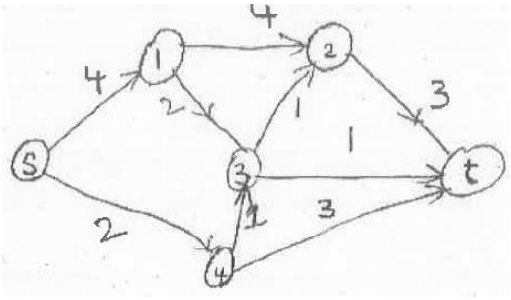
20. State and Prove Maximum Flow Min cut Theorem. **(MAY\JUNE 2016) (R)**
21. Apply the shortest Augmenting Path algorithm to the network shown below. **(MAY\JUNE 2016) (A)**



22. Explain KMP string

matching algorithm for finding a pattern on the text, a analysis the algorithm**.(APR/MAY 2017) (R)**

23. Determine the max-flow in the following network.(13) **(APR/MAY 2019)**



UNIT III

ALGORITHM DESIGN TECHNIQUES

Divide and Conquer methodology: Finding maximum and minimum - Merge sort - Quick sort **Dynamic programming:** Elements of dynamic programming — Matrix-chain multiplication - Multi stage graph — Optimal Binary Search Trees. **Greedy Technique:** Elements of the greedy strategy - Activity-selection problem –- Optimal Merge pattern — Huffman Trees.

PART A

1. **Give the general plan for divide-and-conquer algorithms. (APRIL/MAY 2008) (NOV/DEC 2008) (MAY/JUNE 2016) (R) (Nov 17)**
   The general plan is as follows.
   A problems instance is divided into several smaller instances of the same problem, ideally about the same size.
   The smaller instances are solved, typically recursively.
   If necessary the solutions obtained are combined to get the solution of the original problem.

2. **List the advantages of Divide and Conquer Algorithm.**
   Solving difficult problems, Algorithm efficiency, Parallelism, Memory access, Round off control.
3. **Give the recurrence relation of divide-and-conquer? (R)**
   The recurrence relation is

$$T(n) = \begin{cases} g(n) \\ t(n1) + t(n2) + ... + t(nk) + f(n) \end{cases}$$ g(n)

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

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

6. **List out the Disadvantages in Quick Sort**
   1. It is recursive. Especially if recursion is not available, the implementation is extremely complicated.
   2. It requires quadratic (i.e., n2) time in the worst-case.
   3. It is fragile i.e., a simple mistake in the implementation can go unnoticed and cause it to perform badly.

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

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

9. **List the Steps in Merge Sort**

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

   Recursion Step: Recursively sort array A1 and A2.

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

10. **List out Disadvantages of Divide and Conquer Algorithm**
    Conceptual difficulty
    Recursion overhead
    Repeated subproblems

11. **List out the Advantages in Quick Sort**
    - It is in-place since it uses only a small auxiliary stack.
    - It requires only n log(n) time to sort n items.
    - It has an extremely short inner loop

This algorithm has been subjected to a thorough mathematical analysis, a very precise statement can be made about performance issues.

12. Describe the recurrence relation of merge sort? (R)

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

$T(n)= a$

$2T(n/2)+n$ n = 1, a constant

13. State Master's theorem. (APR/MAY 18)

If $f(n)\ \theta(n_d)$ where $d \geq 0$ in recurrence equation $T(n) = aT(n/b)+f(n)$, then

$\theta(n_d)$ if $a<b_d$ $T(n)\ \theta(n_d \log n)$ if $a=b_d$

$\theta(n \log b_a)$ if $a>b_d$

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

14. List out the Disadvantages in Quick Sort

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

15. What are the differences between dynamic programming and divide and conquer approaches? (NOV/DEC 2018)

Divide and Conquer

Divide and Conquer works by dividing the problem into sub-problems, conquer each sub-problem recursively and combine these solutions.

Dynamic Programming

Dynamic Programming is a technique for solving problems with overlapping subproblems. Each sub-problem is solved only once and the result of each sub-problem is stored in a table (generally implemented as an array or a hash table) for future references. These sub-solutions may be used to obtain the original solution and the technique of storing the sub-problem solutions is known as memoization.

16. What is the time and space complexity of Merge sort? (APR/MAY 2019)

Time complexity = $\theta(n \log n)$

Space complexity = $n+\log_2 n$

$=\theta(n)$

17. Define dynamic programming. (APR/MAY 2017) (R)

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

18. What are the features of dynamic programming? (R)

- Optimal solutions to sub problems are retained so as to avoid re-computing 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? (R)

- 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.(APR/MAY 2017) (R)

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 defines 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.

21. Write the difference between the Greedy method and Dynamic programming.(APRIL/MAY 2011)(NOV/DEC 2012) (AN)

1.Only one sequence of decision is generated.	1.Many number of decisions are
2.It does not guarantee to give an optimal solution always.	2.It definitely gives an optimal

22. Define the principle of optimality. (APR/MAY 2012) (NOV/DEC 2016) (R) (Nov 17)

It states that in an optimal sequence of decisions, each sub sequence must be optimal. Touse dynamic programming the problem must observe the principle of optimality thatwhatever the initial state is, remaining decisions must be optimal with regard the statefollowing from the first decision.

23. What is the difference between dynamic programming and greed algorithm?(APRIL/MAY 2012) (AN)

GREEDY ALGORITHM	DYNAMIC PROGRAMMING
1. GA computes the solution in bottom up technique. It computes the solution from smaller sub-routines and tries many possibilities and choices before	DA computes its solution by making its choices in a serial fashion (never look back irrevocable).

it arrives at the optimal set of choices.	
It does not use the principle of optimality, i.e. there is no test by which one can tell GA will lead to an optimal solution.	It uses the principle of optimality.

24. Define Optimal Binary Search Trees? (R)

A binary search tree is one of the most important data structures in computer science. One of its principal applications is to implement a dictionary, a set of elements with the operations of searching, insertion, and deletion. If probabilities of searching for elements of a set are known e.g., from accumulated data about past searches - it is natural to pose a question about an optimal binary search tree for which the average number of comparisons in a search is the smallest possible.

25. List out the memory functions used under Dynamic programming. (MAY 2015) (R)

Memory functions solve in a top-down manner only sub problems that are necessary. Memory functions are an improvement of dynamic programming because they only solve sub problems that are necessary and do it only once. However they require more because it makes recursive calls which require additional memory.

- Greedy Algorithm
- Branch and Bound
- Genetic Algorithm

26. State the general principle of greedy algorithm. (NOV/DEC 16) (R)

A greedy algorithm is an algorithmic paradigm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum.

27. Define Optimization function ?

Every set of s that satisfies the constraints is a feasible solution.
Every feasible solution that maximizes is an optimal solution.

29.Define Greedy Methodology?

Greedy algorithms are simple and straightforward.
They are shortsighted in their approach
A greedy algorithm is similar to a dynamic programming algorithm, but the difference is that solutions to the sub problems do not have to be known at each stage, instead a "greedy" choice can be made of what looks best for the moment.

30.Define an optimization problem?

Given a problem instance, a set of constraints and an objective function.

Find a feasible solution for the given instance for which the objective function has an optimal value.
Either maximum or minimum depending on the problem being solved.A feasible solution that does this is called optimal solution.

31. Write all the Greedy Properties?

It consists of two property,
1. "greedy-choice property" ->It says that a globally optimal solution can be arrived at by making a locally optimal choice.
2. "optimal substructure" ->A problem exhibits optimal substructure if an optimal solution to the problem contains optimal solutions to the sub-problems.
are two ingredients in the problem that lend to a greedy strategy.

32.State feasible and consrtraint ?

Feasible: A feasible solution satisfies the problem's constraints
Constraints: The constraints specify the limitations on the required solutions

33.Write the Pseudo-code for Greedy Algorithm

```
Algorithm Greedy (a,n)
//a[1:n]contains the n inputs.
{
solution:=0;//initialize the solution.
for i:=1 to n do
{
x:=Select(a);
if Feasible( solution, x) then
solution:=Union(solution,x);
}
return solution
```

34. Compare With Dynamic Programming ?

Greedy And Dynamic Programming are methods for solving optimization problems.
Greedy algorithms are usually more efficient than DP solutions. However, often you need to use dynamic programming since the optimal solution cannot be guaranteed by a greedy algorithm.
DP provides efficient solutions for some problems for which a brute force approach would be very slow.

35. Differentiate PROS AND CONS

PROS:

- They are easier to implement,
- They require much less computing resources,
- They are much faster to execute.
- Greedy algorithms are used to solve optimization problems

CONS:

- Their only disadvantage being that they not always reach the global optimum solution;

- on the other hand, even when the global optimum solution is not reached, most of the times the reached sub-optimal solution is a very good solution

36. Comment on merge pattern?

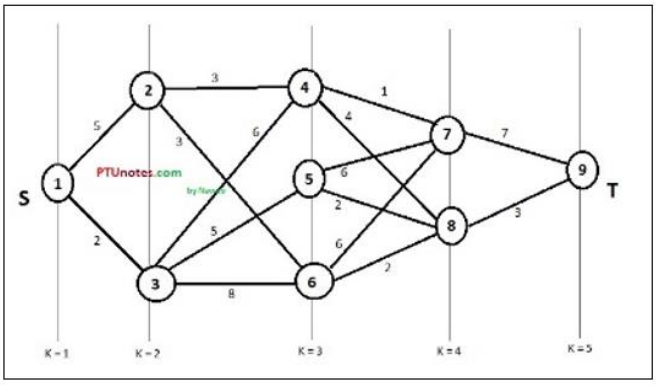
Merge a set of sorted files of different length into a single sorted file. We need to find an optimal solution, where the resultant file will be generated in minimum time.

If the number of sorted files are given, there are many ways to merge them into a single sorted file. This merge can be performed pair wise. Hence, this type of merging is called as **2-way merge patterns**.

37. Define multistage graphs. Give an example. (NOV/DEC 2018)

A multistage graph G = (V, E) is a directed graph where vertices are partitioned into k (where k > 1) number of disjoint subsets S = {s1,s2,…,sk} such that edge (u, v) is in E, then u Є si and v Є s1 + 1 for some subsets in the partition and |s1| = |sk| = 1.

The vertex s Є s1 is called the source and the vertex t Є sk is called sink.



38. State the principle of optimality. (APR/MAY 2019)

In an optimal sequence of decisions or choices, each subsequence must also be optimal. When it is not possible to apply the principle of optimality it is almost impossible to obtain the solution using the dynamic programming approach.
Example: Finding of shortest path in a given graph uses the principle of optimality.

PART-B

1. Define divide and conquer to apply the technique in binary search algorithm and to analysis it. **(APR/MAY 2006) (APR/MAY 2017) (R)**
2. Explain in detail in merge sort give an example **(APR/MAY 2008) (MAY 2016). (R)**
3. What is divide and conquer strategy and explain the binary search with suitable example problem.**(NOV/DEC 2011) (R)**

4. Distinguish between Quick sort and Merge sort, and arrange the following numbers in increasing order using merge sort. (18, 29, 68, 32, 43, 37, 87, 24, 47, 50) **(NOV/DEC 2011) (MAY/JUNE 2013). (A)**
5. Trace the steps of Mergesort algorithm for the elements 122,25,70,175,89,90,95,102,123 and also compute its time complexity**.(NOV/DEC 2012) (A) (Nov 17) (Apr 18)**

6. Write an algorithm to perform binary search on a sorted list of elements. Analyze the algorithm for the best case, average case and worst case.**(APR/MAY 2011). (AN)**

7. Using the divide and conquer approach to find the maximum and minimum in a set of 'n' elements. Also find the recurrence relation for the number of elements compared and solve the same.**(APR/MAY 2011). (A)**

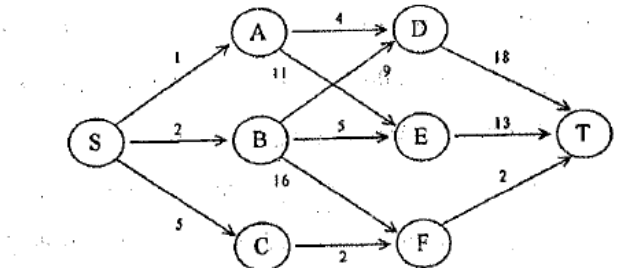
8. Trace maximum and minimum (using divide and conquer) algorithm for the following set of numbers. 20, 35, 18, 8, 14, 41, 3, 39,-20. **(A)**

9. Write a pseudo code using divide and conquer technique for finding the position of the largest element in an array of N numbers. **(A)**

10. Sort the following set of elements using merge sort: 12, 24, 8, 71, 4, 23, 6, 89, and 56. **(A)**
11. Explain in detail quick sorting method. Provide a complete analysis of quick sort. **(APR/MAY 2008) (Nov/Dec 2016) (APR/MAY 2017) (AN)**

12. A pair contains two numbers and its second number is on the right side of the first one in an array. The difference of a pair is the minus result while subtracting the second number from the first one. Implement a function which gets the maximal difference of all pairs in an array (using divide and conquer method). **(MAY/JUNE 2015) ®**
13. Write the algorithm for quick sort. Provide a complete analysis of quick sort for the given set of numbers 12,33,23,43,44,55,64,77 and 76. (13)**(NOV/DEC 2018)**
14. Write the quick sort algorithm and explain it with an example. Derive the worst case and average case time complexity. (5+4+4) **(APR/MAY 2019)**
15. (i)Write an algorithm to construct the optimal binary search tree (or) Discuss the algorithm for finding a minimum cost binary search trees.(8)

    (ii) Explain how dynamic programming is applied to solve travelling salesperson problem. **(APR/MAY 2010)(NOV/DEC 2012)(8) (R)**
16. Using Dynamic approach programming, solve the following graph using the backward approach.**(APRIL/MAY2011) (A)**



17. (i) Let A ={l/119,m/96,c/247,g/283,h/72,f/77,k/92,j/19} be the letters and its frequency of distribution in a text file. Compute a suitable Huffman coding to compress the data effectively. (8) **(MAY 2015)**

(ii)Write an algorithm to construct the optimal binary search tree given the roots r(i,j) ,$0 \le i \le j \le n$. Also prove that this could be performed in time O(n). (8) **(MAY 2015) (AN)**

18. Write the Huffman's Algorithm. Construct the Huffman's tree for the following data and obtain its Huffman's Code. **(APR/AMY 2017) (A)**

Characters	A	B	C	D	E	_
Probability	0.5	0.35	0.5	0.1	0.4	0.2

19. Explain the steps in building a Huffman Tree. Find the codes for the alphabets given below as according to frequency **(NOV/DEC 2017)**

A	2
E	5
H	1
I	2
L	2
M	2
P	2
R	1
S	2
X	1

20. (i) Write the Huffman code algorithm and derive its time complexity. **(5+2) (APR/MAY 2019)**

(ii) Generate the Huffman code for the following data comprising of alphabet and their frequency.(6) (APR/MAY 2019)

a:1,b:1,c:2,d:3,e:5,f:8,g:13,h:21

UNIT IV

STATE SPACE SEARCH ALGORITHMS

Lower - Bound Arguments - P, NP NP- Complete and NP Hard Problems. Backtracking – n-Queen problem - Hamiltonian Circuit Problem – Subset Sum Problem. Branch and Bound – LIFO Search and FIFO search - Assignment problem – Knapsack Problem – Travelling Salesman Problem - Approximation Algorithms for NP-Hard Problems – Travelling Salesman problem – Knapsack problem.

PART - A

1. Differentiate backtracking and Exhaustive search. (AN)

S.No	Backtracking	Exhaustive search
1.	Backtracking is to build up the solution vector one component at a time and to use modified criterion Function Pi(x1,..,xi) (sometimes called bounding function) to test whether the vector being formed has any chance of success.	Exhaustive search is simply a "brute – force" approach to combinatorial problems. It suggests generating each and every element of the problem's domain, selecting those of them that satisfy the problem's constraints, and then finding a desired element.
2.	Backtracking makes it possible to solve many large instances of NP-hard problems in an acceptable amount of time.	Exhaustive search is impractical for large instances, however applicable for small instances of problems.

2. What are the factors that influence the efficiency of the backtracking algorithm?(APRIL/MAY 2008) (R)

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.

3. What is backtracking?(Nov/Dec 2011) (R)

Backtracking constructs solutions one component at a time and such partially constructed solutions are evaluated as follows ._If a partially constructed solution can be developed further without violating the problem's constraints, it is done by taking the first remaining legitimate option for the next component. ._If there is no legitimate option for the next component, no alternatives for 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.

4. What is n-queens problem? (R)

The problem is to place 'n' queens on an n-by-n chessboard so thatno two queens attack each other by being in the same row or in the column orin the same

5. Draw the solution for the 4-queen problem. (R)

	Q		
			Q
Q			
		Q	

6. Define the Hamiltonian cycle.(NOV/DEC 2012) (R)

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 adjacentverticesvi0, 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.

7. What is the subset-sum problem?(Nov/Dec 2012) (R)

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

8. When can a node be terminated in the subset-sum problem?(NOV/DEC 2008) (R)

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+1>d (the sum s' is too large)

$\sum_{j=i\_1}^{n} sj < d$ (the sum s' is too small)

9. How can the output of a backtracking algorithm be thought of? (R)

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 linearlyordered set Si. If such a tuple (x1, …xi) is not a solution, the algorithm finds thenext element in Si+1 that is consistent with the values of (x1, …xi) and theproblem's constraints and adds it to the tuple as its (I+1)st coordinate. If suchan element does not exist, the algorithm backtracks to consider the next valueof xi, and so on.

10. Give a template for a generic backtracking algorithm.(APRIL/MAY 2012) (R)

ALGORITHM Backtrack(X [1..i])

//Gives a template of a generic backtracking algorithm

//Input X[1..i] specifies the first I promising components of a solution

//Output All the tuples representing the problem's solution

if X[1..i] is a solution write X[1..i]

else

for each element x□Si+1 consistent with X[1..i] and the constraints do

X[i+1] = x

Backtrack(X[1..i+1])

11. What is the method used to find the solution in n-queen problem by symmetry? (R)

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

12. State m color-ability decision problem.(NOV/DEC 2012) (R)

Let G be a graph and m be a given positive integer. We want to discover whether thenodes of G can be colored in such a way that no two adjacent nodes have the same color yet only m colors are used.

13. What are the two types of constraints used in Backtracking? (R)

- ✓ Explicit constraints
- ✓ Implicit constraints

14. Define implicit constraint.(APR/MAY 2010& 2012) (R)

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.

15. Define explicit constraint. (APR/MAY 2010& 2012) (R)

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.

16. What is a promising node in the state-space tree? (R) (Nov 17)

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

17. What is a non-promising node in the state-space tree? (R) (Nov 17)

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

otherwise it is called non-promising.

18. What do leaves in the state space tree represent? (R)

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

19. Define state space tree. (R)

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

20. Define a live node.(APR/MAY 2010) (R)

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

21. Define a dead node. (APR/MAY 2010) (R)

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

23. Compared to backtracking and branch and bound?(APRIL/MAY 2008) (AN)

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 optimizationproblems	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

24. When can a search path are terminated in a branch-and-bound technique. (R)

A search path at the current node in a state-space tree of a branch and-bound algorithm can be terminated if the value of the node's bound is not better than the value of the best solution seen so far the node represents no feasible solution because the constraints of the problem are already violated. The subset of feasible solutions represented by the node consists of asingle point in this case compare the value of the objective function for this feasible solution with that of the best solution seen so far andupdate the latter with the former if the new solution is better.

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

A simple way to find the upper bound 'ub' is to add 'v', the total valueof the items already selected, the product of the remaining capacity of theknapsack W-w and the best per unit payoff among the remaining items, whichis vi+1/wi+1, ub = v + (W-w)(vi+1/wi+1)

26. Define bounding. (R)

- ✓ Branch-and-bound method searches a state space tree using any search mechanism in which all children of the E-node are generated before another node becomes the E-node.

- ✓ Each answer node x has a cost c(x) and we have to find a minimum-cost answer node. Common strategies include LC, FIFO, and LIFO.
- ✓ Use a cost function $\hat{c}(\cdot)$ such that $\hat{c}(x) \leq c(x)$ provides lower bound on the solution obtainable from any node x.

27. Define Hamiltonian circuit problem. (R)

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

28. State Assignment problem. (MAY\JUNE 2016) (R)

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 [,] for each pair , = 1, 2, . . . , . The problem is to find an assignment with the minimum total cost.

29. What is state space tree? (MAY\JUNE 2016) (R)

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 evel represent the Choices for the second components & so on.

30. Give the purpose of lower bound. (MAY\JUNE 2016) (R)

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.

31. What is The Euclidean minimum spanning tree problem? (MAY\JUNE 2016) (R) (Apr 18)

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.

32. What is an articulation point in the graph? (APR/MAY 2017) (R)

In a graph, a vertex is called an articulation point if removing it and all the edges associated with it results in the increase of the number of connected components in the graph.

33. Write the formula for decision tree for searching a sorted array? (NOV/DEC 2016) (R)

$$C_{worst}(n) \geq \lceil \log_2 n! \rceil. \qquad \textbf{(11.2)}$$

Using Stirling's formula for $n!$, we get

$$\lceil \log_2 n! \rceil \approx \log_2 \sqrt{2\pi n}(n/e)^n = n \log_2 n - n \log_2 e + \frac{\log_2 n}{2} + \frac{\log_2 2\pi}{2} \approx n \log_2 n.$$

34. State the reason for terminating search path at the current node in branch and bound algorithm. (NOV/DEC 2016) (R)

- The value of the node's bound is not better than the value of the best solution seen so far.
- The node represents no feasible solutions because the constraints of the problem are already violated.
- The subset of feasible solutions represented by the node consists of a single point (and hence no further choices can be made)—in this case 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.

35. Differentiate feasible solution and optimal solution. (NOV/DEC 2017)

A *solution* (set of values for the decision variables) for which all of the constraints in the Solver model are satisfied is called a *feasible solution.* An *optimal solution* is a feasible solution where the objective function reaches its maximum (or minimum) value.

36. How is lower bound found by problem reduction? (APR/MAY 2018)

37. What are tractable and non-tractable problems? (APR/MAY 2018)

Generally we think of problems that are solvable by polynomial time algorithms as being tractable, and problems that require super polynomial time as being intractable.

38. Give an example for sum-of-subset problem. (NOV/DEC 2018)

Subset sum problem is to find subset of elements that are selected from a given set whose sum adds up to a given number K. We are considering the set contains non-negative values. It is assumed that the input set is unique (no duplicates are presented).

39. State Hamiltonian circuit problem. (APR/MAY 2019)

PART –B

1. Explain the n-Queen's problem & discuss the possible solutions.**(APRIL/MAY2008) (APIRL/MAY2009)(NOV/DEC 2008) (R)**
2. Apply backtracking technique to solve the following instance of subset sum problem : S={1,3,4,5} and d=11**(NOV/DEC 2006) (AN)**

3. Explain subset sum problem & discuss the possible solution strategies using backtracking. **(R)**

4. Write a recursive backtracking algorithm to find all the Hamiltonian cycles of a given graph. **(APIRL/MAY2009) (NOV/DEC 2008) (E)**

5. What is backtracking explain detail **(APR/MAY 2007) (R)**

6. Explain graph coloring in detail. (**APIRL/MAY2009) (R)**

7. Give the backtracking algorithm for knapsack problem. **(R)**

8. Explain the algorithm for finding all m-colorings of a graph.**(APRIL/MAY 2012) (R)**

9. Describe the backtracking solution to solve 8-Queens problem**.(APRIL/MAY 2010) (APRIL/MAY2012) (APR/MAY 2017) (A)**

10. With an example, explain Graph Coloring Algorithm.**(APRIL/MAY 2010) (R)**

11. Explain the n-Queen's problem and trace it for n=6.**(APRIL/MAY 2011) (A)**

12.(i)Explain Hamiltonian cycles. **(NOV/DEC 2012)** (8) **(R)**

(ii) With an example, explains Graph Coloring Algorithm (8) **(R)**

`

13. (i)Explain the control abstraction for Backtracking method. Describe the backtracking solution to solve 8-Queens problem. (8)**(NOV/DEC 2012) (A)**

(ii) Let w= {5, 7, 10, 12, 15, 18, 20} and m=35.Find all possible subset of w whose sum is equivalent to m.Draw the portion of state space tree for this problem. (8) **(A)**

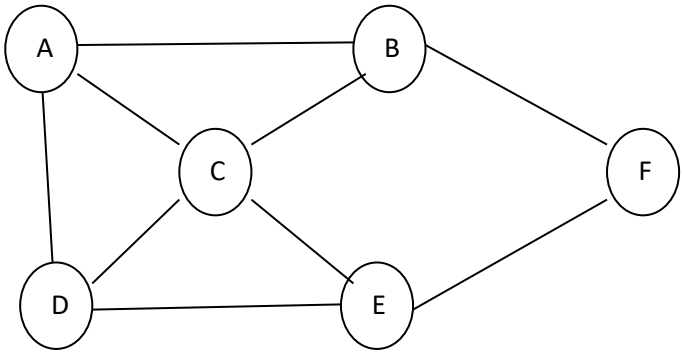
14. How backtracking works on 8-Queens problem with suitable example? **(MAY/JUNE 2013) (A)**

15.(i) Give the backtracking algorithm for knapsack problem. **(MAY/JUNE 2013)** (8)

(ii) Explain elaborately recursive backtracking algorithm. (8) **(R)**

16. Using backtracking, find the optimal solution to a knapsack problem for the knapsack instance n=8, m=110,(p1…p7)=(11,21,31,33,43,53,55,65) and (w1…w7)=(1,11,21,33,43,53,55,65). **(A)**

17. Write an algorithm to determine Hamiltonian cycle in a given graph using back tracking. For the following graph determine the Hamiltonian cycle**. (A)**



18. Write an algorithm to determine the Sum of Subsets for a given sum and a Set of numbers. Draw the tree representation to solve the subset sum problem given the numbers set as {3, 5, 6, 7, 2} with the sum=15. Derive all the subsets**. (A)**

19. Write an algorithm for N QUEENS problem and Trace it for n=6. **(R)**

20. What is Branch and bound? Explain in detail.**(MAY/JUNE 07) (APIRL/MAY2009) (NOV/DEC 2008) (APR/MAY 2017) (R)**

21. (i)Suggest an approximation algorithm for travelling salesperson problem. Assume that the cost function satisfies the triangle inequality. **(MAY 2015) (R)**

    (ii)Explain how job assignment problem could be solved, given n tasks and n agents where each agent has a cost to complete each task, using branch and bound. **(MAY 2015) (AN)**

22. (i)The knight is placed on the first block of an empty board and, moving according to the rules of chess, must visit each square exactly once. Solve the above problem using backtracking procedure. **(MAY 2015) (AN)**

    (ii)Implement an algorithm for Knapsack problem using NP-Hard approach. **(MAY 2015) (R)**

23. State the subset-sum problem and Complete state-space tree of the backtracking algorithm applied to the instance A={3, 5, 6, 7} and d=15 of the subset-sum problem**.(MAY\JUNE 2016) (A)**

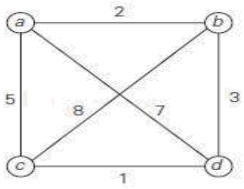
24. (i) Draw a decision tree and find the number of key comparisons in the worst and average cases for the three element bubble sort. (8) **(NOV/DEC 2016) (AN)**

    (ii) Write backtracking algorithm for 4 queen's problem and discuss the possible solution. (8) **(NOV/DEC 2016) (R)**

25. Solve the following instance of knapsack problem by branch and bound algorithm W= 15.(16) **(NOV/DEC 2016) (AN)**

ITEMS	WEIGHT	PROFIT
1	5	40
2	7	35
3	2	18
4	4	4
5	5	10
6	1	2

26. Apply Branch and bound algorithm to solve the travelling salesman problem. **(NOV/DEC 2016) (A)**



27. Give the methods for Establishing Lower Bound**. (NOV/DEC 17)**

28. Find the optimal solution using branch and bound for the following assignment problem.

	Job1	Job 2	Job 3	Job 4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

29. Elaborate on the nearest-neighbor algorithm and multifragment- heuristic algorithm for TSP problem. **(APR/MAY 2018)**
30. Consider the travelling salesperson instances defined by the following cost matrix. **(NOV/DEC 2018)**

∞	20	30	10	11
15	∞	16	4	2
3	5	∞	2	4
19	6	18	∞	3
16	4	7	16	∞

Draw the state space and show the reduced matrices corresponding to each of the node. (13)**(NOV/DEC 2018)**

31. Write an algorithm for subset sum and explain with an example. (13)**(APR/MAY 2019)**

32. Explain the 4-queens problem using backtracking. Write the algorithms . Giv the estimated cost for all possible solutions of 4-queens problem.Specify the implicit and explicit constrints. **(APR/MAY 2019)**

UNIT V

NP-COMPLETE AND APPROXIMATION ALGORITHM

Tractable and intractable problems: Polynomial time algorithms – Venn diagram representation - NP- algorithms - NP-hardness and NP-completeness – Bin Packing problem - Problem reduction: TSP – 3-CNF problem. **Approximation Algorithms**: TSP - **Randomized Algorithms**: concept and application - primality testing - randomized quick sort - Finding k^{th} smallest number

1. **What are NP- hard and NP-complete problems? (R)**

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

2. **Define bounding. (R)**

- ✓ Branch-and-bound method searches a state space tree using any search mechanism in which all children of the E-node are generated before another node becomes the E-node.
- ✓ Each answer node x has a cost c(x) and we have to find a minimum-cost answer node. Common strategies include LC, FIFO, and LIFO.
- ✓ Use a cost function $\hat{c}(\cdot)$ such that $\hat{c}(x) \leq c(x)$ provides lower bound on the solution obtainable from any node x.

3. **List example of NP hard problem. (R)**

NP hard graph problem

- ✓ clique decision problem(CDP)
- ✓ Node cover decision problem(NCDP).

4. **What is meant by NP hard and NP complete problem?(NOV/DEC 2011&NOV/DEC 2012) (R)**

- ✓ NP-Hard Problem: A problem L is NP-hard if any only if satisfy ability reduces to L.
- ✓ NP- Complete: A problem L is NP-complete if and only if L is NP-hard and L ϵ NP.

There are NP-hard problems that are not NP-complete.

Halting problem is NP-hard decision problem, but it is not NP-complete.

5. **An NP-hard problem can be solved in deterministic polynomial time,how?(NOV/DEC 2012)**

- ✓ If there is a polynomial algorithm for any NP-hard problem, then there are polynomial algorithms for all problems in NP, and hence P = NP.
- ✓ If P ≠ NP, then NP-hard problems cannot be solved in polynomial time, while P = NP does not resolve whether the NP-hard problems can be solved in polynomial time.

6. How NP-Hard problems are different from NP-Complete? (R)

These are the problems that are even harder than the NP-complete problems. Note that NP-hard problems do not have to be in NP, and they do not have to be decision problems.

The precise definition here is that a problem X is NP-hard, if there is an NP-complete problem Y, such that Y is reducible to X in polynomial time.

But since any NP-complete problem can be reduced to any other NP-complete problem in polynomial time, all NP-complete problems can be reduced to any NP-hard problem in polynomial time. Then, if there is a solution to one NP-hard problem in polynomial time, there is a solution to all NP problems in polynomial time.

7. Define P and NP problems. (APR/MAY 2017) (R)

P- Polynomial time **solving** . Problems which can be solved in polynomial time, which take time like O(n), O(n2), O(n3). Eg: finding maximum element in an array or to check whether a string is palindrome or not. so there are many problems which can be solved in polynomial time.

NP- Non deterministic Polynomial time solving. Problem which can't be solved in polynomial time like TSP(travelling salesman problem)

8. What are tractable and non-tractable problems? (APR/MAY 2018)

Generally we think of problems that are solvable by polynomial time algorithms as being tractable, and problems that require super polynomial time as being intractable.

9. Define P and NP problems. (NOV/DEC 2018)

All problems in P can be solved with polynomial time algorithms, whereas all problems in NP - P are intractable.

It is not known whether P = NP. However, many problems are known in NP with the property that if they belong to P, then it can be proved that P = NP.

If P ≠ NP, there are problems in NP that are neither in P nor in NP-Complete.

The problem belongs to class P if it's easy to find a solution for the problem. The problem belongs to NP, if it's easy to check a solution that may have been very tedious to find.

10. Define NP completeness and NP hard.(APR/MAY 2019)

NP-complete problems are the hardest problems in NP set. A decision problem L is NP-complete if:

1) L is in NP (Any given solution for NP-complete problems can be verified quickly, but

there is no efficient known solution).
2) Every problem in NP is reducible to L in polynomial time (Reduction is defined below).
A problem is NP-Hard if it follows property 2 mentioned above, doesn't need to follow property 1. Therefore, NP-Complete set is also a subset of NP-Hard set.

11. What do you meant by primality testing?

The basic structure of randomized primality tests is as follows:

- Randomly pick a number a.
- Check equality (corresponding to the chosen test) involving a and the given number n. If the equality fails to hold true, then n is a composite number and a is a witness for the compositeness, and the test stops.
- Get back to the step one until the required accuracy is reached.

After one or more iterations, if n is not found to be a composite number, then it can be declared probably prime.

12. What is Kth smallest number?

Given an array and a number k where k is smaller than the size of the array, we need to find the k'th smallest element in the given array. It is given that all array elements are distinct.

13. How quick sort using random pivoting?

In QuickSort we first partition the array in place such that all elements to the left of the pivot element are smaller, while all elements to the right of the pivot are greater than the pivot. Then we recursively call the same procedure for left and right subarrays.
Unlike merge sort, we don't need to merge the two sorted arrays. Thus Quicksort requires lesser auxiliary space than Merge Sort, which is why it is often preferred to Merge Sort.
Using a randomly generated pivot we can further improve the time complexity of QuickSort.

PART –B

1. Suggest an approximation algorithm for travelling salesperson problem. Assume that the cost function satisfies the triangle inequality. **(MAY 2015) (R)**
2. Implement an algorithm for Knapsack problem using NP-Hard approach. **(MAY 2015) (R)**
3. Discuss the approximation algorithm for NP hard problem? **(NOV/DEC 2016) (R)**
4. What is class NP? Discuss about any five problems for which no polynomial time for TSP problem. **(APR/MAY 2018)**
5. Elaborate on the nearest-neighbor algorithm and multifragment- heuristic algorithm for TSP problem. **(APR/MAY 2018)**
6. Discuss the approximation algorithm for NP-hard problem. (13)**(NOV/DEC 2018)**

7. Write an algorithm to solve the travelling salesman problem and prove that it is a 2 time approximation algorithm.(13)**(APR/MAY 2019)**

COURSE OUTCOMES:

Course Name : U23CST41 - DESIGN AND ANALYSIS OF ALGORITHMS

Year/Semester : II/ IV

Year of Study : 2024 –2025 (R – 2023)

On Completion of this course student will be able to

COURSE OUTCOMES

CO	DESCRIPTION
CO1	Analyze the efficiency of algorithms using various frameworks
CO2	Apply graph algorithms to solve problems and analyze their efficiency.
CO3	Make use of algorithm design techniques like divide and conquer and dynamic programming
CO4	Use design techniques like greedy technique to solve a problem.
CO5	Use the state space tree method for solving problems
CO6	Solve problems using approximation algorithms and randomized algorithms

CO-PO MATRIX:

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO.1	2	1	3	2	-	-	-	-	2	1	2	3
CO.2	2	1	1	1	1	-	-	-	1	3	3	3
CO.3	1	3	3	3	1	-	-	-	1	2	1	2
CO.4	1	3	3	3	1	-	-	-	1	2	1	2
CO.5	1	2	2	3	-	-	-	-	2	3	3	1
CO.6	1	2	3	2	3	-	-	-	3	1	3	3
CO	1	2	2	2	2	-	-	-	2	2	2	2

CO – PSO MATRIX:

CO	PSO1	PSO2	PSO3
CO.1	2	1	1
CO.2	2	3	3
CO.3	2	1	2
CO.4	2	1	2
CO.5	3	1	3
CO.6	1	3	3
CO	2	2	2