RV College of Engineering® , Bengaluru-560059
(Autonomous Institution affiliated to VTU, Belagavi)

Department of Computer Science and Engineering

DESIGN AND ANALYSIS OF ALGORITHMS
SUB CODE: 21CS43

IV SEMESTER B.E

LAB INSTRUCTORS MANUAL

2022-23

Prepared by: Prof. Girish Rao Salanke N S   **Verified by: HOD, CSE**

Vision and Mission of the Department of Computer Science and Engineering

Vision

To achieve leadership in the field of Computer Science & Engineering by strengthening fundamentals and facilitating interdisciplinary sustainable research to meet the ever growing needs of the society.

Mission

- To evolve continually as a center of excellence in quality education in Computers and allied fields.
- To develop state of the art infrastructure and create environment capable for interdisciplinary research and skill enhancement.
- To collaborate with industries and institutions at national and international levels to enhance research in emerging areas.
- To develop professionals having social concern to become leaders in top-notch industries and/or become entrepreneurs with good ethics.

PROGRAM EDUCATIONAL OBJECTIVES (PEO's):

PEO1: Develop Graduates capable of applying the principles of mathematics, science, core engineering and Computer Science to solve real-world problems in interdisciplinary domains.

PEO2: To develop the ability among graduates to analyze and understand current pedagogical techniques, industry accepted computing practices and state-of-art technology.

PEO3: To develop graduates who will exhibit cultural awareness, teamwork with professional ethics, effective communication skills and appropriately apply knowledge of societal impacts of computing technology.

PEO4: To prepare graduates with a capability to successfully get employed in the right role/become entrepreneurs to achieve higher career goals or takeup higher education in pursuit of lifelong learning.

PROGRAM OUTCOMES:

Engineering Graduates will be able to:

PO1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes

PSO1: System Analysis and Design

The student will:

1. Recognize and understand the dynamic nature of developments in computer architecture, data organization and analytical methods.

2. Learn the applicability of various systems software elements for solving real-world design problems.

3. Identify the various analysis & design methodologies for facilitating development of high quality system software products with focus on performance optimization.

4. Display good team participation, communication, project management and document skills.

PSO2: Product Development

The student will:

1. Demonstrate knowledge of the ability to write programs and integrate them resulting in state-of –art hardware/software products in the domains of embedded systems, databases /data analytics, network/web systems and mobile products.

2. Participate in teams for planning and implementing solutions to cater to business – specific requirements displaying good team dynamics and professional ethics.

3. Employee state-of-art methodologies for product development and testing / validation with focus on optimization and quality related aspects

Course Outcomes: After completing the course, the students will be able to:-	
CO1	Apply knowledge of computing and mathematics to algorithm analysis and design
CO2	Analyze a problem and identify the computing requirements appropriate for a solution
CO3	Apply mathematical foundations, algorithmic principles, and computer science theory to the modeling, and evaluation of computer-based solutions in a way that demonstrates comprehension of thetrade-offs involved in design choices.
CO4	Investigate and apply optimal design, development principles, skills and tools in the construction of software solutions of varying complexity.
CO5	Demonstrate critical, innovative thinking, and display competence in oral, written, and visual communication.
CO6	Exhibits positive group communication exchanges in order to accomplish a common goal and engage in continuing professional development.

CO-PO Mapping												
CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	2	-	-	-	-	-	-	-	-	-	-
CO2	-	3	-	-	1	-	-	-	-	-	-	-
CO3	-	-	3	-	-	1	1	-	-	-	-	-
CO4	-	-	-	3	2	-	-	2	-	-	-	-
CO5	-	-	-	-	-	-	-	-	-	3	-	-
CO6	-	-	-	-	-	-	-	-	3	-	-	3

High-3: Medium-2 : Low-1

LIST OF PROGRAMS

PART – A

Sl. No.	Program

1	Write a program to sort a given set of elements using Merge sort method and find the time required to sort the elements.
2	Write a program to obtain the Topological ordering of vertices in a given digraph using DFS method
3.	Write a program to find element uniqueness using presorting
4.	Write a program to implement Horspool's algorithm for String Matching.
5.	Write a program to implement 0/1 Knapsack problem using dynamic programming
6.	Write a program to find the shortest path using Dijkstra's algorithm for a weighted connected graph.
7.	Write a program to find a subset of a given set S = {S1, S2,.....,Sn} of npositive integers whose sum is equal to a given positive integer d. For example, if S ={1, 2,5, 6, 8} and d= 9, there are two solutions {1,2,6}and {1,8}. Display a suitable message, if the given problem instance doesn't have a solution.
8.	Write a program to implement N -queens problem using backtracking.

SOLUTIONS TO LAB PROGRAMS WITH EXPECTED INPUT AND OUTPUT

1. **Write a program to sort a given set of elements using Merge sort method and find the time required to sort the elements.**

```
#include<stdio.h>
#include<stdlib.h>

int count;

void merge(int a[10], int left, int mid, int right)
{
        int i, j, k, b[10];

        i = left;
        j = mid+1;
```

```c
        k = left;
        while( (i<=mid) && (j<=right))
        {
            count++;
                if(a[i] < a[j])
                {
                        b[k++] = a[i++];
                }
                else
                {
                        b[k++] = a[j++];
                }
        }
        while(i <= mid)
        b[k++] = a[i++];
        while(j <= right)
            b[k++] = a[j++];
    for(i=left; i<=right; i++)
                a[i] = b[i];
}
void mergesort(int a[10], int left, int right)
{
        int mid;
        if(left < right)
        {
                mid = (left + right)/2;
                mergesort(a,left,mid);
                mergesort(a,mid+1,right);
                merge(a,left,mid,right);

int main()
{
        int i,j,n,a[10];
        printf("\n Enter no of elements: ");
        scanf("%d",&n);
        printf("\nEnter elements\n");
        for(i=0;i<n;i++)
            scanf("%d",&a[i]);
        mergesort(a,0,n-1);
        printf("\n Sorted elements: \n");
        for(i=0;i<n;i++)
```

```
                    printf("%d\n",a[i]);
            printf("\n Number of counts : %d\n",count);
                  return 0;
        }
```

Output
```
Enter n: 5
Enter elements:1 3 7 2 0
Sorted Elements:
0
1
2
3
7
```

2. Write a program to obtain the Topological ordering of vertices in a given digraph using DFS method

```c
#include<stdio.h>

#include<stdlib.h>


int topo[10],k;


void dfs(int a[10][10],int n,int v[10],int source)

{

 int i;

 v[source]=1;

 for(i=1;i<=n;i++)

 {

    if(v[i]==0 && a[source][i]==1)

      dfs(a,n,v,i);

 }

 topo[++k]=source;

}
```

```c
int main()
{
    int n,i,j,a[10][10],v[10];
    printf("\n Enter the no of Vertices : ");
    scanf("%d",&n);
    printf("\n Enter the Adjacency matrix\n");
    for(i=1;i<=n;i++)
            for(j=1;j<=n;j++)
                    scanf("%d",&a[i][j]);
    for(i=1;i<=n;i++)
     v[i]=0;
    for(i=1;i<=n;i++)
    {
       if(v[i]==0)
         dfs(a,n,v,i);
    }
    printf("\n The topological ordering is\n");
    for(i=k;i>=1;i--)
                    printf("%d\t",topo[i]);
}
```

Output
```
Enter the number of vertices:8
Enter the adjacency matrix

0 1 1 0 1 0 0 0
0 0 1 0 0 1 1 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0

The topological ordering is
3 0 1 6 5 2 4 7
```

4. Write a program to implement Horspool's algorithm for String Matching.

```c
#include<stdio.h>
#include<conio.h>
#include<string.h>

#define MAX 256
int t[MAX];
int count=1;

void shifttable(char pat[])
 {
   int i,j,m;
   m=strlen(pat);
   for(i=0;i<MAX;i++)
   t[i]=m;
   for(j=0;j<m-1;j++)
   t[pat[j]]=m-1-j;

 }

int horspool(char src[],char pat[])
 {
   int i,j,k,m,n;
   n=strlen(src);
   m=strlen(pat);
   i=m-1;
```

```c
   while(i<n)
    {   k=0;
       while((k<m) && (pat[m-1-k]==src[i-k]))
        k++;
       if(k==m)
          return (i-m+1);
       else
       {
           i=i+t[src[i]];
           count=count+1;
       }
     }
     return -1;
  }

int main()
 {
    char src[100],pat[10];
    int pos;
    printf("\n Enter the main source string\n");
    gets(src);
    printf("\n Enter the pattern to be searched\n");
    gets(pat);
    shifttable(pat);
    pos=horspool(src,pat);
    if(pos>=0)
     {
       printf("\n Found at %d position ",pos+1);
       printf("\n number of shifts are %d",count);
     }
    else
       printf("\n String match failed");
    return 0;
 }
```

Output

Enter the Text: ABAABCD

Enter the pattern: ABC

Found at 4 position

5. Write a program to implement 0/1 Knapsack problem using dynamic programming.

```c
#include<stdio.h>
#include<stdlib.h>

int max(int a,int b)
{
  if(a>b)
    return a;
  else
    return b;
}

int main()
{
  int n,M,i,j,w[10],p[10],v[10][10],x[10]={0};
  printf("\n Read the number of objects :");
    scanf("%d",&n);
  printf("\n enter the weights of objects\n");
  for(i=1;i<=n;i++)
    scanf("%d",&w[i]);
  printf("\n enter the profits of objects\n");
```

```c
    for(i=1;i<=n;i++)
      scanf("%d",&p[i]);
    printf("\n enter the knapsack capacity :");
      scanf("%d",&M);
    for(i=0;i<=n;i++)
    {
      for(j=0;j<=M;j++)
        {
           if(i==0 || j==0)
               v[i][j]=0;
           else if(j < w[i])
               v[i][j]=v[i-1][j];
           else
               v[i][j]=max(v[i-1][j],v[i-1][j-w[i]] + p[i]);
               printf("%d\t",v[i][j]);
        }
        printf("\n");
     }
    i=n;
    j=M;
    while(i!=0 && j!=0)
    {
       if(v[i][j]!= v[i-1][j])
        {
           x[i]=1;
           j=j-w[i];
        }
        i=i-1;
    }
    printf("\n The max profit obtained is %d",v[n][M]);
    printf("\n The objects considered are");
    printf("\nobjno\tweight\tprofit\n");
    for(i=1;i<=n;i++)
      if(x[i]==1)
        printf("%d\t%d\t%d\n",i,w[i],p[i]);
    return 0;
  }
```

Output

```
Enter n (no. of items): 4

Enter the knapsack capacity: 5
```

```
enter the weights and profits:

2 12
1 10
3 20
2 15

goal = 37
```

6. Write a program to find the shortest path using Dijkstra's algorithm for a weighted connected graph.

```c
#include <stdio.h>
#include <stdlib.h>
#define INFINITY 999

void dijkstra(int cost[10][10],int n,int source,int v[10],int d[10])
{
    int min,i,j,u;
    v[source] = 1;
    for(i=1; i<=n; i++)
    {
      min = INFINITY;
      for(j=1; j<=n; j++)
      {
       if(v[j] == 0 && d[j] < min)
       {
         min = d[j];
```

```c
                u = j;
            }
        }
        v[u] = 1;
        for(j=1; j<=n; j++)
        {
            if(v[j] == 0 && (d[j] > (d[u] + cost[u][j])))
                d[j] = d[u] + cost[u][j];
        }
    }
}

int main()
{
    int n,cost[10][10],sourceNode,v[10],d[10];
    int i, j;
    printf("Enter no of vertices: ");
    scanf("%d",&n);
    printf("Enter Cost Adjacency matrix:\n");
    for(i=1; i<=n; i++)
    for(j=1; j<=n; j++)
    scanf("%d",&cost[i][j]);
    printf("Enter the source (%d to %d):\n",1,n);
    scanf("%d",&sourceNode);
    for(i=1; i<=n; i++)
    {
        d[i] = cost[sourceNode][i];
        v[i] = 0;
    }
    dijkstra(cost, n, sourceNode, v, d);
    printf("Shortest distance from %d\n\n",sourceNode);
    for(i=1; i<=n; i++)
    printf("%d --> %d = %d\n\n",sourceNode,i,d[i]);
    return 0;
}
```

Output

```
Enter n: 5

Enter the cost matrix:
0 3 999 7 999
3 0 4 2 999
```

```
999 4 0 5 6
7 2 5 0 4
999 999 4 6 0

Enter source: 2




Shortest distance from source 2
2-->1 =3
2-->2 =0
2-->3 =4
2-->4 =2
2-->5 =6
```

7. Write a program to find a subset of a given set S = {S1, S2,.....,Sn} of npositive integers whose sum is equal to a given positive integer d. For example, if S ={1, 2,5, 6, 8} and d= 9, there are two solutions {1,2,6}and {1,8}. Display a suitable message, if the given problem instance doesn't have a solution.

```c
#include<stdio.h>
#include<conio.h>
#define MAX 10
int s[MAX],x[MAX];
int d;
void sumofsub(int p, int k, int r)
{
  int i;
  x[k]=1;
  if ((p + s[k]) == d)
   {
     for(i=1; i<=k; i++)
     if (x[i] == 1)
     printf("%d ",s[i]);
     printf("\n");
   }
   else
     if (p+ s[k] + s[k+1] <= d)
       sumofsub(p + s[k],k+1, r-s[k]);
   if ((p + r - s[k] >= d) && (p + s[k+1] <= d))
    {
      x[k]=0;
      sumofsub(p,k+1,r-s[k]);
    }
 }
void main()
{
  int i,n,sum=0;
  clrscr();
  printf("\n Enter max. number : ");
  scanf("%d",&n);
  printf("\n Enter the set in increasing order :\n");
  for(i=1;i<=n;i++)
  scanf("%d",&s[i]);
  printf("\n Enter the max. subset value : ");
  scanf("%d",&d);
  for(i=1;i<=n;i++)
    sum=sum+s[i];
```

```
   if (sum < d || s[1] > d)
     printf("\n No subset possible");
   else
    sumofsub(0,1,sum);
    getch();
}
```

Output :

Enter max. number : 5
Enter the set in increasing order :
1 2 5 6 8

Enter the max. subset value : 9
1   2   6
1   8

8. Write a program to implement n-queens problem.

```c
#include <stdio.h>

//Function declarations
void nqueens(int n);
int can_place(int c[10],int r);
void display(int c[10],int r);

//Global variable
int count = 0;

int main()
{
    int n;

     //1. Read no. of queens
    printf("Enter n (no of queens): ");
    scanf("%d",&n);

    //2. Call function if solution exist
    if(n == 2 || n == 3)
        printf("Solution does not exist.");
    else
     {
       nqueens(n);
        printf("Total no. of solutions: %d\n",count);
     }

    return 0;
}

void nqueens(int n)
{
    int r;                   //Contains row no.
    int c[10];               //Stores queens positions in each row
     int i;

     r = 0;           //Select first queen (place queen in first
row)
     c[r] = -1;       //Initial position of queen

     while(r >= 0)    //As long as there are solutions
     {
         c[r]++;                          //Place queen in r th
coloumn
```

```c
    //verify there is no attack from any of t previous queens
placed
        while(c[r] < n && !can_place(c,r))
            c[r]++;

        if(c[r] < n)
        {
            if(r == n-1)      /if all n queens - display
            {
                printf("Solution %d: ",++count);
                for(i=0;i<n;i++)
                    printf("%4d",c[i]+1);

                display(c,n);
            }
            else        //else place the next queen in next row
            {
                r++;
                c[r] = -1;
            }
        }
        else
            r--;            //backtracking (go to previous row)
    }
}

//Function to check attack on queen r from 0-(r-1) queens
//return 0: if there is attack, other wise return 1;

int can_place(int c[10],int r)
{
    int i;

    for(i=0; i<r; i++)
    {
        if( (c[i] == c[r]) || (abs(i-r) == abs(c[i] - c[r])) )
            return 0;
    }
    return 1;
}

//Function to create chessboard with queens placed and display
void display(int c[10],int n)
{
    char cb[10][10];
    int i, j;

    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            cb[i][j] = '-';

    for(i=0;i<n;i++)
        cb[i][c[i]] = 'Q';
```

```
        //Display the chess board
        printf("\n\nChessboard: \n");
        for(i=0;i<n;i++)
        {
            for(j=0;j<n;j++)
                printf("%4c",cb[i][j]);
            printf("\n\n");
        }
}
```

Output

Run1

```
Enter the total number of queens: 4
Total number of solutions: 2
Solution 1:
Chessboard
_Q_ _
_ _ _ Q
Q_ _ _
_ _ Q _
Solution 2
Chessboard
_ _ Q _
Q_ _ _
_ _ _ Q
_ Q _ _
```

Run2
```
Enter the number of queens: 3
Solution does not exist
```

1. What is an algorithm?

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

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

1. Sorting 2. Searching

3. Numerical problems 4. Geometric problems

5. Combinatorial Problems 6. Graph Problems

7. String processing Problems

3. Name some basic Efficiency classes

1. Constant 2. Logarithmic 3. Linear 4. nlogn 5. Quadratic 6. Cubic 7. Exponential

8. Factorial

4. What are algorithm design techniques?

Algorithm design techniques (or strategies or paradigms) are general approaches to solving problems algorithmically, applicable to a variety of problems from different areas of computing. General design techniques are:

(i) Brute force (ii) divide and conquer

(iii) Decrease and conquer (iv) transform and conquer

(v) Greedy technique (vi) dynamic programming

(vii) Backtracking (viii) branch and bound

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

Time efficiency indicates how fast the algorithm runs. An algorithm's time efficiency is

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

6. How is the efficiency of the algorithm defined?

The efficiency of an algorithm is defined with the components.

(i) Time efficiency -indicates how fast the algorithm runs

(ii) Space efficiency -indicates how much extra memory the algorithm needs

7. What are the characteristics of an algorithm?

Every algorithm should have the following five characteristics

(i) Input

(ii) Output

(iii) Definiteness

(iv) Effectiveness

(v) Termination

Therefore, an algorithm can be defined as a sequence of definite and effective instructions, which terminates with the production of correct output from the given input.

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

i. Decide on parameter indicating an input's size.

ii. Identify the algorithm's basic operation

iii. Checking the no.of times basic operation executed depends on size of input.

iv. Set up sum expressing the no.of times the basic operation is executed. Depends on some additional property,then best,worst,avg.cases need to be investigated (establishing order of growth)

9. Define the terms: pseudocode, flow chart

A pseudocode is a mixture of a natural language and programming language like constructs. A pseudocode is usually more precise than natural language. A flowchart is a method of expressing an algorithm by a collection of connected geometric shapes containing descriptions of the algorithm's steps.

10. Write general plan for analyzing recursive algorithms.

i. Decide on parameter indicating an input's size.

ii. Identify the algorithm's basic operation

iii. Checking the no.of times basic operation executed depends on size of input. If it depends on some additional property,then best,worst,avg.cases need to be investigated of times the basic operation is executed

iv. Set up the recurrence relation, with an appropriate initial condition for the number

v. Solve recurrence (establishing order of growth)

11. Define the divide an conquer method.

Given a function to compute on 'n' inputs the divide-and-comquer strategy suggests splitting the inputs in to'k' distinct susbsets, 1<k <n, yielding 'k' subproblems. The subproblems must be solved recursively, and then a method must be found to combine subsolutions into a solution of the whole.

12. What is Merge sort?

Merge sort is divide and conquer strategy that works by dividing an input array in to two halves,sorting them recursively and then merging the two sorted halves to get the original array sorted

13. What is general divide and conquer recurrence?

Time efficiency T(n)of many divide and conquer algorithms satisfies the equation T(n)=a.T(n/b)+f(n).This is the general recurrence relation.

14. Describe the recurrence relation for merge sort?

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

T(n) = a n = 1, a a constant

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

15. The relation between order of growth of functions

$O(1) < O(\log n) < O(n) < O(n * \log n) < O(n^2) < O(n^3) < O(2^n)$

16. Asymptotic notations

Big Oh(Worst Case), Big Theta (Average Case), Big Omega(Best Case)

17. What is the time complexity of linear search?

$\Theta(n)$

18. What is the time complexity of binary search?

$\Theta(\log 2n)$ **3) What is the major requirement for binary search?**

The given list should be sorted.

19. What is binary search?

It is an efficient method of finding out a required item from a given list, provided the list is in order. The process is:

1. First the middle item of the sorted list is found.

2. Compare the item with this element.

3. If they are equal search is complete.

4. If the middle element is greater than the item being searched, this process is repeated in the

upper half of the list.

5. If the middle element is lesser than the item being searched, this process is repeated in the

lower half of the list.

20. What is parental dominance?

The key at each node is greater than or equal to the keys at its children.

21. What is heap?

A **heap** can be defined as a binary tree with keys assigned to its nodes (one key per node) provided the following two conditions are met:

1 The tree's shape requirement – The binary tree is essentially complete (or simply complete), that is, all its levels are full except possibly the last level, where only some rightmost leaves may be missing.

2. The parental dominance requirement – The key at each node is greater than or equal to the keys at its children

22. What is height of Binary tree?

It is the longest path from the root to any leaf.

23. What is the time complexity of heap sort?

$\Theta(n \log n)$

24. What is Merge Sort?

Merge sort is an O (n log n) comparison-based sorting algorithm. Where the given array is divided into two equal parts. The left part of the array as well as the right part of the array is

sorted recursively. Later, both the left sorted part and right sorted part are merged into a single sorted array.

25. Who invented Merge Sort?

John Von Neumann in 1945.

26. On which paradigm is it based?

Merge-sort is based on the divide-and-conquer paradigm.

27. What is the time complexity of merge sort?

n log n.

28. What is the space requirement of merge sort?

Space requirement: $\Theta(n)$ (not in-place).

29. When do you say an algorithm is stable and in place?

Stable – if it retains the relative ordering. In – place if it does not use extra memory.

30. Who invented Dijkstra's Algorithm?

Edsger Dijkstra invented this algorithm.

31. What is the other name for Dijkstra's Algorithm?

Single Source Shortest Path Algorithm.

32. Which technique the Dijkstra's algorithm is based on?

Greedy Technique.

33. What is the purpose of Dijkstra's Algorithm?

To find the shortest path from source vertex to all other remaining vertices

34. Name the different algorithms based on Greedy technique.

Prim's Algorithm, Kruskal's algorithm

35. What is the constraint on Dijkstra's Algorithm?

This algorithm is applicable to graphs with non-negative weights only.

36. What is a Weighted Graph?

A weighted graph is a graph with numbers assigned to its edges. These numbers are called weights or costs.

37. What is a Connected Graph?

A graph is said to be connected if for every pair of its vertices u and v there is a path from u to v.

38. What is the time complexity of Dijkstra's algorithm?

For adjacency matrix, It is
(IVI*IVI)

For adjacency list with edges stored as min heap, it is O(IEIlogIVI)

39. Differentiate b/w Traveling Salesman Problem(TSP) and Dijkstra's Algorithm.

In TSP, given n cities with known distances b/w each pair , find the shortest tour that passes through all the cities exactly once before returning to the starting city.

In Dijkstra's Algorithm, find the shortest path from source vertex to all other remaining vertices

40. Differentiate b/w Prim's Algorithm and Dijkstra's Algorithm?

Prim's algorithm is to find minimum cost spanning tree.

Dijkstra's algorithm is to find the shortest path from source vertex to all other remaining vertices.

41. What are the applications of Dijkstra's Algorithm?

It finds its application even in our day to day life while travelling. They are helpful in routing applications.

42. Who was the inventor of the Quicksort?

C.A.R.HOARE, a British Scientist.

43. Which design strategy does Quicksort uses?

Divide and Conquer

44. What is Divide and Conquer Technique?

(I) Divide the instance of a problem into two or more smaller instances

(II) Solve the smaller instances recursively

(III) Obtain solution to original instances by combining these solutions

45. What is the another name for Quicksort?

Partition Exchange Sort

46. Is Quicksort Stable as well as In place?

Not Stable but In place.

47. When do you say that a Quick sort having best case complexity?

When the pivot exactly divides the array into equal half

48. When do you say that Quick sort having worst case complexity?

When any one of the subarray is empty

49. How many more comparisons are needed for average case compared to best

case?

38% more

50. When do you say that the pivot element is in its final position?

When all the elements towards the left of pivot are <= pivot and all the elements towards the right of pivot are >= pivot.

51. What technique does kruskal's algorithm follow.

Greedy technique

37) What is the time complexity of kruskal algorithm.

With an efficient sorting algorithm, the time efficiency of an kruskal's algorithm will be in

O(|E|log|E|).

52. Who is the inventor of kruskal's algorithm.

Joseph Kruskal.

53. Which technique is used to solve BFS & DFS problems?

Decrease and Conquer

54. What is DFS traversal?

Consider an arbitrary vertex v and mark it as visited on each iteration, proceed to an unvisited vertex w adjacent to v. we can explore this new vertex depending upon its adjacent information. This process continues until dead end is encountered.(no adjacent unvisited vertex is available). At the dead end the algorithm backs up by one edge and continues the process of visiting unvisited vertices. This process continues until we reach the starting vertex.

55. What are the various applications of BFS & DFS tree traversal technique?

Application of BFS

Checking connectivity and checking acyclicity of a graph. Check whether there is only one root in the BFS forest or not. If there is only one root in the BFS forest, then it is connected graph otherwise disconnected graph.

For checking cycle presence, we can take advantage of the graph's representation in the form of a BFS forest, if the latter vertex does not have cross edge, then the graph is acyclic.

Application of DFS

To check whether given graph is connected or not.

To check whether the graph is cyclic or not.

To find the spanning tree.

Topological sorting.

56. Which data structures are used in BFS & DFS tree traversal technique?

We use Stack data structure to traverse the graph in DFS traversal.

We use queue data structure to traverse the graph in BFS traversal.

57. What is Dynamic Programming?

It is a technique for solving problems with overlapping sub problems.

58. What does Dynamic Programming have in common with Divide and- Conquer?

Both solve the problems by dividing the problem into sub problems. Using the solutions of sub problems, the solutions for larger instance of the problem can be obtained.

59. What is a principle difference between the two techniques?

Only one instance of the sub problem is computed & stored. If the same instance of sub problem is encountered, the solution is retrieved from the table and never recomputed. Very precisely re - computations are not preformed.

60. What is a spanning tree ?

A spanning tree of a weighted connected graph is a connected acyclic(no cycles) subgraph (i.e. a tree)that contains all the vertices of a graph and number of edges is one less than number of vertices.

61. What is a minimum spanning tree ?

Minimum spanning tree of a connected graph is its spanning tree of smallest weight.

62. Prim's algorithm is based on which technique.

Greedy technique.

63. Name some of the application greedy technique

They are helpful in routing applications: In the case of network where large number of computers are connected, to pass the data from one node to another node we can find the shortest distance easily by this algorithm. Communication Networks: Suppose there are 5 different cities and we want to establish computer connectivity among them, then we take into the account of cost factor for communication links. Building a road network that joins n cities with minimum cost.

64. Does Prim's Algorithm always yield a minimum spanning tree ?

Yes

65. What is the purose of Floyd's algoreithm?

- To find the all pair shortest path.

66. What is the time efficiency of floyd's algorithm? - It is same as warshall's algorithm that is $\theta(n^3)$.

67. What is shortest path?

- It is the path which is having shortest length among all possible paths.

68. Warshall's algorithm is optimization problem or non-optimization problem ?

Non-optimization problem

69. For what purpose we use Warshall's algorithm?

Warshall's algorithm for computing the transitive closure of a directed graph

70. Warshall's algorithm depends on which property?

Transitive property

71. What is the time complexity of Warshall's algorithm?

Time complexity of warshall's algorithm is $\theta(n^3)$.

72. What is state space tree?

Constructing a tree of choices being made and processing is known as state-space tree. Root represents an initial state before the search for solution begins.

73. What are promising and non-promising state-space-tree?

Node which may leads to solution is called promising.

Node which doesn't leads to solution is called non-promising.

74. What are the requirements that are needed for performing Backtracking?

Complex set of constraints. They are: i. Explicit constraints. ii. Implicit constraints.

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

76.   Define feasible and optimal solution.
Given n inputs and we are required to form a subset such that it satisfies some given constraints then such a subset is called feasible solution. A feasible solution either maximizes or minimizes the given objective function is called as optimal solution

77.   What are the constraints of knapsack problem?
To maximize $\sum p_i x_i$

The constraint is : $\sum w_i x_i \geq m$ and $0 \leq x_i \leq 1$ $1 \leq i \leq n$

where m is the bag capacity, n is the number of objects and for each object i w_i and p_i are the weight and profit of object respectively.

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

a) Prim's Algorithm

b) Kruskal's Algorithm

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

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

80. State efficiency of prim's algorithm.

$O(|V|^2)$ (Weight matrix and priority queue as unordered array)

$O(|E| \log|V|)$ (Adjacency list and priority queue as min-heap)

81. State Kruskal Algorithm.

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

82. State efficiency of Dijkstra's algorithm.

$O(|V|^2)$ (Weight matrix and priority queue as unordered array)

$O(|E| \log|V|)$ (Adjacency list and priority queue as min-heap)