VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Belgaum, Karnataka-590 014

Analysis & Design of Algorithms Lab

Subject Code: BCSL404

(As per Visvesvaraya Technological University Syllabus)

B.E- 4th Semester, Information Science and Engineering

Prepared By

Prof. Jagadish N
Assistant Professor

Prof. Sushma T M
Assistant Professor

Reviewed By
Prof. MITHUNA H R
Assistant Professor

Approved By:
Dr. Kala Venugopal
Head of Department Information Science and Engineering

ACHARYA INSTITUTE OF TECHNOLOGY
(Affiliated to VTU, Belgaum, Approved by AICTE, New Delhi and Govt. of Karnataka),
Acharya Dr. Sarvepalli Radhakrishnan Road, Bangalore-560107.
Ph: 91-080-28396011, 23723466, 28376431
URL: www.acharya.ac.in

2024-25

Table of contents

MOTTO

"Nurturing Aspirations Supporting Growth" VISION “Acharya Institute of Technology, committed to the cause of sustainable value-based education in all disciplines, envisions itself as a global fountainhead of innovative human enterprise, with inspirational initiatives for Academic Excellence”.

VISION OF THE INSTITUTE

Acharya Institute of Technology, committed to the cause of value-based education in all disciplines, envisions itself as fountainhead of innovative humanenterprise, with inspirational initiatives for Academic Excellence.

MISSION OF INSTITUTE

“Acharya Institute of Technology strives to provide excellent academic ambiance to the students for achieving global standards of technical education, foster intellectual and personal development, meaningful research and ethical service to sustainable societal needs.”

VISION OF THE DEPARTMENT

"To be center of Academic and Research excellence in the field of Information Technology inculcating value based education for the development of quality Human Resource"

MISSION OF THE DEPARTMENT

"Equip students with fundamental concepts, practical knowledge and professional ethics through dedicated faculty for higher studies and professional career in various Scientific, Engineering and Technological streams leading to proficiency in the field of Information Technology"

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Able to apply knowledge of information management and communication systems to provide secured solutions for real time engineering applications.

PSO2: Apply best software engineering practices, modern tools and technologies to deliver quality products.

PROGRAM OUTCOMES (Pos)

Engineering Graduates will be able to:

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

2. **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.

3. **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.

4. **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.

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

6. **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.

7. **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.

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

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

10. **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.

11. **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.

12. **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.

COURSE OUTCOMES

Course Outcomes-Program Outcomes mapping

CO1	**Develop** programs to solve computational problems using suitable algorithm design strategy.
CO2	**Compare** algorithm design strategies by developing equivalent programs and observing running times for analysis (Empirical).
CO3	Make **use** of suitable IDE tools to develop programs
CO4	**Choose** appropriate algorithm design techniques to develop solution to the computational and complex problems.
CO5	**Demonstrate** and present the development of program, its execution and running time(s) and record the results/inferences

COs	**Program Outcomes**												**Program Specific Outcomes**	
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	3	3	3	1	3				2	2		3	2	2
CO2	3	3	3	1	3				2	1		3	2	2
CO3	2	2	2		3				2	2		3	2	2
CO4	3	3	2	1	1				2	2		3	2	2
CO5	3	3	2	1	3				2	2		3	2	2

Rubrics for assessing student's performance in Laboratory courses

The internals marks of lab for 2022 scheme is 30 Marks for Continuous Evaluation and 20 Marks for Lab Internals.

Continuous Evaluation for 2022 scheme:

Sl No	**Parameters**	**Mark**	**10**	**6**	**2**	**0**
1.	Writing Program/Logic (present week's/previous week's)	10	The student is able to write the program without any logical and syntactical error and proper indentation is	The student is able to write the program with minor logical error	The student has written incomplete program with major logical and syntactical error	The student is not attempted to write program.

			followed.			
2.	Implementation in the target language with different inputs	10	Student is able to execute, debug, and test the program for all possible inputs/test cases.	Student is able to execute the program, but fails to debug, and test the program for all possible inputs/test cases.	Student is executed the program partially(fails to meet desired output)	The student has not executed the program.
	Parameters		6	4	2	0
3.	Record	6	Student submits the record on time and, neatly documented with all possible input/output samples.	Student submits the record on time but not documented properly with all possible input/output samples.	Student fails to submit the record on time.	The student does not submit the record.
	Parameters	4	4	2	1	0
4.	Viva	4	Student answers for at least 80% of questions	Student answers for at least 60% of questions	Student answers for at least 40% of questions	Student fails to answer any question
5.	Internal Assessment	20				

Programming Assignments

LABORATORY PROGRAMS

1. Design and implement C/C++ Program to find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal's algorithm.

```
#include<stdio.h>
void main()
{       int n,v,u,cost[10][10],parent[10]={0},i,j;
        int count=1,mincost=0,min,a,b;
        printf("\n enter no of vertices");
        scanf("%d",&n);
        printf("\n enter cost matrix");
        for(i=1;i<=n;i++)
          for(j=1;j<=n;j++)
          {
                scanf("%d",&cost[i][j]);
                if(cost[i][j]==0)
                cost[i][j]=999;
          }
          while(count<n)
          {   min=999;
                for(i=1;i<=n;i++)
                  for(j=1;j<=n;j++)
                    if(cost[i][j]<min)
                    {    min=cost[i][j];
                         a=u=i;
                         b=v=j;
                    }
          while(parent[u])
          u=parent[u];
          while(parent[v])
          v=parent[v];

          if(u!=v)
          {
             count++;
             printf("\n edge(%d,%d)=%d",a,b,min);
             mincost+=min;
             parent[v]=u;
          }
          cost[a][b]=cost[b][a]=999;
        }  /*End of While */
          printf("\n minimum cost=%d",mincost);
}
```

**********END OF PROGRAM**********

```
/*OUTPUT

Enter no of vertices   6

 Enter cost matrix

 999 7 8 999 999 999
7 999 3 6 999 999
8 3 999 4 9 999
999 6 4 999 1 5
999 999 9 1 999 2
999 999 999 5 2 999

 edge(4,5)=1
 edge(5,6)=2
 edge(2,3)=3
 edge(3,4)=4
 edge(1,2)=7
 minimum cost=17   */
```

2) Design and implement C/C++ Program to find Minimum Cost Spanning Tree of a given connected undirected graph using Prim's algorithm.

```
#include <stdio.h>
int a,b,u,n,i,j,ne=1,v;
int visited[10],min,mincost=0,cost[10][10];
int main()
{
        printf("\n Enter the no of vertices & graph data : ");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        printf("\n(%d,%d):",i,j);
                        scanf("%d",&cost[i][j]);
                        if(cost[i]==0)
                                cost[i][j]=999;
                }
        }
        for(i=0;i<=n;i++)
                visited[i]=0;
        printf("\n The edges of spanning tree are :\n");
        visited[1]=1;
        while(ne<n)
        {
                for(i=1,min=999;i<=n;i++)
                {
                        for(j=1;j<=n;j++)
                        {
                                if(cost[i][j]<min)
                                {
                                        if(visited[i]==0)
                                                continue;
                                        else
                                        {
                                                min=cost[i][j];
                                                a=u=i;
                                                b=v=j;
                                        }
                                }
                        }
                }
                if(visited[u]==0||visited[v]==0)
                {
                        printf("\n %d edge (%d,%d) = %d",ne++,a,b,min);
```

```
                mincost+=min;
                visited[b]=1;
            }
            cost[a][b]=cost[b][a]=999;
        }
        printf("\n\n\n Minimum cost : %d",mincost);
        return 0;
}
```

**********END OF PROGRAM**********

/* OUTPUT

Enter the no of vertices & graph data : 6

(1,1):999

(1,2):1

(1,3):8

(1,4):2

(1,5):999

(1,6):999

(2,1):1

(2,2):999

(2,3):9

(2,4):999

(2,5):999

(2,6):999

(3,1):8

(3,2):9

(3,3):999

(3,4):7

(3,5):6

(3,6):999

(4,1):2

(4,2):999

(4,3):7

(4,4):999

(4,5):5

(4,6):3

(5,1):999

(5,2):999

(5,3):6

(5,4):5

(5,5):999

(5,6):4

(6,1):999

(6,2):999

(6,3):999

(6,4):3

(6,5):4

(6,6):999

The edges of spanning tree are :

1 edge (1,2) = 1

2 edge (1,4) = 2
3 edge (4,6) = 3
4 edge (6,5) = 4
5 edge (5,3) = 6

Minimum cost : 16 */

3a. Design and implement C/C++ Program to solve All-Pairs Shortest Paths problem using Floyd's algorithm.

```
#include<stdio.h>
#include<time.h>
#define INFINITY 999
int min(int a,int b)
{
    return a<b?a:b;
}

void floyd(int w[10][10],int n)
{
    int i,j,k;
    for(k=1;k<=n;k++)
        for(i=1;i<=n;i++)
            for(j=1;j<=n;j++)
                w[i][j]=min(w[i][j],w[i][k]+w[k][j]);
}

int main(void)
{
    int a[10][10],n,i,j;
    double startTime,endTime;
    printf("\n Enter the no. of vertices:");
    scanf("%d",&n);
    printf("\n Enter the cost matrix, 0-self loop and 999-no edge\n");
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            scanf("%d",&a[i][j]);
    startTime=clock();
    floyd(a,n);
    endTime =clock();
    printf("\n Shortest path matrix:\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
            printf("%d\t",a[i][j]);
        printf("\n");
    }
    printf("Time taken is %10.9f\n",(double)(endTime-startTime));
    return 0;
}
```

/******************OUTPUT OFTHE PROGRAM*****************
Enter the no. of vertices:4

Enter the cost matrix, 0-self loop and 999-no edge
0 2 3 4
2 0 5 6
3 5 0 2
4 6 2 0

Shortest path matrix:
0 2 3 4
2 0 5 6
3 5 0 2
4 6 2 0
Time taken is 3.000000000

*******************************/

3b. Design and implement C/C++ Program to find the transitive closure using Warshal's algorithm.

```
#include<stdio.h>
void warshall(int a[10][10],int n);
void main()
{
	int i,j,n,a[10][10];
	printf("enter the no of vertices\n");
	scanf("%d",&n);
	printf("enter the adjacency matrix\n");
	for(i=1;i<=n;i++)
	    for(j=1;j<=n;j++)
	    {
			printf("(%d ,%d):",i,j);
			scanf("%d",&a[i][j]);
	    }
	    warshall(a,n);
	printf("Transitive closure for the given graph is \n");
	for(i=1;i<=n;i++)
	{
	    for(j=1;j<=n;j++)
			printf("%d\t",a[i][j]);
			printf("\n");
	}
}


void warshall(int a[10][10],int n)
{
	int i,j,k;
	for(k=1;k<=n;k++)
	    for(i=1;i<=n;i++)
			for(j=1;j<=n;j++)
				a[i][j]=a[i][j]||(a[i][k]&&a[k][j]);
	return;
}

/*****************END OF THE PROGRAM*****************/
```

/******************OUTPUT**********************

enter the no of vertices
4
enter the adjacency matrix
(1 ,1):0
(1 ,2):0
(1 ,3):1
(1 ,4):0
(2 ,1):1
(2 ,2):0
(2 ,3):1
(2 ,4):0
(3 ,1):0
(3 ,2):1
(3 ,3):0
(3 ,4):1
(4 ,1):1
(4 ,2):0
(4 ,3):1
(4 ,4):0
Transitive closure for the given graph is
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1

**/

4. Design and implement C/C++ Program to find shortest paths from a given vertex in a weighted connected graph to other vertices using Dijkstra's algorithm.

```
#include<stdio.h>
int n,cost[10][10],dist[10];
int min(int a,int b);
void read_mat(int n)
{
	int i,j;
	for(i=1;i<=n;i++)
		for(j=1;j<=n;j++)
		{
			printf("(%d,%d)",i,j);
			scanf("%d",&cost[i][j]);
		}
}

void shortestpath(int n,int s)
{
	int vis[10],c,u,i,k;
	for(i=1;i<=n;i++)
	{
		vis[i]=0;
		dist[i]=cost[s][i];
	}
	dist[s]=0;
	vis[s]=1;
	for(k=1;k<=n;k++)
	{
		c=999;
		for(i=1;i<=n;i++)
		if(vis[i]==0)
		{
			if(dist[i]<=c)
			{
				c=dist[i];
				u=i;
			}
		}
		vis[u]=1;
		for(i=1;i<=n;i++)
		dist[i]=min(dist[i],(dist[u]+cost[u][i]));
	}
}

int min(int a,int b)
```

```
{
        if(a<b)
                return a;
        else
                return b;
}

void main()
{
        int i,j,s;
        printf("enter the num of vertices\n");
        scanf("%d",&n);
        printf("enter the costadjacency matrix(999 for no edge);\n");
        read_mat(n);
        printf("enter source vertex\n");
        scanf("%d",&s);
        shortestpath(n,s);
        printf("the shortest path b/n vertex %d to",s);
        for(i=1;i<=n;i++)
                printf("vertex %d is %d\n",i,dist[i]);
}

/**********************END OF THE PROGRAM*************************************/

/* OUTPUT****
enter the num of vertices
6
enter the costadjacency matrix(999 for no edge);
(1,1)999
(1,2)6
(1,3)999
(1,4)7
(1,5)999
(1,6)999
(2,1)6
(2,2)999
(2,3)2
(2,4)3
(2,5)5
(2,6)999
(3,1)999
(3,2)2
(3,3)999
(3,4)999
(3,5)999
(3,6)4
(4,1)7
```

(4,2)3
(4,3)999
(4,4)999
(4,5)2
(4,6)999
(5,1)999
(5,2)5
(5,3)999
(5,4)2
(5,5)999
(5,6)3
(6,1)999
(6,2)999
(6,3)4
(6,4)999
(6,5)3
(6,6)999
enter source vertex
1
the shortest path b/n vertex 1 tovertex 1 is 0
vertex 2 is 6
vertex 3 is 8
vertex 4 is 7
vertex 5 is 9
vertex 6 is 12
*/

5. Design and implement C/C++ Program to obtain the Topological ordering of vertices in a given digraph.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main()
{
    int i,j,k,n,a[10][10],indeg[10],flag[10],count=0;
    printf("enter the number of vertices\n");
    scanf("%d",&n);
    printf("enter the adjacency matrix \n");
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
                scanf("%d",&a[i][j]);
    for(i=0;i<n;i++)
    {
        indeg[i]=0;
        flag[i]=0;
    }
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
                indeg[i]=indeg[i]+a[j][i];
    printf("topological order is \n");
    while(count<n)
    {
        for(k=0;k<n;k++)
        {
                if((indeg[k]==0)&&(flag[k]==0))
                {
                        printf("%d->",(k+1));
                        flag[k]=1;
                        for(i=0;i<n;i++)
                        {
                                if(a[k][i]==1)
                                indeg[i]--;
                        }
                }
        }
        count++;
    }
    printf("\n");
    return 0;
}
```

/**********END OF THE PROGRAM**********/

/* OUTPUT

```
enter the number of vertices
6
enter the adjacency matrix
0 1 1 0 0 0
0 0 0 1 0 0
0 0 0 1 0 0
0 0 0 0 1 1
0 0 0 0 0 0
0 0 0 0 0 0
topological order is
1->2->3->4->5->6->

*/
```

6. Design and implement C/C++ Program to solve 0/1 Knapsack problem using Dynamic Programming method.

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

int  max(int,int);

int v[10][10],w[10],p[10],x[10],count=0;
int findobjects(int n,int m);
void main()
{
      int m,n,i,j;
      printf("enter the number of objects\n");
      scanf("%d",&n);
      printf("enter the capacity of knapsack\n");
      scanf("%d",&m);
      printf("enter the weights of the objects\n");
      for(i=1;i<=n;i++)
          scanf("%d",&w[i]);
      printf("enter the profits\n");
          for(i=1;i<=n;i++)
                    scanf("%d",&p[i]);
      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("the output is:\n");
      for(i=0;i<=n;i++)
      {
          for(j=0;j<=m;j++)
          {
                    printf("%d\t",v[i][j]);
          }
          printf("\n");
      }
```

```
    printf("maximum profit is:%d\n",v[n][m]);
    findobjects(n,m);
    printf("objects included in knapsack \n");
    for(i=1;i<=count;i++)
        printf("%d\t",x[i]);
}

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

}


int findobjects(int n,int m)
{
    int i,j;
    i=n;
    j=m;
    while(i!=0&&j!=0)
    {
        if(v[i][j]!=v[i-1][j])
        {
                x[++count]=i;
                m=m-w[i];
                j=m;
        }
        i--;
    }
    return 0;
}



/***************END OF THE PROGRAM*****************/
```

```
/* OUTPUT

enter the number of objects
5
enter the capacity of knapsack
6
enter the weights of the objects
5
2
3
1
4
enter the profits
67
23
89
55
61
the output is:
0    0  0     0     0     0     0
0    0  0     0     0     67    67
0    0  23    23    23    67    67
0    0  23    89    89    112   112
0    55 55    89    144   144   167
0    55 55    89    144   144   167
maximum profit is:167
objects included in knapsack
4    3  2

*/
```

7. Design and implement C/C++ Program to solve discrete Knapsack and continuous Knapsack problems using greedy approximation method.

```
#include<stdio.h>
void knapsack(int n, float weight[], float profit[], float capacity)
{
  float x[20], tp = 0;
  int i, j, u;
  u = capacity;

  for (i = 0; i < n; i++)
    x[i] = 0.0;

  for (i = 0; i < n; i++) {
    if (weight[i] > u)
      break;
    else {
      x[i] = 1.0;
      tp = tp + profit[i];
      u = u - weight[i];
    }
  }

  if (i < n)
    x[i] = u / weight[i];

  tp = tp + (x[i] * profit[i]);

  printf("\nThe result vector is:- ");
  for (i = 0; i < n; i++)
    printf("%f\t", x[i]);

  printf("\nMaximum profit is:- %f", tp);

}

int main() {
  float weight[20], profit[20], capacity;
  int num, i, j;
  float ratio[20], temp;

  printf("\nEnter the no. of objects:- ");
  scanf("%d", &num);

  printf("\nEnter the wts and profits of each object:- ");
  for (i = 0; i < num; i++) {
```

```
    scanf("%f %f", &weight[i], &profit[i]);
  }

  printf("\nEnter the capacity of knapsack:- ");
  scanf("%f", &capacity);

  for (i = 0; i < num; i++) {
    ratio[i] = profit[i] / weight[i];
  }

  for (i = 0; i < num; i++) {
    for (j = i + 1; j < num; j++) {
      if (ratio[i] < ratio[j]) {
        temp = ratio[j];
        ratio[j] = ratio[i];
        ratio[i] = temp;

        temp = weight[j];
        weight[j] = weight[i];
        weight[i] = temp;

        temp = profit[j];
        profit[j] = profit[i];
        profit[i] = temp;
      }
    }
  }
  knapsack(num, weight, profit, capacity);
  return(0);
}
```

**************************END OF THE PROGRAM******************************

/*OUTPUT

Enter the no. of objects:- 5

Enter the wts and profits of each object:- 4 44
6 18
2 20
7 35
5 40

Enter the capacity of knapsack:- 10

The result vector is :- 1.000000 1.000000 0.800000 0.000000 0.000000 ***/**

8. Design and implement C/C++ Program to find a subset of a given set S = {sl , s2,.....,sn} of n positive integers whose sum is equal to a given positive integer d.

```
#include "stdio.h"
#include "math.h"
int i,j,k,set[10],d;
int n;
double pow(double,double);
void subset(int [],int,int);

int main()
{
    printf("\n Enter the no of Elements : ");
    scanf("%d",&n);
    printf("\n Enter the Elements \n");
    for(i=0;i<n;i++)
        scanf("%d",&set[i]);
    printf("\n Enter the Sum : ");
    scanf("%d",&d);
    subset(set,n,d);
    return 0;
}


void subset(int set[],int n,int d)
{
    int flag=0,index[30][100],i,j,sum,k,p;
    for(i=0;i<pow(2,n);i++)
    {
        k=i,sum=0;
        for(j=0;j<=n;j++)
        {
                if(k&1)
                {
                        index[i][j]=1;
                        sum+=set[j];
                }
                else
                index[i][j]=0;
                k=k>>1;
        }
        if(sum==d)
        {
                flag=1;
                printf("\n Solutions Are \n");
                for(j=0;j<n;j++)
```

```
            {
                if(index[i][j]==1)
                printf("\n\t %d",set[j]);
            }
        }
    }

if(flag==0)
    printf("\n No solutions are found!!!");
}
```

Output:

```
Enter the no of Elements : 6

Enter the Elements
1 3 5 7 9 11

Enter the Sum : 12

Solutions Are

        5
        7
Solutions Are

        3
        9
Solutions Are

        1
        11
```

9. Design and implement C/C++ Program to sort a given set of n integer elements using Selection Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define MAX 300000
int n;
int main()
{
        int a[MAX];
        double startTime,endTime;
        int i,j,temp,min;
        printf("Enter the number of elements to sort\n");
        scanf("%d",&n);
        //To generate numbers randomly and store in array
        for(i=0;i<n;i++)
                a[i]=rand();  //rand() function to generate n random numbers
        printf("\nBefore Sorting:\n");
        for(i=0;i<n;i++)
                printf("%d\t",a[i]);
        printf("\n");
        startTime=clock();
          for(i=0;i<=n-2;i++)
         {
                  min=i;
                  for(j=i+1;j<=n-1;j++)
                 {
                          if(a[j]<a[min])
                         min=j;
               }
                 temp=a[min];
                 a[min]=a[i];
                 a[i]=temp;
        }
        endTime = clock();
        printf("After Sorting:\n");
        for(i=0;i<n;i++)
          printf("%d\t",a[i]);
        printf("\n");
        printf("Time taken is %10.9f\n",(double)(endTime-startTime));
        return 0;
}
```

10. Design and implement C/C++ Program to sort a given set of n integer elements using Quick Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
#define MAX 500000
int partition(int a[],int low,int high);
int swap(int *a,int *b)
{
        int temp;
        temp=*a;
        *a=*b;
        *b=temp;
        return temp;
}

void quicksort(int a[],int low,int high)
{
        int s;
        if(low<high)
        {
                s=partition(a,low,high);
                quicksort(a,low,s-1);
                quicksort(a,s+1,high);
        }
}


int partition(int a[],int low,int high)
{
        int  i,j,pivot;
        pivot=a[low];
        i=low+1;
        j=high;
        while(1)
        {
                while((i<high)&&(pivot>=a[i]))
                i++;
                while(pivot<a[j])
                j--;
                if(i<j)
                {
                        swap(&a[i],&a[j]);
                }
                else
                {
                        swap(&a[low],&a[j]);
```

```
                                        return j;
                }
        }
}


void main()
{
        int a[MAX],n,i,low,high;
        clock_t start,end;
        float duration;
        printf("\n enter the number of elements to sort");
        scanf("%d", &n);
        printf("\n  elements are:\n");
        for(i=0;i<n;i++)
        {
                //scanf("%d",&a[i]);
                a[i]=random();
                printf("Before Sorting\n");
                printf("%d\t",a[i]);
        }
        printf("\n");
        start=clock();
        low=0,high=n-1;
        quicksort(a,low,high);
        end=clock();
        duration=(end-start);
        printf("\n sorted array is::\n");
        for(i=0;i<n;i++)
                printf("\n\t%d",a[i]);
        printf("\n");
        printf("time taken is %f",duration);
}
```

11. Design and implement C/C++ Program to sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n> 5000, and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
#include<time.h>
#define MAX 300000
void simple_merge(int a[],int low,int mid,int high);
void merge_sort(int a[],int low,int high);
int n;
int main(void)
{
        int a[MAX],i=0;
        double startTime,endTime;
        printf("Enter the number of elements to sort\n");
        scanf("%d",&n);
        //To generate randomly
        for(i=0;i<n;i++)
                a[i]=rand();
        printf("\nBefore Sorting:\n");
        for(i=0;i<n;i++)
                printf("%d\t",a[i]);
        printf("\n");

        startTime=clock();
        merge_sort(a,0,n-1);
        endTime = clock();
        printf("After Sorting:\n");
        for(i=0;i<n;i++)
                printf("%d\t",a[i]);
        printf("\n");
        printf("Time taken is %10.9f\n",(double)(endTime-startTime));
        return 0;
}

/*Function to sort sub arrays elements*/
void simple_merge(int a[],int low,int mid,int high)
{
        int i=low,j=mid+1,k=low,c[MAX];
        while(i<=mid && j<=high)
        {
                if(a[i]<a[j])
                {
                        c[k]=a[i];
                        i++;
```

```
                k++;
            }
            else
            {
                c[k]=a[j];
                j++;
                k++;
            }
        }
        while(i<=mid)
            c[k++]=a[i++];
        while(j<=high)
            c[k++]=a[j++];
        for(i=low;i<=high;i++)
            a[i]=c[i];
}


/*Function to divide the array list */
void merge_sort(int a[],int low,int high)
{
        int mid;
        if(low<high)
        {
            mid=(low+high)/2;
            {
                merge_sort(a,low,mid);
                merge_sort(a,mid+1,high);
            }
        simple_merge(a,low,mid,high);
        }
}
```

12. Design and implement C/C++ Program for N Queen's problem using Backtracking.

```
#include<stdlib.h>
#include <stdio.h>
int count=0,x[5];
int nqueen(int,int);
int main()
{
	int n;
	printf("\n enter the number of queen");
	scanf("%d",&n);
	nqueen(1,n);
	if(count==0)
		printf("\n no solution found");
	else
		printf("\n number of solution found is:%d",count);
	return 0;
}

int place(int k,int i)
{
	int j;
	for(j=1;j<k;j++)
		if((x[j]==i)||(abs(x[j]-i)==abs(j-k)))
	return 0;
	return 1;
}

int nqueen(int k,int n)
{
	int i,j,p;
	for(i=1;i<=n;i++)
	{
		if(place(k,i))
		{
			x[k]=i;
			if(k==n)
			{
				count++;
				printf("solution=%d",count);
				printf("\n");
				for(j=1;j<=n;j++)
				{
```

```
                                        for(p=1;p<=n;p++)
                                                if(x[j]==p)
                                                        printf("Q\t");
                                                else
                                                        printf("0\t");
                                                        printf("\n");
                                }
                                printf("\n");
                        }
                        else
                                nqueen(k+1,n);
                }
        }
        return 0;
}

/**********END OF THE PROGRAM************/


/**************OUTPUT***************

 enter the number of queen4
solution=1
0       Q       0       0
0       0       0       Q
Q       0       0       0
0       0       Q       0

solution=2
0       0       Q       0
Q       0       0       0
0       0       0       Q
0       Q       0       0


 number of solution found is:2

 ************************/
```

VIVA VOCE QUESTIONS

1. **What is an Algorithm?**

   Algorithm is a unambiguous Step by step procedure to solve a given problem which accepts valid finite number of input and generates valid finite number of output and should terminate within finite amount of time.

2. **What is a Flow Chart?**

   Flow chart is a Graphical Representation of a solution to the Problem.

3. **What is the difference between Algorithm, Flow Chart, and Program?**

- Algorithm specifies the different steps to be followed for solving a Problem.
- Flow Chart is a Graphical Representation of an Solution to the Problem.Both Algorithm and Flow Chart are Machine Independent.
- Program is a Set of Instructions which is used as a tool to communicate to the machine to get our work done, Program is Machine Dependent for particular Machine.

4. **What is the Aim of DAA lab or why we need to study DAA Lab?**

   DAA is a discipline, where we are dealing with designing or writing the algorithm keeping in Consideration of Space and Time Complexity, Such that Our Algorithm should execute in a very minimum amount of time by Minimum Space or RAM.

5. **Define Space and Time Complexity? Among this which one is more prioritized?**

   Space Complexeity is a measure of Amount of Space taken by a Program to finish its Execution.

   Time Complexeity is a measure of amount of time taken by a program to complete its Execution.Depending Upon Application it is considered, EX: For Mobile or Handheld Devices,We give Preference for both Space and time.

   For a Huge and Inter active Systems like Web Applications we give more Preferences to time Complexeity.

6. **What is Design and what is Analysis of a Program?**

   Design is a Process of Writing an algorithm to a given Problem so that it should accept finite number of input and generates valid finite number of output and Should exit appropriately.

   Analysis: Analysis is a next Phase of Writing an Algorithm, in this phase we calculate the Efficiency of an Algorithm i.e time and space needed by an algorithm.

7. **Write the general plan for analysing the recursive algorithms.**

   Identify the inputs.

   Identify the output is depended only on number of inputs.

Identify the Basic Operation in ALgorithm.

Form or write the Recursive Relation to the Algorithm.

8. What are the various notations used to write an algorithm.

(i)Pseudocode (ii) Natural Language and etc..

9. What is a Pseudocode?

It's a notation which is having the combination of Programming Constructs and English like Statements.

10. What is Asymptotic Notations? Explain various notations used for the same.

Asymptotic Notation is a way of representing an algorithm time Complexity in terms of Best, Average, and Worst.

There are 3 Notations used to represent (i) Ω (n)-Best Case:-Here the algorithm running time g (n) is less than f (n). for all $n>=n_o$ .Such that there exist a constant C.

$f(n)>=c.g(n)$,for all $n>n_o$

Refer Text book for Graph.

(ii)O-Worst Case:-Here the Algorithm runing time g(n) is greater than f(n) for all $n>=n_o$.

$f(n)<=c.g(n)$,for all $n>=n_o$, Such that there exist a constant C.

Refer Text book for Graph.

(iii) Θ-Average Case:When there is more than one parameter controlling the Execution time and We should go for Amortized analysis or Series of Execution for the Same Input and the Resultant Would be Average Case is which is $c1.g(n)<f(n)<c2.g(n)$,for all $n>=n_o$, Such that there exist a constant C.

Refer Text book for Graph.

11. List out the few topics you have studied or Explain few topics of your Favourite list.

1.Mention a topics Which you are really good in,the topics like

- Searching-Explain any one Searching Algorithm
- Sorting-Explain Any one Sorting Algorithm.
- String Matching-Explain any one method ,Select either Brute Force or Horspool Method.
- Divide and Conquer-Explain either Merge sort or Quick sort which You are More Comfortable.
- Greedy Technique-Define Greedy and Explain Any problem with Greedy Method Ex:Knapsack Problem,Job Sequencing with Deadlines,Prims ,Kruskals and etc..
- dynamic Programming-Explain any Problem with Dynamic Programming Approach

- Decrease and Conquer:Explain any problem which can be Solved Using decrease and Conquer,EX:DFS,BFS,Topological Sorting.
- Limitations of Algorithms:Explain few things about Decision trees and its use in solving a Problem.

- Coping with Limitations of Algorithms:Explain Backtracking and one problem which illustrates about Backtacking method and Wind Up.

12. What is the Time Complexeity of Bubble Sort,Selection Sort ,Merge Sort,Quick Sort?

Bubble Sort-$n^{2,}$

Selection Sort- n^2

Merge Sort-nlog.n

Quick Sort -nLogn, Worst case for Quick Sort- n2

13. Which sorting agorithm is more Efficient and why?

Quick Sorting is More Efficient ,because this algorithm is instable algorithm and inplace.

14. What do you mean by the term Instable Algorithms.

The Instable Algorithms are one,which lower index is greater than higher index

15. Which algorithms are faster?

Instable Algorithms are much Faster compared to Stable Algorithms.

16. For what type of instance Merge sort do better than Quick Sort?

For an Larger input and a sorted input values.

17. For what type of instance Quick sort do better than Merge Sort?

For Smaller Set of input numbers.

18. What are Inplace Algorithms?

Inplace Algorithms are the one which doesn't occupies Extra Space.

19. Write the order of growth terms as per the time Execution in Ascending Order.

logn,n,nlogn,n^2,n^3,.....n^n,2^n,n!

20. What is Brute Force Technique? When We Should USe?

Brute Force is a straight Forward Technique to solve a problem, we used to solve a Problem through this approach when we don't have sufficient data to solve a problem in Efficient Way.

21. What is the difference between Divide and Conquer, Decrease and ConQuer?

Divide and Conquer can be solved to solve a problem with a larger data set and when there is no

dependency between any of the data sets.

- ❖ Divide and Solve as Small as Small sets.
- ❖ Conquer or Merge it get one final resultant data set.

Decrease and Conquer is almost similar to Divide and Conquer but we are finding a solutions to the problem in a different variations, ex: Decrese by Constant (Usually by One), Decrease by Constant factor which is almost similar to Divide and Conquer Technique (Usually by two), Decrease by Variable (The Dividing Criteria changes for eac iteration depends upon the data set.

22. Define Greedy Technique?

Greedy Technique is always applied for the problem of the type optimization type, which reduces loss and increases profit.

23. Define Optimal and Feasible Solution.

Optimal Solution is a solution which is best among N Feasible Solution.

Feasible Solution is a solution which Satisfies a Problem Constraints/conditions.

24. Can A Problem solved by all the algorithmic Techniques.

Yes, but some problems will give better results with some Algorithmic Technique and it may give worst result when it is applied with other technique.

25. State and Explain Knapsack Problem.

Filling the Maximum number of items to the Knapsack (Container) Which Increases the profit and decreases the Loss.

26. State Few Algorithmic Techniques which you have studied?

Brute Force Technique

Divide and Conquer

Greedy Technique

Dynamic Programming

Decrease and Conquer

27. Which one is Most Admired algorithmic Technique?

Dynamic Programming.

28. What is spanning tree and Minimum Spanning tree?

A tree Without Cycles are called as spanning tree.

A Minimum Spanning Tree is a spanning tree which yeilds the very less Cost when all the edges cost summed up.

29. How Many Spanning Tree can a Tree can have?

A tree can have $n^{(n-2)}$ number of Spanning Trees.

30. Differentiate between Prims and Kruskals Algorithm for finding MST.

In Prims We consider any one vertex in the graph as Source and We compute the distance from that source to other vertices ,after computing the vertices which has minimum value among (n-1) vertices is added to tree vertices and that respective edges added to tree Edges Set.The above mentioned Process continues till we reach (n-1) vertices.

In Kruskals we first arrange the edges in Ascending Order and then we start to form the tree which wont form cycles,if adding that edges forms cycles then that edges is dropped from adding to tree edges.The above said process is continues till we reach the count of

(N-1) Vertices.

31. What is the Application of Prims and Kruskals Algorithm?

In Networks to remove the Cyclicity of the Network.

32. Explain Job Sequencing With Deadlines?

Placing or scheduling the maximum number of Jobs to a machine without violating the deadlines constraint of any of the Jobs in Sequence.

33. Why the Name Bubble Sort named?

Because in first Pass the first highest data will bubbles up, so since the largest element bubbles up in the first and second largest element bubbles up in the Second pass and so on, so hence the name bubble sort.

34. Why the Name Selection Sort?

The Selection sort is named because we initially first select an arrays first element as minimum and will compare with other elements ,so in pass one first least element goes to the first position and so on so forth for 2nd,3rd and so on.Selecting

35. What is the difference between Brute force strings matching to Horspool String Matching Method?

In brute Force we compare each and every element of the text to the pattern by shifting the text position by one and in Horspool method we shift it by number of shift positions recorded in the shift table.

36. Explain Merge Sort?

In Merge Sort will divide the entire input set by 2 until we reach low<high and later will find a solution to each item by comparing half of the array data set to the other half array data set and finally we

merge it to form a sinle array(conquer)

37. What is the Basic Operations in Merge sort and Quick sort?

In Merge Sort the Basic Operations is Comparisions and in Quick sort basic Operations is Partitioning and hence also known as partitioning sort.

38. Why the Insertion Sort?

We are Inserting an element to its suitable place by comparing n elements for each pass.

39. What is the Use of DFS and BFS?

DFS and BFS both used to check the Connectivity of a graph, Cyclicity in a graph, Spanning tree of a graph.

40. Differentiate between DFS and BFS.

DFS and BFS are both the Graph Traversing Technique, in which DFS Traverse the Graph in a depth wise (Vertical) and BFS Traverse the Graph from left to right (Horizontal)

41. Which Data structures used in BFS and DFS.

BFS USes Queue as its data structure and DFS uses as stack its Data structure.

42. What are back edges in DFS and Cross Edges in BFS?

Back Edges and Cross edges are the Edges which already visited by an ancestor node.

43. What is Topological Sorting?

Topological Sorting is a Sorting Technique used for sorting Vertices in the Graph.

44. What is the Conditions necessary for a Topological Sorting

For a Topological Sorting the Graph Should be DAG (Directed Acyclic Graph)

45. What are the Different methods used to solve a topological Sorting?

(i) Source Removal Method

(ii) Using DFS based Scheme.

What is the Use of Topological Sorting?

Use of Topological Ordering is in the field of Operating System for Scheduling and in Networks, Automation and Robotics.

46. What is Dijikstra's Algorithm?

Dijikstra's Algorithm is used to find the Single shortest Path from source to the other vertex.

47. What is a graph?

Graph is a component which is having a set of Edges and vertices G= {V, E}

48. What are the different ways that can be represents a graph?

Adjaceny Matrix and Adjacency List.

49. What is Adjacency Matrix?

Is a Matrix which illustrates the Graph in the form of Matrix,if it is a weights Graph then we initialize the value of the cost in that position(i,j) or else simply we write 1 to mention ther exist an edge between (i,j) OR else we use 0 or 9999 to mention non connectivity of a graph.

50. What is the limitations of Algorithms?

Algorithm can't find the better the soltions when we come across the tight lower bound, so we can find the better solutins which is not possible with Algorithmic way.To find the tight lower bound we use Decision Trees.

51. What is tight lower Bound?

It is a Lower bound which is a best lower bound for a problem, beyond that no algorithm will produce better results.

52. What are Decision Trees?

Decision trees are also known as Comparision Trees used to find the tight lower bound for a particular Problem ex: Tight Lower Bound for Sorting is n.logn

and tight lower bound for Searching is logn which is not possible to get the better result.

53. What is a polynomial problem(P-type)

P-type problem are decision problems in which we can find the solutions in a polynomial time and is of type deterministic.

54. What is NP-problem?

NP-Problem belongs to decision problem and these problems are Non Deterministic Polynomial i.e for which the problem doesn't have deterministic solutions and can be solved in Polynomial time

There are 2 phases in solving a problem.

(i)Guessing (Non-Deterministic stage) Producing N number of Candidate Outputs.

(ii) Verification (Deterministic Stage) Verifying The correctness of N Number of Candidate Outputs.

55. What is NP-Complete Problems?

Np_Complete Problems belongs to Decision problems and NP type Problems.

These problems can be find the solutions by converting or reducing to the problem which we know the Solutions.

56. What is a trivial lower bound?

Trivial bound can be derived by formulating the number of inputs that has to be given and number of

outputs that has to be generated.

57. Explain Bactracking W.r.t

(I)Subset Problem (ii) N-Queens Problem

58. Explain Subset Problem.

In a given Set S, find the Subset, in which the sum of all subset elements is equal to the sum d which is predefined in a problem.

59. Explain N-Queens Problem.

N-Queens Problem is of Placing a N-Queens in a N*N Chess board such that No 2-Queens should be placed in the same Row, Column and same diagnol (N=should consider both principal diagonal elements)

60. Wht is Hamilitonin Circuit?

Hamilitonian circuit is a problem in which that circuit starts from a source vertex and has other vertex in any order without repeating and should end with the Source vertex only i.e source and Destination vertex should be same.

61. Explain the Problem of TSP.

Travelling Sales Person Problem is a problem in which he should Visit N number of cities with a minimum number of Cost by visiting every city exactly one and the city what he is started should end with same city.

62. What is the Concept of Dynamic Programming?

Deriving a Solution to the basic Condition and Extracting the solutions for the rest of the other data sets by Previously drawnd Solution.Computer to other algorithmic Technique Dynamic Programming because it avoids lot of reworking on the same Solution what we have solved in the earlier phases of deriving the solution to the problem.

63. What is the goal of Warshalls Algorithm?

Warshall's algorithm is to find the shortest distance between a nodes to all the other nodes in the graph.

It Uses the Property of Transitive Closure i.e if there exist a path between (i, k) and (k, j) then there surely exist a path between (i,j)

(i,k) & (k,j)---□(I,J)

64. What is the use of Floyd's algorithm?

It is use to find the All pairs shortest Path of a Graph.

65. What is Parallel Programming?

Parallel programming is a technique of writing a Program to execute a series of instructions parallel to gain Speed up in Execution.

66. What is openMp?

OpenMp is an abbreviation of Open Multi programming and it is a programming tool which creates an environment to execute a program in the Parallel fashion i.e, N number of Instructions at the same time so as to achieve the spped up.

67. When one can go for Parallel Programming?

Parallel Programming Fashion is can be opted when there is no dependency between the series of Instructions that has to be Executed Parallely.

68. What is the Command used to compile a parallel Program?

Command is **cc-fopen filename.c**

69. What is cc stands for and –fopenmp stands for?

Cc stands for invoking a turbo C Compiler and –fopenmp is an option to the command which directs a compiler to invoke or use the parallel programming API so as to execute the program in a parallel Way.

70. What are the Maximum Number of Possible Ways for a travelling Sales Person with N number of Cities?

(N-1)! i.e for 5 cities (5-1)!= 4 !=24 ways.

71. Give Some Examples for a problem of type NP and NP-Complete?

Knapsack, Job Sequenncing with Deadlines, N-Queens Problem, Subset Problem, TSP and Hamiltonian circuit.

72. Give few Examples for a problem of type P?

Sorting Searching, String Matching and Hashing Problem.

73. What is the difference between Functions, Procedures, SubRoutines, Macros and Pseudo Code?

Functions: are the Constructs which consists of set of Instructions which is meant to Perform some Specific Task.It is the Programming Constructs used in Middle Level/High Level Languages like C/C++.

Procedures: It is same as the Functions but we call it Procedures in Assembly languages like

Masm,Tasm.The Size of the Procedures is small Compared to Functions and it uses the Mnemonics like JLR,ADD,SUB,DIV,CMP and etc.

SubRoutines: Same as above (Procedures, used in languages like FORTRAN)

Macros: Macros are the functions which is small in Size, usually used as Library files, it is invoked only if it is needed.

74. Name Standard Input, Standard output, Standard Error.

Standard Input-Keyboard

Standard Output-Monitor

Standard Error-Monitor

75. What are standard I/O functions?

printf () and scanf()

76. What is the use of using getch () in every Program? Does it Impact to the ouput of the Program?

getch () is a informal input functions which makes the output screen to be displayed until we enter a character(Read-input operations)

getch () doesn't Impact in anyways

77. What are Command line Arguments

The Arguments which are passed at command line to the function main ()

Are called as Command line Arguments.