

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



Analysis and Design Of Algorithms LAB REPORT

(19CS4PCADA)

Submitted by

RAVI SAJJANAR (1BM19CS127)

Under the Guidance of

**Dr. Nagarathna N
Professor, BMSCE**

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING



B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Mar-2021 to Jun-2022

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering



CERTIFICATE

This is to certify that the Lab Assignment work entitled “**Database Management System**” carried out by **RAVI SAJJANAR (1BM19CS127)** who is the bonafide students of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visveswaraiiah Technological University, Belgaum during the year 2021-2022. The Lab report has been approved as it satisfies the academic requirements in respect of **Analysis and Design Of Algorithms (19CS4PADA) LAB** work prescribed for the said degree.

Signature of the Guide Dr.
Dr.Nagarathna N
Professor
BMSCE, Bengaluru

Signature of the HOD
Dr. Umadevi V
Associate Prof.& Head, Dept. of CSE
BMSCE, Bengaluru

External Viva

Name of the Examiner

Signature with date

1. _____

2. _____

INDEX

S.NO	PROGRAM	PAGE
10	Sort a given set of N integer elements using Heap Sort technique and compute its time taken.	4
11	Implement Warshall's algorithm using dynamic programming.	9
12	Implement 0/1 Knapsack problem using dynamic programming.	13
13	Implement All Pair Shortest paths problem using Floyd's algorithm.	18
14	Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.	21
15	Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm.	26
16	From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.	30
17	"Sum of Subsets" problem	35
18	Implement "N-Queens Problem" using Backtracking.	40

PROGRAM 10: Heap Sort

Sort a given set of N integer elements using Heap Sort technique and compute its time taken.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int arr[1000000];

int temp;

void maxheap(int arr[], int size, int i)
{
    int j,k;

    for (k = 0; k < 180; k++)
    {
        for (j = 0; j < 40; j++)
        {

        }
    }
}

int largest = i;
int left = 2 * i + 1;
int right = 2 * i + 2;
```

```
if (left < size && arr[left] > arr[largest])
```

```
    largest = left;
```

```
if (right < size && arr[right] > arr[largest])
```

```
    largest = right;
```

```
if (largest != i)
```

```
{
```

```
    temp = arr[i];
```

```
    arr[i] = arr[largest];
```

```
    arr[largest] = temp;
```

```
    maxheap(arr, size, largest);
```

```
}
```

```
}
```

```
void heapSort(int arr[], int size)
```

```
{
```

```
    int i;
```

```
    for (i = size / 2 - 1; i >= 0; i--)
```

```
        maxheap(arr, size, i);
```

```
    for (i = size - 1; i >= 0; i--)
```

```
    {
```

```
        temp = arr[0];
```

```
        arr[0] = arr[i];
```

```
        arr[i] = temp;
```

```
        maxheap(arr, i, 0);
```

```

    }
}

void printArray(int arr[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}


int main()
{
    time_t start, end;
    int n;
    srand(time(0));
    printf("Enter the no of elements \n");
    scanf("%d", &n);

    for (int i = 0; i < n; i++)
    {
        arr[i] = rand();
    }

    start = time(NULL);
    heapSort(arr, n);
    end = time(NULL);

```

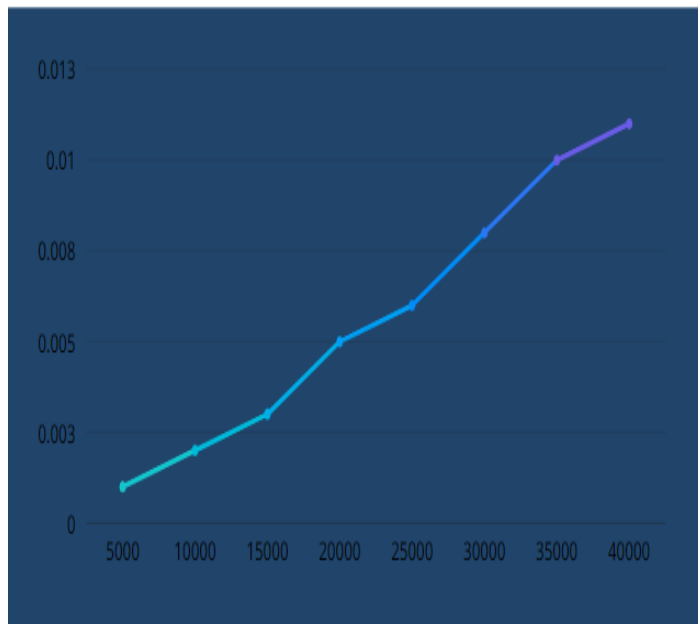
```
printf("The array is sorted\n");  
// printf("The sorted array is: \n");  
// printArray(arr, n);  
  
printf("The time taken is %.10f\n", difftime(end, start) /  
CLOCKS_PER_SEC);  
return 0;  
}
```

 D:\codes\LAB 10.exe

```
Enter the no of elements  
10000  
The array is sorted  
The time taken is 0.0020000000  
  
-----  
Process exited after 9.83 seconds with return value 0  
Press any key to continue . . .
```

HEAP SORT

IBM19CS127



n	t
5000	0.001
10000	0.002
15000	0.003
20000	0.005
25000	0.006
30000	0.008
35000	0.01
40000	0.011

PROGRAM 11: Warshall's algorithm

Implement Warshall's algorithm using dynamic programming.

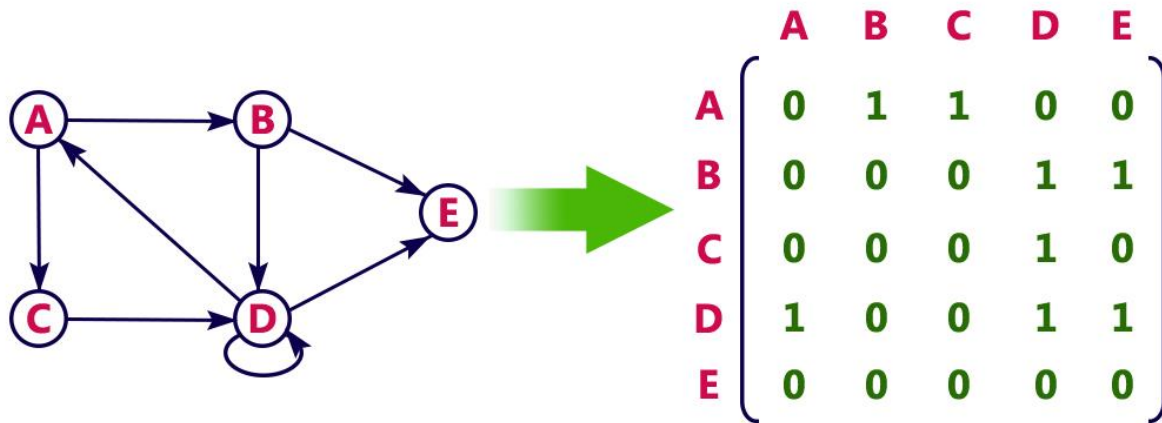
```
#include<stdio.h>
#include<conio.h>
int n,a[10][10],p[10][10];

void warshall(int n,int a[10][10],int p[10][10])
{
int i,j,k;
for(i=0;i<n;i++)
    for(j=0;j<n;j++)
        p[i][j]=a[i][j];

    for(k=0;k<n;k++)
        for(i=0;i<n;i++)
            for(j=0;j<n;j++)
                if((p[i][j]==0) && (p[i][k]==1 && p[k][j]==1))
                    p[i][j]=1;
}

int main()
{
    int i,j;
    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]);
    }
}
warshall(n,a,p);
printf("transitive closure\n");
for(i=0;i<n;i++)
{
    for(j=0;j<n;j++)
    {
        printf("%d\t",p[i][j]);
    }
    printf("\n");
}
}
```

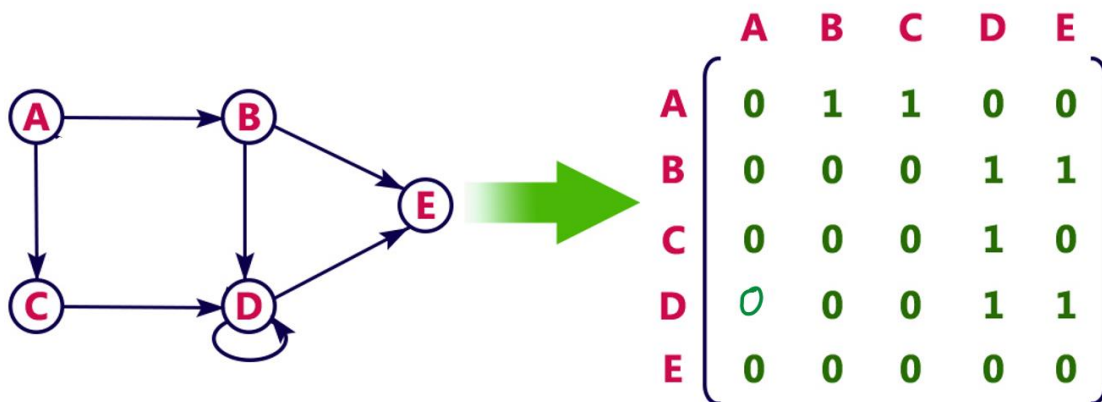


```

D:\codes\LAB 11.exe
enter the number of vertices
5
enter the adjacency matrix
0 1 1 0 0
0 0 0 1 1
0 0 0 1 0
1 0 0 1 1
0 0 0 0 0
transitive closure
1      1      1      1      1
1      1      1      1      1
1      1      1      1      1
1      1      1      1      1
0      0      0      0      0

-----
Process exited after 31.78 seconds with return value 0
Press any key to continue . . .

```



```

D:\codes\LAB 11.exe
enter the number of vertices
5
enter the adjacency matrix
0 1 1 0 0
0 0 0 1 1
0 0 0 1 0
0 0 0 1 1
0 0 0 0 0
transitive closure
0      1      1      1      1
0      0      0      1      1
0      0      0      1      1
0      0      0      1      1
0      0      0      0      0

-----
Process exited after 68.15 seconds with return value 0
Press any key to continue . . .

```

PROGRAM 12: Knapsack problem

Implement 0/1 Knapsack problem using dynamic programming.

Program:

```
#include<stdio.h>
#include<conio.h>
void knapsack();
int max(int,int);
int i,j,n,m,p[10],w[10],v[10][10];
void main()
{
    clrscr();
    printf("\nEnter the no. of items:\t");
    scanf("%d",&n);
    printf("\nEnter the weight of the each item:\n");
    for(i=1;i<=n;i++)
    {
        scanf("%d",&w[i]);
    }
    printf("\nEnter the profit of each item:\n");
    for(i=1;i<=n;i++)
    {
        scanf("%d",&p[i]);
    }
    printf("\nEnter the knapsack's capacity:\t");
    scanf("%d",&m);
    knapsack();
    getch();
}

void knapsack()
{
```

```

int x[10];
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]<0)
        {
            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("\nthe output is:\n");
for(i=0;i<=n;i++)
{
    for(j=0;j<=m;j++)
    {
        printf("%d\t",v[i][j]);
    }
    printf("\n\n");
}
printf("\nthe optimal solution is %d",v[n][m]);
printf("\nthe solution vector is:\n");
for(i=n;i>=1;i--)
{
    if(v[i][m]!=v[i-1][m])
    {
        x[i]=1;
        m=m-w[i];
    }
}

```

```
    }  
    else  
    {  
        x[i]=0;  
    }  
}  
for(i=1;i<=n;i++)  
{  
    printf("%d\t",x[i]);  
}  
}
```

```
int max(int x,int y)  
{  
    if(x>y)  
    {  
        return x;  
    }  
    else  
    {  
        return y;  
    }  
}
```

Output:

Enter the no. of items: 4

Enter the weight of each item:

2 1 3 2

Enter the profit of the each item:

12 10 20 15

Enter the Knapsack's capacity: 5

The output is:

0 0 0 0 0 0

0 0 12 12 12 12

0 10 12 22 22 22

0 10 12 22 30 32

0 10 15 25 30 37

The optimal solution is: 37

The solution vector is:

1 1 0 1

D:\codes\LAB 12.exe

enter the no. of items: 4

enter the weight of the each item:

2 1 3 2

enter the profit of each item:

12 10 20 15

enter the knapsack's capacity: 5

the output is:

0	0	0	0	0	0
0	0	12	12	12	12
0	10	12	22	22	22
0	10	12	22	30	32
0	10	15	25	30	37

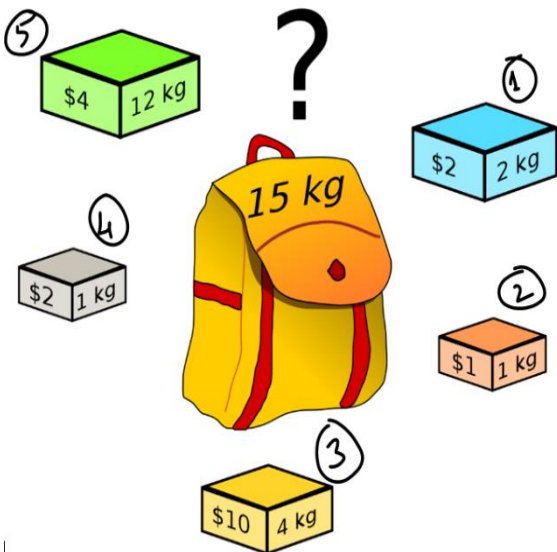
the optimal solution is 37

the solution vector is:

1 1 0 1

Process exited after 27.09 seconds with return value 0

Press any key to continue . . .



D:\codes\LAB 12.exe

2 1 10 2 4

enter the knapsack's capacity: 15

the output is:

0	0	0	0	0	0	0	0	0	0	0	2	2	2
0	2	2	2	2	2	2	2	2	0	1	2	3	3
0	1	2	3	3	3	3	3	3	0	1	2	3	10
0	1	2	3	10	11	12	13	13	13	0	2	3	4
0	2	3	4	10	12	13	14	15	15	0	2	3	4
0	2	3	4	10	12	13	14	15	15	0	2	4	6

the optimal solution is 12

the solution vector is:

1 0 1 0 0

Process exited after 205.4 seconds with return value 0

Press any key to continue . . .

PROGRAM 13: Floyd's algorithm

Implement All Pair Shortest paths problem using Floyd's algorithm.

Program:

```
#include<stdio.h>
#include<conio.h>
int a[10][10],n;
void floyds();
int min(int,int);
void main()
{
    int i,j;
    clrscr();
    printf("\nEnter the no. of vertices:\t");
    scanf("%d",&n);
    printf("\nEnter the cost matrix:\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    floyds();
    getch();
}

void floyds()
{
    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]=min(a[i][j],a[i][k]+a[k][j]);
    }
}
}
printf("\nall pair shortest path matrix is:\n");
for(i=1;i<=n;i++)
{
    for(j=1;j<=n;j++)
    {
        printf("%d\t",a[i][j]);
    }
    printf("\n\n");
}
}

```

```

int min(int x,int y)
{
    if(x<y)
    {
        return x;
    }
    else
    {
        return y;
    }
}

```

Output

Enter the no. of vertices: 4

Enter the cost matrix:

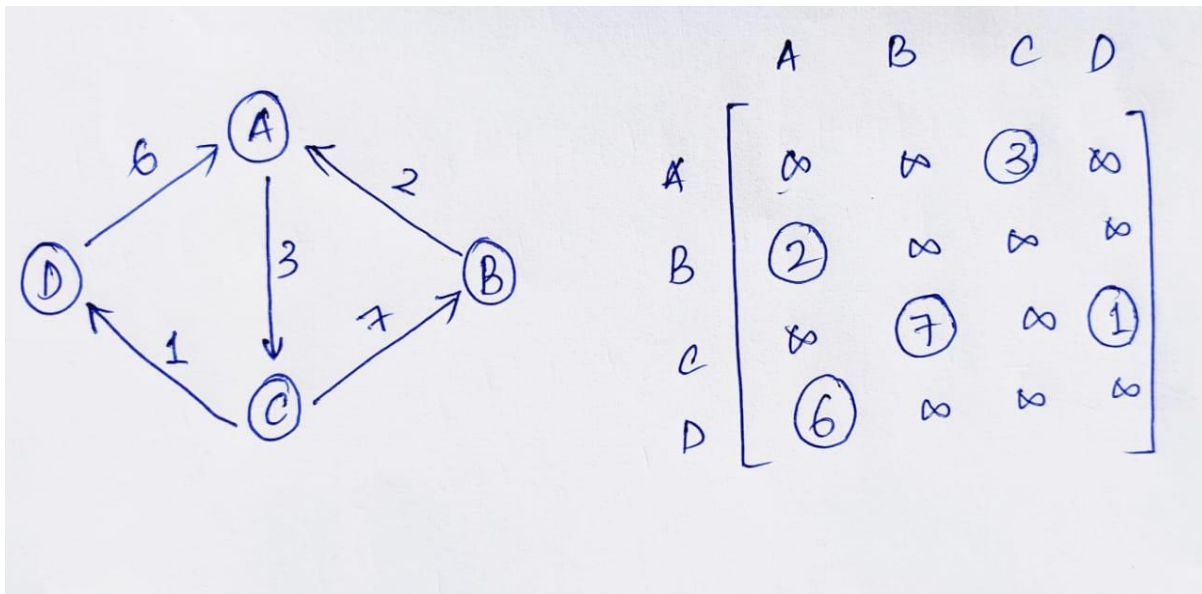
```

9999  9999      3  9999
      2  9999  9999  9999
9999      7  9999      1
      6  9999  9999  9999

```

All pair shortest path matrix is:

10	10	3	4
2	12	5	6
7	7	10	1
6	16	9	10



```

D:\codes\LAB 13.exe
enter the no. of vertices: 4

enter the cost matrix:
999 999 3 999
2 999 999 999
999 7 999 1
6 999 999 999

all pair shortest path matrix is:
10 10 3 4
2 12 5 6
7 7 10 1
6 16 9 10

-----
Process exited after 56.49 seconds with return value 0
Press any key to continue . . .
  
```

PROGRAM 14: Prim's algorithm.

Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
void prims();
int c[10][10],n;

int main()
{
    int i,j;
    printf("\n Enter the no. of vertices: ");
    scanf("%d",&n);
    printf("\n Enter the cost matrix:\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&c[i][j]);
        }
    }
    prims();
}

void prims()
{
    int i,j,u,v,min;
    int ne=0,mincost=0;
    int elec[10];
    for(i=1;i<=n;i++)
    {
        elec[i]=0;
```

```

}
elec[1]=1;
while(ne!=n-1)
{
    min=9999;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            if(elec[i]==1)
            {
                if(c[i][j]<min)
                {
                    min=c[i][j];
                    u=i;
                    v=j;
                }
            }
        }
    }
    if(elec[v]!=1)
    {
        printf("\n\t%c -----> %c = %d\n",u+65,v+65,min);
        elec[v]=1;
        ne=ne+1;
        mincost=mincost+min;
    }
    c[u][v]=c[v][u]=9999;
}
printf("\n\n\t Minimum Cost is =%d\n",mincost);
}

```

=====Output=====

Enter the no. of vertices: 6

Enter the cost matrix:

9999	3	9999	9999	6	5
3	9999	1	6	9999	4
9999	1	9999	6	9999	4
9999	6	6	9999	8	5
6	9999	9999	8	9999	2
5	4	4	5	2	9999

B-----> C = 1

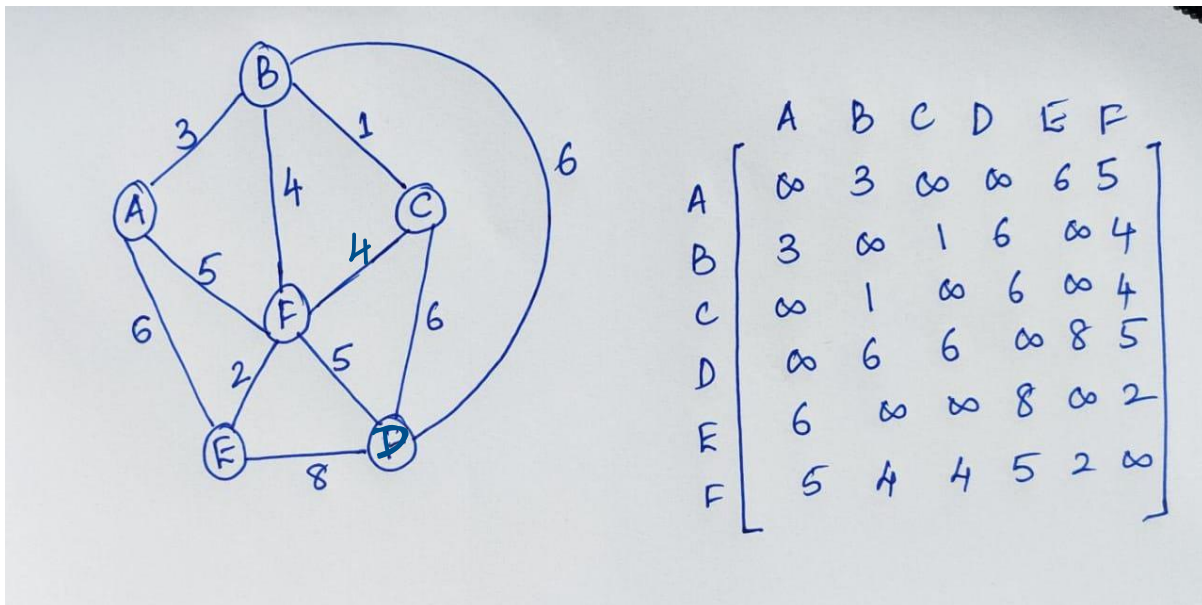
E-----> F = 2

A-----> B = 3

B-----> F = 4

F-----> D = 5

Minimum cost = 15



D:\codes\LAB 14.exe

Enter the no. of vertices: 6

Enter the cost matrix:

9999	3	9999	9999	6	5
3	9999	1	9999	9999	4
9999	1	9999	6	9999	4
9999	6	6	9999	8	5
6	9999	9999	8	9999	2
5	4	4	5	2	9999

A -----> B = 3

B -----> C = 1

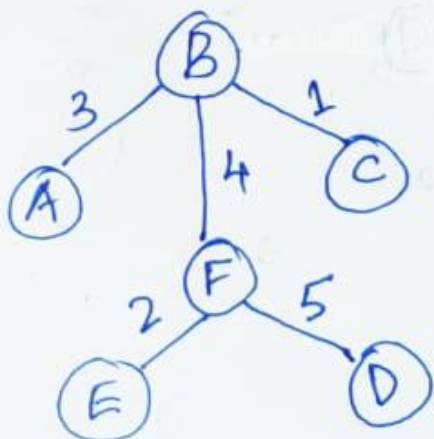
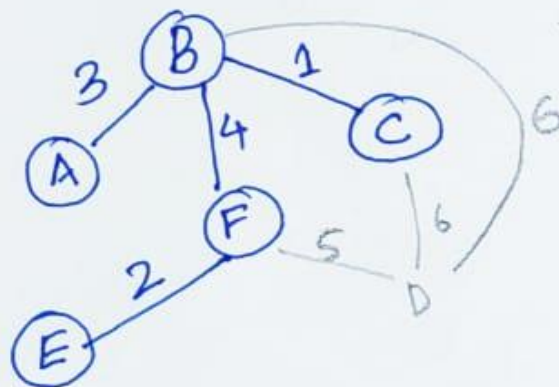
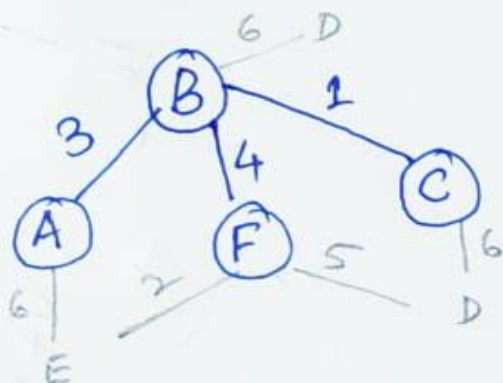
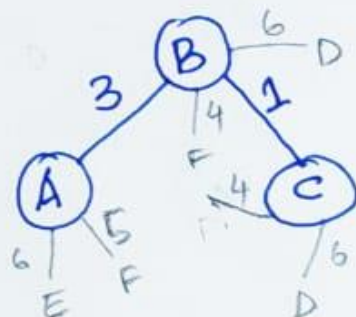
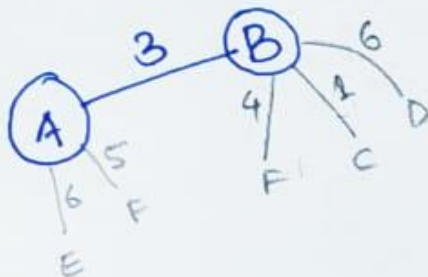
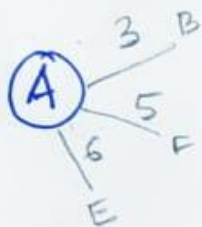
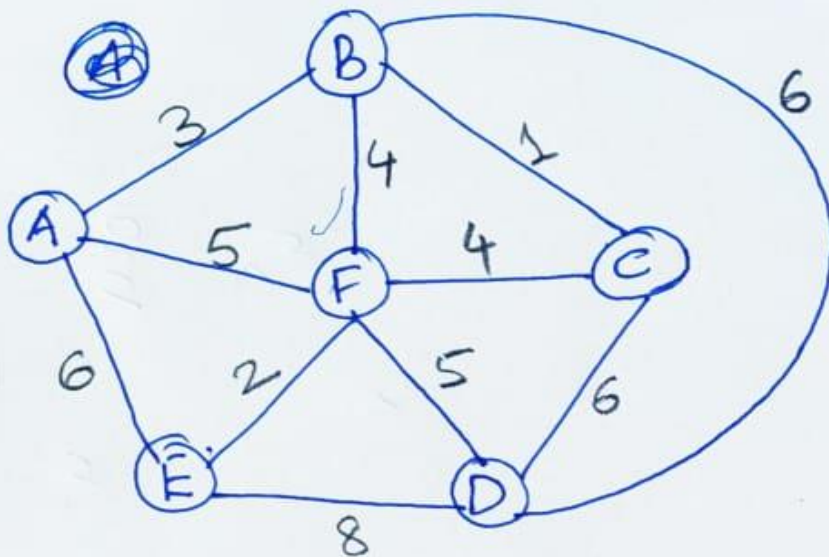
B -----> F = 4

F -----> E = 2

F -----> D = 5

Minimum Cost is =15

Process exited after 15.94 seconds with return value 0
Press any key to continue . . .



$B \leftrightarrow C = 1$
 $E \leftrightarrow F = 2$
 $A \leftrightarrow B = 3$
 $B \leftrightarrow F = 4$
 $F \leftrightarrow D = 5$

PROGRAM 15: Kruskals algorithm

Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm.

```
#include<stdio.h>
#include<conio.h>
void kruskals();
int c[10][10];
int n;

int main()
{
    int i,j;
    printf("\nEnter the no. of vertices: ");
    scanf("%d",&n);
    printf("\nEnter the cost matrix:\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&c[i][j]);
        }
    }
    kruskals();
}

void kruskals()
{
    int i,j,u,v,a,b,min;
    int ne=0,mincost=0;
    int parent[10];

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

```

parent[i]=0;
}
while(ne!=n-1)
{
min=9999;
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
if(c[i][j]<min)
{
min=c[i][j];
u=a=i;
v=b=j;
}
}
}
while(parent[u]!=0)
{
u=parent[u];
}
while(parent[v]!=0)
{
v=parent[v];
}
if(u!=v)
{
printf("\n\t%C <---> %C = %d\n",a+65,b+65,min);
parent[v]=u;
ne=ne+1;
mincost=mincost+min;
}
c[a][b]=c[b][a]=9999;
}
printf("\n Minimum cost is =%d",mincost);
}

```

=====Output=====

Enter the no. of vertices: 6

Enter the cost matrix:

9999	3	9999	9999	6	5
3	9999	1	9999	9999	4
9999	1	9999	6	9999	4
9999	6	6	9999	8	5
6	9999	9999	8	9999	2
5	4	4	5	2	9999

B-----> C = 1

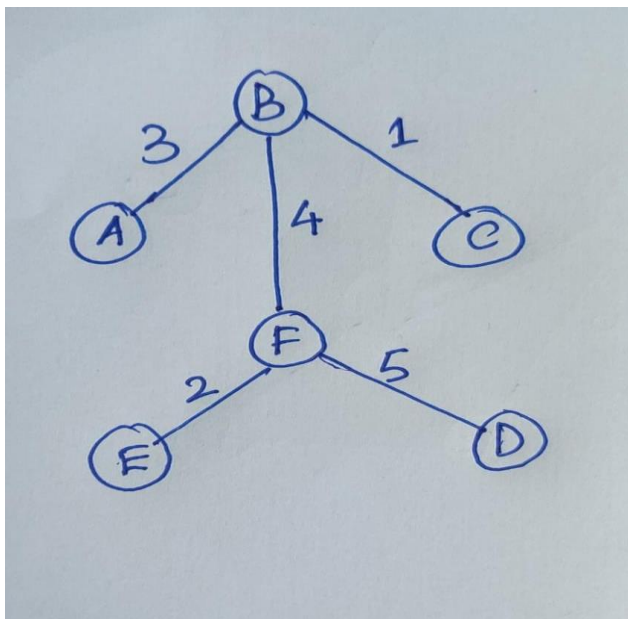
E-----> F = 2

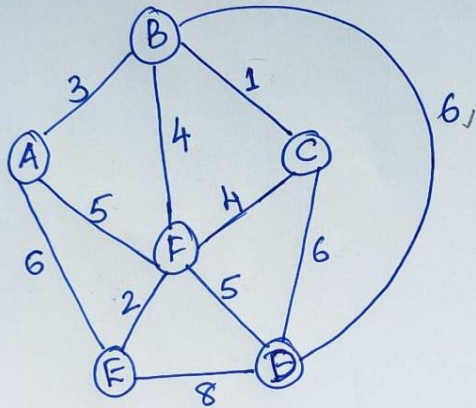
A-----> B = 3

B-----> F = 4

F-----> D = 5

Minimum cost = 15





	A	B	C	D	E	F
A		3	∞	∞	6	5
B	3		1	6	∞	4
C	∞	1		6	∞	4
D	∞	6	6		8	2
E	6	∞	∞	8		2
F	5	4	4	5	2	

Order:

BC	EF	AB	BF	FC	AF	FD	AE	CD	BD	DE
1	2	3	4	4	5	5	6	6	6	8
✓	✓	✓	✓	X	X	✓	-	-	-	-

D:\codes\LAB 15.exe

Enter the no. of vertices: 6

Enter the cost matrix:

```

9999      3      9999      9999      6      5
 3      9999      1      9999      9999      4
9999      1      9999      6      9999      4
9999      6      6      9999      8      5
 6      9999      9999      8      9999      2
 5      4      4      5      2      9999
  
```

B <---> C = 1

E <---> F = 2

A <---> B = 3

B <---> F = 4

D <---> F = 5

Minimum cost is =15

 Process exited after 21.38 seconds with return value 0
 Press any key to continue . . .

PROGRAM 16: Dijkstra's algorithm

From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.

Program:

```
#include <stdio.h>

int minDistance(int dist[], int sptSet[],int V)
{
    int min = 999, min_index;
    int v;
    for ( v = 0; v < V; v++)
        if (sptSet[v] == 0 && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}

int printSolution(int src,int dist[],int V )
{ int i;
    printf("\n\t Vertex \t\t Distance from Source\n\n");
    for (i = 0; i < V; i++)
        printf("\t%c ----> %c \t\t\t %d\n",src+65, i+65, dist[i]);
}

void dijkstra(int graph[10][10], int src,int V )
```

```

{
    int dist[V];
    int i,count,u,v;
    int sptSet[V];
    for ( i = 0; i < V; i++)
        dist[i] = 999, sptSet[i] = 0;

    dist[src] = 0;

    for ( count = 0; count < V - 1; count++) {

        u = minDistance(dist, sptSet,V);
        sptSet[u] = 1;

        for (v = 0; v < V; v++)

            if (!sptSet[v] && graph[u][v] && dist[u] != 999
                && dist[u] + graph[u][v] < dist[v])
                dist[v] = dist[u] + graph[u][v];
    }

    printSolution(src,dist,V);
}

int main()
{
    int i,j,V;

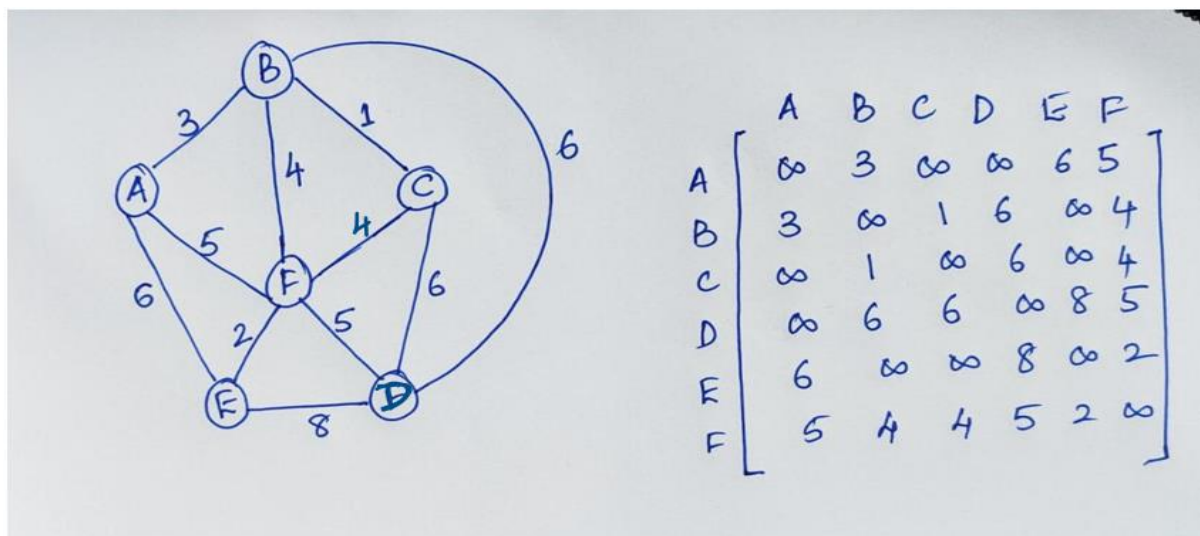
```

```

int graph[10][10];
printf("Enter number of vertices\n");
scanf("%d",&V);
printf("Enter adjacency matrix\n");
for(i=0;i<V;i++)
{
    for(j=0;j<V;j++)
    scanf("%d",&graph[i][j]);
}
for(i=0;i<V;i++){
    dijkstra(graph,i,V );
}
// dijkstra(graph, 0,V );

return 0;
}

```



Dijkstra's Algorithm

(Single Source Shortest Path)

* Relaxation:

$$\text{if } d(u) + c(u, v) < d(v)$$

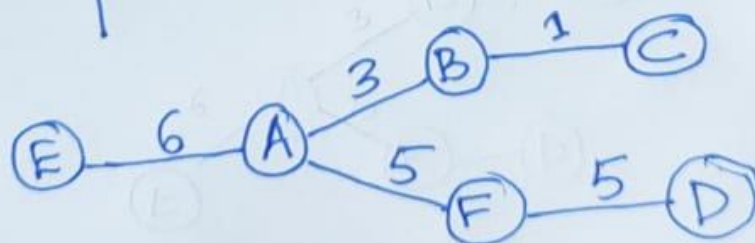
$$d(v) = d(u) + c(u, v)$$

where $u \rightarrow$ Source
 $v \rightarrow$ destination.

Let Source = A

A Source	Destination				
	B	C	D	E	F
(0)	∞	∞	∞	∞	∞
A	(3)	∞	∞	6	5
A, B	(3)	(4)	∞	6	5
A, B, C	(3)	(4)	10	6	(5)
A, B, C, F	(3)	(4)	10	(6)	(5)
A, B, C, F, E	(3)	(4)	(10)	(6)	(5)
A, B, C, F, E, D	(3)	(4)	(10)	(6)	(5)

Optimal



```
Select D:\codes\LAB 16.exe
Enter number of vertices
6
Enter adjacency matrix
9999 3 9999 9999 6 5
3 9999 1 9999 9999 4
9999 1 9999 6 9999 4
9999 6 6 9999 8 5
6 9999 9999 8 9999 2
5 4 4 5 2 9999

Vertex Distance from Source
A ----> A 0
A ----> B 3
A ----> C 4
A ----> D 10
A ----> E 6
A ----> F 5

Vertex Distance from Source
B ----> A 3
B ----> B 0
B ----> C 1
B ----> D 7
B ----> E 6
B ----> F 4

Vertex Distance from Source
C ----> A 4
C ----> B 1
C ----> C 0
C ----> D 6
C ----> E 6
C ----> F 4

Vertex Distance from Source
D ----> A 9
D ----> B 6
D ----> C 6
D ----> D 0
D ----> E 7
D ----> F 5

Vertex Distance from Source
E ----> A 6
E ----> B 6
E ----> C 6
E ----> D 7
E ----> E 0
E ----> F 2

Vertex Distance from Source
F ----> A 5
F ----> B 4
F ----> C 4
F ----> D 5
F ----> E 2
F ----> F 0

-----
Process exited after 3.589 seconds with return value 0
Press any key to continue . . .
```

PROGRAM 17: “Sum of Subsets” problem

Implement “Sum of Subsets” using Backtracking. “Sum of Subsets” problem: Find a subset of a given set $S = \{s_1, s_2, \dots, s_n\}$ of n positive 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 $\{1, 8\}$. A suitable message is to be displayed if the given problem instance doesn't have a solution.

Program:

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

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

```
static int total_nodes;
```

```
void printSubset(int A[], int size)
```

```
{
```

```
    printf("{");
```

```
    for(int i = 0; i < size; i++)
```

```
    {
```

```
        printf(" %d ", A[i]);
```

```
    }
```

```
    printf(" }\n");
```

```
}
```

```
void subset_sum(int s[], int t[], int s_size, int t_size, int sum, int ite, int const target_sum)
```

```
{
```

```

total_nodes++;

if( target_sum == sum )
{
    printSubset(t, t_size);

    if( ite + 1 < s_size && sum - s[ite] + s[ite+1] <= target_sum )
    {
        subset_sum(s, t, s_size, t_size-1, sum - s[ite], ite + 1, target_sum);
    }
    return;
}
else
{
    if( ite < s_size && sum + s[ite] <= target_sum )
    {
        for( int i = ite; i < s_size; i++ )
        {
            t[t_size] = s[i];

            if( sum + s[i] <= target_sum )
            {
                subset_sum(s, t, s_size, t_size + 1, sum + s[i], i + 1, target_sum);
            }
        }
    }
}

```

```

        }

    }

}

void bsort(int s[],int size)
{
    int i,j,temp;
    for (i = 0; i < size-1; i++)
    {
        for (j = 0; j < size-i-1; j++)
        {
            if (s[j] > s[j+1])
            {
                temp=s[j];
                s[j]=s[j+1];
                s[j+1]=temp;
            }
        }
    }
}

```

```

void generateSubsets(int s[], int size, int target_sum)
{
    int *tuplelet_vector = (int *)malloc(size * sizeof(int));

```

```

    int total = 0;
int i;
    bsort(s, size);

    for( int i = 0; i < size; i++ )
    {
        total += s[i];
    }

    if( s[0] <= target_sum && total >= target_sum )
    {

        subset_sum(s, tuple_vector, size, 0, 0, 0, target_sum);

    }

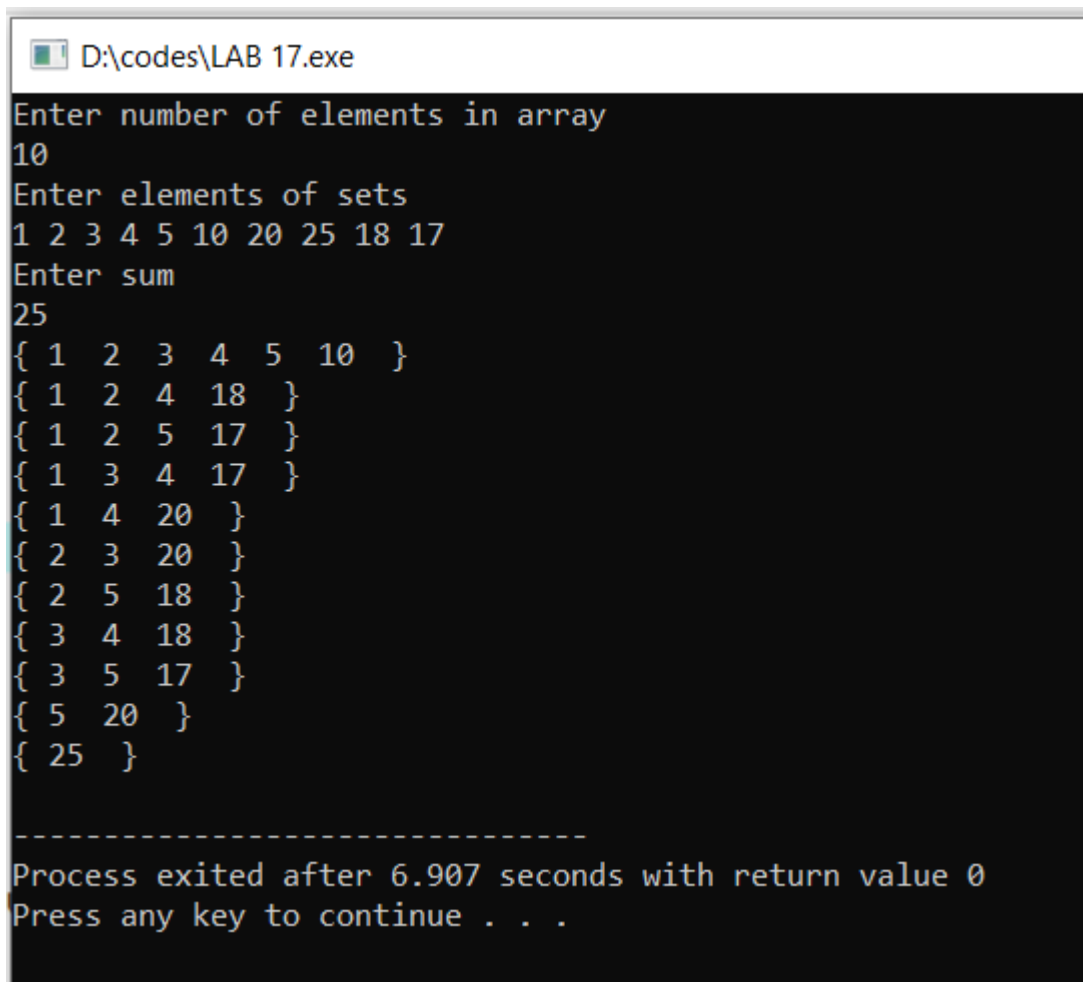
    free(tuple_vector);
}

int main()
{ int i,n;
    int sets[10] ;
    int target ;
    printf("Enter number of elements in array\n");
    scanf("%d",&n);
    printf("Enter elements of sets\n");

```

```
for(i=0;i<n;i++)
scanf("%d",&sets[i]);
printf("Enter sum\n");
scanf("%d",&target);
generateSubsets(sets,n, target);

}
```



```
D:\codes\LAB 17.exe
Enter number of elements in array
10
Enter elements of sets
1 2 3 4 5 10 20 25 18 17
Enter sum
25
{ 1  2  3  4  5  10  }
{ 1  2  4  18  }
{ 1  2  5  17  }
{ 1  3  4  17  }
{ 1  4  20  }
{ 2  3  20  }
{ 2  5  18  }
{ 3  4  18  }
{ 3  5  17  }
{ 5  20  }
{ 25  }

-----
Process exited after 6.907 seconds with return value 0
Press any key to continue . . .
```

PROGRAM 18: N-Queens Problem

Implement “N-Queens Problem” using Backtracking.

```
#define N 4
#include <stdbool.h>
#include <stdio.h>

void printSolution(int board[N][N])
{
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            printf(" %d ", board[i][j]);
        printf("\n");
    }
}

bool isSafe(int board[N][N], int row, int col)
{
    int i, j;
    for (i = 0; i < col; i++)
        if (board[row][i])
            return false;
    for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
        if (board[i][j])
            return false;
    for (i = row, j = col; j >= 0 && i < N; i++, j--)
        if (board[i][j])
            return false;

    return true;
}

bool solveNQUtil(int board[N][N], int col)
{
    if (col >= N)
```



```

        return true;
    for (int i = 0; i < N; i++)
    {
        if (isSafe(board, i, col))
        {
            board[i][col] = 1;
            if (solveNQUtil(board, col + 1))
                return true;
            board[i][col] = 0;
        }
    }
    return false;
}

bool solveNQ()
{
    int board[N][N] = { { 0, 0, 0, 0 },
                        { 0, 0, 0, 0 },
                        { 0, 0, 0, 0 },
                        { 0, 0, 0, 0 } };

    if (solveNQUtil(board, 0) == false) {
        printf("Solution does not exist");
        return false;
    }

    printSolution(board);
    return true;
}

int main()
{
    solveNQ();
    return 0;
}

```

```
D:\codes\LAB 18.exe
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0

-----
Process exited after 5.044 seconds with return value 0
Press any key to continue . . .
```

```
D:\codes\LAB 18.exe
1 0 0 0 0
0 0 0 1 0
0 1 0 0 0
0 0 0 0 1
0 0 1 0 0

-----
Process exited after 0.6572 seconds with return value 0
Press any key to continue . . .
```