

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



LAB REPORT

on

ANALYSIS AND DESIGN OF ALGORITHMS

Submitted by

VIJAYA VERMA(1BM20CS187)

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

May-2022 to July-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 work entitled “**Analysis and design of Algorithms**” carried out by **VIJAYA VERMA (1BM20CS187)**, who is bonafide student 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 Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a Analysis and Design of Algorithms - **(19CS4PCADA)** work prescribed for the said degree.

Mr. Vikranth B.M
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

Index Sheet

Sl. No.	Experiment Title	Page No.
1	Write a recursive program to Solve a) Towers-of-Hanoi problem b) To find GCD	5
2	Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N.	8
3	Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.	15
4	Sort a given set of N integer elements using Insertion Sort technique and compute its time taken.	20
5	Write program to do the following: a)Print all the nodes reachable from a given starting node in a digraph using BFS method. b) Check whether a given graph is connected or not using DFS method.	24
6	Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort	29
7	Write program to obtain the Topological ordering of vertices in a given digraph.	35
8	Sort a given set of N integer elements using Quick Sort technique and compute its time taken.	38
9	Implement Johnson Trotter algorithm to generate permutations.	40
10	Sort a given set of N integer elements using Heap Sort technique and compute its time taken.	45

11	Implement Warshall's algorithm using dynamic programming.	49
12	Implement 0/1 Knapsack problem using dynamic programming.	51
13	Implement All Pair Shortest paths problem using Floyd's algorithm.	54
14	Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.	56
15	Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm.	58
16	From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.	61

Course Outcome

CO1	Ability to analyze time complexity of Recursive and Non-recursive algorithms using asymptotic notations.
CO2	Ability to design efficient algorithms using various design techniques
CO3	Ability to apply the knowledge of complexity classes P, NP, and NP-Complete and prove certain problems are NP-Complete
CO4	Ability to conduct practical experiments to solve problems using an appropriate designing method and find time efficiency.

EXPERIMENT 1

Write a recursive program to

a. Solve Towers-of-Hanoi problem

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

void hanoi(int x, char from, char to, char aux)
{
    if(x==1)
        printf("Move Disk From %c to %c\n",from,to);
    else
    {
        hanoi(x-1,from,aux,to);
        printf("Move Disk From %c to %c\n",from,to);
        hanoi(x-1,aux,to,from);
    }
}

void main()
{
    int disk;
    int moves;
    clrscr();
    printf("Enter the number of disks you want to play with:");
    scanf("%d",&disk);
    moves=pow(2,disk)-1;
    printf("\n\nThe No of moves required is=%d \n",moves);
    hanoi(disk,'A','C','B');
    getch();
}
```

OUTPUT

```
Enter the number of disks you want to play with:3
```

```
The No of moves required is=7
```

```
Move Disk From A to C
```

```
Move Disk From A to B
```

```
Move Disk From C to B
```

```
Move Disk From A to C
```

```
Move Disk From B to A
```

```
Move Disk From B to C
```

```
Move Disk From A to C
```

```
...Program finished with exit code 0
```

```
Press ENTER to exit console. 
```

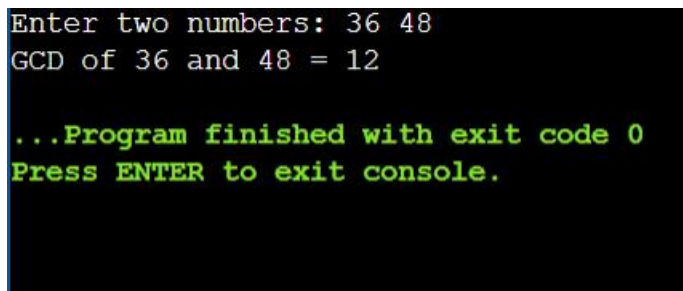
b. To find GCD

```
#include<stdio.h>

int gcd(int a, int b)
{
    if(b!=0)
        return gcd(b, a%b);
    else
        return a;
}

int main()
{
    int n1, n2, result;
    printf("Enter two numbers: ");
    scanf("%d %d",&n1,&n2);
    result = gcd(n1,n2);
    printf("GCD of %d and %d = %d",n1,n2,result);
    return 0;
}
```

OUTPUT



```
Enter two numbers: 36 48
GCD of 36 and 48 = 12

...Program finished with exit code 0
Press ENTER to exit console.
```

EXPERIMENT 2

Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N.

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

int bin_srch(int[] , int , int, int);
int lin_srch(int [], int, int, int);
int n , a[10000];
int main()
{
    int ch , key , search_status ,temp;
    clock_t end, start;
    unsigned long int i , j;
    while(1)
    {
        printf("\n1:Binary Search\t2: Linear Search\t3: Exit\n");
        printf("\nEnter your choice: \t");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1:
                n = 1000;
                while(n<=5000)
                {
```



```

for(i=0; i<n ; i++)
{
    //a[i] = random(1000);
    a[i] = i;
}
key = a[n-1];
start = clock();
search_status = bin_srch(a , 0 , n-1 , key);
if(search_status == -1)
    printf("\nKey not found");
else
    printf("key found at position %d", search_status);

//dummy loop to create delay
for(j = 0; j<500000000 ; j++)
{
    temp = 38/600;
}
end = clock();
printf("\nTime for n = %d is %f Secs " , n , (((double)(end-
start))/CLOCKS_PER_SEC));
n = n+ 1000;

}
break;

```

case 2 :

```

n = 1000;
while(n<=5000)

```

```

    {
        for(i = 0 ; i< n ; i++)
        {
            //a[i] = random(10000);

            a[i] = i;
        }
        key = a[n-1];
        start = clock();
        search_status = lin_srch(a, 0, n-1, key);
        if(search_status == -1)
            printf("\nKey Not Found");
        else
            printf("\nKey found at position %d" , search_status);
        //Dummy loop to create delay
        for(j = 0 ; j<500000000 ; j++)
        {
            temp = 38/600;
        }
        end = clock();
        printf("\nTime for n = %d is %f Secs" , n , (((double)(end-
            start))/CLOCKS_PER_SEC));

        n = n+1000;

    }

    break;

default:
    exit(0);
}

getchar();

```

```

    }
}

int bin_srch(int a[], int low , int high , int key)
{
    int mid;
    if(low > high)
    {
        return -1;
    }
    mid = (low + high)/2;
    if(key == a[mid]) {
        return mid;
    }
    if(key < a[mid])
    {
        return bin_srch(a , low , mid-1 , key);
    }
    else
    {
        return bin_srch(a , mid+1 , high , key);
    }
}

int lin_srch(int a[] , int i , int high , int key)
{
    if(i > high)
    {
        return -1;
    }

```

```

    }
    if(key == a[i])
    {
        return i;
    }
    else
    {
        return lin_srch(a, i+1 , high ,key);
    }
}

```

OUTPUT

```

1:Binary Search  2: Linear Search      3: Exit

Enter your choice:      1
key found at position 999
Time for n = 1000 is 0.803487 Secs  key found at position 1999
Time for n = 2000 is 0.800808 Secs  key found at position 2999
Time for n = 3000 is 0.805030 Secs  key found at position 3999
Time for n = 4000 is 0.803426 Secs  key found at position 4999
Time for n = 5000 is 0.804146 Secs
1:Binary Search  2: Linear Search      3: Exit

```

```
Enter your choice:      2

Key found at position 999
Time for n = 1000 is 0.803073 Secs
Key found at position 1999
Time for n = 2000 is 0.808012 Secs
Key found at position 2999
Time for n = 3000 is 0.806869 Secs
Key found at position 3999
Time for n = 4000 is 0.806740 Secs
Key found at position 4999
Time for n = 5000 is 0.806278 Secs
1: Binary Search  2: Linear Search      3: Exit

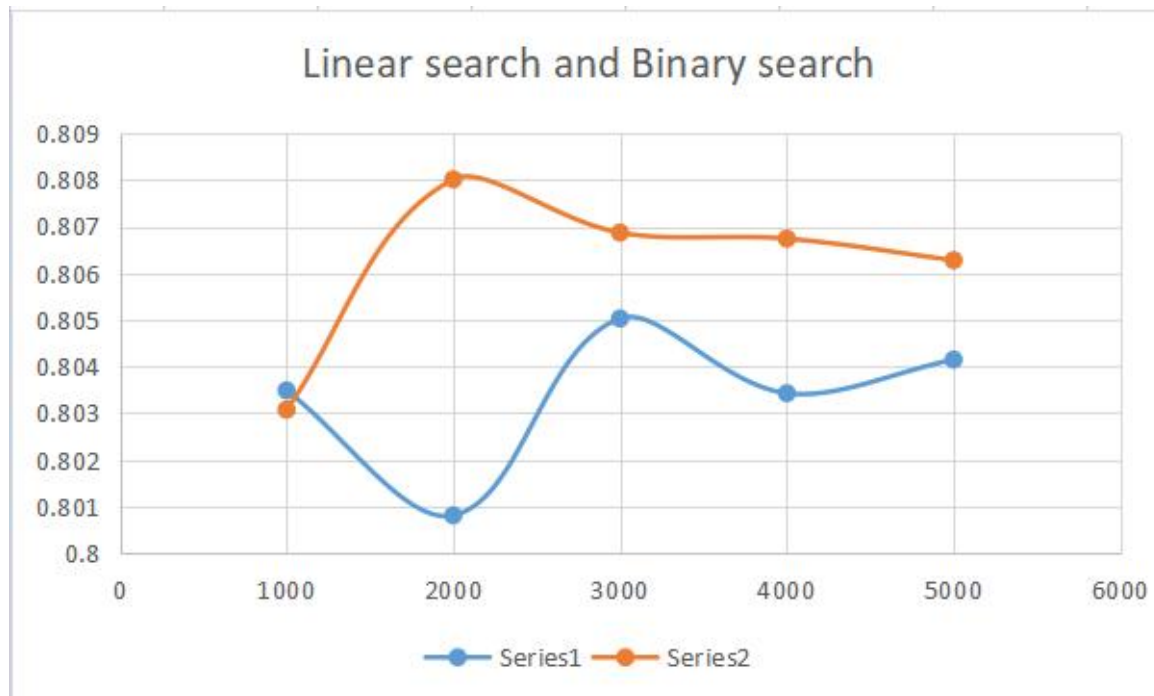
Enter your choice:      3

...Program finished with exit code 0
Press ENTER to exit console.
```

TABLE

N values	BS time	LS time
1000	0.803487	0.803073
2000	0.800808	0.808012
3000	0.80503	0.806869
4000	0.803426	0.80674
5000	0.804146	0.806278

GRAPH



EXPERIMENT 3

Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

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

void selsort(int n, int a[]);

void main()
{
    int a[15000],n,i,j,ch,temp;
    clock_t start,end;
    while(1)
    {
        printf("\n 1:For manual entry of N value and array elements ");
        printf("\n 2:To display time taken for sorting number of elements N in the range 500 to 14500");
        printf("\n 3:To exit ");
        printf("\n Enter your choice:");
        scanf("%d", &ch);

        switch(ch)
        {
            case 1: printf("\nEnter the number of elements: ");
                    scanf("%d",&n);
```

```

printf("\nEnter array elements: ");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
start=clock();
selsort(n,a);
end=clock();

printf("\nSorted array is: ");
for(i=0;i<n;i++)
printf("%d\t",a[i]);

printf("\n Time taken to sort %d numbers is %f Secs",n, (((double)(end-
start))/CLOCKS_PER_SEC));
break;

case 2:
n=500;
while(n<=14500)
{
for(i=0;i<n;i++)
{
a[i]=n-i;
}
start=clock();
selsort(n,a);

for(j=0;j<500000;j++)
{ temp=38/600;}

```



```
end=clock();
```

```
printf("\n Time taken to sort %d numbers is %f Secs ",n, (((double)(end-  
start))/CLOCKS_PER_SEC));
```

```
n=n+1000;
```

```
}
```

```
break;
```

```
case 3: exit(0);
```

```
}
```

```
getchar();
```

```
}
```

```
}
```

```
void selsort(int n,int a[])
```

```
{
```

```
int i,j,t,small,pos;
```

```
for(i=0;i< n-1;i++)
```

```
{
```

```
pos=i;
```

```
small=a[i];
```

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

```
{
```

```
if(a[j]<small)
```

```
{
```

```
small=a[j];
```

```
pos=j;
```

```
}
```

```
}
```

```
t=a[i];
```

```
a[i]=a[pos];  
a[pos]=t;  
}  
}
```

OUTPUT

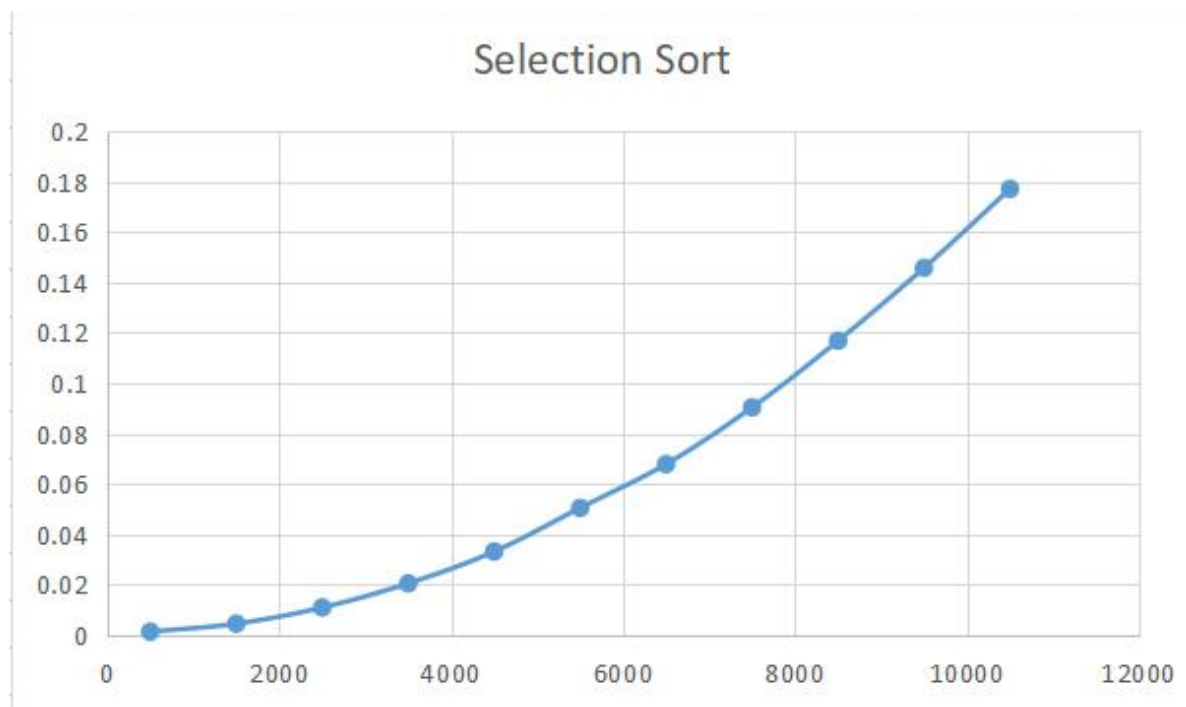
```
1:For manual entry of N value and array elements  
2:To display time taken for sorting number of elements N in the range 500 to 14500  
3:To exit  
Enter your choice:1  
  
Enter the number of elements: 4  
Enter array elements: 56 23 100 0  
Sorted array is: 0      23      56      100  
Time taken to sort 4 numbers is 0.000002 Secs
```

```
1:For manual entry of N value and array elements  
2:To display time taken for sorting number of elements N in the range 500 to 14500  
3:To exit  
Enter your choice:2  
  
Time taken to sort 500 numbers is 0.001452 Secs  
Time taken to sort 1500 numbers is 0.004609 Secs  
Time taken to sort 2500 numbers is 0.011029 Secs  
Time taken to sort 3500 numbers is 0.020602 Secs  
Time taken to sort 4500 numbers is 0.033234 Secs  
Time taken to sort 5500 numbers is 0.050553 Secs  
Time taken to sort 6500 numbers is 0.067920 Secs  
Time taken to sort 7500 numbers is 0.090423 Secs  
Time taken to sort 8500 numbers is 0.116872 Secs  
Time taken to sort 9500 numbers is 0.145797 Secs  
Time taken to sort 10500 numbers is 0.177129 Secs  
Time taken to sort 11500 numbers is 0.212022 Secs  
Time taken to sort 12500 numbers is 0.261201 Secs  
Time taken to sort 13500 numbers is 0.293148 Secs  
Time taken to sort 14500 numbers is 0.335410 Secs
```

TABLE

N values	Sort time
500	0.001452
1500	0.004609
2500	0.011029
3500	0.020602
4500	0.033234
5500	0.050553
6500	0.067920
7500	0.090423
8500	0.116872
9500	0.145797
10500	0.177129

GRAPH



EXPERIMENT 4

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

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

void insertionsort(int n, int a[])
{
    int i, key, j;
    for (i = 1; i < n; i++)
    {
        key = a[i];
        j = i - 1;
        while (j >= 0 && a[j] > key)
        {
            a[j + 1] = a[j];
            j = j - 1;
        }
        a[j + 1] = key;
    }
}

void main()
{
    int a[15000] , n , i , j , ch , temp;
    clock_t start , end;

    while(1)
    {
        printf("\n1:Insertion \n2:Display time taken \n3. Exit");
        printf("\n Enter your choice:");
```

```

scanf("%d", &ch);
switch(ch)
{
    case 1: printf("\nEnter the number of elements: ");
        scanf("%d", &n);
        printf("\nEnter elements: ");
        for(i = 0 ; i<n ; i++)
        {
            scanf("%d", &a[i]);
        }
        start = clock();
        insertionsort(n , a);
        end = clock();
        printf("\nSorted array is : ");
        for(i = 0 ; i< n ; i++)
            printf("%d\t", a[i]);
        printf("\n Time taken to sort %d number is %f Secs" , n , (((double)(end-
            start))/CLOCKS_PER_SEC));

        break;

    case 2 : n = 500;
        while(n <= 14500)
        {
            for(i = 0 ; i <n ; i++)
            {
                a[i] = n - i;
            }
            start = clock();
            insertionsort(n , a);
            for(j = 0 ; j < 500000; j++)
            {
                temp = 38/600;
            }
            end = clock();
            printf("\n Time taken to sort %d number is %f Secs" , n , (((double)(end-
start))/CLOCKS_PER_SEC));
            n = n + 1000;
        }
    }
}

```

```

        }

        break;
    case 3: exit(0);
    }
    getchar();
}
}

```

OUTPUT

```

1:Insertion
2:Display time taken
3. Exit
Enter your choice:1

Enter the number of elements: 5
Enter elements: 5 1 0 2 9

Sorted array is : 0      1      2      5      9
Time taken to sort 5 number is 0.000002 Secs
1:Insertion
2:Display time taken
3. Exit
Enter your choice:2

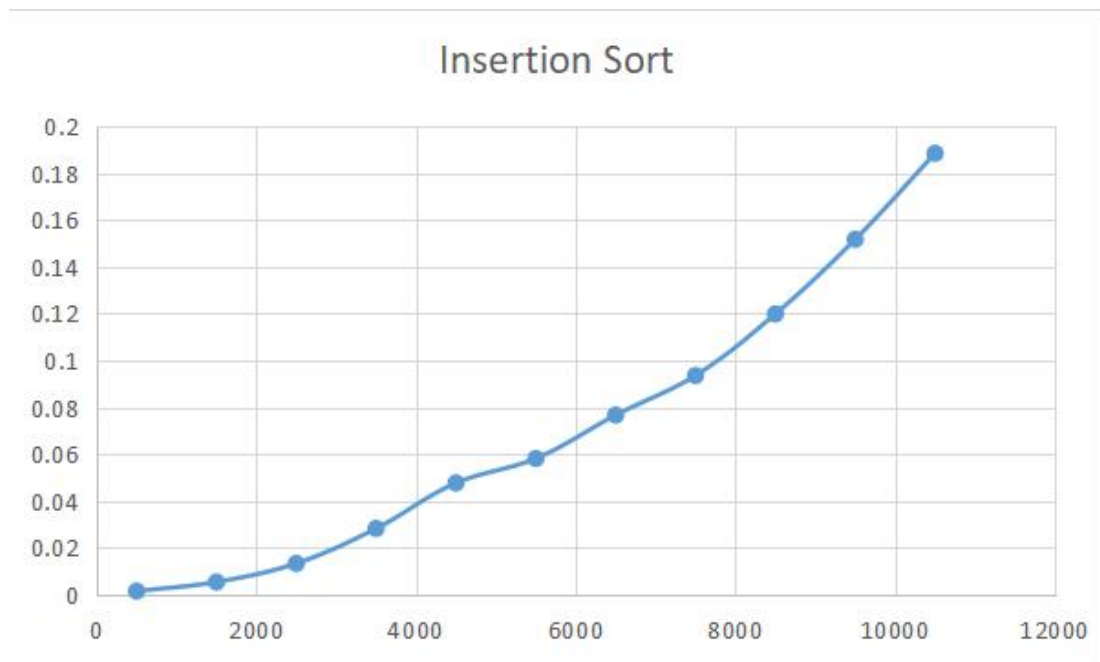
Time taken to sort 500 number is 0.001571 Secs
Time taken to sort 1500 number is 0.005472 Secs
Time taken to sort 2500 number is 0.013418 Secs
Time taken to sort 3500 number is 0.028343 Secs
Time taken to sort 4500 number is 0.047748 Secs
Time taken to sort 5500 number is 0.058195 Secs
Time taken to sort 6500 number is 0.076723 Secs
Time taken to sort 7500 number is 0.093613 Secs
Time taken to sort 8500 number is 0.119916 Secs
Time taken to sort 9500 number is 0.151756 Secs
Time taken to sort 10500 number is 0.188468 Secs
Time taken to sort 11500 number is 0.237878 Secs
Time taken to sort 12500 number is 0.258068 Secs
Time taken to sort 13500 number is 0.304234 Secs
Time taken to sort 14500 number is 0.353789 Secs

```

TABLE

N values	Sort time
500	0.001571
1500	0.005472
2500	0.013418
3500	0.028343
4500	0.047748
5500	0.058195
6500	0.076723
7500	0.093613
8500	0.119916
9500	0.151756
10500	0.188468

GRAPH



EXPERIMENT 5

Write program to do the following:

a. Print all the nodes reachable from a given starting node in a digraph using BFS method.

```
#include<stdio.h>
#include<conio.h>
int a[10][10],n;
void bfs(int);
void main()
{
    int i,j,src;
    //clrscr();
    printf("\nenter the no of nodes:\t");
    scanf("%d",&n);
    printf("\nenter the adjacency matrix:\n ");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("\nenter the source node:\t");
    scanf("%d",&src);
    bfs(src);
    getch();
}
```



```

}

void bfs(int src)
{
int q[10],f=0,r=-1,vis[10],i,j;
for(j=1;j<=n;j++)
{
vis[j]=0;
}
vis[src]=1;
r=r+1;
q[r]=src;
while(f<=r)
{
i=q[f];
f=f+1;
for(j=1;j<=n;j++)
{
if(a[i][j]==1 && vis[j]!=1)
{
vis[j]=1;
r=r+1;
q[r]=j;
}
}
}
for(j=1;j<=n;j++)
{
if(vis[j]!=1)

```

```
{  
printf("\nnode %d is not reachable\n",j);  
}  
else  
{  
printf("\nnode %d is reachable\n",j);  
}  
}  
}
```

OUTPUT

```
enter the no of nodes: 4  
  
enter the adjacency matrix:  
0 1 1 0  
0 0 0 0  
0 0 0 1  
0 1 0 0  
  
enter the source node: 1  
  
node 1 is reachable  
  
node 2 is reachable  
  
node 3 is reachable  
  
node 4 is reachable
```

b. Check whether a given graph is connected or not using DFS method.

```
#include<stdio.h>
#include<conio.h>
int a[10][10],n,vis[10];
int dfs(int);
void main()
{
    int i,j,src,ans;
    for(j=1;j<=n;j++)
    {
        vis[j]=0;
    }
    printf("\nenter the no of nodes:\t");
    scanf("%d",&n);
    printf("\nenter the adjacency matrix:\n ");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }

    printf("\nenter the source node:\t");
    scanf("%d",&src);
    ans=dfs(src);
    if(ans==1){
        printf("graph is connected");
    }
    else{
        printf("graph is not connected");
    }
    getch();
}
int dfs(int src)
```

```

{
    int j;
    vis[src]=1;
    for(j=1;j<=n;j++){
        if(a[src][j]==1 && vis[j]!=1){
            dfs(j);
        }
    }
    for(j=1;j<=n;j++)
    {
        if(vis[j]!=1){
            return 0;
        }
    }
    return 1;
}

```

OUTPUT

```

enter the no of nodes:  4

enter the adjacency matrix:
0 1 1 0
0 0 0 0
0 0 0 1
0 1 0 0

enter the source node:  1
graph is connected

...Program finished with exit code 0
Press ENTER to exit console.

```

EXPERIMENT 6

Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

```
#include<stdio.h>
#include<time.h>
#include<stdlib.h> /* To recognise exit function when compiling with gcc*/
void split(int[],int,int);
void combine(int[],int,int,int);
void main()
{
    int a[15000],n, i,j,ch, temp;
    clock_t start,end;

    while(1)
    {
        printf("\n1:For manual entry of N value and array elements");
        printf("\n2:To display time taken for sorting number of elements N in the range 500 to 14500");
        printf("\n3:To exit");
        printf("\nEnter your choice:");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1: printf("\nEnter the number of elements: ");
                    scanf("%d",&n);
                    printf("\nEnter array elements: ");
                    for(i=0;i<n;i++)
```

```

        {
            scanf("%d",&a[i]);
        }
        start=clock();
        split(a,0,n-1);
        end=clock();
        printf("\nSorted array is: ");
        for(i=0;i<n;i++)
            printf("%d\t",a[i]);
        printf("\n Time taken to sort %d numbers is %f Secs",n, (((double)(end-
start))/CLOCKS_PER_SEC));
        break;
    case 2:
        n=500;
        while(n<=14500) {
            for(i=0;i<n;i++)
            {
                //a[i]=random(1000);
                a[i]=n-i;
            }
            start=clock();
            split(a,0,n-1);
            //Dummy loop to create delay
            for(j=0;j<500000;j++){ temp=38/600;}
            end=clock();
            printf("\n Time taken to sort %d numbers is %f Secs",n, (((double)(end-
start))/CLOCKS_PER_SEC));
            n=n+1000;
        }
        break;

```

```

    case 3: exit(0);
    }
    getchar();
    }
}

```

```

void split(int a[],int low,int high)
{
    int mid;
    if(low<high)
    {
        mid=(low+high)/2;
        split(a,low,mid);
        split(a,mid+1,high);
        combine(a,low,mid,high);
    }
}

```

```

void combine(int a[],int low,int mid,int high)
{
    int c[15000],i,j,k;
    i=k=low;
    j=mid+1;
    while(i<=mid&& j<=high)
    {
        if(a[i]<a[j])
        {
            c[k]=a[i];
            ++k;
            ++i;

```

```

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

```



```
}
```

OUTPUT

```
1:For manual entry of N value and array elements
2:To display time taken for sorting number of elements N in the range 500 to 14500
3:To exit
Enter your choice:1

Enter the number of elements: 4

Enter array elements: 45 11 00 2

Sorted array is: 0      2      11      45
Time taken to sort 4 numbers is 0.000064 Secs
```

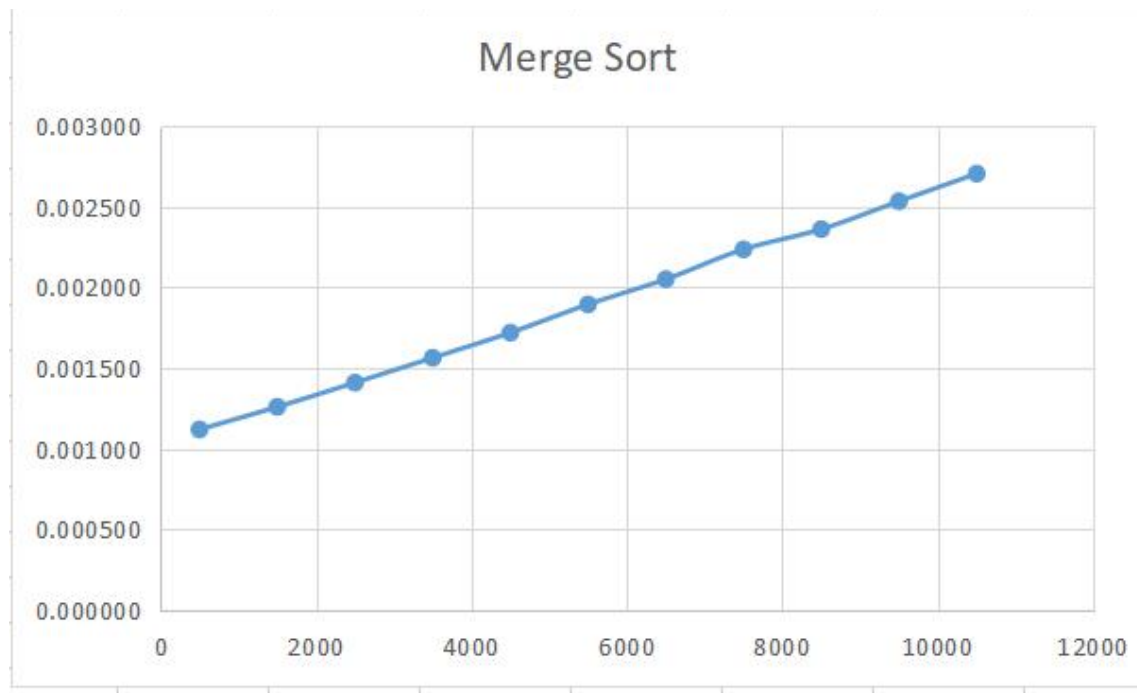
```
1:For manual entry of N value and array elements
2:To display time taken for sorting number of elements N in the range 500 to 14500
3:To exit
Enter your choice:2

Time taken to sort 500 numbers is 0.001120 Secs
Time taken to sort 1500 numbers is 0.001260 Secs
Time taken to sort 2500 numbers is 0.001410 Secs
Time taken to sort 3500 numbers is 0.001564 Secs
Time taken to sort 4500 numbers is 0.001720 Secs
Time taken to sort 5500 numbers is 0.001895 Secs
Time taken to sort 6500 numbers is 0.002051 Secs
Time taken to sort 7500 numbers is 0.002236 Secs
Time taken to sort 8500 numbers is 0.002361 Secs
Time taken to sort 9500 numbers is 0.002534 Secs
Time taken to sort 10500 numbers is 0.002705 Secs
Time taken to sort 11500 numbers is 0.002852 Secs
Time taken to sort 12500 numbers is 0.003023 Secs
Time taken to sort 13500 numbers is 0.003199 Secs
Time taken to sort 14500 numbers is 0.003378 Secs
```

TABLE

N values	Sort time
500	0.001120
1500	0.001260
2500	0.001410
3500	0.001564
4500	0.001720
5500	0.001895
6500	0.002051
7500	0.002236
8500	0.002361
9500	0.002534
10500	0.002705

GRAPH



EXPERIMENT 7

Write program to obtain the Topological ordering of vertices in a given digraph.

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

void source_removal(int n, int a[10][10]) {
    int i, j, k, u, v, top, s[10], t[10], indeg[10], sum;
    for(i = 0; i < n; i++) {
        sum = 0;
        for(j = 0; j < n; j++) {
            sum += a[j][i];
        }
        indeg[i]=sum;
    }
    top = -1;
    for(i=0;i<n;i++) {
        if(indeg[i] == 0) {
            s[++top] = i;
        }
    }
    k = 0;
    while(top != -1) {
        u = s[top--];
        t[k++] = u;
        for(v = 0; v < n; v++) {
```

```

        if(a[u][v] == 1) {
            indeg[v] = indeg[v] - 1;
            if(indeg[v] == 0)
                s[++top] = v;
        }
    }
}

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

}

void main() {
    int i, j, a[10][10], n;
    printf("Enter number of nodes\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]);
        }
    }
    source_removal(n,a);
    getch();
}

```

OUTPUT

```
Enter number of nodes
4
Enter the adjacency matrix
0 1 1 0
0 0 0 0
0 0 0 1
0 1 0 0
0
2
3
1

...Program finished with exit code 0
Press ENTER to exit console.
```

EXPERIMENT 8

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

```
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
void quicksort(int number[5000],int first,int last){
int i, j, pivot, temp;
if(first<last){
pivot=first;
i=first;
j=last;
while(i<j){
for(int x=0;x<100000;x++);
while(number[i]<=number[pivot]&& i<last)
i++;
while(number[j]>number[pivot])
j--;
if(i<j){
temp=number[i];
number[i]=number[j];
number[j]=temp;
}}
temp=number[pivot];
number[pivot]=number[j];
number[j]=temp;
quicksort(number,first,j-1);
quicksort(number,j+1,last);
}}
int main(){
clock_t start,end;
int i, count, number[5000];
printf("No. of elements: ");
```

```

scanf("%d",&count);
printf("Enter %d elements: ", count);
for(i=0;i<count;i++)
number[i] = rand()%1000;
start = clock();
quicksort(number,0,count-1);
end = clock();
printf("Order of Sorted elements: ");
for(i=0;i<count;i++)
printf(" %d",number[i]);
printf("\nSeconds taken %lf",(double)(end-start)/CLOCKS_PER_SEC);
return 0;
}

```

OUTPUT

```

No. of elements: 10
Enter 10 elements:
Order of Sorted elements:  335 383 386 421 492 649 777 793 886 915
Seconds taken 0.002105

...Program finished with exit code 0
Press ENTER to exit console.

```

EXPERIMENT 9

Implement Johnson Trotter algorithm to generate permutations

```
#include <stdio.h>
#include <stdlib.h>
int flag = 0;
int swap(int *a,int *b)
{
    int t = *a;
    *a = *b;
    *b = t;
}
int search(int arr[],int num,int mobile)
{
    int g;
    for(g=0;g<num;g++)
    {
        if(arr[g] == mobile)
        {
            return g+1;
        }
        else
        {
            flag++;
        }
    }
    return -1;
}
```



```

}
int find_Moblie(int arr[],int d[],int num)
{
    int mobile = 0;
    int mobile_p = 0;
    int i;
    for(i=0;i<num;i++)
    {
        if((d[arr[i]-1] == 0) && i != 0)
        {
            if(arr[i]>arr[i-1] && arr[i]>mobile_p)
            {
                mobile = arr[i];
                mobile_p = mobile;
            }
            else
            {
                flag++;
            }
        }
        else if((d[arr[i]-1] == 1) & i != num-1)
        {
            if(arr[i]>arr[i+1] && arr[i]>mobile_p)
            {
                mobile = arr[i];
                mobile_p = mobile;
            }
            else
            {
                flag++;
            }
        }
    }
}

```

```

    }
    }
    else
    {
        flag++;
    }
}

if((mobile_p == 0) && (mobile == 0))
    return 0;
else
    return mobile;
}

void permutations(int arr[],int d[],int num)
{
    int i;
    int mobile = find_Moblie(arr,d,num);
    int pos = search(arr,num,mobile);
    if(d[arr[pos-1]-1]==0)
        swap(&arr[pos-1],&arr[pos-2]);
    else
        swap(&arr[pos-1],&arr[pos]);
    for(int i=0;i<num;i++)
    {
        if(arr[i] > mobile)
        {
            if(d[arr[i]-1]==0)
                d[arr[i]-1] = 1;
            else
                d[arr[i]-1] = 0;
        }
    }
}

```

```

    }
    for(i=0;i<num;i++)
    {
        printf(" %d ",arr[i]);
    } }
int factorial(int k)
{
    int f = 1;
    int i = 0;
    for(i=1;i<k+1;i++)
    {
        f = f*i;
    }
    return f;
}
int main()
{
    int num = 0;
    int i;
    int j;
    int z = 0;
    printf("Johnson trotter algorithm to find all permutations of given numbers \n");
    printf("Enter the number\n");
    scanf("%d",&num);
    int arr[num],d[num];
    z = factorial(num);
    printf("total permutations = %d",z);
    printf("\nAll possible permutations are: \n");
    for(i=0;i<num;i++)
    {

```

```

d[i] = 0;
arr[i] = i+1;
printf(" %d ",arr[i]);
}
printf("\n");
for(j=1;j<z;j++)
{
permutations(arr,d,num);
printf("\n");
} return 0;
}

```

OUTPUT

```

Johnson trotter algorithm to find all permutations of given numbers
Enter the number
3
total permutations = 6
All possible permutations are:
1  2  3
1  3  2
3  1  2
3  2  1
2  3  1
2  1  3

...Program finished with exit code 0
Press ENTER to exit console.

```

EXPERIMENT 10

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

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <math.h>
void swap(int *,int *);
void heapify(int [],int,int);
void heapSort(int[], int);
int main()
{
    int a[15000], n, i, j, ch, temp;
    clock_t start, end;
    while (1)
    {
        printf("\n1:FOR MANUAL ENTRY");
        printf("\n2:DISPLAY TIME TAKEN TO SORT ELEMENTS FROM
RANGE 500 TO 15000");
        printf("\n3:EXIT");
        printf("\nEnter YOUR CHOICE:");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1:
                printf("\nEnter NUMBER OF ARRAY ELEMENTS: ");
                scanf("%d", &n);
                printf("\nEnter ARRAY ELEMENTS: ");
                for (i = 0; i < n; i++)
                {
                    scanf("%d", &a[i]);
                }
                start = clock();
                heapSort(a, n);
                end = clock();
                printf("\nSORTED ARRAY IS: ");
                for (i = n-1; i >= 0; i--)
```

```

        printf("%d\t", a[i]);
        printf("\n TIME TAKEN TO SORT %d NUMBERS IS %f SECS", n,
(((double)(end - start)) / CLOCKS_PER_SEC));
        break;
    case 2:
        n = 500;
        while (n <= 14500)
        {
            for (i = 0; i < n; i++)
            {
                //a[i]=rand()%n;
                a[i] = n - i;
            }
            start = clock();
            heapSort(a, n);
            for (j = 0; j < 500000; j++)
            {
                temp = 38 / 600;
            }
            end = clock();
            printf("\n TIME TAKEN TO SORT %d NUMBERS IS %f SECS",
n, (((double)(end - start)) / CLOCKS_PER_SEC));
            n = n + 1000;
        }
        break;
    case 3:
        exit(0);
    }
    getchar();
}

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void heapify(int arr[], int n, int i)
{

```

```

    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;
    if (left < n && arr[left] > arr[largest])
        largest = left;
    if (right < n && arr[right] > arr[largest])
        largest = right;
    if (largest != i)
    {
        swap(&arr[i], &arr[largest]);
        heapify(arr, n, largest);
    }
}
void heapSort(int arr[], int n)
{
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);
    for (int i = n - 1; i >= 0; i--)
    {
        swap(&arr[0], &arr[i]);
        heapify(arr, i, 0);
    }
}

```

OUTPUT

```

1:FOR MANUAL ENTRY
2:DISPLAY TIME TAKEN TO SORT ELEMENTS FROM RANGE 500 TO 15000
3:EXIT
ENTER YOUR CHOICE:1

ENTER NUMBER OF ARRAY ELEMENTS: 4

ENTER ARRAY ELEMENTS: 12 10 89 4

SORTED ARRAY IS: 89      12      10      4
TIME TAKEN TO SORT 4 NUMBERS IS 0.000002 SECS

```

ENTER YOUR CHOICE:2

TIME TAKEN TO SORT	500 NUMBERS IS	0.000861 SECS
TIME TAKEN TO SORT	1500 NUMBERS IS	0.001063 SECS
TIME TAKEN TO SORT	2500 NUMBERS IS	0.001280 SECS
TIME TAKEN TO SORT	3500 NUMBERS IS	0.001528 SECS
TIME TAKEN TO SORT	4500 NUMBERS IS	0.001769 SECS
TIME TAKEN TO SORT	5500 NUMBERS IS	0.002028 SECS
TIME TAKEN TO SORT	6500 NUMBERS IS	0.002294 SECS
TIME TAKEN TO SORT	7500 NUMBERS IS	0.002586 SECS
TIME TAKEN TO SORT	8500 NUMBERS IS	0.002848 SECS
TIME TAKEN TO SORT	9500 NUMBERS IS	0.003116 SECS
TIME TAKEN TO SORT	10500 NUMBERS IS	0.003368 SECS
TIME TAKEN TO SORT	11500 NUMBERS IS	0.003618 SECS
TIME TAKEN TO SORT	12500 NUMBERS IS	0.003935 SECS
TIME TAKEN TO SORT	13500 NUMBERS IS	0.004109 SECS
TIME TAKEN TO SORT	14500 NUMBERS IS	0.004427 SECS

EXPERIMENT 11

Implement Warshall's algorithm using dynamic programming.

```
#include<stdio.h>
int a[30][30];

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

int main(){
    int n;
    printf("Enter no of vertices: \n");
    scanf("%d",&n);

    printf("Enter adjacency matrix: \n");
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            scanf("%d",&a[i][j]);
    warshall(n);
    printf("Transitive Closure: \n");
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++)
            printf("%d ",a[i][j]);
        printf("\n");
    }
}
```

OUTPUT

```
Enter no of vertices:
4
Enter adjacency matrix:
0 5 9999 10
9999 0 3 9999
9999 9999 0 1
9999 9999 9999 0
Transitive Closure:
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
```

EXPERIMENT 12

Implement 0/1 Knapsack problem using dynamic programming.

```
#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 num 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 num 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
```

EXPERIMENT 13

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

```
#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 num 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 num 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

...Program finished with exit code 255
Press ENTER to exit console.

```

EXPERIMENT 14

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

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

void prims();
int c[10][10], n;

void main() {
    int i, j;
    printf("\n Enter the num of vertices: \t");
    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();
    getch();
}
```

```
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%d-----> %d = %d\n", u, v, min);
    elec[v] = 1;
    ne = ne + 1;
    mincost = mincost + min;
}
c[u][v] = c[v][u] = 9999;
}
printf("\n mincost = %d", mincost);
}

```

OUTPUT

```

Enter the num 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 0000 8 5
6 9999 9999 8 9999 2
5 4 4 5 2 9999

1-----> 2 = 3

2-----> 3 = 1

2-----> 6 = 4

6-----> 5 = 2

6-----> 4 = 5

mincost = 15

...Program finished with exit code 255
Press ENTER to exit console.

```

EXPERIMENT 15

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],n;
void main()
{
    int i,j;
    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",&c[i][j]);
        }
    }
    kruskals();
    getch();
}

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%d----->%d=%d\n",a,b,min);
parent[v]=u;
ne=ne+1;
mincost=mincost+min;
}
c[a][b]=c[b][a]=9999;
}

```

```
printf("\nmincost=%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  
  
2----->3=1  
  
5----->6=2  
  
1----->2=3  
  
2----->6=4  
  
4----->6=5  
  
mincost=15  
  
...Program finished with exit code 255  
Press ENTER to exit console.
```

EXPERIMENT 16

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

```
#include<stdio.h>
#include<conio.h>
void dijkstras();
int c[10][10],n,src;
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",&c[i][j]);
        }
    }
    printf("\nenter the source node:\t");
    scanf("%d",&src);
    dijkstras();
    getch();
}

void dijkstras()
{
    int vis[10],dist[10],u,j,count,min;
    for(j=1;j<=n;j++)
    {
        dist[j]=c[src][j];
    }
    for(j=1;j<=n;j++)
    {

```

```

    vis[j]=0;
}
dist[src]=0;
vis[src]=1;
count=1;
while(count!=n)
{
    min=9999;
    for(j=1;j<=n;j++)
    {
        if(dist[j]<min&&vis[j]!=1)
        {
            min=dist[j];
            u=j;
        }
    }
    vis[u]=1;
    count++;
    for(j=1;j<=n;j++)
    {
        if(min+c[u][j]<dist[j]&&vis[j]!=1)
        {
            dist[j]=min+c[u][j];
        }
    }
}
printf("\nthe shortest distance is:\n");
for(j=1;j<=n;j++)
{
    printf("\n%d----->%d=%d",src,j,dist[j]);
}
}

```

OUTPUT

```
enter the no of vertices:      5

enter the cost matrix:
9999 3 9999 7 9999
3 9999 4 2 9999
9999 4 9999 5 6
7 2 5 9999 4
9999 9999 6 4 9999

enter the source node:  1

the shortest distance is:

1----->1=0
1----->2=3
1----->3=7
1----->4=5
1----->5=9
```