VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

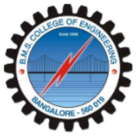
LAB REPORT
on

Analysis and Design of Algorithms

Submitted by

ARAVIND ANAND (1BM21CS030)

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
June-2023 to September-2023

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 **ARAVIND ANAND (1BM21CS030),** who is a 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 academic semester June-2023 to September-2023. The Lab report has been approved as it satisfies the academic requirements in respect of a **Analysis and Design of Algorithms (22CS4PCADA)** work prescribed for the said degree.

Name of the Lab-In charge: Sunayana S	Dr. Jyothi S Nayak
Designation: Assistant Professor	Professor and Head
Department of CSE	Department of CSE
BMSCE, Bengaluru	BMSCE, Bengaluru

Index Sheet

Lab Program No.	Program Details
1	Write program to do the following: a. Print all the nodes reachable from a given starting node in a digraph using the BFS method. b. Check whether a given graph is connected or not using the DFS method.
2	Write a program to obtain the Topological ordering of vertices in a given digraph.
3	Implement Johnson Trotter algorithm to generate permutations.
4	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.
5	Sort a given set of N integer elements using Quick Sort technique and compute its time taken.
6	Sort a given set of N integer elements using Heap Sort technique and compute its time taken.
7	Implement 0/1 Knapsack problem using dynamic programming.
8	Implement All Pair Shortest paths problem using Floyd's algorithm.
9	Find Minimum Cost Spanning Tree of a given undirected graph using Prim's and Kruskal's algorithm.
10	From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.
11	Implement "N-Queens Problem" using Backtracking.

Course Outcome

CO1	Analyze time complexity of Recursive and Non-recursive algorithms using asymptotic notations.
CO2	Apply various design techniques for the given problem.
CO3	Apply the knowledge of complexity classes P, NP, and NP-Complete and prove certain problems are NP-Complete
CO4	Design efficient algorithms and conduct practical experiments to solve problems.

Program-1

Write program to do the following:
a. Print all the nodes reachable from a given starting node in a digraph using the BFS method.
b. Check whether a given graph is connected or not using the DFS method.

```
#include<stdio.h>


int n,graph[50][50];
int visited[50];


void DFS(int root){


   printf("%d ",root);
   visited[root]=1;
   for(int i=0;i<n;i++){
      if(graph[root][i]==1 && visited[i]!=1)
      DFS(i);
   }
}


void BFS(int root){
   int queue[n],rear=0,front=0;
   queue[rear++]=root;
   visited[queue[front]]=1;
   while(front<rear){
      printf("%d ",queue[front++]);


      for(int i=0;i<n;i++){
```

```
        if(visited[i]!=1 && graph[queue[front-1]][i]==1){
            queue[rear++]=i;
            visited[i]=1;
        }
      }
   }
}

void main(){
   printf("Enter number of nodes\n");
   scanf("%d",&n);
   printf("Enter adj matrix\n");
   for(int i=0;i<n;i++)
   for(int j=0;j<n;j++)
   scanf("%d",&graph[i][j]);
   for(int i=0;i<n;i++)
   visited[i]=0;
   printf("DFS: ");
   DFS(0);
   int i=0;
   while(visited[i]!=0 && i<n) i++;
   if(i==n){
      printf("\nConnected graph\n");
   }else{
      printf("\nDisconnected graph\n");
   }
   for(int i=0;i<n;i++)
   visited[i]=0;
   printf("BFS: ");
```

```
    BFS(0);
}
```

OUTPUT:

```
PS C:\Users\aravi\OneDrive\Desktop\notes\ADA>  & 'c:\Users\aravi\.vsco
ugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-I
1onc.kb5' '--stderr=Microsoft-MIEngine-Error-klemtkzx.5fy' '--pid=Micr
ravi\OneDrive\Desktop\my codes\codeblocks\MinGW\bin\gdb.exe' '--interp
Enter number of nodes
6
Enter adj matrix
0 1 1 0 0 0
0 0 0 1 1 0
0 0 0 0 0 1
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
DFS: 0 1 3 4 2 5
Connected graph
BFS: 0 1 2 3 4 5
```

```
PS C:\Users\aravi\OneDrive\Desktop\notes\ADA>  & 'c:\Users\aravi\.vscode\extensio
ugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-urtf0yjb.
txxk.k0v' '--stderr=Microsoft-MIEngine-Error-swqqqcsk.q5i' '--pid=Microsoft-MIEng
ravi\OneDrive\Desktop\my codes\codeblocks\MinGW\bin\gdb.exe' '--interpreter=mi'
Enter number of nodes
5
Enter adj matrix
0 1 0 0 1
1 0 1 1 1
0 1 0 1 0
0 1 1 0 1
1 1 0 1 0
DFS: 0 1 2 3 4
Connected graph
BFS: 0 1 4 2 3
```

Program-2

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

```
#include<stdio.h>

int visited[50],graph[50][50],n,stack[50],top=-1;

void DFS(int root){
   visited[root]=1;

   for(int i=0;i<n;i++){
      if(graph[root][i]==1 && visited[i]!=1){
         DFS(i);
      }
   }

   stack[++top]=root;
}

void main(){
   printf("Enter number of nodes\n");
   scanf("%d",&n);
   printf("Enter graph\n");
   for(int i=0;i<n;i++)
   for(int j=0;j<n;j++)
   scanf("%d",&graph[i][j]);
   for(int i=0;i<n;i++)
   if(visited[i]!=1)
```

```
    DFS(i);
    printf("Topological Ordering: ");
    while(top>=0)
    printf("%d ",stack[top--]);
}
```

OUTPUT:

```
rive\Desktop\my codes\codeblocks\MinGW\bin
\gdb.exe' '--interpreter=mi'
Enter number of nodes
6
Enter graph
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 1 0 0
0 1 0 0 0 0
1 1 0 0 0 0
1 0 1 0 0 0
Topological Ordering: 5 4 2 3 1 0
```

```
MinGW\bin\gdb.exe' '--interpreter=mi'
Enter number of nodes
5
Enter graph
0 1 1 0 0
0 0 0 1 1
0 0 0 1 0
0 0 0 0 1
0 0 0 0 0
Topological Ordering: 0 2 1 3 4
```

Program-3

Implement Johnson Trotter algorithm to generate permutations.

```
#include<stdio.h>

struct element{
   int val;
   int dir;
};
int n;
struct element e[50];

int largest(){
   int index=-1,temp=0;

   for(int i=0;i<n;i++){
      if(e[i].dir==-1 && i>0 && e[i].val>e[i-1].val && e[i].val>temp){
         temp=e[i].val;
         index=i;
      }

      if(e[i].dir==1 && i<n-1 && e[i].val>e[i+1].val && e[i].val>temp){
         temp=e[i].val;
         index=i;
      }
   }
```

```
    return index;
}

void main(){
    printf("Enter number of elements\n");
    scanf("%d",&n);
    printf("Enter elements\n");
    for(int i=0;i<n;i++){
        scanf("%d",&e[i].val);
        e[i].dir=-1;
    }
    printf("Possible permutations:\n");
    while(1){
        for(int i=0;i<n;i++)
        printf("%d",e[i].val);
        printf("\n");

        int index=largest();
        if(index==-1)
        break;

        int large=e[index].val;

        if(e[index].dir==-1){
            struct element temp=e[index];
            e[index]=e[index-1];
            e[index-1]=temp;
        }else{
            struct element temp=e[index];
```

```
            e[index]=e[index+1];
            e[index+1]=temp;
        }

        for(int i=0;i<n;i++){
            if(e[i].val>large){
                e[i].dir=-e[i].dir;
            }
        }
    }
}
```

OUTPUT:

```
ravi\OneDrive\Desktop\my codes\codeblocks\MinGW\bin\gdb.exe' '--interpreter=mi'
Enter number of elements
4
Enter elements
3 5 7 8
Possible permutations:
3578
3587
3857
8357
8375
3875
3785
3758
7358
7385
7835
8735
8753
7853
7583
7538
5738
5783
5873
8573
8537
5837
5387
5378
```

Program-4

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>


int a[50],c[50],n;


void merge(int low,int mid,int high){
   int i=low;
   int j=mid+1;
   int k=0;
   while(i<mid+1 && j<high+1){
      if(a[i]<a[j]){
         c[k++]=a[i];
         i++;
      }else{
         c[k++]=a[j];
         j++;
      }
   }


   while(i<mid+1){
      c[k++]=a[i];
      i++;
   }


   while(j<high+1){
```

```
        c[k++]=a[j];
        j++;
    }

    for(int q=0;q<high-low+1;q++){
        a[low+q]=c[q];
    }
}

void mergesort(int low,int high){
    if(low<high){
    int mid=(low+high)/2;
    mergesort(low,mid);
    mergesort(mid+1,high);
    merge(low,mid,high);
    }
}
void main(){
    printf("Enter number of elements\n");
    scanf("%d",&n);
    printf("Enter elements\n");
    for(int i=0;i<n;i++)
    scanf("%d",&a[i]);
    mergesort(0,n-1);
    printf("Sorted array: ");
    for(int i=0;i<n;i++)
    printf("%d ",a[i]);
}
```

OUTPUT:

```
ravi\OneDrive\Desktop\my codes\codeblocks\MinGW\bin\gdb.exe' '--interpreter=mi'
Enter number of elements
7
Enter elements
12 34 78 65 2 10 22
Sorted array: 2 10 12 22 34 65 78
```

```
Enter number of elements
10
Enter elements
0 9 4 7 2 4 1 6 10 5
Sorted array: 0 1 2 4 4 5 6 7 9 10
```

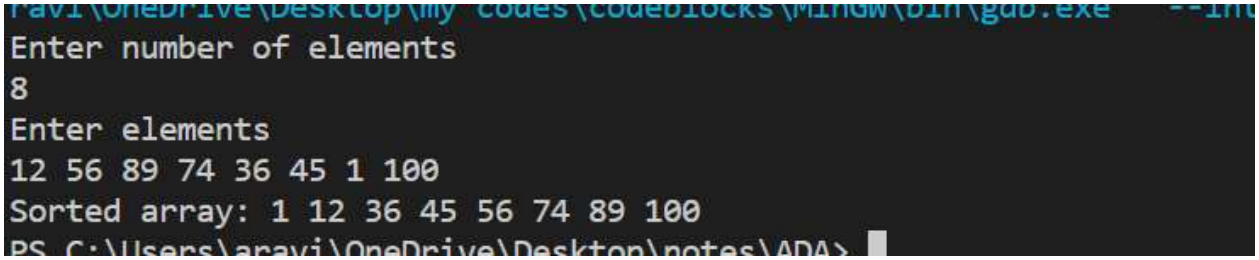
Program-5

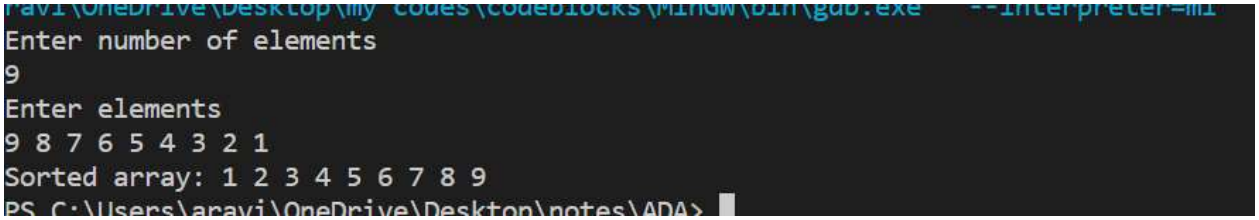
Sort a given set of N integer elements using Quick Sort technique and compute its time taken.

```
#include<stdio.h>


int a[50],n;


int sort(int low,int high){
   int pivot=a[low];
   int i=0,j=high;


   while(1){
      while(a[i]<=pivot) i++;
      while(a[j]>pivot) j--;
      if(i<j){
         int temp=a[i];
         a[i]=a[j];
         a[j]=temp;
   }else{
      int temp=a[low];
         a[low]=a[j];
         a[j]=temp;
      break;
   }
   }
   return j;
}
```

```
void quicksort(int low,int high){
    if(low<high){
    int j=sort(low,high);
    quicksort(low,j-1);
    quicksort(j+1,high);
    }
}

void main(){
    printf("Enter number of elements\n");
    scanf("%d",&n);
    printf("Enter elements\n");
    for(int i=0;i<n;i++)
    scanf("%d",&a[i]);
    quicksort(0,n-1);
    printf("Sorted array: ");
    for(int i=0;i<n;i++)
    printf("%d ",a[i]);
}
```

OUTPUT:





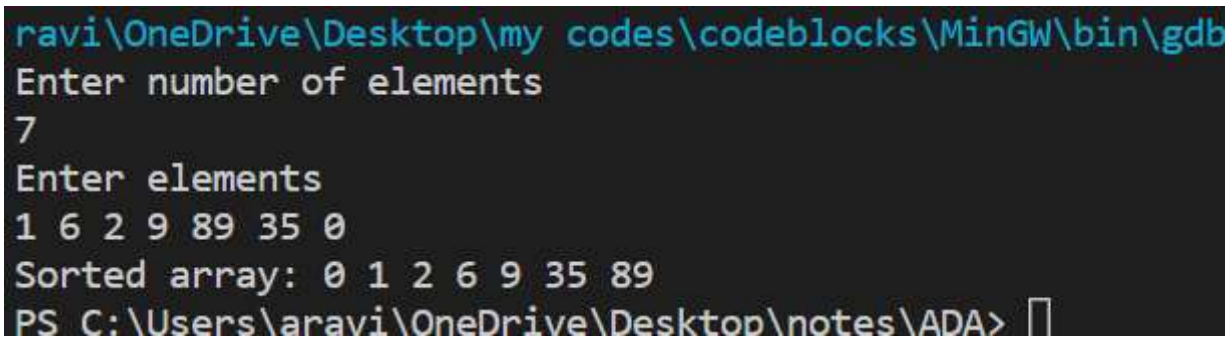
Program-6

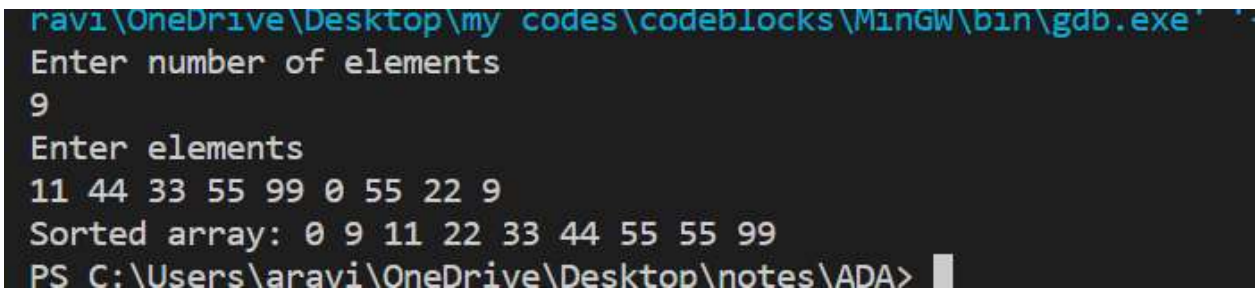
Sort a given set of N integer elements using Heap Sort technique and compute its time taken.

```
#include<stdio.h>

int a[50],n;

void heapify(int root,int end){
   int largest=root;
   int left=2*root+1;
   int right=2*root+2;

   if(left<end && a[left]>a[largest])
   largest=left;

   if(right<end && a[right]>a[largest] )
   largest=right;

   if(largest!=root){
      int temp=a[largest];
      a[largest]=a[root];
      a[root]=temp;
      heapify(largest,end);
   }
}

void heapsort(){
```

```
    for(int i=n/2-1;i>=0;i--)
    heapify(i,n-1);

    for(int i=n-1;i>=0;i--){
       int temp=a[0];
       a[0]=a[i];
       a[i]=temp;
    heapify(0,i);
    }
}

void main(){
    printf("Enter number of elements\n");
    scanf("%d",&n);
    printf("Enter elements\n");
    for(int i=0;i<n;i++)
    scanf("%d",&a[i]);
    heapsort();
    printf("Sorted array: ");
    for(int i=0;i<n;i++)
    printf("%d ",a[i]);
}
```

OUTPUT:


ravi\OneDrive\Desktop\my codes\codeblocks\MinGW\bin\gdb
Enter number of elements
7
Enter elements
1 6 2 9 89 35 0
Sorted array: 0 1 2 6 9 35 89
PS C:\Users\aravi\OneDrive\Desktop\notes\ADA>


ravi\OneDrive\Desktop\my codes\codeblocks\MinGW\bin\gdb.exe'
Enter number of elements
9
Enter elements
11 44 33 55 99 0 55 22 9
Sorted array: 0 9 11 22 33 44 55 55 99
PS C:\Users\aravi\OneDrive\Desktop\notes\ADA>

Program-7

Implement 0/1 Knapsack problem using dynamic programming.

```
#include<stdio.h>

struct item{
   int w;
   int v;
};

int n,cap;

struct item a[20];

int knapsack(){
   int arr[cap+1][n+1];
   for(int i=0;i<=cap;i++) arr[i][0]=0;
   for(int i=0;i<=n;i++) arr[0][i]=0;

   for(int i=1;i<=cap;i++){
      for(int j=1;j<=n;j++){
         if(i>=a[j-1].w){
            arr[i][j]=a[j-1].v+arr[i-a[j-1].w][j-1]>arr[i][j-1]?a[j-1].v+arr[i-a[j-1].w][j-1]:arr[i][j-1];
         }else{
            arr[i][j]=arr[i][j-1];
         }
      }
   }
```

```
    for(int i=0;i<=cap;i++){
        for(int j=0;j<=n;j++)
        printf("%d ",arr[i][j]);
        printf("\n");
    }
    return arr[cap][n];
}

void main(){
    printf("Enter number of items\n");
    scanf("%d",&n);
    printf("Enter weight and value\n");
    for(int i=0;i<n;i++)
    scanf("%d %d",&a[i].w,&a[i].v);
    printf("Enter capacity\n");
    scanf("%d",&cap);
    printf("Maximum profits are %d",knapsack());
}
```

OUTPUT:

```
ravi\OneDrive\Desktop\my codes\codeblocks\MinGW\bin\gdb.exe' '--interpreter=mi'
Enter number of items
4
Enter weight and value
3 2
4 3
6 1
5 4
Enter capacity
8
Maximum profits are 5
PS C:\Users\aravi\OneDrive\Desktop\notes\ADA>
```

```
ravi\OneDrive\Desktop\my codes\codeblocks\MinGW\bin
Enter number of items
3
Enter weight and value
10 60
20 100
30 120
Enter capacity
50
Maximum profits are 220
```

Program-8

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

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

int arr[50][50],c[50][50],n;

void floyds(){
   for(int k=0;k<n;k++){
      for(int i=0;i<n;i++){
         for(int j=0;j<n;j++){
            if(arr[i][k]!=INT_MAX && arr[k][j]!=INT_MAX)
            arr[i][j]= arr[i][j]<arr[i][k]+arr[k][j]? arr[i][j] : arr[i][k]+arr[k][j];
         }
      }
   }

   for(int i=0;i<n;i++){
      for(int j=0;j<n;j++){
         if(arr[i][j]==INT_MAX)
         printf("INF ");
         else
         printf("%d ",arr[i][j]);
      }
      printf("\n");
   }
```

```
}

void main(){
   printf("Enter number of nodes\n");
   scanf("%d",&n);
   printf("Enter weight matrix(-1 for disjoint nodes)\n");
   for(int i=0;i<n;i++){
      for(int j=0;j<n;j++){
      scanf("%d",&c[i][j]);
      if(c[i][j]==-1){
         c[i][j]=INT_MAX;
      }
      arr[i][j]=c[i][j];
      }
   }

   printf("Shortest paths:\n");
   floyds();
}
```

OUTPUT:

```
ravi\OneDrive\Desktop\my codes\codeblocks\MinGW\bin\gdb.exe' '--inte
Enter number of nodes
4
Enter weight matrix(-1 for disjoint nodes)
0 5 -1 10
-1 0 3 -1
-1 -1 0 1
-1 -1 -1 0
Shortest paths:
0 5 8 9
INF 0 3 4
INF INF 0 1
INF INF INF 0
```

```
Enter number of nodes
4
Enter weight matrix(-1 for disjoint nodes)
0 3 -1 5
2 0 -1 4
-1 1 0 -1
-1 -1 2 0
Shortest paths:
0 3 7 5
2 0 6 4
3 1 0 5
5 3 2 0
```

Program-9

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

Prims:

```
#include<stdio.h>
int cost[10][10],n;


void prims(){
   int visited[n],key[n],parent[n];
   for(int i=0;i<n;i++)
   visited[i]=0,key[i]=9999;


   key[0]=0;
   parent[0]=-1;
   for(int i=0;i<n-1;i++){
      int v;
      for(int j=0,min=9999;j<n;j++){
         if(!visited[j] && key[j]<min)
         min=key[j],v=j;
      }
      visited[v]=1;


      for(int j=0;j<n;j++){
         if(!visited[j] && cost[v][j]<key[j])
         key[j]=cost[v][j],parent[j]=v;
      }
```

```
    }

    printf("MST\n");
    int c=0;
    for(int j=1;j<n;j++){
        printf("%d -> %d\n",j,parent[j]);
        c+=cost[j][parent[j]];
    }
    printf("Cost %d",c);
}

void main(){
printf("ENter the number of nodes\n");
scanf("%d",&n);
printf("Enter weight matrix(-1 for disjoint nodes)\n");
for(int i=0;i<n;i++)
for(int j=0;j<n;j++){
    scanf("%d",&cost[i][j]);
    if(cost[i][j]==-1)
    cost[i][j]=9999;
}
prims();
}
```

OUTPUT:

```
ravi\OneDrive\Desktop\my codes\codeblocks\MinGW\bi
ENter the number of nodes
5
Enter weight matrix(-1 for disjoint nodes)
0 2 -1 6 -1
2 0 3 8 5
-1 3 0 -1 7
6 8 -1 0 9
-1 5 7 9 0
MST
1 -> 0
2 -> 1
3 -> 0
4 -> 1
Cost 16
```

```
ravi\OneDrive\Desktop\my codes\codeblocks\MinGW\bin\
ENter the number of nodes
5
Enter weight matrix(-1 for disjoint nodes)
0 -1 3 -1 -1
-1 0 10 4 -1
3 10 0 2 6
-1 4 2 0 1
-1 -1 6 1 0
MST
1 -> 3
2 -> 0
3 -> 2
4 -> 3
Cost 10
```

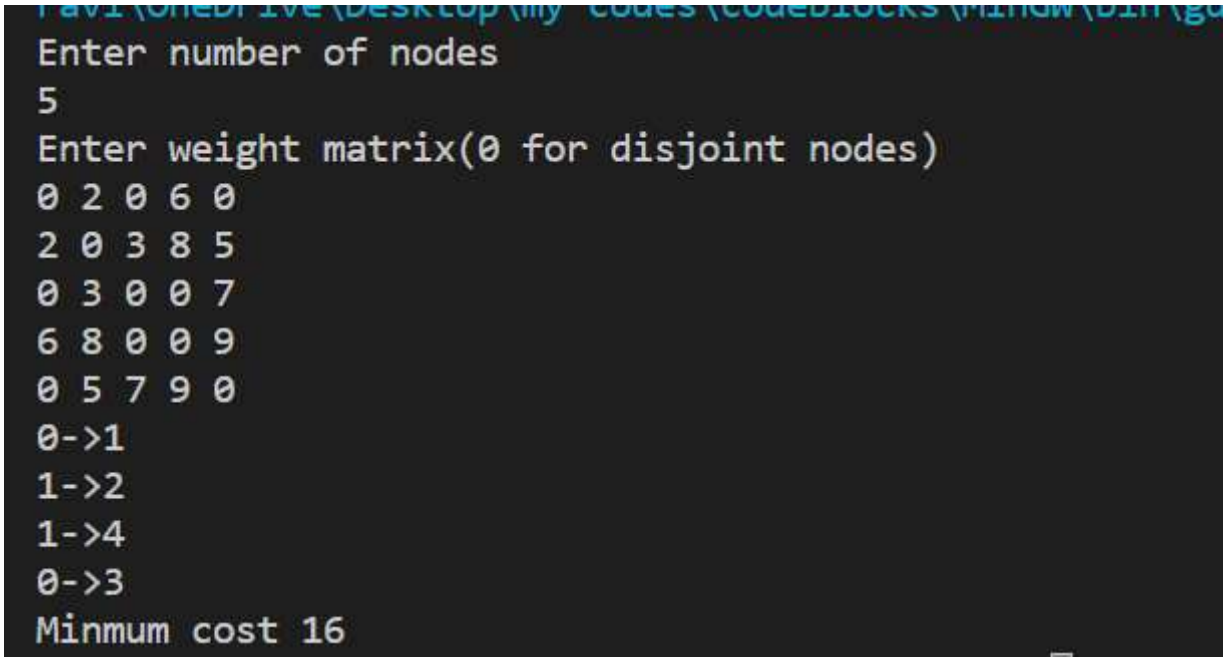
Kruskal:

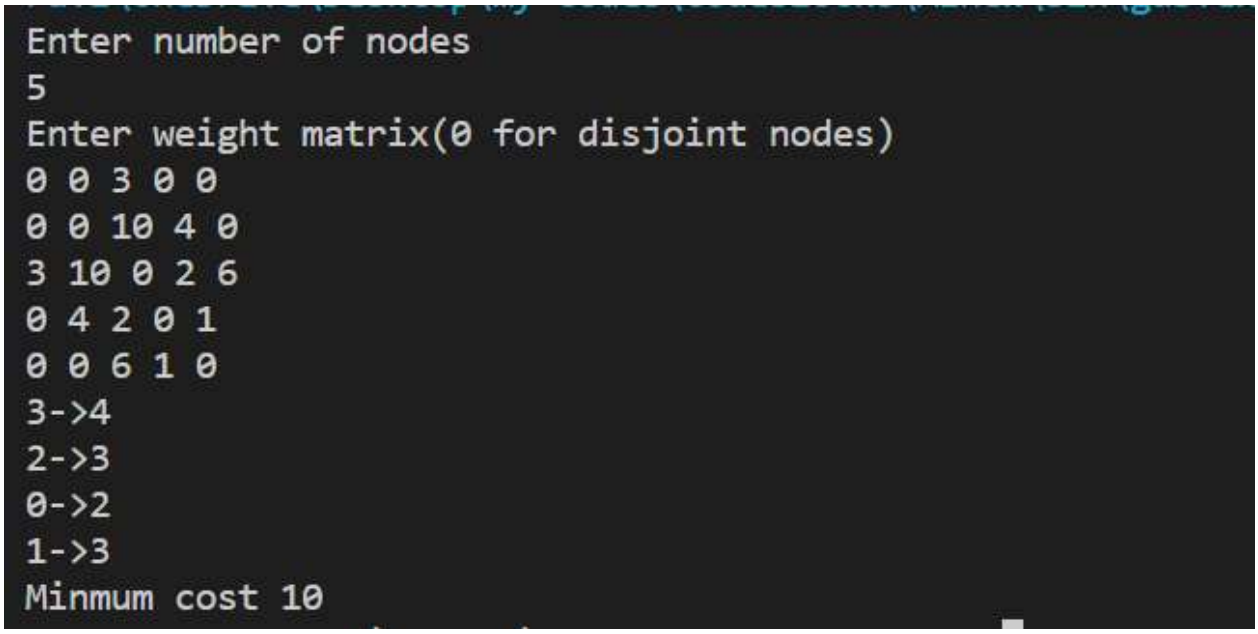
```
#include<stdio.h>

int n,mat[20][20],visited[20],mincost=0;

struct minedge{
int i;
int j;
int w;
};
struct minedge e[10];

void kruskal(){
   int k=0;
   for(int i=0;i<n;i++){
      for(int j=i+1;j<n;j++){
         if(mat[i][j]!=999){
            e[k].i=i;
            e[k].j=j;
            e[k++].w=mat[i][j];
         }
      }
   }

   for(int i=0;i<k;i++){
      for(int j=0;j<k-1;j++){
         if(e[j].w>e[j+1].w){
            struct minedge temp=e[j];
            e[j]=e[j+1];
```

```
            e[j+1]=temp;
          }
        }
    }

    for(int i=0;i<n;i++) visited[i]=0;

    for(int i=0;i<k;i++){
        if(visited[e[i].i]==0 || visited[e[i].j]==0){
            visited[e[i].i]=1;
            visited[e[i].j]=1;
            mincost+=e[i].w;
            printf("%d->%d\n",e[i].i,e[i].j);
        }
    }
    printf("Minmum cost %d",mincost);
}

void main(){
    printf("Enter number of nodes\n");
    scanf("%d",&n);
    printf("Enter weight matrix(0 for disjoint nodes)\n");
    for(int i=0;i<n;i++)
    for(int j=0;j<n;j++){
        scanf("%d",&mat[i][j]);
        if(mat[i][j]==0)
        mat[i][j]=999;
    }
    kruskal();
```

}

OUTPUT:


Enter number of nodes
5
Enter weight matrix(0 for disjoint nodes)
0 2 0 6 0
2 0 3 8 5
0 3 0 0 7
6 8 0 0 9
0 5 7 9 0
0->1
1->2
1->4
0->3
Minmum cost 16


Enter number of nodes
5
Enter weight matrix(0 for disjoint nodes)
0 0 3 0 0
0 0 10 4 0
3 10 0 2 6
0 4 2 0 1
0 0 6 1 0
3->4
2->3
0->2
1->3
Minmum cost 10

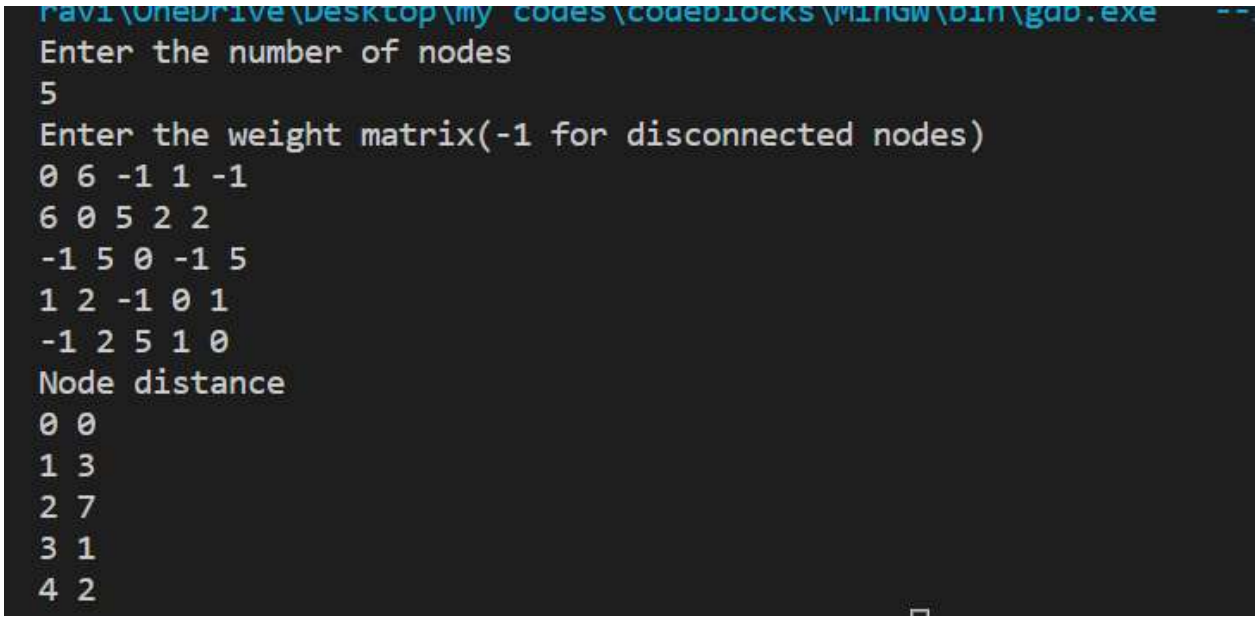
Program-10

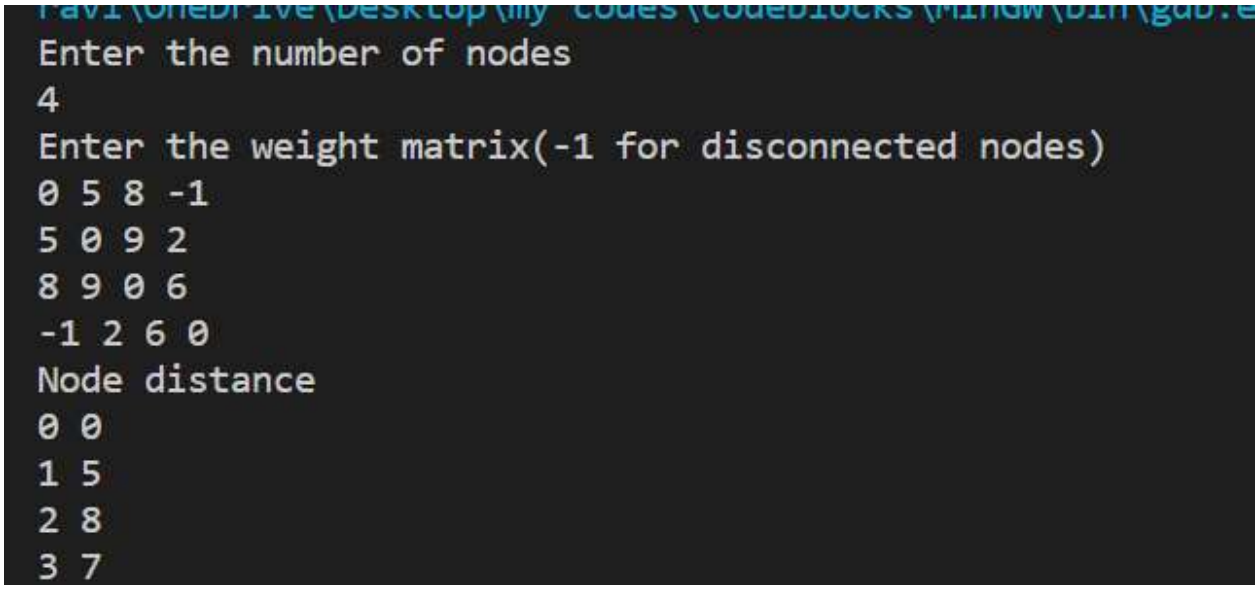
From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.

```
#include<stdio.h>


int mat[20][20],n,dist[20];


void dijkstra(int s){
   int visited[n];
   for(int i=0;i<n;i++) visited[i]=0;
   int c=0;
   for(int i=0;i<n;i++) dist[i]=mat[s][i];
   visited[s]=1;
   c++;


   while(c<n){
      int min=999,index=-1;
      for(int i=0;i<n;i++)
      if(visited[i]==0 && dist[i]<min){
         min=dist[i];
         index=i;
      }


      visited[index]=1;
      c++;
      for(int i=0;i<n;i++){
         if(dist[i]>dist[index]+mat[index][i])
```

```
            dist[i]=dist[index]+mat[index][i];
        }
    }
}

void main(){
    printf("Enter the number of nodes\n");
    scanf("%d",&n);
    printf("Enter the weight matrix(-1 for disconnected nodes)\n");
    for(int i=0;i<n;i++)
    for(int j=0;j<n;j++){
        scanf("%d",&mat[i][j]);
        if(mat[i][j]==-1)
        mat[i][j]=999;
    }
    dijkstra(0);
    printf("Node distance\n");
    for(int i=0;i<n;i++)
    printf("%d %d\n",i,dist[i]);
}
```

OUTPUT:


Enter the number of nodes
5
Enter the weight matrix(-1 for disconnected nodes)
0 6 -1 1 -1
6 0 5 2 2
-1 5 0 -1 5
1 2 -1 0 1
-1 2 5 1 0
Node distance
0 0
1 3
2 7
3 1
4 2


Enter the number of nodes
4
Enter the weight matrix(-1 for disconnected nodes)
0 5 8 -1
5 0 9 2
8 9 0 6
-1 2 6 0
Node distance
0 0
1 5
2 8
3 7

Program-11

Implement "N-Queens Problem" using Backtracking.

```
#include<stdio.h>

int board[10][10],n;

int safe(int row,int col){
   for(int i=0;i<n;i++){
      if(board[i][col])
      return 0;
   }

   for(int i=row,j=col;i>=0 && j>=0;i--,j--)
   if(board[i][j])
   return 0;

   for(int i=row,j=col;i>=0 && j>=0;i--,j++)
   if(board[i][j])
   return 0;

   return 1;
}

void solve(int row){
   if(row==n){
      for(int i=0;i<n;i++){
```

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

    for(int i=0;i<n;i++){
        if(safe(row,i)){
            board[row][i]=1;
            solve(row+1);
            board[row][i]=0;
        }
    }
}

void main(){
    printf("Enter number of queens\n");
    scanf("%d",&n);
    solve(0);
}
```

```
Enter number of queens
4
0 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
```