Problem Statement 1.Program to implement Linear Search algorithm using c language.

```
#include<stdio.h>
#define MAX 100
int lSearch(int *data,int key,int size){
   int i;
   for ( i = 0; i < size; i++)
   {
      if (data[i] == key)
      {
         return i;
      }
   }
   return -1;
}
void printArray(int *data,int size){
   int i;
   for ( i = 0; i < size; i++)
   {
      printf("%d ",data[i]);
   }
}
int main()
{
   int n,i;
   int data[MAX];
   printf("Enter the no. of elements in array:- ");
   scanf("%d",&n);
   printf("Enter elements of array:- ");
   for(i = 0; i < n;i++)
         scanf("%d",&data[i]);
   printf(“Array is:- “);
   printArray(data,n);
   i = lSearch(data,79,n);
   printf("Element find at:- %d",i+1);
   return 0;
}
```

Input:-

```
Enter the no. of elements in array:- 10
Enter elements of array:- 12 52 63 45 85 79 65 32 15 2
```

Output:-

```
Element find at:- 6
Array is:-  12 52 63 45 85 79 65 32 15 2
--------------------------------
Process exited after 36.57 seconds with return value 0
Press any key to continue . . .
```

Problem Statement 2.Program to implement Binary Search algorithms using c language.

```
#include<stdio.h>
#define MAX 100
int bSearch(int *data,int key,int low,int high){
    int mid = (low + high)/2;
    if (data[mid] == key)
    {
        return mid;
    }else if(key < data[mid]){
        bSearch(data,key,low,mid-1);
    }else{
        bSearch(data,key,mid+1,high);
    }
}
void printArray(int *data,int size){
    int i;
    for ( i = 0; i < size; i++)
    {
        printf("%d ",data[i]);
    }
}
int main()
{
    int n,i;
    int data[MAX];
    printf("Enter the no. of elements in array:- ");
    scanf("%d",&n);
    printf("Enter elements of array:- ");
    for(i = 0; i < n;i++)
            scanf("%d",&data[i]);
    printf(“Array is:- “);
    printArray(data,n);
    i = bSearch(data,63,0,n-1);
    printf("Element find at:- %d",i+1);
    return 0;
}
```

Input:-

```
Enter the no. of elements in array:- 10
Enter elements of array:- 12 52 63 45 85 79 65 32 15 2
```

Output:-

```
Element find at:- 3
Array is:-  12 52 63 45 85 79 65 32 15 2
--------------------------------
Process exited after 36.57 seconds with return value 0
Press any key to continue . . .
```

Problem Statement 3.Program to implement Insertion Sort algorithm using c language.

```
#include<stdio.h>
#define MAX 100
void insertionSort(int *data,int size){
    int v;
    int i,j;
    for ( i = 1; i < size; i++)
    {
      v = data[i];
      j = i-1;
      while (data[j] > v && j >= 0)
      {
          data[j+1] = data[j];
          j--;
      }
       data[j+1] = v;
    }
}
void printArray(int *data,int size){
    int i;
    for ( i = 0; i < size; i++)
    {
       printf("%d ",data[i]);
    }
}
int main()
{
    int n,i;
    int data[MAX];
    printf("Enter the no. of elements in array:- ");
    scanf("%d",&n);
    printf("Enter elements of array:- ");
    for(i = 0; i < n;i++)
          scanf("%d",&data[i]);
    insertionSort(data,n);
    printf("\nSorted array:- ");
    printArray(data,n);
    return 0;
}
```

Input:-

```
Enter the no. of elements in array:- 10
Enter elements of array:- 98 78 54 65 2 54 36 1 7 23
```

Output:-

```
Sorted array:- 1 2 7 23 36 54 54 65 78 98
--------------------------------
Process exited after 43.89 seconds with return value 3221225725
Press any key to continue . . .
```

Problem Statement 4.Program to implement Quick Sort algorithm using c language.

```
#include<stdio.h>
#define MAX 100
void swap(int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
int partion(int *data,int low,int high){
    int pivot = data[high];
    int i = low-1;
    int j;
    for ( j = low; j < high; j++)
    {
        if (data[j] <= pivot)
        {
            i++;
            swap(&data[i],&data[j]);
        }
    }
    swap(&data[i+1],&data[j]);
    return i+1;
}
void quickSort(int *data,int low,int high){
    if (low < high)
    {
        int p = partion(data,low,high);
        quickSort(data,low,p-1);
        quickSort(data,p+1,high);
    }
}
void printArray(int *data,int size){
    int i;
    for ( i = 0; i < size; i++)
    {
        printf("%d ",data[i]);
    }
}
int main()
{
    int n,i;
    int data[MAX];
    printf("Enter the no. of elements in array:- ");
    scanf("%d",&n);
    printf("Enter elements of array:- ");
    for(i = 0; i < n;i++)
            scanf("%d",&data[i]);
    quickSort(data,0,n-1);
    printf("\nSorted array:- ");
    printArray(data,n);
    return 0;
}
```

Input:-

```
Enter the no. of elements in array:- 5
Enter elements of array:- 36 54 56 2 1
```

Output:-

```
Sorted array:- 1 2 36 54 56
--------------------------------
Process exited after 14.16 seconds with return value 0
Press any key to continue . . .
```

Problem Statement 5.Program to implement Merge Sort problem using c language.

```
#include<stdio.h>
#define MAX 100
void swap(int *a, int *b){
   int temp = *a;
   *a = *b;
   *b = temp;
}
void merge(int *data,int low,int mid,int high){
   int arr[10];
   int i,j,k,p;
   i = low; k = low;
   j = mid+1;
   while ((i <= mid) && (j <= high))
   {
      if (data[i] <= data[j])
      {
         arr[k++] = data[i++];
      }else{
         arr[k++] = data[j++];
      }
   }
   for ( p = j; p <= high; p++)
   {
      arr[k++] = data[p];
   }
   for ( p = i; p <= mid; p++)
   {
      arr[k++] = data[p];
   }
   for ( p = low; p <= high; p++)
   {
      data[p] = arr[p];
   }
}

void mergeSort(int *data,int low,int high){
   int mid;
   if (low < high)
   {
        mid = (low + high)/2;
        mergeSort(data,low,mid);
        mergeSort(data,mid+1,high);
        merge(data,low,mid,high);
    }
}
void printArray(int *data,int size){
    int i;
    for ( i = 0; i < size; i++)
    {
        printf("%d ",data[i]);
    }
}
int main()
{
    int n,i;
    int data[MAX];
    printf("Enter the no. of elements in array:- ");
    scanf("%d",&n);
    printf("Enter elements of array:- ");
    for(i = 0; i < n;i++)
            scanf("%d",&data[i]);
    quickSort(data,0,n-1);
    printf("\nSorted array:- ");
    printArray(data,n);
    return 0;
}
```

Input:-

```
Enter the no. of elements in array:- 10
Enter elements of array:- 65 85 45 14 21 75 2 5 3 1
```

Output:-

```
Sorted array:- 1 2 3 5 14 21 45 65 75 85
--------------------------------
Process exited after 23.91 seconds with return value 0
Press any key to continue . . .
```

Problem Statement 6.Program to implement Heap Sort algorithm using c language.

```
#include<stdio.h>
#define MAX 100
void swap(int *a, int *b){
   int temp = *a;
   *a = *b;
   *b = temp;
}
void adjust(int *data,int low,int high){
   int ele,i;
   ele = data[low];
   i = 2*low;
   while (i <= high)
   {
      if (i < high && data[i] < data[i+1])
         i++;
      if (ele >= data[i])
         break;
      data[i/2] = data[i];
      i = 2 * i;
   }
   data[i/2] = ele;
}
void heapify(int *data,int size){
   int i;
   for (i = size/2; i >= 0; i--)
   {
      adjust(data,i,size);
   }
}
void heapSort(int *data,int size){
   int i;
   heapify(data,size);
   for ( i = size; i >= 1; i--)
   {
      swap(&data[0],&data[i]);
      adjust(data,0,i-1);
   }
}
void printArray(int *data,int size){
   int i;
   for ( i = 0; i < size; i++)
   {
      printf("%d ",data[i]);
   }
}
int main()
{
   int n,i;
   int data[MAX];
   printf("Enter the no. of elements in array:- ");
   scanf("%d",&n);
   printf("Enter elements of array:- ");
   for(i = 0; i < n;i++)
         scanf("%d",&data[i]);
   quickSort(data,0,n-1);
   printf("\nSorted array:- ");
   printArray(data,n);
   return 0;
}
```

Input:-

```
Enter the no. of elements in array:- 5
Enter elements of array:- 65 89 2 56 1
```

Output:-

```
Sorted array:- 1 2 56 65 89
--------------------------------
Process exited after 10.82 seconds with return value 0
Press any key to continue . . .
```

Problem Statement 7.Program to implement O/1 Knapsack problem using c language.

```
#include<stdio.h>

int main(){
    int wgt,n,i,j,w,a,b,max=0;
    int m[10][2],km[10][10];
    printf("Enter the no. of items:- ");
    scanf("%d",&n);
    printf("\nEnter the input matrix");
    for( i = 0; i < n; i++)
    {
      scanf("%d%d",&m[i][0],&m[i][1]);
    }
    printf("\nEnter capacity:-");
    scanf("%d",&wgt);
    printf("\nInput Matrix");
    printf("\n\tItem\tWeight\tValue\n");
    for ( i = 0; i < n; i++)
    {
        printf("\n\t%d\t%d\t%d",i+1,m[i][0],m[i][1]);
    }
    for (i = 0; i <= n; i++)
    {
        for (j = 0; j <= wgt; j++)
        {
            if (i == 0 && j >= 0)
            {
                km[i][j] = 0;
            }else if (i >= 0 && j == 0)
            {
                km[i][j] = 0;
            }else if((j-m[i-1][0]) < 0){
                km[i][j] = km[i-1][j];
            }else if ((j-m[i-1][0]) >= 0)
            {
                a = km[i-1][j];
                b = m[i-1][1] + km[i-1][j-m[i-1][0]];
                if (a>b)
                {
                    km[i][j] = a;
                }else
                {
                    km[i][j] = b;
                }
            }
        }
    }
    printf("\nKnapsack MAtrix\n");
    for ( i = 0; i <= n; i++)
    {
        for ( j = 0; j <= wgt; j++)
        {
            printf("\t%d",km[i][j]);
        }
        printf("\n");
    }
    i = n;j = wgt; w =0;max= 0;
    printf("\nOptimal set :-\n");
    printf("\n\tInput\tWeight\tValue");
    while ((wgt - w) > 0)
    {
        if (km[i][j] != km[i-1][j])
        {
            printf("\n\t%d\t%d\t%d",i,m[i-1][0],m[i-1][1]);
            max = max + m[i-1][1];
            w = w+m[i-1][0];
            j = wgt - w;
        }
        i--;
        if (i < 1)
        {
            break;
        }
    }
    printf("\n\tMaximum Value In the knap. %d",max);
    return 0;
}
```

Input:-

```
Enter the no. of items:- 3
Enter the input matrix:- 2 200
3 150
7 75
Enter capacity:-6
```

Output:-

```
PS C:\Users\Harsh\Desktop\DAA> cd "c:\Users\Harsh\Desktop\DAA\" ; if ($?) { gcc
KanpSackDP.c -o KanpSackDP } ; if ($?) { .\KanpSackDP }

Input Matrix
        Item    Weight  Value
        1           2      200
        2           3      150
        3          7      75
Knapsack MAtrix
        0      0      0      0      0      0      0
        0      0      200    200    200    200    200
        0      0      200    200    200    350    350
        0      0      200    200    200    350    350

Optimal set :-

        Input   Weight  Value
        2          3      150
        1          2      200
        Maximum Value In the knap. 350
```

Problem Statement 8.Floyd’s Algorithm to solve all pair shortest path problems using c language.

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 10
void floyd(int);
int min(int,int);
int w[MAX][MAX],d[MAX][MAX][MAX];
int main()
{
   int i,j,v,k;
   printf("Enter the no. of nodes:- ");
   scanf("%d",&v);
   printf("Enter weight matrix:- ");
   for (i = 1; i <= v; i++)
   {
      for ( j = 1; j <= v; j++)
      {
         printf("Enter weight of %d to %d:- ",i,j);
         scanf("%d",&w[i][j]);
      }
   }
   floyd(v);
   return 0;
}
int min(int a,int b){
   return a>=b?b:a;
}
void floyd(int v){
   int k,i,j;
   k = 0;
   for (i = 1; i <= v; i++)
   {
      for (j = 1; j <= v; j++)
      {
         d[k][i][j] = w[i][j];
      }
   }
   for (k = 1; k <= v; k++)
   {
        for (i = 1; i <= v; i++)
        {
            for ( j = 1; j <= v; j++)
            {
                d[k][i][j] = min(d[k-1][i][j],(d[k-1][i][k] + d[k-1][k][j]));
            }
        }
    }
    for (k = 0; k <= v; k++)
    {
        printf("%dth matrix:- \n",k);
        for (i = 1; i <= v; i++)
        {
            for ( j = 1; j <= v; j++)
            {
                printf("\t%d ",d[k][i][j]);
            }
            printf("\n");
        }
    }
}
```

Input:-Weight matrix of a weighted graph

Output:-

```
PS C:\Users\Harsh\Desktop\DAA> cd "c:\Users\Harsh\Desktop\DAA\" ; if ($?) { gcc Floyds.c -o Floyds } ; if ($?) { .\Floyds }
Enter the no. of nodes:- 4
Enter weight matrix:-
Enter weight of 1 to 1:- 0
Enter weight of 1 to 2:- 3
Enter weight of 1 to 3:- 99999
Enter weight of 1 to 4:- 5
Enter weight of 2 to 1:- 2
Enter weight of 2 to 2:- 0
Enter weight of 2 to 3:- 99999
Enter weight of 2 to 4:- 4
Enter weight of 3 to 1:- 99999
Enter weight of 3 to 2:- 1
Enter weight of 3 to 3:- 0
Enter weight of 3 to 4:- 99999
Enter weight of 4 to 1:- 99999
Enter weight of 4 to 2:- 99999
Enter weight of 4 to 3:- 2
Enter weight of 4 to 4:- 0
0th matrix:-
        0       3       99999   5
        2       0       99999   4
        99999   1       0       99999
        99999   99999   2       0
1th matrix:-
        0       3       99999   5
        2       0       99999   4
        99999   1       0       99999
        99999   99999   2       0
2th matrix:-
        0       3       99999   5
        2       0       99999   4
        3       1       0       5
        99999   99999   2       0
3th matrix:-
        0       3       99999   5
        2       0       99999   4
        3       1       0       5
        5       3       2       0
4th matrix:-
        0       3       7       5
        2       0       6       4
        3       1       0       5
        5       3       2       0
```

Problem Statement 9.Program to implement Fractional Knapsack problem using c language.

```
#include<stdio.h>
#include<stdlib.h>
void swap(float *a,float *b){
   float t = *a;
   *a = *b;
   *b = t;
}
int main(int argc, char const *argv[])
{
   int n,i,j;
   float m[10][3],x[10];
   float wgt,max,v;
   printf("Enter no of items:- ");
   scanf("%d",&n);
   printf("Enter Weight and Values of items:- ");
   for ( i = 0; i < n; i++)
   {
      printf("Enter weight and value of %dth item:- ",i+1);
      scanf("%f%f",&m[i][0],&m[i][1]);
   }
   printf("Enter capacity of the Knapsack:- ");
   scanf("%f",&wgt);

   //calculating value per unit
   for (i = 0; i < n; i++)
   {
      m[i][2] = m[i][1]/m[i][0];
   }
   //arranging in desending order on value per unit
   for (i = 0; i < n - 1; i++)
   {
      for ( j = 0; j < n - i - 1; j++)
      {
         if (m[j][2] < m[j+1][2])
         {
            swap(&m[j][0],&m[j+1][0]);
            swap(&m[j][1],&m[j+1][1]);
            swap(&m[j][2],&m[j+1][2]);
            }
        }
    }
    //actual logic
    for ( i = 0; i < n; i++)
    {
        x[i] = 0;
    }
    v = wgt;
    for ( i = 0; i < n; i++)
    {
        if (m[i][0] > v) break;
        x[i] = m[i][0];
        v -= m[i][0];
    }
    if(i < n)
        x[i] = v / m[i][0];
    //solution vector
    for ( i = 0; i < n; i++)
    {
        printf("%.2f, ",x[i]);
    }
    max = 0;
    for ( i = 0; i < n; i++)
    {
        max += (m[i][2] * x[i]);
    }
    printf("\nMaximum profit earned:- %.2f",max);
    return 0;
}
```

Input:- N item of known values and weight and also a knapsack of capacity M.

Output:- Maximum profit Earned by selecting item, we are allowed to take a part of item also.

```
Enter no of items:- 4
Enter Weight and Values of items:- Enter weight and value of 1th item:- 20 200
Enter weight and value of 2th item:- 25 100
Enter weight and value of 3th item:- 10 50
Enter weight and value of 4th item:- 20 120
Enter capacity of the Knapsack:- 50
20.00, 20.00, 10.00, 0.00,
Maximum profit earned:- 370.00
--------------------------------
Process exited after 98.38 seconds with return value 0
Press any key to continue . . .
```

Problem Statement 10.Program to implement Chain Matrix Multiplication problem using c language.

```
#include<stdio.h>
#include<stdlib.h>
void cmm(int [][10],int [][10],int [],int);
void ops(int [][10],int,int);
void display(int [][10],int);
int main(int argc, char const *argv[])
{
   int m[10][10] = {0},s[10][10] = {0};
   int p[10] = {0};
   int i,n;
   printf("Enter the no. of matrices:- ");
   scanf("%d",&n);
   printf("Enter the dimension of matrices:- ");
   for (i = 0; i <= n; i++)
   {
      scanf("%d",&p[i]);
   }
   cmm(m,s,p,n);
   printf("\nOptimal solution:- \n");
   display(m,n);
   printf("\nOptimal parenthesization:- \n");
   ops(s,1,n);
   return 0;
}

void cmm(int m[10][10],int s[10][10],int p[10],int n){
   int i,j,k,q,l;
   for (i = 1; i <= n; i++)
   {
      m[i][i] = 0;
   }
   for ( l = 2; l <= n; l++)
   {
      for ( i = 1; i <= n-l+1; i++)
      {
         j = i + l - 1;
         m[i][j] = 99999;
         for ( k = i; k <= j-1; k++)
```

```
            {
                q = m[i][k] + m[k+1][j] + (p[i-1]*p[k]*p[j]);
                if (q < m[i][j])
                {
                    m[i][j] = q;
                    s[i][j] = k;
                }
            }
        }
    }
    printf("\nThe maximum no of scalar multiplication are:- %d",m[1][n]);
}

void ops(int s[10][10],int i,int j){
    if (i == j)
    {
        printf(" A%d ",i);
        return ;
    }else{
        printf("(");
        ops(s,i,s[i][j]);
        ops(s,s[i][j] +1 ,j);
        printf(")");
    }
    return ;
}
void display(int s[10][10],int n){
    int i,j;
    for ( i = 1; i <= n; i++)
    {
        for ( j = 0; j <= n; j++)
        {
            printf("%d ",s[i][j]);
        }
        printf("\n");
    }

}
```

Input:- Dimension of chain of given matrices

Output:- Minimum no of required scalar multiplication to find product of chain and also optimal parenthesization scheme.

```
Enter the no. of matrices:- 3
Enter the dimension of matrices:- 3 4 5 3

The maximum no of scalar multiplication are:- 96
Optimal solution:-
0 0 60 96
0 0 0 60
0 0 0 0

Optimal parenthesization:-
( A1 ( A2  A3 ))
--------------------------------
Process exited after 36.38 seconds with return value 0
Press any key to continue . . .
```

Problem Statement 11.Program to implement NQueen Problem using c language.

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

int place(int k,int i,int *x){
    int j;
    for (j = 1; j <= k-1; j++)
    {
        if (x[j] == i || abs(j-k) == abs(x[j]-i))
        {
            return 0;
        }
    }
    return 1;
}
void NQueen(int k,int n,int *x){
    int i,j;
    for (i = 1; i <= n; i++)
    {
        if (place(k,i,x))
        {
            x[k] = i;
            if (k == n)
            {
                printf("Solutions Matrix is:-\n");
                for (i = 1; i <= n; i++)
                {
                    for ( j = 0; j <= n; j++)
                    {
                        if (x[i] == j)
                        {
                            printf(" Q ");
                        }else{
                            printf(" 0 ");
                        }
                    }
                    printf("\n");
                }
            }else{
                NQueen(k+1,n,x);
            }

        }
    }
}

int main(int argc, char const *argv[])
{
    int x[10],N;
    printf("Enter the number of row:- ");
    scanf("%d",&N);
    NQueen(1,N,x);
    return 0;
}
```

Input:-Dimension of chess board

Output:- Column number so no two queens are in the attacking position.

```
Enter the number of row:- 4
Solutions Matrix is:-
 0  0  Q  0  0
 0  0  0  0  Q
 0  Q  0  0  0
 0  0  0  Q  0
Solutions Matrix is:-
 0  0  0  Q  0
 0  Q  0  0  0
 0  0  0  0  Q
 0  0  Q  0  0

--------------------------------
Process exited after 3.724 seconds with return value 0
Press any key to continue . . .
```

Problem Statement 12.Program to implement Dijkstra’s Algorithm using c Language.

```
#include <stdio.h>
#define INFINITY 9999
#define MAX 10

void Dijkstra(int Graph[MAX][MAX], int n, int start);

void Dijkstra(int Graph[MAX][MAX], int n, int start) {
  int cost[MAX][MAX], distance[MAX], pred[MAX];
  int visited[MAX], count, mindistance, nextnode, i, j;

  // Creating cost matrix
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (Graph[i][j] == 0)
        cost[i][j] = INFINITY;
      else
        cost[i][j] = Graph[i][j];

  for (i = 0; i < n; i++) {
    distance[i] = cost[start][i];
    pred[i] = start;
    visited[i] = 0;
  }

  distance[start] = 0;
  visited[start] = 1;
  count = 1;

  while (count < n - 1) {
    mindistance = INFINITY;

    for (i = 0; i < n; i++)
      if (distance[i] < mindistance && !visited[i]) {
        mindistance = distance[i];
        nextnode = i;
      }

    visited[nextnode] = 1;
    for (i = 0; i < n; i++)
```

```
   if (!visited[i])
    if (mindistance + cost[nextnode][i] < distance[i]) {
     distance[i] = mindistance + cost[nextnode][i];
     pred[i] = nextnode;
    }
  count++;
 }

 // Printing the distance
 for (i = 0; i < n; i++)
  if (i != start) {
   printf("\nDistance from source to %d: %d", i, distance[i]);
  }
}
int main() {
 int Graph[MAX][MAX], i, j, n, u;
 printf("Enter the number of nodes in graph:- ");
 scanf("%d",&n);

 for ( i = 0; i < n; i++)
 {
  for ( j = 0; j < n; j++)
  {
    printf("\nEnter the cost from %d to %d:- ",i+1,j+1);
    scanf("%d",&Graph[i][j]);
  }
 }
 u = 0;
 Dijkstra(Graph, n, u);
 return 0;
}
```

Output:-

```
PS C:\Users\Harsh\Desktop\DAA> cd "c:\Users\Harsh\Desktop\DAA\" ; if ($?) { gcc
Dijkastras.c -o Dijkastras } ; if ($?) { .\Dijkastras }
Enter the number of nodes in graph:- 7

Enter the cost from 1 to 1:- 0

Enter the cost from 1 to 2:- 0

Enter the cost from 1 to 3:- 1

Enter the cost from 1 to 4:- 2

Enter the cost from 1 to 5:- 0

Enter the cost from 1 to 6:- 0

Enter the cost from 1 to 7:- 0

Enter the cost from 2 to 1:- 0

Enter the cost from 2 to 2:- 0

Enter the cost from 2 to 3:- 2

Enter the cost from 2 to 4:- 0

Enter the cost from 2 to 5:- 0

Enter the cost from 2 to 6:- 3

Enter the cost from 2 to 7:- 0

Enter the cost from 3 to 1:- 1

Enter the cost from 3 to 2:- 2

Enter the cost from 3 to 3:- 0
Enter the cost from 3 to 4:- 1

Enter the cost from 3 to 5:- 3

Enter the cost from 3 to 6:- 0

Enter the cost from 3 to 7:- 0

Enter the cost from 4 to 1:- 2

Enter the cost from 4 to 2:- 0

Enter the cost from 4 to 3:- 1

Enter the cost from 4 to 4:- 0

Enter the cost from 4 to 5:- 0

Enter the cost from 4 to 6:- 0

Enter the cost from 4 to 7:- 1

Enter the cost from 5 to 1:- 0

Enter the cost from 5 to 2:- 0

Enter the cost from 5 to 3:- 3

Enter the cost from 5 to 4:- 0

Enter the cost from 5 to 5:- 0

Enter the cost from 5 to 6:- 2

Enter the cost from 5 to 7:- 0

Enter the cost from 6 to 1:- 0

Enter the cost from 6 to 2:- 3
Enter the cost from 6 to 3:- 0

Enter the cost from 6 to 4:- 0

Enter the cost from 6 to 5:- 2

Enter the cost from 6 to 6:- 0

Enter the cost from 6 to 7:- 1

Enter the cost from 7 to 1:- 0

Enter the cost from 7 to 2:- 0

Enter the cost from 7 to 3:- 0

Enter the cost from 7 to 4:- 1

Enter the cost from 7 to 5:- 0

Enter the cost from 7 to 6:- 1

Enter the cost from 7 to 7:- 0

Distance from source to 1: 3
Distance from source to 2: 1
Distance from source to 3: 2
Distance from source to 4: 4
Distance from source to 5: 4
Distance from source to 6: 3
PS C:\Users\Harsh\Desktop\DAA>
```