ADS Assignment 6

Name	: Varad Ghote
Div	: SY CS-D
Roll no	: 62
PRN	: 12211508

KRUSKAL'S ALGORITHM

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

int n;
int X, Y , wei;
int Tweight = 0;
    int E;


struct node
{
    int data;
    int weight;
    struct node *next;
};

/*


                        ADJACENCY LIST


*/

struct List
{
    int num;
    struct node *array[100];
};
```

```c
void Edge(struct List *A, int a, int b)
{
    int w;
    printf("Enter weight : ");
    scanf("%d", &w);
    struct node *temp1 = (struct node *)malloc(sizeof(struct node));
    temp1->data = b;
    temp1->next = NULL;
    temp1->weight = w;

    if (A->array[a] == NULL)
    {
        A->array[a] = temp1;
    }
    else
    {
        struct node *P = A->array[a];
        while (P->next != NULL)
        {
            P = P->next;
        }
        P->next = temp1;
    }

    struct node *temp2 = (struct node *)malloc(sizeof(struct node));
    temp2->data = a;
    temp2->next = NULL;
    temp2->weight = w;

    if (A->array[b] == NULL)
    {
        A->array[b] = temp2;
    }
    else
    {
        struct node *P = A->array[b];
        while (P->next != NULL)
        {
            P = P->next;
        }
        P->next = temp2;
    }
}

void printGraph(struct List *A)
{
    for (int i = 0; i < A->num; ++i)
```

```
    {
        struct node *currentNode = A->array[i];
        printf("Adjacency list of vertex %d: ", i + 1);
        while (currentNode)
        {
            printf("%d -> ", currentNode->data + 1);
            currentNode = currentNode->next;
        }
        printf("NULL\n");
    }
}




void min(struct List *A)
{
    int min = 99;
    int i;
    for (i = 0; i < n; i++)
    {

         struct node *P = A->array[i];
         if (P != NULL)
         {
        do
        {
             if (P->weight < min)
            {
                min = P->weight;
                wei =  P->weight;
                X = i;
                Y = P->data;
            }
            P = P->next;
        } while (P != NULL);
        }
    }
}

int par(int Q, int *parent)
{

    while (parent[Q] != -1)
    {
        Q = parent[Q];
    }
    return Q;
}
```

```
void merge(struct List *A)
{
    struct node *prev1 = NULL;
    struct node *P = A->array[X];

    while (P != NULL && P->data != Y)
    {
        prev1 = P;
        P = P->next;
    }

    if (P == NULL)
    {
        return;
    }

    if (prev1 == NULL)
    {
        A->array[X] = P->next;
    }
    else
    {
        prev1->next = P->next;
    }

    struct node *prev2 = NULL;
    struct node *S = A->array[Y];

    while (S != NULL && S->data != X)
    {
        prev2 = S;
        S = S->next;
    }

    if (S == NULL)
    {
        return;
    }

    if (prev2 == NULL)
    {
        A->array[Y] = S->next;
    }
    else
    {
        prev2->next = S->next;
    }
}
```

```
void Krushkals(struct List *A)
{
    int x, y;
    int parent[10];
    int q=0;
    for (int i = 0; i < 10; i++)
    {
        parent[i] = -1;
    }
    printf("\n\nEdges involved are :\n");
while (q != E)    {
        min(A);

        x = par(X, parent);
        y = par(Y, parent);

        if (x != y)
        {
            parent[y] = x;
            Tweight += wei;
            printf("edge %d  to egge %d      (weight=%d)\n", X + 1, Y + 1,
wei);
        }
        q++;
        merge(A);
    }
    printf("\n\n");
}

/*


                    ADJACENCY MATRIX


*/

struct M_List
{
    int num;
    int matrix[100][100];
};

void M_Edge(struct M_List *B, int a, int b)
{
    int w;
    printf("Enter Weight :");
```

```
    scanf("%d", &w);
    B->matrix[a][b] = w;
    B->matrix[b][a] = w;
}

void M_printGraph(struct M_List *B)
{
    printf("Adjacency Matrix:\n");
    for (int i = 0; i < B->num; ++i)
    {
        for (int j = 0; j < B->num; ++j)
        {
            printf("%d\t", B->matrix[i][j]);
        }
        printf("\n");
    }
    printf("\n\n\n");
}


int M_par(int Q, int *parent)
{

    while (parent[Q] != -1)
    {
        Q = parent[Q];
    }
    return Q;
}

void M_min(struct M_List *B)
{
    int min = 99;
    int i;
    for (i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (B->matrix[i][j] < min)
            {
                min = B->matrix[i][j];
                X = i;
                Y = j;
            }
        }
    }
}

void M_Krushkals(struct M_List *B)
```

```
{
    int x, y;
    int q=0;
    int parent[10];
    for (int i = 0; i < 10; i++)
    {
        parent[i] = -1;
    }

        printf("\n\nEdges involved are :\n");

    while (q != E)
    {
        M_min(B);

        x = M_par(X, parent);
        y = M_par(Y, parent);


        if (x != y)
        {
            parent[y] = x;
            Tweight += B->matrix[X][Y];
            printf("edge %d  to egge %d      (weight=%d)\n", X + 1, Y + 1, B-
>matrix[X][Y]);
        }
        q++;
        B->matrix[X][Y] = 99;
        B->matrix[Y][X] = 99;
    }
    printf("\n\n");
}

int main()
{
    int i;
    int choice;
    struct List *A;
    struct M_List *B;

    printf("Enter the number of vertices : ");
    scanf("%d", &n);
    printf("\nvertices are :");
    for (i = 0; i < n; i++)
    {
        printf("   %d\t", i + 1);
    }
    printf("\n\nEnter the number of edges: ");
```

```
    scanf("%d", &E);

    while (1)
    {
        printf("\n\n1)Adjacency List \n");
        printf("2)Adjacency Matrix\n");
        printf("3)EXIT\n\n");
        printf("enter your choice:");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:

            A = (struct List *)malloc(sizeof(struct List));
            A->num = n;
            for (i = 0; i < n; ++i)
            {
                A->array[i] = NULL;
            }

            for (i = 0; i < E; ++i)
            {
                int a, b;
                do
                {
                    printf("\n\nEnter edge %d (start and end point): \n", i +
1);
                    scanf("%d", &a);
                    scanf("%d", &b);
                    if (a > n || b > n || a == 0 || b == 0 || a == b)
                    {
                        printf("Invalid vertex. Please enter valid
vertices.\n");
                    }
                } while (a > n || b > n || a == 0 || b == 0 || a == b);
                Edge(A, a - 1, b - 1);
            }
            printGraph(A);

            Krushkals(A);
            break;

        case 2:

            B = (struct M_List *)malloc(sizeof(struct M_List));
            B->num = n;
            for (int i = 0; i < n; ++i)
```

```
            {
                for (int j = 0; j < n; ++j)
                {
                    B->matrix[i][j] = 99;
                }
            }

            for (int i = 0; i < E; ++i)
            {
                int a, b;
                do
                {
                    printf("\n\nEnter edge %d (start and end point): \n", i +
1);
                    scanf("%d", &a);
                    scanf("%d", &b);
                    if (a > n || b > n || a == 0 || b == 0 || a == b)
                    {
                        printf("Invalid vertex. Please enter valid
vertices.\n");
                    }
                } while (a > n || b > n || a == 0 || b == 0 || a == b);
                M_Edge(B, a - 1, b - 1);
            }
            M_printGraph(B);

            M_Krushkals(B);

            break;

        case 3:
            printf("\n\n\t!!!!THANK YOU!!!!!\n\n\n");
            return 1;
            break;
        }
    }
    return 0;
}
```

Output

ADJACENCY LIST

```
PS D:\SY 1st sem> cd 'd:\SY 1st sem\ADS\output'
PS D:\SY 1st sem\ADS\output> & .\'6.Krushkals.exe'
Enter the number of vertices : 5

vertices are :   1        2       3       4       5

Enter the number of edges: 7


1)Adjacency List
2)Adjacency Matrix
3)EXIT

enter your choice:1


Enter edge 1 (start and end point):
1
2
Enter weight : 4


Enter edge 2 (start and end point):
1
3
Enter weight : 3


Enter edge 3 (start and end point):
2
3
Enter weight : 2


Enter edge 4 (start and end point):
2
4
Enter weight : 3


Enter edge 5 (start and end point):
2
5
Enter weight : 3


Enter edge 6 (start and end point):
3
5
Enter weight : 1


Enter edge 7 (start and end point):
4
5
Enter weight : 4
Adjacency list of vertex 1: 2 -> 3 -> NULL
Adjacency list of vertex 2: 1 -> 3 -> 4 -> 5 -> NULL
Adjacency list of vertex 3: 1 -> 2 -> 5 -> NULL
Adjacency list of vertex 4: 2 -> 5 -> NULL
Adjacency list of vertex 5: 2 -> 3 -> 4 -> NULL


Edges involved are :
edge 3  to egge 5       (weight=1)
edge 2  to egge 3       (weight=2)
edge 1  to egge 3       (weight=3)
edge 2  to egge 4       (weight=3)



1)Adjacency List
2)Adjacency Matrix
3)EXIT
```

ADJACENCY MATRIX

```
1)Adjacency List
2)Adjacency Matrix
3)EXIT

enter your choice:2


Enter edge 1 (start and end point):
1
2
Enter Weight :4


Enter edge 2 (start and end point):
1
3
Enter Weight :3


Enter edge 3 (start and end point):
2
3
Enter Weight :2


Enter edge 4 (start and end point):
2
4
Enter Weight :3


Enter edge 5 (start and end point):
2
5
Enter Weight :3


Enter edge 6 (start and end point):
3
5
Enter Weight :1


Enter edge 7 (start and end point):
4
5
```

```
Enter edge 7 (start and end point):
4
5
Enter Weight :4
Adjacency Matrix:
99      4       3       99      99
4       99      2       3       3
3       2       99      99      1
99      3       99      99      4
99      3       1       4       99




Edges involved are :
edge 3  to egge 5       (weight=1)
edge 2  to egge 3       (weight=2)
edge 1  to egge 3       (weight=3)
edge 2  to egge 4       (weight=3)



1)Adjacency List
2)Adjacency Matrix
3)EXIT

enter your choice:3


        !!!!THANK YOU!!!!!


PS D:\SY 1st sem\ADS\output>
```