# DAA Lab-10

02/04/2025 –  230962326 – Davasam Karthikeya - AMILB

**Solved question (Prim's algorithm):**

```
#include <stdio.h>

int a[4][4] = {
    {0, 3, 0, 2},
    {3, 0, 1, 0},
    {0, 1, 0, 5},
    {2, 0, 5, 0}
};
int t[4][4], root[4], parent[4], n = 4, i, j, value, e = 4, k = 0;
int ivalue, jvalue, cost = 0, mincost = 0, TV[4], count = 0, present = 0;

int check_reach(int v) {
    for(int p = 1; p <= count; p++)
        if (TV[p] == v)
            return 1;
    return 0;
}

void prims() {
    while (e && k < n - 1) {
        for(i = 0; i < n; i++)
            for(j = 0; j < n; j++)
                if(a[i][j] != 0) {
                    int x = check_reach(i), y = check_reach(j);
                    if(x && !y) {
                        present = 1;
                        if((a[i][j] < cost) || (cost == 0)) {
                            cost = a[i][j];
                            ivalue = i;
                            jvalue = j;
                        }
                    }
                }
        if (present == 0)
            break;
        a[ivalue][jvalue] = 0;
        a[jvalue][ivalue] = 0;
        e--;
        TV[++count] = jvalue;
        t[ivalue][jvalue] = cost;
        k++;
        present = cost = 0;
    }
}

void display() {
    if(k == n - 1) {
        printf("\nMin. cost:");
        for(i = 0; i < n; i++)
            for(j = 0; j < n; j++) {
                if (t[i][j] != 0)
                    printf("\n(%d, %d): %d", i, j, t[i][j]);
                mincost += t[i][j];
            }
        printf("\nTotal: %d\n", mincost);
    }
    else
        printf("\nGraph is not connected\n");
}

void main() {
    TV[++count] = 1;
    prims();
    display();
}
```

Output

```
Min. cost:
(0, 3): 2
(1, 0): 3
(1, 2): 1
Total: 6
```

## Question 1 (Krushkal’s algorithm):

```
#include <stdio.h>

int cost[4][4] = {
    {0, 3, 0, 2},
    {3, 0, 1, 0},
    {0, 1, 0, 5},
    {2, 0, 5, 0}
};
int  p[4] = {};

int findF(int i) {
    while(p[i])
        i = p[i];
    return i;
}

int uniF(int i,int j) {
    if(i != j) {
        p[j] = i;
        return 1;
    }
    return 0;
}

void main() {
    int a, b, u, v, n = 4, min = 0;

    for(int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            if (cost[i][j] == 0) {
                cost[i][j] = 999999;
            }

    printf("Min. cost:\n");
    for(int ne = 1; ne < n; ne++) {
        int minVal = 999999;
        for(int i = 0; i < n; i++)
            for(int j = 0; j < n; j++)
                if(cost[i][j] < minVal) {
                    minVal = cost[i][j];
                    a = u = i;
                    b = v = j;
                }
        u = findF(u);
        v = findF(v);

        if(uniF(u, v)) {
            printf("(%d, %d)\n", a, b);
            min += minVal;
        }

        cost[a][b] = cost[b][a] = 999999;
    }
    printf("Total: %d\n", min);
}
```

Output

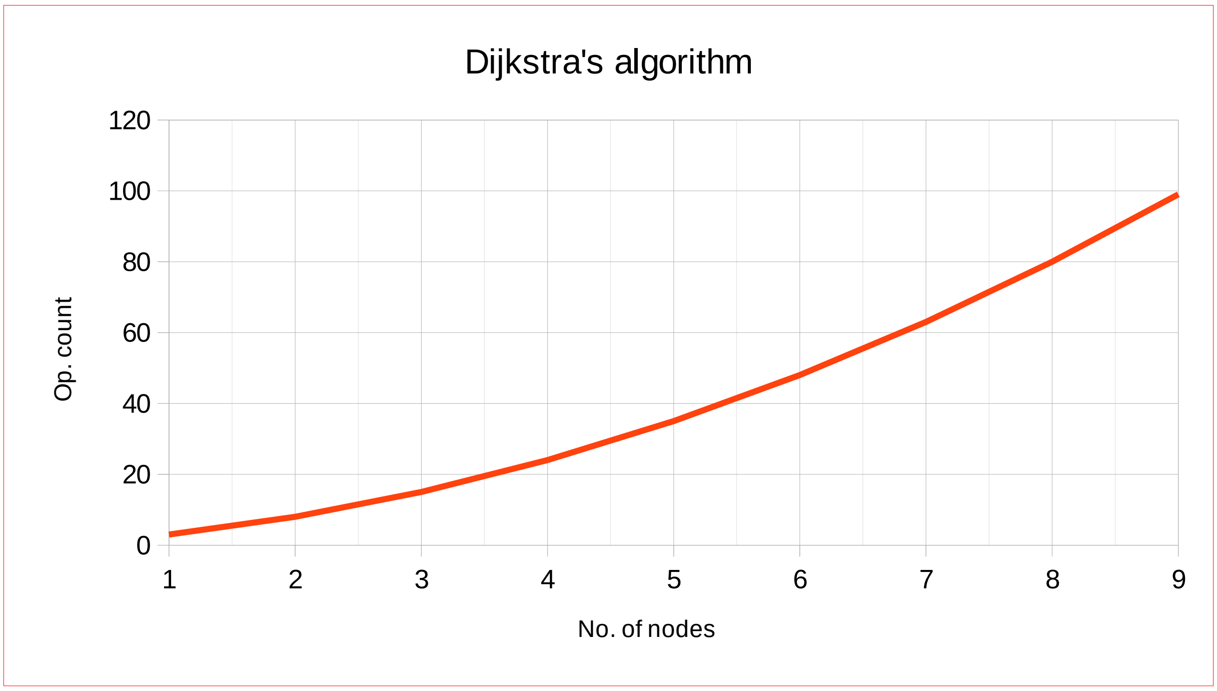
```
Min. cost:
(1, 2)
(0, 3)
(0, 1)
Min. cost: 6
```

## Question 2 (Dijkstra's algorithm):

```
#include <stdio.h>

#define MAX 4

int opCount = 0;

void dijkstra(int G[MAX][MAX], int startnode) {
    int cost[MAX][MAX], dist[MAX], pred[MAX];
    int visited[MAX], minDist, nextNod;

    for (int i = 0; i < MAX; i++)
        for (int j = 0; j < MAX; j++)
            if (G[i][j] == 0)
                cost[i][j] = 999999;
            else
                cost[i][j] = G[i][j];

    opCount = MAX * (MAX + 1);
    for (int i = 0; i < MAX; i++) {
        dist[i] = cost[startnode][i];
        pred[i] = startnode;
        visited[i] = 0;
    }

    dist[startnode] = 0;
    visited[startnode] = 1;

    for(int count = 1; count < MAX - 1; count++) {
        minDist = 999999;
        for (int i = 0; i < MAX; i++)
            if (dist[i] < minDist && !visited[i]) {
                minDist = dist[i];
                nextNod = i;
                opCount++;
            }

        visited[nextNod] = 1;
        for(int i = 0; i < MAX; i++)
            if (!visited[i])
                if (minDist + cost[nextNod][i] < dist[i]) {
                    dist[i] = minDist + cost[nextNod][i];
                    pred[i] = nextNod;
                    opCount += 2;
                }
        }

    for(int i = 0; i < MAX; i++) {
        if (i != startnode) {
            if (dist[i] == 9999) {
                printf("No path for: %d, %d\n", i, startnode);
            } else {
                printf("Cost from %d: %d", i, dist[i]);
                printf("\nPath: %d", i);
                int j = i;
                do {
                    j = pred[j];
                    printf(" -> %d", j);
                } while (j != startnode);
            }
            printf("\n");
        }
    }
}

void main() {
    int G[MAX][MAX] = {
        {0, 3, 0, 2},
        {3, 0, 1, 0},
        {0, 1, 0, 5},
        {2, 0, 5, 0}
    };

    int u;
    printf("Enter start: ");
    scanf("%d", &u);

    dijkstra(G, u);
    printf("\nOp. count: %d\n", opCount);
}
```

Output

```
Enter start: 2
Cost from 0: 4
Path: 0 -> 1 -> 2
Cost from 1: 1
Path: 1 -> 2
Cost from 3: 5
Path: 3 -> 2
Op. count: 24
```


Dijkstra's algorithm
Op. count
0
20
40
60
80
100
120
1
2
3
4
5
6
7
8
9
No. of nodes