Practical programs(21-25)

21. Write a c program to graph traversal using breadth first search

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

#define MAX_VERTICES 100

struct Node {
    int data;
    struct Node* next;
};

struct Graph {
    struct Node* adjList[MAX_VERTICES];
    int visited[MAX_VERTICES];
};

void initializeGraph(struct Graph* graph, int numVertices) {
    for (int i = 0; i < numVertices; i++) {
        graph->adjList[i] = NULL;
        graph->visited[i] = 0;
    }
}

void addEdge(struct Graph* graph, int src, int dest) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = dest;
    newNode->next = graph->adjList[src];
    graph->adjList[src] = newNode;
}

void bfs(struct Graph* graph, int startVertex, int numVertices) {
    int queue[MAX_VERTICES];
    int front = 0, rear = 0;

    graph->visited[startVertex] = 1;
    queue[rear++] = startVertex;

    while (front < rear) {
        int currentVertex = queue[front++];
```

```
        printf("%d ", currentVertex);

        struct Node* temp = graph->adjList[currentVertex];
        while (temp != NULL) {
            int adjVertex = temp->data;
            if (graph->visited[adjVertex] == 0) {
                graph->visited[adjVertex] = 1;
                queue[rear++] = adjVertex;
            }
            temp = temp->next;
        }
    }
}

int main() {
    struct Graph graph;
    int numVertices, numEdges;

    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);

    printf("Enter the number of edges: ");
    scanf("%d", &numEdges);

    initializeGraph(&graph, numVertices);

    printf("Enter edges (source destination):\n");
    for (int i = 0; i < numEdges; i++) {
        int src, dest;
        scanf("%d %d", &src, &dest);
        addEdge(&graph, src, dest);
        addEdge(&graph, dest, src);  // For undirected graph
    }

    int startVertex;
    printf("Enter the starting vertex for BFS: ");
    scanf("%d", &startVertex);

    printf("BFS traversal starting from vertex %d: ", startVertex);
    bfs(&graph, startVertex, numVertices);
    printf("\n");

    return 0;
}
```

Output:

```
Enter the number of vertices: 6
Enter the number of edges: 7
Enter edges (source destination):
0 9
0 9
5 4
9 3
4 6
5 3
4 9
Enter the starting vertex for BFS: 4
BFS traversal starting from vertex 4: 4 9
--------------------------------
Process exited after 98.04 seconds with return value 3221225477
Press any key to continue . . .
```

22 .Write a c program to graph traversal using deapth first search

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

#define MAX_VERTICES 100

struct Node {
    int data;
    struct Node* next;
};

struct Graph {
    struct Node* adjList[MAX_VERTICES];
    int visited[MAX_VERTICES];
};

void initializeGraph(struct Graph* graph, int numVertices) {
    for (int i = 0; i < numVertices; i++) {
        graph->adjList[i] = NULL;
```

```
        graph->visited[i] = 0;
    }
}

void addEdge(struct Graph* graph, int src, int dest) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = dest;
    newNode->next = graph->adjList[src];
    graph->adjList[src] = newNode;
}

void dfs(struct Graph* graph, int vertex) {
    graph->visited[vertex] = 1;
    printf("%d ", vertex);

    struct Node* temp = graph->adjList[vertex];
    while (temp != NULL) {
        int adjVertex = temp->data;
        if (graph->visited[adjVertex] == 0) {
            dfs(graph, adjVertex);
        }
        temp = temp->next;
    }
}

int main() {
    struct Graph graph;
    int numVertices, numEdges;

    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);

    printf("Enter the number of edges: ");
    scanf("%d", &numEdges);

    initializeGraph(&graph, numVertices);

    printf("Enter edges (source destination):\n");
    for (int i = 0; i < numEdges; i++) {
        int src, dest;
        scanf("%d %d", &src, &dest);
        addEdge(&graph, src, dest);
        addEdge(&graph, dest, src);  // For undirected graph
    }
```

```
    int startVertex;
    printf("Enter the starting vertex for DFS: ");
    scanf("%d", &startVertex);

    printf("DFS traversal starting from vertex %d: ", startVertex);
    dfs(&graph, startVertex);
    printf("\n");

    return 0;
}
```

Output:

```
Enter the number of vertices: 5
Enter the number of edges: 6
Enter edges (source destination):
4 9
2 53
6 46
1 64
3 46
7 656
Enter the starting vertex for DFS: 4
DFS traversal starting from vertex 4: 4 9 0
--------------------------------
Process exited after 40.78 seconds with return value 3221225477
Press any key to continue . . .
```

23.implementation of shortest path algorithm using dijkstra's algorithm

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

#define MAX_VERTICES 100

struct Graph {
    int numVertices;
```

```
    int adjMatrix[MAX_VERTICES][MAX_VERTICES];
};

void initializeGraph(struct Graph* graph, int numVertices) {
    graph->numVertices = numVertices;
    for (int i = 0; i < numVertices; i++) {
        for (int j = 0; j < numVertices; j++) {
            graph->adjMatrix[i][j] = 0;  // Initialize with no edges
        }
    }
}

void addEdge(struct Graph* graph, int src, int dest, int weight) {
    graph->adjMatrix[src][dest] = weight;
    graph->adjMatrix[dest][src] = weight;  // For undirected graph
}

int minDistance(int dist[], int visited[], int numVertices) {
    int min = INT_MAX, minIndex;

    for (int v = 0; v < numVertices; v++) {
        if (!visited[v] && dist[v] <= min) {
            min = dist[v];
            minIndex = v;
        }
    }

    return minIndex;
}

void dijkstra(struct Graph* graph, int src) {
    int dist[MAX_VERTICES];
    int visited[MAX_VERTICES];

    for (int i = 0; i < graph->numVertices; i++) {
        dist[i] = INT_MAX;
        visited[i] = 0;
    }

    dist[src] = 0;

    for (int count = 0; count < graph->numVertices - 1; count++) {
        int u = minDistance(dist, visited, graph->numVertices);
        visited[u] = 1;
```

```
        for (int v = 0; v < graph->numVertices; v++) {
            if (!visited[v] && graph->adjMatrix[u][v] && dist[u] != INT_MAX &&
                dist[u] + graph->adjMatrix[u][v] < dist[v]) {
                dist[v] = dist[u] + graph->adjMatrix[u][v];
            }
        }
    }

    printf("Vertex   Distance from Source\n");
    for (int i = 0; i < graph->numVertices; i++) {
        printf("%d         %d\n", i, dist[i]);
    }
}

int main() {
    struct Graph graph;
    int numVertices, numEdges;

    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);

    printf("Enter the number of edges: ");
    scanf("%d", &numEdges);

    initializeGraph(&graph, numVertices);

    printf("Enter edges and weights (source destination weight):\n");
    for (int i = 0; i < numEdges; i++) {
        int src, dest, weight;
        scanf("%d %d %d", &src, &dest, &weight);
        addEdge(&graph, src, dest, weight);
    }

    int startVertex;
    printf("Enter the starting vertex for Dijkstra's algorithm: ");
    scanf("%d", &startVertex);

    dijkstra(&graph, startVertex);

    return 0;
}
```

Output:

```
Enter the number of vertices: 4
Enter the number of edges: 5
Enter edges and weights (source destination weight):
54 94 94
65 97 654
94 99
66 95 95
62 9 65
68 64 65
Enter the starting vertex for Dijkstra's algorithm: Vertex   Distance from Source
0         2147483647
1         2147483647
2         2147483647
3         2147483647

--------------------------------
Process exited after 31.63 seconds with return value 0
Press any key to continue . . .
```

24. Implementation of minimum spanning tree using prim's algorithm

```
#include <stdio.h>
#include <limits.h>
#define MAX_VERTICES 100
struct Graph {
    int numVertices;
    int adjMatrix[MAX_VERTICES][MAX_VERTICES];
};

void initializeGraph(struct Graph* graph, int numVertices) {
    graph->numVertices = numVertices;
    for (int i = 0; i < numVertices; i++) {
        for (int j = 0; j < numVertices; j++) {
            graph->adjMatrix[i][j] = 0;  // Initialize with no edges
        }
    }
}

void addEdge(struct Graph* graph, int src, int dest, int weight) {
    graph->adjMatrix[src][dest] = weight;
    graph->adjMatrix[dest][src] = weight;  // For undirected graph
```

```
}

int minKey(int key[], int mstSet[], int numVertices) {
    int min = INT_MAX, minIndex;

    for (int v = 0; v < numVertices; v++) {
        if (!mstSet[v] && key[v] < min) {
            min = key[v];
            minIndex = v;
        }
    }

    return minIndex;
}
void primMST(struct Graph* graph) {
    int parent[MAX_VERTICES];  // To store the constructed MST
    int key[MAX_VERTICES];     // Key values used to pick minimum weight edge
    int mstSet[MAX_VERTICES];  // To represent set of vertices included in MST

    for (int i = 0; i < graph->numVertices; i++) {
        key[i] = INT_MAX;
        mstSet[i] = 0;
    }

    key[0] = 0;       // Start from the first vertex
    parent[0] = -1;   // First vertex is the root of MST

    for (int count = 0; count < graph->numVertices - 1; count++) {
        int u = minKey(key, mstSet, graph->numVertices);
        mstSet[u] = 1;

        for (int v = 0; v < graph->numVertices; v++) {
            if (graph->adjMatrix[u][v] && !mstSet[v] &&
                graph->adjMatrix[u][v] < key[v]) {
                parent[v] = u;
                key[v] = graph->adjMatrix[u][v];
            }
        }
    }

    printf("Edge   Weight\n");
    for (int i = 1; i < graph->numVertices; i++) {
        printf("%d - %d    %d\n", parent[i], i, graph->adjMatrix[i][parent[i]]);
    }
```

```
}

int main() {
    struct Graph graph;
    int numVertices, numEdges;

    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);

    printf("Enter the number of edges: ");
    scanf("%d", &numEdges);

    initializeGraph(&graph, numVertices);

    printf("Enter edges and weights (source destination weight):\n");
    for (int i = 0; i < numEdges; i++) {
        int src, dest, weight;
        scanf("%d %d %d", &src, &dest, &weight);
        addEdge(&graph, src, dest, weight);
        addEdge(&graph, dest, src, weight);  // For undirected graph
    }
  primMST(&graph);
     return 0;
}
```

Output:

```
Enter the number of vertices: 3
Enter the number of edges: 4
Enter edges and weights (source destination weight):
5 6 1
9 5 6
6 8 9
6 64 9
Edge   Weight
0 - 1    0

--------------------------------
Process exited after 17.26 seconds with return value 3221225477
Press any key to continue . . .
```

25. Implementation of minimum spanning tree using kruskal algorithm

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

#define MAX_VERTICES 100

struct Edge {
    int src, dest, weight;
};

struct Graph {
    int numVertices, numEdges;
    struct Edge edges[MAX_VERTICES];
};

void initializeGraph(struct Graph* graph, int numVertices, int numEdges) {
    graph->numVertices = numVertices;
    graph->numEdges = numEdges;
    for (int i = 0; i < numEdges; i++) {
        graph->edges[i].src = 0;
        graph->edges[i].dest = 0;
        graph->edges[i].weight = 0;
    }
}

int find(int parent[], int vertex) {
    if (parent[vertex] == -1) {
        return vertex;
    }
    return find(parent, parent[vertex]);
}

void unionSets(int parent[], int x, int y) {
    int xroot = find(parent, x);
    int yroot = find(parent, y);
    parent[xroot] = yroot;
}

int compareEdges(const void* a, const void* b) {
    return ((struct Edge*)a)->weight - ((struct Edge*)b)->weight;
}
```

```
void kruskalMST(struct Graph* graph) {
    int parent[MAX_VERTICES];
    for (int i = 0; i < graph->numVertices; i++) {
        parent[i] = -1;
    }

    qsort(graph->edges, graph->numEdges, sizeof(graph->edges[0]), compareEdges);

    printf("Edge   Weight\n");
    for (int i = 0; i < graph->numEdges; i++) {
        int srcRoot = find(parent, graph->edges[i].src);
        int destRoot = find(parent, graph->edges[i].dest);

        if (srcRoot != destRoot) {
            printf("%d - %d    %d\n", graph->edges[i].src, graph->edges[i].dest,
graph->edges[i].weight);
            unionSets(parent, srcRoot, destRoot);
        }
    }
}

int main() {
    struct Graph graph;
    int numVertices, numEdges;

    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);

    printf("Enter the number of edges: ");
    scanf("%d", &numEdges);

    initializeGraph(&graph, numVertices, numEdges);

    printf("Enter edges and weights (source destination weight):\n");
    for (int i = 0; i < numEdges; i++) {
        scanf("%d %d %d", &graph.edges[i].src, &graph.edges[i].dest, &graph.edges[i].weight);
    }

    kruskalMST(&graph);

    return 0;
}
```

Output:

```
Enter the number of vertices: 4
Enter the number of edges: 6
Enter edges and weights (source destination weight):
4 6 7 8
62 5 5 6
1 2 3 4
5 6 7 2
6 2 5 6
Edge   Weight
5 - 6    1
2 - 6    2
2 - 3    4

--------------------------------
Process exited after 24.32 seconds with return value 0
Press any key to continue . . .
```