4. Graph traversals: BFS & DFS (matrix and adjacency list)

Description: Implement BFS and DFS for adjacency matrix and adjacency list representations.

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

// Simple adjacency matrix BFS/DFS
void bfs_matrix(int n, int adj[n][n], int src){
    int *visited = calloc(n, sizeof(int)); int queue[n], front=0, rear=0;
    visited[src]=1; queue[rear++]=src;
    printf("BFS: ");
    while(front<rear){ int u=queue[front++]; printf("%d ", u);
        for(int v=0;v<n;v++) if(adj[u][v] && !visited[v]){ visited[v]=1;
queue[rear++]=v; }
    }
    printf("\n"); free(visited);
}

void dfs_matrix_util(int n, int adj[n][n], int u, int visited[]){
visited[u]=1; printf("%d ", u);
    for(int v=0;v<n;v++) if(adj[u][v] && !visited[v]) dfs_matrix_util(n, adj,
v, visited);
}
void dfs_matrix(int n, int adj[n][n], int src){ int *visited =
calloc(n,sizeof(int)); printf("DFS: "); dfs_matrix_util(n,adj,src,visited);
printf("\n"); free(visited); }

// Adjacency list representation
typedef struct Node{ int v; struct Node* next;} Node;
Node* newNode(int v){ Node* node = malloc(sizeof(Node)); node->v=v;
node->next=NULL; return node; }

void addEdgeList(Node* adj[], int u, int v){ Node* node = newNode(v);
node->next = adj[u]; adj[u] = node; }

void bfs_list(Node* adj[], int n, int src){ int *visited =
calloc(n,sizeof(int)); int q[n], front=0, rear=0; visited[src]=1;
q[rear++]=src; printf("BFS list: ");
    while(front<rear){ int u=q[front++]; printf("%d ", u); for(Node*
p=adj[u]; p; p=p->next) if(!visited[p->v]){ visited[p->v]=1; q[rear++]=p->v;
} }
    printf("\n"); free(visited);
}

void dfs_list_util(Node* adj[], int u, int visited[]){ visited[u]=1;
```

```
printf("%d ", u); for(Node* p=adj[u]; p; p=p->next) if(!visited[p->v])
dfs_list_util(adj, p->v, visited); }
void dfs_list(Node* adj[], int n, int src){ int *visited =
calloc(n,sizeof(int)); printf("DFS list: "); dfs_list_util(adj, src,
visited); printf("\n"); free(visited); }

int main(){
    int n; printf("Enter number of vertices: "); scanf("%d", &n);
    int adj[n][n]; for(int i=0;i<n;i++) for(int j=0;j<n;j++) adj[i][j]=0;
    int m,u,v; printf("Enter number of edges: "); scanf("%d", &m);
    printf("Enter edges (u v) 0-based:\n");
    for(int i=0;i<m;i++){ scanf("%d %d", &u, &v); adj[u][v]=1; /*for
undirected*/ adj[v][u]=1; }
    int src; printf("Enter source vertex: "); scanf("%d", &src);
    bfs_matrix(n, adj, src); dfs_matrix(n, adj, src);
    Node* adjList[n]; for(int i=0;i<n;i++) adjList[i]=NULL;
    for(int i=0;i<n;i++) for(int j=0;j<n;j++) if(adj[i][j])
addEdgeList(adjList, i, j);
    bfs_list(adjList, n, src); dfs_list(adjList, n, src);
    return 0;
}
```
