





**（3）**

typedef struct {

    int numVertices;  // 顶点数量

    int\*\* adjacencyMatrix;  // 邻接矩阵

} Graph;

// 创建图

Graph\* createGraph(int numVertices) {

    Graph\* graph = (Graph\*)malloc(sizeof(Graph));

    graph->numVertices = numVertices;

    // 创建邻接矩阵

    graph->adjacencyMatrix = (int\*\*)malloc(numVertices \* sizeof(int\*));

    for (int i = 0; i < numVertices; i++) {

        graph->adjacencyMatrix[i] = (int\*)calloc(numVertices, sizeof(int));

    }

    return graph;

}

// 添加边

void addEdge(Graph\* graph, int src, int dest, int weight) {

    graph->adjacencyMatrix[src][dest] = weight;

    graph->adjacencyMatrix[dest][src] = weight;  // 如果是无向图，还需要添加反向边

}

**四、算法设计题**

1. 试基于图的深度优先搜索策略写一算法，判别以邻接表方式存储的有向图中是否存在由顶点vi到顶点vj的路径（i≠j）。

// 判断路径是否存在

int isPathExists(Graph\* graph, int start, int target) {

    if (start == target) {

        return 0; // 顶点相同，不存在路径

    }

    int visited[MAX\_VERTICES] = {0}; // 记录顶点的访问状态

    return DFS(graph, start, target, visited);

}