# 引言

杨子凡

Jul 16, 2025

图数据结构在计算机科学中扮演着至关重要的角色，其核心价值在于高效建模复杂关系网络。社交网络中的好友关系、地图导航中的路径规划以及推荐系统中的用户行为分析，都依赖于图的强大表达能力。与线性结构如数组和链表不同，图突破了单一序列的限制；相较于半线性结构如树，图允许任意顶点间的多对多连接，消除了层级约束。本文旨在构建一个完整的认知体系，从理论基础到代码实现，深入剖析图的物理存储、核心操作和实际应用场景，帮助读者掌握这一关系建模的终极工具。

## 1 顶点与边的数学定义

图由顶点（Vertex）和边（Edge）组成，其中顶点代表实体对象，边表示实体间的关系。数学上，一个图可定义为有序对 $G = (V, E)$，其中 $V$ 是顶点集合，$E$ 是边集合。每条边连接两个顶点，若顶点 $u$ 和 $v$ 相连，则记为 $(u, v)$。这种抽象模型能灵活适应各种场景，例如在社交网络中，顶点表示用户，边表示好友关系。

## 2 关键分类标准

图的分类依据多个维度：有向图与无向图的区别体现在边的方向上，有向图如网页链接（从源页面指向目标），无向图如社交好友关系（双向对称）；加权图与无权图则以边上的数值权重为区分，加权图用于路径距离建模，无权图适用于简单关系如好友连接；连通图与非连通图关注整体连接性，非连通图在岛屿问题中常见，表示孤立的子图群。这些分类直接影响工程实现的选择。

## 3 进阶术语

度（Degree）指一个顶点的邻居数量，在有向图中细分为入度（指向该顶点的边数）和出度（从该顶点出发的边数）；路径（Path）是从起点到终点的边序列，环（Cycle）是首尾相接的闭环路径；连通分量描述图中最大连通子集。稀疏图与稠密图的工程意义重大，稀疏图边数 $E$ 远小于顶点数平方 $V^2$（即 $E \ll V^2$），适合邻接表存储，而稠密图 $E \approx V^2$ 则优先邻接矩阵，以减少查询开销。

## 4 邻接矩阵

邻接矩阵使用二维数组实现，其中 `matrix[i][j]` 存储顶点 $i$ 到 $j$ 的边信息（如权重或存在标志）。该方法适用于稠密图，因为边存在判断时间复杂度为 $O(1)$，但空间复杂度高达 $O(V^2)$，对大规模图不友好。例如，在社交网络分析中，若用户数巨大且连接稀疏，矩阵会浪费大量内存存储零值。

## 5 邻接表

邻接表采用哈希表与链表或数组的组合，结构为 Map<Vertex, List<Edge>>，每个顶点映射到其邻居列表。此方法高效处理稀疏图，遍历邻居的时间复杂度为 $O(\text{degree})$，空间复杂度为 $O(V+E)$，支持动态扩展。例如，在推荐系统中，用户的好友列表可快速添加或删除，避免矩阵的静态限制。

## 6 代码选择依据

数据结构选择取决于图密度：稠密图优先矩阵以优化查询，稀疏图选用邻接表节省空间。时间与空间权衡需具体分析，如高频边查询场景中，矩阵的 $O(1)$ 优势显著；而内存敏感应用中，邻接表的 $O(V+E)$ 更可取。工程实践中，需结合查询频率和存储成本制定策略。

顶点操作包括 addVertex(key) 和 removeVertex(key)。添加顶点时，邻接表通过哈希表动态扩容，时间复杂度均摊 $O(1)$；删除顶点需级联处理关联边，有向图中还需清理入边，避免内存泄漏。边操作如 addEdge(src, dest, weight) 在邻接表中尾部插入邻居，权重可选；删除边 removeEdge(src, dest) 涉及链表节点移除或矩阵置零。关键查询操作中，getNeighbors(key) 直接返回邻接链表；hasEdge(src, dest) 在矩阵中为 $O(1)$，但邻接表需 $O(\text{degree})$ 遍历；度计算在无向图直接计数邻居数，有向图则分离入度和出度统计。

## 7 深度优先搜索（DFS）

DFS 通过递归栈或显式栈实现，优先深入探索路径分支。递归版本隐式使用调用栈，显式栈则手动管理顶点访问顺序；核心是 visited 标记策略，防止重复访问。应用场景包括拓扑排序（任务依赖解析）和环路检测（判断图是否无环）。例如，在编译器优化中，DFS 用于识别代码块间的循环依赖。

## 8 广度优先搜索（BFS）

BFS 基于队列实现，按层遍历顶点，确保最短路径优先。队列初始化后，逐层访问邻居，并用 visited 集合记录状态；路径回溯通过 parent 指针实现。应用包括无权图最短路径（如社交网络的三度好友推荐）和关系扩散模型。例如，在疫情模拟中，BFS 追踪感染传播层级。

## 9 核心代码片段

以下 BFS 实现示例展示遍历逻辑：使用队列和 visited 集合，queue.extend 添加未访问邻居。代码中，start 为起点，yield 输出访问顺序，确保高效性和正确性。此片段适用于社交网络分析，计算用户影响力范围。

以下 Python 类实现图的邻接表表示，支持有向/无向图和 BFS 遍历。

```
import collections

class Graph:
    def __init__(self, directed=False):
```

```
        self.adj_list = {} # 哈希表存储顶点及其邻居字典
        self.directed = directed # 有向图标志

    def add_vertex(self, vertex):
        if vertex not in self.adj_list: # 防止顶点重复添加
            self.adj_list[vertex] = {} # 初始化空邻居字典

    def add_edge(self, v1, v2, weight=1):
        self.add_vertex(v1) # 自动添加不存在的顶点
        self.add_vertex(v2)
        self.adj_list[v1][v2] = weight # 添加边及权重
        if not self.directed: # 无向图需对称添加反向边
            self.adj_list[v2][v1] = weight

    def bfs(self, start):
        visited = set() # 记录已访问顶点
        queue = collections.deque([start]) # 队列初始化
        while queue:
            vertex = queue.popleft() # 出队处理
            if vertex not in visited:
                yield vertex # 返回当前顶点
                visited.add(vertex)
                neighbors = self.adj_list[vertex].keys() # 获取邻居集合
                queue.extend(neighbors - visited) # 添加未访问邻居
```

代码解读：__init__ 方法初始化邻接表为字典，directed 参数控制图类型；add_vertex 检查顶点存在性后添加，避免冗余；add_edge 自动处理顶点添加，并根据有向性对称设置边；bfs 方法使用队列和集合实现遍历，yield 生成访问序列，neighbors - visited 确保只添加新邻居，优化性能。此实现适用于动态图场景，如实时推荐系统。

时间复杂度方面，添加顶点或边在邻接表中均摊 $O(1)$（哈希表操作）；查询边 hasEdge 为 $O(\text{degree})$，邻接矩阵则为 $O(1)$。空间优化技巧包括用动态数组替代链表提升缓存局部性，或采用稀疏矩阵压缩存储如 CSR 格式（Compressed Sparse Row），将空间降至 $O(V+E)$。工业级考量涉及并发处理，例如读写锁（如 Python 的 threading.RLock）保护共享图状态；持久化方案中，邻接表序列化为 JSON 或二进制格式，便于存储和恢复。

## 10 社交网络分析

在社交网络中，图模型用户为顶点、好友关系为边。BFS 用于计算三度好友推荐：从用户起点层序遍历，识别二级邻居作为潜在推荐对象；连通分量分析可发现兴趣社群，例如通过 DFS 识别互相关联的用户群组，提升社区划分效率。

# 11 路径规划引擎

加权图建模交通网络，顶点为路口，边权重表示距离或时间。Dijkstra 算法基于此实现最短路径搜索：优先队列管理顶点，逐步松弛边权重。例如，导航系统中，从起点到终点的最优路径计算依赖于图的加权边动态更新。

# 12 任务调度系统

有向无环图（DAG）表示任务依赖，顶点为任务，边为执行顺序。拓扑排序通过 DFS 实现，输出线性序列确保无循环依赖；应用于 CI/CD 流水线，自动化任务调度避免死锁。

进阶算法包括最短路径的 Dijkstra（单源）和 Floyd-Warshall（全源对）、最小生成树的 Prim 和 Kruskal（网络优化）、强连通分量的 Kosaraju（有向图分析）。图数据库如 Neo4j 采用原生图存储理念，优化遍历性能；图神经网络（GNN）入门概念结合深度学习，用于节点分类或链接预测，拓展至推荐系统增强。

图作为关系建模的终极武器，其核心价值在于灵活表达复杂交互。实现选择需权衡时间、空间与工程复杂度：邻接表适于稀疏动态图，矩阵优化稠密查询；实际应用中，没有普适最优结构，只有针对场景的定制方案。未来发展中，图算法与 AI 融合将开启更智能的关系分析时代。