LaTeX 图算法笔记

作者：李明

example@example.edu

如果你有任何问题或评论，可以通过邮件联系我。

Happy LaTeX-ing!

前言

本书是一份关于图搜索算法的简短笔记，面向已经学过基础数据结构、但还没有系统学习图算法的读者。每一章都配有一段可编译运行的伪代码，帮助读者把定义和实现联系起来。

李明

2026 年

目录

第一章 图的基本概念 **1**

1.1 广度优先搜索 1

1.2 深度优先搜索 2

第二章 最短路径 **3**

第一章　图的基本概念

图 $G = (V, E)$ 由顶点集 V 和边集 E 组成。本章介绍广度优先搜索（BFS）与深度优先搜索（DFS）两种最基本的遍历算法。

1.1　广度优先搜索

广度优先搜索从起点开始，逐层向外扩展，使用一个队列维护待访问的顶点。

Algorithm 1: 广度优先搜索（BFS）

Input: 图 G，起点 s

Output: 从 s 可达的所有顶点及其最短距离

初始化队列 $Q \leftarrow \{s\}$，$\mathrm{dist}[s] \leftarrow 0$;
while Q 非空 **do**
    $u \leftarrow Q.\mathrm{dequeue}()$;
    foreach v 使得 $(u, v) \in E$ **do**
        if v 未被访问 **then**
            $\mathrm{dist}[v] \leftarrow \mathrm{dist}[u] + 1$;
            $Q.\mathrm{enqueue}(v)$;
        end
    end
end

下面是一段 Python 实现，对应上面的伪代码：

```
from collections import deque

def bfs(graph, start):
    dist = {start: 0}
    queue = deque([start])
    while queue:
        u = queue.popleft()
        for v in graph[u]:
            if v not in dist:
```

```
                dist[v] = dist[u] + 1
                queue.append(v)
    return dist
```

1.2 深度优先搜索

深度优先搜索沿着一条路径尽可能深入，直到无法前进才回溯，通常用递归或显式栈实现。

```
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)
    for v in graph[start]:
        if v not in visited:
            dfs(graph, v, visited)
    return visited
```

BFS 保证找到的是最短路径（按边数计），而 DFS 更适合探索连通性和拓扑结构，例如检测图中是否存在环。

第二章　最短路径

当图的边带有权重时，BFS 不再保证给出最短路径，需要使用 Dijkstra 算法。

Algorithm 2: Dijkstra 最短路径算法

Input: 带权图 G，起点 s

Output: 从 s 到每个顶点的最短距离

```
dist[s] ← 0，其余顶点距离设为 ∞;
将所有顶点加入优先队列 Q，键值为 dist;
while Q 非空 do
    u ← Q.extractMin();
    foreach v 使得 (u, v) ∈ E，权重为 w(u, v) do
        if dist[u] + w(u, v) < dist[v] then
            dist[v] ← dist[u] + w(u, v);
            更新 Q 中 v 的键值;
        end
    end
end
```

该算法的正确性依赖于所有边权重非负；若存在负权边，需改用 Bellman–Ford 算法。