数据结构 lab-8 报告

林子开 21307110161

2023 年 11 月 21 日

目录

1 算法思路说明

首先，题目提供的是工序流程图，是有向无环图，各个节点的最长或最短距离的更新关系是确定的。因此，可以先对图进行**拓扑排序**，然后基于拓扑序进行距离计算。其中，拓扑排序基于深度优先搜素完成。该部分的算法复杂度为 $\mathcal{O}(V+E)$

其次，虽然题目要求找的是最长带权距离，但如果将所有的边的权重都取成相反数，也即 $(u,v,w) \to (u,v,-w)$，那么，**问题等价于求权重取相反数后的图的最短路径**。因此，仍然可以使用基于拓扑序求有向无环图最短路径的算法, 找到权重取反后的图的“最短的”路径。最后计算距离时，再将该路径长度取相反数，即可得到最长路径距离。该部分的复杂度为 $\mathcal{O}(V+E)$。

总的算法复杂度为 $\mathcal{O}(V+E)$。

2 Python 代码主要功能介绍

在文件 `find-longest-path.py` 文件中，我定义了两个类，分别是表示图节点的 `vertex` 类，和表示有向图的 `Graph` 类。

在 `Graph` 中，基于深度优先搜索（`DFS` 方法）实现了拓扑排序（`topologicalSort` 方法）。基于拓扑排序的结果，实现了 `findShortestPath` 方法，该方法能够找到图中从起点开始到达所有其他点的最短路径。

最后，实现了 `findLongestPath` 方法，该方法先将图中所有边的**权重取相反数**，然后调用 `findShortestPath` 找到权重取反后的图的“最短”路径（也即原图的最长路径），并将该

路径，以及路径对应的长度返回。

`criticalPath` 是辅助性的函数，用于格式化地建立 `Graph` 实例，并打印路径和路径长度。

3 测试用例实验结果

说明：由于题目只要求**找出一条**最长的带权路径，因此以下给出的答案可能不唯一。

3.1 测试用例 1

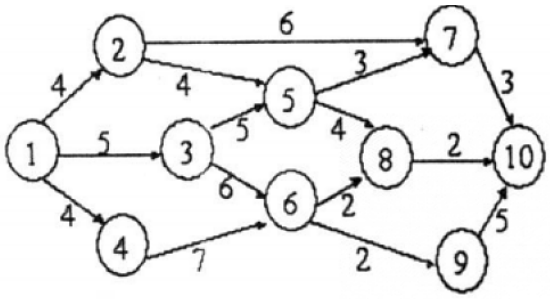

图 1: 测试用例 1 的示意图

最长路径：1 ⟶ 4 ⟶ 6 ⟶ 9 ⟶ 10。路径长度为：18。

3.2 测试用例 2

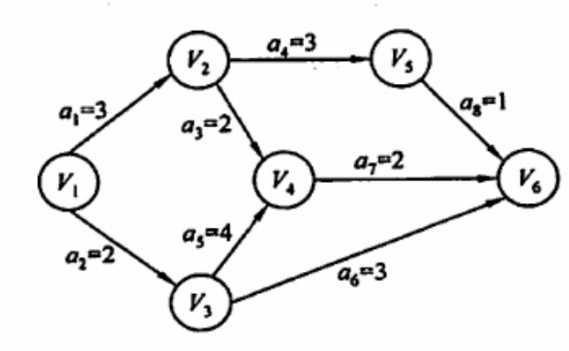

图 2: 测试用例 2 的示意图

最长路径：V1 ⟶ V3 ⟶ V4 ⟶ V6。路径长度为：8。

3.3 测试用例 3

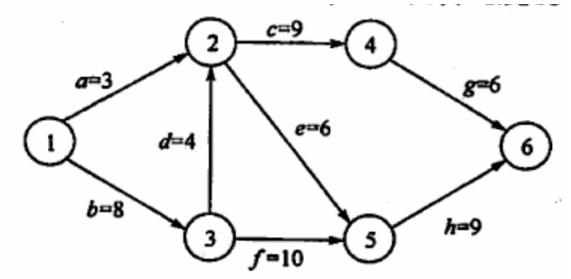

图 3: 测试用例 3 的示意图

最长路径：1 ⟶ 3 ⟶ 5 ⟶ 6。路径长度为：27。

3.4 测试用例 4

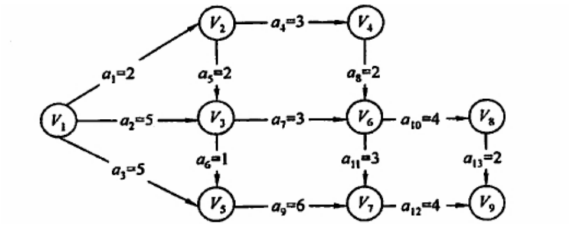

图 4: 测试用例 4 的示意图

最长路径：V1 ⟶ V3 ⟶ V5 ⟶ V7 ⟶ V9。路径长度为：16。