信网研究专题课-Dgrees

罗叶菲
19211399
通信 1908 班

Abstract

在此次信息网络综合专题研究中，我研究了哈弗 CS50 中 lecture0 search 项目中的 degrees 题目，根据课堂所给代码，加上自己的改编，实现了题目要求。

1 项目背景

根据凯文培根的六度游戏，好莱坞电影业的任何人都可以在六个步骤内连接到凯文培根，每个步骤都包括找到一部由两位演员共同出演的电影。

在这个问题中，我们感兴趣的是通过选择连接他们的电影序列来找到任意两个演员之间的最短路径。例如，詹妮弗 · 劳伦斯和汤姆 · 汉克斯之间的最短路径是 2：詹妮弗 · 劳伦斯和凯文 · 培根都出演了《X 战警：头等舱》，而凯文 · 培根和汤姆 · 汉克斯都出演了《阿波罗 13》。”我们可以将其视为一个搜索问题：我们的状态是人。

我们的行为是电影，它将我们从一个演员带到另一个演员（确实，一部电影可以将我们带到多个不同的演员，但这对于这个问题来说是可以的）。我们的初始状态和目标状态由我们试图联系的两个人定义。通过使用广度优先搜索，我们可以找到从一个演员到另一个演员的最短路径。

2 项目分析

分发代码包含两组 CSV 数据文件：一组在大目录中，一组在小目录中。每个都包含具有相同名称和相同结构的文件，但较小的是一个小得多的数据集，以便于测试和实验。每个数据集由三个 CSV 文件组成。CSV 文件（如果不熟悉）只是一种以基于文本的格式组织数据的方式：每一行对应一个数据条目，行中的逗号分隔该条目的值。

（1）打开 small/people.csv。你会看到每个人都有一个唯一的 id，与他们在 IMDb 数据库中的 id 相对应。他们也有名字和出生年份。(id 为 102 的人的 name 和 birth)

（2）接下来，打开 small/movies.csv。您会在此处看到，除了标题和电影发行年份之外，每部电影还有一个唯一的 ID。(id 为 xx 的电影的 title 和 year)

（3）现在，打开 small/stars.csv。此文件在 people.csv 中的人物和 movies.csv 中的电影之间建立关系。每行是一对 person_id 值和 movie_id 值。例如，第一行（忽略标题）表明 id 为 102 的人出演了 id 为 104257 的电影。对照 people.csv 和 movies.csv 进行检查，你会发现这一行是说 Kevin Bacon 出演电影《好男人》。

（4）接下来，看看 degrees.py。在顶部，定义了几个数据结构来存储来自 CSV 文件的信息。names 字典是一种通过名字查找人的方法：它将名字映射到一组相应的 ids（因为多个演员可能有相同的名字）。people 字典将每个人的 id 映射到另一个字典，其中包含该人的姓名、出生年份和他们出演的所有电影的值。电影字典将每部电影的 id 映射到另一个字典，其中包含该电影标题的值，发行年份，以及所有电影明星的集合。load_data 函数将 CSV 文件中的数据加载到这些数据结构中。

（5）该程序中的 main 函数首先将数据加载到内存中（加载数据的目录可以通过命令行参

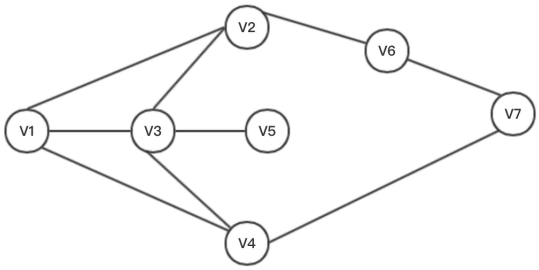

图 1: 初始连通图

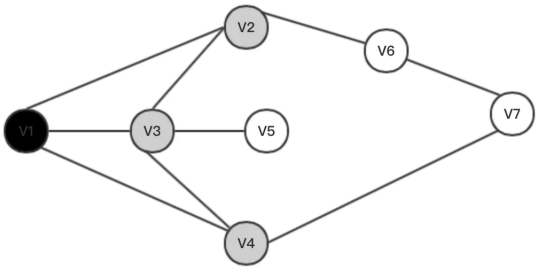

图 3: 第一层搜索

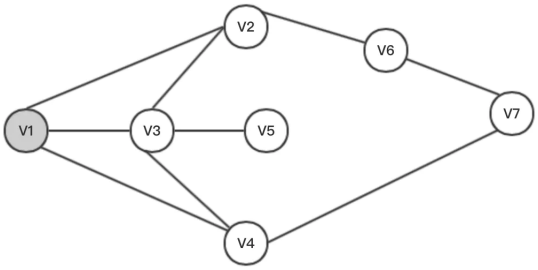

图 2: 设立初始点

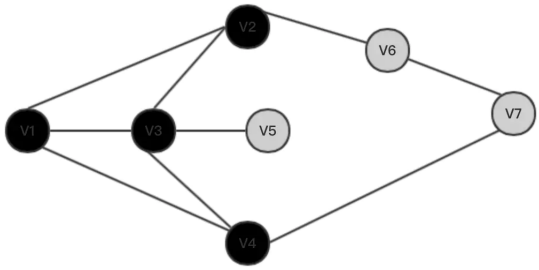

图 4: 继续扩散搜索

数指定）。然后，该函数提示用户输入两个名称。（源与目标）person_id_for_name 函数检索任何人的 id（并在多人同名的情况下处理提示用户澄清）。然后该函数调用 shortest_path 函数计算两个人之间的最短路径，并打印出路径。

3 最短路由算法

3.1 目标

完成 shortest_path 函数的实现，使其返回从 id 为 source 的人到 id 为 target 的人的最短路径。

（1）假设存在从源到目标的路径，您的函数应该返回一个列表，其中每个列表项是从源到目标的路径中的下一个 (movie_id, person_id) 对。每对应该是两个字符串的元组。例如，如果 shortest_path 的返回值为 [(1, 2), (3, 4)]，则表示源与人 2 出演了电影 1，人 2 与人 4 出演了电影 3，并且人 4 是目标。

（2）如果从源到目标有多个最小长度的路径，您的函数可以返回其中任何一个。

（3）如果两个参与者之间没有可能的路径，您的函数应该返回 None。

（4）您可以调用 neighbors_for_person 函数，该函数接受一个人的 id 作为输入，并返回一组 (movie_id, person_id) 对，表示所有与给定人一起出演电影的人。

3.2 算法原理

广度优先搜索，别名又叫 BFS，属于一种盲目搜寻法，目的是系统地展开并检查图中的所有节点，以找寻结果。换句话说，它并不考虑结果的可能位置，彻底地搜索整张图，直到找到结果为止。所谓广度，就是一层一层的，向下遍历，层层堵截。基本过程，BFS 是从根节点开始，沿着树 (图) 的宽度遍历树 (图) 的节点。如果所有节点均被访问，则算法中止。一般用队列数据结构来辅助实现 BFS 算法。

基本步骤：

（1）给出一连通图，如图 1，初始化全是白色（未访问）；

（2）搜索起点 V1（灰色），如图 2；

（3）已搜索 V1（黑色），即将搜索 V2，V3，V4（标灰），如图 3；

（4）对 V2，V3，V4 重复以上操作，如图

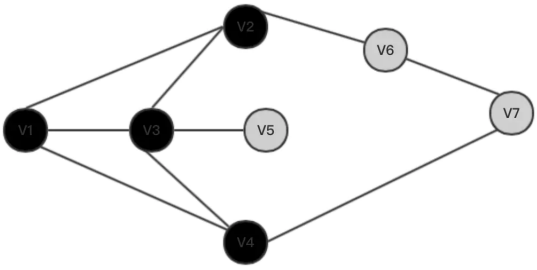

图 5: 搜索完毕

```
Loading data...
Data loaded.
Name: Demi moore
Name: Sally Field
2 degrees of separation.
1: Demi Moore and Michael Caine starred in Flawless
2: Michael Caine and Sally Field starred in Surrender
```

图 6: 结果展示

4;

（5）直到终点 V7 被染灰，即被搜索到，终止程序, 如图 5;

4 项目结果

4.1 结果展示

例如，我们搜索 Demi Moore 和 Sally Field 的关系，可得如图 6 所示. 源节点和目的节点之间只经过一个节点，即 Michael Caine，这样的搜索结果符合项目要求。

4.2 算法总结

广度优先搜索算法将同时跟踪多个方向，在每个可能的方向上走一步，然后在每个方向上走第二步。在这种情况下，边界作为队列数据结构进行管理。需要记住的是“先进先出”。在这种情况下，所有新节点都按顺序相加，并且根据先添加的节点来考虑节点（先到先得!）。这导致搜索算法在每个可能的方向上迈出一步，然后在任何一个方向上迈出第二步。

优点:

该算法保证找到最优解。

缺点:

该算法几乎可以保证比最短运行时间更长

在最坏的情况下，这个算法需要尽可能长的时间来运行

5 项目总结

这个项目还算比较好做，数据导入部分都已经写好, 我们只需要完成最短路由算法即可。广度优先算法其实并不是很难，老师在课堂上也清楚讲述，虽然思路清晰，但是要转化成机器语言仍有一定难度，特别是我开始只知道最简单的 Python，对于队列的代码还不是很熟，BFS 算法的关键特点之一就是利用队列先进先出的特点进行搜索。每层循环结束后，开始下层循环。每次循环到一个节点，该节点就出队。同时，在操作中，可以多定义一个队列，看似增加内存, 实际上省去了搜索已经搜过的位置，节约了内存和时间。

项目视频清楚地展示了代码演示过程。代码和视频均已放在随行的压缩包中。

References

https://cs50.harvard.edu/ai/2020/notes/0/

https://blog.csdn.net/u010772377/article/details/118246396