信息网络综合专题研究课 M3 大作业项目报告

Anonymous EMNLP submission

Abstract

In the third module of the Information Network Comprehensive Special Topic Research Course, our teacher explained some AI algorithms to us, which aroused our interest greatly. And after the class we tried to solve some questions via those algorithms. And here it is.

1 Introduction

本次大作业的选题为哈佛 CS50 课程[1]中的第 0 课搜索的课后项目一：degrees。

根据六度分隔原理，世界上的人都可以在人际网络中以六跳的距离与另一个人建立联系。题目给出了所有好莱坞演员包括出生日期和出演电影在内的基本信息，并认为出演统一电影的演员间存在联系且距离为一，要求求解者利用广度优先的算法寻找联系任意两位演员的最短人际网路径。

2 The files

网站提供了两份不同大小的“演员-电影”文件，每一份文件均包括了三个.CSV 文件，分别存放了演员信息、电影信息和演员出演的电影，提供的文件已对演员和电影进行了编号，方便程序读取，此外，额外提供的演员的出生日期信息主要用于区别同名的演员。

此外，网站还提供了两份.py 代码文件。其中，degrees.py 文件中包含了主函数，且读取文档信息、确定演员编号和寻找邻居节点的自定义函数已事先编写完成。util.py 文件主要包含了三个类供求解者选用，分别为名为 Node 的节点对象、堆栈操作和队列操作。

[1] https://cs50.harvard.edu/ai/2020/1

3 My background

由于在此之前没有接触过 PYthon，本次大作业的主要任务集中在了搭建编程环境和熟悉编程语言上。

所幸源代码中所提供的读取文档信息的函数和堆栈、列表操作大大减少了在编程过程中的实际工作量，最后实际留给解决者的任务只是简单的数据操作和对已有函数的调用。大部分循环语句和逻辑语句基本都是可以直接套用 C，对于源代码中涉及到的 class 类和列表等各类元素则主要在需要使用和遇到 debug 报错的时候进行面向问题的学习。

问题所涉及的核心算法是广度优先的搜索算法，该算法在课上已做了详细介绍，算法本身并不复杂，最后用差不多十多行的篇幅就能完成。由于在课外实现过用 C 编写的利用 Dijkstra 算法求解北京地铁乘坐的最短路径，对理解和实现本问题所涉及的广度优先算法也没有太多困难。

4 My work

根据广度优先算法分别构建节点（node）、待考察节点队列（frontier）和已遍历节点队列（explored）并实现相应操作，最终输出最短路径是待完成的主要任务。其对应的代码部分也就是所给源代码中待完善的自定义函数 shortest path(source, target)。

算法的基本流程如下：

首先，将节点的 state 定义为当前演员的

ID，parent 为通过某一电影与其相关联的上一个演员，而 action 则为联系该两名演员的电影。

其后，根据源节点找到相邻节点并加入到待考察节点队列，然后依次取出队列中的节点判断是否为目标节点，如若判断结果为否，则将其相邻节点加入至队列后，并将该节点加入已遍历的节点队列，如此循环往复，直至找到目标节点。

值得注意的是，根据广度优先算法，待考察的节点一定是通过队列来存储，而已遍历节点则可以用任意形式存储，在本例中采用了堆栈。为了避免产生环路，在将相邻节点加入待考察队列前，应先判断该节点是否已经遍历过或是否已经被加入到了待考察节点队列中，如若是，则直接丢弃。

找到目标节点后，根据当前目标节点的 parent 回溯至上一节点，在已遍历的节点队列中找到该节点，并添加到存放路径的列表 path 中，再根据该节点的 parent 回溯至又上一节点，如此循环往复，直至回溯至源节点，此时就完成了对该最短路径的回溯。最后即可将该路径输出。

5 The results

由于源代码中默认读取的文件是“large”，也即较大的一个，而在实际的运行过程中，该算法需要较长的时间进行计算，基本已经达到了分钟级。由于运算时间较长，我换用了较小的另一个文件对程序能否正常工作进行测试。而这一状况显然是十分具有显示意义的：由于平台所提供的“large”文件就是一组真实的数据，也即该体量的数据量是完全真实存在的，也会是实际工程实践中必然会遭遇的，而鉴于其较长的处理时间，广度优先算法显然在应对该类问题时有明显的缺陷（至少在微型个人计算机上运行时时这样的）。此外，由于在源代码中寻找相邻节点的输出集合是一个无序的集合，也即针对同一个求解问题，相邻节点加入的顺序是随机的，并不相同，最终求解所需的计算时间也并不相同。

换言之，遍历搜索算法普遍在求解效率上存在缺陷，由于其求解过程是对所有的元素进行遍历，待求解问题的数据量会直接影响到求解的速度，由此，一些基于此的改进算法就显得尤为重要。

6 Links

代码链接：

https://github.com/dookie678/tree-holes/tree/dookie678-Harvard-AI50-degrees

演示视频链接：

https://www.bilibili.com/video/BV1cA4y1Z7gQ?p=1&share=1

（由于无法输入下划线，视频链接可能无法打开，完整链接已在分组表格中填写）

References