EAI 320

Practical Assignment 1

Compiled by Hugo Coppejans
Edited and reviewed by Johan Langenhoven

7 Februrary 2018

1 Scenario

The travelling salesman problem (TSP) asks the following question: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city? The TSP has several applications even in its purest formulation, such as planning, logistics, and the manufacture of microchips. Slightly modified, it appears as a sub-problem in many areas, such as DNA sequencing. In these applications, the concept city represents, for example, customers, soldering points, or DNA fragments, and the concept distance represents travelling times or cost, or a similarity measure between DNA fragments. The TSP also appears in astronomy, as astronomers observing many sources will want to minimise the time spent slewing the telescope between the sources. In many applications, additional constraints such as limited resources or time windows may be imposed.

Assignment 1

For Assignment 1 you will investigate the effectiveness of two uninformed (brute force) search techniques, Depth First Search (DFS) and Breadth First Search (BFS) on a TSP problem.
Thus for this practical you have three major goals.

1. Implementing a search tree structure in python.
2. Implementing a DFS and BFS search strategy.
3. Writing a technical report on your findings.

For this assignment we will be using real world locations for the TSP problem. We want to visit each one of the locations, and end again at the starting location. Since this is for the real world, you can conceivably travel from any one location to another, thus this is a fully connected TSP problem.

Task 1

You have to create a search tree structure in Python that can be used by any type of search algorithm. For this assignment we will be using the following five locations:

1. University of Pretoria, Pretoria (Starting location)
2. CSIR, Meiring Naude Road, Pretoria
3. Armscor, Delmas Road, Pretoria
4. Denel Dynamics, Nellmapius Drive, Centurion
5. Air Force Base Waterkloof, Centurion

You are allowed to use any process of method to create the search tree, this is entirely up to you. The recommended way is using a Python Class to represent nodes in a tree. Each of the nodes are linked and a recursive function can be used to build the tree.

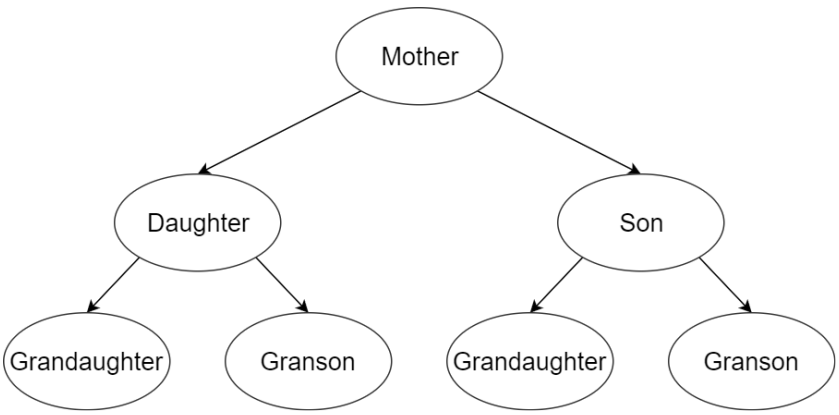
An example code segment is given below:

```
class node(object):
    def __init__(self,value,children = []):
        self.value = value;
        self.children = children;
```

and a tree can be created as follow:

```
tree = node("mother", [
        node("daughter",[
            node("granddaughter"),
            node("grandson")]),
        node("son",[
            node("granddaughter"),
            node("grandson")])
                    ]);
```

which results in the following node-tree:



An example of a tree.

For this assignment you will be using Google Maps to calculate the distances between two locations. We will be using the GoogleMaps API and googlemaps Python libraries. Make sure to install the googlemaps library for Python on your computer. We will require an API key from the Google Developers Console, specifically for this Assignment we will use the Google Maps Distance Matrix API. Look on ClickUp for the instruction on how to use the keys.

To install the googlemaps package, the following steps need to be followed:

1. Open Windows command prompt by pressing Windows-key and searching "cmd".
   - Alternatively, you can press Windows-key + X → Command prompt.
2. Enter:
   pip install googlemaps
   To install the googlemaps library.
3. After the package has been installed, start the Spyder IDE.

To utilize the Google Maps Distance Matrix API we use the following lines of code.

```
import googlemaps

gmaps = googlemaps.Client(
                key='AIzaSyAVz0wSdqPU3bV3y_tJCmV_uLmeZeHy_AQ' )

directions_result = gmaps.distance_matrix(
    "University of Pretoria, Pretoria",
    "CSIR, Meiring Naude Road, Pretoria")
```

This will return a dictionary with all the relevant data. Make sure to go and look up how to interact with a Python Dictionary and how to extract the data. For this practical we will only be working with the distances between locations, not the estimated travel time.

Task 2

Your next task will be to create a DFS and BFS search algorithm that incrementally searches the tree built in Task 1, using the DFS and BFS methodology. For each of the search algorithms you need to track the number of nodes visited by each, as this is a good indication of how well each performs. Make very sure that you count the number of nodes correctly, as this will be used to measure the level of your implementation. Nodes are only counted when they are evaluated, and each node must only be counted once.

Report

You have to write a short technical report for this assignment. Your report must be written in LaTeX. In the report you will give your results as well as provide a discussion on the results. Make sure to compare the depth and breadth search algorithms with each other and speculate on why each of them performed as they did.

Deliverables

- Write a technical report on your finding for this assignment.
- Include your code in the digital submission as an appendix, but leave it out for the hardcopy submission.

Instructions

- All reports must be in PDF format and be named report.pdf.
- Place the software in a folder called SOFTWARE and the report in a folder called REPORT.
- Add the folders to a zip-archive and name it `EAI320_prac1_studnr.zip`.
- All reports and simulation software must be e-mailed to **up.eai320@gmail.com**. no later than 16:00 on 15 February 2018. No late submissions will be accepted.
- Bring your hard copies to the practical session on Thursday, 15 February 2018, where they will be collected.
- Submit your report online on ClickUP using the TurnItIn link.

Additional Instructions

- Do not copy! The copier and the copyee (of software and/or documentation) will receive zero for both the software and the documentation.
- For any questions of appointments email me at **up.eai320@gmail.com**.
- Make sure that you discuss the results that are obtained. This is a large part of writing a technical report.

Marking

Your report will be marked as follow:

- 70% will be awarded for the full implementation of the practical and the subsequent results in the report. For partially completed practicals, marks will be awarded as seen fit by the marker. **Commented code allows for easier marking!**
- 30% will be awarded for the overall report. This includes everything from the report structure, grammar and discussion of results. The discussion will be the bulk of the marks awarded.