CSE 440/EEE 333/ETE 333: Artificial Intelligence (Section 1)
Fall 2015

Due date: Monday 12th, October, 2015

Assignment 1

1. (20 points). Consider the search tree shown in Figure 1. The number next to each edge is the cost of the performing the action corresponding to that edge. List the order in which nodes will be visited using:
   - breadth-first search.
   - depth-first search.
   - iterative deepening search.
   - uniform cost search.

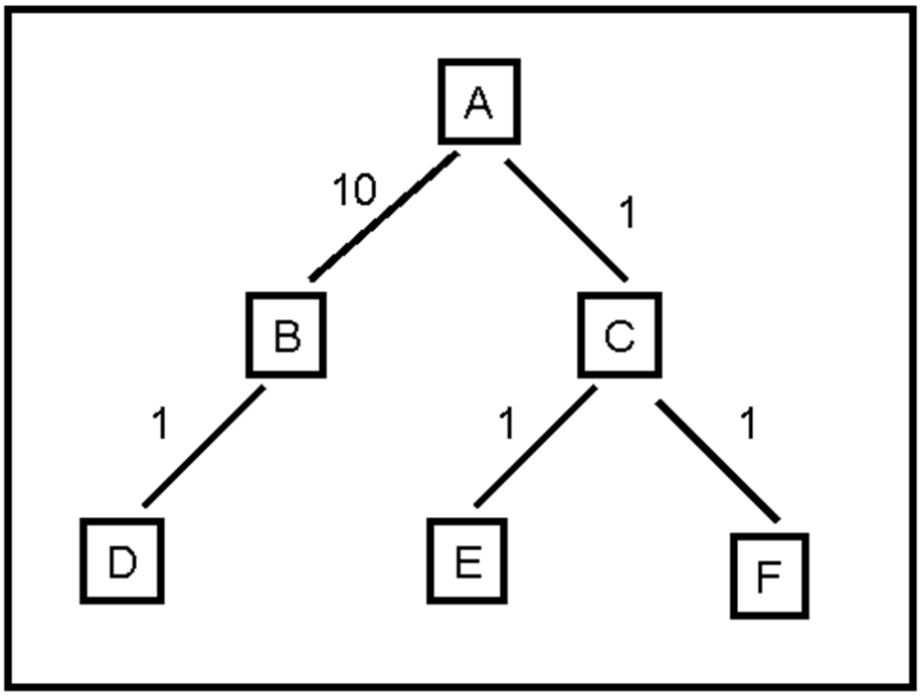

Figure 1. An example search tree.

How to submit

This part of the assignment must be hand written. Use A4 size paper and write on both sides.

At the front page make sure you write the following:
Name
ID
Course Code
Assignment No
Submission Date

Submit the assignment at the beginning of the class in due date. There will be no extension of the deadline. No late submission will be accepted.

You will not get full credit if you do not follow the above guidelines for submission.

2. (80 Points)

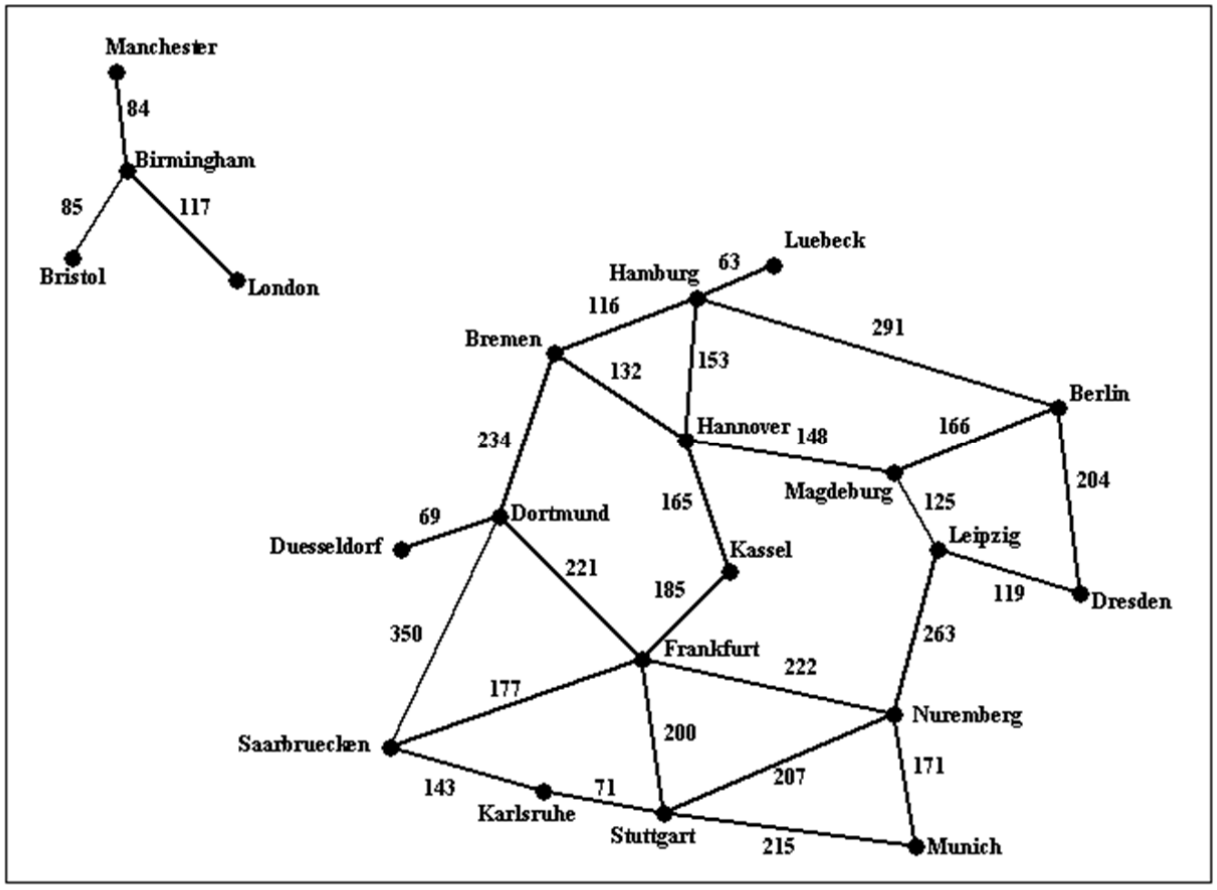

Figure 2: A visual representation of the road system described in file input1.txt.

Uninformed Search

Implement a search algorithm that can find a route between any two cities. Your program will be called find_route.py and will take exactly three inputs, as follows:

input_filename origin_city destination_city

An example is:

input1.txt Munich Berlin

input_filename is the name of a text file such as **input1.txt**, that describes road connections between cities in some part of the world. For example, the road system described by file **input1.txt** can be visualized in **Figure 2** shown above. You can assume that the input file is formatted in the same way as **input1.txt**: each line contains three items. The last line contains the items "END OF INPUT", and that is how the program can detect that it has reached the end of the file. The other lines of the file contain, in this order, a source city, a destination city, and the length in kilometers of the road connecting directly those two cities. Each city name will be a single word (for example, we will use New_York instead of New York), consisting of upper and lowercase letters and possibly underscores.

The program will compute a route between the origin city and the destination city, and will print out both the length of the route and the list of all cities that lie on that route. For example,

input1.txt Bremen Frankfurt

should have the following:

distance: 455 km
route:
Bremen to Dortmund, 234 km
Dortmund to Frankfurt, 221 km

and

input1.txt London Frankfurt

should have the following output:

distance: infinity
route:
none

For full credit, you should produce outputs identical in format to the above two examples.

Grading
The assignments will be graded out of 80 points.

- **40 points:** The program always finds a route between the origin and the destination, as long as such a route exists.
- **20 points:** In addition to the above requirement, the program terminates and reports that no route can be found when indeed no route exists that connects source and destination (e.g., if source is London and destination is Berlin, in the above example).
- **20 points:** In addition to the above requirements, the program always returns optimal routes. In other words, no shorter route exists than the one reported by the program.
- **Negative points:** penalty points will be awarded by the instructor generously and at will, for issues such as: submission not including precise and accurate instructions for how to run the code, wrong compression format for the submission, or other failures to comply with the instructions given for this assignment. Partial credit for incorrect solutions will be given ONLY for code that is well designed and well documented. Code that is badly designed and badly documented can still get full credit as long as it accomplishes the required tasks.

How to submit
Implementation in Python will only be accepted. The assignment should be submitted via **Engrade** using the **Turn in** option. Submit a **ZIPPED** directory called **assignment1.zip** (no other forms of compression accepted). The directory should contain source code file and a file called **readme.txt**, which should specify precisely:

Name, NSU ID of the student and assignment no. (These information must also be in you source code as comments)
How the code is structured.
How to run the code, including very specific compilation instructions, if compilation is needed. Insufficient or unclear instructions will be penalized by up to 20 points.