AI Course Fall HT 2016: Lab Assignments

- For passing the "Laborationer" part of the course, you need to successfully demonstrate and report your solutions to **all** given programming tasks.
- Assignments have to be implemented in **Python**. It is recommended to use a Python version that has some strong development environment. **There is a special lecture on Nov 1st at 17:15 (during the "Övning" time) introducing and reminding you of concepts in Python you will need for implementing the labs**.
- You may work in teams of **2** – working alone is also possible. Works done by more than 2 persons will not be accepted.
- There are two lab times per week – in Kronox as "Övning grupp1" and "Övning grupp 2". There is no mandatory presence during the lab times. We apply some self-organized distribution. Simply come to the group that is best for you. If you think you need more assistance then you may come to both. There is only one teacher, she tries to help everybody who needs help, but there might be some queues.
- Every assignment **needs to be demonstrated to the teacher during the lab times** for showing that your solution works, that you have understood what you did and that you did it yourself. You should be able to explain your code and justify the design decisions that you did.
- After demonstrating, you must write and hand-in a **report**. The role of the report is to document the decisions you did when implementing. It is an exercise in documenting an implementation on the appropriate level of abstraction. It is advisable to follow this structure:
    - Introduction: What was the problem in your own words?
    - Data structure design: Describe how your objects look like and justify why
    - Overall processes description: Describe the overall information flow through your program and how different elements fit together.

  The report should **not** contain the full source code, but a more high-level description, if necessary augmented with code snipplets. Diagrams alone are also not sufficient; they must be accompanied with text describing why you did it that way. A report can be pretty short (2-3 pages). There are no special requirements on formatting, yet it should be reasonable.
  The reports must also contain information about who did the labs and information about **all** material that you used. When you finished the report, send it together with your source code in one zip-file to franziska.klugl@oru.se. It must be clearly identifiable who you are.
- Expect that finishing the assignments **will take more time** than the supervised lab hours. You may also do the labs completely at home and just pass by the lab times for demonstrating. **During the lab times, the teacher will be there, also to help not just with Python-related issues, but also answering questions on algorithms, etc. Use this time as efficient as possible, prepare yourself.** You should make yourself familiar with the lecture slides/contents before you start solving the tasks. Lecture slides are downloadable via Blackboard. Pay special attention to the **pseudo code given in the lecture** for the main algorithms to be used in the labs. If you have problems understanding the pseudo code, ask the teacher; **do not invent your own versions of algorithms** based on what you remember from examples.
- Every assignment comes with **recommended date** when you should have finished it. This indicates how the lab work is synchronized with the lectures advance. The last scheduled lab times are on **January 5th**. Officially this is the last date to demonstrate any of your solutions. With

special appointment, it will be possible to demonstrate your solutions and hand-in your reports until January, 26th (about two weeks after exam). Further delayed demos and reports can be demonstrated and handed in within a time window of one week before to 2 weeks after **each of the re-exams**. There will be no appointment possible between. Contact the teacher in due time for settling a meeting for demonstration.

- There will be new assignments. Solutions to labs from previous years will be not accepted.
- Franziska will be personally available at lab times. At any other time, you may ask questions via email: franziska.klugl@oru.se. Emails will be answered as soon as possible. Additional personal consultation hours are possible by individual arrangement.

Task 1 Search for Routing

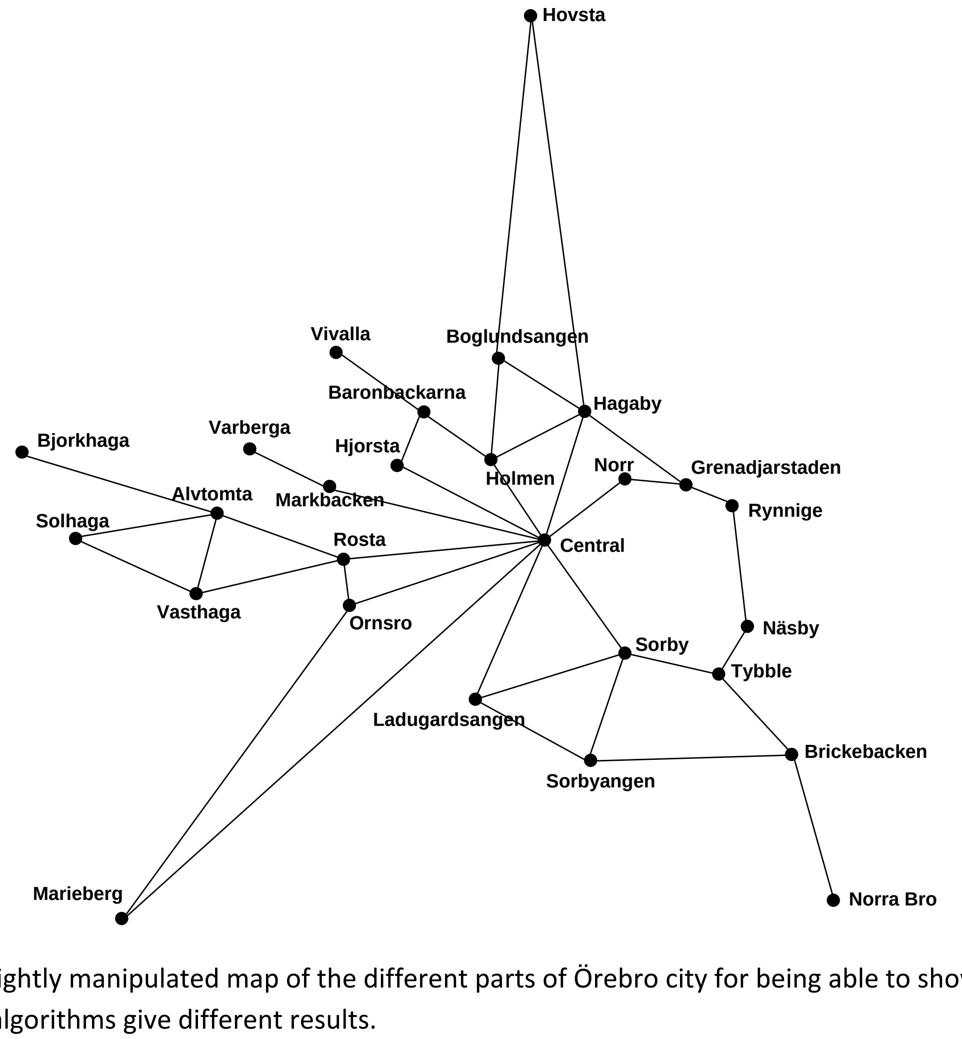
Recommended Deadline: Nov. 24th 2016

In this first lab assignment you will take a look at search algorithms for routing. This is a traditional application of search with a navigation graph as underlying data structure. For use in navigation systems such a graph is generated from a road map, for non-player characters in games the navigation graph is either generated from the geometry of the virtual world or explicitly created by the designer. A state in the navigation graph represents whether the traveling entity is placed at a particular node. Transitions between states are based on roads connecting the locations.

You should implement the algorithms **as presented in the lecture**.

a. Scenario Preparation

Your first task is to generate a data structure that is apt for representing and managing a road network as a search graph. In the subtasks after that, you will search for routes between locations in this graph using different algorithms. In the zip-file map.zip provided with the assignment text, there are two files: a file containing locations (“locations2016.csv”) and a file with links between locations (“connections2016.csv”). The file contains a roadmap such as the one depicted here:



This is a slightly manipulated map of the different parts of Örebro city for being able to show how different algorithms give different results.

The content of the “locations2016.csv” files has the following structure:

```
Name;x;y↵
```

The first line of the file contains the attribute names. From line 2, there is a location given in each line with a string representing the name of the place and the x and y coordinate (integer). The entries of different column are separated with the char “;”; new lines mean new location entries.

The content of the” connections2016.csv” file stores information about reachability between names of locations:

```
Location1;Location2↵
```

Again, the first line contains the headings of the different attributes. Every subsequent line describes a connection between two locations representing a road. The length of the link (representing the costs of going from one node to the connecting one) can be calculated from the coordinates of the two locations. Be aware that each line ends with Line-Feed special character.

Before starting to implement readers, it is always smart to have a look into the files for fully understanding their description.

You do not need to create a visualization of the data.

b) Iterative Deepening Search

In the lecture, several search methods have been given. Here your task is to implement the “best” uninformed search that you learnt: **iterative deepening search**.

Program first a depth-limited search – that means a depth-first-search function that gets a numeric limit as parameter and stops when a node shall be extended that is located deeper than the limit.

Then implement a wrapper around the depth-limited search that increases the depth if the goal has not been found searching to the current limit.

Provide a simple user interface that allows the user to enter the names of start and destination location, calculate the path using the iterative deepening search and print the path as a sequence of locations to pass.

How many nodes have been tested until you reached the destination for generating the route?

c) A*-Search

Modern navigation systems use variants of the A* algorithm for routing from a start location to a destination location. A* is an informed search algorithm that guarantees to find the optimal route if using an A* Heuristic.

Implement a function that uses A* for searching the shortest path between two locations in the given network. For that aim, have a look at the pseudo code given in the lectures for making sure that you do not miss parts of the algorithms. Make sure that you define an appropriate, admissible heuristic function for searching the given network.

How many nodes have been tested until you reached the destination for generating the route?

Task 2. Constraint Problem Solving

Recommended Deadline: Dec. 15th 2016

For solving constraint satisfaction problems, we learnt backtracking search in the lecture. There is a variety of constraint satisfaction problems with high practical relevance in reality beyond cryptoarithmetic puzzles or Sudoku – all kinds of configuration or scheduling problems can be seen as constraint satisfaction or constraint optimization problems. In this task, you shall solve a Cross Word Puzzle as a **constraint satisfaction problem**.

Consider the following crossword puzzle:

A1, D1	D2	D3
A2		
A3		

Word List:

ADD, ADO, AGE, AGO, AID, AIL, AIM, AIR, AND, ANY, APE, APT, ARC, ARE, ARK, ARM, ART, ASH, ASK, AUK, AWE, AWL, AYE, BAD, BAG, BAN, BAT, BEE, BOA, EAR, EEL, EFT, FAR, FAT, FIT, LEE, OAF, RAT, TAR, TIE

You must find six three-letter words: three words read across (A1, A2, A3) and three words read down (D1, D2, D3). Each word must be chosen from the given word list and can only appear once in the puzzle.

a) Prepare data structures for variables and domains as well as methods for testing constraints, testing whether specific values to two variables would fit together.
[There are two different ways of defining variables/domains in this problem: Each cell could be a variable with letters as domains or the word-entries with complete words as variables and domains. My suggestion is to have each word entry (A1,…, D3) as a variable and entries of the word list as possible domains.]

b) Write a method that generates a crossword puzzle by **backtracking search**. After each assignment, do a simple constraint propagation. That means delete values from the domains of the remaining unassigned variables that don't make sense anymore with the assignment.
c) Write a method **arc-consistency** as given in the lecture on Constraint Satisfaction that tests whether the status of variables and their remaining domains is still solvable. Use this method in a function that makes the complete crossword puzzle problem arc consistent by applying the method to each relevant combination of variables.
d) Integrate the method that makes the crossword problem arc-consistent into the backtracking algorithm of task 2b and use it for solving the crossword problem. How many assignments did you need to do after making the network arc-consistent at the beginning?
Print the results in a readable way.

Task 3 Clustering for Data Analysis

Recommended Deadline: Jan.5th 2017

We came across a number of machine learning or data mining methods that may help to learn and analyze big data sets. Clustering methods for example can help you to find out whether the classification suggested for your data set has anything to do with your data. If the clusters that the automated clustering generated is significantly different from a classification you thought of, this indicates that factors played a role when determining a classification that are not (fully) supported by the data but may be based on some information.

In this last task, you shall implement **the k-means clustering algorithm** as given in the lecture for clustering wholesale customer data. This assignment description is accompanied by a "customer.data" file that contains a description of clients of a wholesale distributor in Portugal[1].

Each of the 440 data points corresponds to one client. The following attributes are given for each of the data points:

1) FRESH: annual spending on fresh products (Integer Number);
2) MILK: annual spending on milk products (Integer Number);
3) GROCERY: annual spending on grocery products (Integer Number);
4) FROZEN: annual spending on frozen products (Integer Number)
5) DETERGENTS_PAPER: annual spending on detergents and paper products (Integer Number)
6) DELICATESSEN: annual spending on and delicatessen products (Integer Number);
7) CHANNEL: customers' Channel: 1=HoReCa (Hotel/Restaurant/Café) or 2=Retail channel (Nominal)
8) REGION: customers' Region: 1=Lisbon, 2=Oporto or 3=Other (Nominal)

[1] The data set has been taken from the UCI Machine Learning Data Repository (http://archive.ics.uci.edu/ml/datasets/Wholesale+costomers#).

The first line of the data set contains the attribute names. All columns are separated by “;”.Load the data set into an appropriate data structure of cases/data points. Of course, you may reuse some of your code from task 1. It is always good to have a look into the file, before starting to program.

After loading the data set of 440 clients, run a **k-means analysis** with different k-values. All values are all numeric: for the “continuous” values, you can use any distance metric for numerical values (see for example Minkowski distance in the lecture). Think also about standardizing the distance values as the ranges of values are quite different. For the nominal values, a direct comparison is advisable. Your cluster method shall get one of the attributes as argument. This attribute shall be omitted from similarity calculations and later analyzed in the output of the clustering. For each of the k clusters give the numbers of cases with a particular value (or a value in a particular interval) for the special attribute together with the centroid of the cluster.

Answer the following questions (during demonstration/in your report): What did you learn about the data? Does the outcome change when you re-run the algorithm a couple of times?