AI Course Fall HT 2015: Lab Assignments

- All assignments have to be implemented in **Python**. It is recommended to use a Python version that has some strong development environment. **There will be a special lecture on Nov 10th at 17:15 (during the "Övning" time) introducing concepts in Python you will need for implementing the labs**.
- Every assignment needs to be demonstrated to the teacher for showing that you have understood what you did and did it yourself. There will be no examination beyond that you can explain what you did and running your implementation potentially using an alternative data set.
- After demonstrating, you should write 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 (using UML Class Diagrams if appropriate)
    - Overall processes description (for example using UML Activity Diagrams or Sequence Diagrams)

  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; it must be accompanied with text describing why you did it that way. A report can be pretty short (2-3 pages). Formatting should be reasonable, yet there are no special requirements on formatting. The reports must contain information about who did the labs and information about **all** material that you used.
- You may work in teams of **2**
- 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.
- Every assignment comes with recommended date when you should have finished it. This is some indication for you how the lab work is synchronized with the lecture advance. You should be familiar with the lecture slides/contents that 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.
- The overall last date for demonstration is last lab time on **January 7th**. With special appointment, it will be possible to demonstrate your solutions until January, 28th. Further delayed demos and reports can be demonstrated and handed in within a time window of one week before to 2 weeks **at the re-exams**. 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 available at lab times and at any other time 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

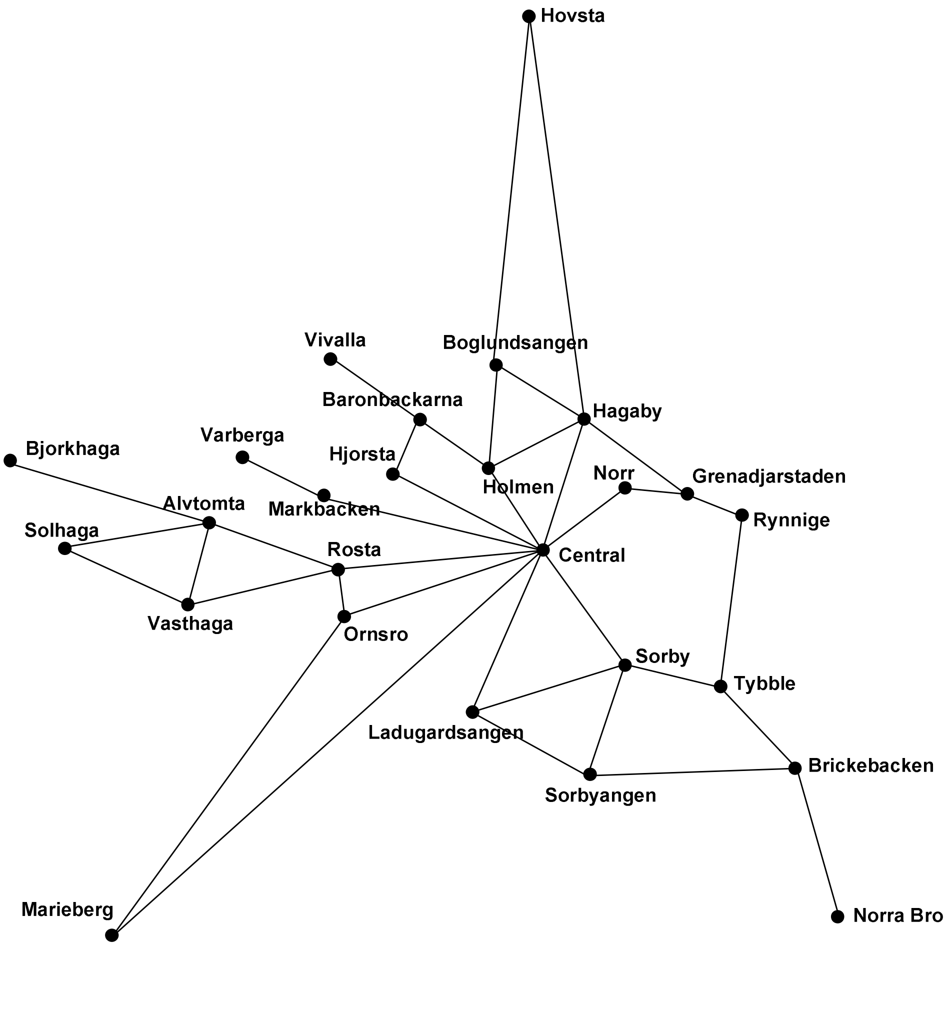
Recommended Deadline: Nov. 26th 2015

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**. Don't try to adapt algorithms found on the Internet.

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 following subtasks 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 ("locations2015.csv") and a file with links between locations ("connections2015.csv"). The file contains a roadmap such as the one depicted here:



This is not a correct map of Örebro and its suburbs, but a slightly manipulated one for being able to show how different algorithms give different results.

The content of the “locations2015.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” connections2015.csv” file stores information about reachability:

```
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. There is no visualization needed.

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

b) Iterative Deepening Search

In the lecture, several search methods have been given. Here your task is to implement the “best” uninformed search: iterative deepening search. Provide a simple user interface that allows the user to enter the names of start and destination location, calculate the path 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 current position to a destination. A* is an informed search algorithm that garantees to find the optimal route if using an A* Heuristic.

Implement a function that uses A* for searching the shortest path between two locations. For that aim, first define an appropriate heuristic function. If your heuristic from subtask b) is admissible for A*, then use it.

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

Task 2. Constraint Problem Solving

Recommended Deadline: Dec. 11th 2014

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 constraint satisfaction problem in two different ways. The problem that you shall solve is a room assignment problem. Given is a set of room (e.g. seminar rooms) with different characteristics. Each course from a set of courses with particular needs must be assigned to a suitable room:

Rooms = {T002, T004, T101, T103, T1148, T127, T133, T203, T221, T227}

With the following characteristics

Room	Places	Projector	Computer	Special Equipment	Level
T002	20	No	Yes	No	0
T004	24	No	Yes	No	0
T101	70	Yes	No	No	1
T103	20	No	No	No	1
T114B	10	No	No	Yes	1
T127	30	Yes	No	No	1
T133	40	Yes	No	No	1
T203	10	No	No	No	2
T221	20	Yes	No	No	2
T227	70	Yes	No	No	2

Your task is to distribute the courses that are given at the same time so to the rooms that a set of given constraints hold:

Course	Program	#Students	Type
A	Data	9	Lecture
B	Sociology	9	Seminar
C	Psychology	65	Lecture
D	Data	20	Labs
E	Mathematics	25	Lecture
F	Criminology	20	Seminar
G	Machine Engineering	15	Labs
H	Machine Engineering	35	Lecture
I	Data	44	Lecture
J	Machine Engineering	27	Lecture

a) Generate appropriate data structures representing this room assignment problem as a constraint satisfaction problem. Hereby separate static and dynamic information. That means, create a data structure in which you store information about rooms and courses that does not change in one data structure (i.e. a collection of objects) and create a second one (e.g. a dictionary) that contains the current assignment.

b) Implement a function

```
        allocate (persons, rooms)
```

that returns an assignment of courses to suitable rooms. The assignment needs to satisfy all the constraints given above.

c) There are alternative algorithms that also work not just for constraint satisfaction problems like this problem, but also for constraint **optimization** problems that can be much more complex. **Your next task is to implement the local search technique "Simulated Annealing"** for using heuristic optimization solving this problem.

- First create a function for checking constraints that instead of returning a boolean for an assignment that is not correct, returns *the number of violated constraints*. A solution is found, when this function returns 0 or an assignment. Test your function with randomly filled assignments.
  This function for calculating the number of violated constraints serves as a fitness function for the simulated annealing algorithm.
- The second element that is missing for the local search technique is the mutation operation generating a new assignment from an old one. For constraint optimization problems often a 2-swap or a 3-swap is a good choice: Two or three rooms exchange their assigned courses. Implement such a mutation algorithm operating on the data structure containing the assignment that with some (low) probability does a 3-swap, otherwise does a 2-swap and returns a new assignment.
- Integrate these elements into an implementation of simulated annealing. The pseudo code can be found on the lecture slides. Start with a random assignment and return if you found the best assignment or after the algorithm has ended.

Comparing the two approaches: What solution do you prefer to solve this kind of problem?

Task 3 Clustering for Data Analysis

Recommended Deadline: Jan.8th 2015

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 the provided one, 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 wines. This assignment description is accompanied by a "wine.data" file that contains the description of wines generated from three different cultivars in some area in Italy[1]. The data contains also a classification into three classes that represent those different cultivars.

All data points are the result of chemical analysis of the different wines. The first attribute is the numeric id of the cultivar that was the basis for this particular wine.

The other attributes are:

2) Alcohol
3) Malic acid
4) Ash
5) Alcalinity of ash
6) Magnesium
7) Total phenols
8) Flavanoids
9) Nonflavanoid phenols
10) Proanthocyanins
11) Color intensity
12) Hue
13) OD280/OD315 of diluted wines
14) Proline

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.

After loading the data set of 178 wines, run a **k-means analysis** with k=3. All values are all numeric, so you can use any distance metric for numerical values (see for example Minkowski distance in the lecture). **For the clustering ignore the cultivar attribute**. After the algorithm ends, three clusters have been generated. The algorithm shall print out:

- The centroid of each of the cluster,
- The number of cultivar entries in each of the three cluster (This indicates whether there is a relation between cultivar and chemical properties of the wine or maybe producers did adulterate/re-label?).
- The average distance of all entries of a cluster to its centroid and the distance between the centroids.

Ideally only wines from the same cultivar are contained in one cluster – can you confirm this? Are there outliers? Does the outcome change when you re-run the algorithm a couple of times? If yes, wrap the algorithm into a loop and output the average values of a number of runs (e.g. n= 25).

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