# IAI - Exercise Sheet 8

*This week we have a set of nine questions about search. They should be read in conjunction with your lecture notes and handouts for Week 8.*

## Question 1

Explain carefully the distinction between a *state space graph* and a *search tree*. Discus the potential trade-off between short solutions and good solutions, and between computational costs and other costs.

## Question 2

*Depth First Iterative Deepening Search* (DFIDS) is a good search algorithm, but it appears to be wasteful because many states are expanded many times. Is it possible to prevent that waste? If so, is it always worth the effort?

## Question 3

What does it mean to say that a search algorithm is *uninformed*? What kinds of information could be useful for an *informed* search algorithm?

## Question 4

Your kitchen has become a health hazard and you want to build an AI agent that will clean it for you. You have got yourself a robot and now you need to start programming it.

a) Define an appropriate set of operators/state transitions for cleaning the kitchen. Take into account such considerations as:

- Cleaning the oven or refrigerator will get the floor dirty
- To clean the oven, it is necessary to apply and then remove oven cleaner
- Before the floor can be washed, it must be swept
- Before the floor can be swept, the rubbish must be taken out
- Cleaning the refrigerator generates rubbish and messes up the work surfaces
- Washing the work surfaces or floor gets the sink dirty

b) Write a description of a likely initial state of a kitchen in need of cleaning. Also, write a description of a desirable goal state.

c) Develop a state space graph for the various states you kitchen might find itself in.

d) Show how you can search for an efficient plan (i.e. sequence of transitions) for reaching the goal state starting from the initial state.

e) Discuss the simplifications made in the scenario.

f) Next time your kitchen gets dirty, see how well your plan works for you.

## Question 5

This is a version of the famous *water jug puzzle*, which can be solved systematically by casting it into the form of a search problem.

You are given a three pint jug named *Three*, and a four pint jug named *Four*. You have no other measuring devices. Either jug can be filled with water from a tap named *Tap*, and water can be discarded from either jug down a drain called *Drain*. Water may also be poured from either jug into the other.

Initially both *Three* and *Four* are empty. Your goal is to find a sequence of operations that will leave you with exactly two pints of water in *Four*.

Set up a state space search formulation of this puzzle, and thus solve the puzzle.

## Question 6

You are on the bank of a river with a boat, a cabbage, a goat, and a wolf, and your task is to get everything to the other side. Only you can handle the boat, and there is only space in the boat for you and one additional item. You can't leave the goat alone with the wolf or the cabbage, or something will get eaten. Use a state space search formulation to solve your problem.

[Hint for questions 5 and 6: You will need to start by defining an appropriate way to represent the states and the transitions between them. You can then define your initial and goal states, and begin to draw up a search tree starting from the initial state. Finally, you will have to determine an efficient search strategy for finding your goal state.]

## Question 7

With *Hill Climbing* or *Gradient Descent* algorithms, is it possible to end up stuck in a non-optimal state? If so, what strategies could we employ to avoid that happening? Is it possible to end up in a final state and not know if better solutions are possible? If so, is there anything we can do to improve our chances of finding an optimal solution?

## Question 8

Consider again the *Romanian route finding problem* from Lecture 8. We saw that *A* search* was better than a *simple greedy best-first search*. Explain how and why it was better. Are there any general limitations to *A* search*? Can you formulate some heuristics that work even better for the route finding problem? [Hint: See Russell & Norvig, Sect 4.1]

## Question 9

Think again about the famous *8-tile puzzle*. Try to formulate some suitable informed search heuristics for that problem. [Hint: See Russell & Norvig, Sect 4.2]