Search and Planning

Alameda Campus

IST @ 2018/2019

October 5, 2018

1. Introduction

The goal of this assignment is to model, in Common Lisp, one of the following problems as a search problem using the code available on the course site in the file ‘*procura.lisp*’.

You must write a function `solve-problem` that accepts as arguments the initial state and the search strategy to be used. When called, this function creates the problem with the given initial state and tries to solve it using the given search strategy.

The argument search strategy must be a string with a name of one of the strategies supported by the procedure *procura* of the supplied code.

The produced program will be tested using the search strategies “profundidade” (depth-first) and “a*”.

2. Proposed problems

The first proposed problem must be solved by students whose group number "Individual" is odd and the second proposed problem must be solved by students whose group number "Individual" is even.

2.1. 20-queens problem

The 20-queens problem is identical to the 8-queens problems presented in class, but where the board is a 20x20 board where 20 queens must be placed without attacking each other.

The representation of the initial state that is supplied on the call of `solve-problem` is 20x20 array where all positions are set to NIL, to represent that they are not occupied by a queen. A position being occupied is set to T. This representation may be converted to a different internal representation if you so wish.[1]

The returned value in case of failure must be `NIL` and in case of success must be the goal state that was reached, written in the representation described above (note that to return the value using the correct representation it is always necessary to transform the result returned by the procedure *procura* of the supplied code).

The calls with which the code will be tested are at least the following:

```
(solve-problem (make-array '(20 20)) "profundidade")

(solve-problem (make-array '(20 20)) "a*")
```

2.2. 15-puzzle problem

The 15-puzzle problem is identical to the 8-puzzle problem presented in class, but with a 4x4 board with 15 tiles instead of a 3x3 board with 8 tiles.

The representation of the initial state that is supplied on the call of `solve-problem` is a 4x4 array where positions are set to an integer from 1 to 15 representing a tile or to NIL representing the empty cell. This representation may be converted to a different internal representation if you so wish.

The returned value, in case of failure must be `NIL` and in case of success must be a list

[1] In the 20 queens problem the initial state will always be an empty board.

with the solution reached. That is, the sequence of states from the initial state up to the goal state (note that to return the result using the correct representation it is always necessary to transform the result returned by the procedure *procura*). The representation for states must be the one described above.

The calls with which the code will be tested are at least the following:

```
(solve-problem (make-array '(4 4)
                  :initial-contents '((1 2 3 4)
                                      (5 6 7 8)
                                      (13 9 10 11)
                                      (14 nil 15 12)))
                  "profundidade")
(solve-problem (make-array '(4 4)
                   :initial-contents '((1 2 3 4)
                                       (5 6 7 8)
                                       (13 9 10 11)
                                       (14 nil 15 12))) "a*")
```

The goal state for the 15-puzzle is:

```
(make-array '(4 4)
            :initial-contents '((1 2 3 4)
                                (5 6 7 8)
                                (9 10 11 12)
                                (13 14 15 nil)))
```

3. Evaluation criteria

The most important factors for the evaluation of this assignment are the following:

- Adequate modelling and correct implementation;
- Correct execution.

Other factors also taken in account are:

- Efficiency of the data structures chosen for the problem representation;
- Quality of the developed heuristic.

4. Registering and delivery of the assignment

The assignment must be developed individually and you must register in Fénix under the group “Individual”.

The delivery must occur up to 23h59 of October 26, 2018 in the following way:

- a file with the automatic player must be delivered by electronic means using the Fénix system in a single Lisp file. The name of the Lisp file must be "GXXX.lisp", where "XXX" is the group number, eg. "G007.lisp".

5. News about the assignment

In case of news about the assignment, these will be published in the course site, so this site must be visited by you daily.