German University in Cairo
Department of Computer Science
Dr. Haythem O. Ismail

Introduction to Artificial Intelligence, Winter Term 2013
Project 1: Where Are My Parts?

Due: November 7 at 23:59

1. **Project Description** A robot inhabits a rectangular grid of squares. The grid is surrounded by a barbed wire fence. The robot's body consists of multiple identical hardware units, each occupying a single square of the grid. The units do not necessarily occupy contiguous squares—the robot's body parts are distributed all over the grid. The robot is controlled by a single station outside the grid. If two robotic parts occupy contiguous squares, then they are assembled into a single, larger part.

   Each of the robotic parts can move as per instructions issued by the controller. The controller issues instructions of the form $\langle p, d \rangle$, where p is a robotic part and $d \in \{North, South, East, West\}$, indicating that part p is to move in direction d. The motion proceeds continuously in the indicated direction and ends in one of the following ways.

   a) p collides with another robotic part. In this case, motion stops and the two parts assemble into a single new part.
   b) p collides with an obstacle. In this case, motion simply stops.
   c) p collides with the barbed wire fence. such collision causes major damage to p's hardware, resulting in failure.

   In this project, you will create part of the robot's controller which is designed to create a plan for the robotic parts to assemble into a single part, if possible. The plan will be formulated using search. Several algorithms will be implemented and each will be used to play the game:

   a) Breadth-first search.
   b) Depth-first search.
   c) Iterative deepening search.
   d) Greedy search with at least two heuristics.
   e) A* search with at least two admissible heuristics. The cost of the movement of a part p is given by $\sum_{u \in U(p)} s(u)$, where $U(p)$ is the set of units in p and $s(u)$ is the number of squares which unit u moves across.

Different solutions should be compared in terms of the number of search tree nodes expanded.

You may use the programming language of your choice.

Your implementation should have two main functions `GenGrid` and `Search`:

- `GenGrid()` randomly generates a grid. The dimensions of the grid, the locations of the robotic units, and the locations of obstacles are all randomly generated.
- `Search(`*grid*`,`*strategy*`,` *visualize*`)` uses search to try to assemble the robotic units into a single part:
  - *grid* is a grid to perform the search on,
  - *strategy* is a symbol indicating the search strategy to be applied:
    - `BF` for breadth-first search,
    - `DF` for depth-first search,
    - `ID` for iterative deepening search,
    - `GR`i for greedy search, with $i \in \{1, 2\}$ distinguishing the two heuristics, and
    - `AS`i for A* search, with $i \in \{1, 2\}$ distinguishing the two heuristics.
  - *visualize* is a Boolean parameter which, when set to `t`, results in your program's side-effecting a visual presentation of the board as it undergoes the different steps of the discovered solution (if one was discovered).

  The function returns a list of three elements, in the following order: (i) a representation of the sequence of moves to reach the goal (if possible), (ii) the cost of the solution computed, and (iii) the number of nodes chosen for expansion during the search.

2. **Groups:** You may work in groups of at most three.

3. **Deliverables**

   a) Source Code

   - You should implement an abstract data type for a search-tree node as presented in class.
   - You should implement an abstract data type for a generic search problem, as presented in class.
   - You should implement the generic search algorithm presented in class, taking a problem and a search strategy as inputs.
   - You should implement a where-are-my-parts instance/subclass of the generic search problem.
   - You should implement all of the search strategies indicated above (together with the required heuristics).
   - No assumptions should be made about the initial state.
   - Your program should implement the specifications indicated above.

- Part of the grade will be on how readable your code is. Use explanatory comments whenever possible

b) Project Report, including the following.

- A brief discussion of the problem.
- A discussion of your implementation of the search-tree node ADT.
- A discussion of your implementation of the search problem ADT.
- A discussion of your implementation of the where-are-my-parts problem ADT.
- A description of the main functions you implemented.
- A discussion of how you implemented the various search algorithms.
- A discussion of the heuristic functions you employed and, in the case of A*, an argument for their admissibility.
- A comparison of the performance of the different algorithms implemented.
- Proper citation of any sources you might have consulted in the course of completing the project.
- If you use code available in library or internet references, make sure you *fully* explain how the code works and be ready for an oral discussion of your work.
- If your program does not run, your report should include a discussion of what you think the problem is and any suggestions you might have for solving it.

4. Important Dates

Source code. On-line submission by November 7 at 23:59 using the "Submission" link on the course web site.

Project Report. A hard-copy should be submitted. You have two options:

a) November 7, by 16:00

b) November 9, by 16:00 provided that an *identical* on-line version is submitted with the code (by November 7 at 23:59).

Brainstorming Session. In tutorials.