The University of Texas at Dallas
CS 6364
Artificial Intelligence
Fall 2020
Instructor: Dr. Sanda Harabagiu
Grader/ Teaching Assistant: Maxwell Weinzierl

Homework 1: 100 points (70 points extra-credit)
Issued September 2, 2020
Due September 21, 2020 before midnight

PROBLEM 1: Intelligent Agents (**15 points**)

Enhance the Kitty robot discussed in class to a model-based agent in which the state has 3 components: [position happiness, battery]; where the battery has a value of 100 for FULL initially and 0 when it is discharged. Also add an admissible new action: TURN-AROUND. The possible positions of Kitty are:

You are asked to:

1. Describe the entire state space (**2 points**)
2. Write a new State-Update function that changes the value of the battery by -1 for every new change of position by walking; by -4 when it turns around; by -5 when it bumps against the wall; by -3 every time Kitty purrs and by -2 every time it meows You should consider that when the battery is discharged, no more actions are possible. (**10 points**)
3. Write a utility function that combines the happiness with the state of the battery. (**3 points**)

PROBLEM 2: Problem Formulation for Search (**55 points**)

The Missionaries and Cannibals Problem is stated as follows. Three missionaries and three cannibals are on one side of the river, along with a boat that can hold one or two people.

Find a way to get everyone to the other side without ever leaving a group of missionaries outnumbered by cannibals in that place.

a. Formulate the search problem formally. State clearly how you represent states, (including the initial state and the goal state(s)), what actions are available in each state, including the cost of each action. (**15 points**)
b. Sketch a solution of the problem. Show the path and its cost. Describe the search strategy that you have used and motivate it. (**15 points** for the manual solution; **5 points** for its motivation). **TOTAL 20 points**

Programming Assignment for Problem 2:

1. Use the implementations of the search strategies available in the aima code; e.g. https://github.com/aimacode/aima-java to write the program that will search for the solution to Missionaries and Cannibals Problem. (**10 points**)

2. Provide the path to the solution generated by your code, indicating each state it has visited when using: (a) uniform-cost search; (b) iterative deepening search; (c) greedy best-first search; (d) A* search and (e) recursive best-first search (**1 point/strategy)**. Describe clearly how you have implemented each state in the search space. (**5 points)**

TOTAL: 10 points

Extra-credit: List the content of the frontier and the list of explored nodes for the first 5 steps of each of the strategies used in 2. (**2 points/step/strategy)**

TOTAL: 50 points ONLY if it is possible to comprehend (1) the current node; (2) the content of the frontier; (3) the content of the list of explored/expanded nodes; (4) the children of the current node for each step.

PROBLEM 3: Searching for Road Trips in the U.S.A. (**30 points**)

Considering the following table:

Table of direct/flight distances to Dallas (in miles).

Austin	182
Charlotte	929
San Francisco	1230
Los Angeles	1100
New York	1368
Chicago	800
Seattle	1670
Santa Fe	560
Bakersville	1282
Boston	1551

1/ You contemplate to search for a trip back to Dallas from Seattle, having access to a road map which should be implemented as a graph. Use the aima code to find the solution and return a simulation of the RBFS strategy by generating automatically the following five

values: (1*) f_limit*; (2) *best*; (3) *alternative*; (4) *current-city*; and (5) *next-city* for each node visited. (**10 points)**

2/ Find the same solution manually and specify how you have computed the following five values: (1*) f_limit*; (2) *best*; (3) *alternative*; (4) *current-city*; and (5) *next-city* for each node visited. Specify at each step the current node and the next node. (**5 points)**

```
The road graph is (in miles):
_____________________________________
Los Angeles --- San Francisco ::: 383
Los Angeles --- Austin        ::: 1377
Los Angeles --- Bakersville   ::: 153
San Francisco --- Bakersville  ::: 283
San Francisco --- Seattle  ::: 807
Seattle  --- Santa Fe  ::: 1463
Seattle --- Chicago  :::  2064
Bakersville -- Santa Fe ::: 864
Austin --- Dallas ::: 195
Santa Fe --- Dallas ::: 640
Boston --- Austin  ::: 1963
Dallas --- New York  ::: 1548
Austin --- Charlotte  ::: 1200
Charlotte -- New York ::: 634
New York --- Boston ::: 225
Boston --- Chicago ::: 983
Chicago -- Santa Fe ::: 1272
Boston --- San Francisco ::: 3095
```

3/ Perform the same search for your optimal road trip from Seattle to Dallas when using A*. List the contents of the frontier and explored list for each node that you visit during search, in the format:

[Current node: X; Evaluation function (current node)=???;
Explored Cities: [….]; Frontier: [(City_X, f(City_X)), ….];] (**10 points)**

4/ Find the same solution manually, specifying:

(a) the current node; If it is not a goal node then also:
(b) the children of the current node and the value of their evaluated function (detailing how you have computed it)
(c) The current path from Seattle to the current city + the cost
(d) The Contents of the Explored Cities list
(e) The Contents of the Frontier
(f) Next node.

(5 points)

Extra-credit Write a program that will check if the heuristic provided in this problem is consistent, given the road graph **TOTAL: 20 points**

Software Engineering (includes documentation for your programming assignments)

Your README file must include the following:

- Your name and email address.
- *Homework number* for this class (AI CS6364), and the *number of the problem* it solves.
- A description of every file for your solution, the programming language used, supporting files from the aima code used, etc.
- How your code operates, in detail.
- A description of special features (or limitations) of your code.

Within Code Documentation:

- Methods/functions/procedures should be documented in a meaningful way. This can mean expressive function/variable names as well as explicit documentation.
- Informative method/procedure/function/variable names.
- Efficient implementation
- Don't hardcode variable values, etc