The University of Texas at Dallas
CS 6364
Artificial Intelligence
Fall 2020

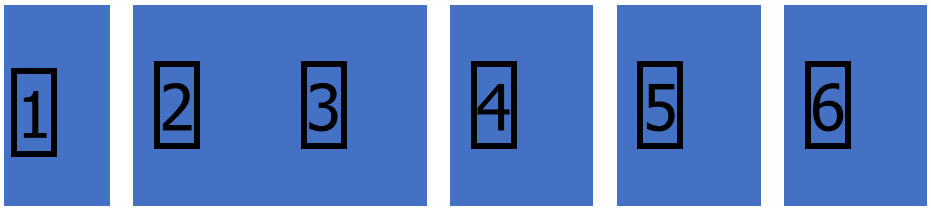
Pratik Rajendra Deshpande NetId: prd190001

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: TURNAROUND. The possible positions of Kitty are:



You are asked to:

1. Describe the entire state space (**2 points**)

The state can be a combination of the position of the kitty, the happiness of the kitty, the battery health (0-100) of the kitty. As we know position can be 1-6. Let's assume happiness can be **Loved > Excited > Happy > Sad > Angry > Sleepy.** At any point of time the state can be a tuple (battery,happiness,position) having different values and combinations. This is a factored representation and the state can have a combination of values from the values above = 100 * 6 * 6

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**)

Percepts: Clap, Pet, Bump
Actions: TurnAround,Walk,Meow,Purr,Blink,Stop
Initial State: 6,Sleepy,100

State Update:

If Battery < 20:

Sleep

Else:

state ←UPDATE-STATE(previous-state, percept, action)

batteryHealth – 1 <- UPDATE-KITTY[batteryHealth,walk]

batteryHealth – 5 <- UPDATE-KITTY[batteryHealth,Bumps]

batteryHealth – 3 <- UPDATE-KITTY[batteryHealth,Purr]

batteryHealth – 2 <- UPDATE-KITTY[batteryHealth,Meow]

timeElapsedSinceLastPercept <- RESET-TIME[timeElapsedSinceLastPercept, percept] currentLocation + 1 ←UPDATE-KITTY[Position,Walk]

if direction == Forward

direction<-UPDATE-KITTY[Backward, TurnAround]

else:

direction<-UPDATE-KITTY[Forward, TurnAround]

currentLocation ← UPDATE-KITTY[Position,Purr|Meow|Blink|Pounce]

3. Write a utility function that combines the happiness with the state of the battery. (**3 points**)

Function Calculate Utility:

Happiness +1 ←UPDATE-KITTY[Happines,Clap]
Happiness -2 ←UPDATE-KITTY[Happines,BatteryHealth.50]
Sleep <- UPDATE-KITTY[Happiness, BatteryHealth.20]
Angry ←UPDATE-KITTY[Happiness,Bump]

The utility function can have values like above wherein the happiness of the kitty can change it between the range as per the percepts, actions and also how the internal battery health of the kitty is.

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**)

The representation of the problem is done as a list of missionaries, cannibals or boat on the wrong side of the river, that is the left side. Whenever the boat is on the right side, using the previous state, the next action selected will be added to the state to generate the next state.

Initial State: (3,3,1)
Goal State (0,0,0)

Possible Actions:
(1,0,1) {can take one person and one boat)
(2,0,1) {can take two people and the boat}
(0,1,1) {can take one cannibal and the boat}
(0,2,1) {can take two cannibals and the boat}
(1,1,1) {can take one person and one cannibal with the boat}

The above actions will be subtracted from the initial state and that is the idea of formation of the search state space.

Path Cost: Every trip of the boat will increment the cost by 1. Which means all step costs are equivalent.

State Space:

State	Possible Actions	Resulting State
(3,3,1) -> (0,0,0)	(0,2,1), (1,1,1)	(3,1,0) (2,2,0)
(3,1,0) -> (0,2,1)	(0,1,1)	(3,2,1)
(2,2,0) -> (1,1,1)	(1,0,1) (0,1,1)	(3,2,1) (2,3,1 #invalid)
(3,2,1) -> (0,1,0)	(1,1,0) (0,2,2)	(2,1,0 #invalid) (3,0,0)
(3,0,0) -> (0,3,1)	(0,2,1) (0,1,1)	(3,2,1) (3,1,1)
(3,1,1) -> (0,2,0)	(2,0,1) (1,1,1)	(1,1,0) (2,0,0 #invalid)
(1,1,0) -> (2,2,1)	(1,1,1) (2,0,1) (0,2,1)	(2,2,1) (3,1,1) (1,3,0 #invalid)
(2,2,1) -> (1,1,0)	(0,2,1) (2,0,1) (1,1,1)	(2,0,0 #inv) (0,2,0) (1,1,0)
(0,2,0) -> (3,1,1)	(0,1,1) (1,1,1) (2,1,1)	(0,3,1) (1,3,1 #inv) (2,3,1 #inv)
(0,3,1) -> (3,0,0)	(0,2,1) (0,1,1)	(0,1,0) (0,2,0)
(0,1,0) -> (3,2,1)	(1,1,1) (2,0,1) (0,2,1)	(1,1,1) (2,1,1 #inv) (0,2,1)
(0,2,1) -> (3,1,0)	(0,1,1) (0,2,1)	(0,1,0) (0,0,0 #goal)

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**

The maximum branching factor is actually 8, but as we can see above as not all resulting states are valid, the maximum branching is of 3 for the initial and most of the time the branching factor is also 1. This means that the resulting tree/graph must be small. As we can see above, some states can be reached via multiple parents, and this can cause looping hence, a lot of graph search algorithms can be used and would be optimal, and I am choosing the algorithm of **Depth first graph search** to display the above problem.

Depth First Graph Search

To Simplify the above notation 33L is the initial state and 00R is the final state (MissionariesCannibalsBoat)

Current Node 33L. is (33L) goal ? NO! frontier: 31R, 22R, 32R closed: 33L

Current Node 31R. is (31R) goal ? NO! frontier: 32L, 22R, 32R closed: 31R, 33L

Current Node 32L. is (32L) goal ? NO! frontier: 30R, 22R, 32R closed: 32L, 31R, 33L

Current Node 30R. is (30R) goal ? NO! frontier: 31L, 22R, 32R closed: 30R, 32L, 31R, 33L

Current Node 31L. is (31L) goal ? NO! frontier: 11R, 30R, 22R, 32R closed: 31L, 30R, 32L, 31R, 33L

Current Node 11R. is (11R) goal ? NO! frontier: 22L, 30R, 22R, 32R
closed: 11R, 31L, 30R, 32L, 31R, 33L

Current Node 22L. is (22L) goal ? NO! frontier: 02R, 30R, 22R, 32R
closed: 22L, 11R, 31L, 30R, 32L, 31R, 33L

Current Node 02R. is (02R) goal ? NO! frontier: 03L, 30R, 22R, 32R
closed: 02R, 22L, 11R, 31L, 30R, 32L, 31R, 33L

Current Node 03L. is (03L) goal ? NO! frontier: 01R, 30R, 22R, 32R
closed: 03L, 02R, 22L, 11R, 31L, 30R, 32L, 31R, 33L

Current Node 01R. is (01R) goal ? NO! frontier: 11L, 02L, 30R, 22R, 32R
closed: 01R, 03L, 02R, 22L, 11R, 31L, 30R, 32L, 31R, 33L

Current Node 11L. is (11L) goal ? NO! frontier: 00R, 01R, 02L, 30R, 22R, 32R
closed: 11L, 01R, 03L, 02R, 22L, 11R, 31L, 30R, 32L, 31R, 33L

Current Node 00R. is (00R) goal ? YES! frontier: 01R, 02L, 30R, 22R, 32R
closed: 00R, 11L, 01R, 03L, 02R, 22L, 11R, 31L, 30R, 32L, 31R, 33L

Path : 33L-> 31R -> 32L -> 30R -> 31L -> 11R -> 22L -> 02R -> 03L -> 02R -> 10L -> 11L -> 00R

DFS found the solution in 12 steps.

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**)

Present in the zip file submitted, please read readme file.

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)**.

Uniform-cost-search:

(3, 3, 1)

(2, 2, 0)

(3, 2, 1)

(3, 0, 0)

(3, 1, 1)

(1, 1, 0)

(2, 2, 1)

(0, 2, 0)

(0, 3, 1)

(0, 1, 0)

(0, 2, 1)

(0, 0, 0)

Iterative deepening search:

(3, 3, 1)

(3, 1, 0)

(3, 2, 1)

(3, 0, 0)

(3, 1, 1)

(2, 0, 0)

(2, 2, 1)

(0, 2, 0)

(0, 3, 1)

(0, 1, 0)

(0, 2, 1)

(0, 0, 0)

Greedy best-first-graph-search:

(3, 3, 1)

(2, 2, 0)

(3, 2, 1)

(3, 0, 0)

(3, 1, 1)

(1, 1, 0)

(2, 2, 1)

(0, 2, 0)

(0, 3, 1)

(0, 1, 0)

(0, 2, 1)

(0, 0, 0)

A*:

(3, 3, 1)

(2, 2, 0)

(3, 2, 1)

(3, 0, 0)

(3, 1, 1)

(1, 1, 0)

(2, 2, 1)

(0, 2, 0)

(0, 3, 1)

(0, 1, 0)

(0, 2, 1)

(0, 0, 0)

RBFS:

(3, 3, 1)

(3, 1, 0)

(3, 2, 1)

(3, 0, 0)

(3, 1, 1)

(2, 0, 0)

(2, 2, 1)

(0, 2, 0)

(0, 3, 1)

(0, 1, 0)

(0, 2, 1)

(0, 0, 0)

Describe clearly how you have implemented each state in the search space. (**5 points)**

- Every Node Object has its instantiation of the state object, A state is a list with self.value = (missionaries, cannibals, boat) and is inititalized in the constructor.
- State-Class has an state.isValid() method which validates according to the constraints of the problem defined.
- The operation on the tuple depends upon the value of the boat, if the value of the boat, is 0 then it adds the action to the tuple and if the value of the boat is 1, then it subtracts from the tuple.
- Valid actions are generated at each state, and the children nodes are calculated, after computing what generates valid actions.
- The actions defined in the above table (manual part of the question) are used to operate on the state, and check whether invalid states are being generated.
- The heuristic used for informed search algorithm is : no of missionaries on initial side + no of cannibals on initial side – 1 as this is a good estimate and solves the relaxed problem.

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.

Uniform Cost Search:

Frontier: [(<<Node <State (3, 3, 1)>>)]

****

Explored: set([])

Frontier: [(<Node <State (2, 2, 0)>>), <Node <State (3, 2, 0)>>, <Node <State (3, 1, 0)>>)]

****

Explored: set([<State (3, 3, 1)>])

Frontier: [(<Node <State (3, 1, 0)>>, <Node <State (3, 2, 0)>>, <Node <State (3, 2, 1)>>]

****

Explored set([<State (3, 3, 1)>, <State (2, 2, 0)>])

Frontier: [<Node <State (3, 2, 0)>>), <Node <State (3, 2, 1)>>]

****

Explored: set([<State (3, 3, 1)>, <State (3, 1, 0)>, <State (2, 2, 0)>])

Frontier: [<Node <State (3, 2, 1)>>)]

****

Explored: set([<State (3, 2, 0)>, <State (3, 3, 1)>, <State (3, 1, 0)>, <State (2, 2, 0)>])

Greedy Best First Search:

[(5, <Node <State (3, 3, 1)>>)]

****

set([])

[(3, <Node <State (2, 2, 0)>>), (4, <Node <State (3, 2, 0)>>), (3, <Node <State (3, 1, 0)>>)]

****

set([<State (3, 3, 1)>])

[(3, <Node <State (3, 1, 0)>>), (4, <Node <State (3, 2, 0)>>), (4, <Node <State (3, 2, 1)>>)]

****

set([<State (3, 3, 1)>, <State (2, 2, 0)>])

[(4, <Node <State (3, 2, 0)>>), (4, <Node <State (3, 2, 1)>>)]

****

set([<State (3, 3, 1)>, <State (3, 1, 0)>, <State (2, 2, 0)>])

[(4, <Node <State (3, 2, 1)>>)]

****

set([<State (3, 2, 0)>, <State (3, 3, 1)>, <State (3, 1, 0)>, <State (2, 2, 0)>])

A* Search:

[(5, <Node <State (3, 3, 1)>>)]

****

set([])

[(4, <Node <State (2, 2, 0)>>), (5, <Node <State (3, 2, 0)>>), (4, <Node <State (3, 1, 0)>>)]

****

set([<State (3, 3, 1)>])

[(4, <Node <State (3, 1, 0)>>), (5, <Node <State (3, 2, 0)>>), (6, <Node <State (3, 2, 1)>>)]

****

set([<State (3, 3, 1)>, <State (2, 2, 0)>])

[(5, <Node <State (3, 2, 0)>>), (6, <Node <State (3, 2, 1)>>)]

****

set([<State (3, 3, 1)>, <State (3, 1, 0)>, <State (2, 2, 0)>])

[(6, <Node <State (3, 2, 1)>>)]

****

set([<State (3, 2, 0)>, <State (3, 3, 1)>, <State (3, 1, 0)>, <State (2, 2, 0)>])

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)
Step:1
Current Node: Seattle
Children: (Santa Fe, 2023), (San Francisco, 2037), (Chicago, 2864)
Current Path: Seattle
Cost: 1670
Frontier: Santa Fe:2023, San Francisco:2037, Chicago:2864
Explored: Seattle
Next Node: Santa Fe

Step:2
Current Node: Santa Fe
Children: (Dallas, 2103), (Chicago, 3535), (Bakersville, 3609), (Seattle, 4596)
Current Path: Seattle -> Santa Fe
Cost: 2023
Frontier: San Francisco:2037, Dallas:2103, Chicago: 2864, Bakersville:3609
Explored: Seattle, Santa Fe
Next Node: San Francisco

Step:3

Current Node: San Francisco
Children: (Los Angeles, 2290), (Bakersville, 2372), (Seattle, 3284), (Boston, 5453)
Current Path: Seattle -> San Francisco
Cost: 2037
Frontier: Dallas:2103, Los Angeles:2290, Bakersville:2372, Chicago:2864, Boston:5453
Explored: Seattle, Santa Fe, San Francisco
Next Node: Dallas

Step:4
Current Node: Dallas
We have found the goal

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:

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

Within Code Documentation:

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