Unity Id: sssaha2

Name: Sourabh Saha

Title: CSC520 Artificial Intelligence Homework Assignment 1

Q1 Agent for “Intelligent” buildings

a) PEAS specification:
   - i. Performance measure :
     - a) Maintain temperature between appropriate range
     - b) Constant monitoring of environment (keep track of people in rooms and keep track of current temperature)
     - c) Adhere to the 20 minute time limit to bring the room to the appropriate temperature
   - ii. Environment:
     - a) Workers/people in the room
     - b) The room itself
   - iii. Actuators:
     - a) Fans
     - b) Radiators
     - c) Lights
     - d) Air-conditioner
   - iv. Sensors:
     - a) Temperature sensors
     - b) Motion tracking sensors

b) It is sufficient for the agent to be simple reflex and the reasons are stated below:
   - i. The agent only needs to keep track of the current temperature in order to initiate the action rules of heating or cooling
   - ii. Also, the agent needs to just keep track of the current state of the room in order to initiate the action rules for turning the lights on/off

c) Randomization of heating/cooling actions would be detrimental to the agent's performance because of the following disadvantages:
   - i. It would lead to unnecessary energy wastage
   - ii. The room might become to hot/cold for the people who are about to enter

d) One improvement that I would suggest, would be to ensure that the agent turns off the heating/cooling systems when there is nobody present in the room (after the 2 minute interval). But the drawback of this improvement is that there would be increase in energy utilization in order to bring the room back to the optimal temperature range once people decide to enter it.

Q2 Watson: An IBM DeepQA project

a) PEAS specification:
   - i. Performance measure:
     - a) Maximum accuracy
     - b) Maximum percentage of questions answered
     - c) Maximize precision
     - d) Answer question before all other contestants
     - e) Maximize revenue generation by optimizing betting strategies
   - ii. Environment:
     - a) Contestants
     - b) Entire collection of local resources
   - iii. Actuator:
     - a) Buzzer
     - b) Text-to-speech Engine
     - c) Speakers
   - iv. Sensors:
     - a) Questions (text-based) which are given as input

b) The Jeopardy problem domain has several important aspects which the researchers building Watson handled through making some design choices. They are as stated below:
   - i. The questions asked in the show are of varying type. Ranging from facts to puzzles, each question required a separate approach in order to get the right answer. The decomposition questions require generation of zero or more decomposition hypotheses for each question as possible interpretations.
   - ii. Answering the question, as fast as possible, while simultaneously searching through huge databases required the researchers to implement a parallel architecture. Multiple estimations of confidence could be achieved simultaneously through this architecture

iii. The answers needed to be be accurate in order to acquire the most amount of revenue and in order to achieve this DeepQA employs more than 100 different techniques for analyzing natural language, identifying sources, scoring hypotheses and ranking them.

c)

i. DeepQA is massively parallel probabilistic evidence based architecture. The most important aspect of the architecture is how the information from the various algorithms are combined such that the overlapping can contribute to improvements in accuracy, confidence and speed. The methodology used in this approach is not specific to Jeopardy, rather is meant to adapt to different businesses, medicine, enterprise search, gaming etc. DeepQA is based on the following principles:

   a) Massive parallelism

   b) Many experts

   c) Pervasive confidence estimations

   d) Integrate shallow and deep knowledge

ii. The six architectural roles in the model are:

   a) Content acquisition

   b) Question Analysis

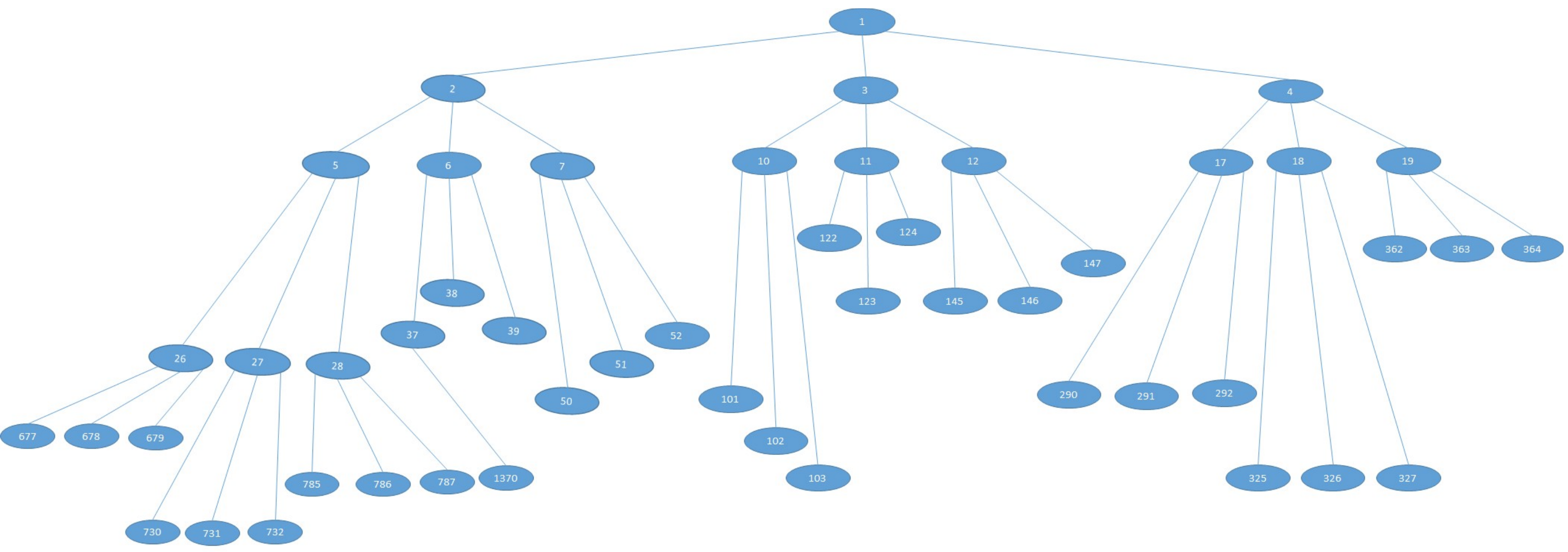
   c) Hypothesis generation

   d) Soft filtering

   e) Hypothesis and evidence scoring

   f) Final merging and ranking

Q3



(All DFS traversals consider top of stack as the right-most node)

a) The order of traversal is : 1,2,3,4,5,6,7,10,11,12,17,18,19,26,27,28,37,38,39,50,51,52,101
b) The order of traversal is : 1,4,19,364,363,362,18,290,291,292,3,12,147,146,145,11,124,123,122,10,103,102,101
c) The order of traversal is : 1,4,3,2,1,4,19,18,17,3,12,11,10,2,7,6,5,1,4,19,364,363,362,18,327,326,325,17,292,291,290,3,12,147,146,145,11,124,123,122,10,103,102,101

Q4

	Fully vs Partially Observable	Deterministic vs Stochastic	Episodic vs sequential	Static vs Dynamic	Discrete vs Continuous	MultiAgent vs Single Agent
Vacuum cleaner agent	Partially observable	Deterministic	Episodic	Dynamic	Discrete	Single
Google Car	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi

	observable					
Search and Rescue bots	Partially Observable	Stochastic	Sequential	Dynamic	Continuous	Multi
Document Categorizer	Fully	Deterministic	Episodic	Static	Discrete	Single
Watson	Fully	Stochastic	Sequential	Dynamic	Discrete	Multi

Q5

a)

i. Lugoj to Bucharest(One solution path):

   a) DFS : lugoj > timisoara > arad > sibiu > rimnicu_vilcea > pitesti > bucharest

   b) BFS : lugoj > mehadia > dobreta > craiova > pitesti > bucharest

ii. Bucharest to Lugoj(One solution path):

   a) DFS : bucharest > pitesti > rimnicu_vilcea > sibiu > arad > timisoara > lugoj

   b) BFS : bucharest > fagaras > sibiu > arad > timisoara > lugoj

iii. Lugoj to Bucharest (All visited nodes):

   a) DFS: lugoj mehadia timisoara arad sibiu zerind oradea fagaras rimnicu_vilcea craiova pitesti bucharest

   b) BFS: lugoj mehadia timisoara dobreta arad craiova sibiu zerind pitesti rimnicu_vilcea fagaras oradea bucharest

iv. Bucharest to Lugoj (All visited nodes):

   a) DFS: bucharest fagaras giurgiu pitesti urziceni hirsova vaslui iasi neamt eforie craiova rimnicu_vilcea sibiu arad oradea zerind timisoara lugoj

   b) BFS: bucharest fagaras giurgiu pitesti urziceni sibiu craiova rimnicu_vilcea hirsova vaslui arad oradea dobreta eforie iasi timisoara zerind mehadia neamt lugoj

v. As one can see that the number of visited nodes are more in BFS than in DFS, irrespective of which start node is chosen. This is because BFS traverses all possible solutions along a particular depth before moving deeper. This results in increase in the number of visited nodes as compared to DFS which traverses as deep as possible before moving on the sibling of the parent node.

b) Consider the case wherein we want to travel from Lugoj to Dobreta (Both my DFS and BFS algorithms follow the convention of sorting the successor node list in alphabetical order, and the case represented here is based on this convention)

i. One solution path of DFS: lugoj > timisoara > arad > sibiu > rimnicu_vilcea > craiova > dobreta

ii. One solution path of BFS: lugoj > mehadia > dobreta

iii. This shows that number of cities traversed by DFS algorithm is more than that of BFS algorithm. This is because since the DFS algorithm uses the stack data structure, the higher alphabetical order path gets followed, whereas the BFS algorithm uses the queue data structure and hence the lower alphabetical order route gets followed. This shows that BFS performs better than DFS when this particular case is considered.

c) Consider the case from the part a. of this question, where we want to travel from Lugoj to Bucharest. In this case, the solution provided by BFS contains more number of nodes than DFS and hence we can see that BFS has worse performance than DFS. This is because the number of nodes that the BFS algorithm traversed in order to find the appropriate path is much more than the number of nodes traversed by the DFS algorithm in this case.

d) Initial cutoff = 1;

Start_node = Fagaras;

Stack = [Fagaras];

Iteration 1:

Stack:[Fagaras

Stack:[Bucharest,Sibiu

Stack:[Bucharest

Stack:[

Nodes expanded: Fagaras, Sibiu, Bucharest

cutoff =2;

Iteration 2:

Stack:[Fagaras

Stack:[Bucharest,Sibiu

Stack:[Bucharest,Arad,Oradea,Rimnicu Vilcea

Stack:[Bucharest,Arad,Oradea

Stack:[Bucharest,Arad

Stack:[Bucharest

Stack:[Giurgiu, Pitesti, Urziceni

Stack:[Giurgiu, Pitesti

Stack:[Giurgiu

Stack:[

Nodes expanded: Fagaras, Sibiu, Rimnicu Vilcea, Oradea, Arad, Bucharest, Giurgiu, Pitesti, Urziceni

cutoff =3;

Iteration 3:

Stack:[Fagaras

Stack:[Bucharest,Sibiu

Stack:[Bucharest,Arad,Oradea,Rimnicu Vilcea

Stack:[Bucharest,Arad,Oradea,Craiova, Pitesti

Stack:[Bucharest,Arad,Oradea, Craiova

Stack:[Bucharest,Arad,Oradea

Stack:[Bucharest,Arad,Zerind

Stack:[Bucharest,Arad

Stack:[Bucharest,Timisoara

Stack:[Bucharest

Stack:[Giurgiu, Urziceni

Stack:[Giurgiu, Hirsova,Valsui

Stack:[Giurgiu, Hirsova]

Stack:[Giurgiu

Stack:[

Nodes expanded: Fagaras, Sibiu, Rimnicu Vilcea, Pitesti, Craiova, Oradea, Zerind, Arad, Timisoara, Bucharest, Giurgiu, Urziceni, Hirsova, Valsui

cutoff =4;

Iteration 4:

Stack:[Fagaras

Stack:[Bucharest,Sibiu

Stack:[Bucharest,Arad,Oradea,Rimnicu Vilcea

Stack:[Bucharest,Arad,Oradea,Craiova, Pitesti

Stack:[Bucharest,Arad,Oradea,Craiova

Stack:[Bucharest,Arad,Oradea,Dobreta

Stack:[Bucharest,Arad,Oradea

Stack:[Bucharest,Arad,Zerind

Stack:[Bucharest,Arad

Stack:[Bucharest,Timisoara

Stack:[Bucharest, Lugoj

Stack:[Bucharest

Stack:[Giurgiu, Urziceni]

Stack:[Giurgiu, Hirsova,Valsui

Stack:[Giurgiu, Hirsova, Iasi

Stack:[Giurgiu, Hirsova

Stack:[Giurgiu, Eforie

Stack:[Giurgiu

Stack:[

Nodes expanded: Fagaras, Sibiu, Rimnicu Vilcea, Pitesti, Craiova, Dobreta Oradea, Zerind, Arad, Timisoara, Lugoj, Bucharest, Urziceni, Valsui, Iasi, Hirsova, Eforie, Giurgiu

cutoff =5;

Iteration 5:

Stack:[Fagaras

Stack:[Bucharest,Sibiu

Stack:[Bucharest,Arad,Oradea,Rimnicu Vilcea

Stack:[Bucharest,Arad,Oradea,Craiova, Pitesti

Stack:[Bucharest,Arad,Oradea,Craiova

Stack:[Bucharest,Arad,Oradea,Dobreta

Stack:[Bucharest,Arad,Oradea,Mehadia

Stack:[Bucharest,Arad,Oradea

Stack:[Bucharest,Arad,Zerind

Stack:[Bucharest,Arad

Stack:[Bucharest,Timisoara

Stack:[Bucharest, Lugoj

Stack:[Bucharest

Stack:[Giurgiu, Urziceni

Stack:[Giurgiu, Hirsova,Valsui

Stack:[Giurgiu, Hirsova, Iasi

Stack:[Giurgiu, Hirsova, Nearnt

Stack:[Giurgiu, Hirsova

Stack:[Giurgiu, Eforie

Stack:[Giurgiu

Stack:[

Nodes expanded: Fagaras, Sibiu, Rimnicu Vilcea, Pitesti, Craiova, Dobreta, Mehadia, Oradea, Zerind, Arad, Timisoara, Lugoj, Bucharest, Urziceni, Valsui, Iasi, Nearnt Hirsova, Eforie, Giurgiu

Nodes expanded for BFS (Final iteration): Fagaras, Bucharest, Sibiu, Giurgiu, Pitesti, Urziceni, Arad, Oradea, Rimnicu Vilcea, Craiova, Hirsova, Valsui, Timisoara, Zerind, Dobreta, Eforie, Iasi, Lugoj, Mehadia, Nearnt

In the BFS expansion the number nodes expanded in every iteration may or may not be more than the number of nodes expanded in the previous iteration. But for Iterative Deepening DFS, the number of nodes have a non-decreasing order. Another difference in the two algorithms is that IDDFS follows the DFS methodology that the leaf nodes are visited first and then the interior nodes, but in the list of BFS, the last nodes expanded in every iteration are the leaf nodes.