**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS**

**Artificial Intelligence (BITS F444/ CS F407)**

**I Semester 2017-18**

**Programming Assignment-1**

**Coding Details**

**(September 14, 2017)**

*Instruction: Type the details precisely and neatly*

1. ID : 2015A7PS0127P

Name : Samip Jasani

1. Mention the names of Submitted files :
   1. dirtgenerator.py
   2. gui.py
   3. h1.py
   4. h2.py
   5. idfs.py
   6. bfs.py
   7. 2015A7PS0127P.docx
   8. driver.py
2. Total number of submitted files: 7
3. Name of the folder : 2015A7PS0127P
4. Have you checked that all the files you are submitting have your name in the top? yes
5. Have you checked that all the files you are submitting are in the folder as specified in 4 ? yes
6. Problem formulation
   1. State representation:

State contains position of agent as row and col with dirtlylist representing the list of dirty tiles in the board.

* 1. How is the Initial state generated?

According to percentage(p) entered by the user. Board with p% random tiles are selected as dirty. Initial State contains row,col of all four resting position in board with dirtylist containing the coordinates of dirty tile got from board.

* 1. What is the goal state?

Goal states contains coordinates of 4 resting states of the board with dirtylist empty.

* 1. Are there more than one goal states?

yes there are 4 goal states

* 1. If yes, then describe all the goal states

goal states contain row,col of corners in the board with dirtylist empty.

* 1. State representation in Python (name the construct and give one small example of a state)

state = mystate([(0,2),(5,3),(4,1)],2,3)

1. Successor function description

Each state first checks which actions are possible then generate child for each possible action. If any heuristics are mentioned then it uses it to add them to stack in order.

1. Uninformed Search Technique (T1) details
   1. Technique used for search: Iterative Deepening Search (IDS) and Breadth First Search (BFS)
   2. Reason for selecting this technique over the other two:

First I used BFS but due to memory limitation converted it to IDS

* 1. Is the search applied on tiles or on states? On States
  2. Error handling and reporting : yes
  3. List the errors handled:
     1. Checking for stack/queue size before popping
     2. Child nodes only generated if action possible
     3. Explored States maintained to prevent infinite loops (based on path cost for IDS)
  4. Data Structure description for the tree node (in maximum two lines):

Each node contains its state, parent node pointer, action from parent, depth, total post till the node

* 1. Code status (implemented fully/ partially/ not done)

Implemented Fully. Code optimized to speed it up without using any heuristics

1. Informed search Technique (T2) details:
   1. Technique used for search: Greedy best-first search and Greedy best-first using heap
   2. Reason for selecting this technique over others:

Because both of this technique gives priority of cleaning dirt over everything else

* 1. Does this technique look at a tile?

No

* 1. Does this technique use a state?

Yes

* 1. Code status (implemented fully/ partially/ not done)

Implemented fully.

* 1. Define the heuristics (in words) used in your program
     1. h1 : It is a DFS where next node for exploration is the one with maximum dirt in its percept. If the node contains dirt at its position +3 is added. If dirt on its any side (1 action difference) +2 is added for each and if along diagonal (2 action difference) +1 is added for each.
     2. h2 : It is a heap based search where score is given to each node based on current percept and it’s parents’s score. The node with best(minimum in our case) score is poped and expanded next.
  2. Compute the heuristic values for the following windows with dirt in the neighborhood of the centre position of the vacuum cleaner.
     1. h1 = 4 , h2(leftchild) = h2(parent) + 1, h2(rightchild) = h2(parent) - matrixsize +1,

h2(upchild) =h2(parent), h2(downchild) = h2(parent) +2

* + 1. h1 = 9 , h2(leftchild) = h2(parent) + 1, h2(rightchild) = h2(parent) - matrixsize ,

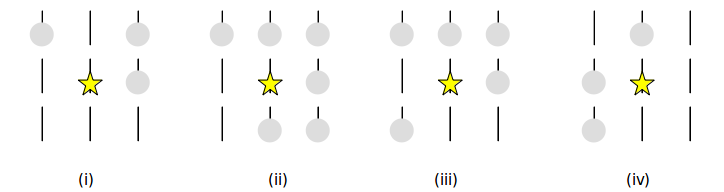
h2(upchild) =h2(parent) - matrixsize, h2(downchild) = h2(parent) - matrixsize +1

* + 1. h1 = 8 , h2(leftchild) = h2(parent) , h2(rightchild) = h2(parent) - matrixsize +1,

h2(upchild) =h2(parent) - matrixsize, h2(downchild) = h2(parent) +1

* + 1. h1 = 4 , h2(leftchild) = h2(parent)- matrixsize + 1, h2(rightchild) = h2(parent) +2,

h2(upchild) =h2(parent) - matrixsize, h2(downchild) = h2(parent) +1



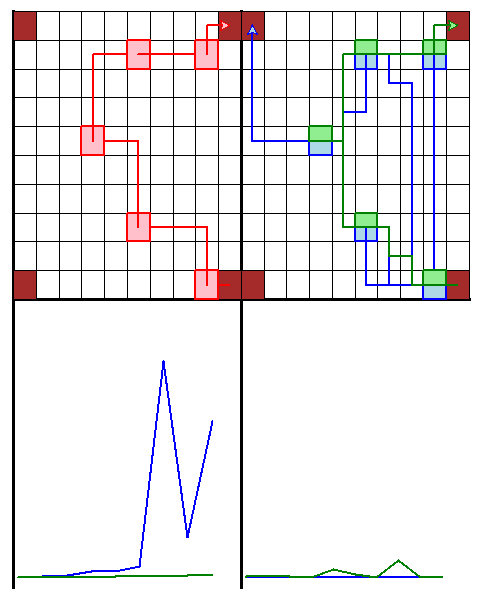
1. GUI details
   1. Created the GUI : yes
   2. Have created it according to the specifications? yes
   3. Which module of Python used for creating graphics? Turtle
   4. Is this under the standard Python library or not? yes
   5. If not, why? --N.A.--
   6. Are the window panes working independently? Yes
2. Graphics details:
   1. Is turtle graphics working fine for movement of the intelligent vacuum cleaner? Yes
   2. How are you creating the room tiles? Using Grid
   3. How are you showing the dirt? Using “light grey” color filling
   4. How are you showing the resting position of the vacuum cleaner? Using “brown” color filling
   5. Are you showing the movement of the vacuum cleaner (turtle cursor) as the execution of T1 goes on? Why? Movement starts after execution of T1 as it is not an online search
   6. Are you showing the movement of the vacuum cleaner (turtle cursor) as the execution of T2 goes on? Why? Movement starts after execution of T2 as it is not an online search
3. Compilation Details:
   1. Code Compiles (Yes/ No): Yes
   2. Mention the .py files that do not compile: --N.A.--
   3. Any specific function that does not compile: --N.A.--
   4. Ensured the compatibility of your code with the specified Python version(yes/no) yes
   5. Instructions for compilation of your files mentioning the multi file compilation process used by you (We may use the replica of these for compiling your files while evaluating your code)

Just run **python (path to the folder)/driver.py**

1. Driver Details: Does it take care of the options specified earlier :yes
2. Execution status (describe in maximum 2 lines)

Code Runs properly. It might take more time on large percentage and large matrix

1. Output Details
   1. Copy and paste the output of four graphs G1-G4 here



G3 is for 100% dust from 3x3 to 20x20 Matrix

G4 is for 10% to 100% dust in 5x5 Matrix

* 1. Write the following values computed by you (refer the details of R1-R11 in the assignment document). Use appropriate units for the values This is for 5% dust

R1: 447397 nodes R2: 104 Bytes R3: 104 Bytes R4: 43 Cost

R5: 78.948 seconds R6: 62/2578 nodes R7: 104/104 Bytes R8: 59/43 cost

R9: 0.005/0.035 seconds R10: 2.95/3.66/192.87 KB R11: 31.4/40.2/35.4 Avg Cost

1. Declaration: I, Samip Jasani (name) declare that I have put my genuine efforts in creating the python code for the given programming assignment and have submitted only the code developed by me. I have not copied any piece of code from any source. If the code is found plagiarized in any form or degree, I understand that a disciplinary action as per the institute rules will be taken against me and I will accept the penalty as decided by the department of Computer Science and Information Systems, BITS, Pilani.

ID : 2015A7PS0127P Name : Samip Jasani

Date: 14 September 2017