TPG - Rush Hour

Pedro Durval (103173)
Guilherme Casal (102587)

Actions

- Uses the function "piece_coordinates(letter)" from common.py in combination with "get_letters()" and "get_all_coords(letters)" to calculate the position of all pieces in the board.
- Following this it iterates in a for loop over the returned values of the previously used "get_letters()" method to determine the orientation of each piece as well as the possible movements relative to the occupied blocks and limits of the board.
- Finally it adds the piece and the list of possible movements in a tuple called "actlist" and adds it to the "value" list along all the other results obtained for the remaining pieces. This last list is gonna be the return value of this method.

Cost

- The cost is only used under specific conditions, that is, if the board is 6x6 or smaller (for processing reasons), and if the previous action was done in a different piece. If both of these are true, 1 is returned else 0.

Heuristic

- Heuristic is a pretty straight forward function which only calculates the sum of the pieces between the car A and the objective plus the distance to the objective.

Result

- Creates a new map object to be used as the new state of the board on the which the previously calculated action will take place. It returns a string value of the new grid.

Satisfies

- As it is the first method of the Ia class to be called in the SearchTree it creates a new Map object to be operated upon. It also contains a second parameter called goal that is only used when plan2 of student.py is in use whose objective is to correct the crazy car back to its place.

Search Tree

- Contains three classes the first of which, SearchProblem, is initiated in the student.py and receives a domain that is a Ia instance and a initial node that is gonna be later used by the SeachTree class to create the root node. SearchProblem also has a goal_test method that calls the satisfies function of Ia.
- SearchNode is the class used to create the nodes as the program evolves and it takes into consideration the current state of the grid, the parent node, the cost and heuristic of the node as well as the action that led to it.
- Finally the SearchTree class receives a SeachProblem object as well as an optional goal parameter from the student.py. With the first value it also creates the root node as well as an open_nodes list containing only the root that will be continuously expanded and popped from and a ext_nodes list with only the state of the root at first, whose purpose is to verify if a node state is being repeated.
- The search method of the SearchTree class works pretty similarly to what we saw on the practical classes with the only distinct features being the reconstruction of the plan once the solution is found which is given by a recursive increment of the reversed node parents and a while loop inside the actions for loop to iterate all the actions in the tuple.

Student

- The student.py file consists of a series of ifs that verifies if their already is a calculated plan, calculating it if needed, then verifies if a random move has occurred. If not it verifies the position of the cursor regarding the piece its supposed to move and moves it to the correct spot to execute the planned action.
- However if there is a random movement, one of two things may occur:

  -If the processing time is longer than 2 seconds it simply creates a new plan with the goal to put the random piece in the former place and continues the plan where it left of

  -If its lower than 2 seconds it re-calculates the entire plan and executes it.

- Recalculates if time is greater than 2s.
- Last “if” from student that compares a tmp variable with a constant