Itziar López Almagro

Artificial Intelligence

December 15, 2023

REPORT OF THE PROJECT

I. Suggest a formal state representation and write initial state, goal state and operators in the representation.

- **STATE REPRESENTATION**

  I used a matrix 2*2 called “state” for representing the state of the node:

  I) If state[i][j]=0 then that position is empty.

  II) If state[i][j]=1 then there is dirt in that position.

  III) If state[i][j]=2 then the vacuum cleaner is in that position.

  IV) If state[i][j]=3 means that there is dirt and the vacuum cleaner is also in that position.

- **GOAL STATE**

  The objective is achieved when there is no dirt in any position. For controlling the quantity of dirt in each state I used a variable called ‘num_dirt’. If the value of 'num_dirt' is 0 then we have successfully reached to a goal state.

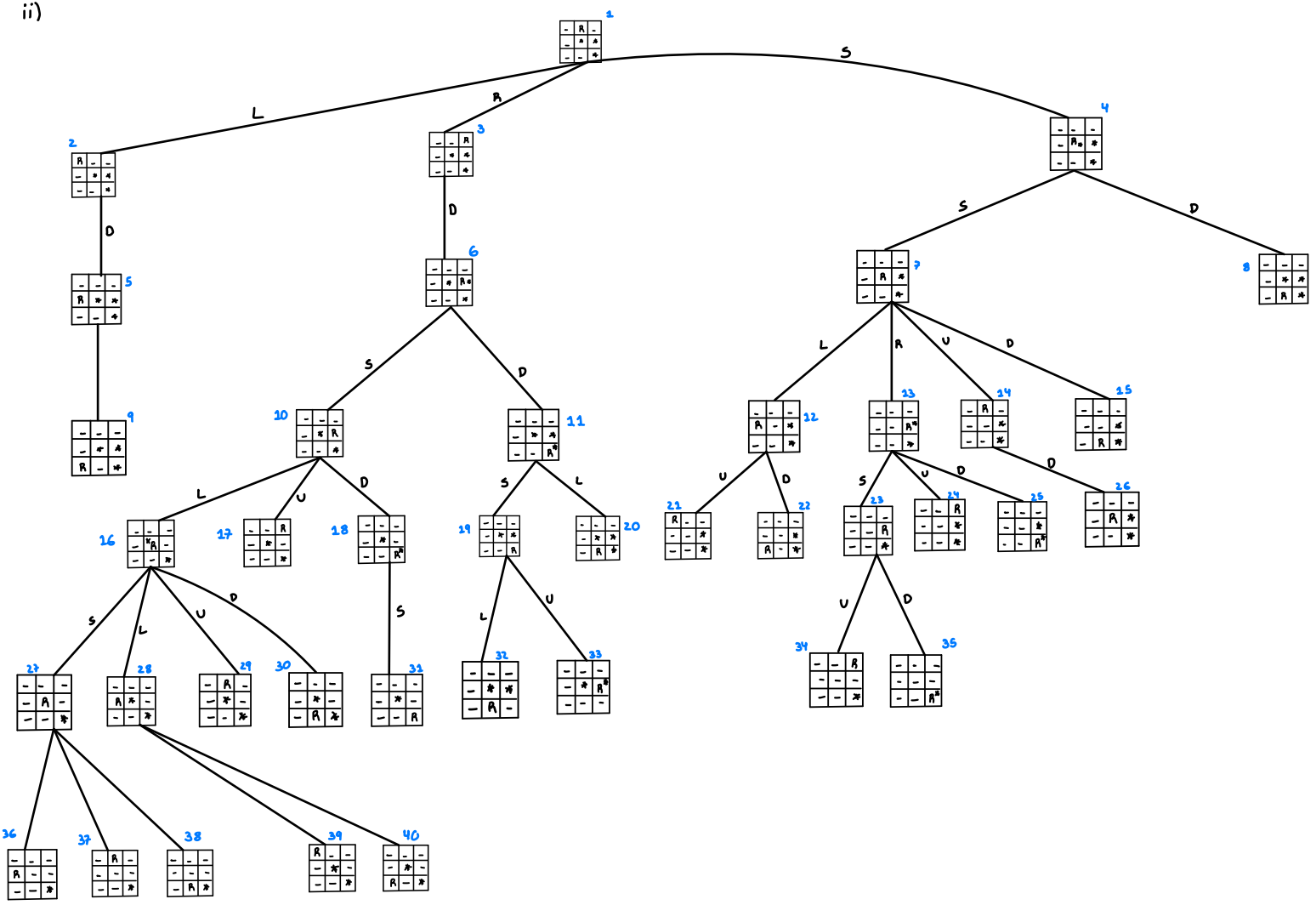
- **INITIAL STATE**

  The initial state will be the one provided by the user.

- **OPERATORS**

  I have implemented a C++ enum to represent operators, replacing numerical codes. This enhances clarity in the code. The enum ,called ' Op', include types such as ' SUCK', ' LEFT', ' RIGHT', ' UP' and ' DOWN'.

ii)



Note: I stopped generating nodes because I didn't have enought space and probably I have been doing something wrong generating so many.

III.

OPEN	CLOSED
[1]	[]
[2,3,4]	[1]
[3,4,5]	[1,2]
[4,5,6]	[1,2,3]
[5,6,7,8]	[1,2,3,4]
[6,7,8,9]	[1,2,3,4,5]
[7,8,9,10,11]	[1,2,3,4,5,6]
[8,9,10,11,12,13,14,15]	[1,2,3,4,5,6,7]
[9,10,11,12,13,14,15]	[1,2,3,4,5,6,7,8]
[10,11,12,13,14,15]	[1,2,3,4,5,6,7,8,9]
[11,12,13,14,15,16,17,18]	[1,2,3,4,5,6,7,8,9,10]
[12,13,14,15,16,17,18,19,20]	[1,2,3,4,5,6,7,8,9,10,11]
[13,14,15,16,17,18,19,20,21,22]	[1,2,3,4,5,6,7,8,9,10,11,12]
[14,15,16,17,18,19,20,21,22,23,24,25]	[1,2,3,4,5,6,7,8,9,10,11,12,13]
[15,16,17,18,19,20,21,22,23,24,25,26]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14]
[16,17,18,19,20,21,22,23,24,25,26]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
[17,18,19,20,21,22,23,24,25,26,27,28,29,30]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]
[18,19,20,21,22,23,24,25,26,27,28,29,30]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]
[19,20,21,22,23,24,25,26,27,28,29,30,31]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18]
[20,21,22,23,24,25,26,27,28,29,30,31,32,33]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]
[21,22,23,24,25,26,27,28,29,30,31,32,33]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
[22,23,24,25,26,27,28,29,30,31,32,33]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21]
[23,24,25,26,27,28,29,30,31,32,33]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22]
[24,25,26,27,28,29,30,31,32,33,34,35]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23]
[25,26,27,28,29,30,31,32,33,34,35]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]
[26,27,28,29,30,31,32,33,34,35]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25]
[27,28,29,30,31,32,33,34,35]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26]
[28,29,30,31,32,33,34,35,36,37,38]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27]
[29,30,31,32,33,34,35,36,37,38,39,40]	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28]

IV. About 30 nodes were expanded.

V. DATA STRUCTURES AND IMPLEMENTATION

1. STRUCT POS: I have designed a data structure consisting of two integers, denoted as 'x' and 'y.' This structure serves the purpose of storing the coordinates representing the position of the vacuum cleaner.
2. ENUM OP: as I explained, this is the variable we will use for the operators.
3. STRUCT NODE: It contains all the essential details needed to provide a description of a single node.
    - Node* parent: it is a pointer to the parent of the node. This will be used later in the function called 'recover()' for giving the solution path. If the pointer is null it means that is the root of the tree.
    - Op parent: it is the operator the parent executed for reaching to the node.
    - Int num_dirt: it represents the quantity of dirt present in the corresponding state of the node.
    - Vector<vector<int>> state: it is a matrix of 3*3 that represents if there is dirt , dirt and the vacuum cleaner or just the vacuum cleaner in each position.
4. STRUCT DFS: it has all the necessary information for executing the DFS.
    - Node actual: node we are about to expand.
    - List<Node> open is the open list.
    - List<Node> closed is the closed list.

VI. Screen shot of the program final output.