Name : Ebrahim Elgaml ID: 28-4498 Tut : 9
Name : RaghdaKhaled ID: 28-3256 Tut : 9
Name : Ansary ID: 28-2563 Tut : 10

1. There is MxN grid representing a maze that has number of pokemons, The goal is to collect the pokemons, move specific number of steps to make the egg hatch and reach the end of the maze.
2. Structure of the node →
    a. State → its type is a State which consists of json object {“Key”: Value}.
    b. Parent → it is a node also.
    c. Depth → it is a number representing the depth of the current node starting from zero.
    d. Operator → its type is Operator which is a class consists of two parameters, First is a function that takes a state and returns a new state and the second is the cost of the operation
    e. PathCost → The cost to reach that node.
    f. EstimatedCost → the cost from the heuristic function.
3. Search problem structure →
    a. List of operators each element in the list is an Operator object as discussed above.
    b. Initial state → its type is state
    c. State space → all the states.
    d. Goal test → function that takes a state and returns boolean true or false either it is the goal node or not.
    e. Path cost function → functions takes two parameters. First one is the old cost and the other one is an operator and returns a new cost of applying the operator and adding its cost to the old cost.
4. Implementation →
    a. First a maze is generated according to the input of its dimensions, It is randomly generated with walls and start and end cells.
    b. The smallest building unit of the maze is cell. Cell is a class containing attributes describe the state of every cell in the maze(has right, left, lower, upper walls and has pokemons or not and also the position of the cell.
    c. The pokemons are generated randomly also in the maze by setting the boolean attribute in the cell by true or false.
    d. After generating the maze a random end and random start point is set for it.
    e. Also after creating the maze the number of required steps for the egg to hatch is generated.
    f. For initializing the search problem :-
        i. Each state is represented by { current cell, remaining steps for the egg to hatch, The positions of the remaining pokemons.
        ii. The state space is the maze
        iii. There are four operators each one with cost 1 (moveUp, moveDown, moveLeft, moveRight)
        iv. Goal test is a function which takes a state and returns true only if the cell in the state is the end cell, the remaining steps for hatching the egg is zero, and the array of pokemons positions is empty.

v. Path cost is a function that takes the accumulated cost and an operator and returns a new cost which is the old cost plus the cost of doing that operation.

5. Main functions :
    a. GenMaze(M, N) →
        i. It calls the constructor of Maze class to generate MxN maze plus generate random value for the steps to hatch the egg, It places random pokemons in each cell in the maze. Moreover, it sets end and start point for the maze.
    b. Heuristic function for each heuristic approach will be discussed later.
    c. search(problem:SearchProblem) →
        i. It is the general search algorithm. It extract the initial node of the problem and enqueue it in a queue and enters a loop that runs until the queue is empty. Each time it extract a node it inserts its children in the queue according to the queuing function for the algorithm.
        ii. The number of expanded nodes is calculated inside it.
        iii. A check for the repeated state is done inside it by checking for every parent of the node until we reach the first parent to check that we have not visited the expanded node before.
    d. ordered (nodes:Node[], node:Node) →
        i. Insertion sort for the node in the array of nodes. It is used for implementing priority queuing function of uniform cost search.
    e. DepthlimitedSearch: search(problem:SearchProblem, depth:number):any: →
        i. It performs the search algorithm function but until a certain depth(depth).
        ii. It is used in iterative deepening as we loop from depth equal zero till we find the goal and at each iteration this functions is called.
    f. goalTest(state) →
        i. Test if the state is the goal or not by checking if it is the end cell, the number of steps for hatching the egg is zero and all the pokemons are collected.
    g. BestFirstSearch(problem:SearchProblem, evalInfo:any, evalFunc:(node:Node, information:any) => void) →
        i. Function for best first search algorithm. It takes an evaluation functions and create a queuing functions with it, After that it call the general search algorithm using the generated queuing function on the input problem.
        ii. It is used for the greedy and A*.

6. Search Algorithms:
    a. BFS: The BFS was implemented by expanding all the nodes in their order of depth, where nodes at depth d were expanded before nodes at depth d+1, the queueing function of the BFS inserts nodes at the end of the queue
    b. UniformCost: It was implemented the same way as the BFS, however the difference is that the nodes with lower path cost from the root were expanded

first, the queueing function of it inserts the nodes bases on the lower cost priority, where nodes with lower path cost function are inserted first.

c. DFS: It was implemented by expanding the deeper nodes first. If a dead end was hit, it backtracks , the queuing function of the DFS inserts nodes at the front of the queue.
d. Depth Limited Search: It was implemented the same way as the DFS, but based on a specific limited depth.
e. Iterative Deepening Search: It's the same as the Depth Limited Search, but the whole algorithm is repeated from the 1st depth till infinity.

7. Heuristic Functions:

- Manhattan Distance between the current position and the position at the goal state:
  - The Manhattan distance between two points denotes the sum of dx and dy, where dx represents the horizontal difference and dy represents the vertical difference.
  - Argument of admissibility: This heuristic obtains the distance that would be covered to reach the final position if no pokemons are remaining, the egg has already hatched and no walls exist. It will always underestimate because the actual distance will always be greater.

- Minimum spanning tree of the positions representing the current position, all pokemon positions and the end position.
  - Argument of admissibility: the spanning tree gets the minimum cost of connecting all nodes in a tree making sure no cycles exist. The total cost of the resulting edges is less than the actual distance that would be moved to reach all positions since we ignore walls, we use Manhattan to identify the cost of each edge, and we don't account for steps needed for the egg to hatch.

- Distance between the two furthest positions (current point, pokemon positions, endpoint)
  - Argument of admissibility: This distance, again Manhattan distance, represents the longest edge in the tree of positions. To collect all pokemons we sure need to move more than this distance otherwise we will not have collected all pokemons or reached the final position.

Name : Ebrahim Elgaml ID: 28-4498 Tut : 9
Name : RaghdaKhaled ID: 28-3256 Tut : 9
Name : Ansary ID: 28-2563 Tut : 10

8. Testing on the same 3X3 maze →

Strategy: BF

Search Started
Passed goalTest and node depth is 9
No. of nodes: 50
No. of repeated states: 38

Strategy: UC

Search Started
Passed goalTest and node depth is 9
No. of nodes: 50
No. of repeated states: 38

Strategy: DFS

Search Started
Passed goalTest and node depth is 11
No. of nodes: 13
No. of repeated states: 8

Strategy: ID

Search Started
No solution for depth: 0
Nodes expanded: 0
No solution for depth: 1
Nodes expanded: 1
No solution for depth: 2
Nodes expanded: 3
No solution for depth: 3
Nodes expanded: 7
No solution for depth: 4
Nodes expanded: 15
No solution for depth: 5
Nodes expanded: 22
No solution for depth: 6
Nodes expanded: 30
No solution for depth: 7
Nodes expanded: 35
No solution for depth: 8
Nodes expanded: 42
No solution for depth: 9
Nodes expanded: 46

Passed goalTest for depth: 10
No. of nodes: 28
No. of repeated states: 21

Strategy: GR1
Search Started
Passed goalTest and node depth is 9
No. of nodes: 44
No. of repeated states: 32

Strategy: AS1
Search Started
Passed goalTest and node depth is 9
No. of nodes: 41
No. of repeated states: 30

Strategy: GR2
Search Started
Passed goalTest and node depth is 9
No. of nodes: 35
No. of repeated states: 26

Strategy: AS2
Search Started
Passed goalTest and node depth is 9
No. of nodes: 33
No. of repeated states: 24

Strategy: GR3
Search Started
Passed goalTest and node depth is 9
No. of nodes: 19
No. of repeated states: 10

Strategy: AS3
Search Started
Passed goalTest and node depth is 9
No. of nodes: 44
No. of repeated states: 33

The only two optimal algorithms or strategies are A* and UC, the obtained results support the fact that there is no optimal algorithm that expands less nodes than the A* algorithms.

The maze will always lead to a solution through any path due to the way the mazeGeneration algorithm is implemented.

Since we handle repeated cases, non-complete algorithms , DF and Gr, always find an answer.

Name : Ebrahim Elgaml ID: 28-4498 Tut : 9
Name : RaghdaKhaled ID: 28-3256 Tut : 9
Name : Ansary ID: 28-2563 Tut : 10

9. Citation: https://en.wikipedia.org/wiki/Maze_generation_algorithm
10. The program runs.