Artificial Intelligence Laboratory

Uninformed Search

Kunal Kumar
Aditya Jha

13th Jan 2020

Problem Statement:

Teach Pacman how to intelligently find his food! The objective of this task is to simulate *Breadth-first search, Depth-first search*, and *Depth first Iterative Deepening* in the state space. The state-space consists of an M*N grid. The start state is (0,0). The goal state is the position of (*) in the grid. The Pacman is allowed to move UP, DOWN, LEFT and RIGHT (except for boundary).
Compare the above search methods on two accounts:
1. Length of the path (from the initial state to the goal state) that each algorithm finds
2. Number of states explored (visited) during the search.
(The length of the path and the number of visited states are two different things). Also, analyze whether or not the result depends on the order in which the neighbors of each node are added into the list should be done.

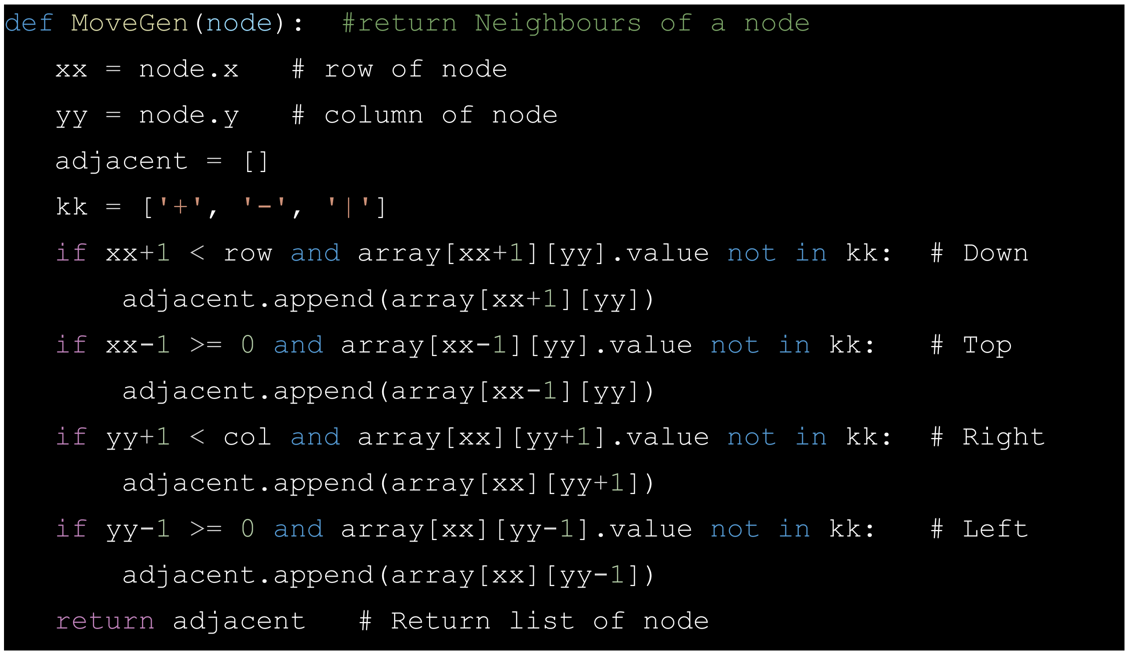
Pseudo Code:

We have a 2D-Array of size M*N. For each element in array we created a Class Node. We set node location as position of that node in Array. We assign color to node as:

White ---> Not visited
Gray ---> Visiting and looking for neighbours
Black ---> Visited node

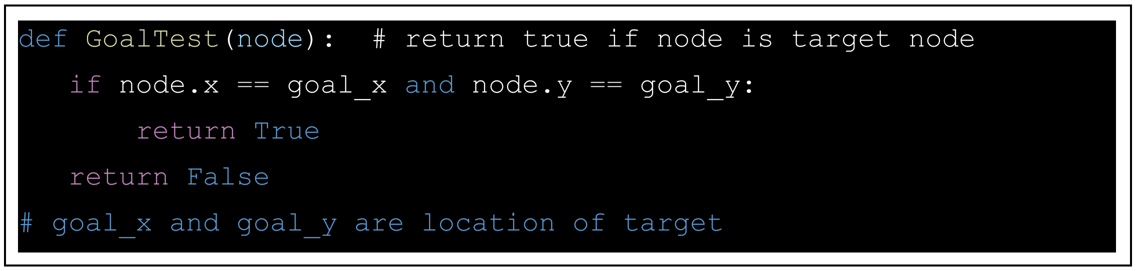
```
class Node:
   def __init__(self):
       self.value = ''     # initializing with blank space
       self.color = 'white'
       self.dis = -1       #distance of node from (0,0) in different
tree
       self.parent = None
       self.x = -1       # x-coordinate of node
       self.y = -1       # y-coordinate of node
```

MoveGen():



```python
def MoveGen(node):  #return Neighbours of a node
   xx = node.x   # row of node
   yy = node.y   # column of node
   adjacent = []
   kk = ['+', '-', '|']
   if xx+1 < row and array[xx+1][yy].value not in kk:  # Down
       adjacent.append(array[xx+1][yy])
   if xx-1 >= 0 and array[xx-1][yy].value not in kk:   # Top
       adjacent.append(array[xx-1][yy])
   if yy+1 < col and array[xx][yy+1].value not in kk:  # Right
       adjacent.append(array[xx][yy+1])
   if yy-1 >= 0 and array[xx][yy-1].value not in kk:   # Left
       adjacent.append(array[xx][yy-1])
   return adjacent   # Return list of node
```

GoalTest():



```python
def GoalTest(node):  # return true if node is target node
   if node.x == goal_x and node.y == goal_y:
       return True
   return False
# goal_x and goal_y are location of target
```

Results:

Down > Up > Right > Left

Type	Sample Input	Grid Size				Metrics	
		Cells Across	Cells Down	Cells Width	Cells Height	Path length	States explored
BFS	1	4	4	4	3	24	42
	4	5	5	4	3	33	59
DFS	2	4	4	4	3	26	46
	5	5	5	4	3	37	82
DFID	3	4	4	4	3	24	862
	6	5	5	4	3	33	2172

Results:

Left > Right > Up > Down

Type	Sample Input	Grid Size				Metrics	
		Cells Across	Cells Down	Cells Width	Cells Height	Path length	States explored
BFS	1	4	4	4	3	24	42
	4	5	5	4	3	33	59
DFS	2	4	4	4	3	24	24
	5	5	5	4	3	33	41
DFID	3	4	4	4	3	24	621
	6	5	5	4	3	33	1358

Dependence of results on the order of neighbors added:

BFS:

1. Path length does not depend on the order in which the neighbours of each node are added.
2. The number of nodes explored depends on the order in which the neighbours of each node are added.

DFS:

1. Path length depends on the order in which the neighbours of each node are added.
2. The number of nodes explored also depends on the order in which the neighbours of each node are added.

DFID:

1. Path length does not depend on the order in which the neighbours of each node are added.
2. The number of nodes explored depends on the order in which the neighbours of each node are added.