WOJSKOWA AKADEMIA TECHNICZNA

Sprawozdanie z przedmiotu:

Sztuczna Inteligencja

Prowadzący: mgr inż. Przemysław Czuba

Autor: Mateusz Jasiński WCY20IY2S1

Algorytm DFS

Zaimplementowałem algorytm DFS używając rekurencji.
Kod funkcji tego algorytmu wygląda następująco:

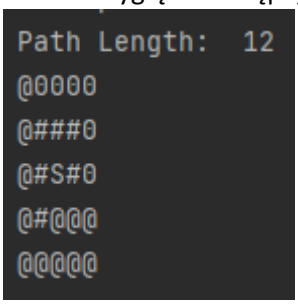
```
def run_dfs(maze_puzzle, current_point, visited_points):
    visited_points.append(current_point)
    neighbors = maze_puzzle.get_neighbors(current_point)
    for neighbor in neighbors:
        if not is_in_visited_points(neighbor, visited_points):
            neighbor.set_parent(current_point)
            if maze_puzzle.get_current_point_value(neighbor) == '*':
                return neighbor
            w = run_dfs(maze_puzzle, neighbor, visited_points)
            if w != 'No path to the goal found.':
                return w
    return 'No path to the goal found.'
```

Kod działa następująco:
1) Oznacza obecnie odwiedzany punkt jako „odwiedzony”,
2) Pobiera listę sąsiednich punktów (które nie są ścianami),
3) Po kolei dla każdego z tych punktów:
-Sprawdza czy już nie był już odwiedzony, jeżeli był to idzie do przechodzi do następnego punktu,
-Jeżeli rozpatrywany punkt nie był odwiedzony to ustawia obecny punkt jako „poprzednika”,
-Jeżeli rozpatrywany punkt jest celem to zwraca współrzędne tego punktu,
-Jeżeli rozpatrywany punkt nie jest celem to wracamy do podpunktu 1) używając współrzędnych rozpatrywanego punktu,
-Jeżeli wartość otrzymana z rekurencji nie jest powiadomieniem o braku drogi to zwraca otrzymaną wartość (współrzędne celu),
4) Zwraca powiadomienie o braku drogi.

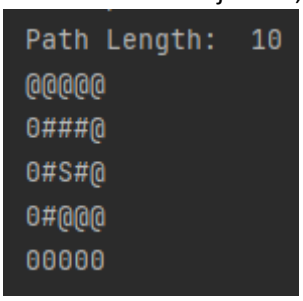
Wpływ kolejności akcji na wynik algorytmu DFS

Kolejność akcji (góra, dół, prawo, lewo) ma olbrzymie znaczenie na kształt ścieżki jaką zwróci algorytm. Algorytm ten polega na sprawdzaniu ścieżki do końca, a w przypadku, gdy nie była ona drogą do celu, to cofa się i sprawdza pozostałe możliwości (też w określonej kolejności). To, że algorytm zwróci inne ścieżki w zależności od kolejności akcji możemy zauważyć nawet w podstawowym labiryncie.

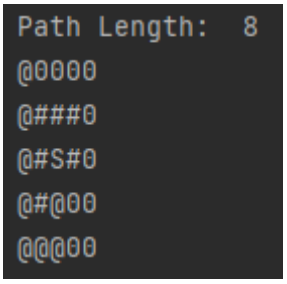
Ścieżka wygląda następująco dla akcji lewo, prawo, dół, góra:



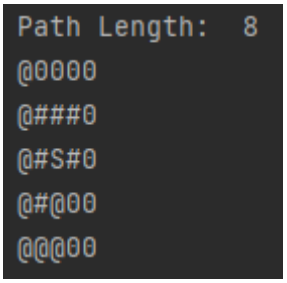
Natomiast dla akcji lewo, prawo, góra, dół:



Jeszcze inna ścieżka powstaje dla akcji lewo, dół, prawo, góra:



Oczywiście istnieje ograniczona liczba ścieżek i powtarzają się one dla niektórych akcji np. lewo, dół, prawo, góra oraz dół, lewo, góra, prawo dadzą tą samą drogę:



Automatyczna generacja mapy

Mój kod do generowania mapy wygląda następująco:

```
self.maze = []
for j in range(self.maze_size_y):
    self.maze.append('')
    for i in range(self.maze_size_x):
        if j % 2 == 0:
            self.maze[j] += '0'
        else:
            self.maze[j] += '#'
for j in range(1, self.maze_size_y, 2):
    for i in range(randint(1, math.ceil(maze_size_x / 3))):
        rand = randrange(self.maze_size_x)
        self.maze[j] = self.maze[j][:rand] + '0' + self.maze[j][rand + 1:]
rand = randrange(self.maze_size_x)
self.maze[maze_size_y - 1] = self.maze[maze_size_y - 1][:rand] + '*' +
self.maze[maze_size_y - 1][rand + 1:]
```

Działa on tak:
-Tworzy wiersze z pustymi polami oraz wiersze ze ścianami na przemian,
-Robi od 1 do sufitu z 1/3 wielkości mapy „dziur” w wierszach ze ścianami,
-Ustawia cel w losowym punkcie w „najniższym” wierszu.

Porównanie DFS i BFS

Wymiary mapy	DFS		BFS	
	Czas [ms]	Droga	Czas [ms]	Droga
5x5	1	10	1	10
10x10	2	30	2	16
25x25	7	207	14	47
50x50	93	654	147	168
90x90	1152	2751	1432	251
250x250	Błąd	Błąd	91614	1054

Algorytm DFS jest optymalniejszy pod względem czasu wykonania, ale przeważnie nie daje optymalnej ścieżki oraz potrzebuje o wiele więcej pamięci operacyjnej.
Algorytm BFS daje optymalną ścieżkę oraz wymaga mniej pamięci operacyjnej, ale zajmuje mu to więcej czasu.

Z powodu tego, że zaimplementowałem algorytm DFS w sposób rekurencyjny środowisko PyCharm oraz moje podzespoły nie radzą sobie z dużą liczbą rekurencji. Pomimo zwiększenia maksymalnej liczby rekurencji z 997 na 3000 program dalej ma problemy i nie może się wykonać dla map z wymiarami powyżej 90x90 oraz zwraca przy tym błędy.

```
---Depth-first Search---
Podaj długość boku labirytnu: 100

Process finished with exit code -1073741571 (0xC00000FD)
```

Algorytm BFS bezproblemowo wyszukiwałby ścieżki dla map z wymiarami większymi niż 250x250, ale zajęłoby to dużo czasu, z powodu dużej (i szybko rosnącej) złożoności obliczeniowej.