Programming Project: Assignment Part I

ThankYouEnjoy

Woo Jung
Thomas Tarantino

October
2019

Group declaration: FILL IN

Level	Frontier	Time	Memory Used	Solution length	States Generated
SAD1	BFS	0.201 s	24.33 MB	19	80
SAD1	DFS	0.204 s	13.95 MB	27	75
SAD2	BFS	—	exceeded	—	—
SAD2	DFS	0.184 s	10.85 MB	25	86
SAfriendofDFS	BFS	—	exceeded	—	—
SAfriendofDFS	DFS	0.225 s	31.83 MB	60	305
SAfriendofBFS	BFS	0.686 s	138.64 MB	3	1,227
SAfriendofBFS	DFS	—	exceeded	—	—
SAFirefly	BFS	—	exceeded	—	—
SAFirefly	DFS	—	exceeded	—	—
SACrunch	BFS	—	exceeded	—	—
SACrunch	DFS	—	exceeded	—	—

Table 1: Benchmarks table for Exercise 1

1 Exercise 1

(a) Refer to Table 1.

(b) There are three boxes adjacent to the agent in the initial state. Using BFS, the agent is forced to consider neighbor boxes at the present depth prior to moving on in each iteration. This means, the agent has significantly more states to iterate through before moving onto a higher depth state.

(c) Our implementation of DFS differs from the implementation of BFS in that it uses a stack instead of a queue since DFS requires LIFO data structure.

(d) Our BFS algorithm has a much better performance on this level because the initial state of the level is close to the goal state in the sense that it does not require much moving of the boxes. On the other hand, our DFS algorithm performed poorly because the algorithm has to iterate through each state until success or failure which resulted in a much longer search time – well it didn't end up solving.

(e) Our BFS algorithm performs poorly because there are too many boxes in the map so the search space becomes exponential. Since DFS is not limited to searching the entire depth level's search space before progressing, the DFS is not bound by the number of boxes in the map.

(f) Refer to Table 1.

2 Exercise 2

3 Exercise 3