Lab2

Teodor Riddarhaage (teori199)
Pontus Svensson (ponsv690)

October 2018

1 Theory

1. The states were positions in the world and the actions were movements. The branching factor is four since each node has at most four adjacent nodes.

2. There is no difference if the cost is one since the ordering for UCS will essentially be a regular FIFO queue just like for BFS.

3. a and c.

   (a) This is simply the average of h_1 and h_2 and since both are admissible their average must also be admissible.

   (b) $2h_1$ is not guaranteed to be admissible since this value might be larger than the actual cost.

   (c) Since both h_1 and h_2 are admissible then the maximum of these must, of course, also be admissible.

4. An admissible heuristic (h) could be the straight line path from the current node to the goal node and the cost function (g) could be the number of squares the agent has traversed to get to the current node from its starting position (assuming that turning is cost free).

5. The domain we chose is presented in Figure 1. This problem was solved using our three favorite search algorithms below.

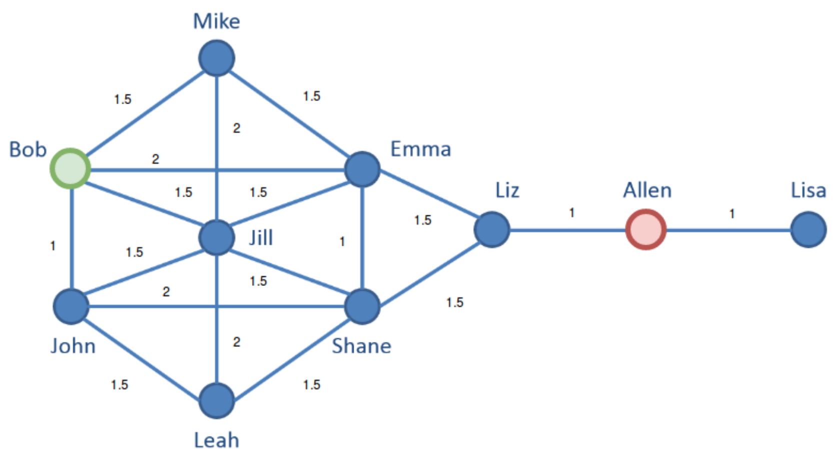

Figure 1: Search domain

Breadth-first search

The search tree is presented in Figure 2. Starting with Bob, his neighbors are added, in a clockwise direction, to the search tree. These neighbors are then expanded, from left to right. This process is repeated for all levels in the search tree until the goal node is found.

The memory usage for this search algorithm is at most the number of nodes (states) in the domain.

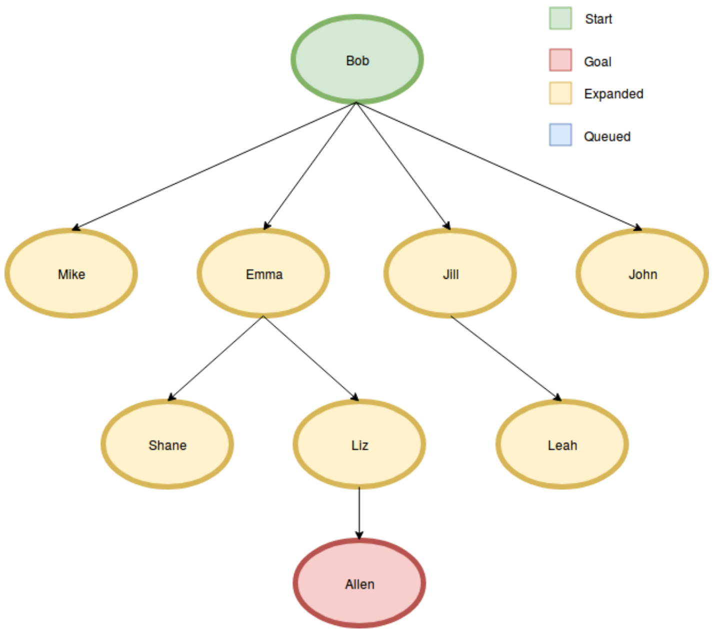

Figure 2: Search tree for BFS

Depth-first search

The search tree is presented in Figure 3. This algorithm expands the tree downwards from left to right as far as possible or until the goal node is found. The difference between this algorithm and BFS is the order in which the nodes in the search tree are expanded. In DFS, the nodes are primarily expanded vertically rather than horizontally.

The memory usage for this search algorithm is at most the number of nodes (states) in the domain.

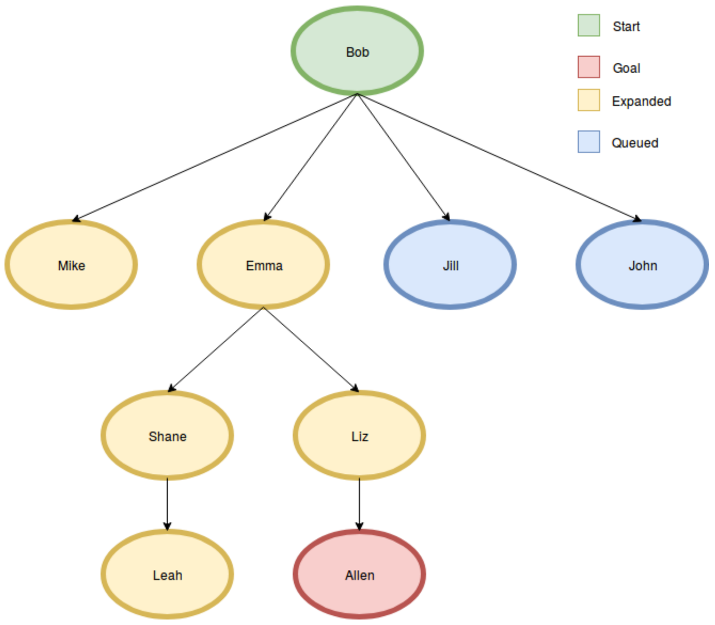

Figure 3: Search tree for DFS

Best-first search

This algorithm uses the edge costs to find the shortest path. The node with the lowest cost is expanded until the goal node is found. The algorithm works like BFS and DFS except that it expands nodes based on their distance from the start node rather than vertically or horizontally.

The memory usage for this search algorithm is at most the number of nodes (states) in the domain.

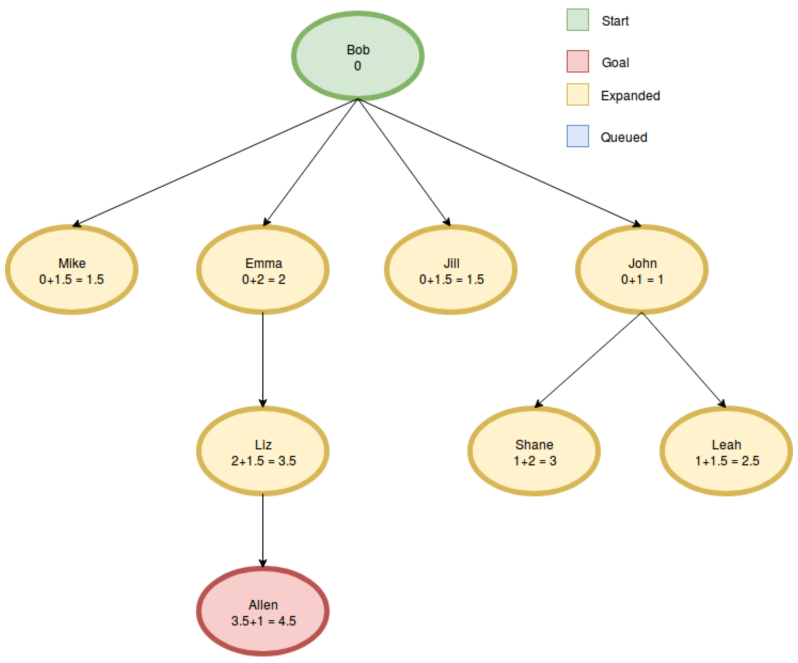

Figure 4: Search tree for Best first search

6. Comparison between different offline search algorithms

Criterion	Breadth-first	Uniform-cost	Depth-first	Depth-limited	Iterative deepening	Bidirectional	A*
Complete?	Yes	Yes	No	No	Yes	Yes	Yes
Optimal?	Yes	Yes	No	No	Yes	Yes	Yes

7. We think the most suited search algorithm for the environment in lab 1 would be breadth-first search, searching for unknown tiles.