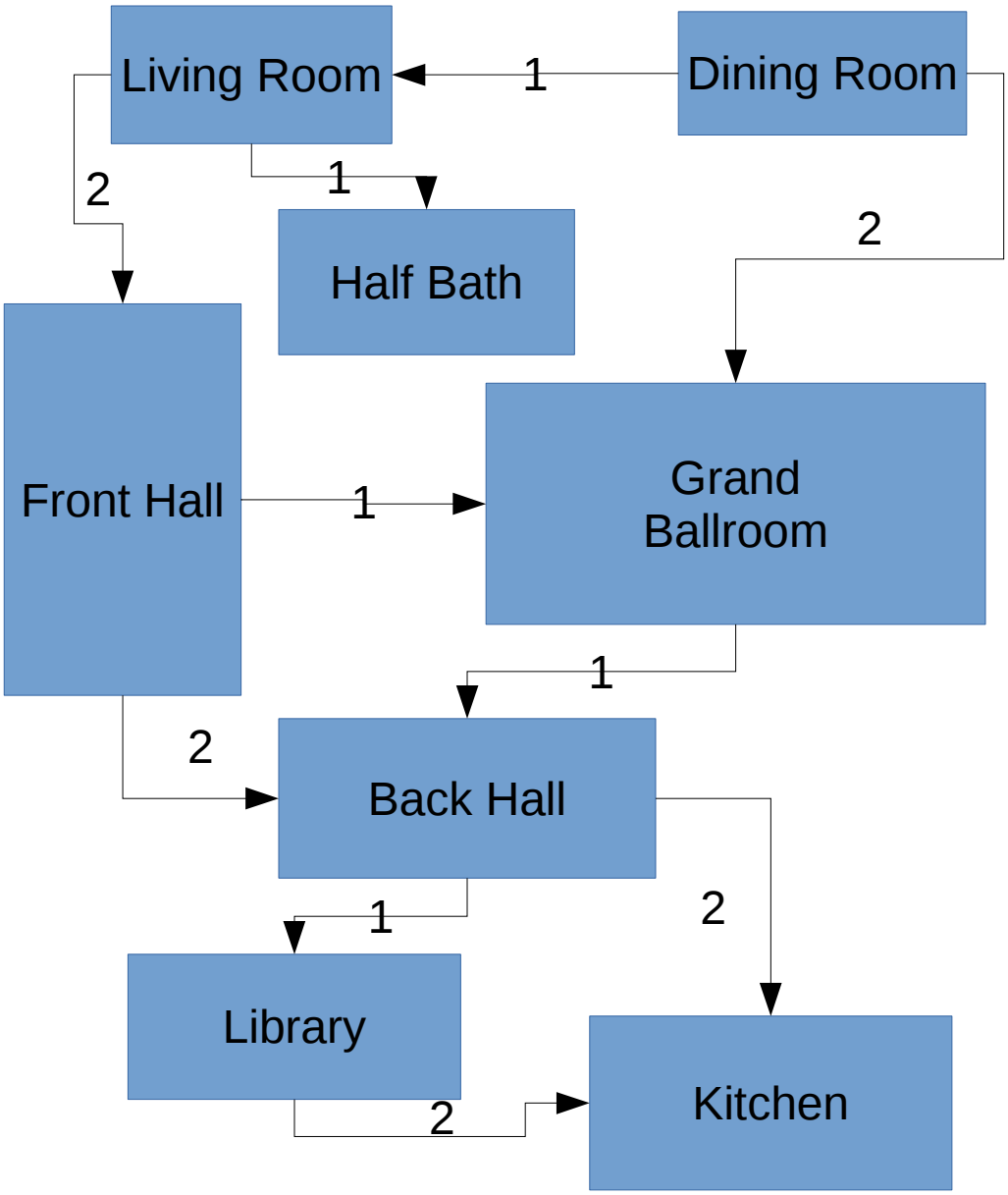
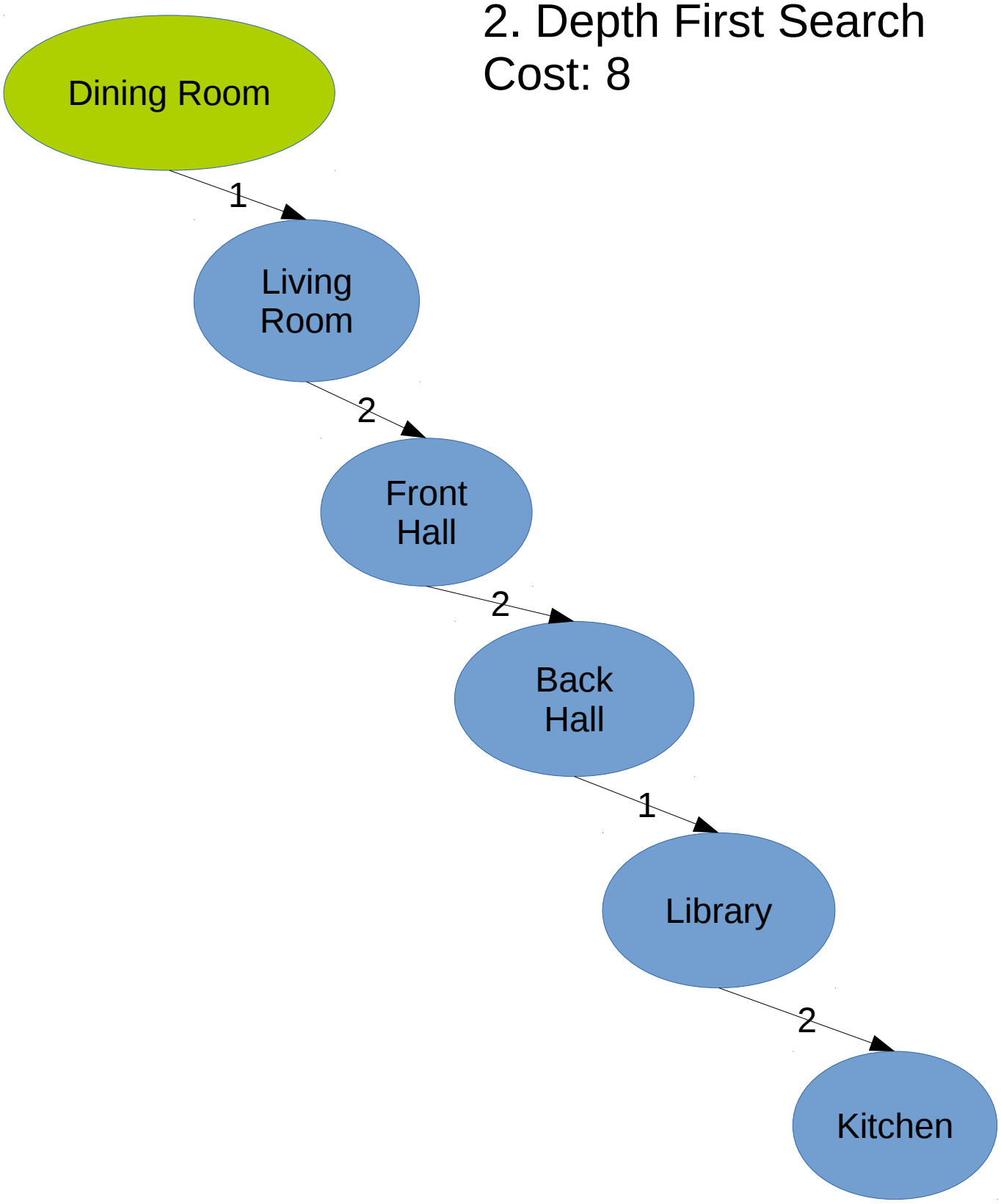
Written Questions
Mary Yen; myen3@jhu.edu

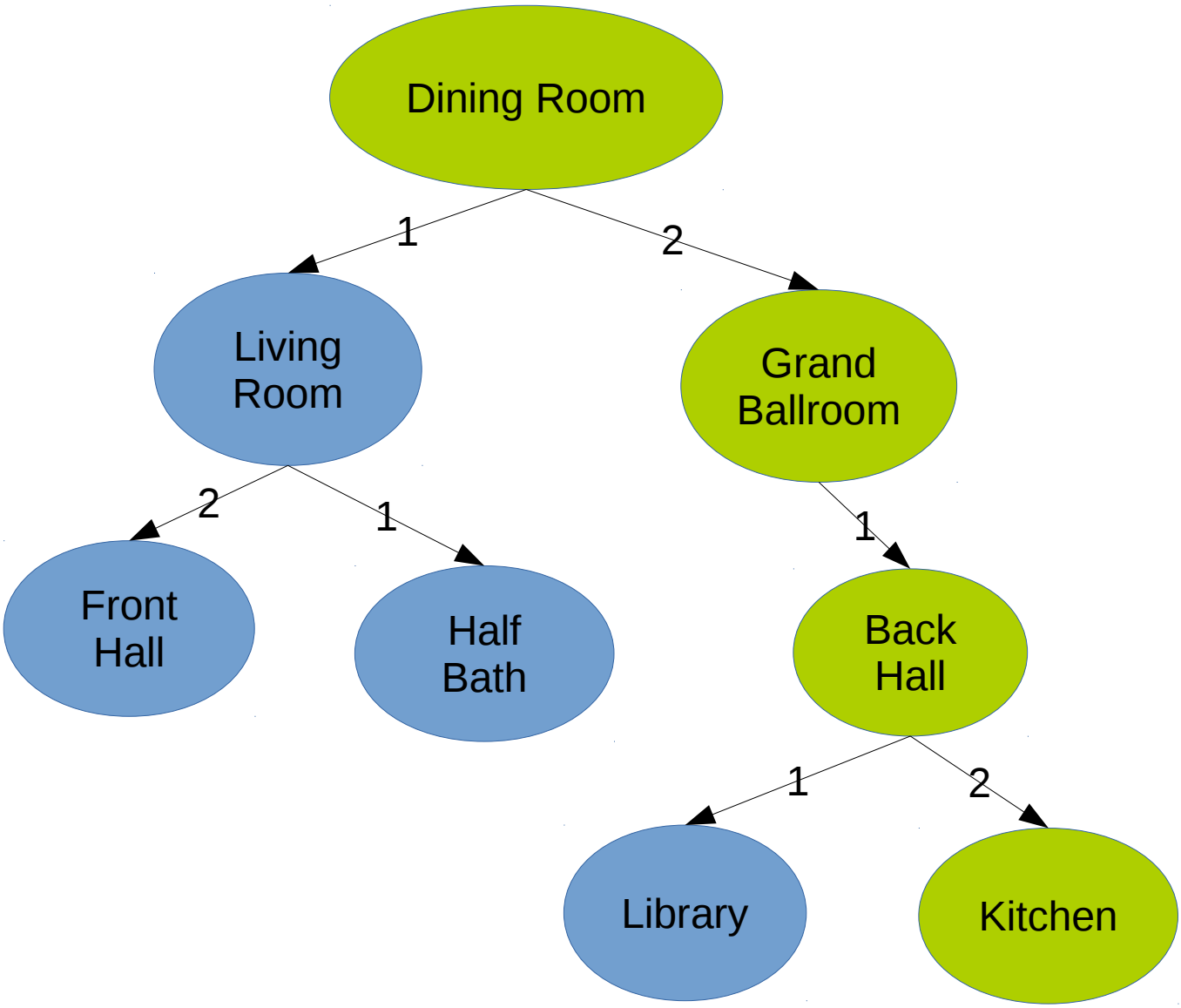
1. State-Space of Robot Agent



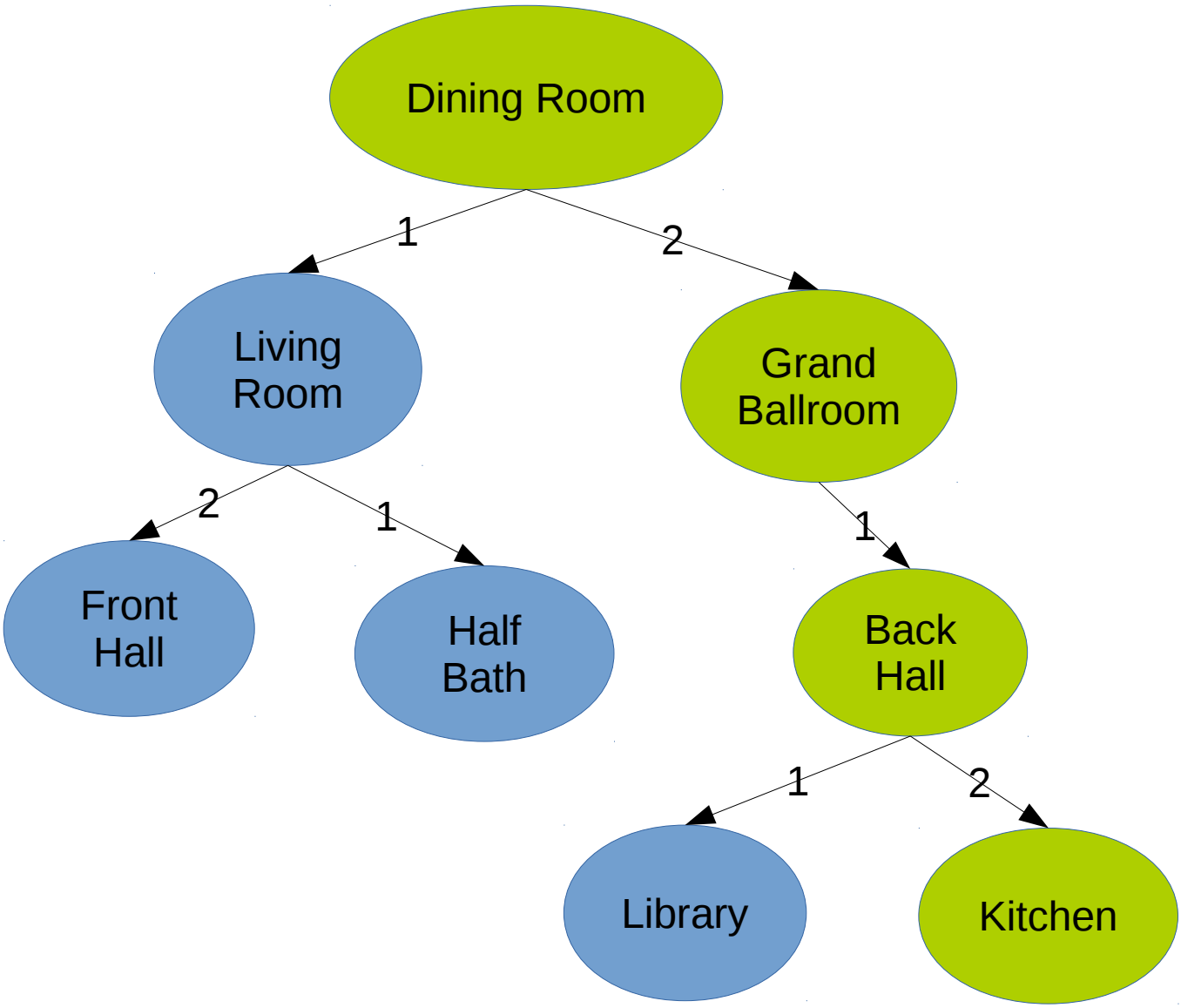
Arrows are arbitrary since the cost of movement is the same in both directions.


2. Depth First Search
Cost: 8
Dining Room
1
Living Room
2
Front Hall
2
Back Hall
1
Library
2
Kitchen

3. Breadth First Search
Cost: 5 (Along the green route)
Note that we do not allow node repeats.



4. Uniform Cost Search
Cost: 5 (Along the green route)
Note that we do not allow node repeats.



5. Comparing Search Algorithms

Map: map file number
Nodes: number of nodes expanded
Time: using 'time' command, the user value gives the program runtime in seconds.
Cost: cost of the path to the goal.

BFS			
Map	nodes	time	cost
1	7	.073	6
2	14	.068	13
3	32	.06	13
4	42	.074	21
5	44	.051	20
6	926958	18.352	9498
7	38	.065	infinite
8	22	.076	20

DFS			
Map	nodes	time	cost
1	7	.066	6
2	14	.085	13
3	25	.056	13
4	34	.065	33
5	23	.071	23
6	943188	31.02	37082
7	38	.068	infinite
8	13	.07	20

A*S			
Map	nodes	time	cost
1	7	.067	6
2	14	.051	13
3	31	.079	13
4	42	.064	21
5	44	.070	14
6	926880	19.8	9498
7	38	.047	infinite
8	14	.080	20

6. My A* heuristic first estimates the Manhattan distance from the current node to the goal node, first. If there are enough periods (cost = 1; the minimum cost) to populate the distance, then we will assume the Manhattan distance is the answer.
However, if there are not enough periods, we assume that the rest of the Manhattan distance is made of commas (cost=2). Note that in the algorithm, one element on the path to the goal node is guaranteed a cost of 1 (this is because it costs 1 to move to the goal node).
If there are not enough periods and commas to reach the goal node, this map does not have a reachable goal node.
This heuristic is admissible since it will not overestimate the path cost (It defaults to the minimum path of periods if available), but is slightly more accurate should there be commas.
Later maps will find this heuristic more efficient since the heuristic helps determine a possibly more cost efficient path.

7. A DFS would fail if there were any cycles in a tree to get stuck in. BFS will continue just fine.

8. BFS would fail if a tree had too much branching and elements, which may cause the program to fail if too much memory is required. DFS could potentially succeed if it picks the correct branch and traverses to the goal early enough.

9. A*'s performance will be identical (and therefore A* is not an improvement) if there is only one path to the goal and therefore only one path cost available for the map.

10.

• h(n) = 2 for all n
Non-admissible and therefore not useful. Not admissible since unless the map has only commas, the cost will be overestimated. Not useful since giving the same estimated h(n) will not help in determining which path to take, and therefore, this is just as bad as random.

• h(n) = 0 if n is the goal, 1 otherwise
Admissible since this heuristic will never over-estimate a path length since the lowest possible path cost will be a path of periods (all of cost 1). Not useful since this does not differentiate between the viability of different paths, and like the first example, will not be better than a random heuristic.

• h(n) = Euclidean distance from current node to goal node
Admissible since this heuristic will never over-estimate the path length. Since we can only move in the cardinal directions in this problem, the lowest path cost would be the Manhattan distance. Euclidean distance is diagonal to the goal, and the hypotenuse of a triangle will always be less than its leg sizes, so the Euclidean distance will always be less than the Manhattan distance.
This is not that useful since the cost will always be underestimated, but the underestimation will always be proportionate to the Manhattan distance and therefore this heuristic is most useful when there are not many obstructions.

• h(n) = Twice the Euclidean distance from current node to goal node
Non-admissible since if we consider the following example:
Start node at (0,0) and goal node at (3,3).
The Euclidean distance is sqrt(18).
The minimum distance (the Manhattan distance due to only being able to move in cardinal directions) is 6 (3 right, 3 down).
$6<2\sqrt{18}$
Therefore, it's not useful.

• h(n) = Manhattan distance from current node to goal node
Admissible since the Manhattan distance is the minimum cost from the start node to the goal node (In this way, we assume that the path is made entirely of periods). Without obstacles, agents are required to move in cardinal directions, and therefore, to reach any other node, an agent must move in the x distance between the goal and current node plus the y distance between the current node and goal.
This heuristic is useful if there are not many obstructions to prevent the path from being viable. Otherwise, it may cause the program to choose poor paths.

• h(n) = One half the Manhattan distance from current node to goal node
Admissible since this heuristic will always underestimate the cost to the goal node.
Due to the fact that this heuristic will always underestimate, it may not provide useful information as to what's the next most viable path (but the results will be proportional to the Manhattan distance, which is most useful without many obstructions).