CS 471, Intro to Artificial Intelligence
Fall 2022
Jose Renteria

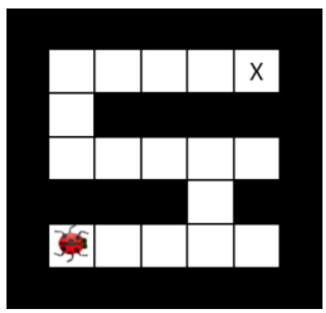
Written Assignment 1

Due Wednesday, October 12, 2022

Q1. Hive Minds

1.1. Lonely Bug

Start state: first node, bottom left corner
Goal test: Reach the destination node X in the top right

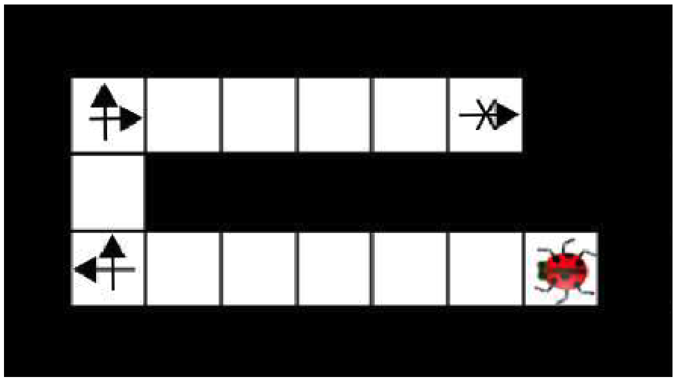


We can provide a tuple (x,y) which encodes the x and y coordinates of the insect. The position tuple is enough information for us to be able to determine whether we have reached the goal test. If our tuple (x,y) is equal to the position of our destination node, then we know we have succeeded.

The size of the state space is M * N because there are M*N total values that the position (x,y) can take.

1.2. Jumping Bug

Start state: first node, bottom left corner
Goal test: Reach the destination node X in the top right



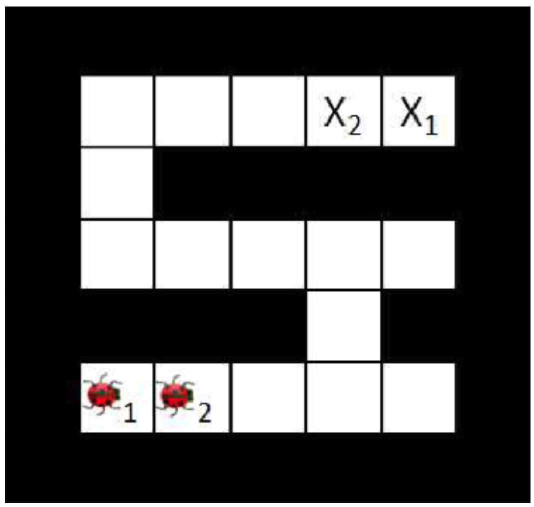
We can represent the minimal state with a tuple (x,y) that represents the position of the insect on our board, plus the direction that the insect is facing since it now takes the insect a time step to change the direction it is facing.

The size of the state space is now 4(M * N), since the insect can face four different directions and be charged a time step for each turn.

1.3. Swarm Movement

Start state: first two nodes, 1 and 2 in adjacent positions in the bottom left. We have a unique start state for each insect, with different positions.

Goal test: two unique destination nodes, X_1 and X_2 in adjacent positions at the top right of the graph. They each have different positions on our board.



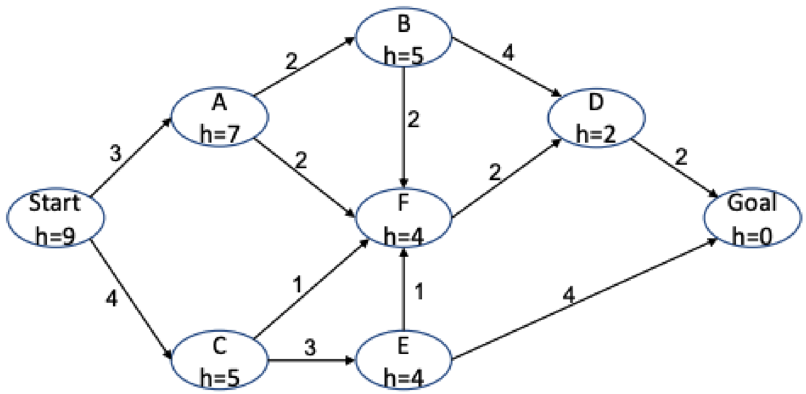
In order to represent a minimal correct state space representation, we must provide K tuples $((x_1, y_1), (x_2, y_2), \ldots, (x_K, y_K))$ where each tuple is a pair of ordered pairs from our starting position to any position on the board K. The x and y coordinates will still be representing the position of the insect on the board, but the tuple will be ordered. This means the first set of coordinates represents the position of insect 1, and the second set represents the position of insect 2.

The size of the state space is now $(M * N)^K$, which is the size of our board raised to the power of K. This will account for every possible combination of the two insects for any K number of tuples on our board.

Q2. Graph Search

2.1. Search Algorithms

For the following questions I will refer back to this graph.



In this graph, the start state is our start node *S* and the goal state is the rightmost node which we will call *G*.

1. Assuming that DFS will terminate as soon as it reaches the goal state, the order states are expanded by a Depth-First Search from our starting vertex will be:
    a. *S -> A -> B -> F -> D -> G -> C -> E.* The DFS will reach all vertices in the graph in this order.
    b. The path will be

        S -> A -> B -> F -> D -> G.

        Note: DFS will prefer to expand to *B* from *A* as opposed to *F* since ties are broken in alphabetical order.
2. The order of states expanded using Breadth-First Search will be:
    a. *S -> A -> C -> B -> F -> E -> D -> G*. The BFS will reach all vertices in the graph in this order.
    b. The path will be:

        S -> C -> E -> G

        Note: This will take the same cost as DFS, it will just expand to one less node.

3. Uniform-cost orders by path cost, or backward cost *g(n)*. The order of states expanded using Uniform-Cost Search will be:
    a. *S -> A -> B -> F -> D -> G -> C -> E*. The UCS will reach all vertices in the graph in this order.
    b. The path will be:

        S -> A -> F -> D -> G.

4. Greedy orders by goal proximity, or forward cost *h(n)*, and thus the order of states expanded will be:
    a. *S -> C -> E -> G -> F -> D -> A -> B*. The greedy search will expand to all nodes using only their heuristic value in this order.
    b. The path will be:

        S -> C -> E -> G

        Note: It just so happens this is the same path as BFS.

5. The A* search combines UCS and Greedy and orders by the sum: *f(n) = g(n) + h(n)* where *g(n)* is our path cost and *h(n)* is our heuristic or goal proximity. Thus the order of states expanded using A* Search will be:
    a. *S -> C -> A -> F -> E -> B -> G -> D*

    We can represent the results of our function using an adjacency matrix. Note: If there are multiple paths to a single vertex in our graph, I simply took the *MIN(f(u), f(v))*.

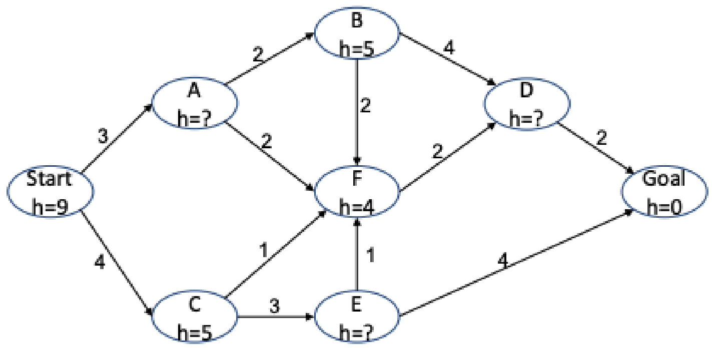
Node	*h(n)*	*g(n)*	*f(n) = h(n) + g(n)*
S	9	0	9
A	16	3	19
B	21	5	26
C	14	4	18
D	22	7	29
E	18	7	25
F	18	5	23
G	18	9	27

    b. The path returned by our A* search will be:

        S -> C -> E -> G, since this has the least total path cost according to our function *f(n).*

2.2. Missing Heuristic Values

For the following questions I will refer back to this graph.



State	Range for $h(s)$
A	$7 \leq h(A) \leq 9$
D	$2 \leq h(D) \leq 6$
E	$4 \leq h(E) \leq 5$

We only need to verify that our heuristic is consistent. Since, by definition, a consistent heuristic is implied to also be admissible. A consistent heuristic is one that satisfies the property

$$h(s) \leq c(s, s') + h(s')$$

For our current search graph, this means that

For $s = A$:

$h(Start) \leq c(Start, A) + h(A) \Rightarrow h(A) \leq 9$

$h(A) \leq c(A, B) + h(B) \Rightarrow h(A) \geq 7$

$h(A) \leq c(A, F) + h(F) \Rightarrow h(A) \geq 6$ to be consistent.

For $s = D$:

$h(D) \leq c(D, G) + h(G) \Rightarrow h(A) \geq 2$

$h(B) \leq c(B, D) + h(B) \Rightarrow h(A) \leq 9$

$h(F) \leq c(F, D) + h(F) \Rightarrow h(A) \leq 6$ to be consistent.

For $s = E$:

$h(E) \leq c(E, G) + h(G) \Rightarrow h(A) \geq 4$

$h(B) \leq c(C, E) + h(C) \Rightarrow h(A) \leq 8$

$h(F) \leq c(F, E) + h(F) \Rightarrow h(A) \leq 5$ to be consistent.