Searching Algorithms

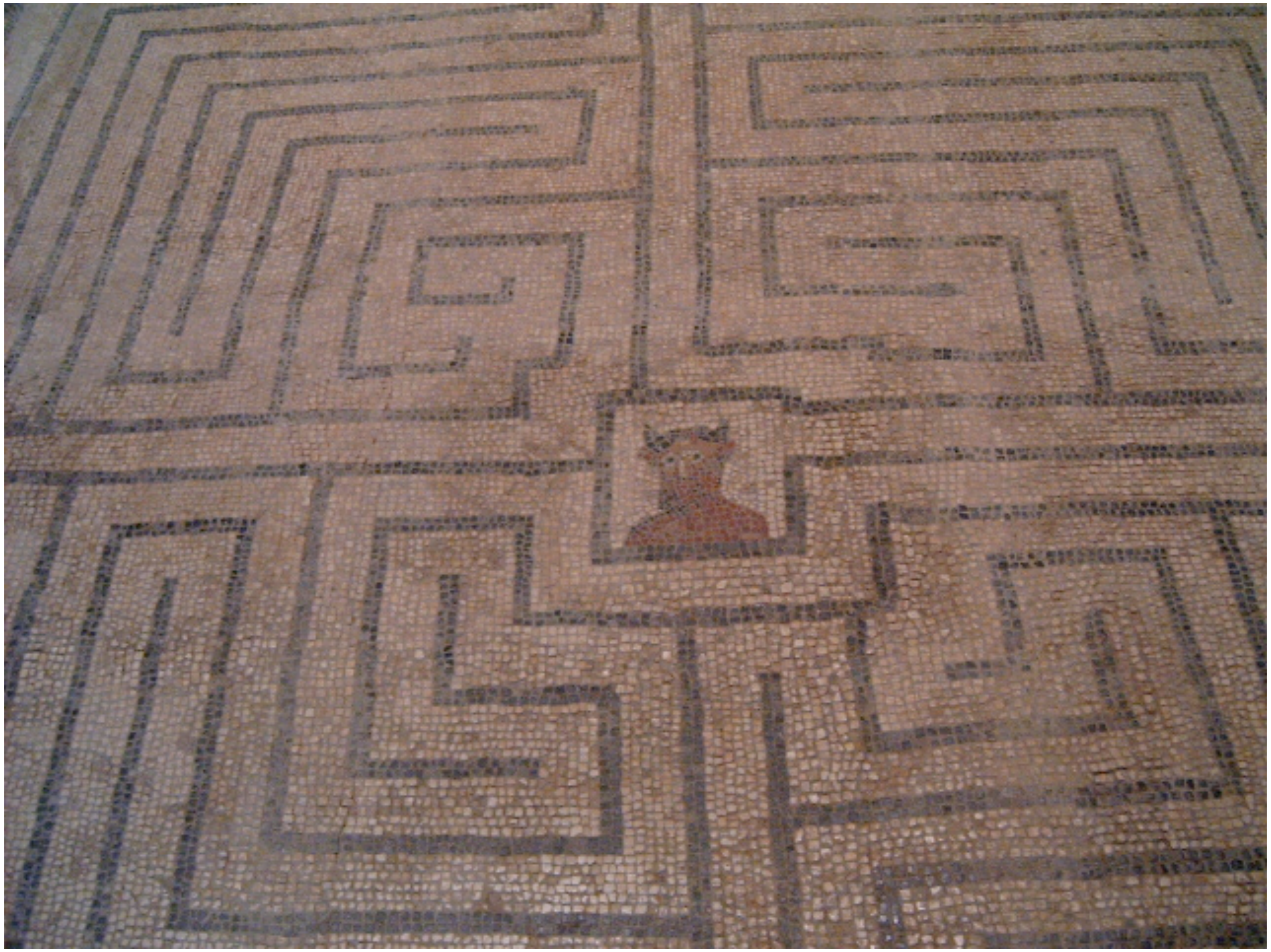
FIT3094 AI, A-Life and Virtual Environments
Alan Dorin

MONASH University
Information Technology

What were the difficulties of navigating the ancient labyrinths as discussed by Pliny?

Pliny the Elder, *Natural Histories*, Chapter 19(13)

A virtual environment designed to test your navigational abilities…



Task: Write an algorithm to solve a maze of the type illustrated above in which you must travel from an entrance to the Minotaur's den.

Learning Objectives

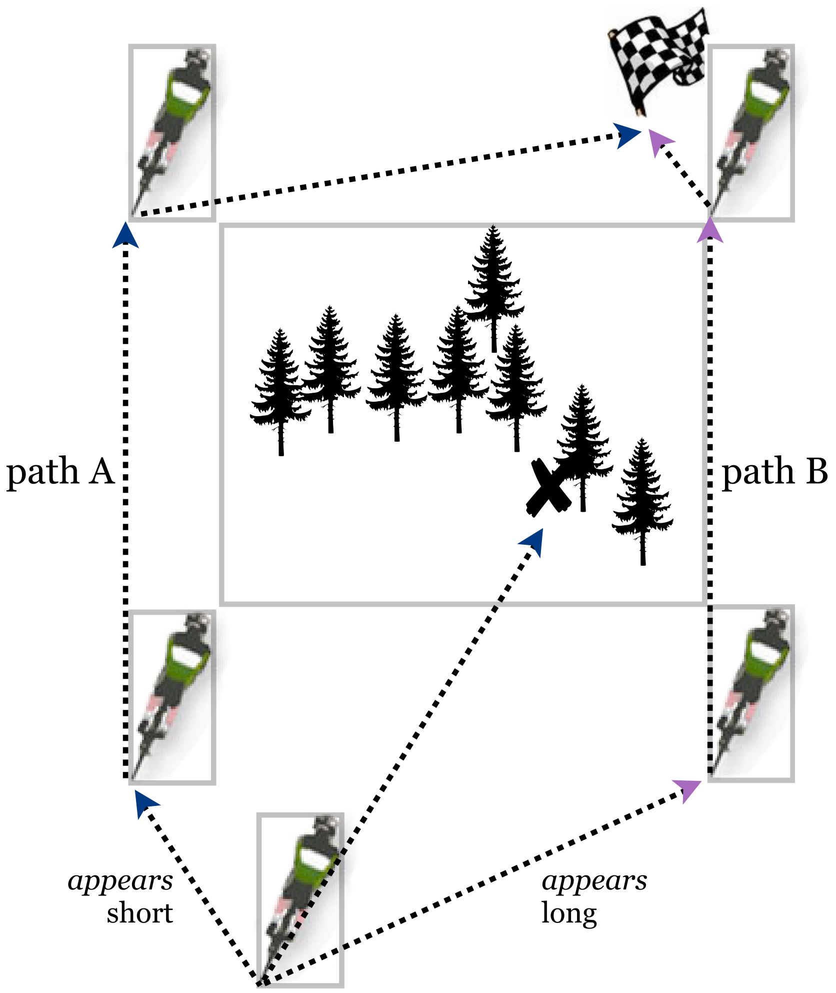
To know how to read an existing search tree or graph.

To know how to construct a search tree or graph for an agent exploring a space.

To understand how to apply depth first search and breadth first search.

To know how to convert a continuous search problem for an agent into a discrete problem.

How does an agent *choose* between multiple paths?



It may *search* the possible options to find the most satisfactory solution.

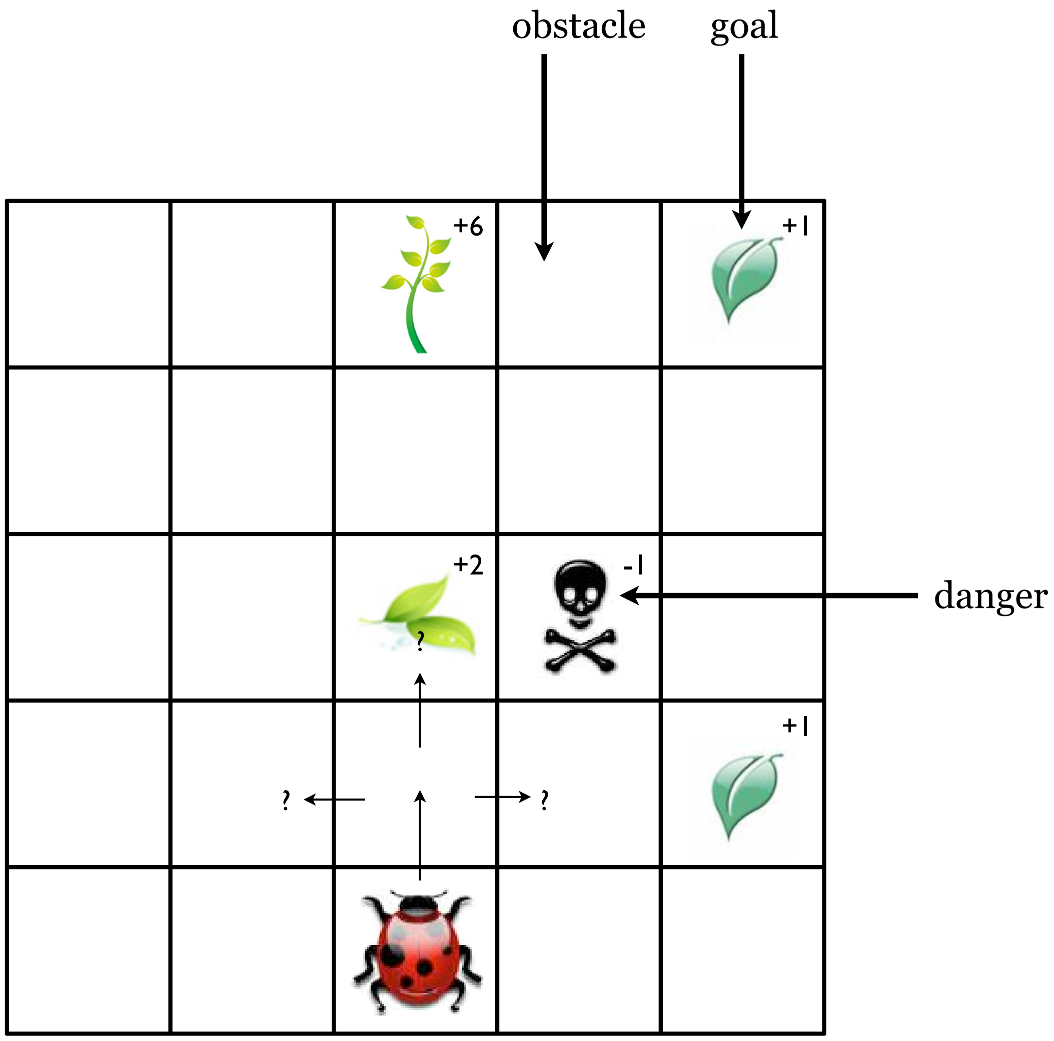
An agent's ability to search is governed by:

Its knowledge of the world
Can it see behind objects?
Can it anticipate changes in the environment?

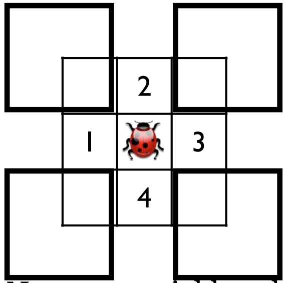
Its ability to reason
Can it accurately evaluate alternatives?
How many options can it evaluate within the available time?

A discrete environment

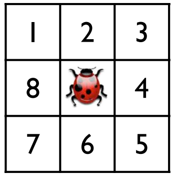
Here is a leaf-seeking lady-bird in a discrete environment with different objects and obstacles.



The lady-bird (agent) can move into any unblocked, neighbouring square, *not* including diagonals. This is its *von Neumann neighbourhood*.



von Neumann neighbourhood



Moore neighbourhood

Depending on the circumstances, the lady-bird might need to know the optimum path through the world, or possibly just to one of the available goals from its starting position

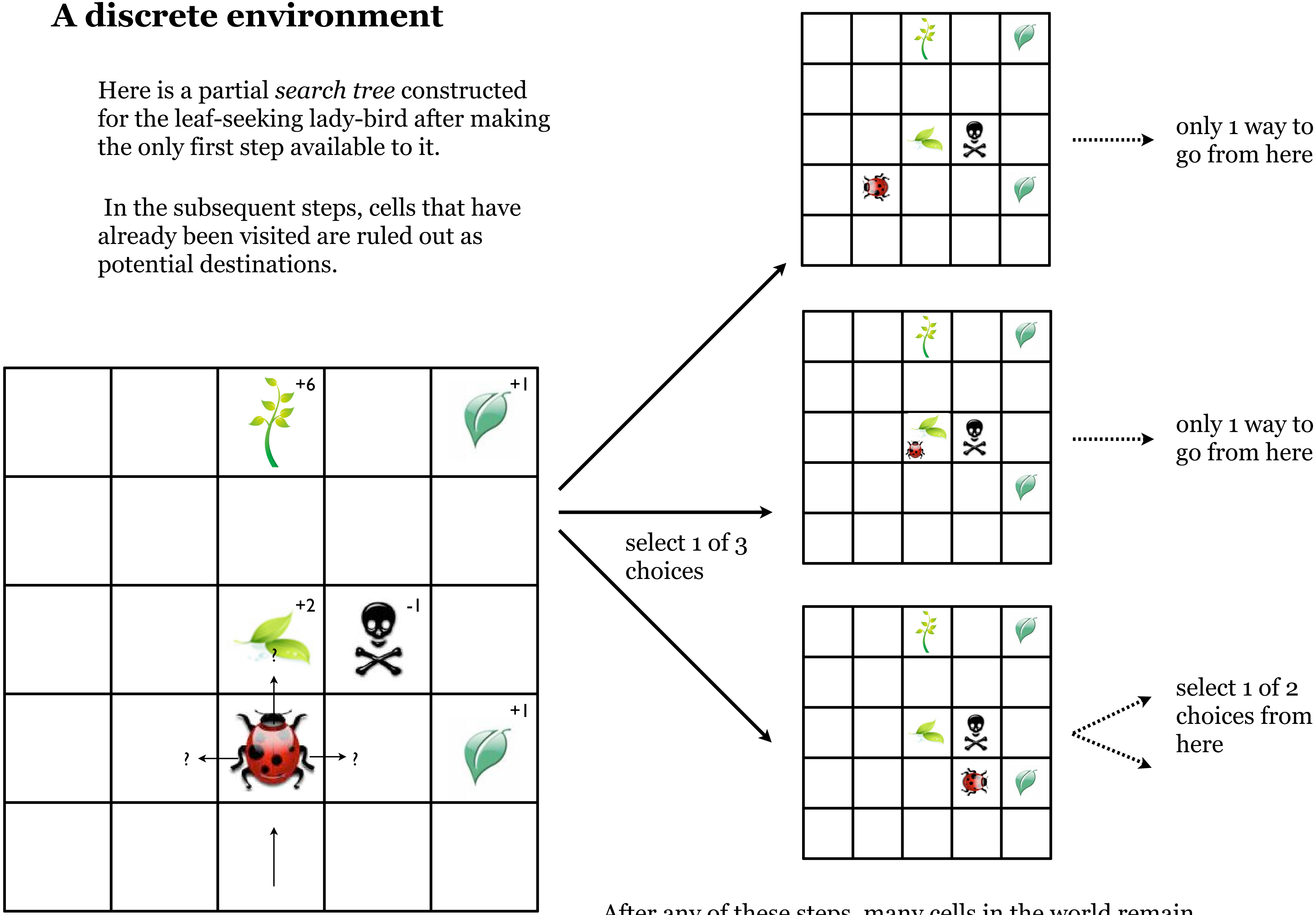
We'll assume that the ladybird keeps track of the cells she has visited to avoid returning to them.

How do we find a path to a goal so that the lady-bird can follow it?

A discrete environment

Here is a partial *search tree* constructed for the leaf-seeking lady-bird after making the only first step available to it.

In the subsequent steps, cells that have already been visited are ruled out as potential destinations.



select 1 of 3 choices

only 1 way to go from here

only 1 way to go from here

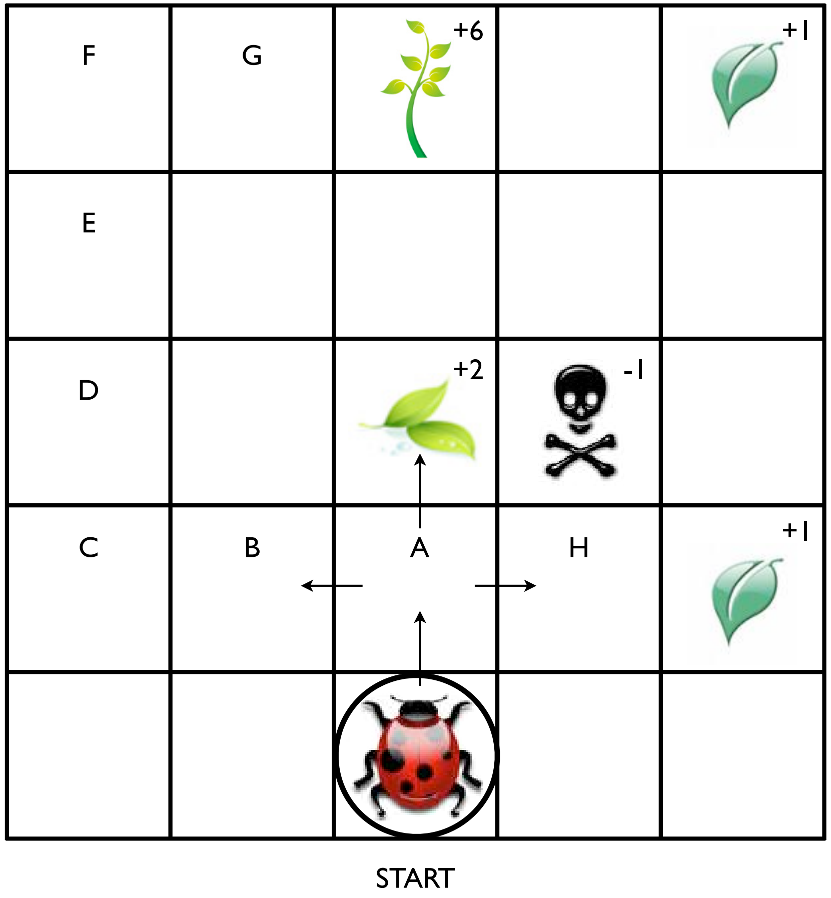
select 1 of 2 choices from here

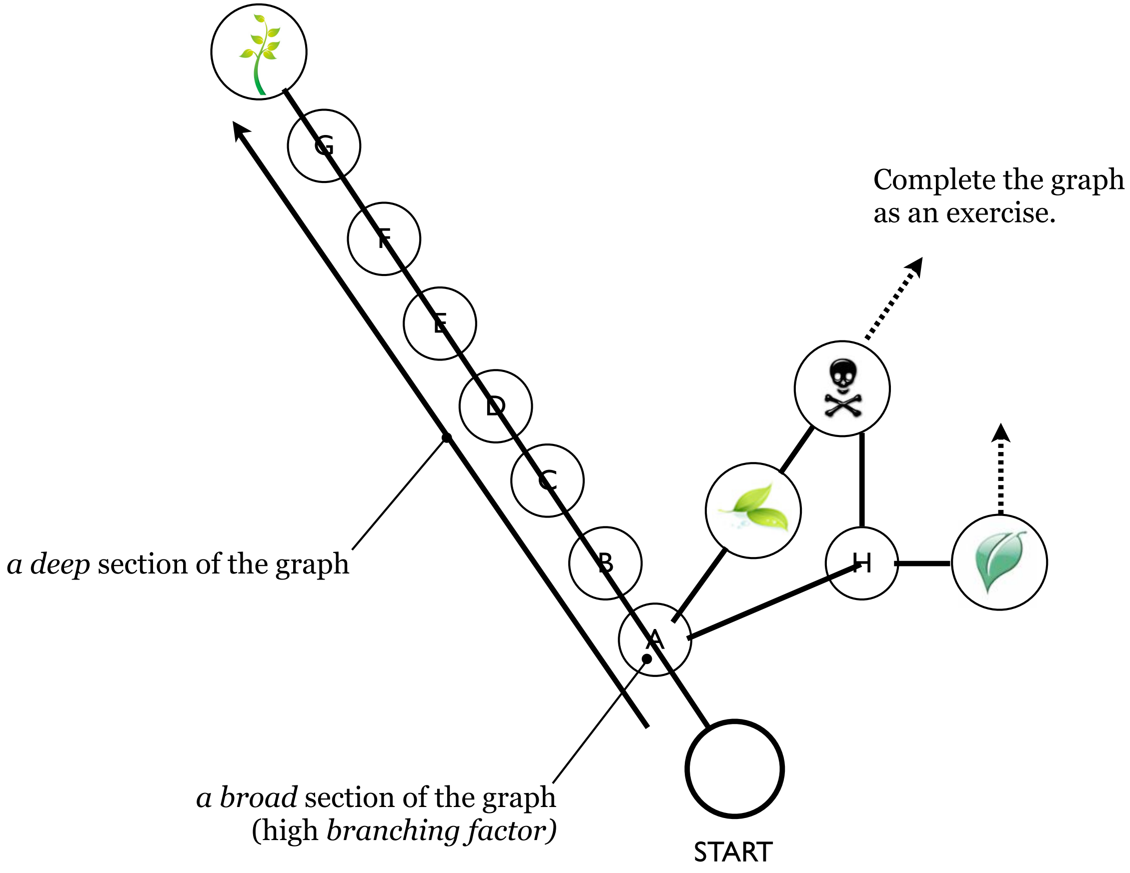
After any of these steps, many cells in the world remain unexplored. We need to search more than one step ahead! How?

A discrete environment

Here is some more detail of the *search graph* representing the paths the lady-bird *could* take.

This is a *graph* rather than a *tree* because it has cycles (e.g. through A, H, skull, double-leaf and back to A). Trees have no cycles.





Complete the graph as an exercise.

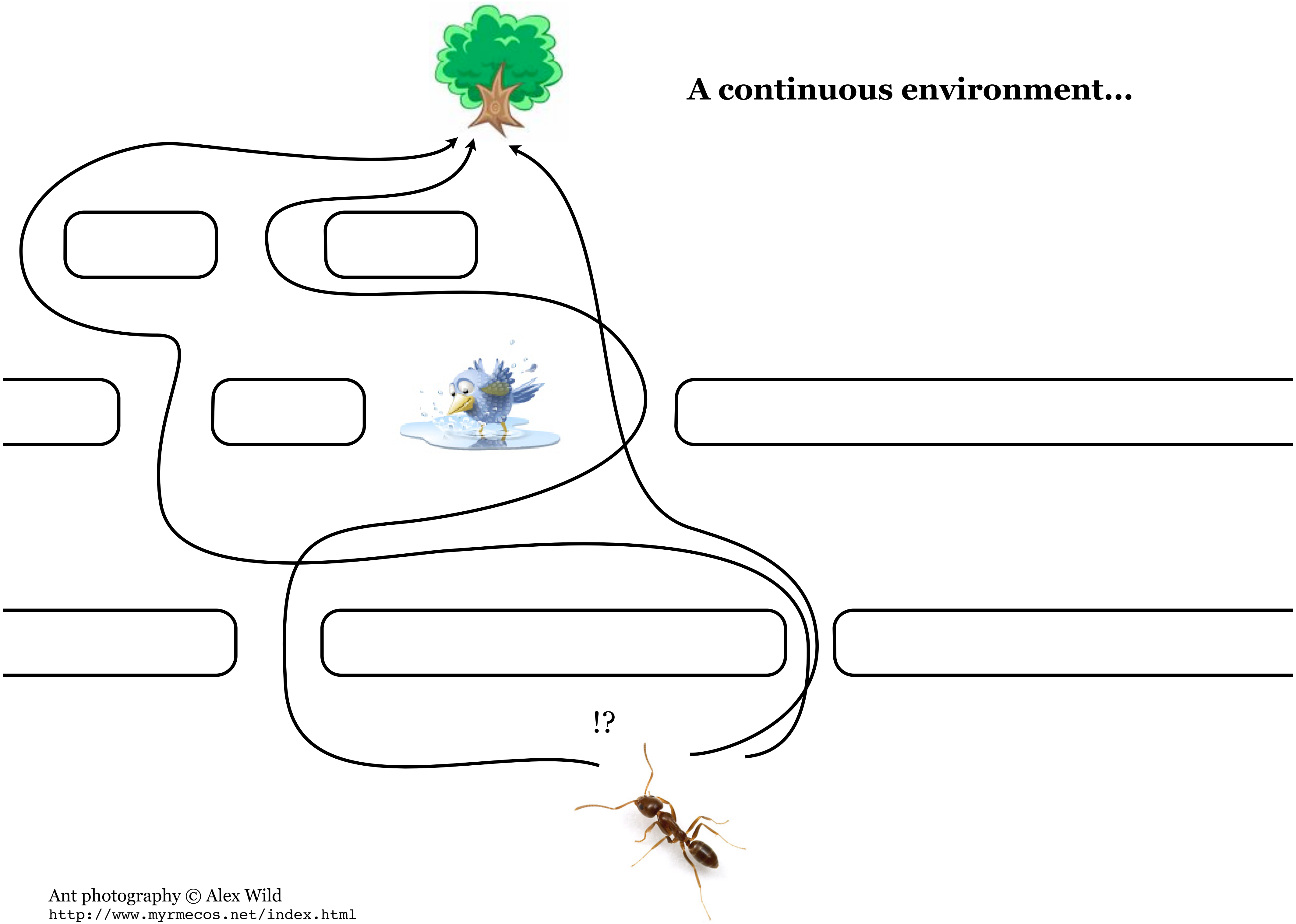
a deep section of the graph

a broad section of the graph (high *branching factor)*

How many paths are there from START to ☠ ?

Does this graphical representation make it easy to tell?

A continuous environment...

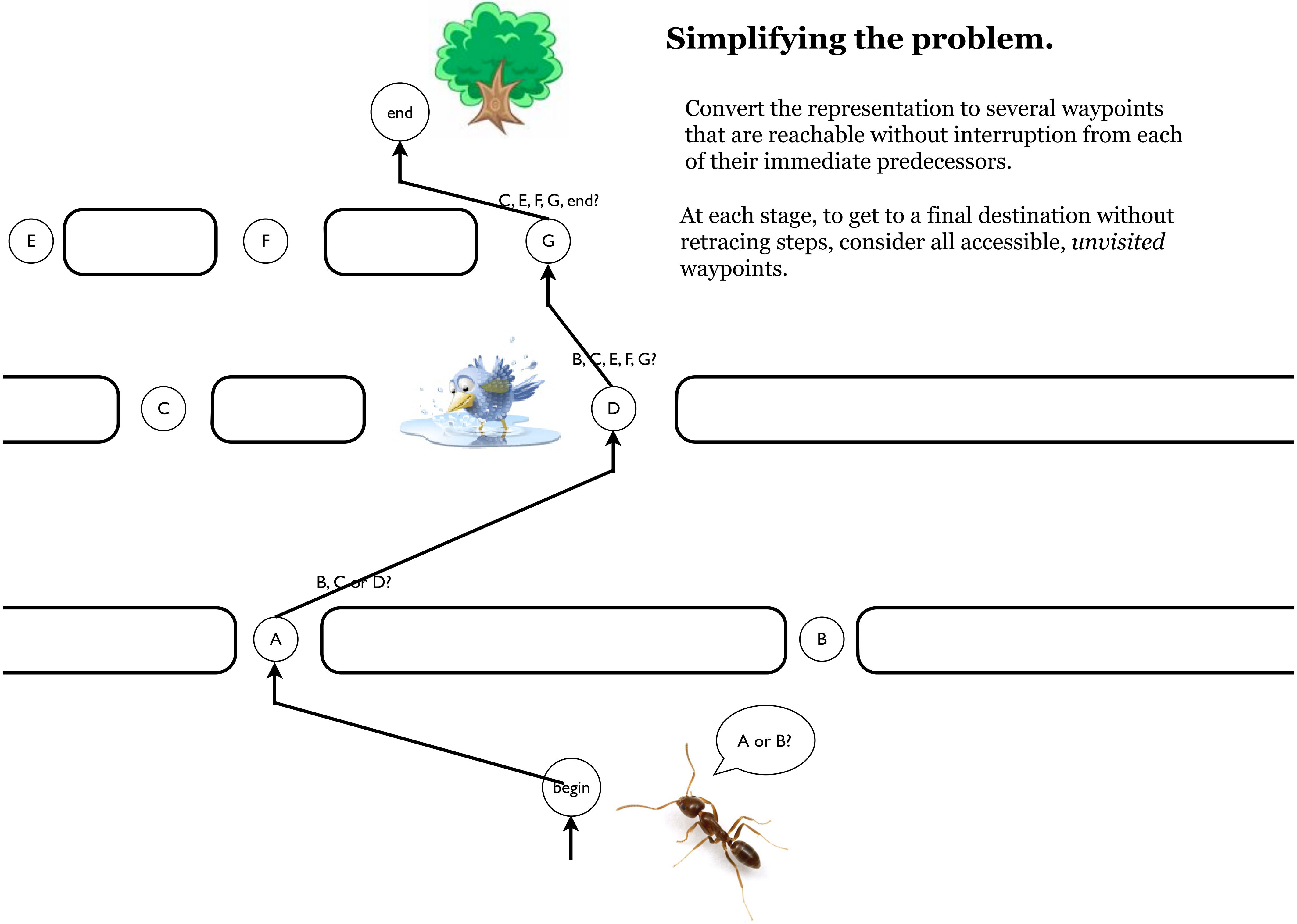


Ant photography © Alex Wild
http://www.myrmecos.net/index.html

Simplifying the problem.

Convert the representation to several waypoints that are reachable without interruption from each of their immediate predecessors.

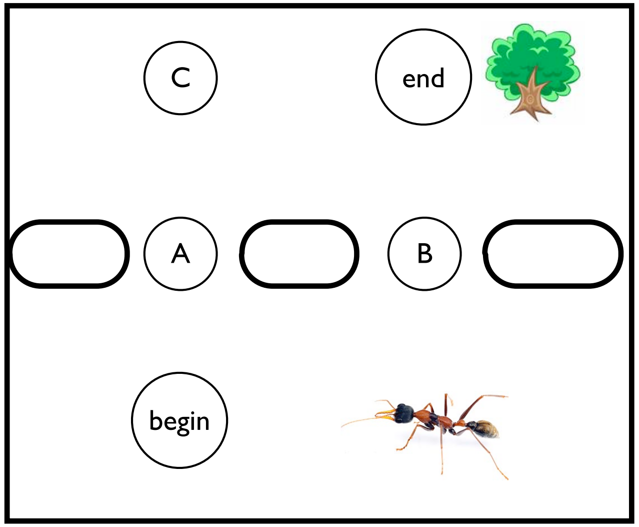
At each stage, to get to a final destination without retracing steps, consider all accessible, *unvisited* waypoints.

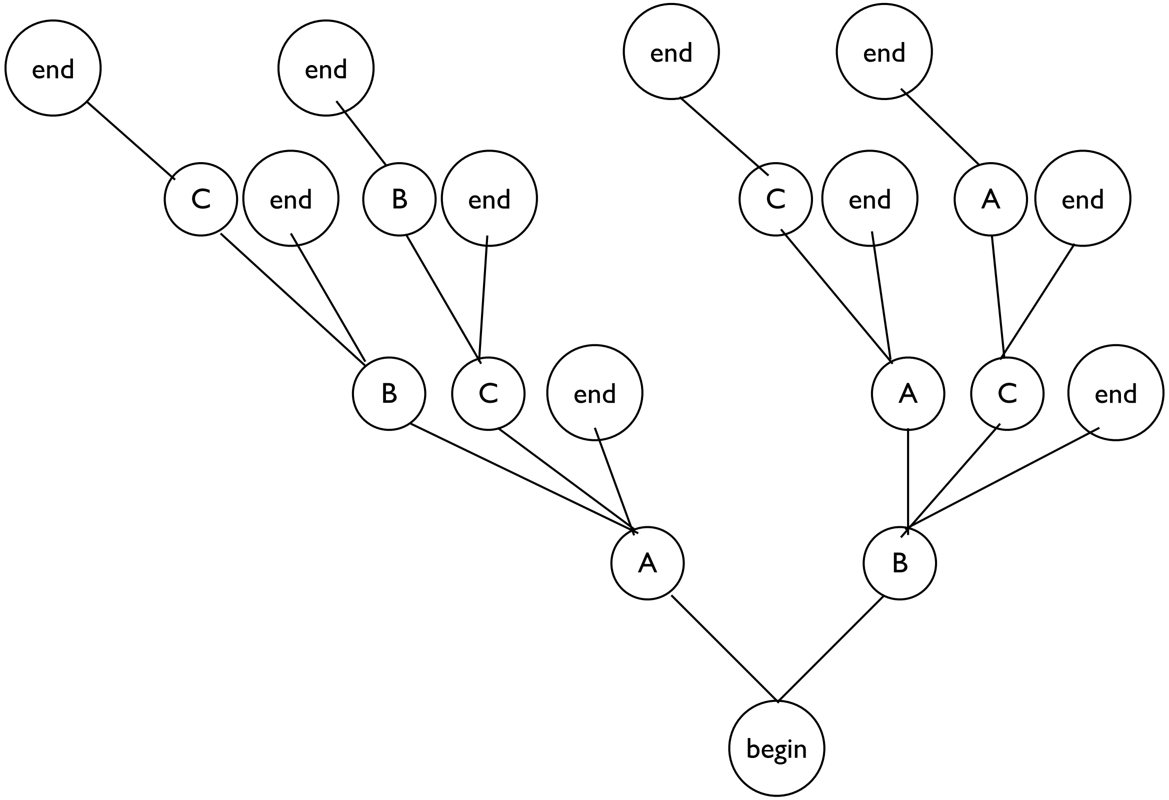


Building a search tree

How many paths are there from begin to end?

Does this graphical representation make it easy to tell?





This *tree* shows *all* the path options available to the agent in this simple example.

Note that in the tree there are no cycles. Each path here is set out as a sequence from begin to end.

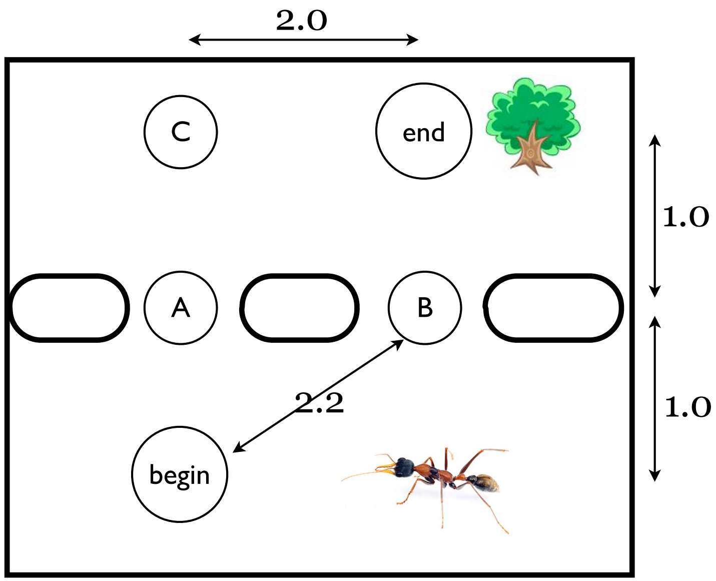
Associated with each section of the path is a distance* that the agent must travel...

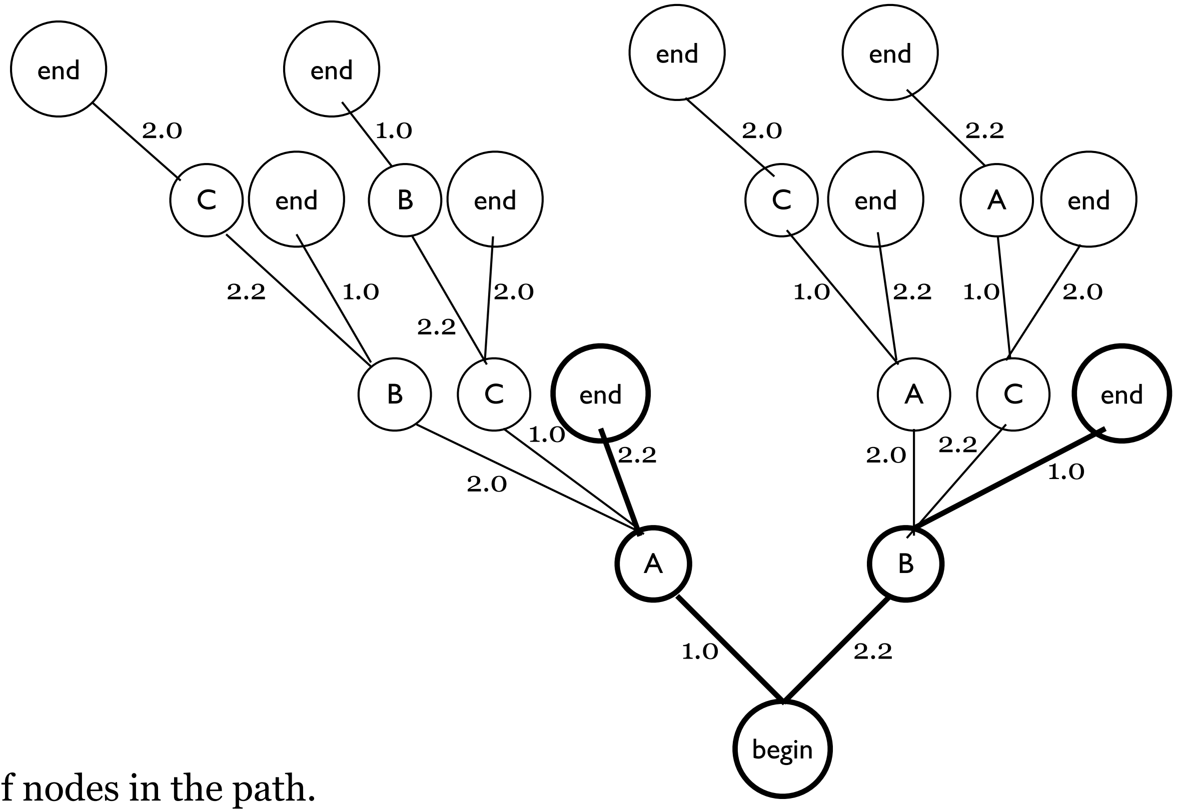
* Remember how to calculate the distance between two points?

Finding the shortest path.

A *brute force** algorithm examines the whole tree.

How many duplicate subtrees can you find in this diagram?





```
For every path in the tree
  Sum the distances between each pair of nodes in the path.
  If this path was the shortest so far, remember it.
```

The more waypoints there are, the bigger the tree and the longer it will take to search!

Is this feasible? It depends on how big the tree is.

* Check lecture 2a to remind yourself of Shannon's *Type A* Chess algorithm

What if… ?

Suppose we need to find a path to a known goal, but we are not sure:

(i) How long such a path might be
(ii) If there *is* a path!

All we can do is start searching the available options in a methodical way, i.e. our search *explores* the space.

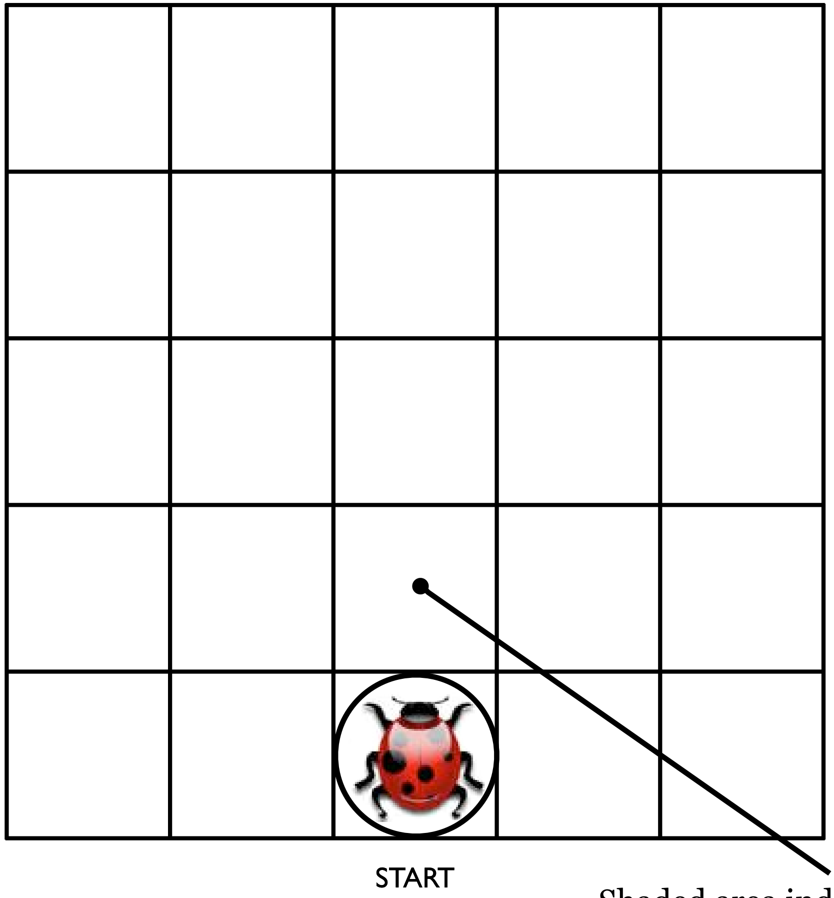
We can do this using a brute force approach… try all the options until we find the goal (whilst hoping that we won't run out of time in our search and that there *is* a path to the goal).

Depth first search and *Breadth first search* are two common, brute force, tree searching algorithms.

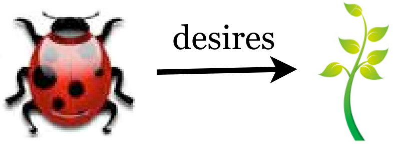
Depth first search

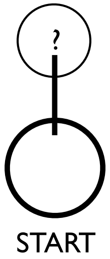
This strategy explores as far *into* the search tree as possible before turning around and going back to explore the last unexplored alternative path.

Think of exploring a cave by going deeper and deeper into the ground, always taking the first available option until a dead-end is reached. Then going back to the last available *unexplored* option and going deeper from there...

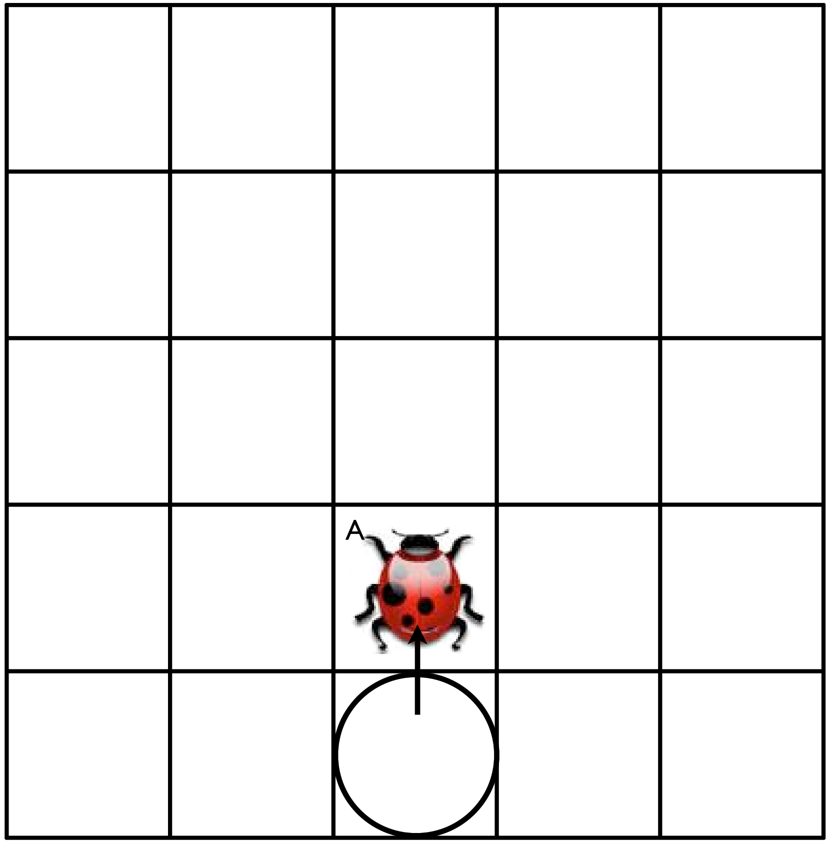


Shaded area indicates *available* paths that have been seen but not necessarily visited. Only when they have been visited does the agent know what lies there.

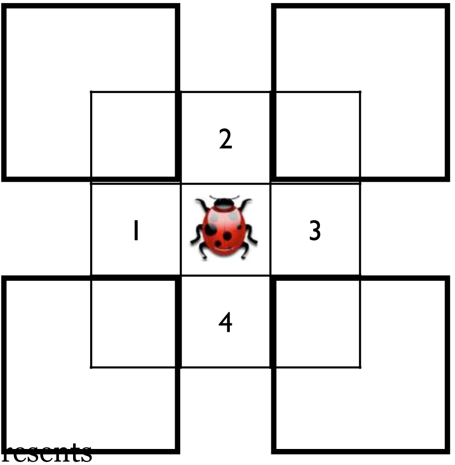




Depth first search



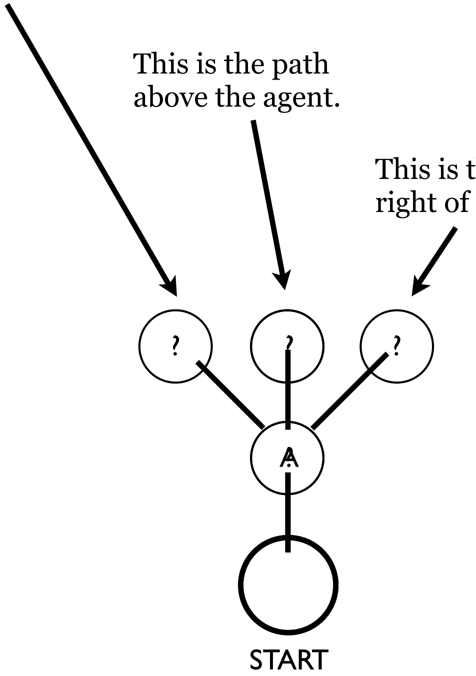
From cell A, the agent has 3 choices. For consistency, we will assume the nodes are ordered as they were in the diagram of the von Neumann neighbourhood...



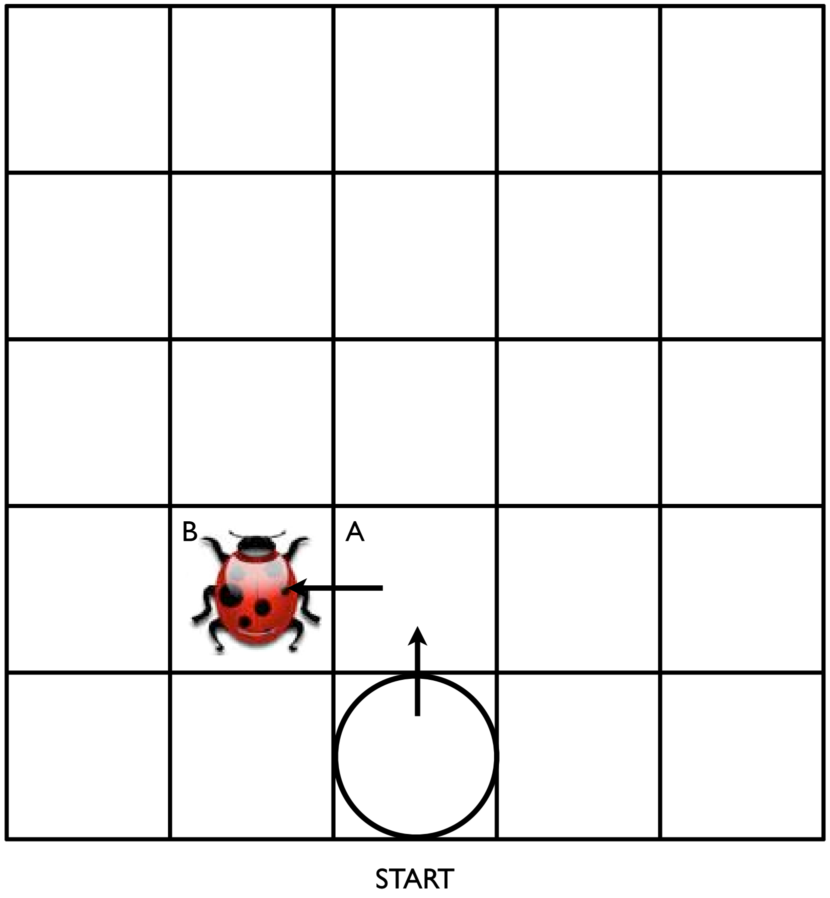
Hence, this node represents the path left of the agent.

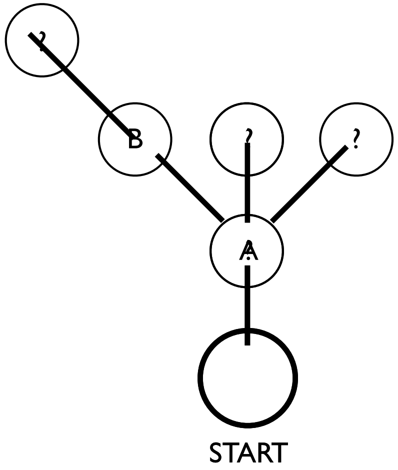
This is the path above the agent.

This is the path right of the agent.

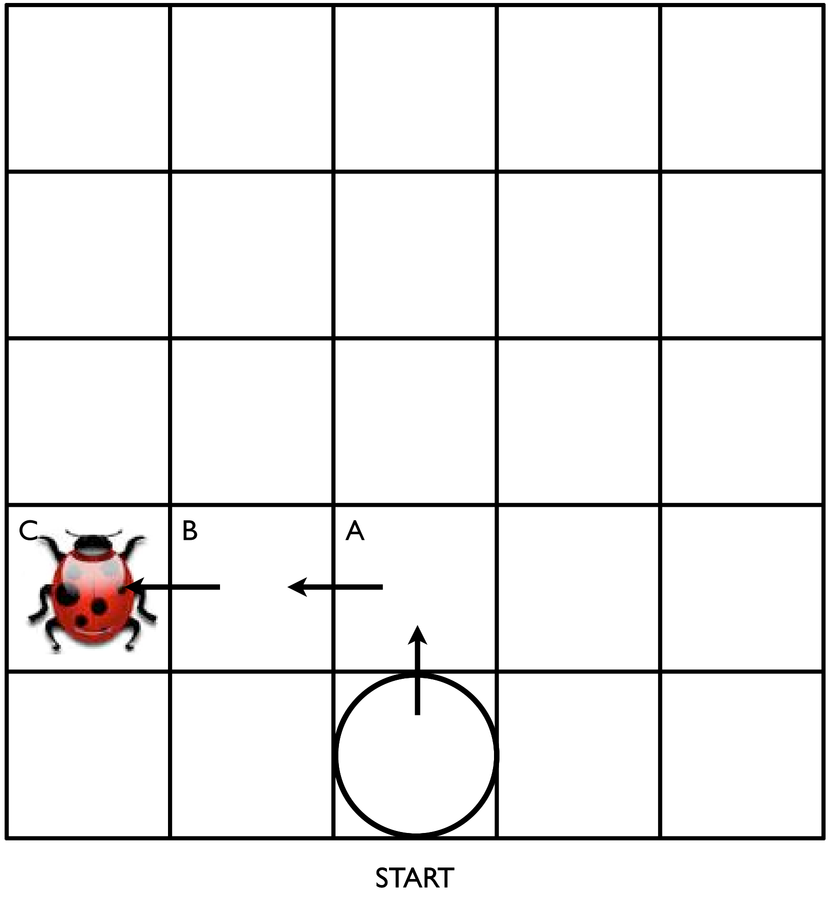


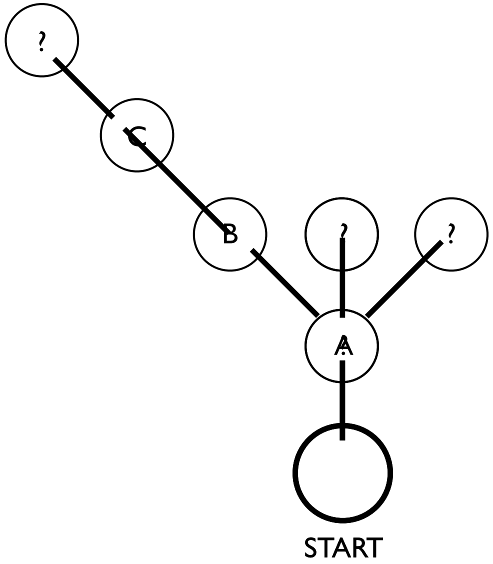
Depth first search



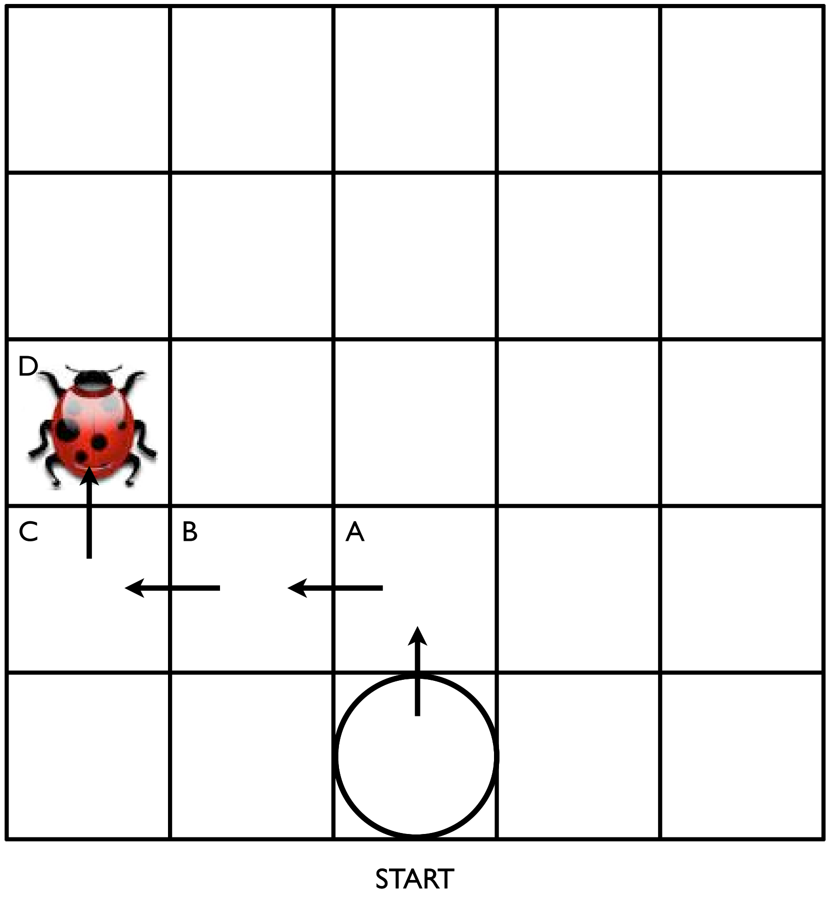


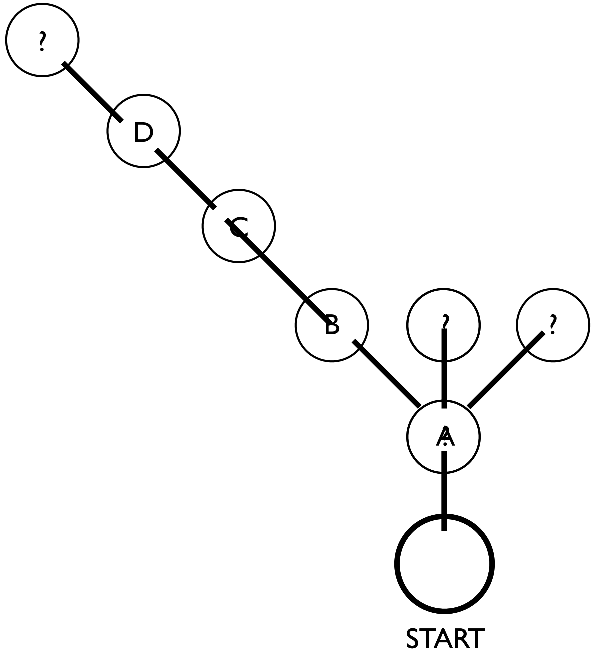
Depth first search



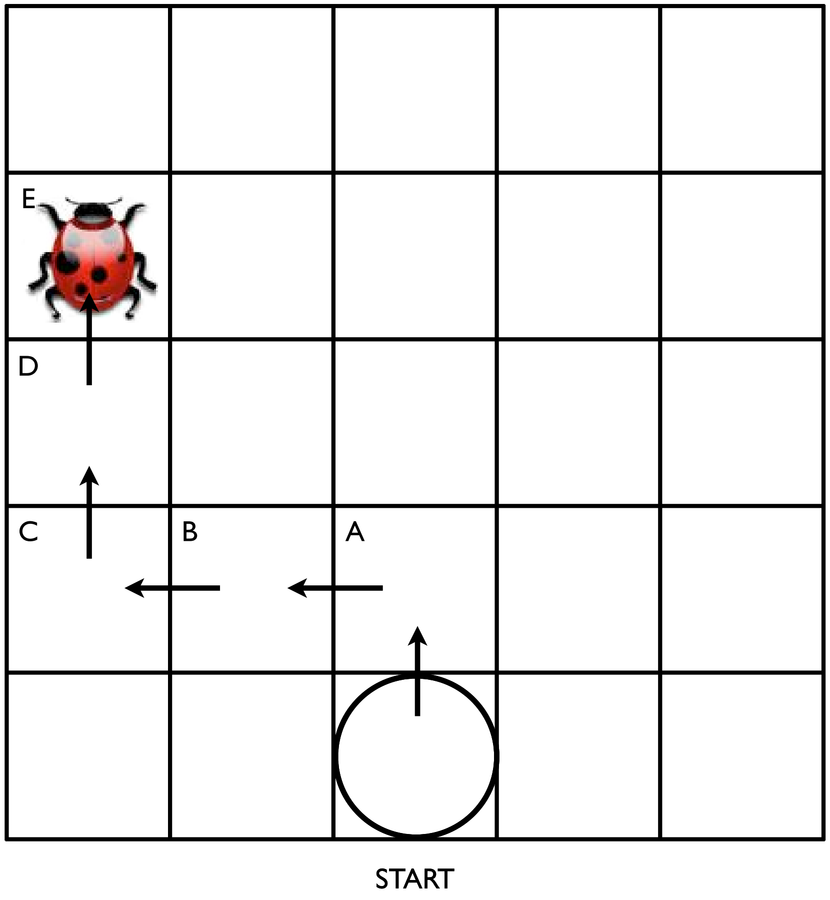


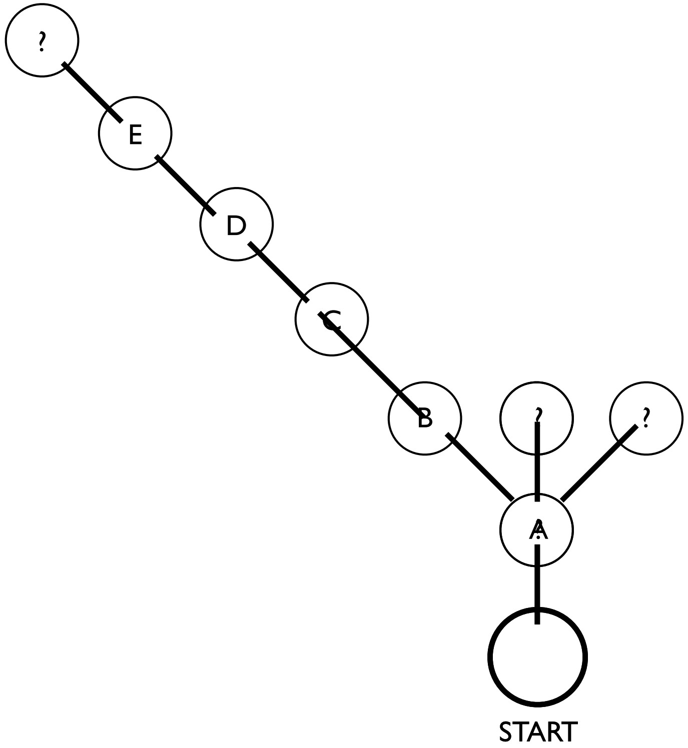
Depth first search



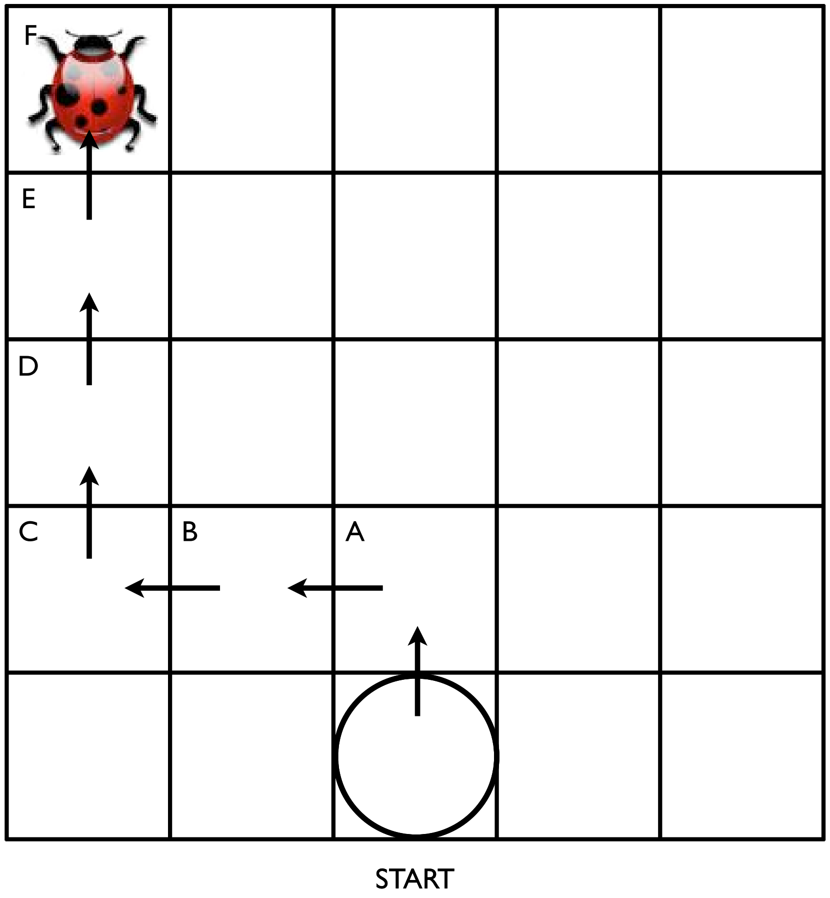


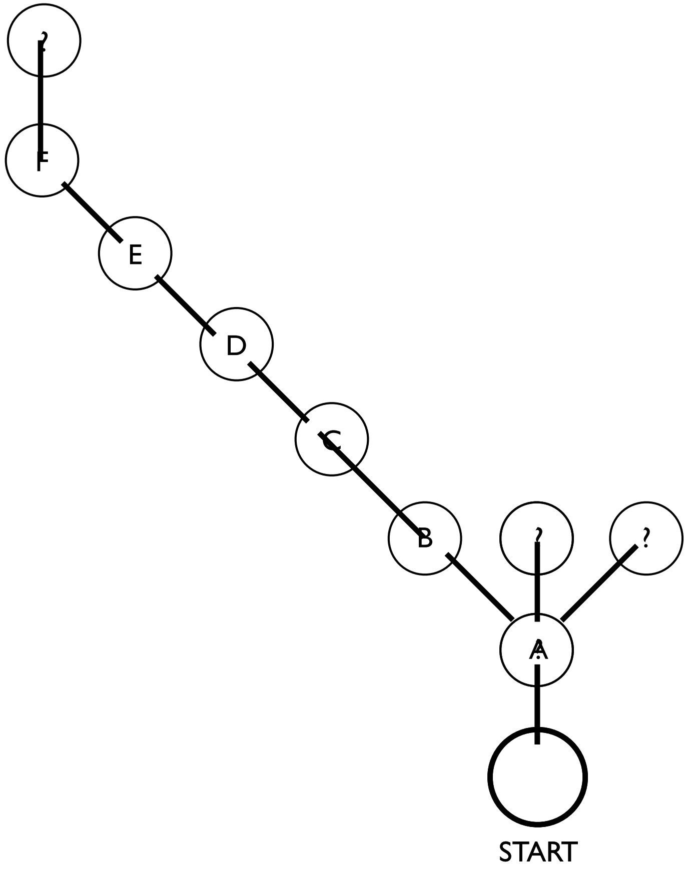
Depth first search



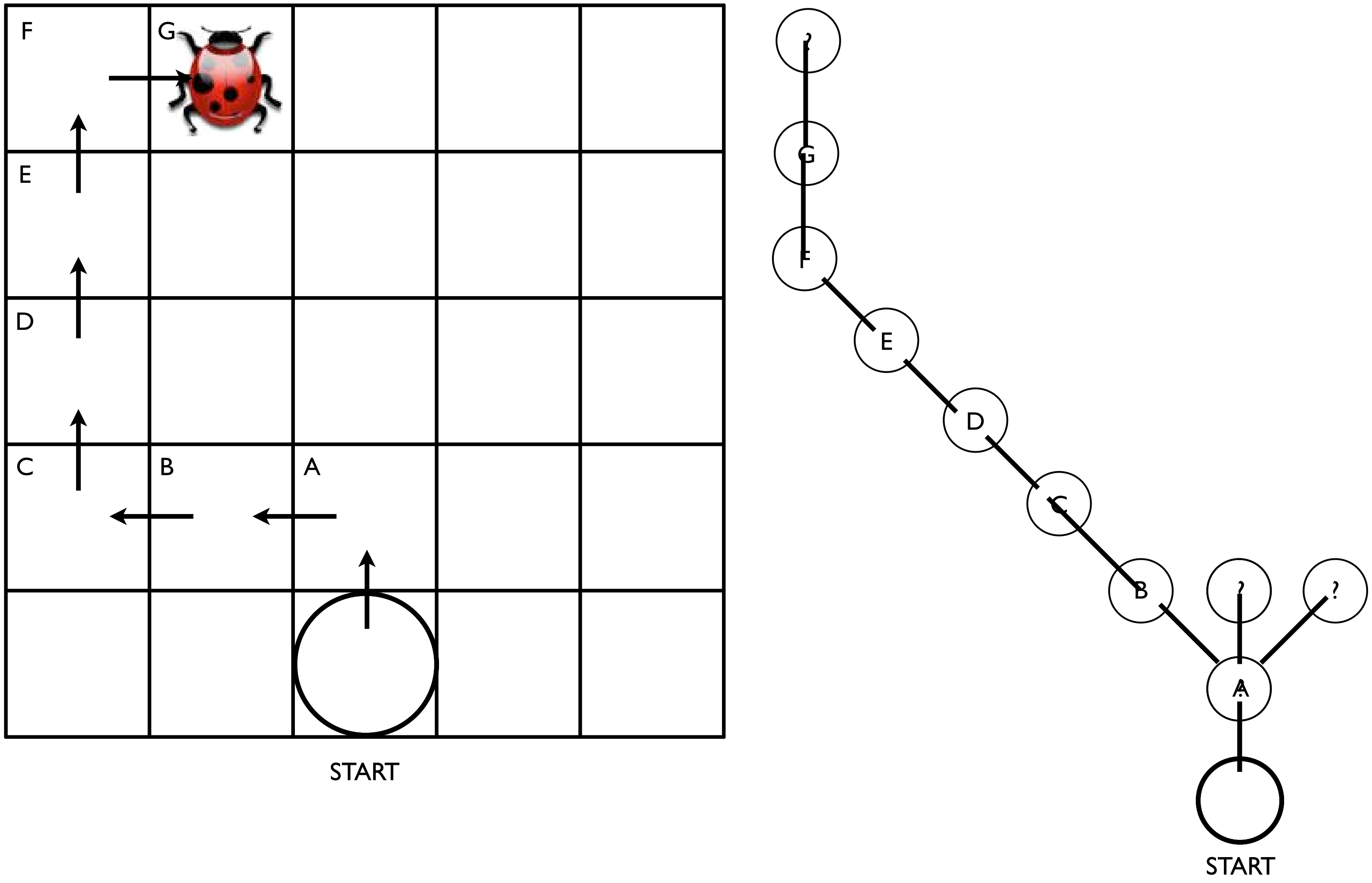


Depth first search

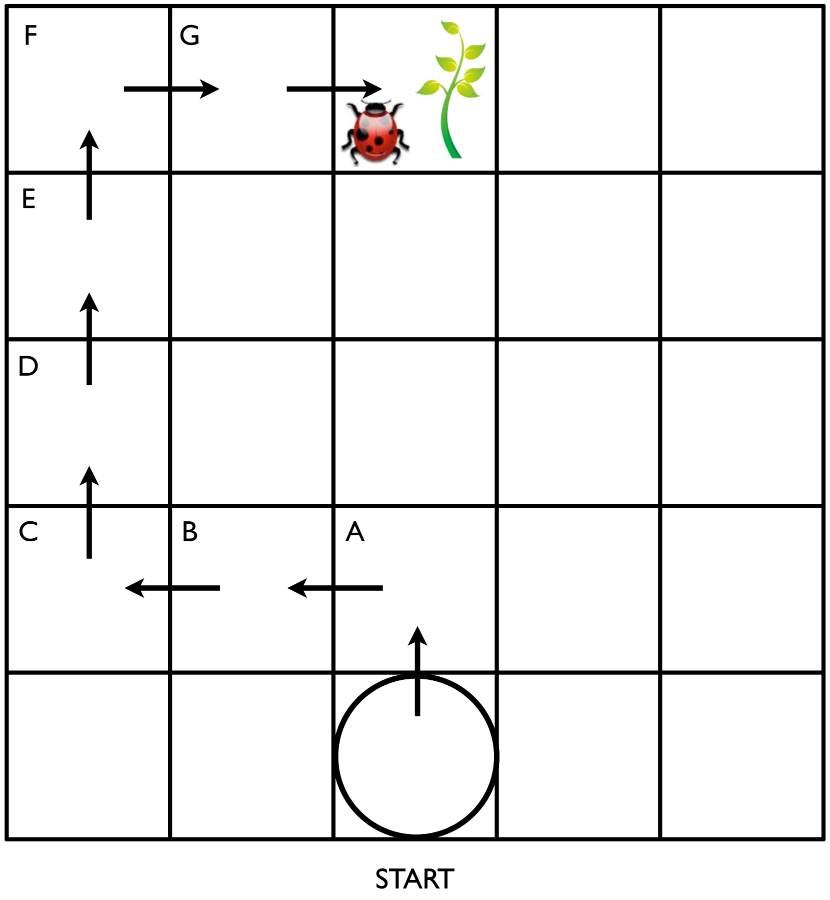


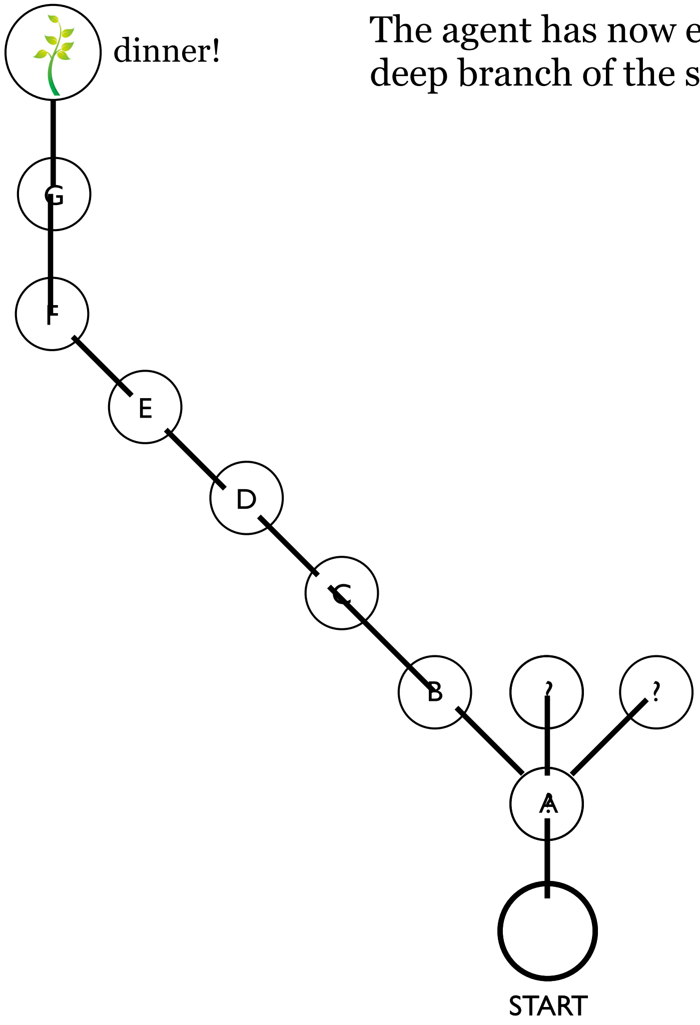


Depth first search



Depth first search

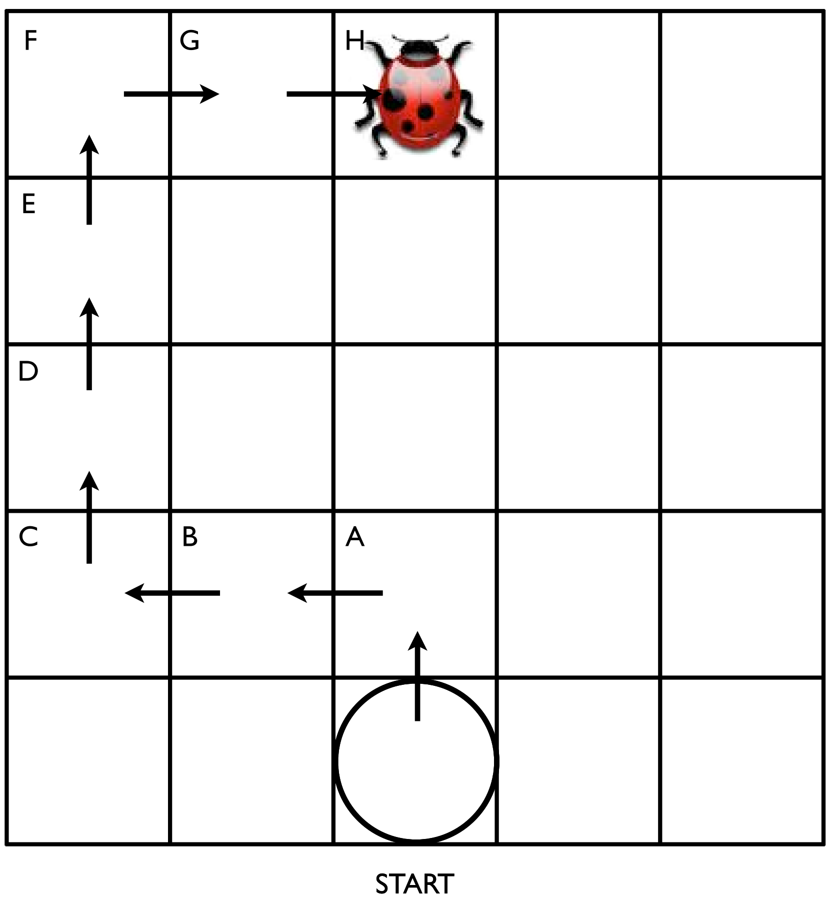


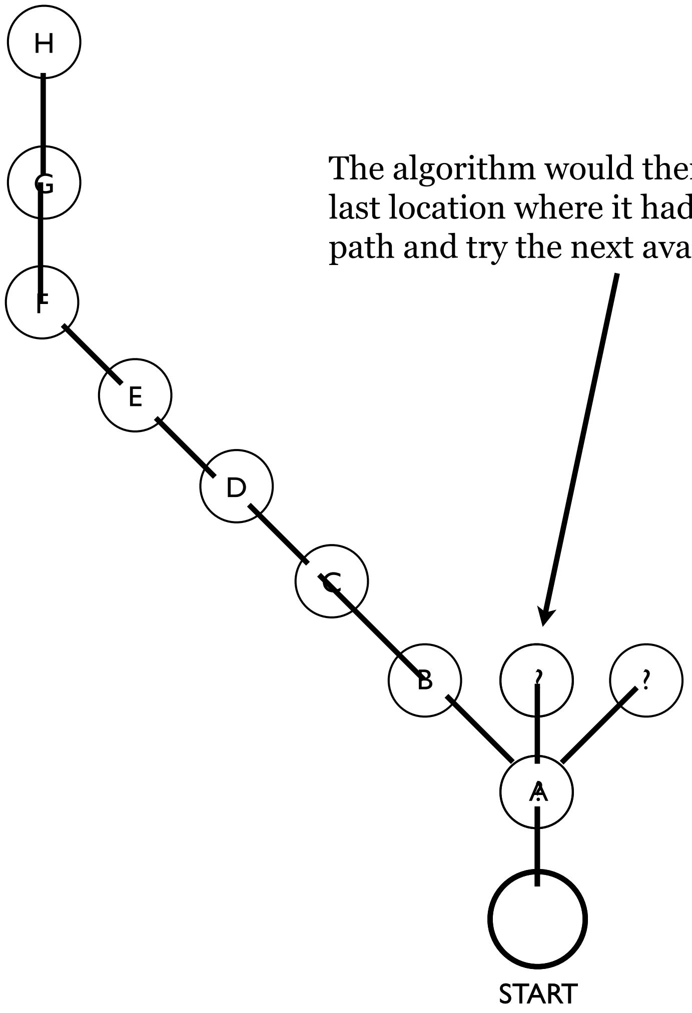


The agent has now explored one deep branch of the search space.

Depth first search *dead-end*!

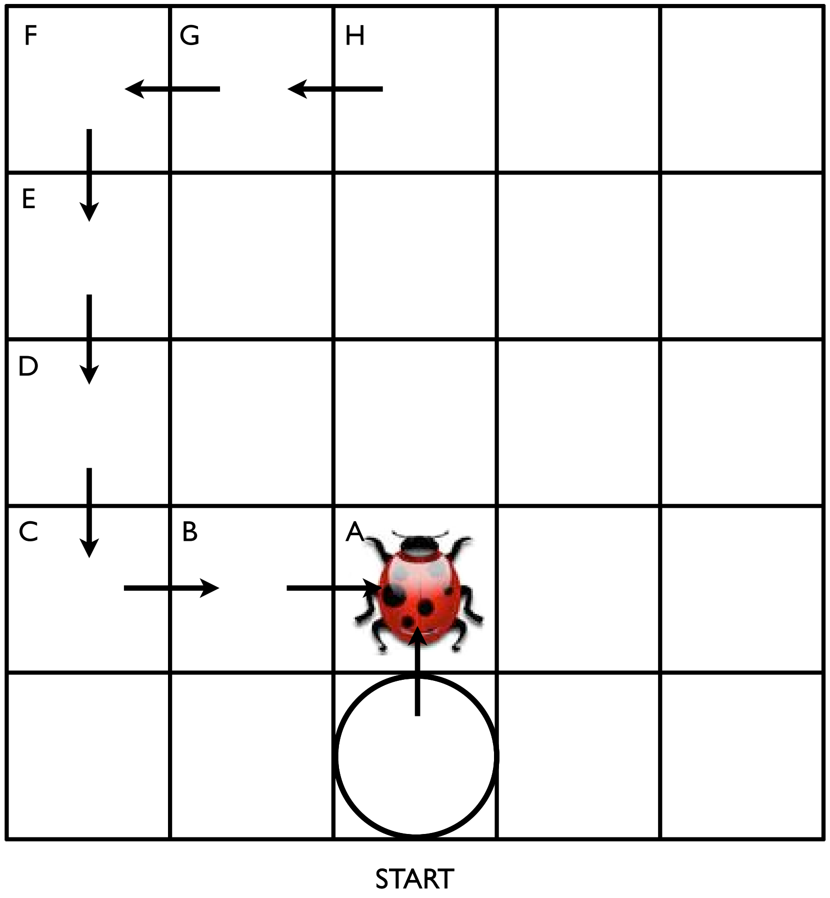
Suppose instead of a plant, the agent found a dead-end, H...

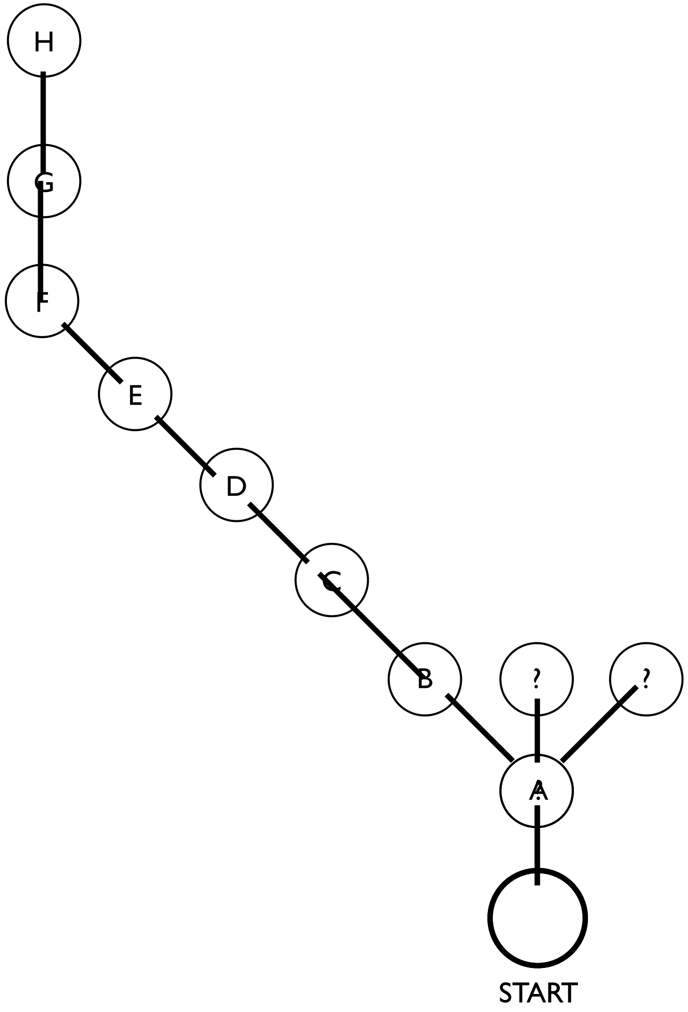




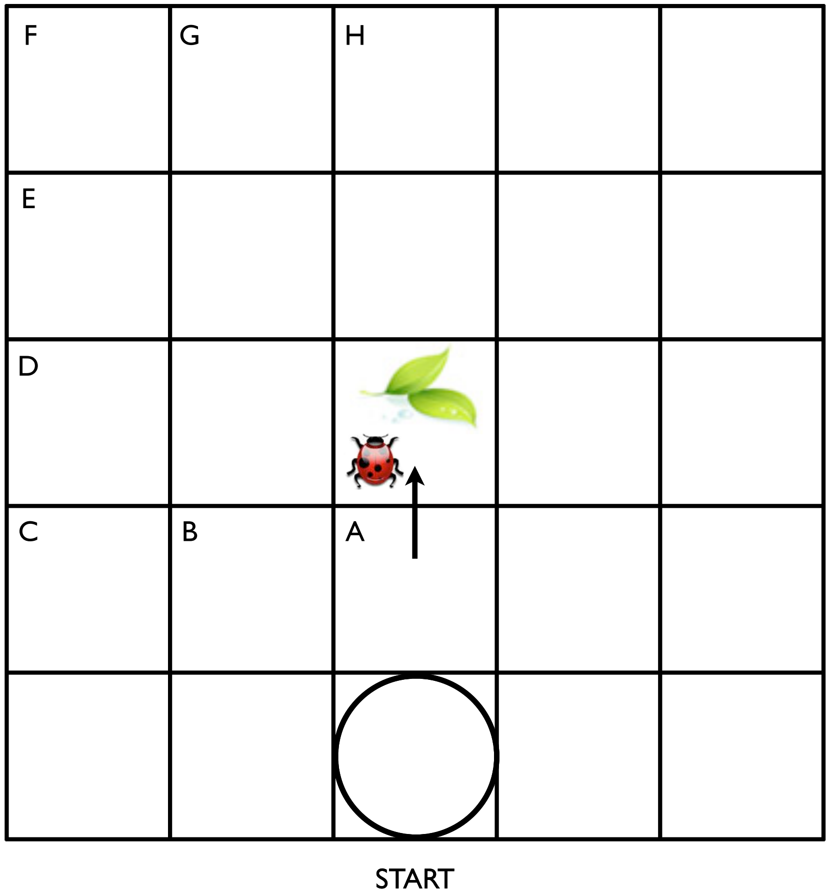
The algorithm would then “back up” to the last location where it had an unexplored path and try the next available option.

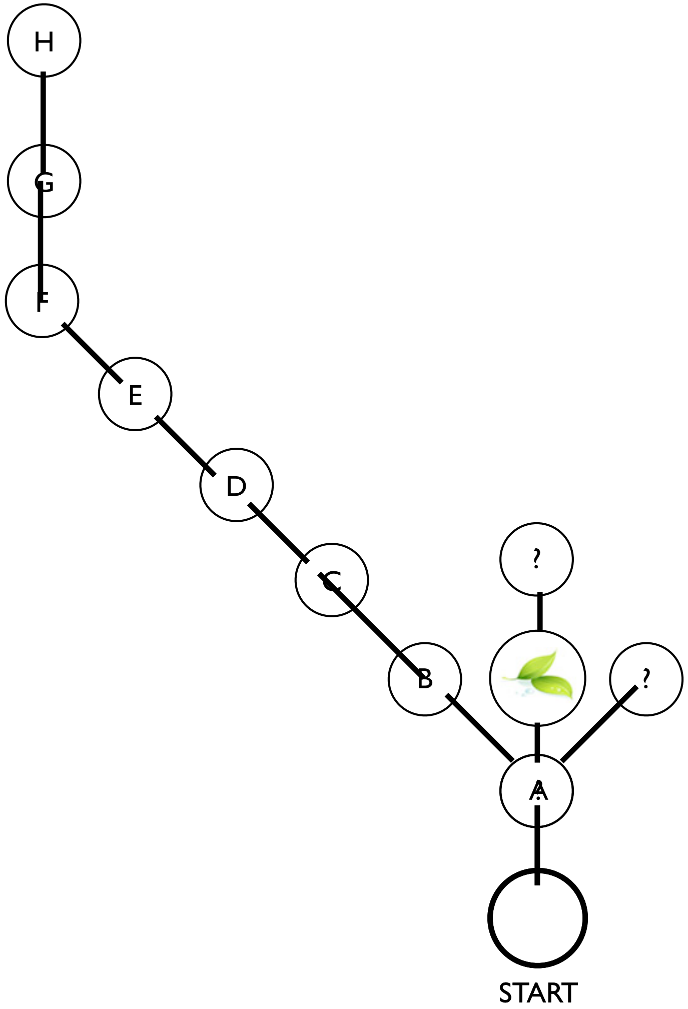
Depth first search back-tracking



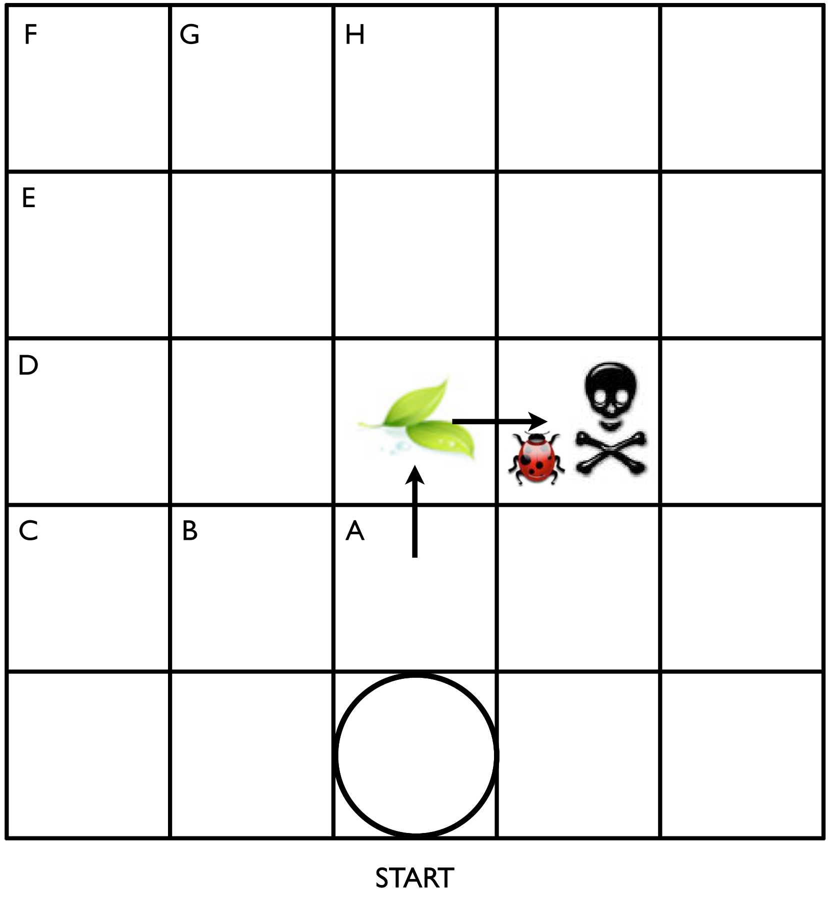


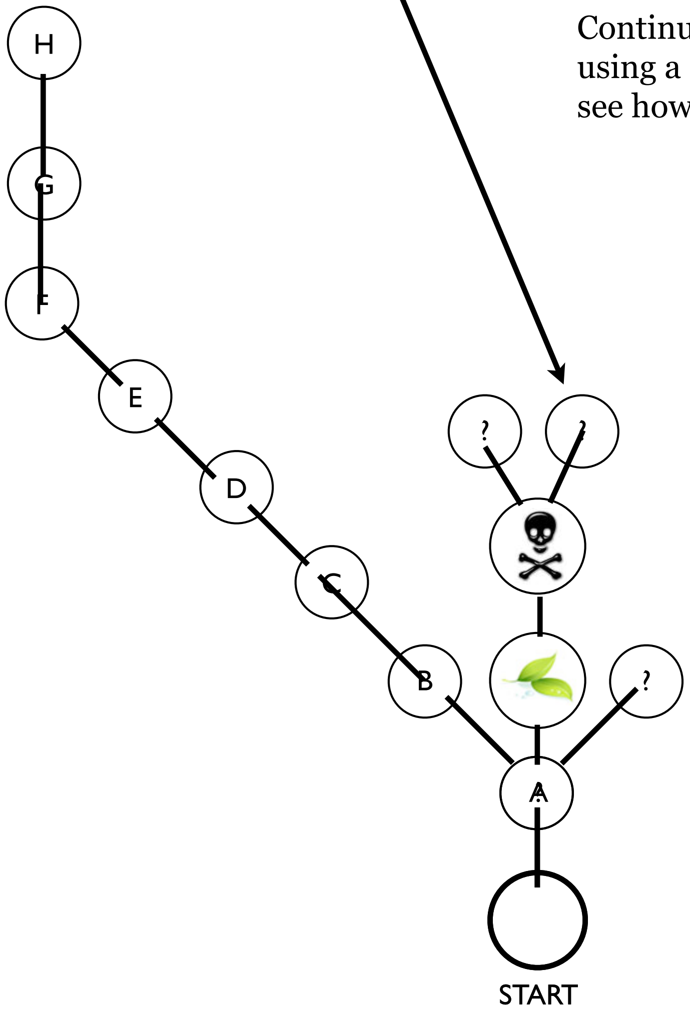
Depth first search continuation after back-tracking





Depth first search *cyclic path*!



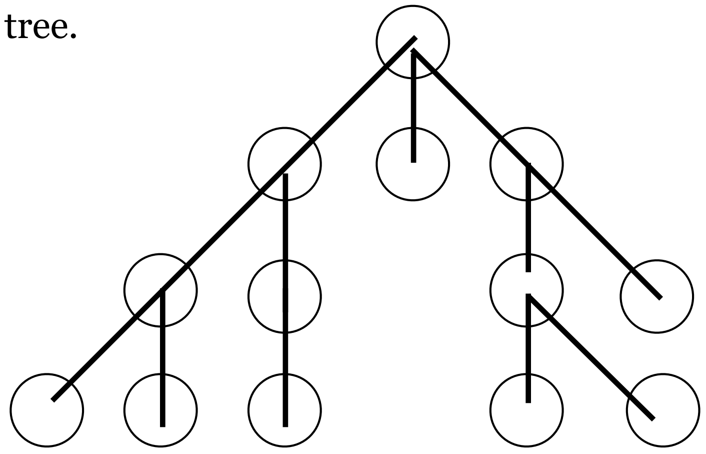


What is this option?
When will we get here?
What will happen when we explore A’s third exit?

Continue the exploration using a depth first search to see how things work out!

Depth First Search Exercise

Here is a simple search tree.

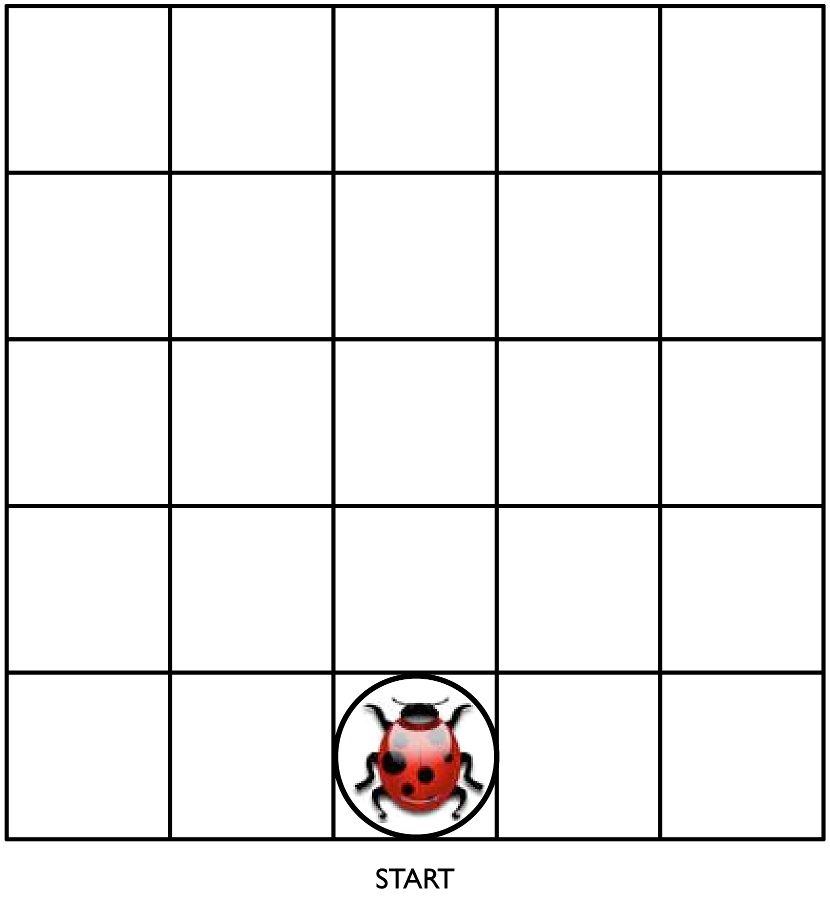


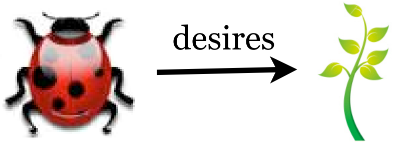
Exercise 1. Number the nodes in the order that a *depth first* search would visit them.

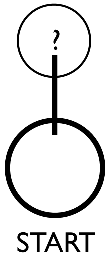
Breadth first search

This strategy explores *across* the complete search graph before taking a step deeper.

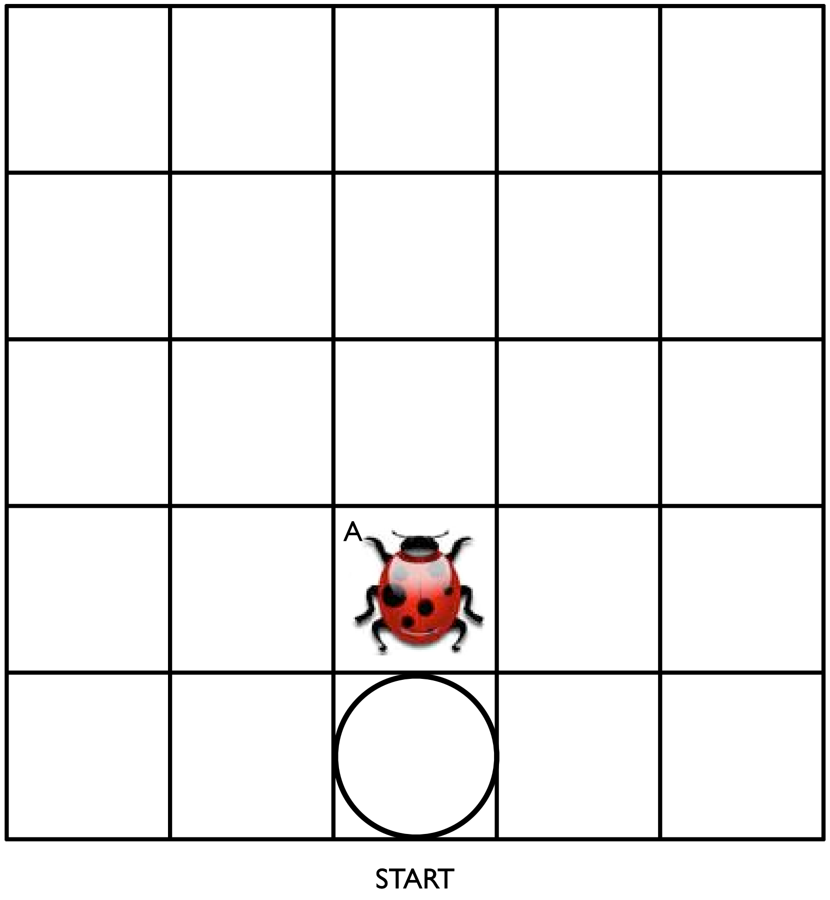
Think of exploring a cave by peeping into each and every opening from the current location first, before moving one step deeper into the ground and repeating the process for all the options you saw in the previous level.



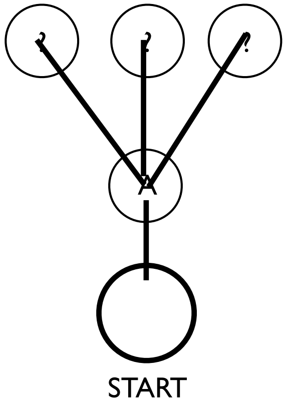




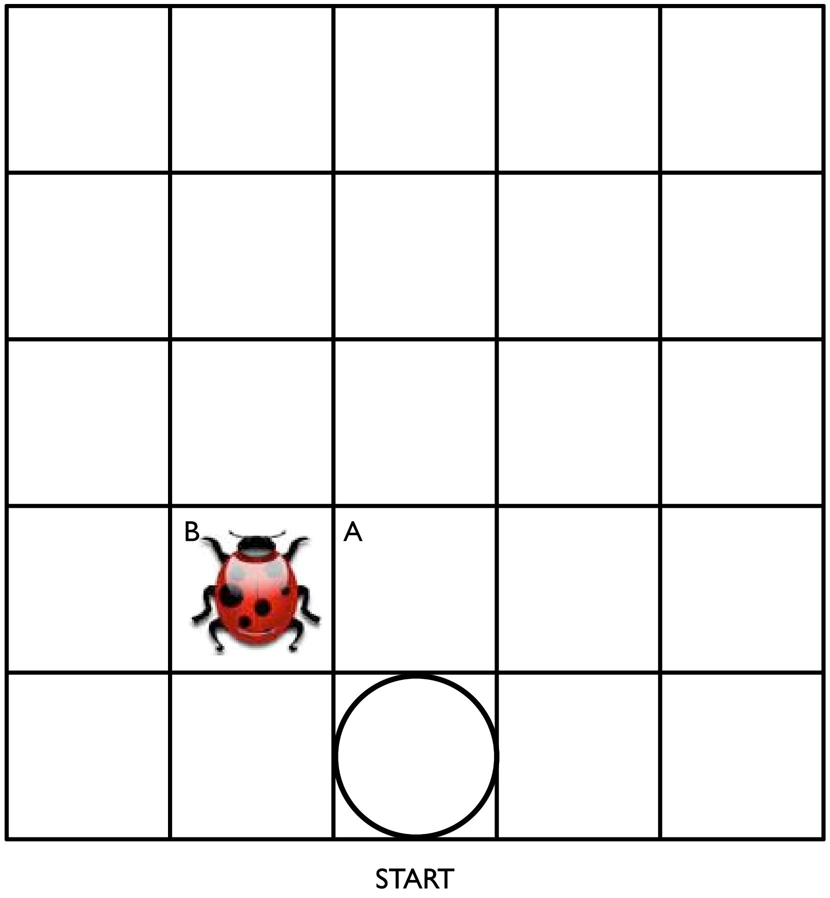
Breadth first search



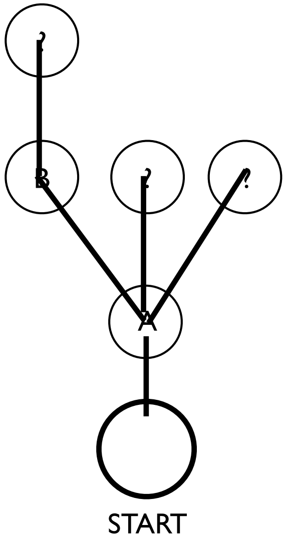
From cell A, the lady-bird has 3 unvisited options.



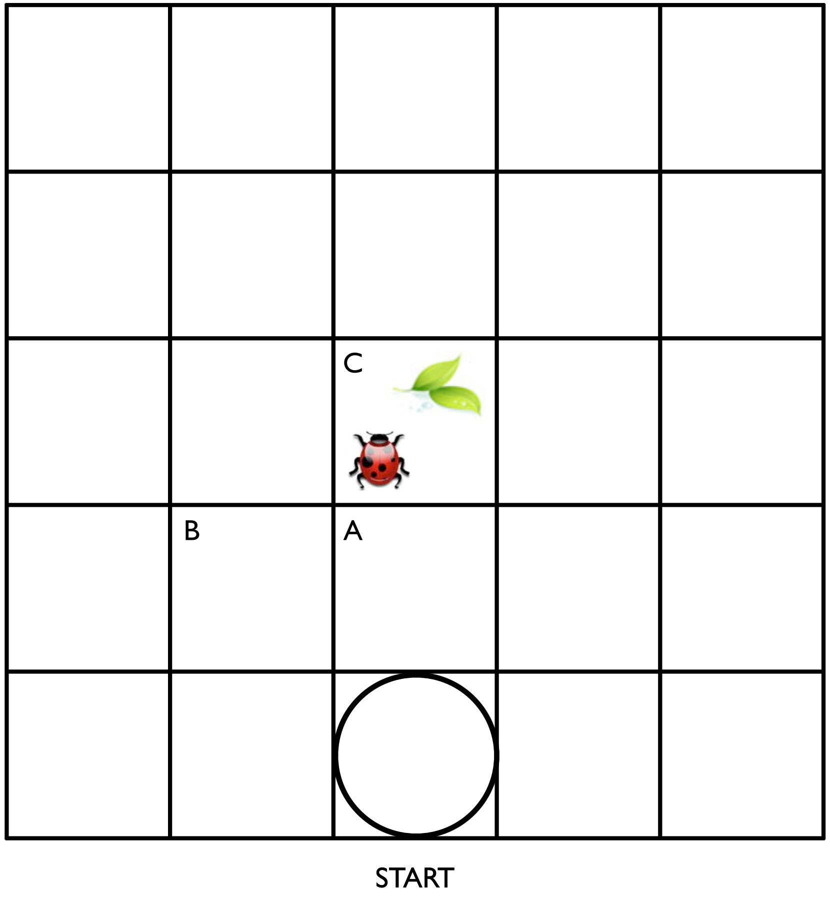
Breadth first search



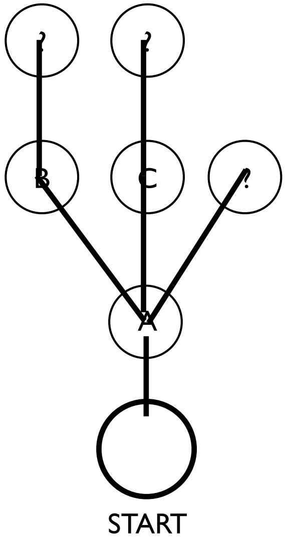
From cell A, the lady-bird takes the first available path.



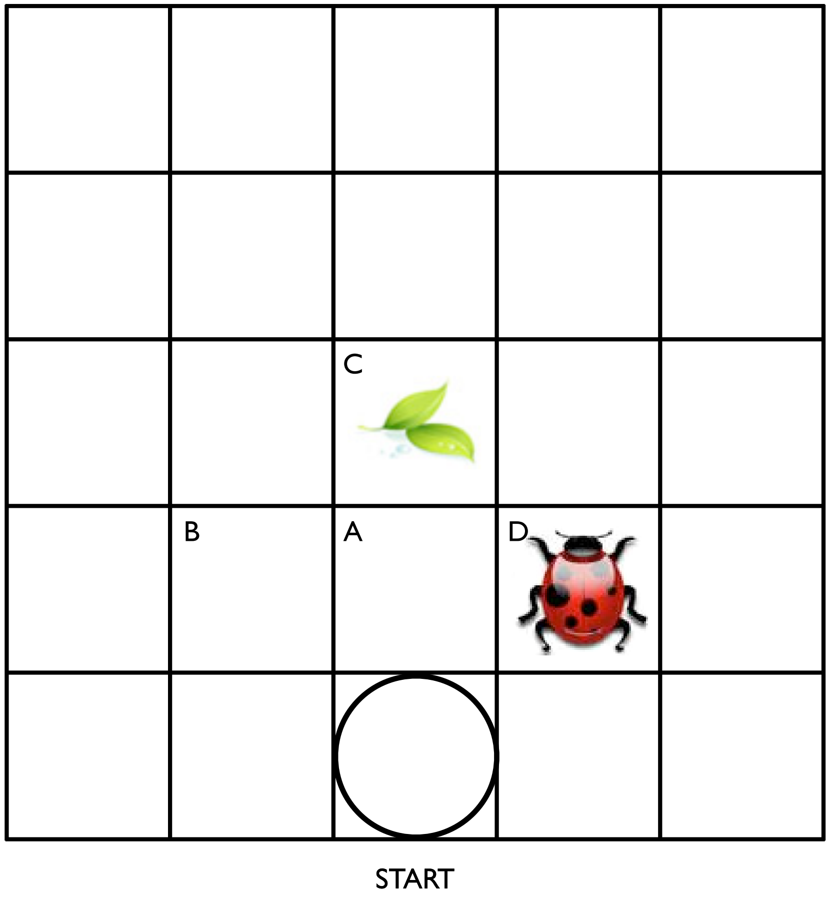
Breadth first search



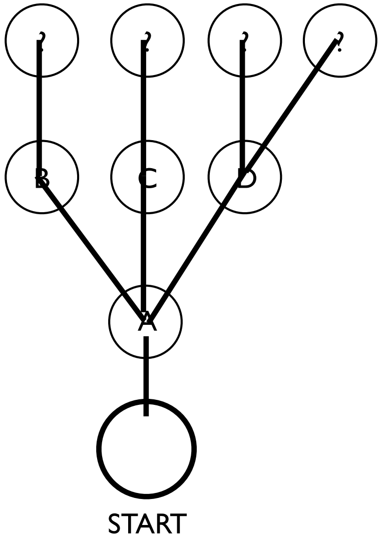
Next, the lady-bird takes the second available path from A.



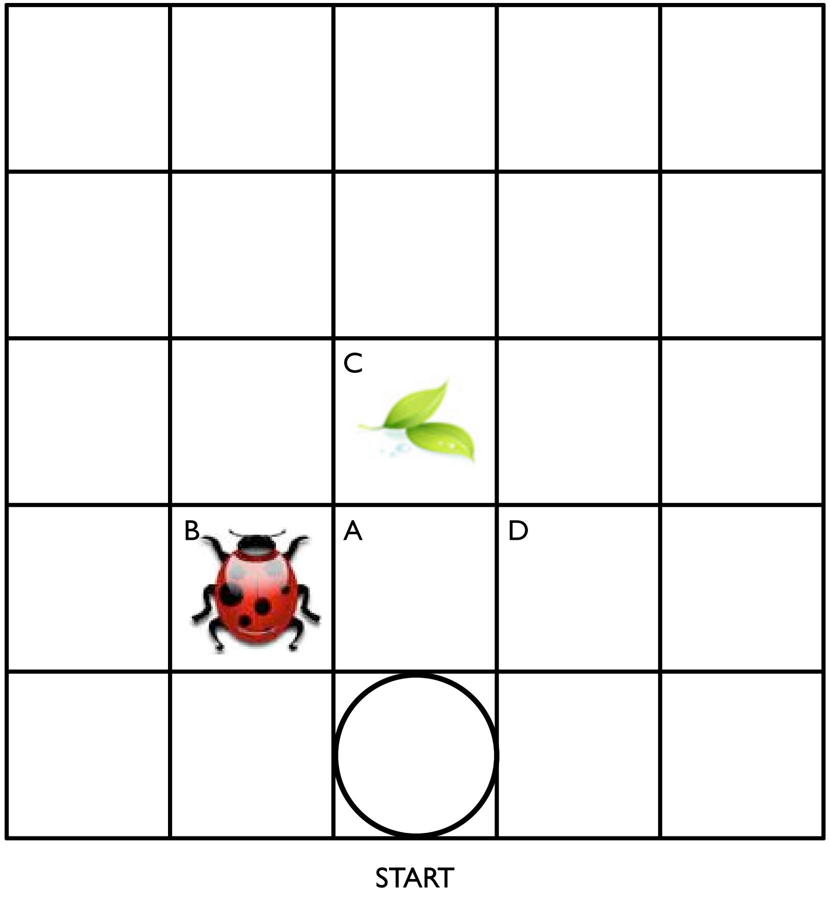
Breadth first search



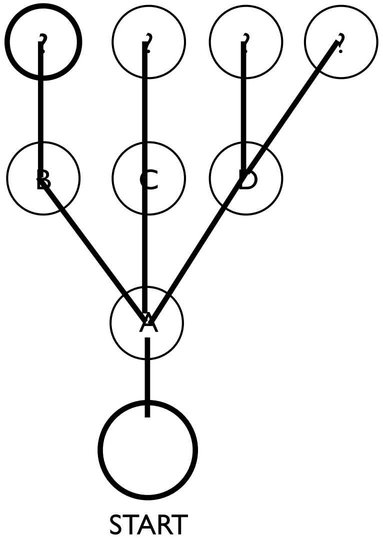
Then, the lady-bird takes the third available path from A.



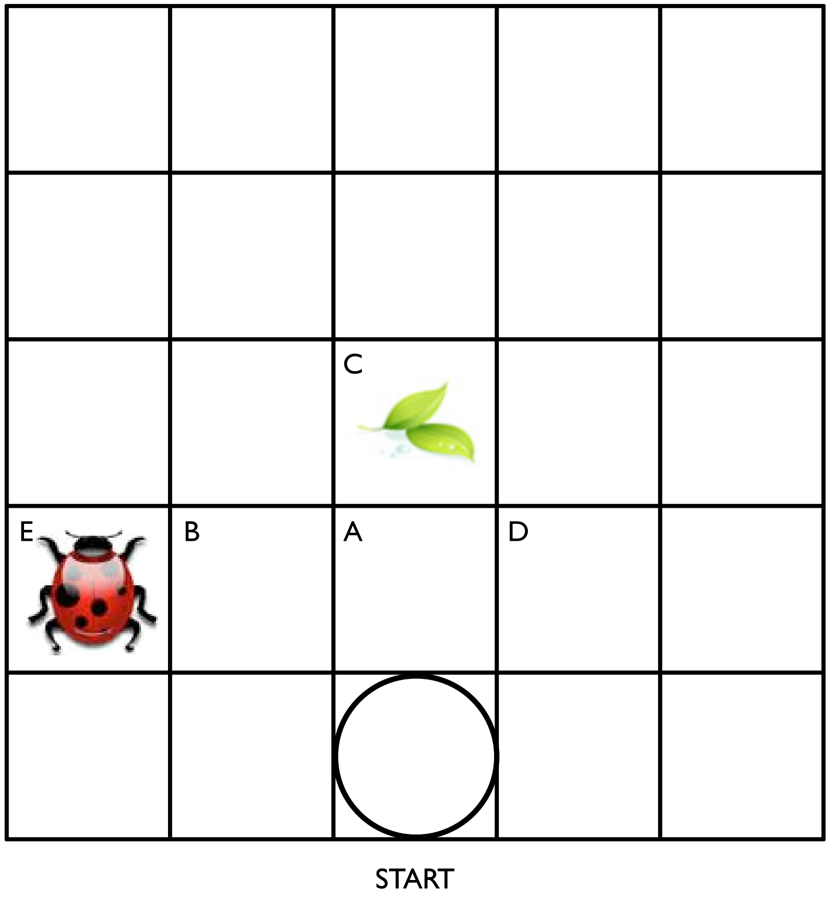
Breadth first search

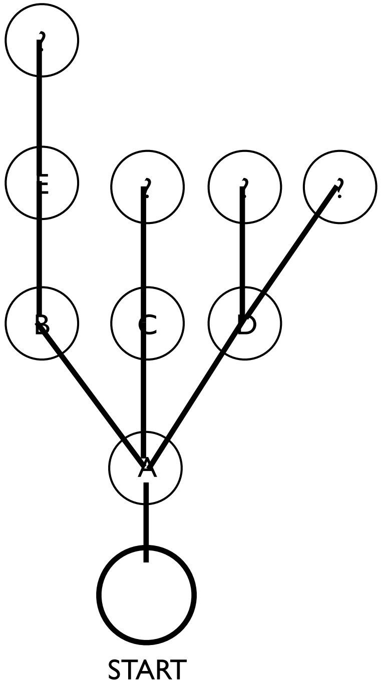


Since the lady-bird has now explored the first level of the tree completely, it returns to the start of the level (cell B) and will begin to explore the paths from there...

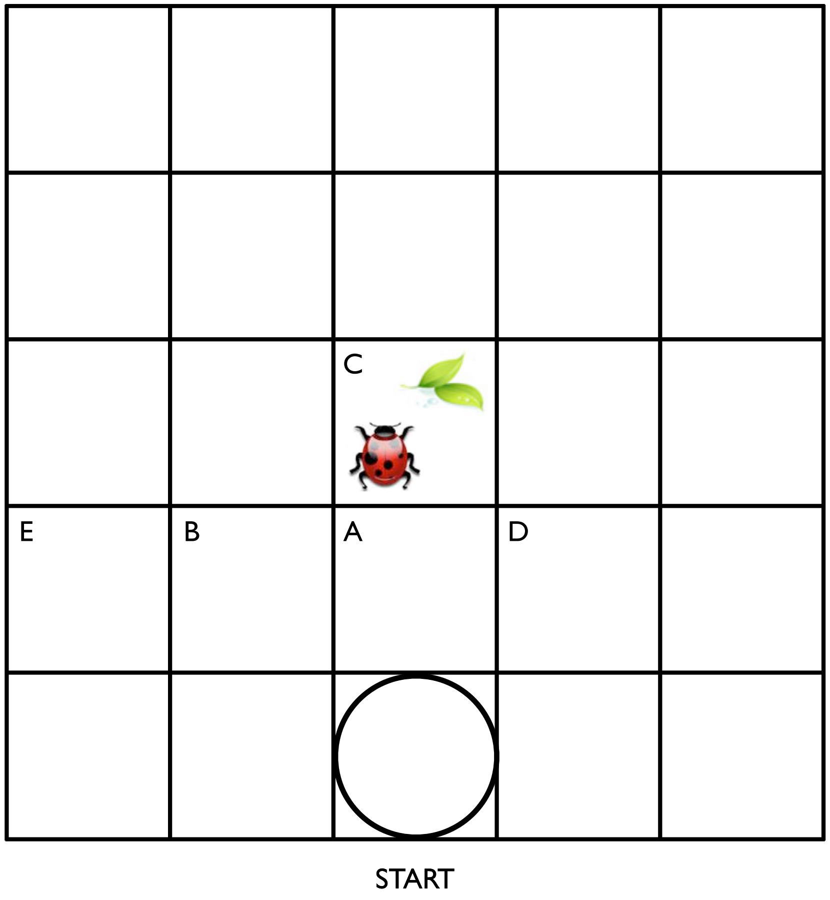


Breadth first search

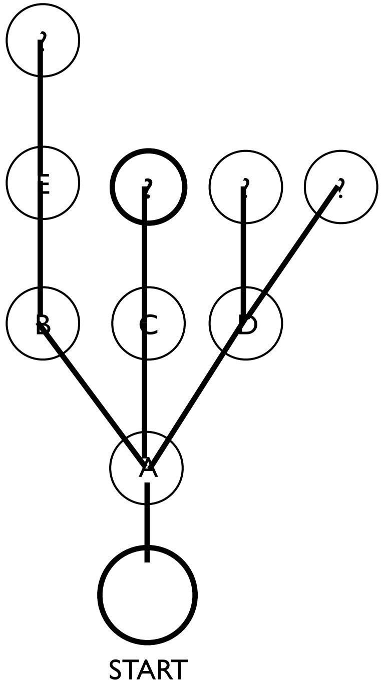




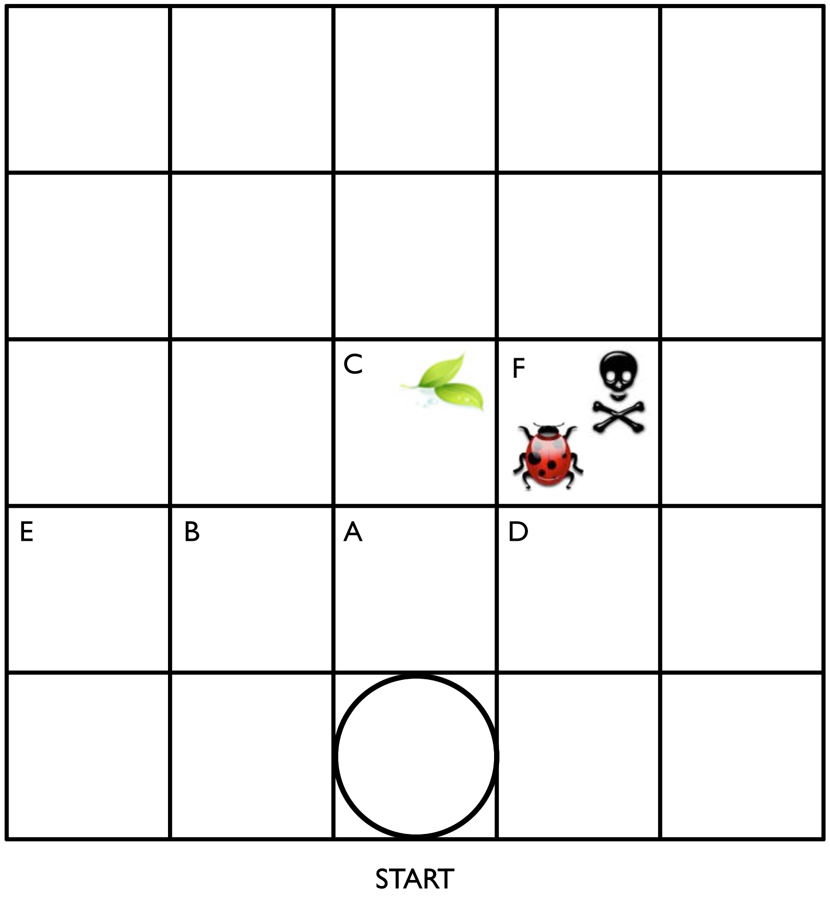
Breadth first search

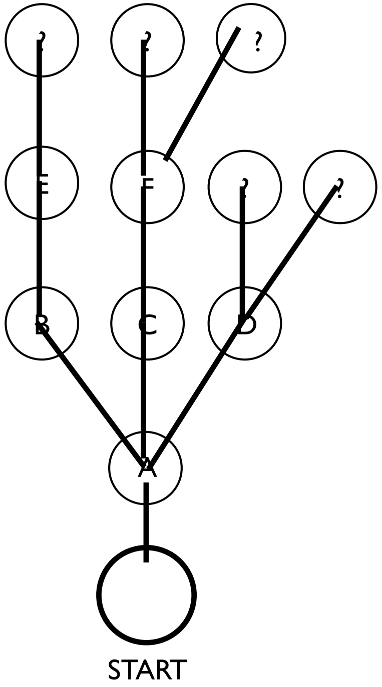


Since cell B has no more unexplored paths, the ladybird moves on to the next cell *at the same level as B*, cell C and explores the paths from there...

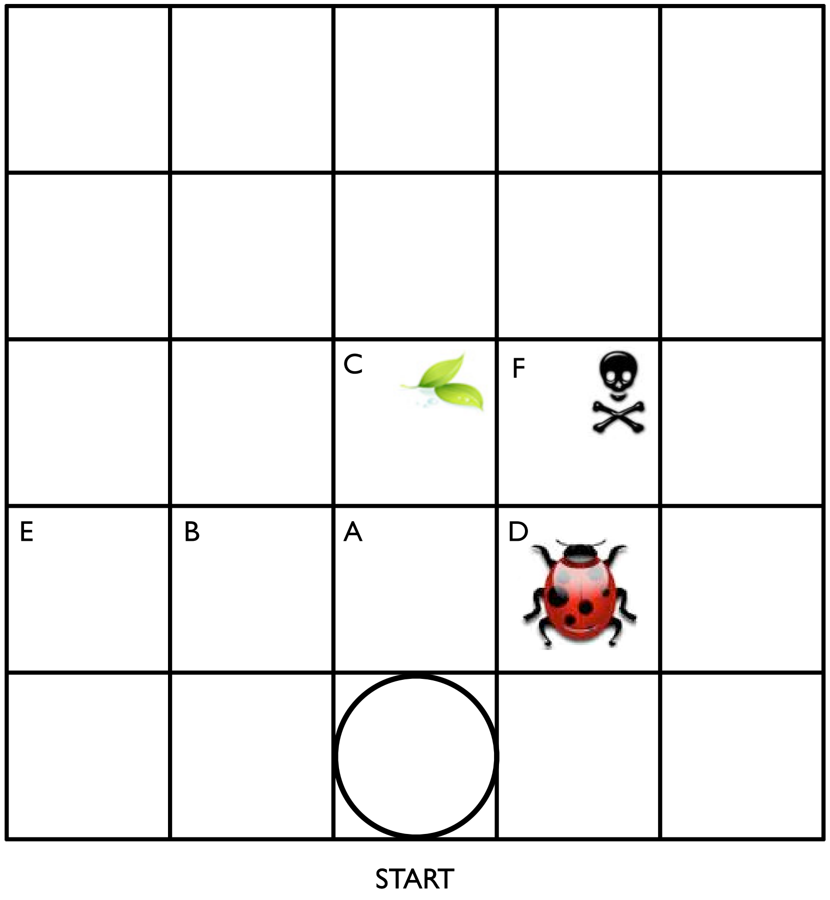


Breadth first search





Breadth first search

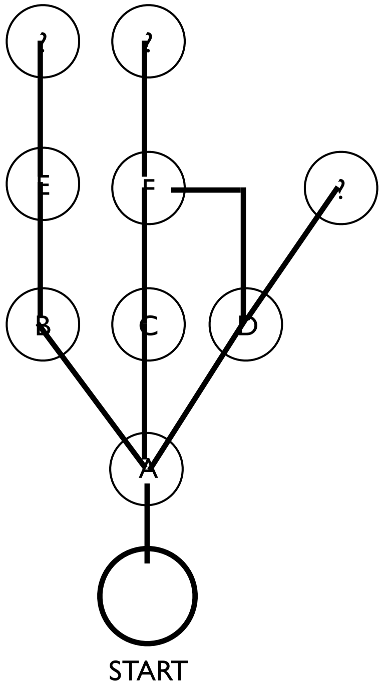


Now that we've explored beyond C and found F, we realise that the first "unexplored" path from D is actually F, we've been there already!

The lady-bird can ignore this option and explore the next one.

1. We could connect D to the existing node for F.
2. We could instead duplicate the subtree from F and beyond (it already appears connected to C).

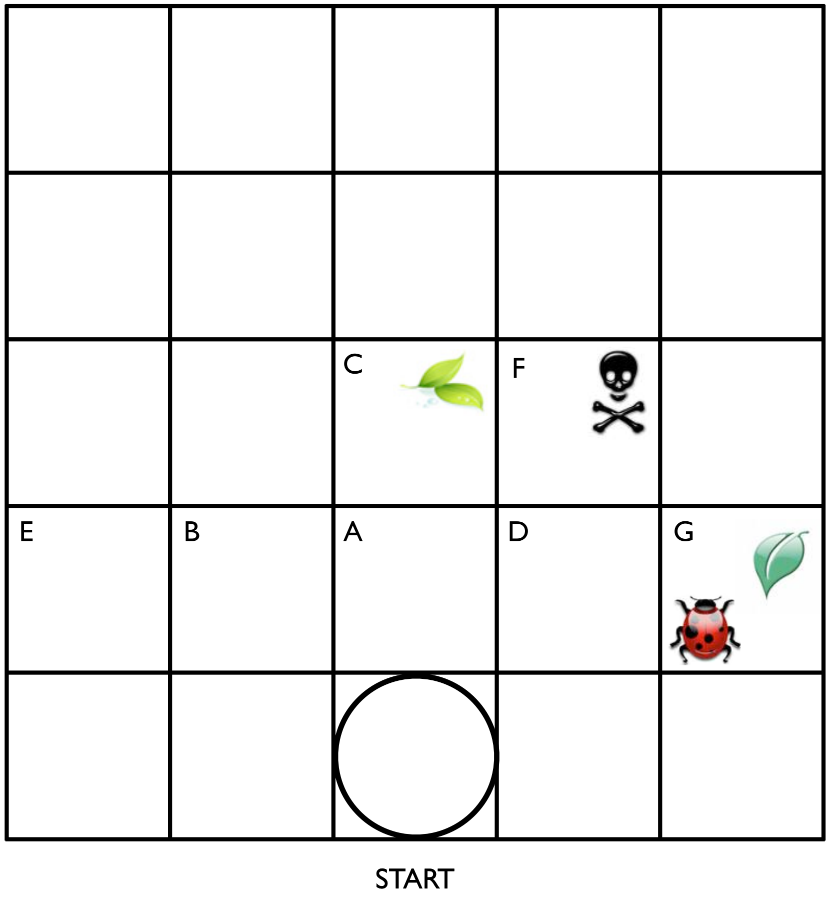
We have seen examples of both kinds of graph/tree in the notes already. It will depend on what we want the tree for!

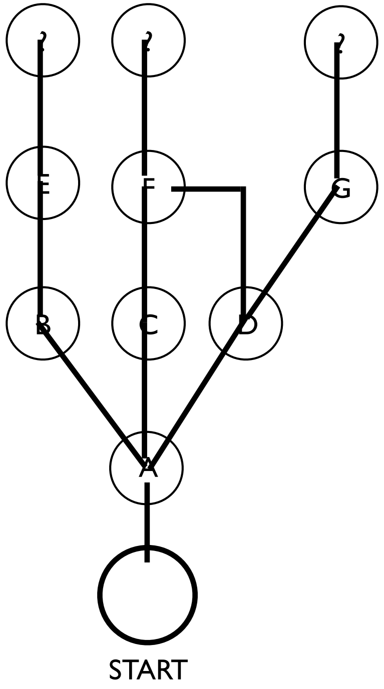


Breadth first search

The ladybird has now explored 3 levels of the tree fully and needs to move on to explore the 4th level.

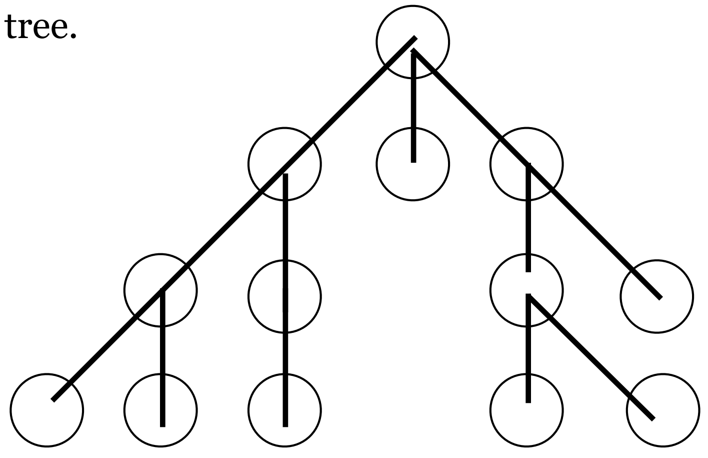
Complete the exploration using a breadth first search in your own time to test your understanding.





Breadth First Search Exercise

Here is a simple search tree.



Exercise 2. Number the nodes in the order that a *breadth first* search would visit them.

Write an algorithm for a depth first search.

Have you met the learning objectives?

Can you read and interpret search trees and graphs?

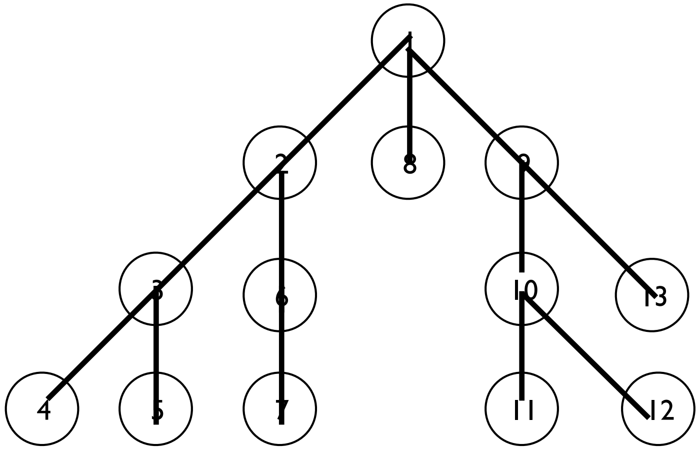
Can you construct a search tree or graph for an agent exploring a space.

Do you understand how to apply depth first search and breadth first search?

Draw a *simple*, continuous space maze with a single entrance and an exit. Explore the maze using different strategies to test your understanding.

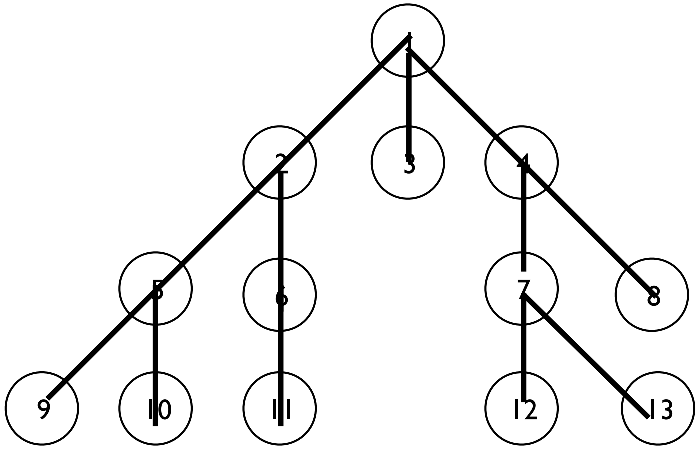
Which algorithm finds the exit the quickest?
Which algorithm constructs the largest tree before it finds the exit?
Is this true for all starting positions within the maze?

Answer 1. Depth First



question △

Answer 2. Breadth First



question △