UNIT 2 UNINFORMED SEARCH OR BRUTE FORCE SEARCH

CONTENTS

1.0 INTRODUCTION

In computer science, uninform-cost search (UCS) is a tree search algorithm used for traversing or searching a weighted tree, tree structure, or graph. The search begins at the root node. The search continues by visiting the next node which has the least total cost from the root. Nodes are visited in this manner until a goal state is reached.

Typically, the search algorithm involves expanding nodes by adding all unexpanded neighbouring nodes that are connected by directed paths to a priority queue. In the queue, each node is associated with its total path cost from the root, where the least-cost paths are given highest priority. The node at the head of the queue is subsequently expanded, adding the next set of connected nodes with the total path cost from the root to the respective node. The uniform-cost search is complete and optimal if the cost of each step exceeds some positive bound ε.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

Explain uninformed search
List two types of uninformed search
Describe depth first and breadth first search
Solve simple problems on uninformed search.

3.0 MAIN CONTENT

3.1 Uninformed Search

Sometimes we may not get much relevant information to solve a problem. Suppose we lost our car key and we are not able to recall where we left, we have to search for the key with some information such as in which places we used to place it. It may be our pant pocket or may be the table drawer. If it is not there then we have to search the whole house to get it. The best solution would be to search in the places from the table to the wardrobe. Here we need to search blindly with fewer clues. This type of search is called uninformed search or blind search. There are two popular AI search techniques in this category: breadth first search and depth first search.

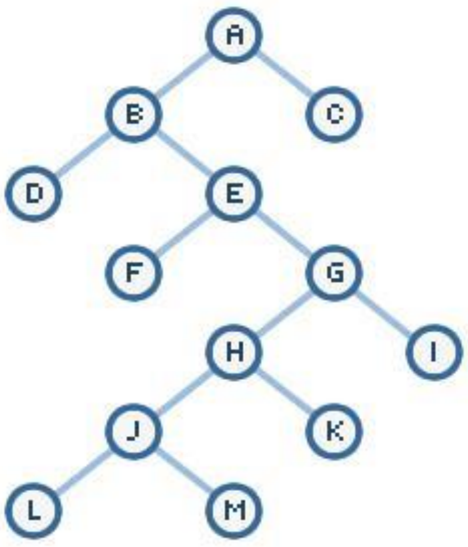
3.2 Depth First and Breadth First Search

If you want to go from Point A to Point B, you are employing some kind of search. For a direction finder, going from Point A to Point B literally means finding a path between where you are now and your intended destination. For a chess game, Point A to Point B might be two points between its current position and its position 5 moves from now. For a genome sequence, Points A and B could be a link between two DNA sequences.

As you can tell, going from Point A to Point B is different for every situation. If there is a vast amount of interconnected data, and you are trying to find a relation between few such pieces of data, you would use search. In this unit, you will learn about two forms of searching, depth first and breadth first.

Our Search Representation

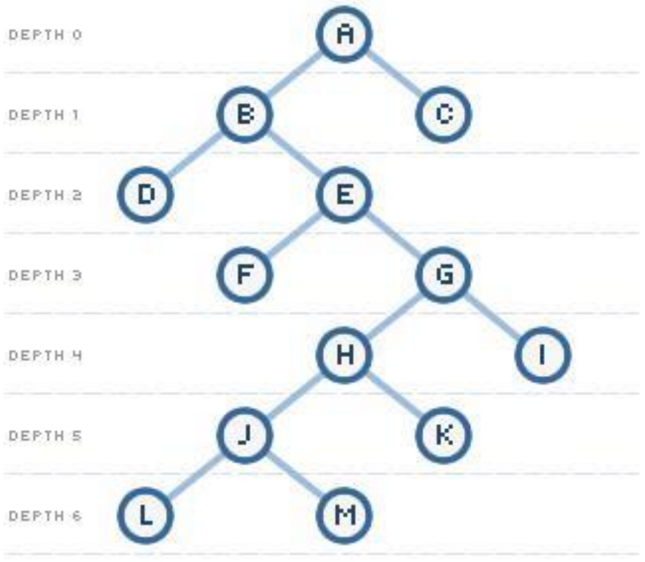
Let’s you learn how we humans could solve a search problem. First, we need a representation of how our search problem will exist. The following is an example of our search tree. It is a series of interconnected nodes that we will be searching through:



In our above graph, the path connections are not two-way. All paths go only from top to bottom. In other words, A has a path to B and C, but B and C do not have a path to A. It is basically like a one-way street.

Each lettered circle in our graph is a node. A node can be connected to other via our edge/path, and those nodes that are connected to be called neighbours. B and C are neighbours of A. E and D are neighbors of B, and B is not a neighbour of D or E because B cannot be reached using either D or E.

Our search graph also contains depth:

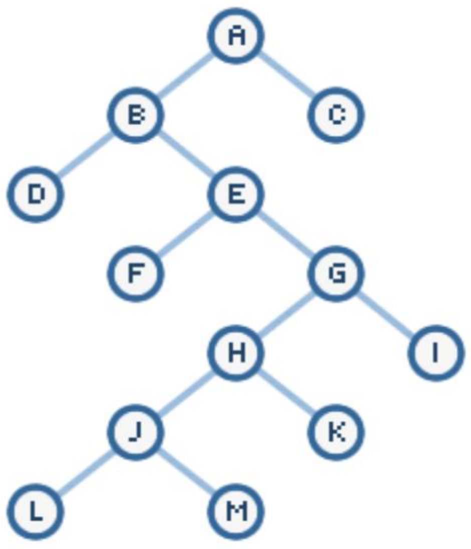


We now have a way of describing location in our graph. We know how the various nodes (the lettered circles) are related to each other(neighbours), and we have a way of characterizing the depth each belongs in. Knowing this information isn't directly relevant in creating our search algorithm, but they do help us to better understand the problem.

3.2.1 Depth First Search

Depth first search works by taking a node, checking its neighbors, expanding the first node it finds among the neighbors, checking if that expanded node is our destination, and if not, continue exploring more nodes.

The above explanation is probably confusing if this is your first exposure to depth first search. I hope the following demonstration will help you more. Using our same search tree, let's find a path between nodes A and F:



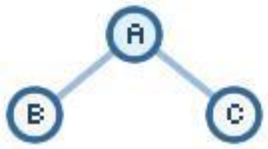
Step 0

let’s start with our root/goal node:

I will be using two lists to keep track of what we are doing - an Open list and a Closed List. An Open list keeps track of what you need to do, and the Closed List keeps track of what you have already done. Right now, we only have our starting point, node A. We haven't done anything to it yet, so let's add it to our Open list.

Open List: A
Closed List: <empty>

Step 1
Now, let's explore the neighbors of our A node. To put another way, let's take the first item from our Open list and explore its neighbors:



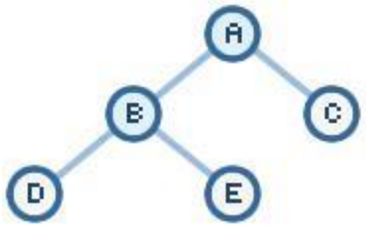
Node A's neighbors are the B and C nodes. Because we are now done with our A node, we can remove it from our Open list and add it to our Closed List. You aren't done with this step though. You now have two new nodes B and C that need exploring. Add those two nodes to our Open list.

Our current Open and Closed Lists contain the following data:

Open List: B, C
Closed List: A

Step 2

Our Open list contains two items. For depth first search and breadth first search, you always explore the first item from our Open list. The first item in our Open list is the B node. B is not our destination, so let's explore its neighbors:



Because I have now expanded B, I am going to remove it from the Open list and add it to the Closed List. Our new nodes are D and E, and we add these nodes to the *beginning* of our Open list:

Open List: D, E, C
Closed List: A, B

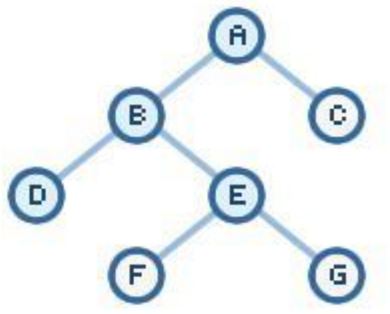
Step 3

You should start to see a pattern forming. Because D is at the beginning of our Open List, we expand it. D isn't our destination, and it does not contain any neighbors. All you do in this step is remove D from our Open List and add it to our Closed List:

Open List: E, C
Closed List: A, B, D

Step 4

We now expand the E node from our Open list. E is not our destination, so we explore its neighbors and find out that it contains the neighbors F and G. Remember, F is our target, but we don't stop here though. Despite F being on our path, we only end when we are about to *expand* our target Node - F in this case:

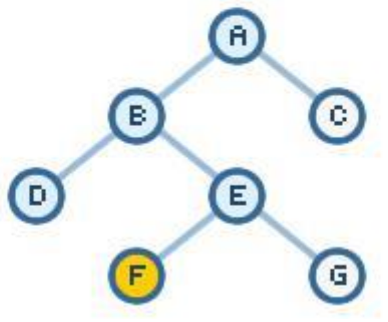


Our Open list will have the E node removed and the F and G nodes added. The removed E node will be added to our Closed List:

Open List: F, G, C
Closed List: A, B, D, E

Step 5

We now expand the F node. Since it is our intended destination, we stop:



We remove F from our Open list and add it to our Closed List. Since we are at our destination, there is no need to expand F in order to find its neighbors. Our final Open and Closed Lists contain the following data:

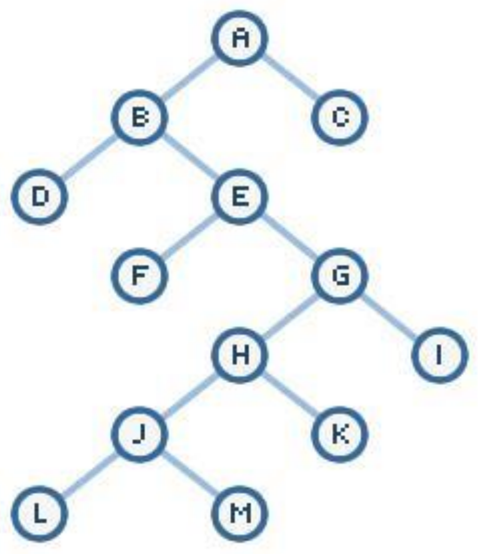
Open List: G, C
Closed List: A, B, D, E, F

The final path taken by our depth first search method is what the final value of our Closed List is: A, B, D, E, F. Towards the end of this tutorial, I will analyze these results in greater detail so that you have a better understanding of this search method.

3.2.2 Breadth First Search

The reason I cover both depth and breadth first search methods in the same unit is because they are both similar. In depth first search, newly explored nodes were added to the beginning of your Open list. In breadth first search, newly explored nodes are added to the end of your Open list.

Let's see how that change will affect our results. For reference, here is our original search tree:



Let's try to find a path between nodes A and E.

Step 0

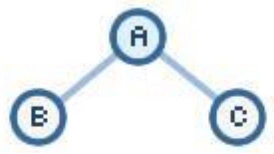
let’s start with our root/goal node:

Like before, I will continue to employ the Open and Closed Lists to keep track of what needs to be done:

Open List: A
Closed List: <empty>

Step 1

Now, let's explore the neighbours of our A node. So far, we are following in depth first's footsteps:



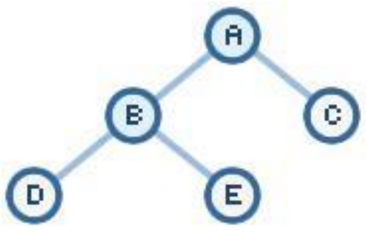
We remove A from our Open list and add A to our Closed List. A's neighbours, the B and C nodes, are added to our Open list. They are added to the end of our Open list, but since our Open list was empty (after removing A), it's hard to show that in this step.

Our current Open and Closed Lists contain the following data:

Open List: B, C
Closed List: A

Step 2

Here is where things start to diverge from our depth first search method. We take a look the B node because it appears first in our Open List. Because B isn't our intended destination, we explore its neighbours:

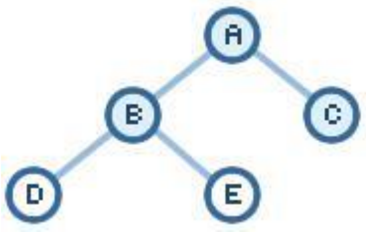


B is now moved to our Closed List, but the neighbours of B, nodes D and E are added to the *end* of our Open list:

Open List: C, D, E
Closed List: A, B

Step 3

We now expand our C node:



Since C has no neighbours, all we do is remove C from our Closed List and move on:

Open List: D, E
Closed List: A, B, C

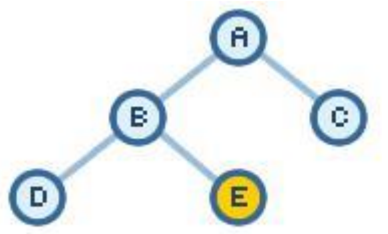
Step 4

Similar to Step 3, we expand node D. Since it isn't our destination, and it too does not have any neighbours, we simply remove D from our to Open list, add D to our Closed List, and continue on:

Open List: E
Closed List: A, B, C, D

Step 5

Because our Open list only has one item, we have no choice but to take a look at node E. Since node E is our destination, we can stop here:



Our final versions of the Open and Closed Lists contain the following data:

Open List: <empty>
Closed List: A, B, C, D, E

Traveling from A to E takes you through B, C, and D using breadth first search

4.0 CONCLUSION

1. Uninformed search strategies -Also known as "blind search," uninformed search strategies use no information about the likely "direction" of the goal node(s).
2. Uninformed search major methods are Breadth-first and depth-first

5.0 SUMMARY

In this unit, you learnt that:

Uninformed strategies use only the information available in the problem definition.

Some such strategies considered :

- Breadth-first search
- Tree Search
- Depth-first search

6.0 TUTOR -MARKED ASSIGNMENT

Water Jug Problem

i. Given a 5-gallon jug and a 2-gallon jug, with the 5-gallon jug initially full of water and the 2-gallon jug empty, the goal is to fill the 2-gallon jug with exactly one gallon of water.

ii. 8-Puzzle
Given an initial configuration of eight numbered tiles on a 3 x 3 board, move the tiles in such a way so as to produce a desired goal configuration of the tiles.

iii. Missionaries and Cannibals
There are three missionaries, three cannibals, and 1 boat that can carry up to two people on one side of a river. Goal: Move all the missionaries and cannibals across the river.

iv. Remove five Sticks
Given the following configuration of sticks; remove exactly five sticks in such a way that the remaining configuration forms exactly three squares.

7.0 REFERENCES/FURTHER READING

Christopher, D. M. & Hinrich, S., *Foundations of Statistical Natural Language Processing.* The MIT Press.

Wooldridge, M. *An Introduction to Multiagent Systems*. John Wiley & Sons, Ltd.

Russell, S. J. & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach* (2nd ed.). Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2, http://aima.cs.berkeley.edu/

Russell, S. & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3 ed.). Prentice Hall. ISBN 978-0-13-6042594.