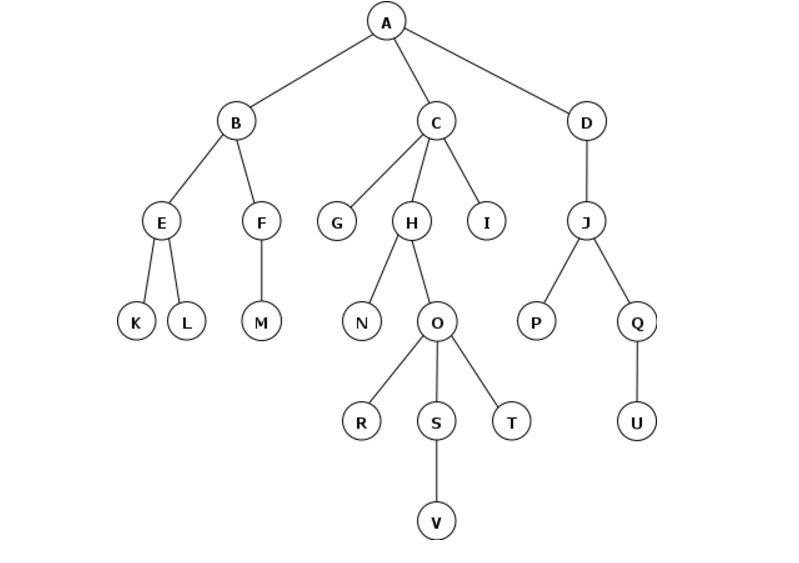
Artificial Intelligence (CS F407)



1. Problem Description

The problem demands the development of an intelligent vacuum cleaner that can sense its immediate environment and act accordingly to minimise the cost and time required to eliminate dirt from a 2D environment. The PEAS analysis of the agent includes: Persepts – Presence or absence of dirt, Environment – a 2D floor with dirt on various tiles, Actuators – Motors and tyres in the vacuum cleaner and Sensors – cameras for detecting dirt.

1. What is informed search

Informed searching algorithms include those that have predefined heuristics that instruct the agent to react in a certain manner under given conditions. These predefined rules or heuristics cam help to reduce humongous computations that are practically impossible to work out to simple algorithms that can be run even on personal computers. Some of the popular methods that fall in this category are: best-first search, greedy best-first search, beam search, hill climbing, algorithm A, algorithm A\* etc.

1. What is uninformed search

Uninformed search algorithms can also be termed as brute force or blind algorithms. These usually have no predefined notions on reaction to different environments and try to compute all possible alternatives to find the best possible solution to a set problem. These can create huge computational and memory loads on processors and often require exceptionally high performance computers to generate the solution. Various types of algorithms that fall into this category can be broadly classified as breadth-first search, depth-first search, depth first iterative deepening. Bidirectional search.

1. Iterative Deepening

The algorithm that has been used for the uninformed search for this problem is iterative deepening algorithm. The depths are continuously incremented by a common difference and the complete tree is searched for the goal state node. If not found in the current depth, the depth of the tree is again incremented and the entire process of searching the tree is repeated.

1. Heuristics Used

To speed up the otherwise impossible algorithms a heuristic has been applied. This heuristic assigns score to each possible movement of the intelligent agent based on the observable environment. The agent then makes the movement that has the maximum score.

Another heuristic that has been used to make the agent cost efficient is to avoid returning to the same state that it has been through again. This heuristic made the agent highly effective in terms of finding the shortest path to the goal state and significantly minimised the time required to arrive at the solution.

1. Relative Performance of heuristics

As already mentioned the second heuristic performed much better than the first heuristic to and returned a far better path.

1. Time constraint

Time constraint always plays a major role in such algorithms as a huge number of nodes have to be created checked and indexed during its implementation. The first trial using an uninformed breadth first search ran for straight 6 hours without any results showing how lengthy the computations for such algorithms can get if proper heuristics are not employed.

1. Memory constraint

During another attempt in using iterative deepening algorithm, a huge number of nodes started to pile up in the memory raising the memory used by the running process to 4 GBs within minutes. Hence memory is also a resource that is to be considered scarce during the implementations of such algorithms.

1. About the program

Run the file named **run\_prog.py** to see the animation of Heuristic 2(improved path) and the graph comparison of the two heuristics.

To run the algorithm with uninformed search, run the run\_uninformed.py file. (Takes a long time to return results).

The graphs will start to appear after ‘Done’ has been printed 4 times in the shell.

To run programs individually, there are py files named (run\_uninformed.py, run\_heuristic1.py and run\_heuristic2.py) corresponding to each algorithm for the search.