SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, KTR

18CSC305J- Artificial Intelligence

Subject Faculty: Dr. Deepa R(Assistant Professor)

CHEAT SHEET FOR A STAR SEARCH ALGORITHM

TEAM MEMBERS:

RA2111003010693 - Divyanshu Yadav
RA2111003010703 - Khushi Mishra
RA2111003010752 - Aparna Singh

A* Search is an informed best-first search algorithm that efficiently determines the lowest cost path between any two nodes in a directed weighted graph with non-negaΘve edge weights. This algorithm is a variant of Dijkstra's algorithm. A slight difference arises from an evaluaΘon funcΘon determining which node to explore next.

$f(x) = g(x) + h(x)$

The Algorithm

The A* algorithm is implemented in a similar way to Dijkstra's algorithm. Given a weighted graph with non-negaΘve edge weights, to find the lowest-cost path from a start node S to a goal node G, two lists are used:

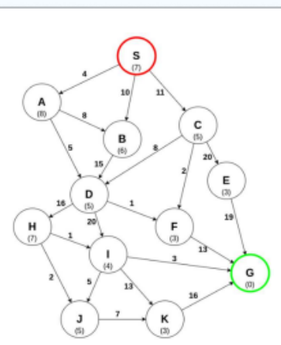
An open list is implemented as a priority queue, which stores the next nodes to be explored. Because this is a priority queue, the most promising candidate node (the one with the lowest value from the evaluaΘon funcΘon) is always at the top. IniΘally, the only node in this list is the start node S.

A closed list that stores the nodes that have already been evaluated. When a node is in the closed list, it means that the lowest-cost path to that node has been found. To find the lowest cost path, a search tree is constructed in the following way:

- Initalize a tree with the root node being the start node S.
- Remove the top node from the open list for exploraΘon.
- Add the current node to the closed list.
- Add all nodes with an incoming edge from the current node as child nodes in the tree.
- Update the lowest cost to reach the child node.
- Compute the evaluaΘon funcΘon for every child node and add them to the open list.

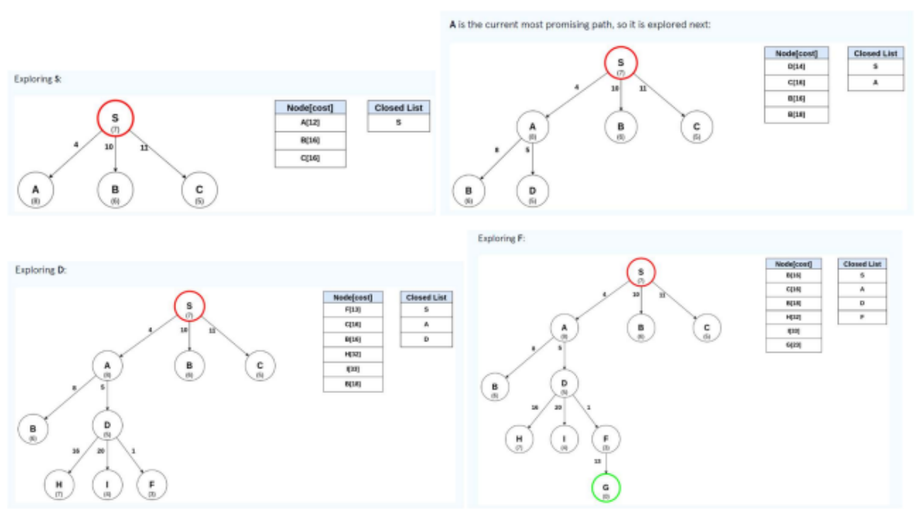
current_lowest_cost = min(current_lowest_cost, parent_node_cost + edge_weight)

Consider the following example of trying to find the shortest path from S to G in the following graph:

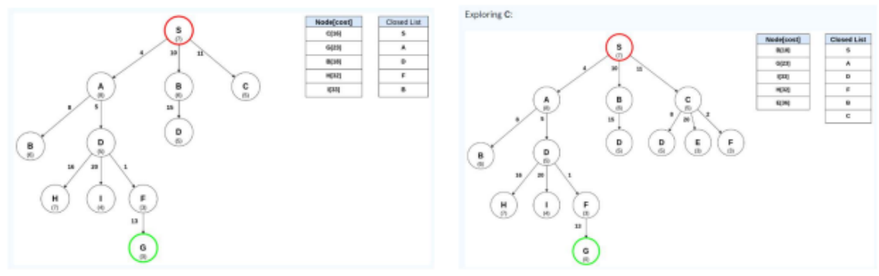


Each edge has an **associated weight**, and each node has a **heuristic cost** (in parentheses).

An open list is maintained in which the node S is the only node in the list. The search tree can now be constructed.

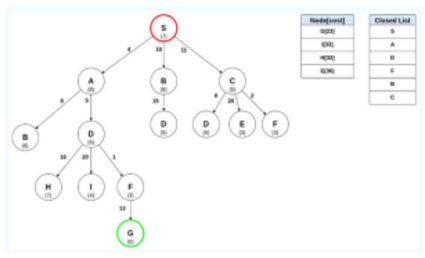


Notice that the goal node G has been found. However, it hasn't been explored, so the algorithm continues because there may be a shorter path to G. The node B has two entries in the open list: one at a cost of 16 (child of S) and one at a cost of 18 (child of A). The one with the lowest cost is explored next:

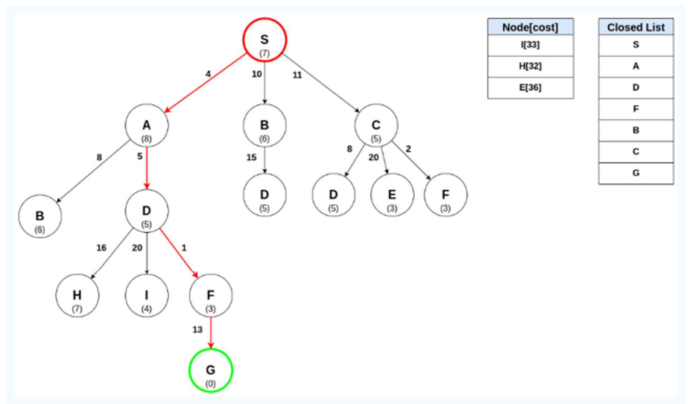


The next node in the open list is again B. However, because B has already been explored, meaning a shortest path to B has been found, it is not explored again and the algorithm continues to the next candidate.

The next node in the open list is again B. However, because B has already been explored, meaning a shortest path to B has been found, it is not explored again and the algorithm continues to the next candidate.



The next node to be explored is the goal node G, meaning the shortest path to G has been found! The path is constructed by tracing the graph backward from G to S:



Using the A* Algorithm

This algorithm is guaranteed to find a shortest path if one exists. One of the main uses of this algorithm is route planning. However, there are many other uses.