**EXPERIMENT 7 - AO\* Algorithm**

**Aim:**

Write a program to implement AO\* Algorithm.

**Theoretical Background:**

The AO\* algorithm is a knowledge based search technique, meaning the start state and the goal state is already defined, and the best path is found using heuristics. The time complexity of the algorithm is significantly reduced due to the informed search technique.

AO\* algorithm is a best first search algorithm. AO\* algorithm uses the concept of AND-OR graphs to decompose any complex problem given into smaller sets of problems which are further solved. AND-OR graphs are specialized graphs that are used in problems that can be broken down into subproblems where AND side of the graph represents a set of tasks that need to be done to achieve the main goal, whereas the OR side of the graph represents the different ways of performing tasks to achieve the same main goal.

**Algorithm:**

Step 1: Start.

Step 2: Let GRAPH consists only of the node representing the initial state (call this node as INIT). Compute h’(INIT).

Step 3: Until INIT is labeled SOLVED or until h’ value becomes greater than FUTILITY, repeat the following procedures:

Step 3.1: Trace the labeled arcs from INIT and select for expansion one of the as yet unexpanded nodes that occurs on this path. Call the selected node NODE.

Step 3.2: Generate the successor of NODE. If there are none, then assign FUTILITY as the h’ value of NODE. This is equivalent to saying that NODE is not solvable. If there are successors, then for each one(called SUCCESSOR) that is not also an ancestor of the NODE, do the following.

Step 3.2.1: Add SUCCESSOR to GRAPH.

Step 3.2.2: If SUCCESSOR is a terminal node, label it SOLVED and assign it an h’ value of 0.

Step 3.2.3: If SUCCESSOR is not a terminal node, compute its h’ value.

Step 3.3: Propagate the newly discovered information up the graph by doing the following.

Let S be a set of nodes that have been labeled SOLVED or whose h’ values have been changed and so need to have values propagated back to their parents. Initialise S to NODE. Until S is empty, repeat the following steps.

Step 3.3.1: If possible, select from S a node, none of whose descendents in GRAPH occurs in S. If there is no such node, select any node from S. Call this node CURRENT and remove it from S.

Step 3.3.2: Compute the cost of each of the arcs emerging from CURRENT. The cost of each arc is equal to the sum of the h’ values of each of the nodes at the end of the arc plus whatever the cost of the arc itself is. Assign as CURRENT S new h’ value the minimum of the costs just computed for the arcs emerging from it.

Step 3.3.3: Mark the best path out of CURRENT by marking the arc that had the minimum cost as computed in the previous step.

Step 3.3.4: Mark CURRENT SOLVED if all of the nodes connected to it through the new labeled arc have been SOLVED.

Step 3.3.5: If CURRENT has been labeled SOLVED or if the cost of CURRENT was just changed, then its new status must be propagated back up the graph. So add all of the ancestors of CURRENT to S.

Step 4: Display the solved path.

Step 5: Stop.

**Conclusion:**

Python program to implement AO\* algorithm is successfully implemented.