**PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE**

**ACADEMIC YEAR: 2021-22**

## **DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT**

**CLASS: T.E. SEMESTER: II**

**SUBJECT: LP2**

|  |  |
| --- | --- |
| **ASSINGMENT NO.** | 2 |
| **TITLE** | A\* Algorithm |
| **PROBLEM STATEMENT /DEFINITION** | Implement A\* algorithm for any game search problem |
| **OBJECTIVE** | 1. To understand the informed and un informed searching techniques. 2. To analyze A\* Algorithm With respect to completeness, optimality, time complexity and space complexity 3. To apply A\* algorithm for one of the gaming application. |
| **OUTCOME** | The student will be able to   1. Learn how A\* algorithm works. 2. Know advantages and disadvantages of Heuristic search based on different parameters. 3. Apply A\* algorithm for various AI Problems |
| **S/W PACKAGES AND**  **HARDWARE APPARATUS USED** | Hardware- 64 bit Windows OS and Linux  Software- C/C++/Java/Python |
| **REFERENCES** | 1. Stuart Russell and Peter Norvig, “Artificial Intelligence: A Modern Approach”, Third  edition, Pearson, 2003, ISBN :10: 0136042597  2. Deepak Khemani, “A First Course in Artificial Intelligence”, McGraw Hill Education(India),  2013, ISBN : 978-1-25-902998-1  3. Elaine Rich, Kevin Knight and Nair, “Artificial Intelligence”, TMH, ISBN-978-0-07-  008770-5 |
| **STEPS** |  |
| **INSTRUCTIONS FOR**  **WRITING JOURNAL** | 1. Date  2. Assignment no.  3. Problem definition  4. Learning objective  5. Learning Outcome  6. Concepts related Theory  7. Algorithm  8. Test cases  10. Conclusion/Analysis |

**Prerequisites:**

**Concepts related Theory:**

**Informed search strategy**: Informed search uses problem-specific knowledge beyond the definition of the problem itself—can find solutions more efficiently than uninformed strategy. Informed search algorithm uses the idea of heuristic, so it is also called Heuristic search.

**A\*** is a graph traversal and path search algorithm, which is often used in many fields of computer science due to its completeness, optimality, and optimal efficiency

A\* is an informed search algorithm, or a best-first search, meaning that it is formulated in terms of weighted graphs: starting from a specific starting node of a graph, it aims to find a path to the given goal node having the smallest cost

The choice of an appropriate heuristic evaluation function, **h(n)**, is still crucial to the behavior of this algorithm.

In general, we want to choose a heuristic evaluation function **h(n)** which is as close as possible to the *actual* cost of getting to a goal state.

If we can choose a function **h(n)** which never *overestimates* the actual cost of getting to the goal state, then we have a very useful property. Such a **h(n)** is said to be *admissible*.

Best-first search, where the agenda is sorted according to the function **f(n) = g(n) + h(n)** and where the function **h(n)** is admissible, can be proven to always find an optimal solution. This is known as *Algorithm A\*.*

**Calculating heuristics**

Heuristics are rules of thumb that may find a solution but are not guaranteed to.Heuristic functions have also been defined as evaluation functions that **estimate** the cost from a node to the goal node. The incorporation of domain knowledge into the search process by means of heuristics is meant to speed up the search process.

Heuristic functions are not guaranteed to be completely accurate. Heuristic values are greater than and equal to zero for all nodes. Heuristic values are seen as an approximate cost of finding a solution. A heuristic value of zero indicates that the state is a goal state.

Heuristic functions are the most common form in which additional knowledge of the problem is imparted to the search algorithm.

**Heuristics function:** Heuristic is a function which is used in Informed Search, and it finds the most promising path. It takes the current state of the agent as its input and produces the estimation of how close agent is from the goal. The heuristic method, however, might not always give the best solution, but it guaranteed to find a good solution in reasonable time. It is represented by h(n)

G is the movement cost (in number of squares for this game) from the start point A to the current position

H is the estimated movement cost (in number of squares for this game) from the current square to the destination point. The closer the estimated movement cost is to the actual cost, the more accurate the final path will be. If the estimate is off, it is possible the path generated will not be the shortest

**Path Sore**

We’ll give each square a score ***G + H*** where:

***G*** is the movement cost from the start point A to the current square. So for a square adjacent to the start point A, this would be 1, but this will increase as we get farther away from the start point.

***H*** is the estimated movement cost from the current square to the destination point (we’ll call this point B for Bone!) This is often called the heuristic because we don’t really know the cost yet – it’s just an estimate.

You may be wondering what we mean by “movement cost”. Well in this game it will be quite simple – just the number of squares.

However, keep in mind that you can make this different for our game. For example:

If you allowed diagonal movement, you might make the movement cost a bit bigger for diagonal moves.

If you had different terrain types, you might make some cost more to move through – for example a swamp, water, or a Catwoman poster ;-)

That’s the general idea – now let’s dive into more specifics about figuring out G and H.

**The A\* Algorithm**

* So now that you know how to compute the score of each square (we’ll call this ***F***, which again is equal to ***G + H***), let’s see how the A\* algorithm works.
* The cat will find the shortest path by repeating the following steps:
* Get the square on the open list which has the lowest score. Let’s call this square S.
* Remove S from the open list and add S to the closed list.
* For each square T in S’s walkable adjacent tiles:
  + ***If T is in the closed list***: Ignore it.
  + ***If T is not in the open list***: Add it and compute its score.
  + ***If T is already in the open list***: Check if the F score is lower when we use the current generated path to get there. If it is, update its score and update its parent as well.
* Don’t worry if you’re still a bit confused about how this works – we’ll walk through an example so you can see it working step by step! :]

Consider the following route finding problem: To find route from Arad to Bucharest

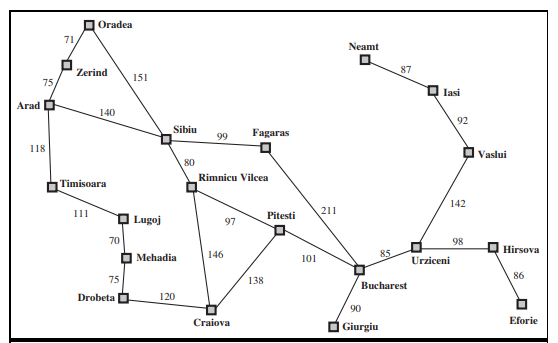


Figure 1: A simplified road map of part of Romania.

The straight line distance is used as heuristic, which can be called hSLD . The goal is Bucharest. Following are straight line distance to Bucharest:

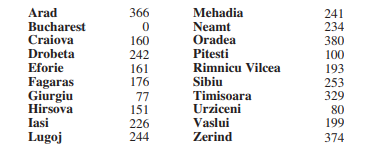


Figure 2: Values of hSLD—straight-line distances to Bucharest.

Example, hSLD (In(Arad)) = 366.

Notice that the values of hSLD cannot be computed from the problem description itself. Moreover, it takes a certain amount of experience to know that hSLD is correlated with actual road distances and is, therefore, a useful heuristic.

*Figure 3* shows the progress of a greedy best-first tree search using hSLD to find a path from Arad to Bucharest.

1. The first node to be expanded from Arad will be Sibiu because it is closer to Bucharest than either Zerind or Timisoara.
2. The next node to be expanded will be Fagaras because it is closest.
3. Fagaras in turn generates Bucharest, which is the goal.

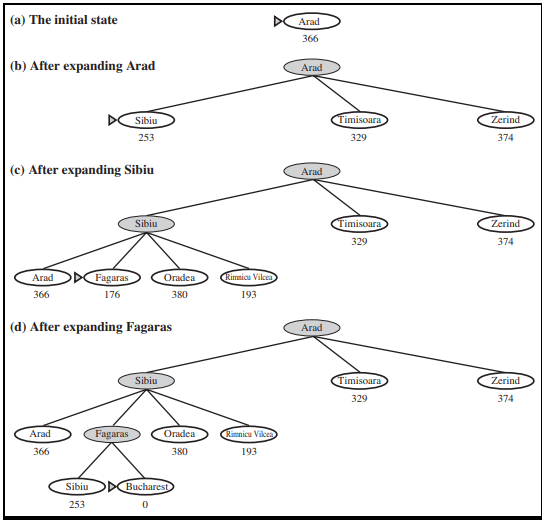


Figure 3: Stages in a greedy best-first tree search for Bucharest with the straight-line distance heuristic hSLD . Nodes are labeled with their h-values.

**Analysis**

A search algorithm’s performance can be analyzed in four ways:

• Completeness: Is the algorithm guaranteed to find a solution when there is one?

• Optimality: Does the strategy find the optimal solution

• Time complexity: How long does it take to find a solution?

• Space complexity: How much memory is needed to perform the search?

**Optimality:**. Yes

**Completeness: Yes (unless there are infinitely many nodes with f *≤ f(G)* )**

**Time and Space complexity:** Exponential time growth and Keeps all nodes in memory

**Conclusion:**

Thus A\* algorithm was studied and implemented.

**Review Questions**:

What are informed search techniques?

What are uninformed search techniques?

What is heuristic function?

What is space and time complexity of A\* algorithm?

Comment on optimality of of A\* algorithm

Is A\* complete? Explain.

What do you mean by Admissible heuristics?