**PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE**

**ACADEMIC YEAR: 2021-22**

## **DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT**

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

**SUBJECT: LP2**

|  |  |
| --- | --- |
| **ASSINGMENT NO.** | 3 |
| **TITLE** | Greedy Search Algorithm |
| **PROBLEM STATEMENT /DEFINITION** | Implement Greedy search algorithm for any one of the following application:  I                      Selection Sort  II                   Minimum Spanning Tree  III                 Single-Source Shortest Path Problem  IV                Job Scheduling Problem  V                   Prim's Minimal Spanning Tree Algorithm  VI                Kruskal's Minimal Spanning Tree Algorithm  VII              Dijkstra's Minimal Spanning Tree Algorithm |
| **OBJECTIVE** | 1. To understand the informed search technique- Greedy best first search algorithm 2. To analyze Greedy Search Algorithm according to completeness, optimality, time complexity and space complexity 3. To apply Greedy Best Search algorithm for one of the above mentioned application. |
| **OUTCOME** | The student will be able to   1. Learn how search proceeds during a greedy search. 2. Know advantages and disadvantages of greedy search according to analysis parameters. 3. Apply greedy search to solve 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.

**Greedy best-first search**

Greedy best-first search algorithm always selects the path which appears best at that moment.. Best-first search is an instance of the general TREE-SEARCH or GRAPH-SEARCH algorithm in which a node is selected for expansion based on an evaluation function, f(n). The evaluation function is constructed as a cost estimate, so the node with the lowest evaluation is expanded first. The choice of f determines the search strategy. Most best-first algorithms include as a component of f a heuristic function, denoted by h(n):

**h(n) = estimated cost of the cheapest path from the state at node n to a goal state.**

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

Greedy best-first search tries to expand the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly. Thus, it evaluates nodes by using just the heuristic function; that is, f(n) = h(n).

**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)

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

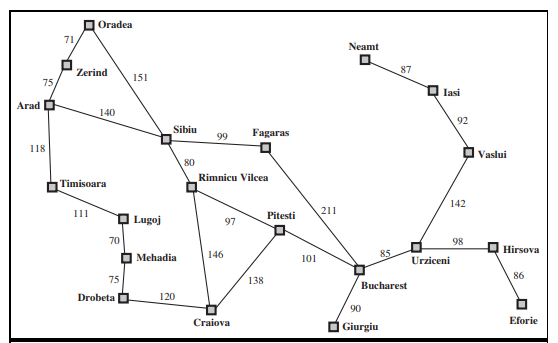


Figure 1: A simplified road map of part of Romania.

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

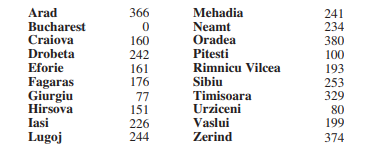


Figure 2: Values of hSLD—straight-line distances to Bucharest.

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.

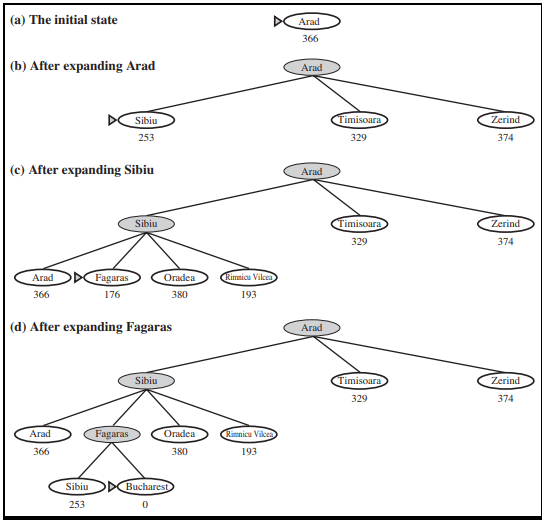


Figure 3: Stages in a greedy best-first tree search for Bucharest with the straight-line distance heuristic hSLD . Nodes are labeled with their h-values.

**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:** For this particular problem, greedy best-first search using hSLD finds a solution without ever expanding a node that is not on the solution path; hence, its search cost is minimal. It is not optimal., however: the path via Sibiu and Fagaras to Bucharest is 32 kilometers longer than the path through Rimnicu Vilcea and Pitesti. This shows why the algorithm is called “greedy”—at each step it tries to get as close to the goal as it can. Thus greedy search does not guarantee optimal solution.

**Completeness:** The greedy search algorithm is complete in finite state space but not in infinite one.

**Time and Space complexity:** The time and space complexity of greedy search depends on the accuracy of heuristic function. The worst-case time and space complexity is O(bm), where m is the maximum depth of the search space.

**Algorithm:**

**Step 1:** Place the starting node into the FRONTIER list.

**Step 2:** If the FRONTIER list is empty, Stop and return failure.

**Step 3:** Remove the node n, from the FRONTIER list which has the lowest value of h(n), and places it in the EXPLORED list.

**Step 4:** Expand the node n, and generate the successors of node n.

**Step 5:** Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.

**Step 6:** For each successor node, algorithm checks for evaluation function h(n), and then check if the node has been in either FRONTIER or EXPLORED list. If the node has not been in both list, then add it to the FRONTIER list.

**Step 7:** Return to Step 2.

**Conclusion:**

Thus greedy search 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 greedy search?

Comment on optimality of of greedy search.

Is greedy search complete? Explain.