Datta Meghe College of Engineering **Airoli, Navi Mumbai**	**DEPARTMENT OF ELECTRONICS ENGINEERING** **ACADEMIC YEAR : 2021 – 22 (TERM – II)** **List of Experiments** **Course Name : Artificial Intelligence** **Course Code : CSC604/CSL604**

Sr. No	Name of experiment	COs Covered	Page No.	Date of Performance	Date of Submission	Marks & Signature
1	One case study on AI applications published in IEEE/ACM/Springer or any prominent journal.	CSC 604.6				
2	Assignments on State space formulation and PEAS representation for various AI applications	CSC 604.3				
3	Program on uninformed search methods.	CSC 604.3				
4	Program on informed search methods.	CSC 604.3				
5	Program on Game playing algorithms.	CSC 604.3				
6	Program for first order Logic(Mini Project)	CSC 604.4 CSC 604.6				
7	Planning Programming	CSC 604.5				
8	Implementation for Bayes Belief Network	CSC 604.5				
9	Assignment 1	CSC 604.1, CSC 604.2 ,CSC 604.3				
10	Assignment 2	CSC 604.4, CSC 604.5, CSC 604.6				

This is to certify that Mr. / Miss _____ of _____ Roll No. _____ has performed the Experiments / Assignments / Tutorials / Case Study Work mentioned above in the premises of the institution.

_____

Practical Incharge

DATTA MEGHE COLLEGE OF ENGINEERING,

AIROLI, NAVI MUMBAI

DEPARTMENT OF COMPUTER ENGINEERING

Institute Vision : To create value - based technocrats to fit in the world of work and research

Institute Mission : To adapt the best practices for creating competent human beings to work in the world of technology and research.

Department Vision : To provide an intellectually stimulating environment for education, technological excellence in computer engineering field and professional training along with human values.

Department Mission :

M1: To promote an educational environment that combines academics with intellectual curiosity.

M2: To develop human resource with sound knowledge of theory and practical in the discipline of Computer Engineering and the ability to apply the knowledge to the benefit of society at large.

M3: To assimilate creative research and new technologies in order to facilitate students to be a lifelong learner who will contribute positively to the economic well-being of the nation.

Program Educational Objectives (PEO):

PEO1: To explicate optimal solutions through application of innovative computer science techniques that aid towards betterment of society.

PEO2: To adapt recent emerging technologies for enhancing their career opportunity prospects.

PEO3: To effectively communicate and collaborate as a member or leader in a team to manage multidisciplinary projects

PEO4: To prepare graduates to involve in research, higher studies or to become entrepreneurs in long run.

Program Specific Outcomes (PSO):

PSO1 : To apply basic and advanced computational and logical skills to provide solutions to computer engineering problems

PSO2 : Ability to apply standard practices and strategies in design and development of software and hardware based systems and adapt to evolutionary changes in computing to meet the challenges of the future.

PSO3: To develop an approach for lifelong learning and utilize multi-disciplinary knowledge required for satisfying industry or global requirements.

Program Outcomes as defined by NBA (PO)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

DATTA MEGHE COLLEGE OF ENGINEERING

Department of Computer Engineering

Course Name: Artificial Intelligence Lab (R-19)

Course Code: CSC604

Year of Study: T.E., Semester: VI

Course Outcomes

CSC604.1	Ability to develop a basic understanding of AI building blocks
CSC 604.2	Ability to identify the suitable intelligent agent
CSC 604.3	Ability to choose an appropriate problem solving method and to analyze the strength and weaknesses of AI approaches to knowledge– intensive problem solving.
CSC 604.4	Ability to choose appropriate knowledge representation technique and to design models for reasoning with uncertainty as well as the use of unreliable information.
CSC 604.5	To understand the role of planning and learning in intelligent systems
CSC 604.6	Ability to design and develop AI applications in real world scenarios

DATTA MEGHE COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

ACADEMIC YEAR 2021-22 (TERM II)

SUBJECT: ARTIFICIAL INTELLIGENCE

SEM: VI

RUBRICS FOR GRADING EXPERIMENTS

Rubric Number	Rubric Title	Criteria	Marks* (out of 10)
R1	Timeliness	On-time	2
		Delayed by not more than a Week	1
R2	Knowledge	Able to answer all questions	4
		Able to answer most of the questions	3
		Able to answer some of the questions	2
		Able to answer very few of the questions	1
R3	Implementation	Correct Logic, Well formatted and Structured program	4
		Correct Logic, formatted and Structured program	3
		Correct Logic and Structured program	2
		Correct Logic	1

DATTA MEGHE COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

ACADEMIC YEAR 2021-22 (TERM II)

SUBJECT: ARTIFICIAL INTELLIGENCE

SEM: VI

RUBRICS FOR GRADING MINI PROJECT

Rubric Number	Rubric Title	Criteria	Marks* (out of 10)
R1	Design of Knowledge base and Conversion to FOL	Proper Design and Conversion with explanation	2
		Proper Design and Conversion	1
R2	Implementation in Prolog	Implementation of Complete Solution	2
		Implementation of Partial Solution	1
R3	Clarity of Concept	Clear understanding	1
		Concepts are not clear	0

DATTA MEGHE COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

ACADEMIC YEAR 2021-22 (TERM II)

SUBJECT: ARTIFICIAL INTELLIGENCE

SEM: VI

RUBRICS FOR GRADING CASE STUDY

Rubric Number	Rubric Title	Criteria	Marks* (out of 10)
R1	Understanding of Problem Definition & Objectives	Clear Understanding of Problem & Objectives	4
		Fair Understanding of Problem & Objectives	3
		Clear Understanding of Problem	2
		Fair Understanding of Problem	1
R2	Literature Review	In Depth Literature Survey with Findings	4
		In Depth Literature Survey	3
		Fair Literature Survey	2
		Not enough Literature Survey	1

R3	**Presentation & Communication Skills**	**Presented and communicated very well**	**2**
		Presented and communicated poorly	**1**

DATTA MEGHE COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

ACADEMIC YEAR 2021-22 (TERM II)

SUBJECT: ARTIFICIAL INTELLIGENCE

SEM: VI

RUBRICS FOR GRADING ASSIGNMENT

Rubric Number	Rubric Title	Criteria	Marks* (out of 10)
R1	**Timeline**	**On-time**	**1**
		Delayed	**0**
R2	**Knowledge**	**Able to answer all questions**	**2**
		Able to answer some of the questions	**1**
R3	**Neatness**	**Well Written**	**2**
		Fairly Written	**1**

EXPERIMENT NO. 1

Date of Performance :

Date of Submission :

AIM: Case study of any Artificial Intelligence Application.

THEORY:

1. To carry out case study for any application Artificial Intelligence, download the paper from IEEE/ACM/Springer/Elsevier or any other prominent journal not older than 2019.

2. Download the references given in that paper for literature survey.

3. Carry out the summarized findings and identify the gaps present in the study.

4. Provide the solutions for gaps and suggest what can be done to overcome the limitations of the present study.

5. Prepare a report containing:

Abstract

An undeviating use of Artificial Intelligence and its subsets in agriculture can serve to be an embodiment of a shift in the way that farming is exercised during the present time. The agricultural domain faces countless obstacles for instance disease, improper soil analysis, pest infestation, irrigation, inadequate drainage, and a lot more. These challenges lead to dangerous environmental hazards and intense crop loss as a result of using redundant chemicals. The realm of Artificial Intelligence along with its meticulous learning abilities has evolved to form a key approach for dealing with diverse farming-related issues. The United Nations Food and Agriculture Organization asserts that the whole world's population would rise by an additional 2 billion in 2050, whereas at that time, the increased land area for farming will solely account for 4%. In the aforementioned situation, more coherent cultivation practices need to be accomplished by utilizing modern technological advancements and unraveling the ongoing barriers in farming.

Introduction

The extensive predicament in agriculture is yield protection, insufficient use of chemicals, pest and disease infestation, inadequate irrigation and drainage, weed control, and much more. A similar field of much priority is farming where around 30.7% of the entire world's population is precisely committed to 2781 million hectares of agricultural land
Artificial intelligence denotes the imitation of human intelligence in machines that are designed to

think like humans and replicate their behavior such as learning and problem-solving. Agriculture is a vigorous field where circumstances cannot be concluded to provide a collective explanation. Artificial Intelligence techniques have empowered us to seize the elaborate specifics of every circumstance and deliver the answer that is the finest fist for that specific problem. Progressively appropriate compound problems are being unraveled through the progress of numerous AI techniques.
The core focus of this paper is on the significant Artificial Intelligence (AI) techniques that are used to face the issues in agriculture. The three essential AI techniques: Expert Systems, Fuzzy Systems, and Artificial Neural Networks are measured as the concentrated areas. This paper discourses the use of Artificial Intelligence techniques in a vast subdomain of farming to capture the measured growth of the agro-intelligent systems.

Literature Review

Crop models and decision-making tools are being progressively used in the agricultural field to improve production and resource use efficiency, there is an enormous scope for Artificial Intelligence to revolutionize agriculture by integrating advanced technologies to forecast agricultural productivity [2]. The prospect of agriculture and the farming industry relies heavily on inventive concepts and technological developments to intensify yields and improved utilization of resources with the help of unconventional computing tools.
To solve the current difficulties in agriculture ,many methodologies have been proposed, starting from databases to the decision support systems [3]. Among these elucidations, systems that make use of Artificial Intelligence are tremendous performers as long as the robustness and accuracy are concerned.In addition to this, the disorder of supply systems and food production is threatened due to the COVID-19 pandemic, [4]. Such factors are a threat to the sustainability of the environment, of the present and the future food source chain [5]. Substantial inventions are always a necessity to stay ahead of the persistent climate change [6]. The understandable problem here is by what means to harvest adequate quantity of food for the ever growing population. The research scientists are continuously applying state-of-the-art expertise and discovering new ways to assimilate them into the agricultural system [7]. Climatic change, increase in the cost of production , decreasing water supply for irrigation and inclusive drop in the farm workforce have caused a lot of trouble to the agriculture production systems over the last few decades [7].
Proposed system
A lot of data is needed by the AI system to train the machines, for making accurate predictions. In cases of enormous land area for farming, it is easy to collect spatial data whereas the accomplishment of temporal data is challenging. It is also problematic to develop the knowledge-based rules and put them in a correct sequence for a huge number of parameters. Numerous crop-specific information could be attained once in a year only when the crops are grown. Because it takes time for the database to mature, it comprises a considerable amount of stretch to build a robust AI ML model. This is the foremost purpose for the application of AI in agriculture-related products like pesticides, fertilizers, and seeds.
Another decisive factor is the expensive cost of many cognitive solutions for agriculture that are readily accessible in the market. The AI-inspired solutions have to be more feasible to ensure that the technology influences the agricultural community. The AI cognitive solutions if offered in some open-source platform then that would help to make the solutions additionally reasonably priced, which will then lead to earlier adoption and better insight among the farmers.
Conclusion
The expert systems based on rules were comprehensively utilized from the 1980s to the 1990s, whereas from the onset of 1990, fuzzy inference systems and the artificial neural network took the major role. In the current years, the practice of hybrid systems such as image processing or neuro-fuzzy combined with artificial neural networks is in use.
The usage of AI will benefit the farmers to attain their objective of a healthier harvest by taking improved decisions in the field. The supremacy of data can be utilized more resourcefully to predict risk and analyze scenarios and therefore take an action before hunger lingers as a humanitarian crisis, which in turn can help in the complete development of the world because food is the most important necessity for human beings. The substantial methods can assure the farmers with suitable field management and healthier crops. Timely information is provided by the AI from the right channels and can therefore build resilience among the

users.

References

[1] E.Rich and Kevin Knight. "Artificial intelligence", New Delhi: McGraw-Hill, 1991.
[2] Dutta, Suchandra & Rakshit, Shantanu & Chatterjee, Dvyan. (2020). Use of Artificial Intelligence in Indian Agriculture. 1. 65-72.
[3] Thorpe, K. W. , Ridgway, R. L. , & Webb, R. E. (1992). A computerized data management and decision support system for gypsy moth management in suburban parks. Computers and electronics in agriculture, 6(4), 333-345.
[4] J.L. Outlaw, B.L. Fischer, D.P. Anderson, S.L. Klose, L.A. Ribera, J.M. Raulston, G.M. Knapek, B.K. Herbst, J.R. Benavidez, H.L. Bryant, D.P. Ernstes COVID-19 Impact on Texas Production Agriculture Agricultural & Food Policy Center, Texas A&M University Research (2020).
[5] M. A. Andersen, J. M. Alston, P. G. Pardey, A. Smith A century of U.S. productivity growth: a surge then a slowdown Am J Agric Econ, 93 (2018), pp. 1257-1277.
[6] J. Hatfield, G. Takle, R. Grotjahn, P. Holden, R.C. Izaurralde, T. Mader, E. Marshall, D. Liverman Ch. 6: Agriculture J.M. Melillo, T. Richmond, G.W. Yohe (Eds.), Climate change in the United States: The Third National Climate Assessment, U.S. Global Change Research Program (2014), pp. 50-174.
[7] Jinha Jung, Murilo Maeda, Anjin Chang, Mahendra Bhandari, Akash Ashapure, Juan Landivar-Bowles, The potential of remote sensing and artificial intelligence as tools to improve the resilience of agriculture production systems, Current Opinion in Biotechnology, Volume 70, 2021, Pages 15-22.

SIGN AND REMARK

DATE

R1 (4 Marks)	R2 (4 Marks)	R3 (2 Marks)	Total (10 Marks)	Signature

EXPERIMENT NO. 2

Date of Performance :

Date of Submission :

AIM: Assignments on State space formulation and PEAS representation for various AI applications.

THEORY:

1.1 Specifying Task Environment

Task environments, which are essentially the "problems" to which rational agents are the "solutions." In designing an agent, the first step must always be to specify the task environment as fully as possible with the help of PEAS (Performance, Environment, Actuators, Sensors) description.

1.2 Example of PEAS

Problem Definition:

An agricultural robot is a robot deployed for agricultural purposes. The main area of application of robots in agriculture today is at the harvesting stage. According to Verified Market Research, the agricultural robots market is expected to reach $11.58 billion by 2025.

PEAS Descriptors:

PEAS stands for Performance measure, Environment, Actuator, Sensor.

The PEAS description for the above-mentioned problem description is as follows:

Performance Measure: Make Goal, time, field accuracy

Environment: Farm, soil, water, crops

Actuator: Navigator, Legs of agriculture robot, wheels

Sensor: Camera, Temperature, Chemical Sensors

A goal directed agent needs to achieve certain goals. Such an agent selects its actions based on the goal it has. Many problems can be represented as a set of states and a set of rules of how one state is transformed to another. Each state is an abstract representation of the agent's environment. It is an abstraction that denotes a configuration of the agent.

1.3 Problem Formulation

• Formulate a problem as a state space search by showing the legal problem states, the legal operators, and the initial and goal states .

• A state is defined by the specification of the values of all attributes of interest in the world

• An operator changes one state into the other; it has a precondition which is the value of certain attributes prior to the application of the operator, and a set of effects, which are the attributes altered by the operator

• The initial state is where you start

• The goal state is the partial description of the solution

An initial state is the description of the starting configuration of the agent An action or an operator takes the agent from one state to another state which is called a successor state. A state can have a number of successor states. A plan is a sequence of actions. The cost of a plan is referred to as the path cost. The path cost is a positive.

Problem formulation means choosing a relevant set of states to consider, and a feasible set of operators for moving from one state to another. Search is the process of considering various possible sequences of operators applied to the initial state, and finding out a sequence which culminates in a goal state.

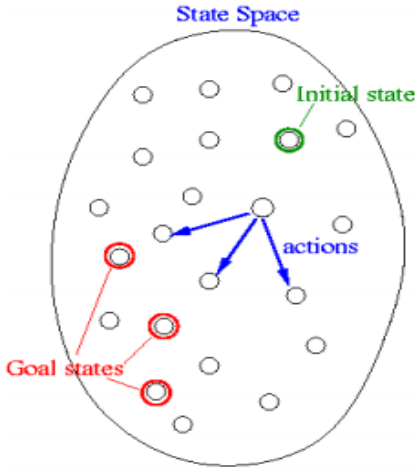
Search Problem is formally described as following:

• S: the full set of states

• s0 : the initial state

• A:S→S is a set of operators

• G is the set of final states. Note that G ⊆S



The search problem is to find a sequence of actions which transforms the agent from the initial state to a goal state g∈G. A sequence of states is called a path. The cost of a path is a positive number. In many cases the path cost is computed by taking the sum of the costs of each action.

1.4 Example of Problem formulation

Problem Formulation:

Initial state: One of tasks at farm and agriculture robot at one of the corners of the field.

Successor Function: Identify tasks in the field.

Goal test: No task at farm

Path cost: Time taken for harvesting and picking, weed control , etc.

CONCLUSION:

Thus, we have understood the concept of PEAS description and problem formulation and have implemented it by identifying a problem and writing a PEAS description and problem formulation for the same.

SIGN AND REMARK:

DATE:

R1 (2 Marks)	R2 (4 Marks)	R3 (4 Marks)	Total (10 Marks)	Signature

EXPERIMENT 3

Date of Performance :

Date of Submission :

AIM: To implement Breadth First Search as Uninformed Search.

S/W USED : C/C++/Java/Prolog

THEORY:

Breadth First Search

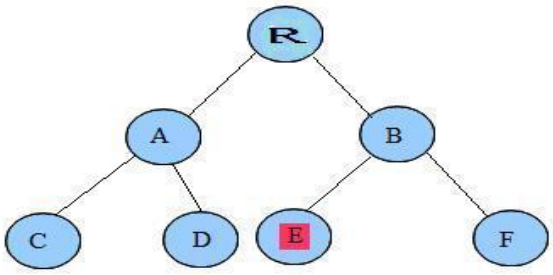
Breadth First Search (BFS) searches breadth-wise in the problem space. Breadth-First search is like traversing a tree where each node is a state which may a be a potential candidate for solution. It expands nodes from the root of the tree and then generates one level of the tree at a time until a solution is found. It is very easily implemented by maintaining a queue of nodes. Initially the queue contains just the root. In each iteration, node at the head of the queue is removed and then expanded. The generated child nodes are then added to the tail of the queue.

Algorithm: Breadth-First Search

1. Create a variable called NODE-LIST and set it to the initial state.
2. Loop until the goal state is found or NODE-LIST is empty.
    a. Remove the first element, say E, from the NODE-LIST. If NODE-LIST was empty then quit.
    b. For each way that each rule can match the state described in E do:

i) Apply the rule to generate a new state.
ii) If the new state is the goal state, quit and return this state.
iii) Otherwise add this state to the end of NODE-LIST

Since it never generates a node in the tree until all the nodes at shallower levels have been generated, *breadth-first search* always finds a shortest path to a goal. Since each node can be generated in constant time, the amount of time used by Breadth first search is proportional to the number of nodes generated, which is a function of the branching factor b and the solution d. Since the number of nodes at level d is b^d, the total number of nodes generated in the worst case is $b + b^2 + b^3 + \ldots + b^d$ i.e. $O(b^d)$, the asymptotic time complexity of breadth first search.



Breadth First Search

Look at the above tree with nodes starting from root node, R at the first level, A and B at the second level and C, D, E and F at the third level. If we want to search for node E then BFS will search level by level. First it will check if E exists at the root. Then it will check nodes at the second level. Finally it will find E a the third level.

Advantages Of Breadth-First Search

1. Breadth first search will never get trapped exploring the useless path forever except if the no. of successors to any are infinite
2. BFS is **complete** i.e.If there is a solution, BFS will definitely find it out.
3. BFS is **optima**l i.e. If there is more than one solution then BFS can find the minimal one that requires less number of steps.

Disadvantages Of Breadth-First Search

1. The main drawback of Breadth first search is its memory requirement. Since each level of the tree must be saved in order to generate the next level, and the amount of memory is proportional to the number of nodes stored, the space complexity of BFS is $O(b^d)$. As a result, BFS is severely space-bound in practice so will exhaust the memory available on typical computers in a matter of minutes.
2. If the solution is farther away from the root, breath first search will consume lot of time.

CONCLUSION:

We have successfully implemented Breadth First Search as Uninformed Search. Breadth Search Algorithm comes with some great advantages to recommend it. One of the many applications of the BFS algorithm is to calculate the shortest path. It is also used in networking to find neighboring nodes and can be found in social networking sites, network broadcasting, and garbage collection.

PROGRAM:

```
def BFS(graph,start,dest) -> list(): #Input parameters for this method are

#1.Graph in which we're going to search for our destination(dest) node

#2.start which is our start node and dest which is our destination node

queue = list()

visited = list()

queue.append(start)

print('Visited',start)

result = ["Not reachable",list()]

while queue:

node = queue.pop(0)

visited.append(node)

if node==dest:
```

```
print('Destination node found',node)
result[0] = 'Reachable'
break
print(node,'Is not a destination node')
for child in graph[node]:
if child not in visited:
queue.append(child)
result[1] = visited
return result

graph = {
'A': ['B', 'C'],
'B': ['A', 'D', 'E'],
'C': ['A', 'F', 'G'],
'D': ['B', 'E'],
'E': ['B', 'D'],
'F': ['C', 'H'],
'G': ['C'],
'H': ['F']
}
result = BFS(graph, "A", "E")
print(result[0])
print("Path used to traverse :-" , result[1])
```

OUTPUT:

```
Visited A
A Is not a destination node
B Is not a destination node
C Is not a destination node
D Is not a destination node
```

Destination node found E

Reachable

Path used to traverse :- ['A', 'B', 'C', 'D', 'E']

SIGN AND REMARK:

DATE:

R1 (2 Marks)	R2 (4 Marks)	R3 (4 Marks)	Total (10 Marks)	Signature

EXPERIMENT NO. 4

Date of Performance :

Date of Submission :

AIM: To implement A* Informed Search algorithm.

SOFTWARE USED: Java/Python

THEORY:

Informed search (Heuristics search) makes use of the fact that most problem spaces provide some information that distinguishes among states in terms of their likelihood of leading to a goal. This information is called a heuristic evaluation function. In other words, the goal of a heuristic search is to reduce the number of nodes searched in seeking a goal.

Most widely used best first search form is called A*, which is pronounced as A star. It is a heuristic searching method, and used to minimize the search cost in a given problem. It aims to find the least-cost path from a given initial node to the specific goal. It is an extended form of best-first search algorithm. Best firstsearch algorithm tries to find a solution to minimize the total cost of the search pathway, too. However, the difference from Best-First Search is that A* also takes into account the cost from the start, and not simply the local cost from the previously been node. Best-first search finds a goal state in any predetermined problem space. However, it cannot guarantee that it will choose the shortest path to the goal . For instance, if there are two options to chose from, one of which is a long way from the initial point but has a slightly shorter estimate of distance to the goal, and another that is very close to the initial state but has a slightly longer estimate of distance to the goal, best-first search will always choose to expand next the state with the shorter estimate. The A* algorithm fixes the best first search's this particular drawback.

In short, A* algorithm searches all possible routes from a starting point until it finds the shortest path or cheapest cost to a goal. The terms like shortest path, cheapest cost here refer to a general notion. It could be some other alternative term depending on the problem. A* evaluates nodes by combining g(n) and h(n).

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

f(n) is the total search cost, g(n) is actual lowest cost(shortest distance traveled) of the path from initial start point to the node n, h(n) is the estimated of cost of cheapest(distance) from the node n to a goal node. This part of the equation is also called heuristic function/estimation. At each node, the lowest f value is chosen to be the next step to expand until the goal node is chosen and reached for expansion. (Pearl & Korf, 1987). Whenever the heuristic function satisfies certain conditions, A* search is both complete and optimal (Russell & Norvig, 2003).

Characteristics of A* Search Algorithm:

Admissibility Strategies which guarantee optimal solution, if there is any solution, are called admissible. There are few items which are needed to be satisfied for A* to be admissible A* is optimal, if h(n) is an admissible heuristic. h*(n) = the true minimal cost to goal from n. A heuristic h is admissible if h(n) <= h*(n) for all states n.

Problem with A* Search Algorithm:

According to Pearl & Korf (1987) the main shortcoming of A*, and any best-first search, is its memory requirement. Because the entire open pathway list must be saved, A* is space-limited in practice and is no more practical than breadth first search. For large search spaces, A* will run out of memory

Algorithm:

A* Algorithm pseudocode

The goal node is denoted by node_goal and the source node is denoted by node_start

We maintain two lists:

OPEN and CLOSE:

OPEN consists on nodes that have been visited but not expanded (meaning that sucessors have not been explored yet). This is the list of pending tasks.

CLOSE consists on nodes that have been visited and expanded (sucessors have been explored already and included in the open list, if this was the case).

1 Put node_start in the OPEN list with f(node_start) = h(node_start) (initialization)

2 while the OPEN list is not empty {

3 Take from the open list the node node_current with the lowest

4 f(node_current) = g(node_current) + h(node_current)

5 if node_current is node_goal we have found the solution; break

6 Generate each state node_successor that come after node_current

7 for each node_successor of node_current {

8 Set successor_current_cost = g(node_current) + w(node_current, node_successor)

9 if node_successor is in the OPEN list {

10 if g(node_successor) ≤ successor_current_cost continue (to line 20)

11 } else if node_successor is in the CLOSED list {

12 if g(node_successor) ≤ successor_current_cost continue (to line 20)

13 Move node_successor from the CLOSED list to the OPEN list

14 } else {

15 Add node_successor to the OPEN list

16 Set h(node_successor) to be the heuristic distance to node_goal

17 }

18 Set g(node_successor) = successor_current_

19 Set the parent of node_successor to node_current

20 }

21 Add node_current to the CLOSED list

22 }

23 if(node_current != node_goal) exit with error (the OPEN list is empty)

PROGRAM:

```
from collections import deque

class Graph:
    def __init__(self, adjac_lis):
        self.adjac_lis = adjac_lis

    def get_neighbors(self, v):
        return self.adjac_lis[v]

    # This is heuristic function which is having equal values for all nodes
    def h(self, n):
        H = {
            'Arad': 366,
            'Zerind':374,
            'Sibiu': 253,
            'Timisoara': 329,
            'Oradea' : 380,
            'Fagaras': 178,
            'Rimnicu Vilcea': 193,
            'Pitesti': 98,
            'Craiova': 160,
            'Bucharest': 0,
            'Lugoj': 244,
            'Mehadia': 241,
            'Drobeta': 242,
            'Giurgiu': 77

        }
```

```
        return H[n]

    def a_star_algorithm(self, start, stop):
        # In this open_lst is a lisy of nodes which have been visited, but who's
        # neighbours haven't all been always inspected, It starts off with the start
#node
        # And closed_lst is a list of nodes which have been visited
        # and who's neighbors have been always inspected
        open_lst = set([start])
        closed_lst = set([])

        # poo has present distances from start to all other nodes
        # the default value is +infinity
        poo = {}
        poo[start] = 0

        # par contains an adjac mapping of all nodes
        par = {}
        par[start] = start

        while len(open_lst) > 0:
            n = None

            # it will find a node with the lowest value of f() -
            for v in open_lst:
                if n == None or poo[v] + self.h(v) < poo[n] + self.h(n):
                    n = v;

            if n == None:
                print('Path does not exist!')
                return None
```

```
            # if the current node is the stop
            # then we start again from start
            if n == stop:
                reconst_path = []

                while par[n] != n:
                    reconst_path.append(n)
                    n = par[n]

                reconst_path.append(start)

                reconst_path.reverse()

                print('Path found: {}'.format(reconst_path))
                return reconst_path

            # for all the neighbors of the current node do
            for (m, weight) in self.get_neighbors(n):
              # if the current node is not presentin both open_lst and closed_lst
                # add it to open_lst and note n as it's par
                if m not in open_lst and m not in closed_lst:
                    open_lst.add(m)
                    par[m] = n
                    poo[m] = poo[n] + weight

                # otherwise, check if it's quicker to first visit n, then m
                # and if it is, update par data and poo data
                # and if the node was in the closed_lst, move it to open_lst
                else:
                    if poo[m] > poo[n] + weight:
                        poo[m] = poo[n] + weight
```

```
                        par[m] = n

                        if m in closed_lst:
                            closed_lst.remove(m)
                            open_lst.add(m)

            # remove n from the open_lst, and add it to closed_lst
            # because all of his neighbors were inspected
            open_lst.remove(n)
            closed_lst.add(n)

        print('Path does not exist!')
        return None

adjac_lis = {
    'Arad': [('Sibiu', 140), ('Timisoara', 118), ('Zerind', 75)],
    'Zerind': [('Arad', 75), ('Oradea', 71)],
    'Oradea': [('Zerind', 71), ('Sibiu', 151)],
    'Sibiu' : [('Oradea', 151), ('Fagaras', 99), ('Rimnicu Vilcea',80)],
    'Rimnicu Vilcea': [('Sibiu', 80), ('Pitesti', 97), ('Craiova',146)],
    'Pitesti': [('Rimnicu Vilcea', 97), ('Craiova', 138), ('Bucharest', 101)],
    'Craiova': [('Drobeta',120,), ('Pitesti',138), ('Rimnicu Vilcea', 146)],
    'Bucharest': [('Pitesti', 101), ('Fagaras',211), ('Giurgiu', 90)],
    'Timisoara': [('Arad',118),('Lugoj',111)],
    'Lugoj': [('Timisoara', 111), ('Mehadia', 70)],
    'Mehadia': [('Lugoj', 70), ('Drobeta', 75)],
    'Drobeta': [('Mehadia', 75), ('Craiova', 120)],
    'Giurgiu': [('Bucharest', 90)],
    'Fagaras': [('Sibiu', 99),('Bucharest',211)]

}
```

```
graph1 = Graph(adjac_lis)
initial_state = input("Enter initial state: ")
goal_state = input("Enter goal state: ")
graph1.a_star_algorithm(initial_state, goal_state)
```

INPUT & OUTPUT:

```
Enter initial state: Timisoara
Enter goal state: Pitesti
Path found: ['Timisoara', 'Arad', 'Sibiu', 'Rimnicu Vilcea', 'Pitesti']

['Timisoara', 'Arad', 'Sibiu', 'Rimnicu Vilcea', 'Pitesti']
```

CONCLUSION:

We have successfully implemented A* Informed Search algorithm. A* is a very powerful algorithm with almost unlimited potential. However, it is only as good as its heuristic function, which can be highly variable considering the nature of a problem. It has found applications in many software systems, from Machine

Learning and Search Optimization to game development where NPC characters navigate through complex terrain and obstacles to reach the player.

SIGN AND REMARK

DATE

R1 (2 Marks)	R2 (4 Marks)	R3 (4 Marks)	Total (10 Marks)	Signature