Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that Mr./~~Miss~~ Shirish Shetty

of COMPS Department, Semester VI with

Roll No. 2103164 has completed a course of the necessary

experiments in the subject AI under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2023 - 2024

Teacher In- Charge

Head of the Department

Date 6/04/2023

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

Experiment No. 1

Title: Case study on AI applications published in IEEE/ACM/Springer/Elsevier or any prominent journal

Introduction

The case study mainly focus on the negative impact of human activities on the environment, particularly in relation to air pollution. It discusses the various forms of pollution, such as pollution of the seas, global warming, acid rain, and smog, and how these affect natural processes and environments. Furthermore, it emphasizes the invisible nature of air pollution and how it poses a significant risk to human health and ecological balance. The case study also touches on the sources of air pollution and the challenges in predicting and measuring pollutant distribution, highlighting the importance of accurate measurement through remote sensing.

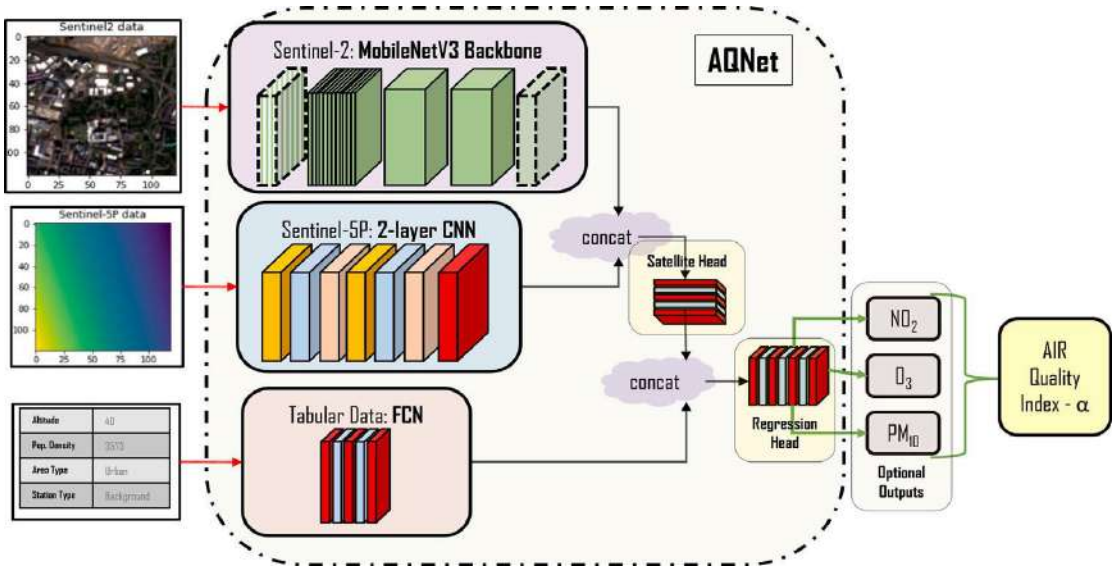
Domain

The domain for the case study is environmental science and technology, with a specific focus on air quality in urban environments and its impact on human health and the environment.

Problem identified

The case study identifies the negative impact of human activity on the environment through various forms of pollution, particularly air pollution, which affects natural processes and environments. It specifically mentions pollution of the seas, global warming, acid rain, and smog as key manifestations of human impact. The sources of air pollution, such as industrial processes, vehicle emissions, and other catalytic reactions, have been highlighted. Additionally, it discusses the challenge of accurately measuring air pollutant distribution, especially within cities, and emphasizes the role of remote sensing in providing accurate tools for measurement. The identified problems can be categorized as the impact of human activity on the environment through air pollution, the challenges in predicting and measuring air pollutant distribution, and the need for accurate measurement tools through remote sensing

Algorithmic Methodologies Applied



AQNet model Architecture. Named as AQNet-single if it returns only one of the three pollutants as output.

The algorithm mentioned in the case study is the AQNet, a multimodal AI architecture designed for air quality measurement using a combination of satellite imagery and tabular location information. The algorithm leverages machine learning (ML) and deep learning techniques to predict air quality metrics and pollutant concentrations. It utilizes different architectures to process and fuse from satellite imagery, tabular data, and ground concentration measurements.

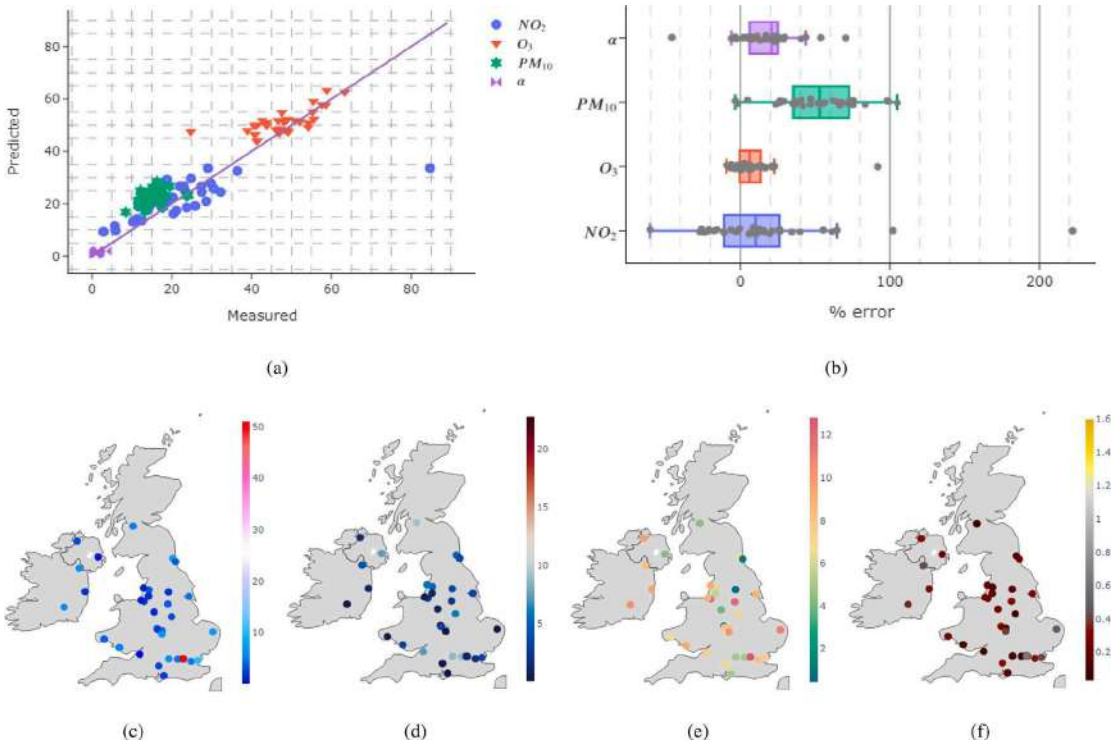
The key components of the algorithm include:

1. Satellite Data Processing: The algorithm preprocesses the satellite data by removing clouds and negative weather conditions. It then maps the data to a 10x10 km grid over Europe and matches the time of air quality measurements on the ground to the nearest time when the satellite passed over the monitoring station. This results in a high-resolution (10m) of the area.

2. Tabular Data Backbone: A 2-layer fully connected neural network (FCN) with ReLu activation functions processes the tabular data. This tabular data includes information such as altitude, population density, and categorical features indicating the type of monitoring (e.g., rural, suburban, urban; traffic, industrial, or background monitoring station). The network creates 32 features, which are then fused with the satellite data in the regression head.

3. Regression Head: This part of the algorithm is designed to create three outputs for the pollutants NO2, O3, and PM10. It allows for the simultaneous prediction of all three pollutants or individual predictions based on specific needs.

Solution



(a) Regression plots of true and predicted values of each pollutant and air quality - α. (b) Boxplots of % error between the measured and predicted values of each pollutant.(c)–(f) Absolute error scatter maps for NO2, O3, PM10 and air quality metric - α , respectively.

The case study discusses a model's predictions, noting an average overestimation of around 20% for all pollutants. It emphasizes that while O3 (ozone) predictions are closest to parity, there are significant errors in predicting NO2 and PM10 concentrations. The overestimation of PM10 predictions is suggested to be due to the small value range of PM10 measurements in the dataset, leading to a high percentage error.

Pros of the model mentioned in the case study include:

- Effective handling of large amounts of input data for predictive insight into environmental remote sensing problems.
- Highlighted potential for creating maps of PM2.5 distributions around Great Britain using random forests of decision tree algorithms.
- Discussion of using Sentinel 2 bands to create useful indices for air pollution prediction, indicating potential for improvement.

In summary, the model shows potential in providing predictive insights into air pollutant distributions, particularly through advanced statistical and machine learning techniques. It also demonstrates promise in leveraging satellite data for valuable information in environmental remote sensing problems.

Areas for Future Exploration

The case study outlines potential areas for future work and improvements in the field of environmental remote sensing and machine learning algorithms. It suggests several directions for research and development. Here are some of the key points from the case study:

1. Incorporating Meteorology: The case study highlights the importance of considering meteorological factors as inputs to the predictive models. The impact of weather conditions, such as rainfall, humidity, temperature, wind speed, and visibility on air pollutant transport, is significant. Therefore, future work could involve integrating meteorological data to enhance the predictivity of the models.

2. Geographical Information: The case study suggests that models could benefit from the inclusion of geographical information beyond the sovereign state where each pollutant monitoring station is located. This could involve encoding location features, such as latitude and longitude, to better predict pollution levels.

3. Expanding Feature Set: It is proposed that the models should include a broader range of features relating to land use and land cover (LULC) characteristics, potentially enhancing predictivity. This could involve explicit statements of land cover concentrations and the creation of useful indices for the purpose of air pollution prediction.

4. Utilization of Sentinel 2 Bands: The case study suggests considering the utilization of Sentinel 2 bands to create useful indices for air pollution prediction. Developed indices from the Sentinel 2 data might be incorporated into the tabular data to improve model performance.

5. Multi-Temporal Samples: Future research could address the challenge of sourcing suitable multi-temporal samples to run the model post-development, which is crucial for multimodal techniques in remote sensing.

Overall, the case study identifies the need to consider meteorological and geographical factors, expand the feature set, and explore the potential of satellite data for creating indices to improve the predictivity of machine learning models in environmental remote sensing. These directions for future work aim to enhance the accuracy and reliability of air quality prediction models.

Conclusion

The case study presents a comprehensive investigation of the AQNet algorithm for air quality measurement, utilizing satellite imagery and tabular location information. It introduces an air quality index and 3-pollutant dataset aimed at predicting air quality across different regions. The study delves into the influence of binary features, such as area and station type, on average pollutant concentrations, shedding light on the critical role of geographic and environmental factors in air quality. Moreover, the analysis acknowledges the potential impact of COVID-related lockdowns on air pollution reduction and highlights the need for further investigation into this area.

The case study also outlines potential areas for future work, focusing on the inclusion of meteorological and geographical data, expanding feature sets, and exploring the potential of satellite data for creating useful indices. The limitations and potential areas for improvement, such as the consideration of meteorological and geographical factors, suggest an ongoing need for refining air quality prediction models. Furthermore, the study critically examines the AQNet's use of Convolutional Neural Networks and raises the question of its suitability for the task. The case study concludes by emphasizing the ongoing pursuit of enhancements and the investigation of multi-task learning approaches for better performance in future releases of the AQNet.

In summary, the case study provides a detailed and nuanced exploration of the AQNet algorithm's development, its potential limitations, and future research directions, thus contributing valuable insights to the field of air quality prediction through remote sensing and machine learning.

Experiment No: 2

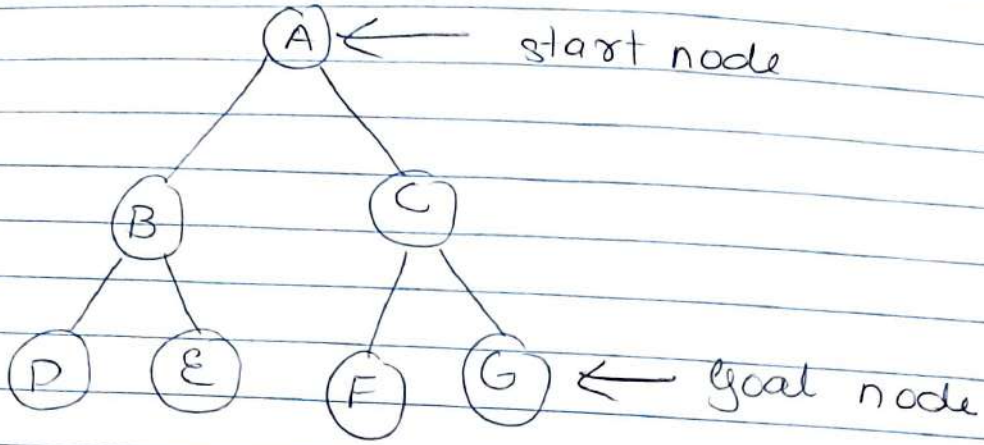
Title :- Implementing Depth First Search algorithm

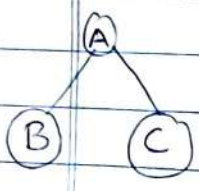
Problem Statement :- Consider a weighted graph connected with start node to goal node G
i) Apply DFS algorithm & show how state space tree will be created while searching a path from initial node to goal Node G
ii) Show contents of open & closed list after each iteration

Theory :- Uniformed search is a class of general purpose search algorithm which operates in brute force way uniformed search algorithm do not have additional information about state or search space other than how to traverse the tree, so it is also called a blind search.
following are the types i) DFS ii) BFS iii) DLS iv) DFID v) UCS.

DFS for a graph is similar to depth First search of tree. The only catch here is that, unlike tree, graph may contain cycle a node may be visited twice. To avoid that, use boolean visited array. A graph can more than one DFS traversal

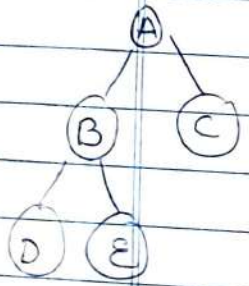
Let understand with an example





Visited : A

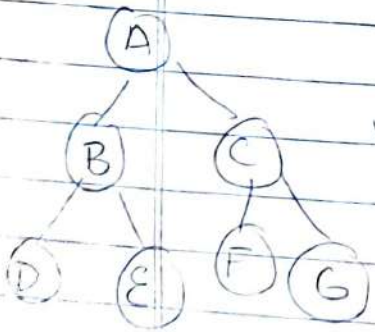
Stack: B, C — pop.



Visited A B

Stack: D, E, C — pop; E, C — pop; C — pop

① ② ③



visited ⇒ A B D E C

Stack:-

F, G — pop; G — pop.

Finally → A → B → D → E → C → F → G

Algorithm :→ DFS(G, v) {

```
stack S = { };
for each vertex v, set visited[v] = False;
push S, v;
while (S is not empty) do
    u = pop S;
    if (not visited[v]) then
        visited[v] = True;
        For each unvisited neighbour w of uv
        push S, w;
    end if
end while } END DFS()
```

Evaluation parameter :-

(i) Completness :- Not complete
(ii) Time complexity :- $O(b^m)$
(iii) Space :- $O(b \times m)$
(iv) Optimality :- Non optimal

Conclusion :- Learnt about DFS approach in AI

EXP 2

```
class Graph:
    def __init__(self, vertices):
        self.vertices = vertices
        self.graph = [[] for _ in range(vertices)]

    def add_edge(self, u, v):
        self.graph[u].append(v)

    def dfs_util(self, v, visited, goal):
        if v == goal:
            return True
        visited[v] = True
        for i in self.graph[v]:
            if not visited[i]:
                if self.dfs_util(i, visited, goal):
                    return True
        return False

    def dfs(self, start, goal):
        visited = [False] * self.vertices
        return self.dfs_util(start, visited, goal)

    def dls(self, start, goal, depth_limit):
        visited = [False] * self.vertices
        return self.dls_util(start, visited, goal, depth_limit)

    def dls_util(self, v, visited, goal, depth_limit):
        if depth_limit <= 0:
            return False
        if v == goal:
```

```
            return True
        visited[v] = True
        for i in self.graph[v]:
            if not visited[i]:
                if self.dls_util(i, visited, goal, depth_limit - 1):
                    return True
        return False

    def dfid(self, start, goal):
        depth = 0
        while True:
            if self.dls(start, goal, depth):
                return True
            depth += 1

def menu():
    print("Select Search Algorithm:")
    print("1. Depth-First Search (DFS)")
    print("2. Depth-Limited Search (DLS)")
    print("3. Iterative Deepening Depth-First Search (DFID)")
    print("4. Exit")

def main():
    vertices = 7  # Example graph with 7 vertices
    graph = Graph(vertices)
    graph.add_edge(0, 1)
    graph.add_edge(0, 2)
    graph.add_edge(1, 3)
    graph.add_edge(1, 4)
    graph.add_edge(2, 5)
    graph.add_edge(2, 6)
```

```
while True:
    menu()
    choice = int(input("Enter your choice: "))
    if choice == 1:
        start = int(input("Enter start node: "))
        goal = int(input("Enter goal node: "))
        if graph.dfs(start, goal):
            print("Goal node found using DFS")
        else:
            print("Goal node not found")
    elif choice == 2:
        start = int(input("Enter start node: "))
        goal = int(input("Enter goal node: "))
        depth_limit = int(input("Enter depth limit: "))
        if graph.dls(start, goal, depth_limit):
            print("Goal node found using DLS")
        else:
            print("Goal node not found within depth limit")
    elif choice == 3:
        start = int(input("Enter start node: "))
        goal = int(input("Enter goal node: "))
        if graph.dfid(start, goal):
            print("Goal node found using DFID")
        else:
            print("Goal node not found")
    elif choice == 4:
        print("Exiting...")
        break
    else:
        print("Invalid choice. Please try again.")
```

```
if __name__ == "__main__":
    main()
```

output:

```
choice 1
start node 0
goal node 6
goal found
```

Experiment No 3

Aim :- Implement Breadth First search algorithm in Python

Theory :-

a.) BFS

→ Breadth First search is most common search stragery for traversing a tree or graph

→ This algorithm searches breadthwise in tree or graph

→ BFS starts searching from root node of tree and expands all successor node at current level before moving to nodes of next level.

→ It is implemented using FIFO queue data structure

→ The Algorithm for BFS can be written as :-

Algo :-

1.) START

2.) PUT root node in open queue

3.) while (queue != empty)

a.) Remove node from queue front

i) If (node == goal node)

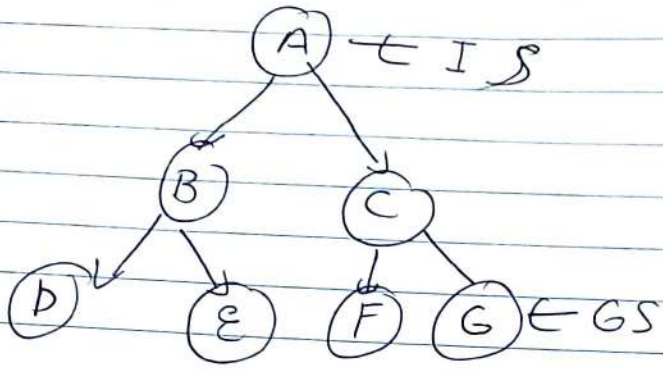
Return success;

ii) Else

Put all children of node on queue rear

4.) Return failure

Example :-



Soln :-

	X	open	close
		{A}	-
1.)	{A}	{B, C}	{A}
2.)	{B}	{C, D, E}	{A, B}
3.)	{C}	{D, E, F, G}	{A, B, C}
4.)	{D}	{E, F, G}	{A, B, C, D}
5.)	{E}	{F, G}	{A, B, C, D, E}
6.)	{F}	{G}	{A, B, C, D, E, F}
7.)	{G}	{-}	{A, B, C, D, E, F, G}

Goal State Reached

∴ Path :— A → B → C → D → E → F → G

Problem Statement: Consider a weighted connected graph with start node S & goal node. G.

i:) Apply DFS algorithm & show how the state space tree will be created while searching a path from initial node to goal node G

ii:) Show the contents of OPEN LIST & CLOSED LIST after each iteration.

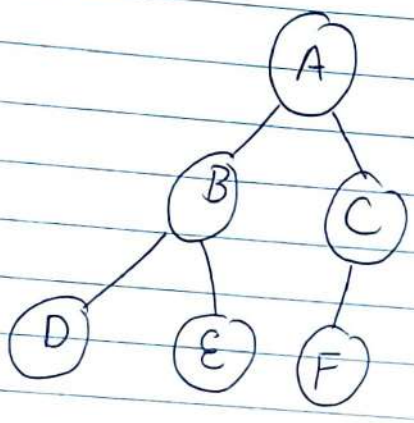
Theory :-

Uninformed search is a class of general-purpose search algorithms which include operating in a brute-force way. They do not have additional information about state or search space other than how to traverse the tree, so it called uninformed search or blind search

DFS i e Depth First search is a graph traversal algorithm that explores as far as possible along each branch before backtracking. Its limitations include potentially getting stuck in infinite-depth branches & its non optimality when solution is deeper than DFID i:e Depth-first iterative deepening was developed to address the limitations by combining the completnes of DFS with the optimality of Breadth First search (BFS).

It iteratively performs DFS with increasing depth limit until the goal is found, ensuring both completeness & optimality while controlling memory usage effectively

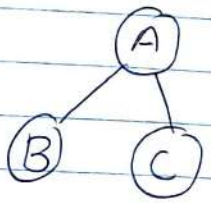
Lets say we have to search for the node 'E' in the following tree.



iteration 1:

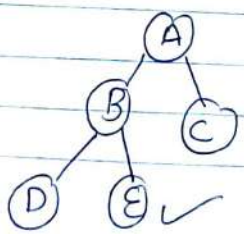
Not found.

iteration 2



Not found

Iteration 3



found

Algorithm:-

i) set initial depth limit to 1

ii) start DFS from root node with current depth limit.

iii) Traverse the tree until depth limit is searched

iv) If goal node is found at current depth, return it, else depth = depth + 1 & go to step 2 again.

Complete — No
Time — $O(b^m)$
Space — $O(bm)$
Optimal — No

Conclusion - learnt about BFS approach in AI.

EXP 3:

```
import heapq


class Graph:
    def __init__(self):
        self.vertices = {}

    def add_edge(self, u, v, weight):
        if u not in self.vertices:
            self.vertices[u] = []
        if v not in self.vertices:
            self.vertices[v] = []
        self.vertices[u].append((v, weight))
        self.vertices[v].append((u, weight))

    def bfs(self, start, goal):
        visited = set()
        queue = [[start]]
        if start == goal:
            return "Start is the goal!"
        while queue:
            path = queue.pop(0)
            node = path[-1]
            if node not in visited:
                neighbors = self.vertices[node]
                for neighbor, _ in neighbors:
                    new_path = list(path)
                    new_path.append(neighbor)
                    queue.append(new_path)
                    if neighbor == goal:
                        return new_path
```

```
                visited.add(node)
        return "No path found"

    def ucs(self, start, goal):
        visited = set()
        queue = [(0, start, [])]
        while queue:
            cost, node, path = heapq.heappop(queue)
            if node not in visited:
                path = path + [node]
                if node == goal:
                    return path
                visited.add(node)
                for neighbor, weight in self.vertices[node]:
                    if neighbor not in visited:
                        heapq.heappush(queue, (cost + weight, neighbor, path))
        return "No path found"


def menu():
    print("Choose Algorithm:")
    print("1. Breadth First Search (BFS)")
    print("2. Uniform Cost Search (UCS)")
    print("3. Exit")


if __name__ == "__main__":
    g = Graph()
    g.add_edge('A', 'B', 4)
    g.add_edge('A', 'C', 2)
    g.add_edge('B', 'C', 5)
```

```
g.add_edge('B', 'D', 10)
g.add_edge('C', 'D', 3)

while True:
    menu()
    choice = int(input("Enter choice: "))
    if choice == 1:
        start = input("Enter start node: ")
        goal = input("Enter goal node: ")
        print("Path using BFS:", g.bfs(start, goal))
    elif choice == 2:
        start = input("Enter start node: ")
        goal = input("Enter goal node: ")
        print("Path using UCS:", g.ucs(start, goal))
    elif choice == 3:
        print("Exiting...")
        break
    else:
        print("Invalid choice. Please try again.")
```

OUTPUT:

```
1
START A
GOAL D
A B D
2
START A
GOAL D
A C D
```

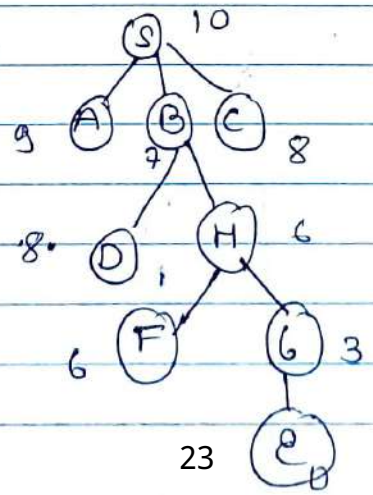
Experiment No: 4

AIM :- Implement Greedy Best First Search Algorithm

Theory :- Greedy Best First search is an AI search algorithm that attempt to find the most promising path from a given starting point to the goal. It prioritize the path that appear to be most promising, regardless of whether or not, they are actually shortest path. The algorithm work by evaluating cost of each possible path & then expanding the path with lowest cost. This process is repeated until goal node is reached

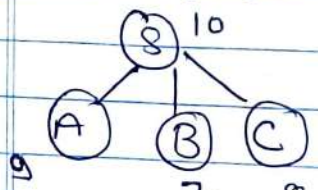
The algorithm work by heuristic function to determine which path is most promising. The heuristic function takes into count the cost of current path and estimated cost of remaining path. If cost of current path is lower than the estimated remaining path, then current path is chosen. This process is repeated until goal is reached.

Example Given :-



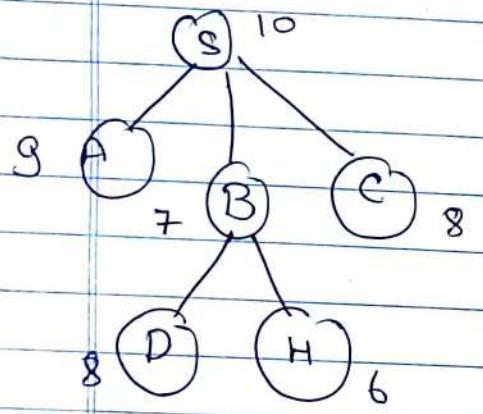
Open		Closed	
Node	H(n)	Node	Parent
S	10		

① 1st iteration



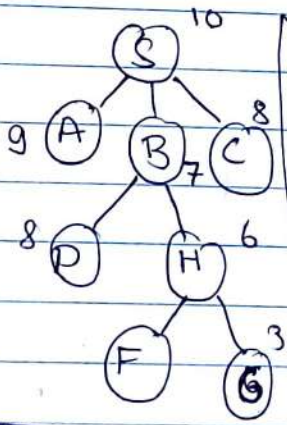
Open		Closed	
Node	H(n)	Node	Parent
B	7	S	S
C	8	B	S
A	9		

② 2nd iteration



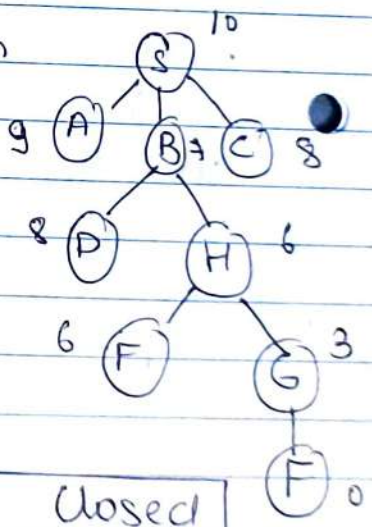
Open		Closed	
Node	H(n)	Node	Parent
H	6	S	
C	8	B	S
D	8	H	B
A	9		

③ 3rd iteration



Open		Closed	
Node	H(n)	Node	Parent
G	3	S	
F	6	B	S
C	8	H	B
D	8	G	H
A	9		

④ 4th iteration



Open		Closed	
Node	H(n)	Node	Parent
E	0	S	
F	6	B	S
C	8	H	B
D	8	G	H
A	9	F	G

Path :- S → B → H → G → E

Algorithm :- Greedy-Best-First-Search (start, goal):

```
Front = priority Queue()
Front.push(start, heuristic(start, goal))
explored = set()
while not Frontier.isEmpty():
    current = Frontier.pop()
    if current = goal:
        return "Goal Found"
    explored.add(current)
    For neighbor in current.neighbors():
        if neighbor not in explored and
        neighbor not in Front:
            Front.push(neighbor, heuristic(neighbour, goal))
    else if neighbor in Front:
        if heuristic(neighbor, goal) < Front.get Priority:
            Front.updatePriority(neighbor, heuristic(neighbor, goal))
return "Goal Not Found"
```

Evaluation :-

Time Complexity :- $O(b^d)$ Space Complexity :- $O(b^d)$

Optimality :- Not optimal Completness :- No

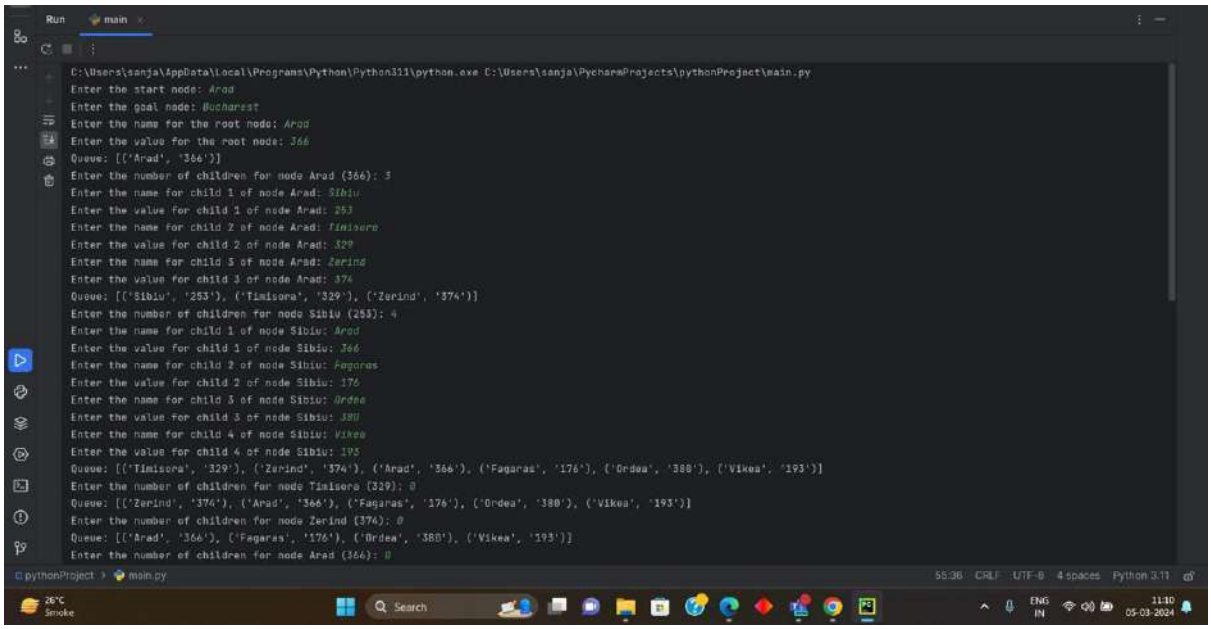
Conclusion :- Learnt about Greedy Best First Search Algorithm in AI.

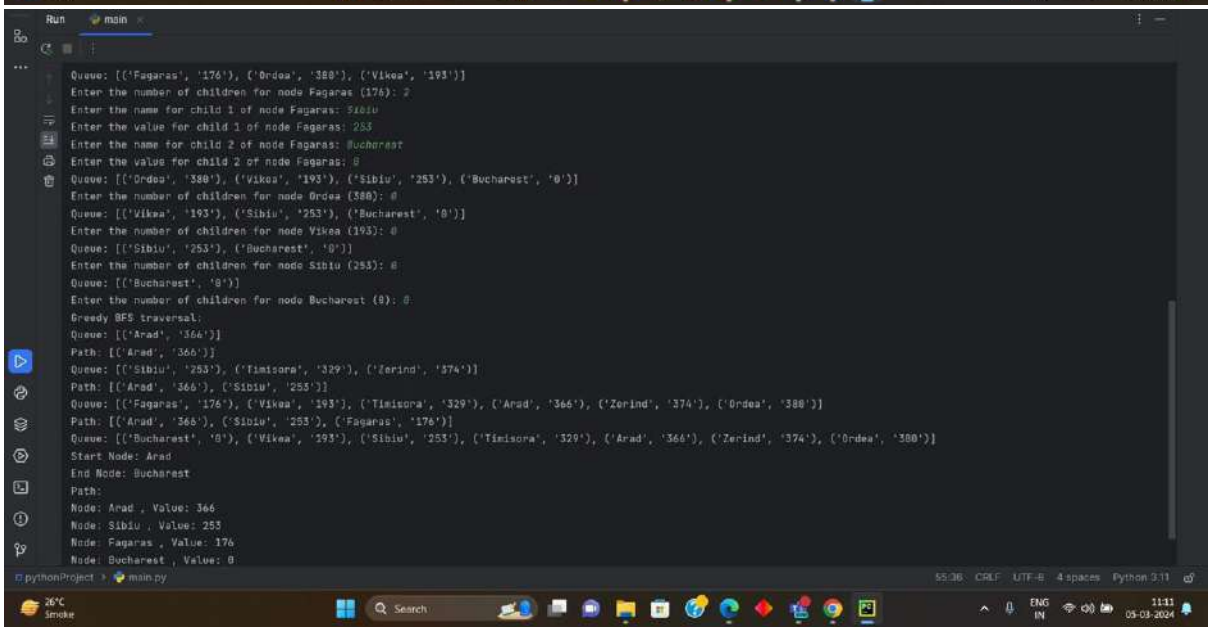
```
class Node:
    def __init__(self, name, value):
        self.name = name
        self.value = value
        self.children = []


def build_tree():
    root_name = input("Enter the name for the root node:
")
    root_value = input("Enter the value for the root node:
")
    root = Node(root_name, root_value)
    queue = [root]

    while queue:
        print("Queue:", [(node.name, node.value) for node
in queue])
        current_node = queue.pop(0)
        num_children = int(input(f"Enter the number of
children for node {current_node.name}
({current_node.value}): "))
        for i in range(num_children):
```

```
            child_name = input(f"Enter the name for child
{i + 1} of node {current_node.name}: ")
            child_value = input(f"Enter the value for
child {i + 1} of node {current_node.name}: ")
            child_node = Node(child_name, child_value)
            current_node.children.append(child_node)
            queue.append(child_node)

    return root



def greedy_bfs(root, goal_node):
    queue = [root]
    path = []

    while queue:
        print("Queue:", [(node.name, node.value) for node
in queue])
        current_node = queue.pop(0)
        path.append((current_node.name,
current_node.value))
        if current_node.name == goal_node:
            break
```

```
        if current_node.children:
            queue = sorted(queue + current_node.children,
key=lambda x: x.value)
        print("Path:", path)

    return path



def main():
    start_node = input("Enter the start node: ")
    end_node = input("Enter the goal node: ")
    root = build_tree()
    print("Greedy BFS traversal:")
    path = greedy_bfs(root, end_node)
    print("Start Node:", start_node)
    print("End Node:", end_node)
    print("Path:")
    for node in path:
        print("Node:", node[0], ", Value:", node[1])
        if node[0] == end_node:
            break
```

```
if __name__ == "__main__":
    main()
```



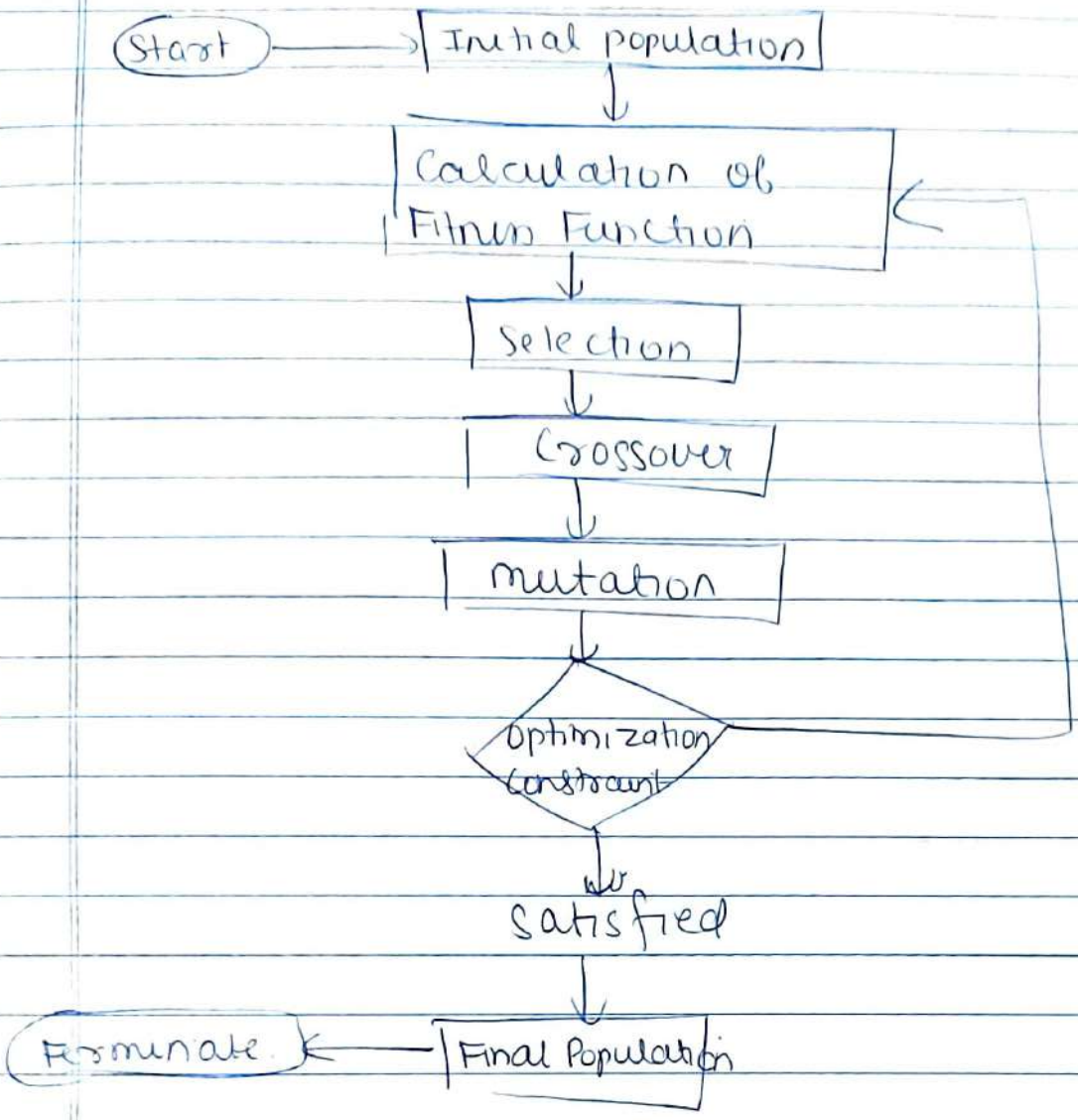


Experiment No: 5

Aim:- Implement Genetics Algorithm

Theory:-
Genetics Algorithm are adaptive heuristic search algorithm that belong to larger part of evolutionary algorithm. Based on idea of natural selection and genetics. Simulate the procss of natural selection i.e these species that can adapt to changes in environment can survive and reproduce and go to next generation. Each generation consist of a population of individual and each individual represent a point in search space & possible solution. Each individual is represented as a string of char limit / Float / bits. This string is analagous to chromosome.

Algorithm:-
1) Determine the number of chromosome, generation, mutation rate and crossover rate value.
2) Generative chromosone number of chromosome, population and initalization value of genes, chromosomes. Chromose with ~~initalization~~ random value
3) Procss steps 4-7 repeated until no of generation
4) Evaluation of fitness value of chromosomes by calculating objective function
5) Chromosome selection
6) Crossover
7) Mutation
8) Best Solution



Operation - 1) Selection :- The idea is to give preference to individual with good fitness scores & allow them to pass their genes to successive generation

2) Crossover - This represent mating between individual. Two individuals are selected using selection operator & crossover sites are chosen randomly

3) Mutation - The key idea is to insert random genes in offspring to maintain the diversity in population to avoid premature convergence

Conclusion - learnt about Genetics Algorithm in AI

```
import random

POPULATION_SIZE = 6
GENES = [str(i) for i in range(10)]
TARGET = 30
MAX_ITERATIONS = 50

class Chromosome:
    def __init__(self, genes):
        self.genes = genes
        self.fitness = self.calculate_fitness()

    @classmethod
    def create_chromosome(cls):
        return [random.choice(GENES) for _ in range(4)]

    def calculate_fitness(self):
        return abs(sum((i + 1) * int(gene) for i, gene in
enumerate(self.genes)) - TARGET)

def selection(population):
    sorted_population = sorted(population, key=lambda x:
x.fitness)
    return sorted_population[:2]

def crossover(parent1, parent2):
    # Two-point crossover
    crossover_points = sorted([random.randint(1,
len(parent1.genes) - 1) for _ in range(2)])
    child_genes = (
        parent1.genes[:crossover_points[0]] +

parent2.genes[crossover_points[0]:crossover_points[1]] +
        parent1.genes[crossover_points[1]:]
    )
    return Chromosome(child_genes)

def mutation(child):
    # Randomly mutate one or two genes
```

```
    mutated_gene_indices =
random.sample(range(len(child.genes)), random.randint(1,
2))
    for index in mutated_gene_indices:
        child.genes[index] = random.choice(GENES)
    return child

def main():
    # Given input
    initial_chromosomes = [
        [12, 5, 23, 8],
        [2, 21, 18, 3],
        [10, 4, 13, 14],
        [20, 1, 10, 6],
        [1, 4, 13, 19],
        [20, 5, 17, 1]
    ]

    population = [Chromosome(chromosome) for chromosome in
initial_chromosomes]
    iteration = 0
    best_solution = None

    while iteration < MAX_ITERATIONS:
        parent1, parent2 = selection(population)
        child = crossover(parent1, parent2)
        child = mutation(child)

        population.remove(max(population, key=lambda x:
x.fitness))
        population.append(child)

        best_chromosome = min(population, key=lambda x:
x.fitness)
        if best_solution is None or
best_chromosome.fitness < best_solution.fitness:
            best_solution = best_chromosome

        iteration += 1
```

```
    print("Initial Chromosomes:")
    for i, chromosome in enumerate(initial_chromosomes,
1):
        print(f"Chromosome {i}: {chromosome}")

    print("\nFinal Chromosomes:")
    for i, chromosome in enumerate(population, 1):
        print(f"Chromosome {i}: {chromosome.genes}")

    print(f"\nBest Chromosome: {best_solution.genes}")
    print(f"Best Solution: {sum((i + 1) * int(gene) for i,
gene in enumerate(best_solution.genes))}")
    print(f"Values of ['a', 'b', 'c', 'd']:
{tuple(best_solution.genes)}")
    print(f"Number of Iterations: {iteration}")

if __name__ == "__main__":
    main()
```

```
Initial Chromosomes:
Chromosome 1: ['3', '1', '7', '1']
Chromosome 2: ['2', '1', '7', '1']
Chromosome 3: ['2', '1', '7', '1']
Chromosome 4: ['3', '1', '6', '1']
Chromosome 5: ['3', '9', '6', '1']

Final Chromosomes:
Chromosome 1: ['3', '1', '7', '1']
Chromosome 2: ['2', '1', '7', '1']
Chromosome 3: ['2', '1', '7', '1']
Chromosome 4: ['3', '1', '6', '1']
Chromosome 5: ['3', '9', '6', '1']

Best Chromosome: ['3', '1', '7', '1']
Best Solution: 30
Values of ['a', 'b', 'c', 'd']: (3, 1, 7, 1)
Number of Iterations: 3

Process finished with exit code 0
```

Experiment No: 6

Title :- Knowledge representation & creating Knowledge base for Wumpus World.

Introduction :- The Wumpus World agent is an example of knowledge based agent that represent knowledge representation, reasoning and planning. Knowledge Based agent link general knowledge with current percepts to infer hidden characters of current state before selecting action. Its necessary is vital in partially observable environment.

Q1.) Represent the following sentence in First Order Logic using consicent vocabulary.

a.) Some student took French in Spring 2001.
b.) Every student who takes French passes it.
c.) Only one student took Greek in Spring 2001.
d.) The best score in Greek is always higher than best score in French.
e.) Every person who buys policy is smart.
f.) No person buys an expensive policy.
g.) There is an agent who sells policies only to people who are not insured.
h.) There is a barber who shaves all men who do not shave themselves.
i.) A person ~~who~~ born in UK each of whose parent in a UK citzen or UK resident, is a UK citizen by birth.

y) A person born outside UK, one of whose parent is a UK citizen by birth, is a UK citizen by descent.

Solution ⇒

To represent these sentence in First Order Logic (FOL), we define define a consistent vocabulary.

~~Student(x) : x is a student~~
~~Friend(y), Greek(y)~~ :
~~Predicates :⇒~~

~~Take(s, c, t) : Predicate : Student s take / took course c in term t.~~

Let :→

(1) took (student, course, semester) - represent that student took coarse in given semester
(2) passes (student, course) - represent that student passes a coarse.
(3) BestScore (coarse, score) - represent that best score in coarse is specific coarse
(4) ~~higher~~ Score (s, c, t) - The score of student s in course c in term t.
(5) Expensive (y) - y is expensive
Agent (x) - x is an agent
Sells Policy (x, y) : x sells a policy to y
Insured (x) : x is insured
Barber (x) : x is a barber
~~Man(y) : y is a man in town~~

Shirish Shetty C32 2103164

Shares(x, y) : x shares y
Born(x, y) : x was born in country y
Parent(x, y) : x is y's parent.
Resident(x, y) : x is resident of country y.
CitizenBy Birth(x, y) : x is citizen of country y by birth.
CitizenBy Descent(x, y) :- x is citizen of country y by descent
French :- Constant, Course teaching language French
Spring2001 - Constant, The spring term in year 2001

Person(x) - x is a person
Policy(y) - y is Policy.
Buys(x, y) : x buys y.
Smart(x) : x is smart

a.) $\exists$ s Takes(s, French, Spring2001).
b.) $\forall$ s,t Takes(s, French, t) $\rightarrow$ pass(s, French)
c.) $\exists$ s Takes(s, Greek, Spring2001) $\wedge$ ($\forall$ z Takes(s, Greek, Spring 2001) $\rightarrow$ s = z)
d.) $\forall$ t, s1, s2 [($\forall$ s8 > (Score(s1, Greek, t), Score(s8, Greek, t))) $\wedge$ ($\forall$ s9 > (score(s2, Greek, t), score(s9, Greek, t))] $\rightarrow$ > (Score(s1, Greek, t), Score(s2, French, t))
e.) ($\forall$ x)($\forall$ y) Person(x) $\wedge$ Policy(y) $\wedge$ Buys(x, y) $\Rightarrow$ Smart(x)
f.) ($\forall$ x)($\forall$ y) Person(x) $\wedge$ Policy(y) $\wedge$ Buys(x, y) $\Rightarrow$ ~Expensive(y)
g.) ($\exists$ x)(Agent(x) $\wedge$ ($\forall$ y) SellsPolicy(x, y) $\Rightarrow$ ~ Insured(y))

h.) $(\exists x)(Barber(x) \wedge (\forall y) man(y) \wedge shaves(x,y) \Rightarrow \sim shaves(y,y))$

i.) $(\forall x)((Born(x,UK) \wedge (\forall y)(Parent(y,x) \wedge (Citizen(y,UK) \vee Resident(y,UK))) \Rightarrow CitizenByBirth(x,UK))$

Now

Q2.) Introduction of Wumpus World is explained Earlier.

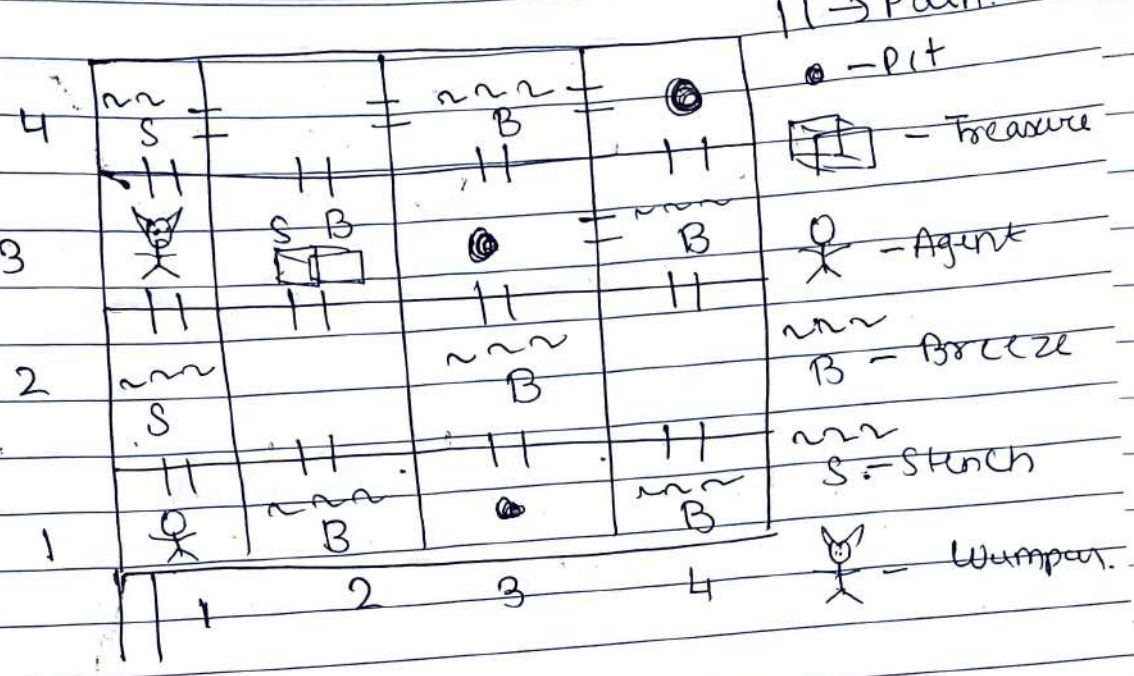
PEAS for Wumpus World problem are:-

① Performance measure
- Agent gets gold and return back safe = +1000 points
- Agent dies = -1000 points
- Each move of agent = -1 point
- Agent uses the arrow = -10 points

② Environment:-
- a cave with 16 rooms
- Room adjacent to wumpus are stinking
- Room adjacent to pit are breezy
- Room with gold glitter.
- Agent initial position - Room[1,1] facing at RHS.
- Location of wumpus, gold and a pit can be anywhere except Room[1,1].

③ Actuators
- move forward
- Turn right
- Turn left
- Shoot
- Release
- Grab

④ Sensors
- Breeze
- Stench
- Glitter
- Scream
- Bump



The agent start visiting from first square [1,1], we already, it is safe for agent. To build knowledge base for it, we will use some Rules and atomic proposition. We need symbol i,j for each location in wumpus world where i is row, j is column

b4	2,4 P?	3,4	4,4
b3 W?	2,3 S G B	3,3	4,3
b2	2,2 V P?	3,2	4,2
b1 A	2,1 B V	3,1 P?	4,1

OK OK

P_{ij} be true if thru pit in room [i,j]
B_{ij} be true if agent perceives breeze in [i,j]
W_{ij} be true if thru is wumpus in square [i,j]
S_{ij} be true if agent perceive stench in square [i,j]
V_{ij} be true if square [i,j] is visited
G_{ij} be true if gold in the square [i,j]
OK_{ij} be true if room is safe

Shirish Shetty C32 2103164

Some Preposition Rules

R1 $\neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

R2 $\neg S_{21} \rightarrow \neg W_{11} \wedge \neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}$

R3 $\neg S_{12} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{22} \wedge \neg W_{13}$

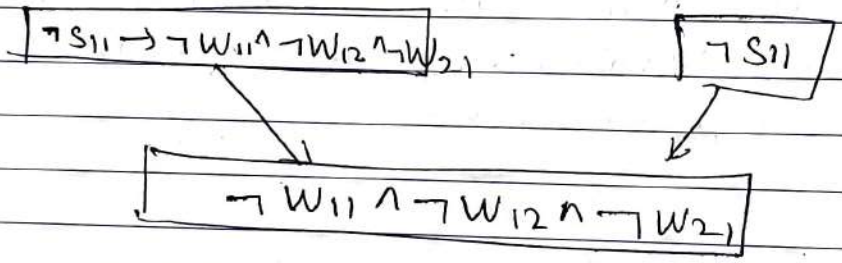
R4 $S_{12} \rightarrow W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$

Knowledge Base Representation.

$\neg W_{11}$	$\neg S_{11}$	$\neg P_{11}$	$\neg B_{11}$	$\neg G_{11}$	V_{11}	OK_{11}
$\neg W_{12}$	---	$\neg P_{12}$	---	---	$\neg V_{21}$	OK_{12}
$\neg W_{21}$	$\neg S_{21}$	$\neg P_{21}$	B_{21}	$\neg G_{21}$	V_{21}	OK_{21}

Now Prove that wumpus in the room (1,3)

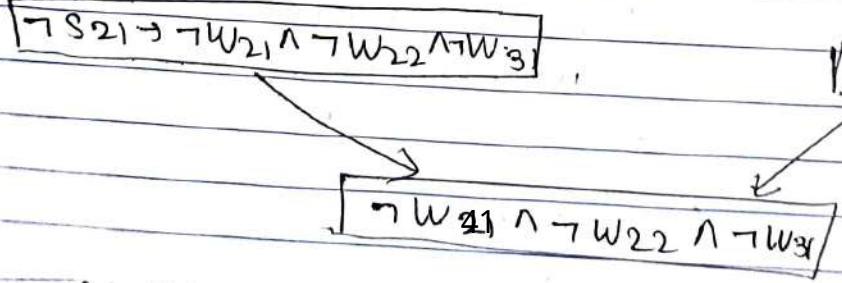
- Apply Modus Ponens with $\neg S_{11}$ and R1



- Apply And Elimation Rule

After apply this we get 3 statement $\neg W_{11}$, $\neg W_{12}$ and $\neg W_{21}$

- Apply Modus Ponens to $\neg S_{21}$ and R2.

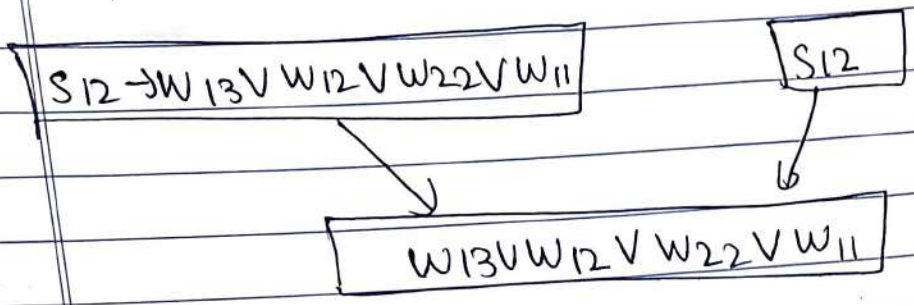


- Shoot - Grab

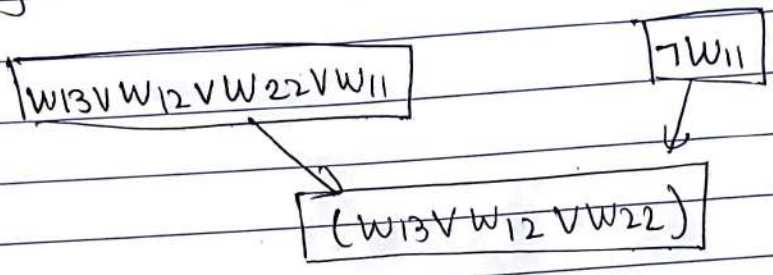
• Apply And - Elimination rule:-

After that we get $\neg W_{21}, \neg W_{22}$ & $\neg W_{31}$

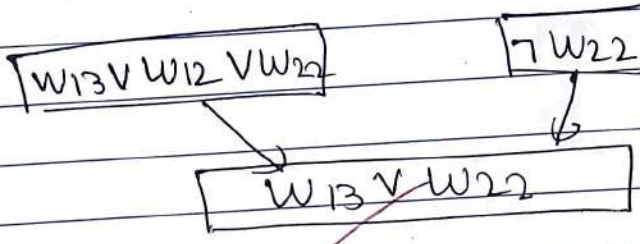
• Apply MP to S_{12} and R_4:-



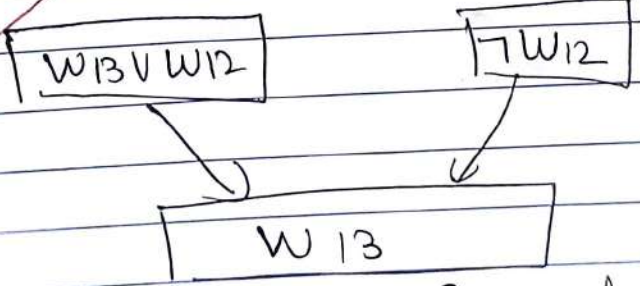
• Apply Unit resolution $W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$ & $\neg W_{11}$:-



• Apply Unit resolution on $W_{13} \vee W_{12} \vee W_{22}$ and $\neg W_{22}$.



• Apply Unit Resolution on $W_{13} \vee W_{12}$ and $\neg W_{12}$



Hence Proved

Conclusion → Learnt about Wumpus World in AI and Knowledge base Approach.

Experiment No: 6

Title :- Knowledge representation & creating Knowledge base for Wumpus World.

Introduction :- The Wumpus World agent is an example of knowledge based agent that represent knowledge representation, reasoning and planning. Knowledge Based agent link general knowledge with current percepts to infer hidden characters of current state before selecting action. Its necessary is vital in partially observable environment.

Q1.) Represent the following sentence in First Order Logic using consicent vocabulary.

a.) Some student took French in Spring 2001.
b.) Every student who takes French passes it.
c.) Only one student took Greek in Spring 2001.
d.) The best score in Greek is always higher than best score in French.
e.) Every person who buys policy is smart.
f.) No person buys an expensive policy.
g.) There is an agent who sells policies only to people who are not insured.
h.) There is a barber who shaves all men who do not shave themselves.
i.) A person ~~who~~ born in UK each of whose parent in a UK citzen or UK resident, is a UK citizen by birth.

Experiment No 7

Title :- Planning for Blocks World Problem

Q:) Explain the Block World Planning Problem and describe how it can be solved using Classical Planning method.

Soln →

Planning problem in AI is about the decision making performed by intelligent creature like robot, human or program when trying to achieve the goal.

- It involves choosing a sequence of actions that will transform the state of world step by step so that it satisfy the goals.

∴ Planning → Set of Actions.

∴ Actions :→ Precondition :→ For actions to be performed there is some need of some precondition.

∴ Effect :- Effect is result of action performed

Planning consist of :-

I) Choose best rule

II) Applying rule on problem to get new state

III) Detect if solution is found

IV) Detect dead end so that new direction will get explored.

Shirish Shetty C32 2103164

What is Block world problem?

- classical problem in AI and planning which involves moving block from an initial state to goal state.

It consist of :-

1. Block :- square block of same size can be stack on one another
2. Flat surface / Table
3. Robotic arm
4. Stack.

Initial state :-

Consist of set of block placed on table or each other forming a stack structure. Each block have labels & their position is determined from where they are placed.

Goal state :-

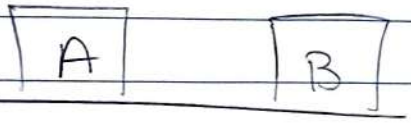
It specifies desired configuration of blocks It would be particular arrangement of block on table or on top of each other. The goal state define what final configuration of block look like

Step cost :-

Each action has step cost associated with it. It represent the cost occurred to perform action & to find optimal solution

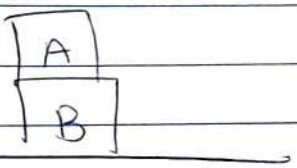
learnt about Block World Problem in AI

Eg :-



Inital state

Inital description
on (A, table)
on (B, table)



Goal State

goal description
On(A, B)

* Rules for Provide solution for it

Rule	Precondition	Action
ck up(X) up block X	on (X, table) clear (X) hand empty	holding(X)
own(Y) block Y on table	Holding (X)	on (X, table) clear (X) hand empty
(X, Y) ock X on block Y	holding (X) Clear (Y)	On(X, Y) Clear (X)
(a (X, Y) nove block X m block Y.	On(X, Y) clear (X)	holding (X) Clear (Y)

Solution for example

Rules :-

I-) Compare metal state with goal state if they match then stop Else step (2)

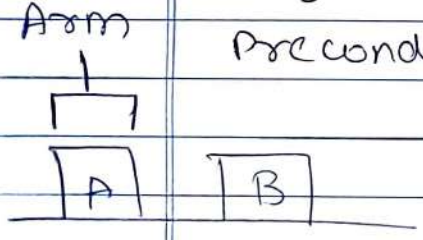
II-) POP goal state from stack & apply suitable rule to perform action

III-) If all precondition of rule satisfied then perform action

IV-) If any precondition of certain rule satisfied then just POP that from stack, don't perform any action

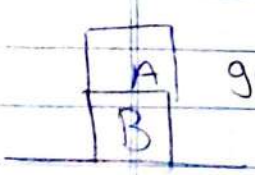
Now By using the above, solution is :-⇒

Action : stack(A, B)
Precondition : holding(A)
clear(B)

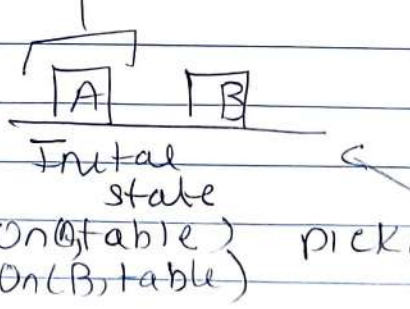


Initial state
on(A, table)
on(B, table)

Action : pickup(A)
Precondition :- on(A, table)
clear(A)
hand empty



goal description
on(A, B)



Goal state

Conclusion — learnt about Block World Problem in AI.

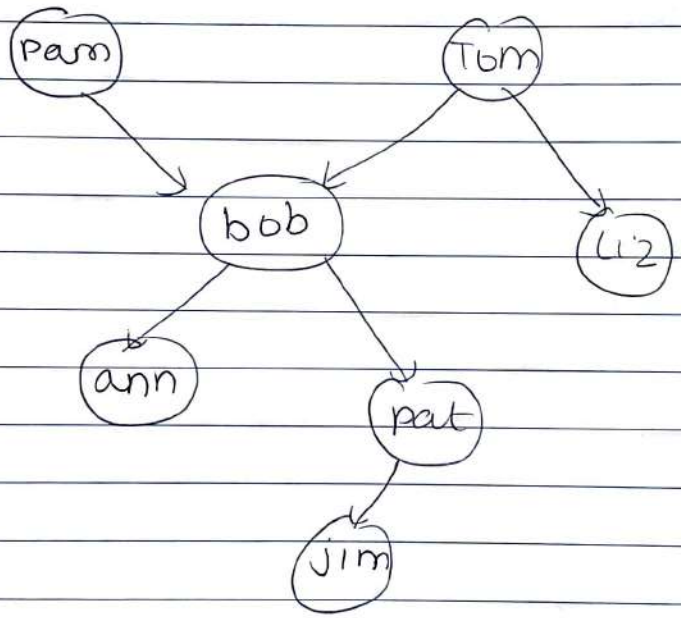
Experiment 8.

Title :- Implementation Family Tree using prolog.

Theory :-

Prolog is a programming language for symbolic, non numeric computation. It is specially well suited for solving problems that involve object and relation between object

In prolog we can write the family Tree as

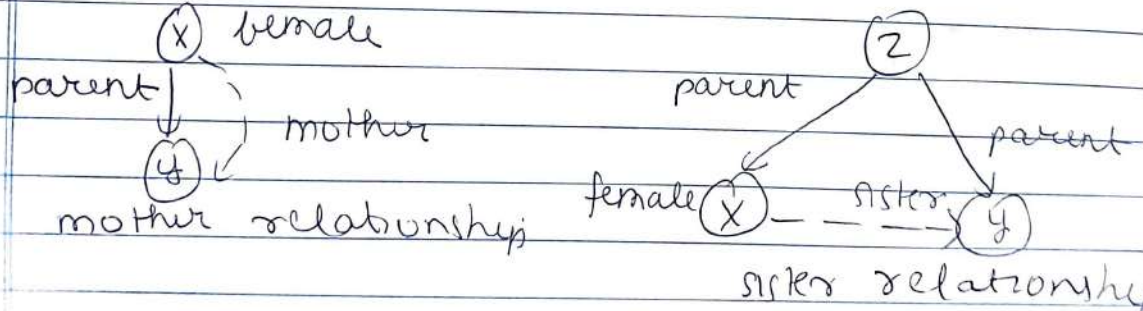



```
parent (pam, bob).
parent (tom, bob).
parent (tom, liz).
parent (bob, ann).
parent (pat, jim).
parent (bob, peter).
parent (peter, jim).
```

These help use to illustrate important point

- We have defined parent relation by stating n tuples of object based on given info in family tree
- The user can easily query the Prolog system about relation defined in program.
- A prolog program consist of clauses terminated by full stop
- The argument of relation can be concrete object constant or general object such as X and Y. Object of first kind in our program are called atoms. Object of second kind are called variable
- Question to system consist of one or more goals

There are additional rule :-
Lets see how can we define mother and sister relationship



mother(X,Y) :- parent(X,Y), female(X).
sister(X,Y) :- parent(Z,X), parent(Z,Y), female(X), X/==Y.

Similarily we can define :-

father(X, Y) :- parent(X, Y), male(X).

hasChild(X) :- parent(X, _).

brother(X, Y) :- parent(Z, X), parent(Z, Y), male(X), X \== Y.

For eg :->

* Facts *

male(john)

female(lisa)

~~parent(john, lisa~~ female(sarah)

parent(john, lisa)

parent(john, sarah).

* Query *

? - father(john, lisa)

true

? - mother(jane, sarah)

false. % Jane is not defined in fact, Prolong cannot find her as mother of Sarah.

? - parent(john, sarah).

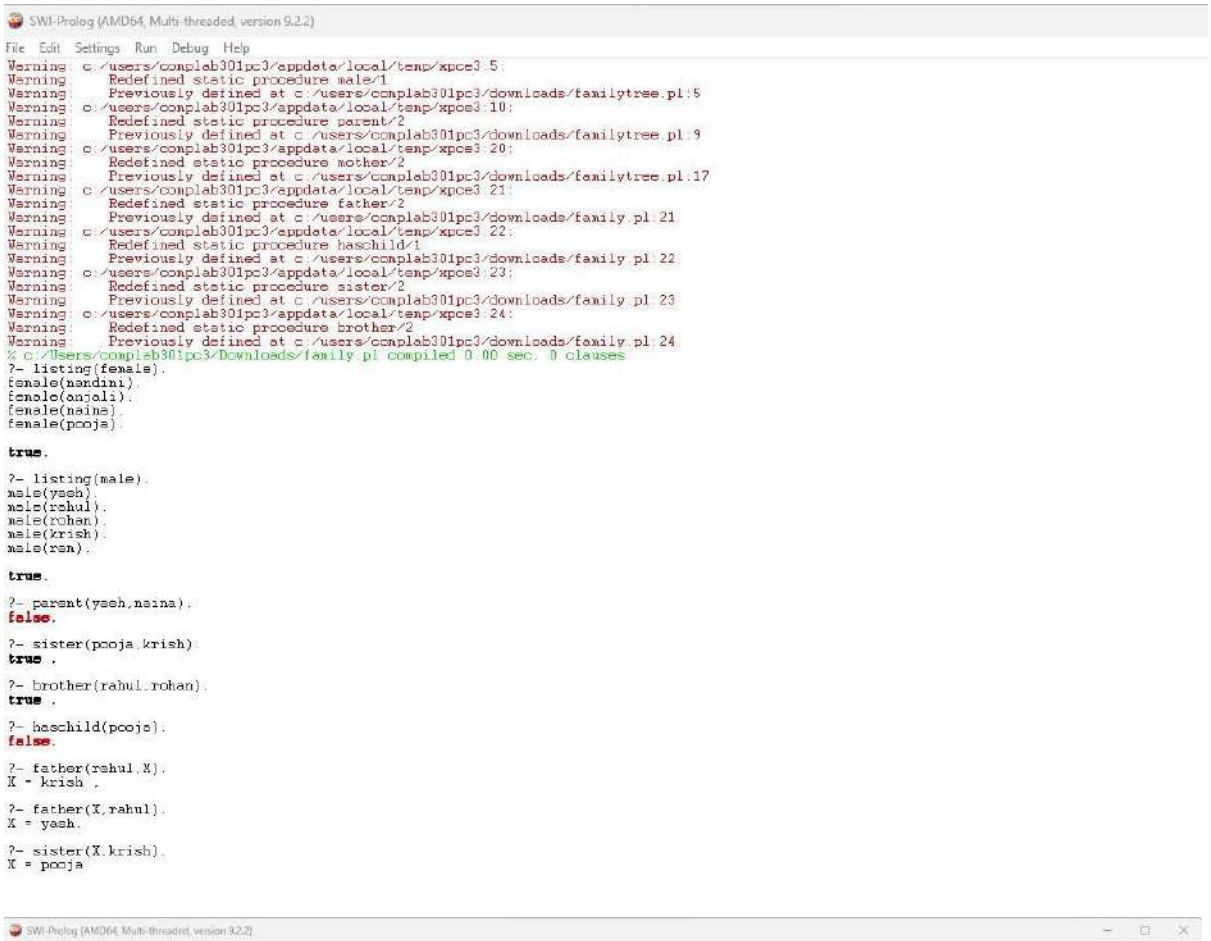
true. % John is parent of Sarah.

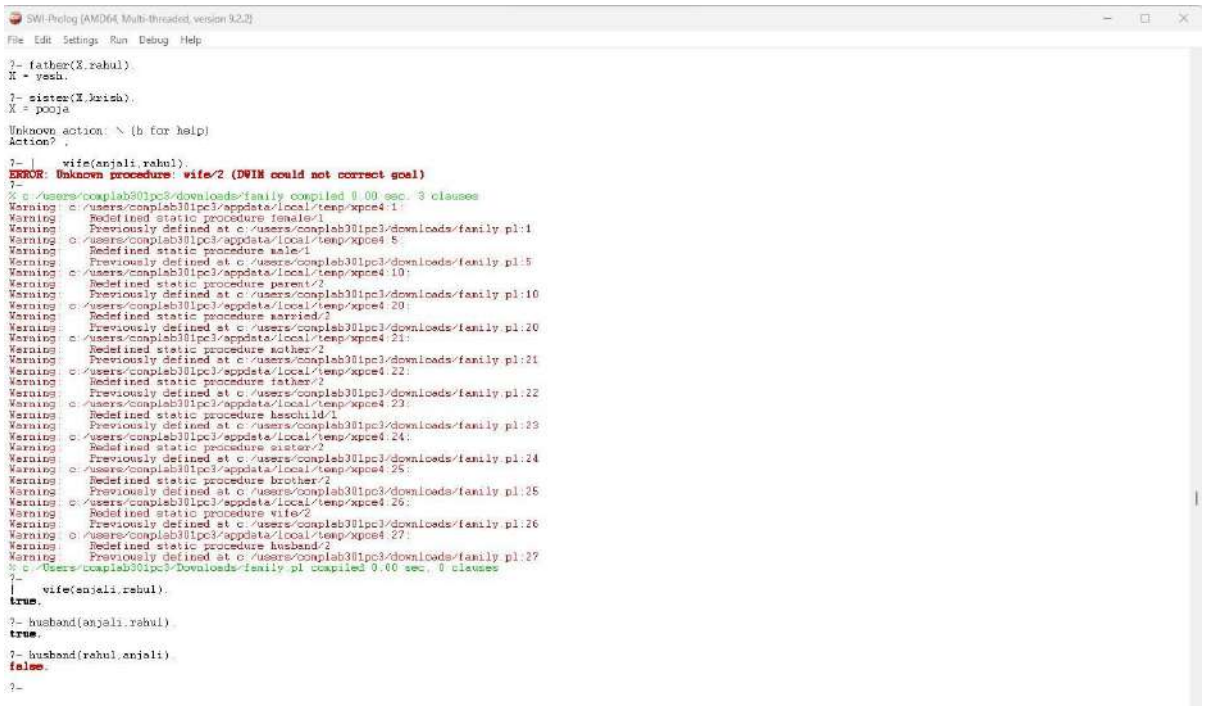
Conclusion :- Learnt about prolong and how to understand family Tree using it in AI.

CODE:

```
female(nandini) .
female(anjali) .
female(naina) .
female(pooja) .
male(yash) .
male(rahul) .
male(rohan) .
male(krish) .
male(ram) .
parent(nandini,rahul) .
parent(nandini,rohan) .
parent(yash,rahul) .
parent(yash,rohan) .
parent(anjali,krish) .
parent(anjali,pooja) .
parent(rahul,krish) .
parent(rahul,pooja) .
parent(rohan,ram) .
parent(naina,ram) .
married(anjali,rahul).
mother(X,Y) :- parent(X,Y),female(X).
father(X,Y) :- parent(X,Y),male(X).
haschild(X) :- parent(X,_) .
sister(X,Y) :- parent(Z,X),parent(Z,X),female(X),X\==Y .
brother(X,Y) :- parent(Z,X),parent(Z,Y),male(X),X\==y.
wife(X,Y):- married(X,Y),female(X).
husband(X,Y):- married(X,Y),male(Y).
```

OUTPUT:


SWI-Prolog (AMD64, Multi-threaded, version 9.2.2)
File Edit Settings Run Debug Help
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce3:5:
Warning: Redefined static procedure male/1
Warning: Previously defined at c:/users/conplab301pc3/downloads/familytree.pl:5
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce3:10:
Warning: Redefined static procedure parent/2
Warning: Previously defined at c:/users/conplab301pc3/downloads/familytree.pl:9
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce3:20:
Warning: Redefined static procedure mother/2
Warning: Previously defined at c:/users/conplab301pc3/downloads/familytree.pl:17
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce3:21:
Warning: Redefined static procedure father/2
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:21
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce3:22:
Warning: Redefined static procedure haschild/1
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:22
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce3:23:
Warning: Redefined static procedure sister/2
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:23
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce3:24:
Warning: Redefined static procedure brother/2
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:24
% c:/Users/conplab301pc3/Downloads/family.pl compiled 0.00 sec, 0 clauses
?- listing(female).
female(nandini).
female(anjali).
female(naina).
female(pooja).
true.
?- listing(male).
male(yash).
male(rahul).
male(rohan).
male(krish).
male(ram).
true.
?- parent(yash,naina).
false.
?- sister(pooja,krish).
true .
?- brother(rahul,rohan).
true .
?- haschild(pooja).
false.
?- father(rahul,X).
X = krish ;
?- father(X,rahul).
X = yash.
?- sister(X,krish).
X = pooja


SWI-Prolog (AMD64, Multi-threaded, version 9.2.2)
File Edit Settings Run Debug Help
?- father(X,rahul).
X = yash.
?- sister(X,krish).
X = pooja
Unknown action: ^ (h for help)
Action? .
?- wife(anjali,rahul).
ERROR: Unknown procedure: wife/2 (DWIM could not correct goal)
?-
% c:/Users/conplab301pc3/Downloads/family compiled 0.00 sec, 3 clauses
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce4:1:
Warning: Redefined static procedure female/1
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:1
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce4:5:
Warning: Redefined static procedure male/1
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:5
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce4:10:
Warning: Redefined static procedure parent/2
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:10
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce4:20:
Warning: Redefined static procedure married/2
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:20
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce4:21:
Warning: Redefined static procedure mother/2
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:21
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce4:22:
Warning: Redefined static procedure father/2
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:22
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce4:23:
Warning: Redefined static procedure haschild/1
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:23
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce4:24:
Warning: Redefined static procedure sister/2
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:24
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce4:25:
Warning: Redefined static procedure brother/2
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:25
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce4:26:
Warning: Redefined static procedure wife/2
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:26
Warning: c:/users/conplab301pc3/appdata/local/temp/xpce4:27:
Warning: Redefined static procedure husband/2
Warning: Previously defined at c:/users/conplab301pc3/downloads/family.pl:27
% c:/Users/conplab301pc3/Downloads/family.pl compiled 0.00 sec, 0 clauses
?-
wife(anjali,rahul).
true.
?- husband(anjali,rahul).
true.
?- husband(rahul,anjali).
false.
?-

Shirish Shetty C32 2103164

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Assignment No - 1

Q1) Give one definition on AI for each of the following

1.) Acting Humanly.

- Acting humanly in the context of AI refers to the ability of a machine or system to mimic human behaviour and perform tasks in a way that is indistinguishable from how a human would act.
- This approach focuses on replicating human actions, gestures and responses in various situations.
- Achieving acting humanly involves understanding natural language, recognizing emotion and exhibiting socially acceptable behaviour.

2.) Thinking humanly

- Thinking humanly in AI pertain to designing system that emulate the cognitive process and thought pattern of humans.
- This approach aim to model the inner workings of human mind, encompasing perception, problem solving, learning and decision making.
- Achieving thinking humanly remain a challenging goal, as it requires a deep understanding of human psychology cognition & consciousness.

3.) Acting Rationally

- Acting Rationally in AI refer to ability of a system to make decision and take action that optimize it chances of achieving a specific goal.
- This approach emphasizes logical reasoning & rational decision-making based on available information
- It does not necessarily require machine to mimic human behaviour but focuses on achieving optimal outcomes.

4.) Thinking Rationally.

- Thinking Rationally in AI involves designing system that follow principles of logic and reason to drive conclusion and make decision.
- This approach aim to capture the essence of human rationality, emphasizing sound interference and deduction
- Thinking rationally requires the ability to analyze information draw logical inferences, and reach conclusions based on established rules and knowledge.

Q.2) Explain components of AI system in detail.

- AI is a type of technology that uses intelligent system to perform tasks that typically requires human intelligence
- It consist of component listed below :-

1.) Perception :-

- It involves the ability of a system to interpret and understand information from surrounding environment.
- This include acquiring data from various sources such as sensors, images or text and processing it to extract meaningful insights.
- Eg :- Computer vision, Speech recognigation etc.

2.) Knowledge representation

- This is procss of organizing & structing information in a format that a computer system can use to reason, draw inferences and solves problem.
- It inveles creating model to represent facts, concepts & relationship within a domain.

3.) Learning :-

- Learning in AI refer to capability of a system to improve its performance time through experience.
- ML algorithm enable system to recognize pattern

adapt to new info, and make predictions or decisions without being explicitly programmed.

4.) Reasoning

- It involves the ability to draw logical inferences from the available knowledge.
- Reasoning is crucial for decision making processes within AI system, enabling them to make informed choices based on info at hand

5.) Problem Solving

- This in AI involves devising algorithm and techniques to find solution to complex issues.
- This include defining problem, breaking them down into subproblem and developing strategies to solve them.

6. NLP (Natural language Processing) :-

- NLP is a subfield of AI focused on enabling machine to understand, interpret and generate human language.
- It involves the development of algorithms and model to process and analyze text or speech.

ategorization of AI.

I can be categorized based on its capabilities nd functionalies.
The 3 primary catgaries are:-
Weak AI
General AI
Strong AI

Weak AI:-
It is designed to perform specific task or Solve or particular problem. The systems excels within a defined context but lack the ability to generalize their intelligence to other domain
Eg:- Virtual assistants, recommendation, speech recongnizations etc.

General AI:-
It represent a higher level of AI where machine posses the ability to understand, learn & apply knowledge across diverse tasks.
Unlike weak AI, which is tasks specific, general AI would have congnitive capacity to perform any task that a human can.

Strong AI:-

This goes beyond specialized tasks and aims to replicate human level intelligence across a wide

range of activities.
- It implies machine with capability to not only understand and learn but also exhibit consciousness and self-awareness

2.4) Explain problem formulation with help of example.

- Problem formulation is a crucial step in solving problem using AI and other problem-solving techniques. It involves defining the various component of a problem ~~of pro~~ in a structured manner. lets break down of elment of problem formulation using example of classic-solving scenario- the "missionaries and cannibals" problem.

Problem :- Missionaries and Cannibals.

1.) State formulation :-

- The state describe the current situation of problem. In this case, a state represent the position of the missionaries, cannibals and boat
- A state can be represented as (M, C, B) where M is number of missionaries on left bank, C is number of cannibals on left bank, B is position of boat (left or right)

example :- (3,3,L) three ~~are~~ missionaries & three cannibals on left bank with boat on the left

"criminal"

- ...ning would start with known fact like "American (Robert)" and iteratively apply rules until reaching the conclusion that "Robert is criminal"

2) Initial State

- The initial state is starting point of problem. It represent the configuration from which problem-solving process begins.
- Example :- Initial state :- (3, 3, L) :- starting with 3 missionaries and 3 cannibals on left bank with boat on left.

3.) Goal test

- The goal test is a condition that defins when problem is solved. It checks whether current states matches the desired goal state.
- Example goal state :- (0, 0, R) - all missionaries and cannibals are on the right bank with boat on the right

4.) Action sequence

- Actions are possible moves or operation that can be performed to transition from one state to another. In problem, the action are moving missionaries, moving cannibals & moving the boat. Actions can be represented as (m, c, b) where m is number of missionaries to move, c is number of cannibals to move, and b is direction of the boat.

Example :- action : (1, 1, R) - move one missionary & one cannibal to the right bank.

5) Path Cost :-

- The path cost is numerical cost associated with each action. It represent the effort, time, or any other resource required to perform a particular action
- For this problem, the path cost can be defined as the number of actions taken to reach the goal state.

In summary, the problem formulation for Missionaries and cannibals problem involves defining the states, inital state, goal test and path cost. This structured approach helps in systematically solving problem using various problem solving algorithms and techniques

25) Explain PEAS properities in detail.

- PEAS stands Performance measure, Environment, Actuators and sensors. These properities are fundamental components used to describe an intelligent agent in field of AI.

1.) Performance measure
- The performance measure is a criterion or metric used to evaluate how agent is accomplishing its objective or goals.
- It defines the success criteria for agent and guids its decision-making process

- The goal is to design is an agent that maximes or optimizes it performance measure
- Example:- for chess playing agent, the performance measure could be wining the game. The agent aim to make that lead to a wining position while minimizing the number of moves or time taken.

2.) Environment:-
- The environment is external context or surrounding in which agent operates and interacts
- It includes everything outside the agent that may affect or be affected by agent's actions.
- The environment can be simple or complex, deterministic and static and dynamic.
- Eg:- in a vaccum cleaner agent, environment include the rooms, furniture, dirt and any obstacles. The state of environment changes as agent moves & cleans.

) Actulators:-
- Actualators is mechanism or component that carrys out the action or execute the decision made by the agent
They are responsible for transforming the agent decision into physical or digital action that affect environment
Acctulators can range from simple motor to complex robotic limbs or software modules.
Example - for Robotic arm, used in manufacturing, the actuators are motors or hydraulics responsible

for moving the arm to pick, place or
around the object

4) Sensors :-

- Sensors are input devices or component that allow agent to preceive or gather information about the environment
- They provide feedback to agent, enabling it to make informed decision based on current state of environment.
- Sensors can include camera, microphone, touch, sensors, and other sensory devices
- Example :- In autonomous vehicle, sensors such as camera, LiDar and radar are used to preceive the surrounding environment, detect obstacles & nagavigate safely.

Example Of PEAS Properities for medical Diagonsis System :-

- Performance Measure :- Accurany of diagansis, speed of diagonosis, patient outcomes.
- Environment :- Hospital, patient medical record, medical imaging data
- Actuators :- Report generation system, communicaton modules to inform healthcare profissionals.
- Sensors - Medical imaging devices, patient records, laboratory test results

) Describe different types of environment with suitable examples.

Environments in context of AI and reinforcement learning can be characterized along various dimension.

Fully and Partially observable

- Fully observable environment provide an agent with complete information about the current state, while partially observable environments whereas poker is partially observable.

Deterministic VS stochastic environment
- Deterministic environment have outcomes entirely determined by their current state & action taken while stochastic environments involve some level of randomness.
- eg - Deterministic → Physics simulation.
Stochastic → Game involving dice rolls.

Dynamic vs Static
- Dynamic environment changes over time, while static environment remain constant.
Dynamic environment could be traffic system, while static environment might be a fixed puzzle.

4.) Episodic vs Sequential.

- Episodic environment consist of isolated episodes without influencing future states such as tic-tac-toe.
- Sequential environment involve a sequence of action leading to cumulative effect like chess game.

5.) Discrete vs Continous.

- Discrete environment have distinct, seperate states and action like chessboard.
- Continous environment have continous state and action space such as robotic arm's movement.

6.) Single Agent vs Multiagent.

- Single-agent environment involve a solitary decision maker, while multiagent environment have multiple independent decision makers.
- Single agent scenario → Maze-solving robot
- Multi agent scenario → Multiple robots co-operating or compressed.

Brilliant 164 A+

Roll no - 2103164 Shirish Shetty C3 C32

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Assignment -2

Q1) Foward and Backward Chaining

Forward and Backward Chaining are two fundamental strategies in AI used by inference engine to make deduction based on available data and goal respectively.

* Forward Chaining

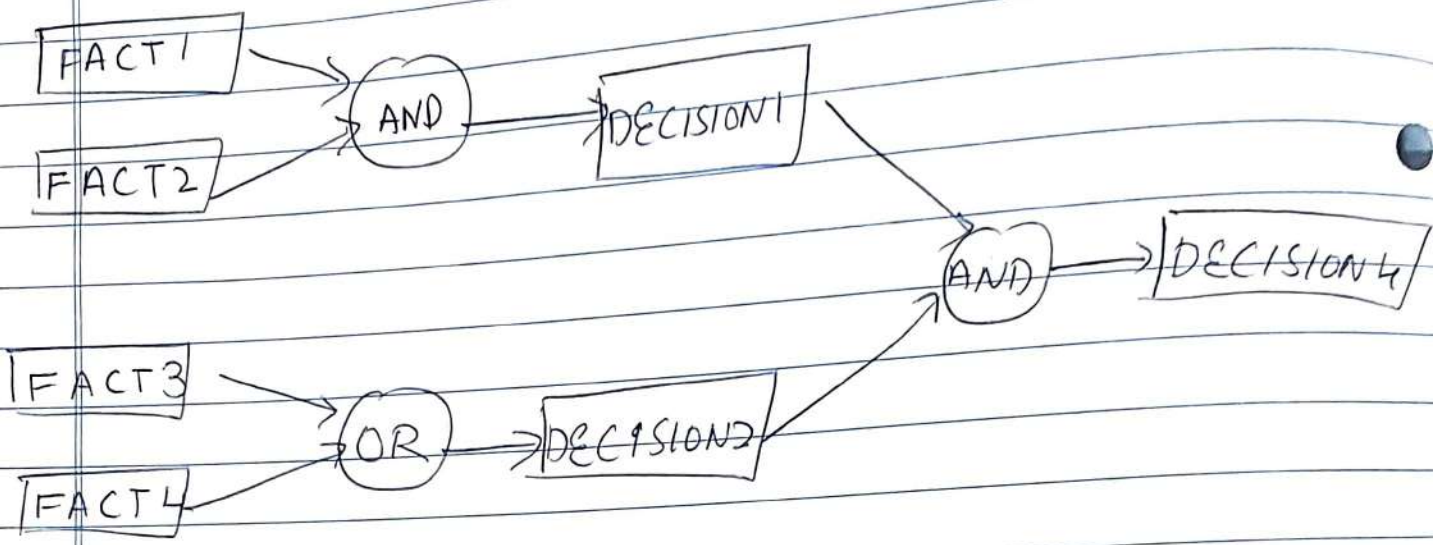
- Defination :- Forward chaining, also known as forward deducation or forward reasing, start from known facts, trigger rules whose premises are satisfied and adds their conclusion to known facts until problem is solved.

- Proces :- It work from intial state toward the goal by applying interence rule to reach a conclusion

- Example :- To prove that "Robert is criminal" forward chaining would start with known fact like "American (Robert)" and iteratively apply rules until reaching the conclusion that "Robert is criminal"

- What can happen next approach

- Properties :- It is data driven uses BFS strategy, aim to get conclusion, operates in forward direction & used in planning, monitoring, control & interpretation applications.



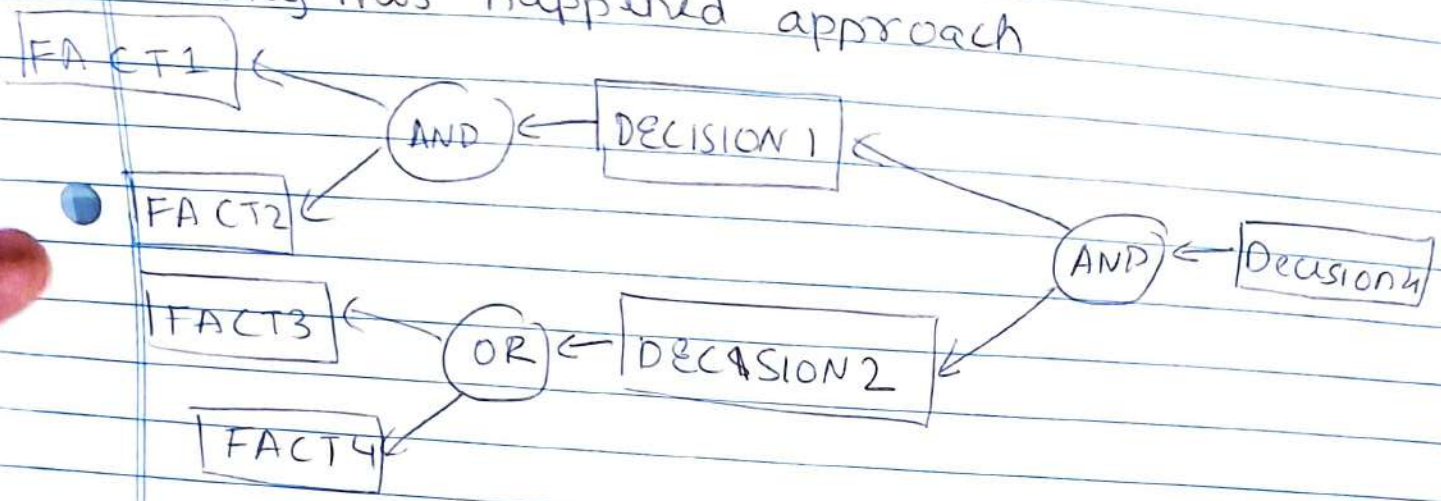
* Backward Chaining

Definition :- Backward chaining, also known as backward deduction or backward reasoning, start from goal & works backward through rules to find known facts that support the goal.

Process :- It begin with goal & work toward the initial state by inferring facts that lead to signal

Example :- Starting with goal "Robert is criminal", backward chaining would infer fact like "If he is running, he sweats" to establish that "He is running".

- goal driven uses DFS strategy and aim to find possible fact or required data, operates in backward direction used in automated inference engines & other AI application
- Why this happened approach



Q2) AI in NLP and robotics

AI in NLP:-

- Enhances language understanding:- AI models comprehend and generate human-like text, enabling complex interaction & understanding of context

Improves information extraction :- AI can identify and extract specific information from large volume of text, making data processing more efficient.

Power Chatbots & virtual assistant - Leveraging AI, these platform can understand & respond to human queries in natural manner, offering personalized assistance

Conducts sentiment analysis :- AI algorithm analyze text to determine the sentiment behind, useful in monitoring social media, customer feedback & market research

AI in robotics :-

Enables autonomous navigation :- Robots can indipendtly navigate & adapt environment, using AI to proces sensory data & avoid obstacles

Facilitates object recognitation & manipulation - AI allow robot to identify, classify & interact object ~~with~~ in their surrounding, crucial for tasks in manufacturing, logestics & service industries

Improves human robot interaction - AI enchance ability of robot to understand & respond to human language & gestures, making interaction more natural & intuitive

Automates complex tasks :- From assembling product to performing surgery, AI - equipped robot can execute precise & repetitive task, improving efficiency and safety.