Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

A.Y. 2021-2022

Name: Meith Navlakha

SAP ID: 60004190068

Branch: Computer Engineering

Subject: Artificial Intelligence

Academic Year: 2021-2022

Semester: V

Division: B

Batch: B1

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

EXPERIMENT 1

AIM: AI problem formulation and understanding its PEAS**.**

THEORY:

In Artificial Intelligence an Intelligent Agent (IA) is an entity that makes a decision, that enables artificial intelligence to be put into action. It can also be described as a software entity that conducts operations in the place of users or programs after sensing the environment. It uses actuators to initiate action in that environment.

PEAS is a type of model on which an AI agent works upon. When we define an AI agent or rational agent, then we can group its properties under PEAS representation model.

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

1. **Performance Measure:** Performance measure is the unit to define the success of an agent. Performance varies with agents based on their different precept.
2. **Environment:** Environment is the surrounding of an agent at every instant. It keeps changing with time if the agent is set in motion. The major types of environments:
    - Fully Observable & Partially Observable
    - Episodic & Sequential
    - Static & Dynamic
    - Discrete & Continuous
    - Deterministic & Stochastic
3. **Actuator:** Actuator is a part of the agent that delivers the output of an action to the environment.
4. **Sensor:** Sensors are the receptive parts of an agent which takes in the input for the agent.

PEAS

1) **AI Assistant (Alexa/Siri)**

- **Define:** AI Assistant viz, Siri and Alexa are intelligent agents using sensors, such as microphones and other inputs, to perceive a request and they draw on their collective experience and knowl
- edge to make a decision.
- **PEAS:**
    - **Performance:** Efficient Search , Speed , Security , Battery life
    - **Environment:** Person, Web browser, Internet Connectivity

- **Actuators:** Speaker , Monitor , Lights
- **Sensors:** Microphone , Speech Detector , Touch Sensors

- **Environment:** Partially observable, dynamic, Stochastic, collaborative, Single Agent, Continuous.

2) Automated Gun

- **Define:** The Automated Gun agent is an intelligent agent that detects the target and shoots on movement while not shooting other innocents or civilians.
- **PEAS:**
  - **Performance:** Locking correct Target, No innocent killing, Security
  - **Environment:** Target, , Operator, Other Civilians/Pedestrians, Location, Weather.
  - **Actuators:** Trigger, Reload, Movable Arms and Stand,
  - **Sensors:** Movement detection, SONAR, Visual Sensors, Wind Sensors, Heat Sensors

- **Environment:** Partially observable, Dynamic, Stochastic, Single-Agent, Continuous, Episodic.

3) Fruit-Segregation Robot

- **Define:** The Fruit Segregation Robot scans the fruit and places it into its respective bin. Thus, helps in segregating the fruits from the pile based on their size and quality.

- **PEAS:**
  - **Performance:** Precision(percentage of Fruits in the correct bins), Time, Less bad quality fruits picked, Efficient segregation based on shape
  - **Environment:** Bins, Conveyor belt, Placement of Fruits , Supervisor
  - **Actuators:** Robot's Arm, Wheels, Robot's Grippers
  - **Sensors:** Camera , Arm Activators, Fruit shape and quality Detector

- **Environment:** Partially observable, Dynamic , Stochastic, Single Agent, episodic, Discrete.

4) Football Playing Bot

- **Define:** A football bot is an intelligent agent that will play along with the other players in a multiplayer football game.

- **PEAS:**
  - **Performance:** Score Goal , Less Foul play, Win , Playing according to the role assigned ,
  - **Environment:** Team players , opponents , Network, Referee, Football Field
  - **Actuators**: Bot's Legs, Head and Hands, Navigators
  - **Sensors:** Distance from ball detector, Orientation detector, Other Players detector, position in the field, speed detector

- **Environment:** Partially observable , dynamic , Stochastic , competitive, Sequential, Multi-Agent, Continuous.

5) Delivery Robots (COVID)

- **Define:** An autonomous self driving robots which are used for delivery of food and other utilities.
- **PEAS:**
  - **Performance:** Safety, time, Abiding traffic rules, Comfort , Cost
  - **Environment**: Road , Customer , Pedestrian ,
  - **Actuators:** Robot wheels , Robot Arms , Speaker
  - **Sensors:** Camera , SONAR , Microphone , Keyboard

- **Environment:** Partially observable , dynamic , Stochastic , collaborative, Single Agent, Continuous.

6) Ticket Checker (COVID)

- **Define:** An automated ticket checker which is used for checking the authenticity of the ticket during the covid by scanning the specified features which it has learnt during the training.
- **PEAS:**
  - **Performance:** Correct Detection(Efficiency) , Time
  - **Environment**: Customer, Entrance-Exit Location, Operator
  - **Actuators:** Monitor, Ticket Insert Rollers, Ticket Shredder, Speaker
  - **Sensors:** Camera, UV/Infrared Scanner, Ticket, QR code scanner

- **Environment:** Fully observable , Dynamic , Deterministic , Single Agent, Discrete.

7) Sanitizer Dispenser (COVID)

- **Define:** It is an automated system which would help in dispensing sanitizer without the usage of one's own hands. Needed especially during the pandemic so as to prevent the spread chain link.
- **PEAS :**
  - **Performance:** Amount of sanitizer dispensed, speed of dispensing, area covered,
  - **Environment**: User, targets (like objects)
  - **Actuators:** Nozzle/Lid, pipe/arms moving it
  - **Sensors:** Proximity sensor, SONAR, infrared sensors,

- **Environment:** Partially observable, sequential, dynamic, stochastic, single-agent and discrete.

8) Automatic Vacuum Cleaner

- **Define:** The vacuum cleaner that cleans all the dirt within the specified place automatically detecting where the dirt is and dodging other obstacles are.

- **PEAS :**
  - **Performance:** Cleanliness, efficiency, battery life, distance travelled , security, safety
  - **Environment:** Obstacles, dirt, flooring, carpet, Operator
  - **Actuators:** Brushes, Wheels, Suction machine , vacuum extractor
  - **Sensors:** Camera, SONAR, Infrared Sensor, Dirt identification sensor, bump sensors

- **Environment:** Partially observable, Dynamic, Stochastic, Single Agent, Continuous.

9) Taxi Drivers

- **Define:** Automated taxi drivers which are intelligent agents that can be used for building self-driving cars.

- **PEAS :**
  - **Performance:** Safety, time, Abiding traffic rules, Comfort , Cost, Distance travelled
  - **Environment:** Road, Pedestrian , Road Signs, Customer

  - **Actuators**: Accelerator , Steering , Horn , Gear Shifting
  - **Sensors:** Camera, Infrared Sensors(SONAR) , odometer, keyboard

- **Environment**: Partially observable , Dynamic , Stochastic, Single Agent, Continuous.

10)Medical Diagnosis

- **Define:** Medical Diagnosis intelligent agent is used to scan and determine the severity of the medical disease and give efficient recommendations to the detected disease.

- **PEAS:**
  - **Performance:** Precision(Correct Prediction), Healthy, Cost Effective, Suitable to the particular Patient
  - **Environment:** Patient, Doctor , Staff , Hospital
  - **Actuators:** Treatments, Monitor, Referral Doctors
  - **Sensors:** Camera , Keyboard , Patient's answers, Infrared Sensors

- **Environment:** Partially observable, Dynamic, Stochastic, collaborative, Single Agent, Continuous.

CONCLUSION:

Thus, in this experiment I have formulated PEAS for 10 different Intelligent Agents which includes 3 agents related to the current COVID pandemic crisis. PEAS are used to categorize similar agents together and it also delivers the performance measure with respect to the environment, actuators and sensors of the respective agent.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

EXPEIMENT 2

AIM: Identify and analyze Uninformed Search Algorithm to solve the problem.

1. BFS
2. DFS
3. BFID

THEORY:

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

BFS:

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

The breadth-first search algorithm is an example of a general-graph search algorithm.

Breadth-first search implemented using FIFO queue data structure.

Time Complexity T(b) : $1+b^2+b^3+.....+ b^s = O (b^s)$
Where, s= depth of shallowest solution; b is a node at every state.

Space Complexity: $O(b^d)$.

Completeness: BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.

Optimality: BFS is optimal if path cost is a non-decreasing function of the depth of the node.

Advantages:

- BFS is comlete.
- If there are more than one solutions for a given problem, then BFS will provide the minimal solution requiring the least number of steps.

Disadvantages:

- It requires lots of memory since each level of the tree must be saved into memory to expand the next level.
- BFS needs lots of time if the solution is far away from the root node.

DFS:

Depth-first search is a recursive algorithm for traversing a tree or graph data structure.

It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.

DFS uses a stack data structure for its implementation.

Completeness: DFS search algorithm is complete within finite state space as it will expand every node within a limited search tree.

Time Complexity T(n) = $1+ n^2+ n^3 +.....+ n^d = O(n^d)$
where, m= maximum depth of any node and this can be much larger than d (Shallowest solution depth)

Space Complexity: O(nd) => DFS algorithm needs to store only single path from the root node, hence space complexity of DFS is equivalent to the size of the fringe set (=n).

Optimal: DFS search algorithm is non-optimal, as it may generate a large number of steps or high cost to reach to the goal node.

Advantage:

- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.

Disadvantage:

- There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go to the infinite loop

DFID

The iterative deepening algorithm is a combination of DFS and BFS algorithms. DFID search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.

It performs depth-first search up to a certain "depth limit", and it keeps increasing the depth limit after each iteration until the goal node is found.

This search algorithm combines the benefits of Breadth-first search's fast search and depth-first search's memory efficiency.

The iterative search algorithm is useful uninformed search when search space is large, and depth of goal node is unknown.

Completeness: Generates a complete solution if the branching factor is finite.

Space Complexity: O(d).
Time Complexity: $O(b^d)$
Where, b is the branching factor and d is the current depth.

Optimal: DFID algorithm is optimal if path cost is a non- decreasing function of the depth of the node.

Advantages:

- It combines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

Disadvantages:

- The main drawback of DFID is that it repeats all the work of the previous phase.

CODE:

1) BFS & DFS

```
#include <conio.h>
#include<stdio.h>
#include<ctype.h>
#include <stdbool.h>
#define MAX 20

int a[MAX][MAX], visited[MAX], dfs_list[MAX], k = 0, queue[MAX], front = 0, rear = -1, goal[MAX], goal_pending;

bool isGoal(int s, int len) {

  for (int i = 0; i < len; i++) {
    if (s == goal[i]) {
      return true;
    }
  }

  return false;

}

void dfs(int s, int n, int goal_num) {
```

```
  visited[s] = 1; // checked its entry
  //dfs_list[k++] =s ;

  if (isGoal(s, goal_num)) {
    printf(" [%d] \t", s);
    goal_pending--;
  } else {
    printf(" %d \t", s);
  }

  for (int i = 0; i < n; i++) {

    if (a[s][i] == 1 && visited[i] == 0) {
      //passing the current node as parent to traverse in depth.
      dfs(i, n, goal_num);

    }
  }

}

void bfs(int s, int n, int goal_num) {

  for (int i = 0; i < n; i++) {

    if (a[s][i] == 1 && visited[i] == 0) {
      //adding all the unvisited childs
      queue[++rear] = i;
      visited[i] = 1; //make it visited

      //  if(i == goal){
      if (isGoal(i, goal_num)) {
        printf(" [%d] ", i);
        goal_pending--;
      } else {
        printf(" %d ", i);

      }
    }

  }

  if (front <= rear) {
    bfs(queue[++front], n, goal_num);
  }

}
```

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

```
void main() {

  int n, s, key = -1;

  printf("\n Enter the number of Vertices  : ");
  scanf("%d", & n);

  printf("\n Enter the Matrix  of Vertices  : \n");
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      scanf("%d", & a[i][j]);

    }

  }

  printf("\n The Adjacency Matrix  : \n\n");

  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      printf("%d \t", a[i][j]);
    }

    printf("\n");

  }

  do {

    // Clearing
    for (int j = 0; j < MAX; j++) {
      visited[j] = 0;
      queue[j] = 0;
      goal[j] = 0;

    }

    front = 0;
    rear = -1;

    printf("\n \n ***** MAIN MENU ***** \n");
    printf("\n  1)BFS ");
    printf("\n  2)DFS");
    printf("\n -1)Exit ");
    printf("\n \n Enter your Choice : ");
    scanf("%d", & key);

    switch (key) {
```

```
case 1: {

  printf("\n ***** BFS *****");
  printf("\n\n Enter Goal Vertex(s) :");
  printf("\n To exit -1 ");

  int k = 0;
  printf("\n\n Enter Goal %d :", k + 1);
  scanf("%d", & goal[k]);
  while (goal[k] != -1) {
    printf(" Enter Goal %d :", k + 2);
    scanf("%d", & goal[++k]);
  }

  int num_goals = k;
  goal_pending = k;

  //Select your Start vertex.
  printf("\n Select your Start Vertex : ");
  scanf("%d", & s);
  queue[++rear] = s;
  visited[queue[front]] = 1; //make it visited

  printf("\n *****BFS TRAVERSAL*****\n\n");

  if (isGoal(s, num_goals)) {
    printf(" [%d] ", s);
  } else {
    printf(" %d ", s);
  }

  bfs(s, n, num_goals);
  if (goal_pending > 0) {
    printf("\n No. of Goal Nodes still pending : %d", goal_pending);
  } else {
    printf("\n All Goal Nodes reached!!");
  }
  printf("\n\n**********");

  printf("\n");
  break;
}
case 2: {

  printf("\n ***** DFS *****");

  printf("\n\n Enter Goal Vertex(s) :");
  printf("\n To exit -1 ");
```

```
    int k = 0;
    printf("\n\n Enter Goal %d :", k + 1);
    scanf("%d", & goal[k]);
    while (goal[k] != -1) {
      printf(" Enter Goal %d :", k + 2);
      scanf("%d", & goal[++k]);
    }

    int num_goals = k;
    goal_pending = k;

    //Select your Start vertex.
    printf("\n \n Select your Start Vertex : ");
    scanf("%d", & s);

    printf("\n *****DFS TRAVERSAL*****\n\n");
    dfs(s, n, num_goals);

    if (goal_pending > 0) {
      printf("\n No. of Goal Nodes still pending : %d", goal_pending);
    } else {
      printf("\n All Goal Nodes reached!!");
    }
    printf("\n\n*********\n");

    break;

   }

   case -1: {
    printf("\n \n***** END ***** \n");
    break;
   }

   default: {
    printf("\n INVALID KEY!! ");
    break;
   }

   }

  } while (key != -1);

}
```

2) DFID

```
#include <conio.h>
#include<stdio.h>
#include<ctype.h>
#include <stdbool.h>
#define MAX 20

int a[MAX][MAX], visited[MAX], dfs_list[MAX], k = 0, queue[MAX], front = 0, rear = -1
,goal[MAX], goal_pending ;

int MAX_DEPTH ;

bool isGoal(int s , int len){

   for(int i = 0 ; i<len ; i++){
      if(s == goal[i]){
         return true;
      }
   }
   return false ;
}
void dfs(int s , int n,  int goal_num , int depth ,int limit) {

      visited[s] = 1; // checked its entry
      int d= depth ;

       if(isGoal(s ,goal_num)){
         printf(" [%d] ", s);
         goal_pending-- ;
      }else{
         printf(" %d ", s);
      }
      for (int i = 0; i < n; i++) {

         if(depth<=limit){
            if (a[s][i] == 1 && visited[i] == 0) {
            //passing the current node as parent to traverse in depth.
            dfs(i, n, goal_num , ++d , limit);

            }
         }

       }

      }
```

```
bool isGoalPending(){

   if(goal_pending >0){
      return true ;
   }else{
      return false;
   }
}

void DFID(int s, int n, int num_goal){

   printf("\n Level %d :  %d" , 0 , s) ;
   printf("\n %d goal(s) found ", (num_goal- goal_pending)) ;
   printf("\n %d goal(s) pending ", goal_pending) ;

   for(int i=0; i<=MAX_DEPTH ; i++){

   // Clearing
      for (int j = 0; j < MAX ; j++) {
      visited[j] = 0;
      queue[j] = 0;

      }
   // Restoring total goal before next loop
      goal_pending = num_goal ;
      int depth=0 ;

      if(isGoalPending()){
         // Next Level
         printf("\n\n Level %d : " , i+1) ;
         dfs(s, n , num_goal , depth, i) ;
         printf("\n %d goal(s) found ", (num_goal- goal_pending)) ;
         printf("\n %d goal(s) pending ", goal_pending) ;

      }else{

         printf("\n All goals reached!! ") ;
         break ;
   }
}

void main() {
  int n, s,g, key = -1;

  printf("\n Enter the number of Vertices  : ");
```

```
 scanf("%d", &n);

 printf("\n Enter the Matrix  of Vertices  : \n");
 for (int i = 0; i < n; i++) {
  for (int j = 0; j < n; j++) {
    scanf("%d", & a[i][j]);

  }

 }

  // Clearing
  for (int j = 0; j < MAX ; j++) {
    visited[j] = 0;
    queue[j] = 0;
    goal[j] = 0 ;

  }

  printf("\n Enter the MAX DEPTH : ") ;
  scanf("%d", &MAX_DEPTH) ;
  printf("\n Enter the Start Node : ") ;
  scanf("%d", &s) ;


  printf("\n\n Enter Goal Vertex(s) :");
  printf("\n To exit -1 ");

      int k =0 ;
      printf("\n\n Enter Goal %d :", k+1);
      scanf("%d", &goal[k]);
      while(goal[k] != -1){
      printf(" Enter Goal %d :", k+2);
        scanf("%d", &goal[++k]);
      }

      int num_goal = k ;
      goal_pending = k ;

  printf("\n ***** DFID***** \n");
  DFID(s, n, num_goal) ;


}
```

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

OUTPUT

1) BFS

```
C:\Windows\System32\cmd.exe - search
 Enter the number of Vertices  : 9

 Enter the Matrix  of Vertices  :
0 1 0 0 0 0 0 1 0
1 0 1 0 0 0 0 1 0
0 1 0 1 0 1 0 0 1
0 0 1 0 1 1 0 0 0
0 0 0 1 0 1 0 0 0
0 0 1 1 1 0 1 0 0
0 0 0 0 0 1 0 1 1
1 1 0 0 0 0 1 0 1
0 0 1 0 0 0 1 1 0

 The Adjacency Matrix  :

0       1       0       0       0       0       0       1       0
1       0       1       0       0       0       0       1       0
0       1       0       1       0       1       0       0       1
0       0       1       0       1       1       0       0       0
0       0       0       1       0       1       0       0       0
0       0       1       1       1       0       1       0       0
0       0       0       0       0       1       0       1       1
1       1       0       0       0       0       1       0       1
0       0       1       0       0       0       1       1       0


 ***** MAIN MENU *****

  1)BFS
  2)DFS
 -1)Exit

 Enter your Choice : 1

 ***** BFS *****
 Enter Goal Vertex(s) :
 To exit -1

 Enter Goal 1 :2
 Enter Goal 2 :4
 Enter Goal 3 :11
 Enter Goal 4 :-1

 Select your Start Vertex : 6

 *****BFS TRAVERSAL*****

 6  5  7  8  [2]  3  [4]  0  1
 No. of Goal Nodes still pending : 1

*********


 ***** MAIN MENU *****

  1)BFS
  2)DFS
 -1)Exit

 Enter your Choice : 1

 ***** BFS *****

 Enter Goal Vertex(s) :
 To exit -1

 Enter Goal 1 :2
 Enter Goal 2 :6
 Enter Goal 3 :0
 Enter Goal 4 :-1

 Select your Start Vertex : 3

 *****BFS TRAVERSAL*****

 3  [2]  4  5  1  8  [6]  [0]  7
 All Goal Nodes reached!!

*********
```

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

2) DFS

```
 Enter the number of Vertices  : 9

 Enter the Matrix  of Vertices  :
0 1 0 0 0 0 0 1 0
1 0 1 0 0 0 0 1 0
0 1 0 1 0 1 0 0 1
0 0 1 0 1 1 0 0 0
0 0 0 1 0 1 0 0 0
0 0 1 1 1 0 1 0 0
0 0 0 0 0 1 0 1 1
1 1 0 0 0 0 1 0 1
0 0 1 0 0 0 1 1 0

 The Adjacency Matrix  :

0       1       0       0       0       0       0       1       0
1       0       1       0       0       0       0       1       0
0       1       0       1       0       1       0       0       1
0       0       1       0       1       1       0       0       0
0       0       0       1       0       1       0       0       0
0       0       1       1       1       0       1       0       0
0       0       0       0       0       1       0       1       1
1       1       0       0       0       0       1       0       1
0       0       1       0       0       0       1       1       0


 ***** MAIN MENU *****

  1)BFS
  2)DFS
 -1)Exit

 Enter your Choice : 2

 ***** DFS *****
```

```
 Enter Goal Vertex(s) :
 To exit -1

 Enter Goal 1 :2
 Enter Goal 2 :6
 Enter Goal 3 :17
 Enter Goal 4 :-1


 Select your Start Vertex : 8

 *****DFS TRAVERSAL*****

 8       [2]     1       0       7       [6]     5       3       4
 No. of Goal Nodes still pending : 1

*********


 ***** MAIN MENU *****

  1)BFS
  2)DFS
 -1)Exit

 Enter your Choice : 2

 ***** DFS *****

 Enter Goal Vertex(s) :
 To exit -1

 Enter Goal 1 :3
 Enter Goal 2 :4
 Enter Goal 3 :8
 Enter Goal 4 :-1


 Select your Start Vertex : 0

 *****DFS TRAVERSAL*****

 0       1       2       [3]     [4]     5       6       7       [8]
 All Goal Nodes reached!!
```

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

3) DFID

```
C:\Users\meith\Desktop\SEM 5\AI>DFID

 Enter the number of Vertices  : 15

 Enter the Matrix  of Vertices  :
0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

 Enter the MAX DEPTH : 4

 Enter the Start Node : 0


 Enter Goal Vertex(s) :
 To exit -1

 Enter Goal 1 :9
 Enter Goal 2 :15
 Enter Goal 3 :3
 Enter Goal 4 :-1
```

```
 ***** DFID*****

 Level 0 :  0
 0 goal(s) found
 3 goal(s) pending

 Level 1 :  0  1  2
 0 goal(s) found
 3 goal(s) pending

 Level 2 :  0  1  [3]  4  2
 1 goal(s) found
 2 goal(s) pending

 Level 3 :  0  1  [3]  7  8  4  2  5  6
 1 goal(s) found
 2 goal(s) pending

 Level 4 :  0  1  [3]  7  8  4  [9]  10  2  5  11  12  6
 2 goal(s) found
 1 goal(s) pending

 Level 5 :  0  1  [3]  7  8  4  [9]  10  2  5  11  12  6  13  14
 2 goal(s) found
 1 goal(s) pending
C:\Users\meith\Desktop\SEM 5\AI>
```

CONCLUSION:

In this experiment I implemented 3 differnet Uninformed Search Algorithms viz, BFS, DFS and DFID. BFS gives a complete solution but has too large space and time complexities. DFS on the other hand, requires less memory space and than BFS, but it is not complete. There's a possiblity that the states may be re-curring thus not possibility of a solution. Finally, DFID is a combination of BFS and DFS that mutually provides the pros of both the algorithms.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

EXPERIMENT 3

AIM: Implement A* Search Algorithm to reach the goal state.

THEORY:

A * algorithm is a searching algorithm that searches for the shortest path between the initial and the final state. It is used in various applications, such as maps. It is based on heuristic methods which helps to achieve optimality. It also offers completeness i.e., if there is any existing solution, then the algorithm will definitely find it. A* is a different form of the best-first algorithm.

When A* enters into a problem, firstly it calculates the cost to travel to the neighbouring nodes and chooses the node with the lowest cost. If The f(n) denotes the cost, A* chooses the node with the lowest f(n) value. Here 'n' denotes the neighbouring nodes. The calculation of the value can be done as shown below:

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

where,
g(n) = shows the shortest path's value from the starting node to node n
h(n) = The heuristic approximation of the value of the node(Estimated Cost)

The heuristic value has an important role in the efficiency of the A* algorithm. To find the best solution, you might have to use different heuristic function according to the type of the problem.

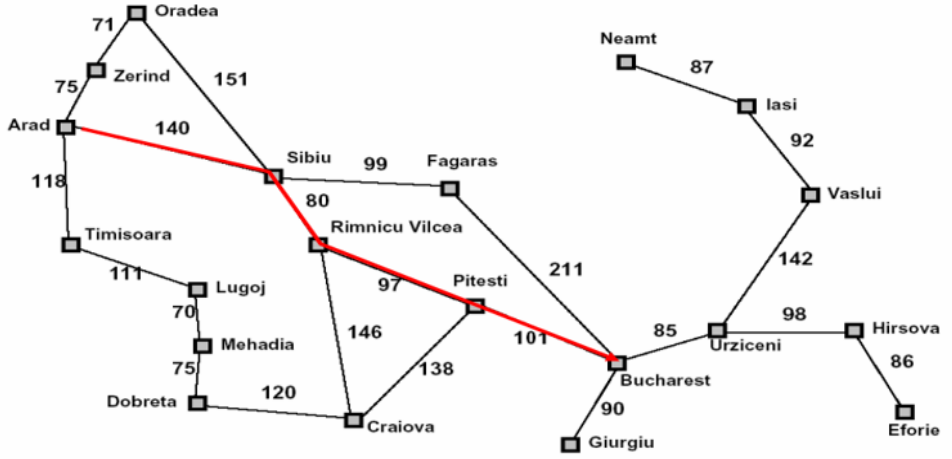
Problem:

- Creation of optimal heuristic functions is a difficult task.
- Space Complexity is $O(b^d)$, as it stores all generated nodes in memory.

Problem Statement: Identify and analyze informed search Algorithm to find an optimal solution from Arad to Bucharest on the given map of Romania. Implement A* Algorithm to reach to the goal state.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

Map of Romania



$h(n)$

Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

CODE:

```
import java.util.*;
import java.util.Arrays;
import java.lang.Integer;
public class AStar {
  public static void calcAstar(int[][] gN, int[] hN, String[] city, int cur, int goal, int N) {
    int min = Integer.MAX_VALUE;
    int minIndex = 0;
    int[] visited = new int[N];
    visited[cur] = 1;

    System.out.print("\n\n ***** A* PATH *****");
    System.out.print("\n\n " + city[cur] + " ");

    while (cur != goal) {

      for (int j = 0; j < N; j++) {

        if (gN[cur][j] > 0) {

          if (visited[j] != 1 && min > gN[cur][j] + hN[j]) {
            min = gN[cur][j] + hN[j];
            minIndex = j;
          }
        }
      }
```

```
      System.out.print(" -> " + city[minIndex] + " ");
      cur = minIndex;
      min = Integer.MAX_VALUE;
      visited[minIndex] = 1;

    }

  }

  public static void main(String args[]) {
    Scanner sc = new Scanner(System.in);

    System.out.print("Enter the number of Nodes : ");
    // int n = sc.nextInt() ;
    int N = 20;

    // int[] hN = new int[N] ;
    // int[] city = new int[N] ;
    int[][] gN = new int[N][N];
int[] hN = {366,0,160,242,161,178,77,151,226,244,241,234,380,98,193,253,329,80,199,374
} ;
String[] city = {"Arad" , "Bucharest" , "Craivoa" , "Dobreta" , "Eforie","Fagaras", "Guirgiu"
,"Hirsova", "Iasi", "Lugoj" ,"Mehadia" , "Neamt" , "Oradea", "Pitesti","Rimnieu Vilcea" ,
"Sibiu", "Timisoara", "Urzieeni","Vaslui" , "Zerind" } ;
    System.out.print("\n\n HEURISTIC VALUES : ");
    for (int i = 0; i < N; i++) {
      if (i == 14) {
        System.out.print("\n " + city[i] + ": " + hN[i]);

      } else {
        System.out.print("\n " + city[i] + " : \t " + hN[i]);

      }

    }

    // Distances
    System.out.print("\n\n G(n) for CITIES \n");

    for (int i = 0; i < N; i++) {
      for (int j = 0; j < N; j++) {
        gN[i][j] = sc.nextInt();
      }
}
    calcAstar(gN, hN, city, 0, 1, N);
  }
}
```

OUTPUT

```
HEURISTIC VALUES :
Arad :              366
Bucharest :         0
Craivoa :           160
Dobreta :           242
Eforie :            161
Fagaras :           178
Guirgiu :           77
Hirsova :           151
Iasi :              226
Lugoj :             244
Mehadia :           241
Neamt :             234
Oradea :            380
Pitesti :           98
Rimnieu Vilcea:     193
Sibiu :             253
Timisoara :         329
Urzieeni :          80
Vaslui :            199
Zerind :            374

***** A* PATH *****

 Arad  -> Sibiu  -> Rimnieu Vilcea  -> Pitesti  -> Bucharest
C:\Users\meith\Desktop\SEM 5\AI>
```

CONCLUSION:

In this experiment I implemented A* Algorithm to identify and analyze informed search Algorithm to find an optimal solution from Arad to Bucharest on the given map of Romania. If the algorithm had only considered h(n), based on the incurring cost then, Fagaras (h(n)= 178) would have been selected instead of Rimnieu Vilcea(h(n)=193) thus deviating from the optimal path. But when considering f(n) = g(n)+h(n), that is considering the actual cost of traveling between the nodes along with the incurring cost, then Rimnieu Vilcea is selected over Fagaras. Thus, A* Algorithm gives an optimal, complete solution.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

EXPERIMENT 4

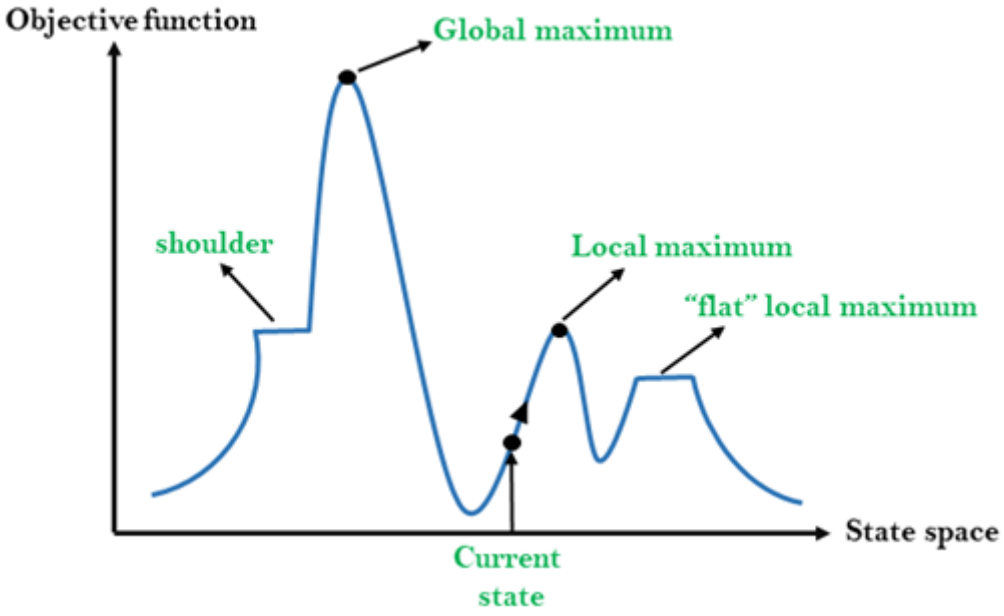
AIM: Program to implement Local Search algorithm using Hill climbing search

THEORY:

Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbour has a higher value.

Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling-salesman Problem in which we need to minimize the distance travelled by the salesman.

A node of hill climbing algorithm has two components which are state and value.



Steepest-Ascent Hill Climbing :

It first examines all the neighboring nodes and then selects the node closest to the solution state as of next node.
Algorithm for Simple Hill climbing :

Step 1 : Evaluate the initial state. If it is a goal state then stop and return success. Otherwise, make initial state as current state.

Step 2 : Repeat these steps until a solution is found or current state does not change

a) Select a state that has not been yet applied to the current state.
b) Initialize a new 'best state' equal to current state and apply it to produce a new state.

c) Perform these to evaluate new state
   i. If the current state is a goal state, then stop and return success.
   ii. If it is better then best state, then make it best state else continue loop with another new state.
d) Make best state as current state and go to Step 2: b) part.

Step 3 : Exit

CODE:

```
import java.util.*;
import java.util.Arrays;
import java.lang.Integer;
import java.lang.Math;

public class hill {
  public static int MAX, MIN;
  public static int[] hill = new int[10];
  public static int hill_pointer = 0;

  public static int heuristic(int[] arr, int end) {

    int total = 0;
    for (int i = 0; i < end; i++) {

      if (i > 0) {
        if (arr[i] != i) {
          total -= i;

        } else {
          total += i;
        }

      }
    }

    return total;
  }

  public static void displayArr(int[] arr, int len) {
    System.out.print("[");
    for (int i = 0; i < len; i++) {
      System.out.print(" " + arr[i] + "");
    }
    System.out.print(" ]");
```

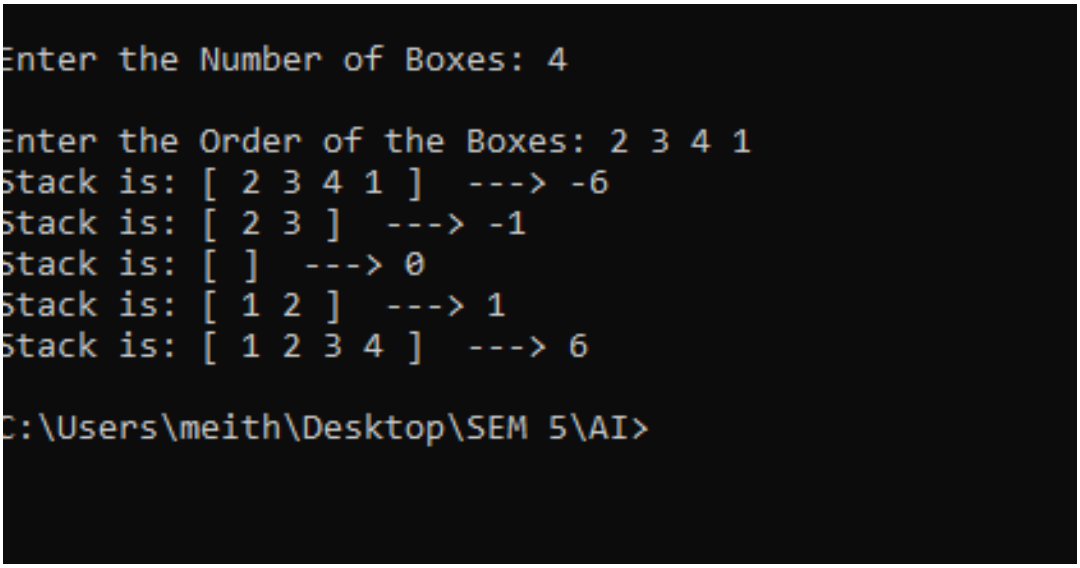
```
}

public static void displayPop(int[] arr, int len) {
  for (int i = 0; i < len; i++) {
    System.out.print(" [" + arr[i] + "]");
  }
}

public static void swap(int elem) {
  hill[hill_pointer] = elem;
}

public static void push(int elem) {
  hill_pointer++;
  hill[hill_pointer] = elem;
}

public static void pop() {
  hill_pointer--;
}

public static void calcHill(int[] arr, int num) {

  int[] popped = new int[num];
  int pointer = 0;
  int step = 0;
  int curHeuristic = heuristic(arr, arr.length);
  int end = arr.length - 1;

  // POPPING
  while (curHeuristic < MAX && end >= 0) {
    int nextHeuristic = heuristic(arr, end);

    if (step % 2 == 0) {
      System.out.print("\n\n Stack is: ");
      displayArr(arr, end + 1);
      displayPop(popped, pointer);
      System.out.print(" --> " + curHeuristic);
      // System.out.print("\n POPPED ARRAY : ") ;
    }

    if (nextHeuristic > curHeuristic) {
      popped[pointer] = arr[end];
      curHeuristic = nextHeuristic;
      end--;
      pointer++;
```

```
      } else {

        break;
      }

      step++;

    }

    // COPY REST INTO HILL
    System.arraycopy(arr, 0, hill, 0, end + 1);
    hill_pointer = end;

    int[] visited_popped = new int[num];

    // END =0
    if (end == 0) {

      int temp = hill[hill_pointer];
      for (int i = 0; i <= pointer; i++) {
        if (hill[hill_pointer] > popped[i]) {
          swap(popped[i]);
        }
      }
      visited_popped[hill[hill_pointer]] = 1;
      popped[pointer] = temp;
    }
    System.out.print("\n\n Stack is : ");
    displayArr(hill, hill_pointer + 1);
    System.out.print(" --> " + curHeuristic);

// PUSH
    while (curHeuristic < MAX) {
      // All empty-- Find next push
      int maxHeuristic = Integer.MIN_VALUE;
      int pushVal = -1;
      int index = -1;
      hill_pointer++;

      for (int i = 0; i <= pointer; i++) {

        if (visited_popped[i] == 0) {
          swap(popped[i]);

          if (maxHeuristic < heuristic(hill, hill_pointer + 1)) {
```

```
            index = i;
            pushVal = popped[i];
            maxHeuristic = heuristic(hill, hill_pointer + 1);
          }
        }
      }
      // pop() ;
      swap(pushVal);
      visited_popped[index] = 1;
      curHeuristic = maxHeuristic;
      step++;
    }

    System.out.print("\n\n Stack is : ");
    displayArr(hill, hill_pointer + 1);
    System.out.print(" --> " + curHeuristic);
    System.out.print("\n GOAL STATE REACHED!!");

  }

  public static void main(String args[]) {

    Scanner sc = new Scanner(System.in);
    System.out.print("\n ENTER THE NUMBER OF BLOCKS : ");
    int num = sc.nextInt();

    int[] arr = new int[num];
    System.out.print("\n ENTER BLOCKS ORDER : ");

    for (int i = 0; i < num; i++) {
      arr[i] = sc.nextInt();
    }

    MAX = num * (num - 1) / 2;
    MIN = MAX * (-1);

    calcHill(arr, num);

    sc.close();

  }

}
```

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

OUTPUT:



CONCLUSION:

In this experiment I implemented Hill Climbing Search using Steepest Ascend with step size equal to 2. Here I have calculated a Heuristic function to provide the heuristic value of each state with respect to the previous state, if the value is greater then we move ahead else the previous state was better than the next state. Finally, we reach the goal state. In steepest ascend instead of moving step wise we skip the smaller steps in the same direction, keeping in mind the the jump is not too large.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

EXPERIMENT 5

AIM: Program on Genetic Algorithm to solve an optimization problem in AI.

THEORY:

Genetic Algorithms(GAs) are adaptive heuristic search algorithms that are based on the ideas of natural selection and genetics. Genetic algorithms simulate the process of natural selection which means those species who can adapt to changes in their environment are able to survive and reproduce and go to next generation.

The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

They are commonly used to generate high-quality solutions for optimization problems and search problems.

A typical genetic algorithm requires:

1. a genetic representation of the solution domain,
2. a fitness function to evaluate the solution domain.

Application of Genetic Algorithms

- Recurrent Neural Network
- Mutation testing
- Code breaking
- Filtering and signal processing
- Learning fuzzy rule base

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

PROBLEM STATEMENT:

Consider the population of string with 10 bits each. The objective function can assume number of 1's in a given string. The fitness function then perform "divide by 10"operation to normalizer the objective function. Show computation of minimum of two generations. Assume crossover rate as 0.5 and mutation probability rate as 0.05.

CODE:

```
import java.util.*;
import java.lang.Integer;
import java.util.Arrays;
import java.text.DecimalFormat;

public class Genetic{
   public static int MAX =1024 , BITS=10, string=4;
   public static int[][] GeneString= new int[string][BITS+1] ;
   private static final DecimalFormat df = new DecimalFormat("0.00");
   public static String displayGene(int[] arr){
      String str ="" ;
      for(int i=BITS; i>0 ; i--){
        str += ""+arr[i]+"" ;
      }

      return str;
   }


   public static int toDecimal(int[] arr){

      int num = 0;
      for(int i =arr.length -1; i>0; i--){
         num = num*2 + arr[i];
      }

      return num ;
   }


   public static int[] getGenes(){

      int m = (int)(Math.floor((Math.random())*MAX));
      int[] arr = new int[BITS+1] ;
      arr[0] = m;
      int count=1;


      while(m>0){
```

```
        arr[count] = m%2 ;
        count++ ;
        m /=2;
    }

    return arr;
}


public static double[][] calcObjective(){

    double avg_f=0 , total_f=0 ,  Max_f=Integer.MIN_VALUE ;
    double[] arr_fX = new double[string];
    double[][] result = new double[string+3][4]; // Strings,Sum,Avg,Max

    for(int i=0; i<string ; i++){
        int numOnes =0  ;
        for(int j=1; j<BITS+1 ; j++){

            if(GeneString[i][j] == 1){
                numOnes++ ;
            }
        }

        double fX = (double)numOnes/10 ;
        arr_fX[i] =fX ;
        total_f += fX;

        if(Max_f < fX){
            Max_f = fX ;
        }

    }

    avg_f = total_f/string;
    double min_expected = Integer.MAX_VALUE ;


    // RESULTS
    for(int i=0; i<string ; i++){

        // F(x)
        result[i][0] = arr_fX[i];
        // Fi / tot_f
        result[i][1] = (double)arr_fX[i]/total_f ;
        // Fi / avg_f
        result[i][2] = (double)arr_fX[i]/avg_f ;
        // ACTUAL COUNT
        result[i][3] = Math.round(result[i][2]) ;
```

```
            if(min_expected > result[i][2]){
                min_expected= result[i][2] ;
            }
        }

        // SUM
          result[string][0] =total_f;
          result[string][1] = (double)(total_f/total_f) ;
          result[string][2] = (double)(total_f/avg_f) ;
        //   AVG
        result[string+1][0] =avg_f;
        result[string+1][1] = avg_f/total_f ;
        result[string+1][2] = avg_f/avg_f ;
        //  MAX
        result[string+2][0] =Max_f;
        result[string+2][1] = Max_f/total_f ;
        result[string+2][2] = Max_f/avg_f ;
        return result;
    }

    public static int[] displayInitialise(double[][] result ,int iter){
        double min_Actual=Integer.MAX_VALUE ;
        double max_Actual=Integer.MIN_VALUE ;
        int min_index= -1,  max_index= -1;
        int[] index = new int[2] ;

        System.out.print("\n\n String
No\tPopulation(P"+iter+")\tX\tf(X)\tFi/SUM(f)\tfi/f\tActual Count" ) ;

        for(int i=0; i<string; i++){
            System.out.print("\n    "+(i+1)+"\t\t"+displayGene(GeneString[i])+"\t"+
Math.round(GeneString[i][0])+"\t"+
            df.format( result[i][0])+"\t"+df.format(result[i][1])
+"\t\t"+df.format(result[i][2])+"\t"+
            Math.round(result[i][3])  );

            if(min_Actual> result[i][3]){
                min_index = i;
                min_Actual= result[i][3] ;
            }
            if(max_Actual<= result[i][3]){
                max_index = i;
                max_Actual= result[i][3] ;
            }
        }
        index[0] = min_index;
        index[1] = max_index ;
```

```
        // SUM
        System.out.print("\n\n  \t\tSUM\t\tSUM(f)\t"+
        df.format( result[string][0])+"\t"+df.format(result[string][1])
+"\t"+df.format(result[string][2]) );
        // AVG
        System.out.print("\n  \t\tAVG\t\tf\t"+
        df.format( result[string+1][0])+"\t"+df.format(result[string+1][1])
+"\t"+df.format(result[string+1][2]) );
        // MAX
        System.out.print("\n  \t\tMAX\t \t"+
        df.format( result[string+2][0])+"\t"+df.format(result[string+2][1])
+"\t"+df.format(result[string+2][2]) );

        return index;
    }

    public static void replaceWeak(int[] index ){

        System.arraycopy(GeneString[index[1]], 0, GeneString[index[0]], 0,
GeneString[0].length);
    }

    public static int[] replaceMate(int cur , int mate , int crossSite){
        int length = 5 ;
        int[] arr = new int[BITS+1] ;


        for(int i=crossSite+1; i<BITS+1; i++){
            arr[i] = GeneString[cur][i] ;
        }

        for(int i=1; i<=crossSite; i++){
            arr[i] = GeneString[mate][i] ;
        }

        return arr ;
    }

    public static void displayCrossOver(int iter){

        System.out.print("\n\n\n CROSS OVER : ");
        System.out.print("\n String No\tPopulation(P"+iter+")\tMate\tCrossover Site\tNew
Pop(P"+(iter+1)+")\tNEW X" ) ;
        int[][] newGeneString= new int[string][BITS+1] ;


        for(int i=0; i<string; i++){
            int mate = (i%2==0)? i+1: i-1 ;
            int crossSite= 5;
            // int[] arr = new int[BITS+1] ;
```

```
            newGeneString[i] = replaceMate(i , mate, crossSite) ;
            newGeneString[i][0]= toDecimal(newGeneString[i] ) ;

            System.out.print("\n
"+(i+1)+"\t\t"+displayGene(GeneString[i])+"\t"+(mate+1)+"\t5\t\t"+
            displayGene(newGeneString[i])+"\t"+newGeneString[i][0] );


        }

        GeneString= newGeneString;


    }

    public static void mutation(){

        System.out.print("\n\n String No\tPopulation(P2)\tP2 X\t  Mutated
Population\tMUTATED X" ) ;
        int[][] gen2 = new int[string][BITS+1] ;

        for(int i=0; i<string;i++){
            int index;
            System.arraycopy(GeneString[i], 0, gen2[i], 0, BITS+1);

            do{
                index =(int)(Math.floor((Math.random()*BITS))) +1;
                if(GeneString[i][index] == 0){
                    GeneString[i][index] =1; //mutate
                    break;
                }else{

                }
            }while(GeneString[i][index] != 0) ;
            GeneString[i][0] = toDecimal(GeneString[i]);

            System.out.print("\n  "+(i+1)+"\t\t"+displayGene(gen2[i])+"\t"+ gen2[i][0]+"\t
"+displayGene(GeneString[i])+"\t\t"+GeneString[i][0] ) ;
        }

    }

    public static void GeneticAlgo(){

        System.out.print("\n\n\n *****GENERATION 1*****");
        for(int i=0; i<string; i++){
            GeneString[i] = getGenes() ;
            // System.out.print("\n "+GeneString[i][0]+" : "+displayGene(GeneString[i] ));
        }
```

```
        double[][] result = calcObjective() ;
        int[] index = displayInitialise(result, 0);
        // REPLACE WEAK
        replaceWeak(index);
        displayCrossOver(0);

        System.out.print("\n\n\n *****GENERATION 2*****");
        double[][] result2 = calcObjective() ;
        index = displayInitialise(result2, 1);
        // REPLACE WEAK
        replaceWeak(index);
        displayCrossOver(1);

        // MUTATION
        System.out.print("\n\n\n *****FINAL GENERATION POST MUTATION*****");
        mutation() ;

    }

public static void main(String args[]){

    Scanner sc = new Scanner(System.in) ;

    System.out.print("\n Enter the number of Strings :");
     string= sc.nextInt() ;

    GeneticAlgo();
    sc.close() ;
    }
}
```

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

OUTPUT:

```
C:\Windows\System32\cmd.exe
Enter the number of Strings : 4


*****GENERATION 1*****

String No      Population(P0)  X       f(X)    Fi/SUM(f)       fi/f    Actual Count
   1           0000011001      25      0.30    0.19            0.75    1
   2           0110101110      430     0.60    0.38            1.50    2
   3           0100101010      298     0.40    0.25            1.00    1
   4           0110000001      385     0.30    0.19            0.75    1

               SUM             SUM(f)  1.60    1.00    4.00
               AVG             f       0.40    0.25    1.00
               MAX             0.60    0.38    1.50


CROSS OVER :
String No      Population(P0)  Mate    Crossover Site  New Pop(P1)     NEW X
   1           0110101110      2       5               0110101110      430
   2           0110101110      1       5               0110101110      430
   3           0100101010      4       5               0100100001      289
   4           0110000001      3       5               0110001010      394


*****GENERATION 2*****

String No      Population(P1)  X       f(X)    Fi/SUM(f)       fi/f    Actual Count
   1           0110101110      430     0.60    0.32            1.26    1
   2           0110101110      430     0.60    0.32            1.26    1
   3           0100100001      289     0.30    0.16            0.63    1
   4           0110001010      394     0.40    0.21            0.84    1

               SUM             SUM(f)  1.90    1.00    4.00
               AVG             f       0.47    0.25    1.00
               MAX             0.60    0.32    1.26


CROSS OVER :
String No      Population(P1)  Mate    Crossover Site  New Pop(P2)     NEW X
   1           0110001010      2       5               0110001110      398
   2           0110101110      1       5               0110101010      426
   3           0100100001      4       5               0100101010      298
   4           0110001010      3       5               0110000001      385


*****FINAL GENERATION POST MUTATION*****

String No      Population(P2)  P2 X    Mutated Population      MUTATED X
 1             0110001110      398     0110001111              399
 2             0110101010      426     0111101010              490
 3             0100101010      298     0100101011              299
 4             0110000001      385     0111000001              449
C:\Users\meith\Desktop\SEM 5\AI>
```

CONCLUSION:

In this experiment, I implemented Genetic Algorithm for a population of string with 10 bits each, the objective function is the number of 1's in a given string and the fitness function is "divide by 10" operation to normalize the objective function, assuming the crossover rate as 0.5 and mutation probability rate as 0.05. The algorithm iterated for 2 generations and in the final generation any random 0 of each population string was mutated to 1. The final mutated generation is displayed in the output.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

EXPERIMENT 6

AIM: Case study on Artificial Intelligence applications.

TOPIC: Application of Convolutional Neural Network in Handwritten Chinese Character Recognition

ANALYSIS:

The paper was based on the recognition and classification of the HWDB dataset which was established by the National Laboratory for Pattern Recognition (NLPR) of the Institute of Automation, Chinese Academy of Sciences. Results obtained by the earlier models on HWDB handwritten Chinese character recognition, based on convolutional neural networks differ noticeably at different learning rates; if the learning rate was too low, the CNN's convergence speed slowed, resulting in low training efficiency. This research mainly focused on improving the accuracy by pre-processing of the dataset, selecting an apt activation function, construction and parameterizing the CNN and finally optimizing the model. The research successfully achieved an accuracy of 93% in the recognition and classification of handwritten Chinese characters.

In the HWDB dataset, the handwriting samples were made on paper by 1,020 writers using Anoto pens. The size and the label of the pictures were not uniformly processed which were needed to be pre-processed, including unify the size of the pictures and label the corresponding pictures. The font shape of Chinese characters were square, and the relative size of the original image were distributed between 40–100 pixels, thus unified to a pixel size of 50∗50. Four activation functions: sigmoid, Tanh, ReLU and Mish were selected for data analysis using the CNN model to compare the influence of each activation functions on accuracy. For the Sigmoid activation function, the accuracy rate was very low, around 10%, clearly indicating that it wasn't suitable in that scenario. When compared to Mish and ReLU activation functions, Tanh had a greater overall learning efficiency, could converge to a higher accuracy rate faster, and the final recognition accuracy could be stabilised at around 85%. Although, in the final recognition rate, Mish can be stabilized at about 90% with the fluctuation range is about 2%. Tanh, ReLU, and Mish had produced convincing results but among these three activation functions the recognition rate and stability of the Mish activation function produced the best results.

After selecting an apt activating function, an optimising algorithm was formulated in order compensate with the large fluctuations in the value of loss function around the minimum value, which hinder to reach to the optimal value. Three Loss functions: SmoothL1Loss(), BCEWithLogitsLoss() and MSELoss() were selected to measure the inconsistency between the predicted value and the real value of the model. No significant influence was observed on the final recognition accuracy, but a certain influence was seen on the initial convergence speed.

Finally, through the learning rate tuning, the weights of the CNN model were adjusted. As the learning rate reduced, the accuracy rate had a significant decrease after training 30,000 times.

The appropriate range of learning rate determined was between 0.01 ~0.08. With the learning rate being 0.08, the highest accuracy is obtained, which could be stabilized to about 93%.

CONCLUSION:

In this experiment, I performed case study on ‘Application of Convolutional Neural Network in Handwritten Chinese Character Recognition’ research paper published on IEEEE. The paper was based on the recognition and classification of the HWDB dataset which was established by the National Laboratory for Pattern Recognition (NLPR) of the Institute of Automation, Chinese Academy of Sciences. The research mainly focused on improving the accuracy by pre-processing of the dataset by unifying the images to 50*50 pixels, selecting an apt activation function, construction and parameterizing CNN layers. and finally optimizing the model. The research successfully achieved an accuracy of 93% in the recognition and classification of handwritten Chinese characters using learning rate 0.08 along with Mish activation function.

Application of Convolutional Neural Network in Handwritten Chinese Character Recognition

Hongliang Guo[1],Likun Ai[2],Shuxin Chen[1,2*]
1. Department of Computer Science and Technology, Renai College of Tianjin University, Tianjin, China
2. College of Mechanical and Electrical Engineering, Qiqihar University, Qiqihar, China
304473675@qq.com, ailikun034@163.com, shuxinfriend@126.com
Corresponding Author: Shuxin Chen Email:shuxinfriend@126.com

***Abstract*—The handwritten Chinese character data set HWDB, designs and trains a convolutional neural network to recognize handwritten Chinese characters. On this basis, this article tested and studied the effects of activation function, loss function, and learning rate on the performance of convolutional neural networks, and further analyzed the reasons for their effects, and accumulated for the subsequent application of convolutional neural networks in other aspects. The valuable experimental experience and theoretical basis were laid. This paper is based on the relevant technical process, analyzed the experimental effect, summarized the influence of important parameters, and summarized the design ideas of neural network for different image types, laying a foundation for further research in the future.**

Keywords—deep learning; convolutional neural network; handwritten Chinese character recognition; learning rate

I. INTRODUCTION

Deep learning is an important research direction in the field of computer science. By designing artificial neural networks about the internal logic of sample data, and then achieve the effect of predicting the same type of data, such as text recognition, image classification, and speech recognition. The data field is in the stage of rapid development of the Internet and computer performance, and various types of data are also growing rapidly[1]. The mass data of text, pictures, and sound have also been sorted out more reasonably, which also provided a necessary for the further development of deep learning. These data can mine effective information.

Artificial intelligence emerged in the 1950s, has experienced three major developments, which is now in the third stage of development[2]. In the past five years, related applications of deep learning have become more and more closely related to our lives. For example, specific applications for image feature learning include OCR recognition, face recognition, and gesture recognition. Especially the auto-driving technology that is being vigorously developed in the automotive-related field mainly relies on the analysis of the characteristics of road conditions. HWDB handwritten Chinese character recognition based on convolutional neural network has obvious differences in the results produced at different learning rates, if the learning rate is too small, the convergence speed of the convolutional neural network will slow down, resulting in low training efficiency. It can be expected that the development of this technology will radiate to all aspects of society in the future.

II. THE APPLICATION OF CNN IN HANDWRITTEN CHINESE CHARACTER RECOGNITION

The HWDB dataset was established by the National Laboratory for Pattern Recognition (NLPR) of the Institute of Automation, Chinese Academy of Sciences. The handwriting samples were made on paper by 1,020 writers using Anoto pens. The dataset used in this experiment with 10 sets of data in the HWDB dataset. The PNG image of the dataset are obtained through the open source dataset, as shown in Fig.1.

Fig. 1. HWDB raw data picture

A. HWDB image Dataset Preprocessing

By observing the above Fig.1, we can find the original data set, the size and label of the pictures are not uniformly processed. Therefore, if we want to use the dataset, we need to perform certain preprocessing. Including unify the size of the pictures and label the corresponding pictures.

Fig. 2. Pre-processed HWDB image data

Taking into account that the font shape of Chinese characters is square, and the relative size of the original image is distributed between 40-100 pixels, here it is unified to a pixel size of 50*50. The labeling method is shown in Fig.2.

B. Importance of experimental results

(1) Sigmoid

The function image of sigmoid is shown in Fig.3. Its function is expressed as formula (1):

$$f(x) = \frac{1}{1+e^{-x}}) \tag{1}$$

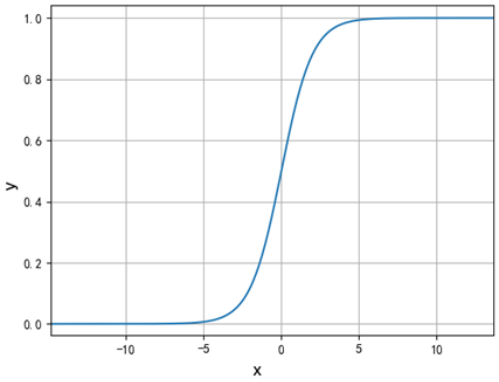


Fig. 3. Sigmoid function image

(2) Tanh

The Tanh function is expressed as formula (2):

$$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2}$$

(3) ReLU

The ReLU function is expressed as formula (3):

$$f(x) = max(0, x) \tag{3}$$

(4) Mish

The Mish function is expressed as formula (4):

$$f(x) = x * tanh(ln(1 + e^{x})) \tag{4}$$

The boundlessness of the Mish activation function avoids the saturation due to capping. In theory, negative values are slightly allowed instead of hard zero boundaries like in ReLU. At the same time, the smooth activation function allows more information to penetrate the neural network, resulting in better accuracy and generalization ability. The function image of Mish is shown in Fig. 4.

C. Importance of Sampling Construction Convolutional Neural Network Model

Based on the experience summarized in the previous design of the MNIST neural network, the structure of the convolutional neural network is designed as:

(1) Input layer: Process the HWDB data set in advance to obtain grayscale data with the size of 50*50*1;

(2) Convolutional layer 1: Use 6 layer(5*5) convolution kernels to perform convolution operation on the image of the input layer, with the moving step of 1;

(3) Pooling layer 1: Dimensionality reduction is performed on the output features of convolutional layer 1 through the maximum pooling operation. The size of the pooling window is 2*2, and the moving step of the pooling operation is 2.

(4) Convolutional layer 2: Use 16 layer (3*3) convolution kernels to perform convolution operations on the image, with a moving step of 1.

(5) Pooling layer 2: Same pooling layer 1.

(6) Fully connected layer: There are 200 nodes in total to connect the data passing through the pooling layer 2.

(7) Output layer: 10 nodes, corresponding to the classification of the picture, in order to achieve data normalization, the Mish activation function is used the dataset and parameters of the training process:

I. Number of training sets: 1800 sheets;

II. The number of test sets: 200 sheets;

III. Training times: 30000 times;

IV. Every 100 times of training, use the test set to count the change of accuracy rate.

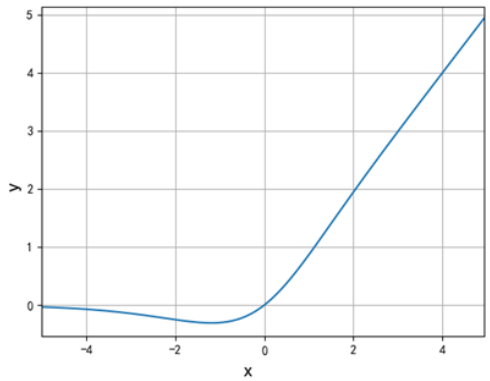


Fig. 4. Mish activation function image

D. Experimental results using different activation functions

The above four activation functions sigmoid, Tanh, ReLU and Mish were selected for data analysis. The use of convolutional neural network HWDB for handwritten Chinese character recognition is shown in Fig. 5, which compared the influence of four activation functions on accuracy. The X-axis value is the number of training times, and the Y-axis value is the accuracy. Conclusion of the experiment:

(1) For the Sigmoid activation function, the accuracy rate is basically 10%. Compared with other activation functions, it can be clearly found that the sigmoid activation function is not suitable for image classification under the convolutional neural network in this problem scenario.

(2) For the Tanh activation function, its overall learning efficiency is higher. Compared with other Mish and ReLU

activation functions, it can converge to a higher accuracy rate faster, and the final recognition accuracy can be stabilized at about 85%. Compared with the other two, the accuracy rate is slightly lower, and the fluctuation range is about 3%, which is relatively stable.

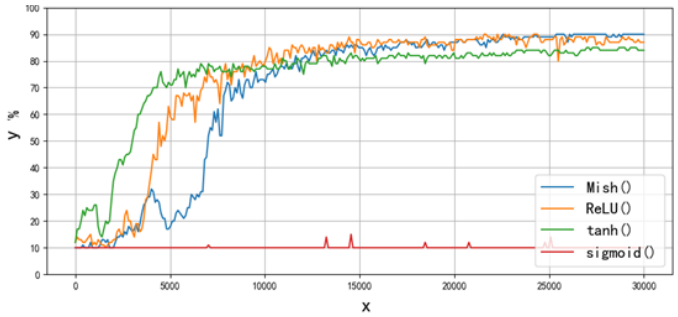


Fig. 5. Comparison of effects of different activation functions

(3) For ReLU activation function and Mish activation function: In terms of convergence speed, ReLU is faster; in the final recognition rate, Mish can be stabilized at about 90%, the fluctuation range is about 2%, and the accuracy of ReLU can be basically stabilized at About 87%, but its fluctuation range is about 5%.

(4) For the three activation functions of Tanh, ReLU, and Mish, the recognition of handwritten Chinese characters can be successfully realized. Among the three, the recognition rate and stability of the Mish activation function are the best, but for The sigmoid activation function has a final accuracy rate of only 10%, which is a training failure.

III. EXPERIMENT OPTIMIZATION

Optimization algorithm is almost the most necessary algorithm for machine learning model training. In the early stage of algorithm optimization, learning is accelerated, making the model more accessible to local or global optimal solutions. However, in the later period, there will be a large fluctuation, and even the value of loss function will fluctuate around the minimum value, which makes it difficult to reach the optimal value. In the preceding chapter, the difference between model distribution and target distribution are calculated by samples. However, in practice, the label of an event is known to be invariant, that is, the entropy of target distribution is constant. Therefore, instead of calculating KL divergence, the loss values of model distribution and target distribution can be obtained by calculating cross entropy. It is known that the difference between model distribution and target distribution can be replaced by cross entropy under the condition that the target distribution is constant. If the target distribution changes, then cross entropy cannot be used.

(1) Mean square error loss function

In PyTorch, the mean square error loss function is the MSELoss loss function, which is expressed as formula (5):

$$MSE(x,y) = \frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n} \tag{5}$$

(2) Huber loss function

In PyTorch, it is the SmoothL1Loss loss function. The function is expressed as formula (6):

$$Huber(x_i, y_j) = \begin{cases} 0.5(x_i - y_i)^2, & if \quad |x_i - y_i| < 1 \\ |x_i - y_i| - 0.5, & otherwise \end{cases} \tag{6}$$

(3) Cross entropy loss function

For the cross-entropy loss function used for classification problems, the sigmoid layer is added, which can make the numerical effect more stable. The corresponding specific function is named BCEWithLogitsLoss, and the function is expressed as formula (7):

$$loss(x_i, y_j) = -w_i[y_i * logx_i + (1 - y_i) * log(1 - x_i)] \tag{7}$$

Among them, x is the actual output, y is the label of the sample, and w is the weight of the item.

A. Experimental Results Using Different Loss Functions

The above three common loss functions, SmoothL1Loss(), BCEWithLogitsLoss() and MSELoss(), are selected. The loss function is used to measure the inconsistency between the predicted value and the real value of the model, so as to measure the prediction quality of the model. The convolutional neural network is used for handwriting Chinese character recognition as shown in Fig. 6, the feedback influence of different loss functions on accuracy. The X-axis value is the number of training times, and the Y-axis value is the accuracy.

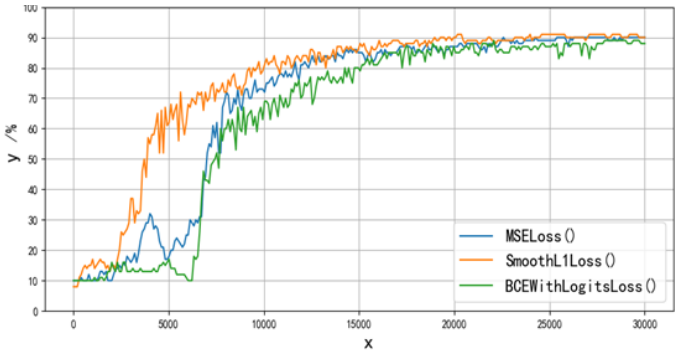


Fig. 6. Effect comparison diagram of different loss functions.The convolutional neural network structure and dataset default parameters;Activation function: Mish; Loss function: Huber loss function.

According to the analysis of the change curve of accuracy in Fig. 6, the three loss functions have no significant influence on the final recognition accuracy, but have a certain influence on the initial convergence speed. On the whole, it can be seen better than the effect of using Huber loss function.

B. Experimental Results Using Different Learning Rate

The learning rate is a super parameter, which controls the degree of adjusting the network weight. The lower the value, the slower it will descend along the gradient, so as to ensure that we will not miss any local lowest point, but it will take a long time to converge. 0.1 in the optimizer is the learning rate

for the training parameter. The learning rate is big shock, which does not converge, the learning rate is small convergence speed is slow. Experimental data show the influence of learning rate on the accuracy of handwritten Chinese character recognition using convolutional neural network. The X-axis is the number of training times, and the Y-axis is the accuracy. As FIG. 7 shows the influence of learning rate on the recognition accuracy within the range of 0.1~0.01, and FIG. 8 shows the influence of learning rate within the range of 0.01~0.001. These two groups of comparative experiments comprehensively cover the effect on learning rate of the recognition accuracy about the range of 10^{-2} and 10^{-3}, and also verify the adverse effect whether high or low learning rate on the training results.

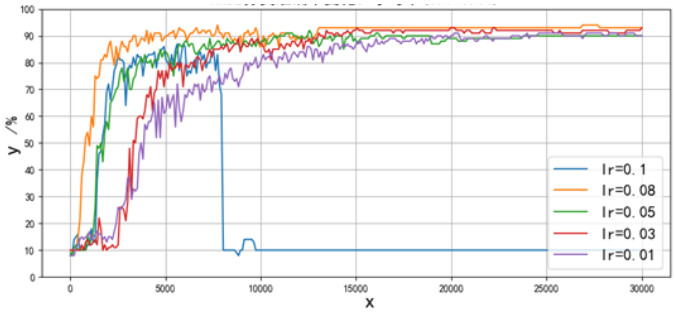


Fig. 7. Effect comparison diagram of the learning rate (0.1~0.01) on accuracy

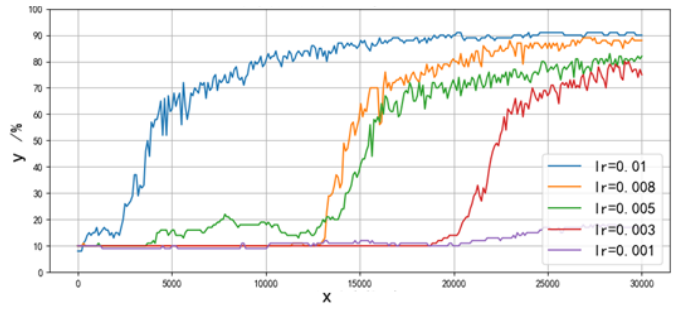


Fig. 8. Effect comparison diagram of the learning rate (0.01~0.001) on accuracy

Through the analysis of Fig. 8, it is concluded that as the learning rate shrinks, the main improvement phase of the learning rate during the training process is significantly shifted back, and the accuracy rate also has a significant decrease after training 30,000 times. According to the theory, the learning rate is too small, the convergence speed of the convolutional neural network will slower down, resulting in low training efficiency.

In summary, in this experiment, it is more appropriate to set the learning rate between 0.01~0.08. A neural network with relatively stable performance and better generalization ability can be trained within a reasonable number of training times. While the learning rate is 0.08, the accuracy is the highest, which can be stabilized at about 93%.

IV. Algorithm Summary

Through the study of deep learning, this paper realizes the recognition and classification of the HWDB dataset summarized the experimental process of the former, and successfully achieved 93% accuracy in the recognition and classification of handwritten Chinese characters.

(1) Through exploring the development process of deep learning and major technological breakthroughs at various stages, we have a certain understanding of deep learning and the range of problems.

(2) By summarizing the experimental process of the former, we successfully realized the training and recognition and classification of handwritten Chinese characters by convolutional neural network. This process includes data preprocessing, convolutional neural network construction, parameter comparison and debugging, and finally realized 93% accuracy rate.

Through the research and practice of image feature learning, it is found that convolutional neural networks have better results in processing image classification problems. Compared with other neural networks, convolutional neural networks can more extract image features. And the feature extraction is not defined by humans, but through massive data, the neural network learns independently through the error back propagation algorithm, this learning strategy is directly determined, and the algorithm can be applied to other practical problems, such as food energy analysis, garbage classification, item identification. But it is not without its drawbacks. how to use these data to mine effective information, this has also become a research direction with great potential value. The content that still needs to be studied in the future.

Acknowledgment

This work is supported by the National Science Foundation for Young Scientists of China under Grant No. 11800313 and University-Industry Collaborative Education Program under Grant No. 201802252028.

References

[1] B. Du, M. Zhang, L. Zhang, et al. " Pltd: Patch-based low-rank tensor decomposition for hyper-spectral images. " IEEE Transactions on Multimedia, vol.19, issue 1, pp. 67-79, 2016.

[2] L.Y. Zhang, and B.R. Tao, "A framework for ontology integration based on genetic algorithm." Journal of Intelligent & Fuzzy Systems 30, pp.1651, 2016.

[3] Q. Zhang, L. T. Yang, Z. Chen, et al. "High-order possibilistic c-means algorithms based on tensor decompositions for big data in iot. " Information Fusion, vol.39, pp. 72-80, 2018.

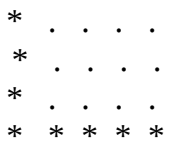
[4] L.Y. Zhang, J.D. Ren and Xianwei Li, "OIM-SM: A method for ontology integration based on semantic mapping", Journal of Intelligent & Fuzzy Systems vol.32, pp. 1983–1995, 2017.

[5] S.X. Chen, W.M. Sun, X. Kong, "Clustering analysis of line indices for LAMOST spectra with AstroStat." Research in Astronomy and Astrophysics, vol.18, issue 6, pp.78, 2018.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

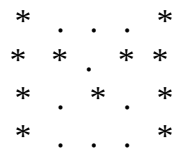
EXPERIMENT 7

AIM: To design a pattern classifier using perceptron training algorithm for the given pattern of alphabets L and M.



Pattern (a): L

and



Pattern (b): M

THEORY:

In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class. It is a type of linear classifier, i.e. a classification algorithm that makes it's predictions based on a linear predictor function combining a set of weights with the feature vector.

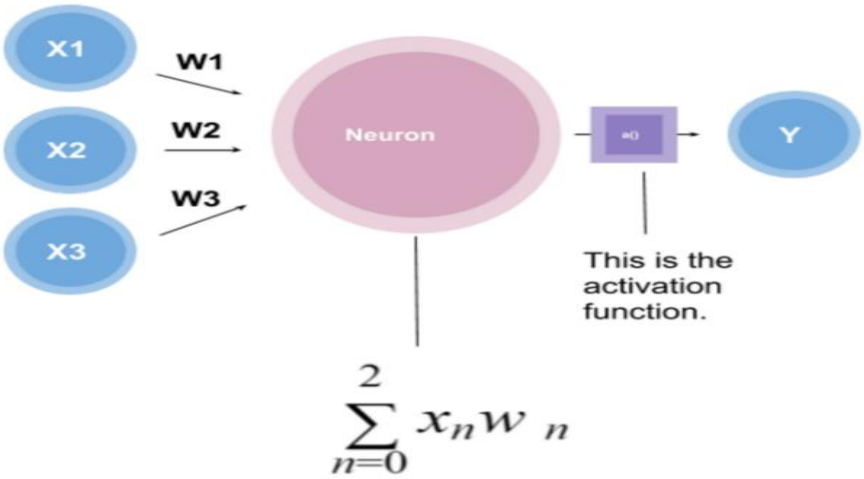
Perceptron has a threshold function: a function that maps its input X(a real-valued vector) to an output value f(X)(a single binary value):

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

where **w** is a vector of real-valued weights, **w.X** is the dot product, $\sum \mathbf{w_i.X_i}$ where m is the number of inputs to the perceptron, and **b** is the bias. The bias shifts the decision boundary away from the origin and does not depend on any input value.

The value of f(X) => (0 or 1) is used to classify X as either a positive or a negative instance, in the case of a binary classification problem. If b is negative, then the weighted combination of inputs must produce a positive value greater than |b| in order to push the classifier neuron over the 0 **threshold**. Spatially, the bias alters the position (though not the orientation) of the decision boundary. The perceptron learning algorithm does not terminate if the learning set is not linearly separable. If the vectors are not linearly separable learning will never reach a point where all vectors are classified properly.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical



A perceptron is an artificial neuron using the **Heaviside step function** as the activation function.

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import datasets

class Perceptron:
 def __init__(self, learning_rate=0.01, n_iters=1000):
  self.lr = learning_rate
  self.n_iters = n_iters
  self.activation_func = self._unit_step_func
  self.weights = None
  self.bias = None

 def fit(self, X, y):
  n_samples, n_features = X.shape
  # initialise parameters
  self.weights = np.zeros(n_features)
  self.bias = 0
  y_ = np.array([1 if i > 0 else 0 for i in y])
  for _ in range(self.n_iters):
    for idx, x_i in enumerate(X):
      linear_output = np.dot(x_i, self.weights) + self.bias
      y_predicted = self.activation_func(linear_output)
      # Perceptron update rule
      update = self.lr * (y_[idx] - y_predicted)
      self.weights += update * x_i
```

```
      self.bias += update

  def predict(self, X):
   linear_output = np.dot(X, self.weights) + self.bias
   y_predicted = self.activation_func(linear_output)
   return y_predicted

  def _unit_step_func(self, x):
   return np.where(x >= 0, 1, 0)

if __name__ == "__main__":

  def accuracy(y_true, y_pred):
   accuracy = np.sum(y_true == y_pred) / len(y_true)
   return accuracy

  def displayIP(arr):
     for i in range(len(arr)):
       if i%5==0 and i !=0:
         print("")

       print(arr[i], end=" ")

# TRAINING DATA
  X = np.array([(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1),
  (1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1),
  (1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1),
  (1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1),
  (1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1),
  (1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1),
  (1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1),
  (1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1),
  (1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1),
  (1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1),
  (1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1),
  (1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1),
  (1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1),
  (1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1),
  (1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1),
  (1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1),
  (1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1),
  (1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1),
  (1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1),
```

```
 (1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1)])

 y = np.array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,
random_state=123 )

 p = Perceptron(learning_rate=0.01, n_iters=50)
 p.fit(X_train, y_train)
 predictions = p.predict(X_test)
 # ACCURACY
 print("Perceptron Classification Accuracy- TEST SET : ", accuracy(y_test,
predictions))

# TEST 1
test_1 = np.array([1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1])
pred_1 = p.predict(test_1)
if pred_1 == 1:
 output = "L"
else:
 output = "M"
print("\nTEST INPUT:")
displayIP(test_1)
print("\nPREDICTED OUTPUT : ", output)
# TEST 2
test_2 = np.array([1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1])
pred_2 = p.predict(test_2)
if pred_2 == 1:
 output = "L"
else:
 output = "M"
print("\nTEST INPUT 2:")
displayIP(test_2)
print("\nPREDICTED OUTPUT : ", output)
```

OUTPUT:

```
Perceptron Classification Accuracy- TEST SET :  1.0

TEST INPUT 1:
1 0 1 0 0
1 0 0 0 0
1 0 0 0 0
1 1 1 1 1
PREDICTED OUTPUT :  L

TEST INPUT 2:
1 0 0 0 1
1 1 0 1 1
1 0 1 0 1
1 0 1 0 1
PREDICTED OUTPUT :  M
```

CONCLUSION: In this experiment, I designed a pattern classifier using perceptron training algorithm for the alphabets L and M. I made a class Perceptron with basic parameters like learning_rate(default=0.01), n_iters(default=1000), activation function as unit step function and the initial weights and bias to none. The accuracy of the test input for the perceptron classifier was 1.0, overfitting as the dataset is quite small. The current perceptron model creates a decision boundary for binary classification i.e. output is labelled ‘L’ if the model output is 1 else ‘M’. Finally, when I fed the model with distorted L and M inputs it successfully classified between the alphabets.

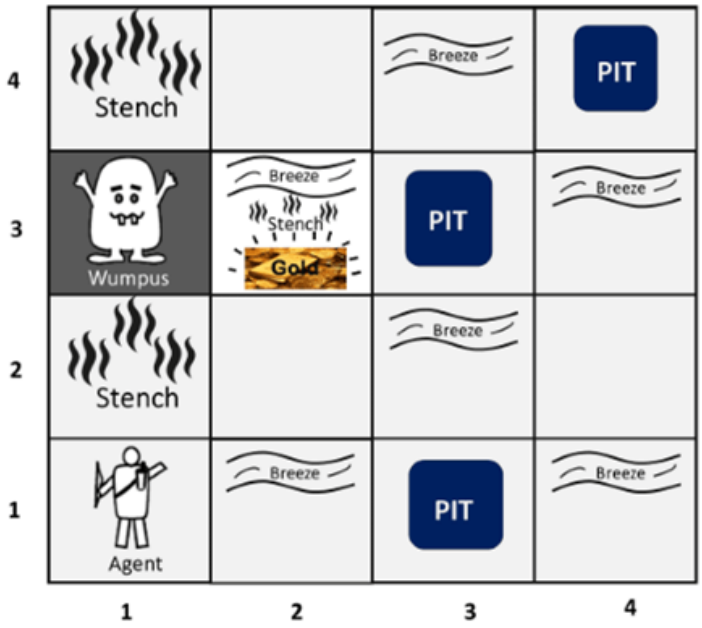
NAME: Meith Navlakha
SAP: 60004190068
AI Practical

EXPERIMENT 8

AIM: Implementation on AI Problem: Wumpus World

THEORY:

The Wumpus world is a cave which has 4/4 rooms connected with passageways. So there are total 16 rooms which are connected with each other. We have a knowledge-based agent who will go forward in this world. The cave has a room with a beast which is called Wumpus, who eats anyone who enters the room. The Wumpus can be shot by the agent, but the agent has a single arrow. In the Wumpus world, there are some Pits rooms which are bottomless, and if agent falls in Pits, then he will be stuck there forever. The exciting thing with this cave is that in one room there is a possibility of finding a heap of gold. So the agent goal is to find the gold and climb out the cave without falling into the Pits or eaten by Wumpus. The agent will get a reward if he comes out with gold, and he will get a penalty if eaten by Wumpus or falls in the pit.



The components of Wumpus World are:

- The rooms adjacent to the Wumpus room are smelly, so that it would have some stench.
- The room adjacent to PITs has a breeze, so if the agent reaches near to PIT, then he will perceive the breeze.
- There will be glitter in the room if and only if the room has gold.
- The Wumpus can be killed by the agent if the agent is facing to it, and Wumpus will emit a horrible scream which can be heard anywhere in the cave.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

PEAS description of Wumpus world:

Performance measure:

- +1000 reward points if the agent comes out of the cave with the gold.
- -1000 points penalty for being eaten by the Wumpus or falling into the pit.
- -1 for each action, and -10 for using an arrow.
- The game ends if either agent dies or came out of the cave.

Environment:

- A 4*4 grid of rooms.
- The agent initially in room square [1, 1], facing toward the right.
- Location of Wumpus and gold are chosen randomly except the first square [1,1].
- Each square of the cave can be a pit with probability 0.2 except the first square.

Actuators:

Left turn, Right turn, Move forward, Grab, Release, Shoot.

Sensors:

- The agent will perceive the stench if he is in the room adjacent to the Wumpus. (Not diagonally).
- The agent will perceive breeze if he is in the room directly adjacent to the Pit.
- The agent will perceive the glitter in the room where the gold is present.
- The agent will perceive the bump if he walks into a wall.
- When the Wumpus is shot, it emits a horrible scream which can be perceived anywhere in the cave.
- These percepts can be represented as five element list, in which we will have different indicators for each sensor.

Example if agent perceives stench, breeze, but no glitter, no bump, and no scream then it can be represented as:
[Stench, Breeze, None, None, None].

The Wumpus World Properties:

- **Partially observable:** The Wumpus world is partially observable because the agent can only perceive the close environment such as an adjacent room.
- **Deterministic:** It is deterministic, as the result and outcome of the world are already known.
- **Sequential:** The order is important, so it is sequential.
- **Static:** It is static as Wumpus and Pits are not moving.
- **Discrete:** The environment is discrete.
- **One agent:** The environment is a single agent as we have one agent only and Wumpus is not considered as an agent

CODE:

```
import java.util.Scanner;

class Block {
   public static final int NOT_PRESENT = 0;
   public static final int UNSURE = 1;

   public boolean hasBreeze, hasPit;
   public int pitStatus = UNSURE;

   public boolean hasStench, hasWumpus;
   public int wumpusStatus = UNSURE;

   public boolean hasGold;

   public boolean isVisited;
}
class WumpusWorld {
   static Block[][] maze;
   static int n;

   public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    System.out.print("Enter the order of the maze: ");
    n = sc.nextInt();

    maze = new Block[n][n];
    for(int i=0; i<n; i++) {
       maze[i] = new Block[n];
```

```
    for(int j=0; j<n; j++)
        maze[i][j] = new Block();
}

System.out.print("\n Enter the number of pits: ");
int pits = sc.nextInt();

for(int i=0; i<pits; i++) {
    System.out.print("Enter the location of pit " + (i+1) + ": ");
    addPit(n-sc.nextInt(), sc.nextInt()-1);
}

System.out.print("\nEnter the location of wumpus: ");
addWumpus(n-sc.nextInt(), sc.nextInt()-1);

System.out.print("\nEnter the location of gold: ");
addGold(n-sc.nextInt(), sc.nextInt()-1);

System.out.print("\nEnter the starting location: ");
int r = n - sc.nextInt();
int c = sc.nextInt() - 1;
int rPrev = -1, cPrev = -1;

int moves = 0;
System.out.println("\nInitial state:");
printMaze(r, c);

while(!maze[r][c].hasGold) {
    maze[r][c].isVisited = true;
    maze[r][c].pitStatus = Block.NOT_PRESENT;
    maze[r][c].wumpusStatus = Block.NOT_PRESENT;

    if(!maze[r][c].hasBreeze) {
        if(r >= 1 && maze[r-1][c].pitStatus == Block.UNSURE)
            maze[r-1][c].pitStatus = Block.NOT_PRESENT;
        if(r <= (n-2) && maze[r+1][c].pitStatus == Block.UNSURE)
            maze[r+1][c].pitStatus = Block.NOT_PRESENT;
        if(c >= 1 && maze[r][c-1].pitStatus == Block.UNSURE)
            maze[r][c-1].pitStatus = Block.NOT_PRESENT;
        if(c <= (n-2) && maze[r][c+1].pitStatus == Block.UNSURE)
            maze[r][c+1].pitStatus = Block.NOT_PRESENT;
    }

    if(!maze[r][c].hasStench) {
        if(r >= 1 && maze[r-1][c].wumpusStatus == Block.UNSURE)
            maze[r-1][c].wumpusStatus = Block.NOT_PRESENT;
        if(r <= (n-2) && maze[r+1][c].wumpusStatus == Block.UNSURE)
            maze[r+1][c].wumpusStatus = Block.NOT_PRESENT;
        if(c >= 1 && maze[r][c-1].wumpusStatus == Block.UNSURE)
            maze[r][c-1].wumpusStatus = Block.NOT_PRESENT;
```

```
            if(c <= (n-2) && maze[r][c+1].wumpusStatus == Block.UNSURE)
                maze[r][c+1].wumpusStatus = Block.NOT_PRESENT;
        }

        boolean foundNewPath = false;

        if(r >= 1 && !((r-1) == rPrev && c == cPrev) && maze[r-1][c].isVisited == false &&
maze[r-1][c].pitStatus == Block.NOT_PRESENT && maze[r-
1][c].wumpusStatus == Block.NOT_PRESENT) {
            rPrev = r;
            cPrev = c;

            r--;
            foundNewPath = true;
        }
        else if(r <= (n-2) && !((r+1) == rPrev && c == cPrev) && maze[r+1][c].isVisited ==
false && maze[r+1][c].pitStatus == Block.NOT_PRESENT &&
maze[r+1][c].wumpusStatus == Block.NOT_PRESENT) {
            rPrev = r;
            cPrev = c;

            r++;
            foundNewPath = true;
        }
        else if(c >= 1 && !(r == rPrev && (c-1) == cPrev) && maze[r][c-1].isVisited == false
&& maze[r][c-1].pitStatus == Block.NOT_PRESENT && maze[r][c-
1].wumpusStatus == Block.NOT_PRESENT) {
            rPrev = r;
            cPrev = c;

            c--;
            foundNewPath = true;
        }
        else if(c <= (n-2) && !(r == rPrev && (c+1) == cPrev) && maze[r][c+1].isVisited ==
false && maze[r][c+1].pitStatus == Block.NOT_PRESENT &&
maze[r][c+1].wumpusStatus == Block.NOT_PRESENT) {
            rPrev = r;
            cPrev = c;

            c++;
            foundNewPath = true;
        }

        if(!foundNewPath) {
            int temp1 = rPrev;
            int temp2 = cPrev;

            rPrev = r;
            cPrev = c;
```

```
            r = temp1;
            c = temp2;
        }

        moves++;

        System.out.println("\n\nMove " + moves + ":");
        printMaze(r, c);

        if(moves > n*n) {
            System.out.println("\nNo solution found!");
            break;
        }
    }

    if(moves <= n*n)
        System.out.println("\nFound gold in " + moves + " moves.");

    sc.close();
}

static void addPit(int r, int c) {
    maze[r][c].hasPit = true;

    if(r >= 1)
        maze[r-1][c].hasBreeze = true;
    if(r <= (n-2))
        maze[r+1][c].hasBreeze = true;
    if(c >= 1)
        maze[r][c-1].hasBreeze = true;
    if(c <= (n-2))
        maze[r][c+1].hasBreeze = true;
}

static void addWumpus(int r, int c) {
    maze[r][c].hasWumpus = true;

    if(r >= 1)
        maze[r-1][c].hasStench = true;
    if(r <= (n-2))
        maze[r+1][c].hasStench = true;
    if(c >= 1)
        maze[r][c-1].hasStench = true;
    if(c <= (n-2))
        maze[r][c+1].hasStench = true;
}

static void addGold(int r, int c) {
    maze[r][c].hasGold = true;
}
```

```
    static void printMaze(int r, int c) {
     for(int i=0; i<n; i++) {
         for(int j=0; j<n; j++) {
             char charToPrint = '-';
             if(r == i && c == j)
                 charToPrint = '*';
             else if(maze[i][j].hasPit)
                 charToPrint = 'O';
             else if(maze[i][j].hasWumpus)
                 charToPrint = 'X';
             else if(maze[i][j].hasGold)
                 charToPrint = '$';
            else if(maze[i][j].hasBreeze)
                 charToPrint = 'B';
            else if(maze[i][j].hasStench)
                 charToPrint = 'S';

             System.out.print(charToPrint + "\t");
         }
         System.out.println();
     }
    }
}
```

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

OUTPUT:

```
C:\Windows\System32\cmd.exe
C:\Users\meith\OneDrive\Desktop\SEM 5\AI>javac WumpusWorld.java

C:\Users\meith\OneDrive\Desktop\SEM 5\AI>java WumpusWorld
Enter the order of the maze: 4

Enter the number of pits: 3
Enter the location of pit 1: 1 3
Enter the location of pit 2: 3 3
Enter the location of pit 3: 4 4

Enter the location of wumpus: 3 1

Enter the location of gold: 3 2

Enter the starting location: 1 1

Initial state:
S       -       B       O
X       $       O       B
S       -       B       -
*       B       O       B


Move 1:
S       -       B       O
X       $       O       B
*       -       B       -
-       B       O       B


Move 2:
S       -       B       O
X       $       O       B
S       -       B       -
*       B       O       B


Move 3:
S       -       B       O
X       $       O       B
S       -       B       -
-       *       O       B
```

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

```
C:\Windows\System32\cmd.exe
X       $       O       B
S       -       B       -
*       B       O       B


Move 1:
S       -       B       O
X       $       O       B
*       -       B       -
-       B       O       B


Move 2:
S       -       B       O
X       $       O       B
S       -       B       -
*       B       O       B


Move 3:
S       -       B       O
X       $       O       B
S       -       B       -
-       *       O       B


Move 4:
S       -       B       O
X       $       O       B
S       *       B       -
-       B       O       B


Move 5:
S       -       B       O
X       *       O       B
S       -       B       -
-       B       O       B

Found gold in 5 moves.

C:\Users\meith\OneDrive\Desktop\SEM 5\AI>
```

CONCLUSION: In this Experiment, I implemented the Wumpus World program which is a knowledge-based agent. Here, the agent can perform 4 movements, forward, backward, left or right. The user can sense the stench or breeze and make appropriate actions accordingly using the knowledge base. Also, the agent is awarded or penalised accordingly based on the actions. The game stops once the user finds the gold. After each attempt it adds information, it learnt from the previous attempt into the knowledge base thereby continually updating the knowledge base.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

EXPERIMENT 9

AIM: Program to implement Family Tree in Prolog.

THEORY:

Prolog is a logic programming language. It has important role in artificial intelligence. Unlike many other programming languages, Prolog is intended primarily as a declarative programming language. In prolog, logic is expressed as relations (called as Facts and Rules). Core heart of prolog lies at the logic being applied. Formulation or Computation is carried out by running a query over these relations.

Syntax and Basic Fields:

In prolog, we declare some facts. These facts constitute the Knowledge Base of the system. We can query against the Knowledge Base. We get output as affirmative if our query is already in the knowledge Base or it is implied by Knowledge Base, otherwise we get output as negative. So, Knowledge Base can be considered similar to database, against which we can query. Prolog facts are expressed in definite pattern. Facts contain entities and their relation. Entities are written within the parenthesis separated by comma (,). Their relation is expressed at the start and outside the parenthesis. Every fact/rule ends with a dot (.). So, a typical prolog fact goes as follows:

Format: relation(entity1, entity2, ...k'th entity).

Example:

friends(Suresh, Mahesh).

singer(Micca).

odd_number(7).

Explanation:

These facts can be interpreted as:

Suresh and Mahesh are friends.

Micca is a singer.

7 is an odd number.

Key Features:

1. Unification: The basic idea is, can the given terms be made to represent the same structure.
2. Backtracking: When a task fails, prolog traces backwards and tries to satisfy previous task.
3. Recursion: Recursion is the basis for any search in program.

Running queries:
A typical prolog query can be asked as:

- Query 1: ?- singer(sonu).
  Output: Yes.
  Explanation: As our knowledge base contains the above fact, so output was 'Yes', otherwise it would have been 'No'.

- Query 2: ?- odd_number(7).
  Output: No.
  Explanation: As our knowledge base does not contain the above fact, so output was 'No'.

Advantages:

1. Easy to build database. Doesn't need a lot of programming effort.
2. Pattern matching is easy. Search is recursion based.
3. It has built in list handling. Makes it easier to play with any algorithm involving lists.

Disadvantages:

1. LISP (another logic programming language) dominates over prolog with respect to I/O features.
2. Sometimes input and output is not easy.

Applications:

Prolog is highly used in artificial intelligence (AI). Prolog is also used for pattern matching over natural language parse trees.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

CODE:

```
female(manisha).
female(sneha).
female(trushi).
female(neeta).
male(meith).
male(jatan).
male(hitesh).
male(bhavesh).
male(dhishil).
male(prakash).
parent(jatan,hitesh).
parent(jatan,bhavesh).
parent(prakash,manisha).
parent(hitesh,meith).
parent(prakash,sneha).
parent(manisha,meith).
parent(sneha,trushi).
parent(sneha,dhishil).
parent(bhavesh,trushi).
parent(bhavesh,dhishil).
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,Y),female(X),X\==Y.
brother(X,Y):-parent(Z,X),parent(Z,Y),male(X),X\==Y.
grandparent(X,Y):-parent(X,Z),parent(Z,Y).
grandmother(X,Z):-mother(X,Y),parent(Y,Z).
grandfather(X,Z):-father(X,Y),parent(Y,Z).
wife(X,Y):-parent(X,Z),parent(Y,Z),female(X),male(Y).
uncle(X,Z):-brother(X,Y),parent(Y,Z).
```

OUTPUT:

```
SWI-Prolog (AMD64, Multi-threaded, version 8.4.1)
File  Edit  Settings  Run  Debug  Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.1)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/meith/OneDrive/Desktop/SEM 5/AI/prolog.pl compiled 0.00 sec, 30 clauses
?- haschild(meith).
false.

?- haschild(hitesh).
true.

?- sister(manisha,sneha)
|    .
true.

?- father(bhavesh,X).
X = trushi .

?- grandfather(jatan,X).
X = meith ;
X = trushi ;
X = dhishil.

?- mother(sneha,X).
X = trushi ;
X = dhishil.

?- wife(manisha,hitesh).
true .

?- uncle(bhavesh,X).
X = meith ;
```

CONCLUSION:

In this experiment, I implemented prolog family tree code on my family tree. Prolog can generate complex relations based on a few basic relations, having its roots in first-order logic. In this experiment, the only information I provided was the gender of the individual and basic relation i.e., who is a parent of whom. Prolog then used this information and allowed us to write code and on complex relations. For example, in the experiment, the father function tells us if the person is the father of other mentioned individual or can also be used to list all the children of the specified person. Similarly for other complex relations like wife or uncle. Thus, prolog allows us to draw complex inferences from simpler relations or logic.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

EXPERIMENT 10

AIM: Identify, analyze, implement a planning problem/Rule based Expert System in a real world scenario.

THEORY:

Partial-order planning is an approach to automated planning that maintains a partial ordering between actions and only commits ordering between actions when forced to that is, ordering of actions is partial. Also, this planning doesn't specify which action will come out first when two actions are processed. By contrast, total-order planning maintains a total ordering between all actions at every stage of planning. Given a problem in which some sequence of actions is required in order to achieve a goal, a partial-order plan specifies all actions that need to be taken, but specifies an ordering between actions only where necessary.

A partial-order plan or partial plan is a plan which specifies all actions that need to be taken, but only specifies the order between actions when necessary. It is the result of a partial-order planner. A partial-order plan consists of four components:

- A set of actions (also known as operators).
- A partial order for the actions. It specifies the conditions about the order of some actions.
- A set of causal links. It specifies which actions meet which preconditions of other actions. Alternatively, a set of bindings between the variables in actions.
- A set of open preconditions. It specifies which preconditions are not fulfilled by any action in the partial-order plan.

In order to keep the possible orders of the actions as open as possible, the set of order conditions and causal links must be as small as possible.

A plan is a solution if the set of open preconditions is empty.

A linearization of a partial order plan is a total order plan derived from the particular partial order plan; in other words, both order plans consist of the same actions, with the order in the linearization being a linear extension of the partial order in the original partial order plan.

NAME: Meith Navlakha
SAP: 60004190068
AI Practical

PERFORMANCE:

Chosen Problem: Vacuum Cleaner

Problem Description:

Goal (Room1Clean ^ Room2Clean)

Init ()

Action: Room1Clean

- PRECOND: ScanRoom1
- EFFECT: Room1Cleaned

Action: ScanRoom1

- PRECOND: None
- EFFECT: Room1Scanned

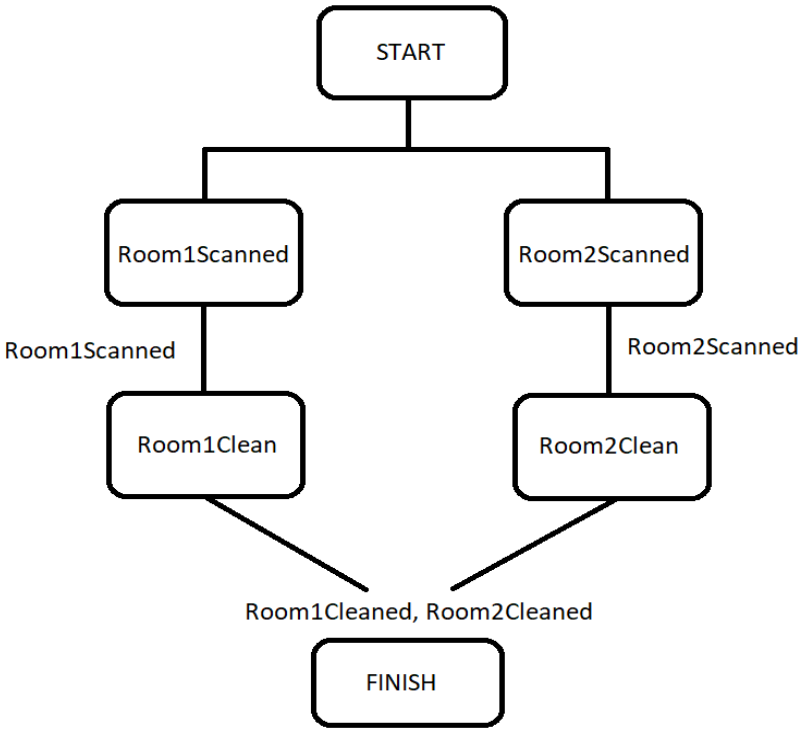
Action: Room2Clean

- PRECOND: ScanRoom2
- EFFECT: Room2Cleaned

Action: ScanRoom2

- PRECOND: None
- EFFECT: Room2Scanned

Partial Order Plan:



Components in Final Plan:

Actions: {ScanRoom, SuckGarbage, Left, Right, Start, Finish}

Orderings: {ScanRoom < SuckGarbage}

Open preconditions: {}

Links:

Room1 --- Room1Dirty ---> SuckGarbage

Room1 --- Room1Clean ---> Finish

Room2 --- Room2Dirty ---> SuckGarbage

Room2 --- Room2Clean ---> Finish

CONCLUSION:

In this experiment, I have successfully implemented Partial Order Planning with an example of an automatic Vacuum Cleaner. The tasks of the cleaner are to check whether the room is dirty or clean, and clean if it is dirty. The task is completed when both rooms are clean. One drawback of this planning system is that it requires a lot more computational power for each node because the algorithm for partial order planning is more complex than others. Partial order planning has huge importance and usage in artificial intelligence applications.