**GODAVARI FOUNDATION’S**

**GODAVARI COLLEGE OF ENGINEERING, JALGAON**

**DEPARTMENT OF COMPUTER ENGINEERING**

**L.Y. SEM-VII**

**2023-24**

# Lab Manual

**ARTIFICIAL INTELLIGENCE LAB**

**B.Tech Computer**

**GODAVARI FOUNDATION’S**

GODAVARI **COLLEGE OF ENGINEERING,** **JALGAON**

**DEPARTMENT OF COMPUTER ENGINEERING**

## L.Y.2023-24 SEM–VII

### Sub: Artificial Intelligence Lab Class : Final Year B.Tech

**List of Practicals**

|  |  |
| --- | --- |
| **1.** | **Study of PROLOG. Write the following programs using PROLOG** |
| **2.** | **Writeaprogramtosolve8queens problem** |
| **3.** | **Solve any problem using depth first search.** |
| **4.** | **Solve any problem using best first search.** |
| **5.** | **Solve8-puzzleproblemusingbestfirstsearch** |
| **6.** | **Solve traveling salesman problem** |

## Godavari College of Engineering ,Jalgaon

## Department of Computer Engineering

**Roll No:-------- Date of Exp:------------------------**

### Class:------------

**================================================================PRACTICAL NO. 1**

**Aim :** Study of PROLOG

**Theory:**

PROLOG stands for Programming, In Logic an idea that emerged in the early 1970'stouselogicas programming language. To early developers of this idea included Robert Kowaiski at Edinburgh (on the theoretical side), Manten van Emdenat Edinburgh (experimental demonstration) and Alian Colmerauerat Marseilles(implementation).

David D. H. Warren's efficient implementation at Edinburgh inthemid-1970'sgreatly contributed to the popularity of PROLOG. PROLOG is a programming language centred around a small set of basic mechanisms, Including pattern matching, tree based data structuring and automatic backtracking. This Small set constitutes a surprisingly powerful and flexible programming frame work. PROLOG is especially well suited for problems that involve objects- in particular, structured objects- and relations between them.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

**Syntax rules:**

1. The names of all relationships and objects must begin with a lower case letter. For example: likes, john, rachel.
2. The relationship is written first, and the objects are written separated by commas, and the objects are enclosed by a pair of round brackets.
3. The character ‘.’ must come at the end of each fact.

**For Example:**

1. Ramlikes mango.
2. Seemaisa girl.
3. Billlikes Cindy.
4. Roseis red.
5. John owns gold.

**Program:**

Clauses likes(ram,mango). girl(seema). red(rose). likes(bill,cindy).

owns(john,gold).

**Output:**

quenes

?-likes(ram,What). What=mango

?-likes(Who,cindy). Who=cindy

?-red(What). What=rose ?- owns(Who,What). Who=john What=gold.

**Conclusion:**

The conclusion o f a prolog practical in AI may vary depending on the specific topic and task , but generally it should summarize the main objectives, methods, results, and implications o f t he prolog program. It should also high

light the strengths and limit at ions of prolog as a tool for AI, and suggest possible

improvements or extensions for future work.

## Godavari College of Engineering ,Jalgaon

## Department of Computer Engineering

**Roll No:-------- Date of Exp:------------------------**

### Class:------------

### PRACTICALNO.2

**Aim:** Write a program to solve 8queens problem.

**Procedure:**

STEP1: Represent the board positions as 8\*8 vector, i.e., [1, 2, 3, 4, 5, 6, 7, 8].Store the set of queens in the list ‘Q’.

STEP2:Calculate the permutation of the above eight numbers stored in set P.

STEP3:Let the position where the first queen to be placed be(1,Y), for second be(2,Y1) and so on and store the positions in Q.

STEP4 :Check for the safety of the queens through the predicate ,‘no attack()’.

STEP5:CalculateY1-YandY-Y1. If both are not equal toXdist, which is the X–distance between the first queen and others, then go to Step 6 else go to Step 7.

STEP6:Increment Xdistby1.

STEP7:Repeat above for the rest of the queens, until the end of the list is reached.

STEP8: Print Q as answer .

STEP9:Exit

.

**Program:**

domains H=integer T=integer\* predicatessafe(T) solution(T) permutation(T,T) del(H,T,T) noattack(H,T,H) clauses

del(I,[I|L],L)./\*totakeapositionfromthepermutationoflist\*/ del(I,[F|L],[F|L1]):- del(I,L,L1).

permutation([],[])./\*tofindthepossiblepositions\*/ permutation([H|T],PL):- permutation(T,PT),\ del(H,PL,PT). solution(Q):-/\*finalsolutionisstoredinQ\*/ permutation([1,2,3,4,5,6,7,8],Q), safe(Q). safe([])./\*Qissafesuchthatnoqueensattackeachother\*/ safe([Q|others]):- safe(others), noattack(Q,others,1). noattack(\_,[],\_)./\*tofindifthequeensareinsamerow,columnordiagonal\*/ noattack(Y,[Y1|Ydist],Xdist):- Y1-Y<>Xdist, Y-Y1<>Xdist, dist1=Xdist, noattack(Y,Ydist,dist1).

**Output :**

goal:-solution(Q).

Q=[“3”,”8”,”4”,”7”,”1”,”6”,2”,”5”]

**Conclusion:**

In conclusion, solving the 8 Queens problem using artificial intelligence is an interesting and challenging task that can be accomplished using genetic algorithms. Genetic algorithms can generate smart and efficient solutions without the need for manual intervention

## Godavari College of Engineering ,Jalgaon

## Department of Computer Engineering

**Roll No:-------- Date of Exp:------------------------**

### Class:------------

### PRACTICALNO.3

**Aim:** Solve any problem using depth first search.

**Theory:**

In the water jug problem in Artificial Intelligence, we are provided with two jugs :one having the capacity to hold 3gallons of water and the other has the capacity to hold 4 gallons of water. There is no other measuring equipment available and the jugs also do not have any kind of marking on them. So, the agent's task here is to fill the 4-gallon jug with 2 gallons of water by using only these two jug sand no other material .Initially, both our jugs are empty .So, to solve this problem, following set of mies were proposed: Production rules for solving the water jug problem.

**Program:**

solve\_dfs(State,History,[]):-

final\_state(State).solve\_dfs(State,History,[Move[ Moves]):•move(State, Move),

update(State, Move,State)legal(Statel), not(member(Statel,History)), solve\_dfs(Statel,[StateI/History],Moves).

test\_dfs(Problem,Moves):• initial\_state(Problem,State),solve\_dfs(State,[State],Moves).cap acity(1, 10). capacity(2, 7).

initial\_state(jugs,jugs(0,0)).final\_state(jugs(6,0)).

%final\_state(jugs(4, 0)).

legal(jugs(Vl,V2)).

move(jugs(Vl,V2),fill()):-capacity(1, C1),VI <Cl,capacity(2, C2),V2<C2. move(jugs(VI,V2),fill(2)):-capacity(2, C2),V2<C2,capacity(1, C1),VI <Cl. move(jugs(Vl,V2),empty(l)):-Vl>0. move(jugs(Vl,V2),empty(2)):-V2>0. move(jugs(VI,V2),transfer(1, 2)). move(jugs(Vl,V2),transfer(2,

1)).adjust(Liquid,Excess,Liquid,0):-

Excess=<0.adjust(Liquid,Excess,V,Excess) :-Excess>0,VisLiquidExcess.

update(jugs(VI,V2),fill(), jugs(Cl,V2)):-capacity(1, C1). update(jugs(VI, V2),fill(2),jugs(Vl,C2)):-

capacity(2,C2).update(jugs(Vl,V2),empty(l),jugs(0,V2)).update(jugs(Vl,V2),empty(2),jugs(Vl,0

)).

update(jugs(VI,V2),transfer(1,2).jugs(NewVI,NewV2)):• capacity(2,C2), LiquidisVl+V2,ExcessisLiquid-C2, adjust(Liquid,Excess,NewV2,NewVI).update(jugs(VI,V2),transfer2,D).jugs(NewVI

,NewV2)):• capacity(1, C1), Liquidis Vl+V2, ExcessisLiquid-C1,

adjust(Liquid,Excess,NewVI,NewV2).

**Output:**

Query:test\_dfs(jugs,Moves).Moves=[fill(l), transfer(l,2), empty(l), transfer(2,1), fill(2), transfer(2,1), empty(l), transfer(2,1), fill(2), transfer(2,1), empty(l), transfer(2,1), fill(2), transfer(2,1), fill(2), transfer(2,1),

empty(l), transfer(2,1),

fill(2), transfer(2,D.empt y(l), transfer(2,1),

fill(2), transfer(2,1), fill(2), transfer(2,1), empty(l),

transfer(2,1)]

**Conclusion:**

The conclusion of a Depth First Search in Artificial Intelligence is that **this** searching algorithm provides an efficient way for computers to traverse through graphs and trees. By using the stack data structure, DFS can access any branch of a graph or tree with one simple procedure and structure their output accordingly.

## Godavari College of Engineering ,Jalgaon

## Department of Computer Engineering

**Roll No:-------- Date of Exp:------------------------**

### Class:------------

### PRACTICALNO.4

**Aim:** Solve any problem using best first search.

**Theory:**

The game Tic Tac Toe is also known as Noughts and Crosses or X sand Os, the player needs to take turns marking the spaces in a 3x3 grid with their own marks, if 3 consecutive marks(Horizontal , Vertical ,Diagonal)are formed then the player who owns these moves get won.

**Program:**

win(Board,Player):-rowwin(Board,Player). win(Board,Player):-colwin(Board,Player). win(Board,Player):-diagwin(Board,Player).

rowwin(Board,Player):-Board=[Player,Player,Player,\_,\_,\_,\_,\_,\_J. rowwin(Board,Player):-Board=[\_,\_,\_,Player,Player,Player,\_,\_,\_].

rowwin(Board,Player):-Board=[\_,\_,\_,\_,\_,\_,Player,Player,Player].

colwin(Board, Player):-Board=[Player,\_,\_,Player,\_,\_,Player,\_,\_J. colwin(Board,Player) :- Board= [\_,Player,\_,\_,Player,\_,\_,Player,\_]. colwin(Board,Player):-Board= [\_.\_.Player,\_.\_,Player,\_.\_,Player].

diagwin(Board,Player):-Board=[Player,\_,\_,\_,Player,\_,\_,\_,Player].

diagwin(Board,Player):-Board=[\_,\_,Player,\_,Player,\_,Player,\_,\_J.

%Helpingpredicateforalternatingplayina"self'game:

other(x,o).

other(o,x).

game(Board,Player):-win(Board,Player),!,write([player,Player,wins]).

game(Board,Player):• other(Player,Otherplayer), move(Board,Player,Newboard),

I.,display(Newboard),game(Newboard,Otherplayer).

move([b,B,C,D,E,F,G,H,I],Player,[Player,B,C,D,E,F,G,H,I]). move([A,b,C,D,E,F,G,H,I],Player,[A,Player,C,D,E,F,G,H,I]). move([A,B,b,D,E,F,G,H,I],Player,[A,B,Player,D,E,F,G,H,I]). move([A,B,C,b,E,F,G,H,I],Player,[A,B,C,Player,E,F,G,H,I]). move([A,B,C,D,b,F,G,H,I],Player,[A,B,C,D,Player,F,G,H,I]). move([A,B,C,D,E,b,G,H,I],Player,[A,B,C,D,E,Player,G,H,I]). move([A,B,C,D,E,F,b,H,I],Player,[A,B,C,D,E,F,Player,H,I]). move([A,B,C,D,E,F,G,b,I],Player,[A,B,C,D,E,F,G,Player,I]). move([A,B,C,D,E,F,G,H,b],Player,[A,B,C,D,E,F,G,H,Player]).

display([A,B,C,D,E,F,G,H,I]):-

write([A,B,C]),nl,write([D,E,F]),nLwrite([G,HI)),nl,nl

selfgame:-game([b,b,b,b,b,b,b,b,b],x).

%Predicatestosupportplayingagamewiththe user: x\_can\_win\_in\_one(Board):-move(Board,x,Newboard),win(Newboard,x).

%Thepredicateorespondgeneratesthecomputer's(playingo)reponse %fromthecurrentBoard.

orespond(Board,Newboard):• move(Board,o,Newboard),

win(Newboard,o),

!.

orespond(Board,Newboard):• move(Board, o,Newboard), not(x\_can\_win\_in\_one(Newboard)). orespond(Board,Newboard):•

move(Board,o,Newboard). orespond(Board,Newboard):• not(member(b,Board)),

I., write('Catsgame!'),nl,Newboard=Board.

%Thefollowingtranslatesfromanintegerdescription %ofx'smovetoaboardtransformation.

xmove([b,B,C,D,E,F,G,H,I],1, [x,B,C,D,E,F,G,H,I]). xmove([A,b,C,D,E,F,G,H,I],2,[A,x,C,D,E,F,G,H,I]). xmove([A,B,b,D,E,F,G,H,I],3,[A,B,x,D,E,F,G,H,I]). xmove([A,B,C,b,E,F,G,H,I],4,[A,B,C,x,E,F,G,H,I]). xmove([A,B,C,D,b,F,G,H,I],5,[A,B,C,D,x,F,G,H,I]). xmove([A,B,C,D,E,b,G,H,I],6,[A,B,C,D,E,x,G,H,I]). xmove([A,B,C,D,E,F,b,H,I],7,[A,B,C,D,E,F,x,H,I]). xmove([A,B,C,D,E,F,G,b,I],8,[A,B,C,D,E,F,G,x,I]). xmove([A,B,C,D,E,F,G,H,b],9,[A,B,C,D,E,F,G,H,x]).

xmove(Board,\_,Board):-write('Illegalmove.'),nl.

%The0-

placepredicateplayostartsagamewiththeuser.playo:explain,play:from([b,b,b,b,b,b,b,b,b]).

explain:-

write('YouplayXbyenteringintegerpositions followedbyaperiod.'), nl, display([l,2,3,4,5,6,7,8,9]).

play:from(Board):-win(Board,x),write('Youwin!'). playfrom(Board):-win(Board,0),write('Iwin!').

playfrom(Board):-read(N),

xmove(Board,N,Newboard), display(Newboard), orespond(Newboard,Newnewboard),

**Conclusion:**

Best first search is a heuristic search algorithm that uses an evaluation function to decide which node is the most promising to explore next . It tries to find the shortest path from

the initial state to the goal state by expanding the nodes with the lowest cost.

## Godavari College of Engineering, Jalgaon

## Department of Computer Engineering

**Roll No:-------- Date of Exp:------------------------**

### Class:------------

### PRACTICALNO.5

**Aim:** Solve 8-puzzle problem using best first search

**Theory :**

The title of this section refers to a familiar and popular sliding tile puzzle that has been around for at least forty years. The most frequent older versions of this puzzle have numbers or letters an the sliding tiles, and the player is supposed to slide tiles into newpositions in order to realign a scrambled puzzle back into a goal alignment.

**Production Rules:-**

h\_function(Puzz,H)p\_fcn(Puzz,P),s\_fcn(Puzz,S),HisP+3\*S.

The'move'productionsared efinedasfollows. move(P,C,left) left(P,C).

move(P,C,up) up(P,C).

move(P,C,right) right(P,C).

move(P,C,down )down(P,C). p\_fcn(A/B/C/D/E/F/G/H/I,P)a(A,Pa),b(B,Pb),c(C,Pc),

d(D,Pd),e(E,Pe),f(F,Pf),

g(G,Pg),h(H,Ph),i(I,Pi),

PisPa+Pb+Pc+Pd+Pe+Pf+Pg+Ph+Pg+Pi.

s\_fcn(A/B/C/D/E/F/G/H/I,S)1s\_aux(A,B,S1),s\_aux(B,C,S2), s\_aux(C,F,S3),s\_aux(F,I,S4), s\_aux(I,H,S5),s\_aux(H,G,S), s\_aux(G,D,S7),s\_aux(D,A,8), s\_aux(E,S9)

Sis S1+S2+S3+S4+S5+S6+S7+S8+S9.

Theheuristicfunctionweusehereisacombinationoftwootherestimators:p\_fcn,the

Manhattandistancefunction,ands\_fcn,thesequencefunction,allasexplainedinNilsson (1980), which estimates how badly out- of-sequence the tiles are (around the outside).

h\_function(Puzz,H):- p\_fcn(Puzz,P), s\_fcn(Puzz,S), H is P + 3\*S.

The'move'predicateisdefinedasfollows.move(P,C,left):-left(P,C). move(P,C,up):-up(P,C).

move(P,C,right):-right(P,C). move(P,C,down):-down(P,C).Hereisthecodeforpands.

%%%Manhattandistancep\_fcn(A/B/C/D/E/F/G/H/I,P):- a(A,Pa),b(B,Pb),c(C,Pc), d(D,Pd),e(E,Pe),f(F,Pf), g(G,Pg),h(H,Ph),i(I,Pi),

PisPa+Pb+Pc+Pd+Pe+Pf+Pg+Ph+Pg+Pi.

a(0,0).a(1,0).a(2,1).a(3,2).a(4,3).a(5,4).a(6,3).a(7,2).a(8,1). b(0,0).b(1,0).b(2,0).b(3,1).b(4,2).b(5,3).b(6,2).b(7,3).b(8,2). c(0,0).c(1,2).c(2,1).c(3,0).c(4,1).c(5,2).c(6,3).c(7,4).c(8,3). d(0,0).d(1,1).d(2,2).d(3,3).d(4,2).d(5,3).d(6,2).d(7,2).d(8,0). e(0,0).e(1,2).e(2,1).e(3,2).e(4,1).e(5,2).e(6,1).e(7,2).e(8,1). f(0,0).f(1,3).f(2,2).f(3,1).f(4,0).f(5,1).f(6,2).f(7,3).f(8,2). g(0,0).g(1,2).g(2,3).g(3,4).g(4,3).g(5,2).g(6,2).g(7,0).g(8,1). h(0,0).h(1,3).h(2,3).h(3,3).h(4,2).h(5,1).h(6,0).h(7,1).h(8,2).

i(0,0).i(1,4).i(2,3).i(3,2).i(4,1).i(5,0).i(6,1).i(7,2). i(8,3).

%%%theout-of-cyclefunctions\_fcn(A/B/C/D/E/F/G/H/I,S):- s\_aux(A,B,S1),s\_aux(B,C,S2),s\_aux(C,F,S3), s\_aux(F,I,S4),s\_aux(I,H,S5),s\_aux(H,G,S6), s\_aux(G,D,S7),s\_aux(D,A,S8),s\_aux(E,S9),SisS1+S2+S3+S4+S5+S6+S7+S8+S9.

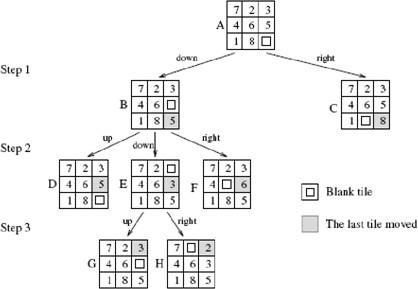
s\_aux(0,0):-!.s\_aux(\_,1).

s\_aux(X,Y,0):-Y is X+1, !.

s\_aux(8,1,0):-!.s\_aux(\_,\_,2).

ThePrologprogramfromtheprevioussectionandtheprogramoutlinedinthissectioncanbeusedasan 8- puzzle solver.

?-solve(0/8/1/2/4/3/7/6/5,S).



**Solution:**

left( A/0/C/D/E/F/H/I/J , 0/A/C/D/E/F/H/I/J ). left( A/B/C/D/0/F/H/I/J, A/B/C/0/D/F/H/I/J). left(A/B/C/D/E/F/H/0/J,A/B/C/D/E/F/0/H/J). left( A/B/0/D/E/F/H/I/J , A/0/B/D/E/F/H/I/J ). left(A/B/C/D/E/0/H/I/J,A/B/C/D/0/E/H/I/J). left( A/B/C/D/E/F/H/I/0,A/B/C/D/E/F/H/0/I).

up( A/B/C/0/E/F/H/I/J , 0/B/C/A/E/F/H/I/J ). up(A/B/C/D/0/F/H/I/J,A/0/C/D/B/F/H/I/J). up(A/B/C/D/E/0/H/I/J,A/B/0/D/E/C/H/I/J). up( A/B/C/D/E/F/0/I/J , A/B/C/0/E/F/D/I/J ). up(A/B/C/D/E/F/H/0/J,A/B/C/D/0/F/H/E/J).

up(A/B/C/D/E/F/H/I/0,A/B/C/D/E/0/H/I/F).

right(A/0/C/D/E/F/H/I/J,A/C/0/D/E/F/H/I/J). right(A/B/C/D/0/F/H/I/J,A/B/C/D/F/0/H/I/J). right(A/B/C/D/E/F/H/0/J,A/B/C/D/E/F/H/J/0). right( 0/B/C/D/E/F/H/I/J , B/0/C/D/E/F/H/I/J ). right( A/B/C/0/E/F/H/I/J , A/B/C/E/0/F/H/I/J ).

right( A/B/C/D/E/F/0/I/J , A/B/C/D/E/F/I/0/J ).

down(A/B/C/0/E/F/H/I/J,A/B/C/H/E/F/0/I/J). down(A/B/C/D/0/F/H/I/J,A/B/C/D/I/F/H/0/J). down(A/B/C/D/E/0/H/I/J,A/B/C/D/E/J/H/I/0). down(0/B/C/D/E/F/H/I/J,D/B/C/0/E/F/H/I/J). down(A/0/C/D/E/F/H/I/J,A/E/C/D/0/F/H/I/J). down(A/B/0/D/E/F/H/I/J,A/B/F/D/E/0/H/I/J).

**Conclusion:**

One possible conclusion of using best first search for the 8 puzzle problem is that it can find a solution faster and more efficiently than uninformed search algorithms, such as breadthfirst search or depth-first search, which do not use any domain knowledge

## Godavari College of Engineering, Jalgaon

## Department of Computer Engineering

**Roll No:-------- Date of Exp:------------------------**

### Class:-----------

### PRACTICALNO.6

**Aim:** Solve traveling salesman problem.

**Theory:**

The Traveling Salesman Problem (TSP) is a classic problem in the field of computer science and artificial intelligence. Prolog, being a logic programming language, is well-suited for solving such combinatorial problems. Here's an example of how you can solve the Traveling Salesman Problem using Prolog:

**Production Rules:-**

%Facts

dist(a,b,10). dist(a,c,15). dist(a,d,20). dist(b,c,35). dist(b,d,25).

dist(c,d,30).

%Predicatetofindtheshortestroute tsp(Path, Cost) :- findall([X,Y,D],dist(X,Y,D),Distances), findall(City, member([City, \_,\_],Distances), Cities), permutation(Cities, Path), calculate\_cost(Path,Distances,Cost).

%Predicatetocalculatethetotalcostoftheroute calculate\_cost([City1, City2 |Rest], Distances, Cost) :- member([City1,City2,D],Distances), calculate\_cost([City2 |Rest], Distances, PartialCost), Cost is PartialCost + D.

calculate\_cost([\_],\_,0).

%Findtheshortestrouteanditscost

?-tsp(Path, Cost),write('ShortestPath:'),write(Path),nl,write('Total Cost:'),write(Cost).

**Solution:**

Inthis Prologcode:

* The**dist/3**predicaterepresentsthedistancesbetweencities.
* The**tsp/2**predicatefindsallpermutationsofcities,calculatesthetotalcostforeach permutation, and backtracks to find the shortest path and its cost.
* The**calculate\_cost/3**predicaterecursivelycalculatesthetotalcostofaroute.

To run this Prolog code,you can use aProlog interpreter. Make sure to load the code into the interpreter, and then query **tsp(Path, Cost).** to find the shortest path and its cost.

**Conclusion:**

The travelling salesman problem (TSP) is a classic NP-hard problem in computer science and artificial intelligence (AI). It asks for the shortest possible route that visits every city in a given set exactly once and returns to the starting point. The TSP has many applications in logistics, planning, manufacturing, and genetics.