ARTIFICIAL INTELLIGENCE

MINOR PROJECT

“ Searching In A Maze ”

Project Report

submitted in partial fulfillment of the requirements for the award of the degree of MSC IT

(G.G.D.S.D COLLEGE , CHANDIGARH)

SESSION : 2023-2024

SUBMITTED TO:	**SUBMITTED BY:**
Dr. HIMANI MITTAL	SIMRANJEET KAUR
	ROLL NO. - 2341753
	CLASS : M.Sc IT
	SEMESTER - 2

INDEX

Acknowledgement

I would like to place on record my deep sense of gratitude to Dr. Virender Singh, HOD-Dept of IT, GGDSD Sector-32, Chandigarh, for his general guidance, help and suggestions.

I express my sincere gratitude to Dr. Himani Mittal, Dept of Information and Technology, GGDSD Sector- 32, Chandigarh, for her stimulating guidance, continuous encouragement and supervision throughout the course of present work.

Simranjeet Kaur

2341753

INTRODUCTION TO PROLOG :

Prolog stands for programming in logic. In the logic programming paradigm, prolog language is most widely available. Prolog is a declarative language, which means that a program consists of data based on the facts and rules (Logical relationship) rather than computing how to find a solution. A logical relationship describes the relationships which hold for the given application.

- To obtain the solution, the user asks a question rather than running a program. When a user asks a question, then to determine the answer, the run time system searches through the database of facts and rules.
- Prolog features are 'Logical variable', which means that they behave like uniform data structure, a backtracking strategy to search for proofs, a pattern-matching facility, mathematical variable, and input and out are interchangeable.
- To deduce the answer, there will be more than one way. In such case, the run time system will be asked to find another solution. To generate another solution, use the backtracking strategy. Prolog is a weakly typed language with static scope rules and dynamic type checking.
- Prolog is a declarative language that means we can specify what problem we want to solve rather than how to solve it.
- Prolog is used in some areas like database, natural language processing, artificial intelligence, but it is pretty useless in some areas like a numerical algorithm or instance graphics.

In artificial intelligence applications, prolog is used. The artificial intelligence applications can be automated reasoning systems, natural language interfaces, and expert systems. The expert system consists of an interface engine and a database of facts. The prolog's run time system provides the service of an interface engine.

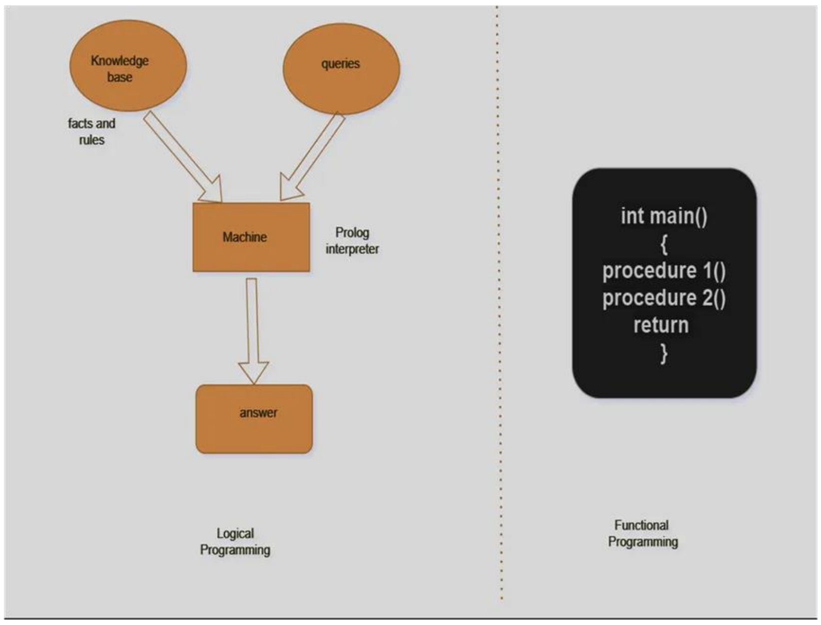


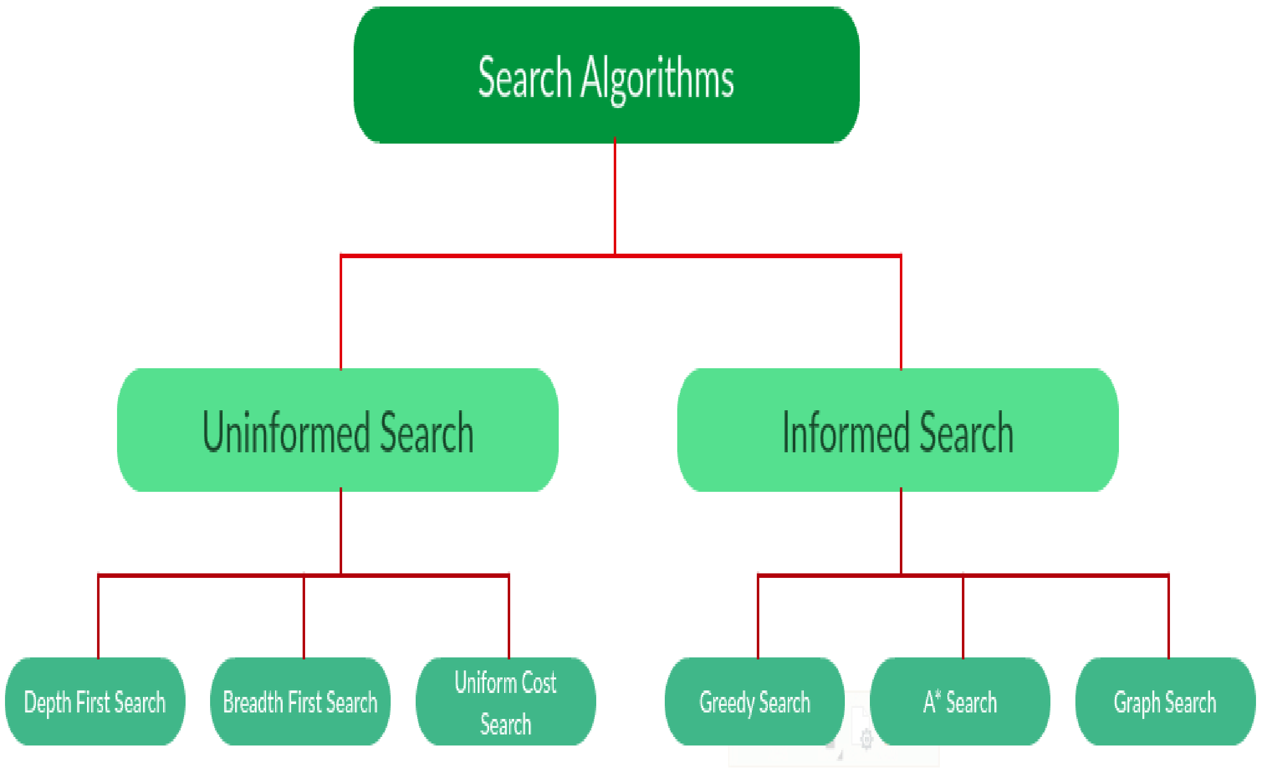
Image1: Logical & Functional programming

Applications of Prolog:

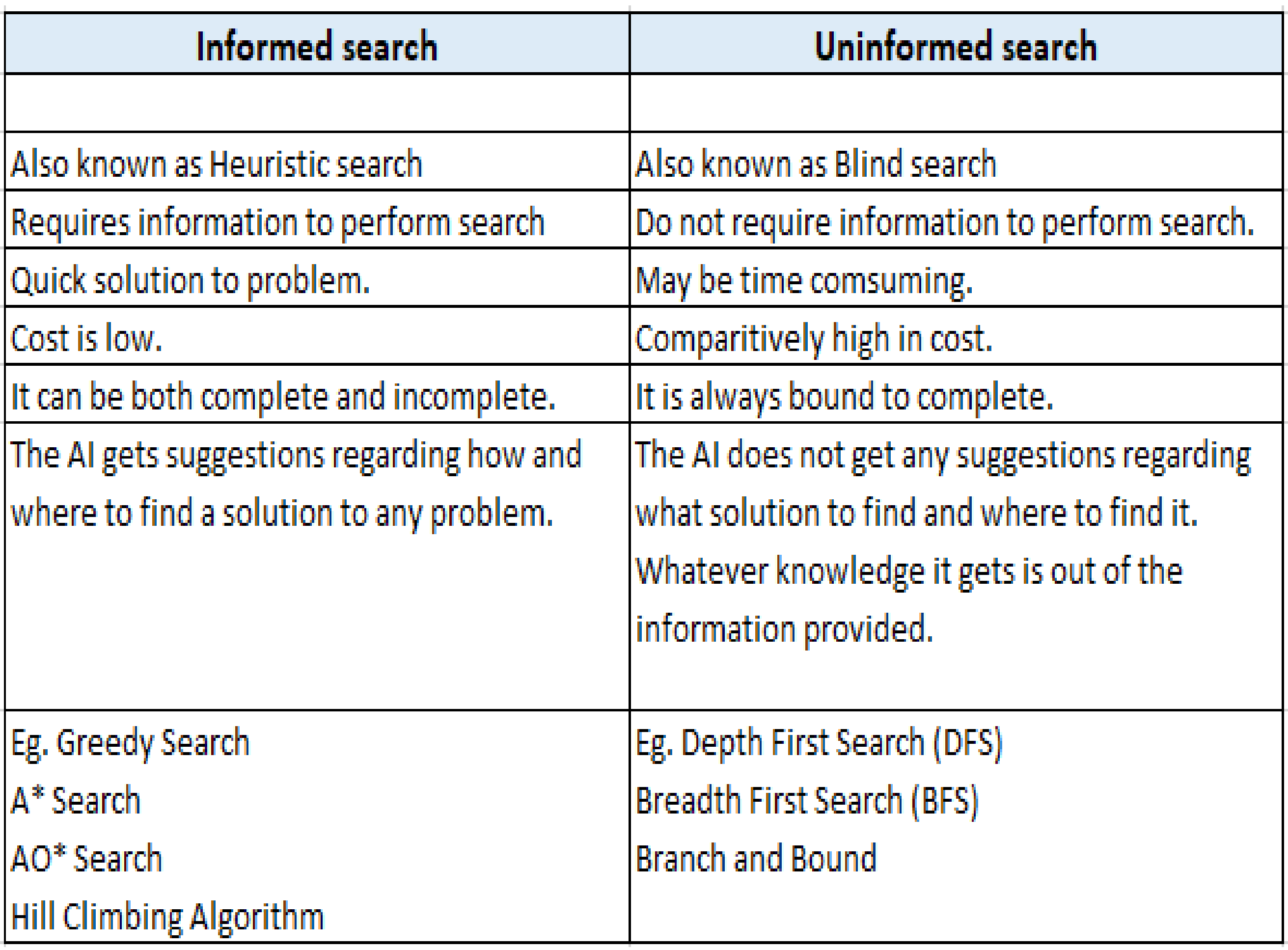
- Specification Language
- Robot Planning
- Natural language understanding
- Machine Learning
- Problem Solving
- Intelligent Database retrieval
- Expert System
- Automated Reasoning

TYPES OF SEARCH ALGORITHM-

- There are far too many powerful search algorithms out there to fit in a single article. Instead, this article will discuss *six* of the fundamental search algorithms, divided into *two* categories, as shown below.



DIFFERECE-



Informed search	Uninformed search
Also known as Heuristic search	Also known as Blind search
Requires information to perform search	Do not require information to perform search.
Quick solution to problem.	May be time comsuming.
Cost is low.	Comparitively high in cost.
It can be both complete and incomplete.	It is always bound to complete.
The AI gets suggestions regarding how and where to find a solution to any problem.	The AI does not get any suggestions regarding what solution to find and where to find it. Whatever knowledge it gets is out of the information provided.
Eg. Greedy Search A* Search AO* Search Hill Climbing Algorithm	Eg. Depth First Search (DFS) Breadth First Search (BFS) Branch and Bound

SEARCH PROBLEM-

- A well-defined problem can be described by
- Operators (Action) A:
- The actions available-often defined in terms of a mapping from a state to its successor.
- ➤ Is something that the agent can choose to do.
- Search tree:➤ (A graph with no undirected loops) in which the root node is the start state and the set of children for each node consists of the states reachable by taking any action.
- Search node:➤ Is a node in the search tree.

Goal: Is a state that the agent (solution) is trying to reach.

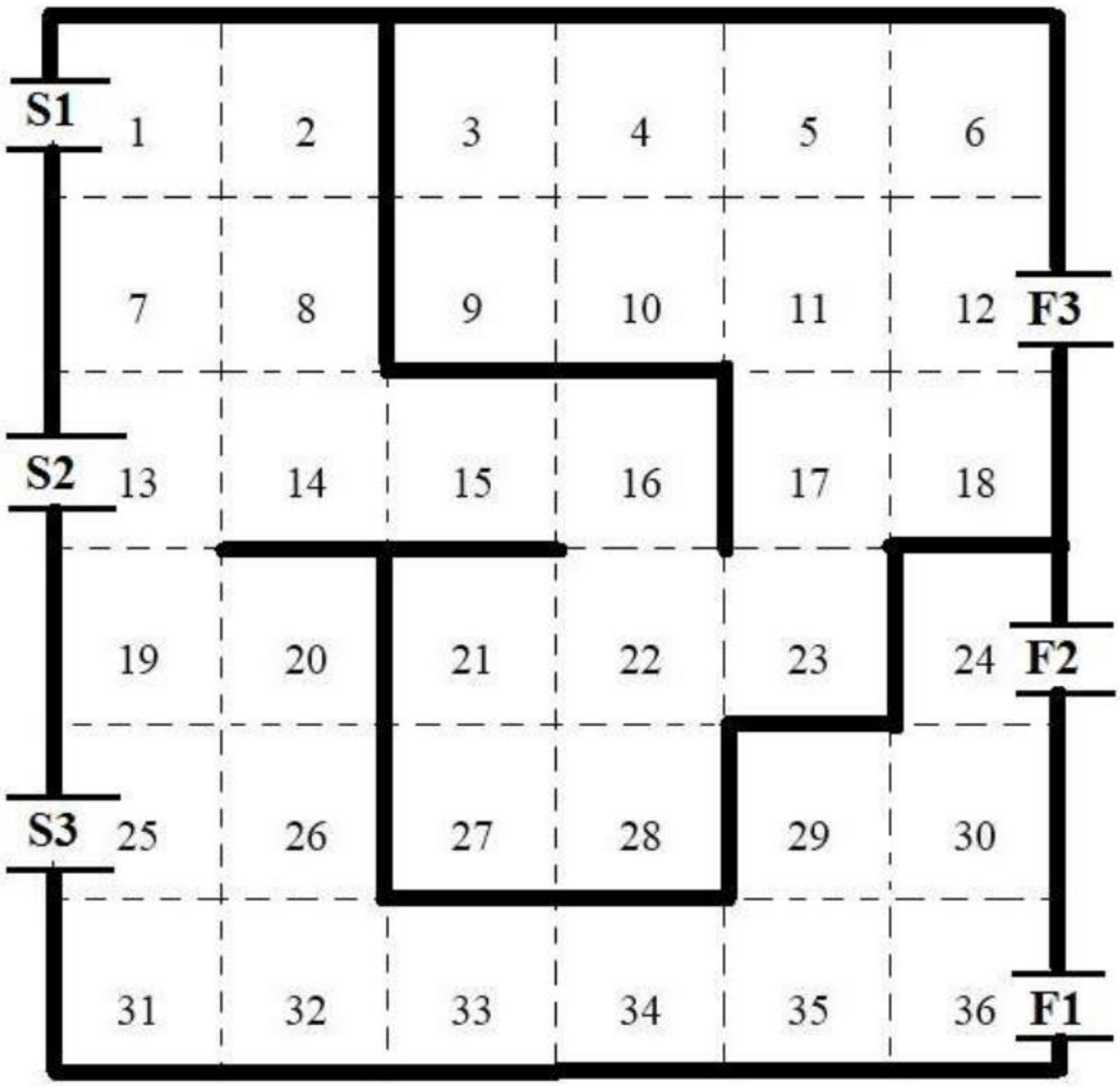
DEFINING A SERACH PROBLEM-

A well-defined problem can be described byW

- State space S: all possible configurations(situations) of the domain of interest.
- All states reachable from initial by any sequence of actions.
- ➤ Is a graph whose nodes are the set of all states, and whose links are actions that transform one state into another.
- Each state is an abstract representation of the environment.
- ➤ The state space is discrete.
- So An initial (start) state: ES
- ➤ Usually the current state.
- ➤ Sometimes one or several hypothetical states ("what if ...")
- Goal states G C S:➤ The set of end states - Often defined by a goal test rather than enumerating a set of states.

GRAPH FOR OPTIMAL PATHS-

- It contains the prolog code for finding all paths between start and finish point. The prolog implementations of single source single sink path and all source path are coded here. The project contain two tasks:
- **all path** : Given a maze it finds all path between all start and end points.
- **optimal path** : Given a maze it finds all optimal path between all start and end points



used to code the maze to be used by prolog for making judgements. It indicates you can move from 1 --> 2.

CanMove(1,2).

CanMove(1,7).

CanMove(1,8).

- To compile and run the query use:

  $ swipl -s optimalPathinMaze.pl.

- $?- gPath().

- **All Path**

  Code uses backtracking of prolog to find all possible path between the sourse and destination.

- It has two query:

  ?- path(s1,f3).

- given the start **S** and finish **F** it finds all path from start to finish.

  ?- gPath().

- It finds all path from all combinations of start and finish points.

- **Optimal Path**

  A similar backtracking based approach but uses a global variable *minPath* to keep track of minimum path length and use pruning to make code efficient.

- It has two query:

  ?- path(s1,f3).

- given the start **S** and finish **F** it finds the optimal path, returns all such paths and the optimal path length.

  gPath().

  It finds all optimal paths from all combinations of start and finish points. It returns the array of paths and *minLength* for each path.

- To stop the program or exit use:

  $ halt.

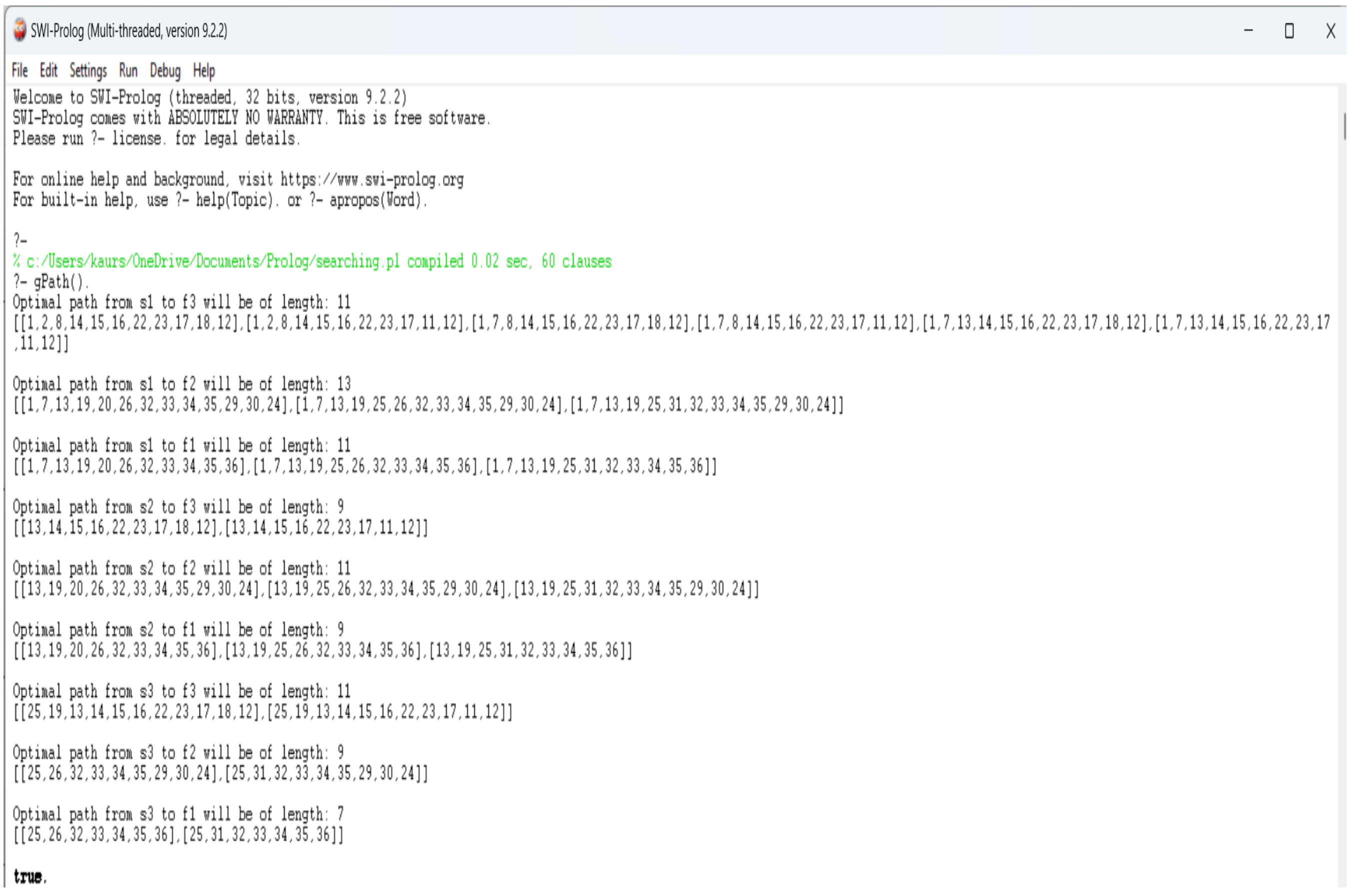
Program

```
%!	The maze is coded as facts denoting where can we move from one point taking a single step.

%!	These facts are used to move in a maze.

canMove(1,2).

canMove(1,7).

canMove(2,8).

canMove(3,4).

canMove(3,9).

canMove(4,5).

canMove(4,10).

canMove(5,6).

canMove(5,11).

canMove(6,12).

canMove(7,8).

canMove(7,13).

canMove(8,14).

canMove(9,10).

canMove(10,11).

canMove(11,12).

canMove(11,17).

canMove(12,18).

canMove(13,14).

canMove(13,19).

canMove(14,15).
canMove(15,16).
canMove(16,22).
canMove(17,18).
canMove(17,23).
canMove(19,20).
canMove(19,25).
canMove(20,26).
canMove(21,22).
canMove(21,27).
canMove(22,28).
canMove(22,23).
canMove(24,30).
canMove(25,26).
canMove(25,31).
canMove(26,32).
canMove(27,28).
canMove(29,30).
canMove(29,35).
canMove(31,32).
canMove(32,33).
canMove(33,34).
canMove(34,35).
canMove(35,36).

%! Shows if A can go to B then B can also go to A.
move(X,Y):-canMove(X,Y).
move(X,Y):-canMove(Y,X).
```

```
%! list of points from start to end
sPoints([s1,s2,s3]).
fPoints([f3,f2,f1]).

%! This fact maps the start points to square number in maze.
start(s1,1).
start(s2,13).
start(s3,25).

%! This fact maps the finish points to square number in maze.
end(f3,12).
end(f2,24).
end(f1,36).

%! adding variables to start query. PLength stores the current path length
%
path(S,F):-path(S,F,_,_,_).

%! Finds the intermediate paths
%! Visit array makes sure we do not end in a loop.
%! A global variable stores the min. Pathlength. Once we reach a smaller path
min Pathlength is readjusted.
%

interPath(S,F,_,[S,F],PLength):-move(S,F),PLength1 is PLength +
1,nb_getval(minPath,M),PLength1=<M,nb_setval(minPath,PLength1).
```

```
interPath(S,F,Visit,[S|Ss],PLength):-
move(S,Z),Z\=F,not(member(Z,Visit)),PLength1 is PLength +
1,nb_getval(minPath,M),PLength1=<M,interPath(Z,F,[S|Visit],Ss,PLength1).


%! compares path to minPathLength

comparePathLength(PathList,Pathlength,OptPath):-
member(A,PathList),length(A,PL1),PL1=Pathlength,OptPath=A.


%! finds the minPath for a pair of start and end, then lists the optimal paths for it.

path(S,F,Visit,Path,PLength):-
start(S,SPoint),end(F,FPoint),nb_setval(minPath,1000),PLength=1,findall(Path,(
Visit=[],interPath(SPoint,FPoint,Visit,Path,PLength)),Z),nb_getval(minPath,Pri
ntPath),findall(OptPath2,comparePathLength(Z,PrintPath,OptPath2),ZOpt),for
mat('Optimal path from ~w to ~w will be of length:
~w\n',[S,F,PrintPath]),write(ZOpt),write('\n\n').


gPath():-
sPoints(SList),fPoints(FList),findall((member(S,SList),member(F,FList)),path(S
,F),_).
```

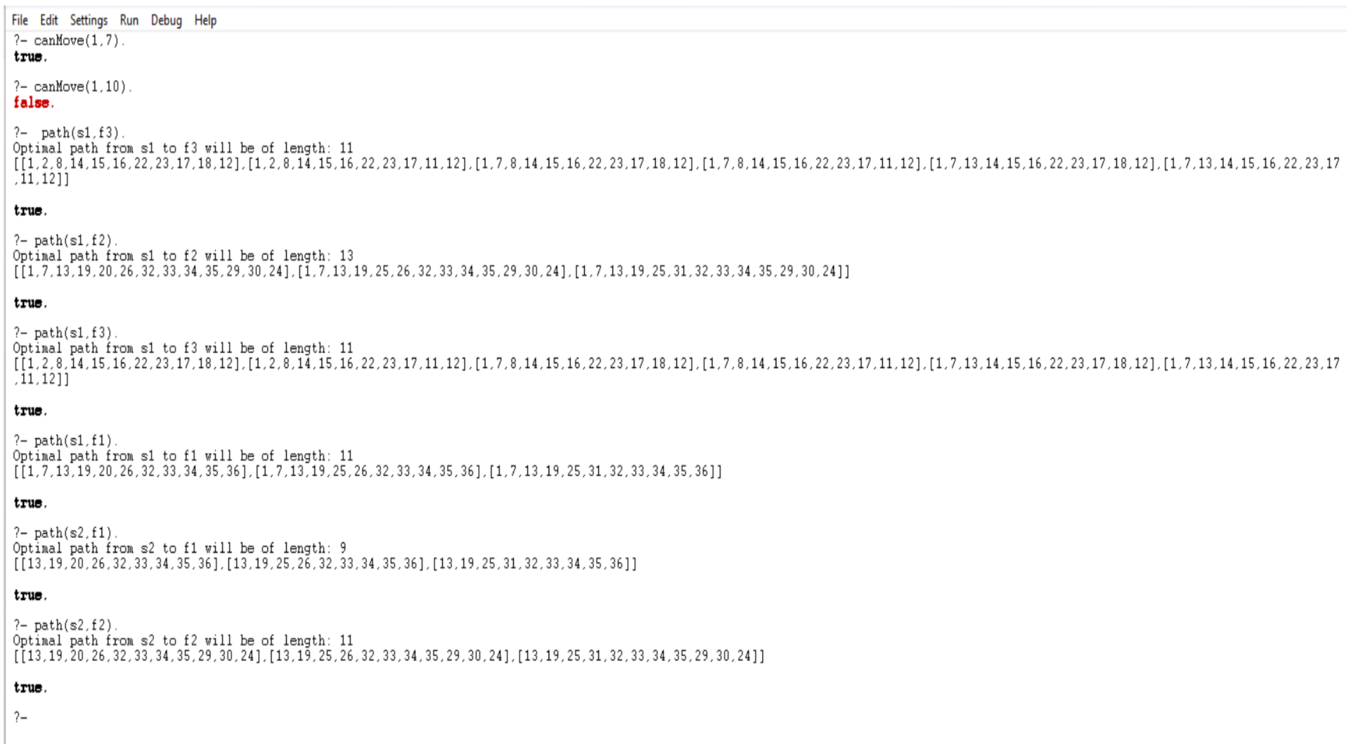
OUTPUT



SWI-Prolog (Multi-threaded, version 9.2.2)

```
Welcome to SWI-Prolog (threaded, 32 bits, version 9.2.2)
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/kaurs/OneDrive/Documents/Prolog/searching.pl compiled 0.02 sec, 60 clauses
?- gPath().
Optimal path from s1 to f3 will be of length: 11
[[1,2,8,14,15,16,22,23,17,18,12],[1,2,8,14,15,16,22,23,17,11,12],[1,7,8,14,15,16,22,23,17,18,12],[1,7,8,14,15,16,22,23,17,11,12],[1,7,13,14,15,16,22,23,17,18,12],[1,7,13,14,15,16,22,23,17
,11,12]]

Optimal path from s1 to f2 will be of length: 13
[[1,7,13,19,20,26,32,33,34,35,29,30,24],[1,7,13,19,25,26,32,33,34,35,29,30,24],[1,7,13,19,25,31,32,33,34,35,29,30,24]]

Optimal path from s1 to f1 will be of length: 11
[[1,7,13,19,20,26,32,33,34,35,36],[1,7,13,19,25,26,32,33,34,35,36],[1,7,13,19,25,31,32,33,34,35,36]]

Optimal path from s2 to f3 will be of length: 9
[[13,14,15,16,22,23,17,18,12],[13,14,15,16,22,23,17,11,12]]

Optimal path from s2 to f2 will be of length: 11
[[13,19,20,26,32,33,34,35,29,30,24],[13,19,25,26,32,33,34,35,29,30,24],[13,19,25,31,32,33,34,35,29,30,24]]

Optimal path from s2 to f1 will be of length: 9
[[13,19,20,26,32,33,34,35,36],[13,19,25,26,32,33,34,35,36],[13,19,25,31,32,33,34,35,36]]

Optimal path from s3 to f3 will be of length: 11
[[25,19,13,14,15,16,22,23,17,18,12],[25,19,13,14,15,16,22,23,17,11,12]]

Optimal path from s3 to f2 will be of length: 9
[[25,26,32,33,34,35,29,30,24],[25,31,32,33,34,35,29,30,24]]

Optimal path from s3 to f1 will be of length: 7
[[25,26,32,33,34,35,36],[25,31,32,33,34,35,36]]

true.
```

15.



```
?- canMove(1,7).
true.

?- canMove(1,10).
false.

?-  path(s1,f3).
Optimal path from s1 to f3 will be of length: 11
[[1,2,8,14,15,16,22,23,17,18,12],[1,2,8,14,15,16,22,23,17,11,12],[1,7,8,14,15,16,22,23,17,18,12],[1,7,8,14,15,16,22,23,17,11,12],[1,7,13,14,15,16,22,23,17,18,12],[1,7,13,14,15,16,22,23,17
,11,12]]

true.

?- path(s1,f2).
Optimal path from s1 to f2 will be of length: 13
[[1,7,13,19,20,26,32,33,34,35,29,30,24],[1,7,13,19,25,26,32,33,34,35,29,30,24],[1,7,13,19,25,31,32,33,34,35,29,30,24]]

true.

?- path(s1,f3).
Optimal path from s1 to f3 will be of length: 11
[[1,2,8,14,15,16,22,23,17,18,12],[1,2,8,14,15,16,22,23,17,11,12],[1,7,8,14,15,16,22,23,17,18,12],[1,7,8,14,15,16,22,23,17,11,12],[1,7,13,14,15,16,22,23,17,18,12],[1,7,13,14,15,16,22,23,17
,11,12]]

true.

?- path(s1,f1).
Optimal path from s1 to f1 will be of length: 11
[[1,7,13,19,20,26,32,33,34,35,36],[1,7,13,19,25,26,32,33,34,35,36],[1,7,13,19,25,31,32,33,34,35,36]]

true.

?- path(s2,f1).
Optimal path from s2 to f1 will be of length: 9
[[13,19,20,26,32,33,34,35,36],[13,19,25,26,32,33,34,35,36],[13,19,25,31,32,33,34,35,36]]

true.

?- path(s2,f2).
Optimal path from s2 to f2 will be of length: 11
[[13,19,20,26,32,33,34,35,29,30,24],[13,19,25,26,32,33,34,35,29,30,24],[13,19,25,31,32,33,34,35,29,30,24]]

true.

?-
```

Hardware Configuration:

RAM	512 MB
Hard disk	10 GB
Processor	1.0 GHz
Network	Broadband

Software Configuration:

Operating System	Windows
Application Software	SWI PROLOG
Programming Languages	PROLOG

REFRENCES

- **https://www.geeksforgeeks.org/prolog-an-introduction/**

- **https://study.com/academy/lesson/prolog-in-ai-definition-uses.html**

- ARTIFICIAL INTELLIGENCE by - Elaine Rich , Kevin Knight , Shivashankar B Nair

- https://www.scribd.com/document/59655200/7348208-Prolog-Program-1