Artificial Intelligence
Assignment 8

Adm. No.: U19CS082
Name: SOURABH PATEL

Q). Implement N queens problem using the below algorithms in prolog.
Compare the complexity of both algorithms.
Which algorithm is best suited for implementing the N queens problem and why?

1. **Breadth-First Search**
2. **Depth First Search**

```
%dfs
row(0,[]).
row(N,[0|T]) :- N>0,
       N1 is N-1,
       row(N1,T).
col(0,_,[]).
col(N,Row,[Row|T]) :- N>0,
             N1 is N-1,
             col(N1,Row,T).


empty(N,Board) :- row(N,Row),col(N,Row,Board).


getXY(X,Y,[_|Mt],Z) :- Y>0,
           Y1 is Y-1,
           getXY(X,Y1,Mt,Z),!.


getXY(X,0,[M|_],Z) :- getXYx(X,M,Z).


getXYx(X,[_|Mt],Z) :- X>0,
              X1 is X-1,
              getXYx(X1,Mt,Z).


getXYx(0,[M|_],Z) :- Z is M.
```

```
changeXY(X,Y,[M|Mt],N) :- Y>0,
              Y1 is Y-1,
              changeXY(X,Y1,Mt,N1),
              N=[M|N1].


changeXY(X,0,[M|Mt],N) :- changeX(X,M,Nr),
              N=[Nr|Mt].


changeX(X,[H|T],R) :- X>0,
             X1 is X-1,
             changeX(X1,T,N1),
             R=[H|N1].

changeX(0,[_|T],[1|T]).
checkup(-1,_,_).


checkup(X,Y,Board) :- X>=0,
              X1 is X-1,
              getXY(X,Y,Board,Val),
              Val is 0,
              checkup(X1,Y,Board).


checkupleft(-1,_,_).
checkupleft(_,-1,_).
checkupleft(X,Y,Board) :- X>=0,
              Y>=0,
              X1 is X-1,
              Y1 is Y-1,
              getXY(X,Y,Board,Val),
              Val is 0,
              checkupleft(X1,Y1,Board).
```

```
checkupright(_,N,N,_).
checkupright(-1,_,_,_).
checkupright(X,Y,N,Board) :- X>=0,
                X1 is X-1,
                Y<N,
                Y1 is Y+1,
                getXY(X,Y,Board,Val),
                Val is 0,
```

```
                checkupright(X1,Y1,N,Board).


chk(I,N,J,Board,Res) :- checkup(I,J,Board),
            checkupleft(I,J,Board),
            checkupright(I,J,N,Board),
changeXY(I,J,Board,Res).
chk(I,N,J,Board,Res) :- J>0,
            J1 is J-1,
            chk(I,N,J1,Board,Res).


placeQueen(I,N,Board,Res) :- chk(I,N,N,Board,Res).


dfs(B1,N,N,B2) :- B2=B1.
dfs(EmptyBoard,I,N,Board) :- I<N,
                J is I+1,
                placeQueen(I,N,EmptyBoard,TempBoard),
                dfs(TempBoard,J,N,Board).


nqueens(N,Board) :- empty(N,EmptyBoard),
           dfs(EmptyBoard,0,N,Board).
printP([]).
printP([X|Pt]) :-  print (X),nl,printP(Pt).


%:- nqueens(8,X),printP(X).
```

```
%bfs
nqueenbfs(N,Boards) :-
empty(N,EmptyBoard),bfs([[EmptyBoard,0]],N,Boards).


bfs([],_,[]).
bfs([[InputBoard,I]|T],N,Boards) :- I is N,
                    bfs(T,N,Tb),
                    Boards=[InputBoard|Tb].
bfs([[InputBoard,I]|T],N,Boards) :- I<N, J is I+1,
                    placeQueen(I,N,InputBoard,TempBoard),
                    append(T,[[TempBoard,J]],BoardsT),
                    bfs(BoardsT,N,Boards).


%:- nqueenbfs(8,[H|_]),printP(H)
```

```
?- nqueens(8,X),printP(X).
[0,0,1,0,0,0,0,0]
[0,0,0,0,0,1,0,0]
[0,0,0,1,0,0,0,0]
[0,1,0,0,0,0,0,0]
[0,0,0,0,0,0,0,1]
[0,0,0,0,1,0,0,0]
[0,0,0,0,0,0,1,0]
[1,0,0,0,0,0,0,0]
X = [[0, 0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 1, 0|...], [0, 0, 0, 1, 0, 0|...]
, [0, 1, 0, 0, 0|...], [0, 0, 0, 0|...], [0, 0, 0|...], [0, 0|...], [1|...]] .

?- nqueenbfs(8,[H|_]),printP(H).
[0,0,1,0,0,0,0,0]
[0,0,0,0,0,1,0,0]
[0,0,0,1,0,0,0,0]
[0,1,0,0,0,0,0,0]
[0,0,0,0,0,0,0,1]
[0,0,0,0,1,0,0,0]
[0,0,0,0,0,0,1,0]
[1,0,0,0,0,0,0,0]
H = [[0, 0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 1, 0|...], [0, 0, 0, 1, 0, 0|...]
, [0, 1, 0, 0, 0|...], [0, 0, 0, 0|...], [0, 0, 0|...], [0, 0|...], [1|...]] .

?-
```