**11. 8 QUEENS**

:- use\_module(library(clpfd)).

:- initialization(main).

main :-

N = 8,

n\_queens(N, Qs),

write(Qs), nl,

halt.

n\_queens(N, Qs) :-

length(Qs, N),

Qs ins 1..N,

safe\_queens(Qs),

labeling([], Qs).

safe\_queens([]).

safe\_queens([Q|Qs]) :-

safe\_queens(Qs),

safe\_queens(Qs, Q, 1).

safe\_queens([], \_, \_).

safe\_queens([Q|Qs], Q0, D0) :-

Q0 #\= Q,

abs(Q0 - Q) #\= D0,

D1 #= D0 + 1,

safe\_queens(Qs, Q0, D1).

**QUERY:**

n\_queens(8, Qs).

**13. DFS**

solve( Node, Solution) :-

depthfirst( [], Node, Solution).

depthfirst( Path, Node, [Node | Path] ) :-

goal( Node).

depthfirst( Path, Node, Sol) :-

s( Node, Node1),

\+ member( Node1, Path),

depthfirst( [Node | Path], Node1, Sol).

depthfirst2( Node, [Node], \_) :-

goal( Node).

depthfirst2( Node, [Node | Sol], Maxdepth) :-

Maxdepth > 0,

s( Node, Node1),

Max1 is Maxdepth - 1,

depthfirst2( Node1, Sol, Max1).

goal(f).

goal(j).

s(a,b).

s(a,c).

s(b,d).

s(b,e).

s(c,f).

s(c,g).

s(d,h).

s(e,i).

s(e,j).

**QUERY:**

solve(a, Solution)

**14. 8 PUZZLE**

ids :-

start(State),

length(Moves, N),

dfs([State], Moves, Path), !,

show([start|Moves], Path),

format('~nmoves = ~w~n', [N]).

dfs([State|States], [], Path) :-

goal(State), !,

reverse([State|States], Path).

dfs([State|States], [Move|Moves], Path) :-

move(State, Next, Move),

not(memberchk(Next, [State|States])),

dfs([Next,State|States], Moves, Path).

show([], \_).

show([Move|Moves], [State|States]) :-

State = state(A,B,C,D,E,F,G,H,I),

format('~n~w~n~n', [Move]),

format('~w ~w ~w~n',[A,B,C]),

format('~w ~w ~w~n',[D,E,F]),

format('~w ~w ~w~n',[G,H,I]),

show(Moves, States).

% Empty position is marked with '\*'

start( state(6,1,3,4,\*,5,7,2,0) ).

goal( state(\*,0,1,2,3,4,5,6,7) ).

move( state(\*,B,C,D,E,F,G,H,J), state(B,\*,C,D,E,F,G,H,J), right).

move( state(\*,B,C,D,E,F,G,H,J), state(D,B,C,\*,E,F,G,H,J), down ).

move( state(A,\*,C,D,E,F,G,H,J), state(\*,A,C,D,E,F,G,H,J), left ).

move( state(A,\*,C,D,E,F,G,H,J), state(A,C,\*,D,E,F,G,H,J), right).

move( state(A,\*,C,D,E,F,G,H,J), state(A,E,C,D,\*,F,G,H,J), down ).

move( state(A,B,\*,D,E,F,G,H,J), state(A,\*,B,D,E,F,G,H,J), left ).

move( state(A,B,\*,D,E,F,G,H,J), state(A,B,F,D,E,\*,G,H,J), down ).

move( state(A,B,C,\*,E,F,G,H,J), state(\*,B,C,A,E,F,G,H,J), up ).

move( state(A,B,C,\*,E,F,G,H,J), state(A,B,C,E,\*,F,G,H,J), right).

move( state(A,B,C,\*,E,F,G,H,J), state(A,B,C,G,E,F,\*,H,J), down ).

move( state(A,B,C,D,\*,F,G,H,J), state(A,\*,C,D,B,F,G,H,J), up ).

move( state(A,B,C,D,\*,F,G,H,J), state(A,B,C,D,F,\*,G,H,J), right).

move( state(A,B,C,D,\*,F,G,H,J), state(A,B,C,D,H,F,G,\*,J), down ).

move( state(A,B,C,D,\*,F,G,H,J), state(A,B,C,\*,D,F,G,H,J), left ).

move( state(A,B,C,D,E,\*,G,H,J), state(A,B,\*,D,E,C,G,H,J), up ).

move( state(A,B,C,D,E,\*,G,H,J), state(A,B,C,D,\*,E,G,H,J), left ).

move( state(A,B,C,D,E,\*,G,H,J), state(A,B,C,D,E,J,G,H,\*), down ).

move( state(A,B,C,D,E,F,\*,H,J), state(A,B,C,D,E,F,H,\*,J), left ).

move( state(A,B,C,D,E,F,\*,H,J), state(A,B,C,\*,E,F,D,H,J), up ).

move( state(A,B,C,D,E,F,G,\*,J), state(A,B,C,D,E,F,\*,G,J), left ).

move( state(A,B,C,D,E,F,G,\*,J), state(A,B,C,D,\*,F,G,E,J), up ).

move( state(A,B,C,D,E,F,G,\*,J), state(A,B,C,D,E,F,G,J,\*), right).

move( state(A,B,C,D,E,F,G,H,\*), state(A,B,C,D,E,\*,G,H,F), up ).

move( state(A,B,C,D,E,F,G,H,\*), state(A,B,C,D,E,F,G,\*,H), left )

**QUERY:**

ids.

**15. TRAVELLING SALESMAN**

road("tampa","houston",200).

road("gordon","tampa",300).

road("houston","gordon",100).

road("houston","kansas\_city",120).

road("gordon","kansas\_city",130).

route(Town1,Town2,Distance):-

road(Town1,Town2,Distance).

route(Town1,Town2,Distance):-

road(Town1,X,Dist1),

route(X,Town2,Dist2),

Distance=Dist1+Dist2,!.

**QUERY:**

route("tampa", "houston", Distance).