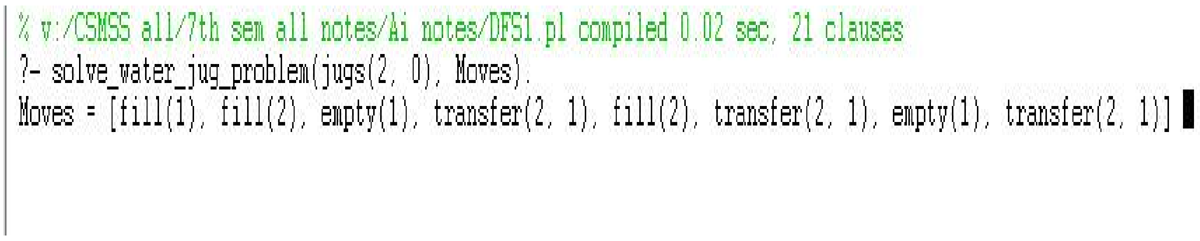
Water Jug problem using DFS

Program:-

```
% Solve the Water Jug Problem using DFS

% solve_dfs(State, History, Moves, FinalState):
% If the State is the FinalState, return an empty list of Moves.
solve_dfs(State, History, [], State) :- % Removed FinalState
    true. % Always succeeds

% solve_dfs(State, History, Moves, FinalState):
% Move to the next state, update the history, and continue searching.
solve_dfs(State, History, [Move | Moves], FinalState) :-
    move(State, Move),
    update(State, Move, State1),
    legal(State1),
    not(member(State1, History)),
    solve_dfs(State1, [State1 | History], Moves, FinalState).

% Query to find a solution to the Water Jug Problem.
solve_water_jug_problem(FinalState, Moves) :- % Pass FinalState as an argument
    initial_state(jugs(0, 0)),
    solve_dfs(jugs(0, 0), [jugs(0, 0)], Moves, FinalState). % Use FinalState as the final goal

% Define the capacity of the jugs as constants.
capacity(1, 4).
capacity(2, 3).

% Define initial states.
initial_state(jugs(0, 0)).

% Define the legal states.
legal(jugs(V1, V2)) :- V1 >= 0, V2 >= 0.

% Define the available moves.
move(jugs(V1, V2), fill(1)) :- V1 < 4.
move(jugs(V1, V2), fill(2)) :- V2 < 3.
move(jugs(V1, V2), empty(1)) :- V1 > 0.
move(jugs(V1, V2), empty(2)) :- V2 > 0.
move(jugs(V1, V2), transfer(1, 2)) :- V1 > 0, V2 < 3.
move(jugs(V1, V2), transfer(2, 1)) :- V2 > 0, V1 < 4.

% Define how to update the state after a move.
update(jugs(V1, V2), fill(1), jugs(4, V2)).
update(jugs(V1, V2), fill(2), jugs(V1, 3)).
update(jugs(V1, V2), empty(1), jugs(0, V2)).
update(jugs(V1, V2), empty(2), jugs(V1, 0)).
update(jugs(V1, V2), transfer(1, 2), jugs(NewV1, NewV2)) :-
    Liquid is V1 + V2,
    (Liquid =< 3, NewV1 = 0, NewV2 = Liquid;
      Liquid > 3, NewV1 = Liquid - 3, NewV2 = 3).
update(jugs(V1, V2), transfer(2, 1), jugs(NewV1, NewV2)) :-
    Liquid is V1 + V2,
    (Liquid =< 4, NewV1 = Liquid, NewV2 = 0;
      Liquid > 4, NewV1 = 4, NewV2 = Liquid - 4).
```

```
% Adjust the liquid between the jugs.
adjust(Liquid, Excess, Liquid, 0) :- Excess =< 0.
adjust(Liquid, Excess, V, Excess) :- Excess > 0, V is Liquid - Excess.
```

OUTPUT:-



```
% v:/CSMSS all/7th sem all notes/Ai notes/DFS1.pl compiled 0.02 sec, 21 clauses
?- solve_water_jug_problem(jugs(2, 0), Moves).
Moves = [fill(1), fill(2), empty(1), transfer(2, 1), fill(2), transfer(2, 1), empty(1), transfer(2, 1)]
```