GraphColoring Code:

```prolog
:- use_module(library(pce)).
% ---------------- Colors ----------------
color(red).
color(green).
color(blue).
color(yellow).


color_list([red, green, blue, yellow]).


% ---------------- Adjacency ----------------
neighbor(arad, zerind).
neighbor(arad, sibiu).
neighbor(arad, timisoara).
neighbor(zerind, oradea).
neighbor(sibiu, oradea).
neighbor(sibiu, fagaras).
neighbor(sibiu, rimnicu_vilcea).
neighbor(rimnicu_vilcea, pitesti).
neighbor(rimnicu_vilcea, craiova).
neighbor(fagaras, bucharest).
neighbor(pitesti, bucharest).
neighbor(craiova, dobreta).

% Bidirectional edges
connected(X,Y) :- neighbor(X,Y).
connected(X,Y) :- neighbor(Y,X).

% ---------------- City Positions for Graphics ----------------
```

```prolog
city_position(arad, 100, 100).
city_position(zerind, 200, 50).
city_position(oradea, 300, 50).
city_position(sibiu, 250, 150).
city_position(timisoara, 100, 200).
city_position(fagaras, 350, 150).
city_position(rimnicu_vilcea, 300, 250).
city_position(pitesti, 400, 250).
city_position(bucharest, 450, 350).
city_position(craiova, 250, 350).
city_position(dobreta, 150, 350).


% ---------------- Validation ----------------
valid_coloring([]).
valid_coloring([(City,Color)|Rest]) :-
    forall((connected(City,Other), member((Other,OtherColor), Rest)),
        Color \= OtherColor),
    valid_coloring(Rest).


% ---------------- Ask User with Drop-down ----------------
ask_colors(Cities, Solution) :-
    new(D, dialog('Select Colors for Cities')),

    % Create drop-down menus for each city
    forall(member(City, Cities),
        ( color_list(Colors),
          new(Menu, menu(City, cycle)),
          forall(member(C, Colors), send(Menu, append, C)),
          send(D, append, Menu)
```

```prolog
    )),

    % Add OK button to confirm
    send(D, append,
        button(ok, message(D, return, ok))),

    get(D, confirm, _),   % waits until user presses OK
    % Collect user choices
    findall((City,Color),
          ( member(City, Cities),
            get(D, member, City, Menu),
            get(Menu, selection, Color)
          ),
          Solution),
    free(D).

% ---------------- Graphics Display ----------------
show_coloring(Solution) :-
    new(Window, picture('Romania Manual Coloring')),
    send(Window, size, size(600,500)),

    % Draw edges
    forall(connected(C1,C2),
        ( city_position(C1, X1, Y1),
          city_position(C2, X2, Y2),
          new(Line, line(X1+20,Y1+20,X2+20,Y2+20)),
          send(Window, display, Line)
        )),
```

```prolog
    % Draw cities
    forall(member((City,Color), Solution),
        ( city_position(City, X, Y),
          new(Circle, circle(40)),
          send(Circle, fill_pattern, colour(Color)),
          send(Window, display, Circle, point(X,Y)),
          new(Label, text(City)),
          send(Window, display, Label, point(X,Y+50))
        )),
    send(Window, open).


% ---------------- Run ----------------
run_manual :-
    Cities = [arad, zerind, oradea, sibiu, timisoara,
              fagaras, rimnicu_vilcea, pitesti,
              bucharest, craiova, dobreta],
    ask_colors(Cities, Solution),
    (   valid_coloring(Solution)
    ->  writeln('? Coloring is valid!'),
        show_coloring(Solution)
    ;   writeln('? Invalid coloring! Adjacent cities have same color.'),
        show_coloring(Solution)
    ).
```

Output:

Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)

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
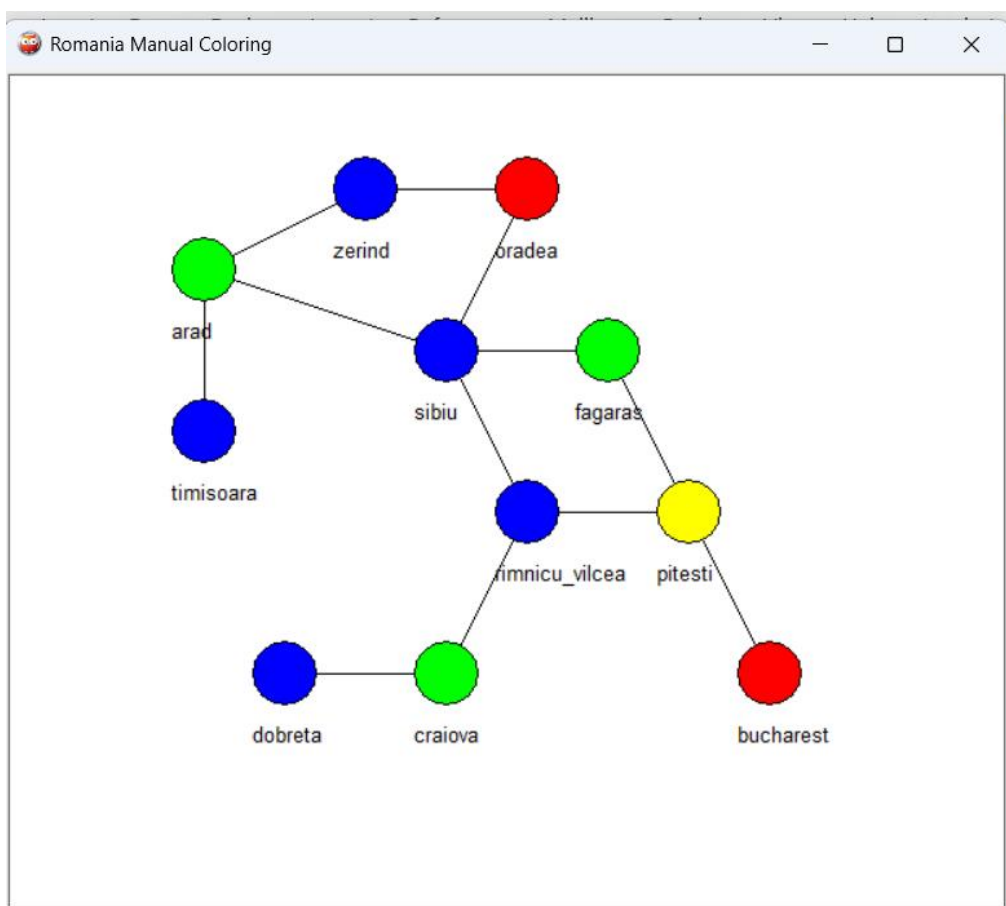
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['C:/Users/pruth/OneDrive/Desktop/grapph_coloring.pl'].
true.

?- run_manual
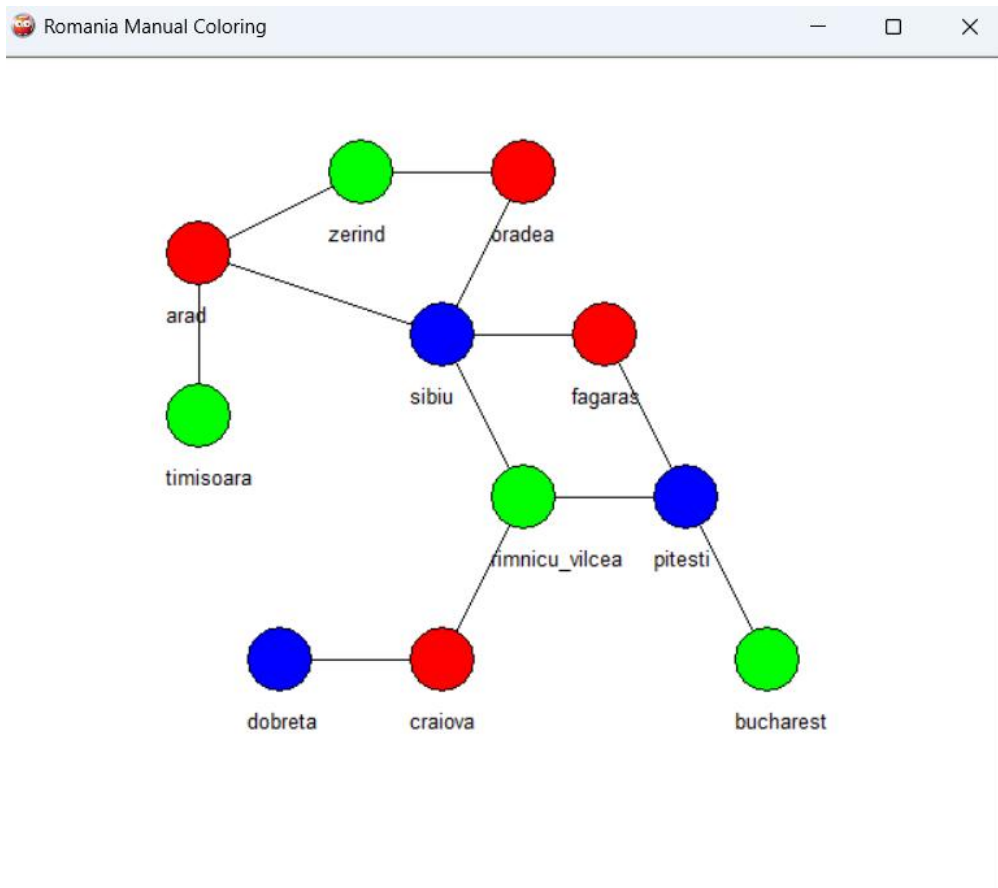? Invalid coloring! Adjacent cities have same color.
true..

?- run_manual.

? Coloring is valid!

true.

Mobile View App