```
:- use_module(library(pce)).

% Sample graph
edge(a, b).
edge(a, c).
edge(b, d).
edge(b, e).
edge(c, f).
edge(e, g).

% BFS algorithm
bfs(Start, Goal, Path) :-
   bfs_helper([[Start]], Goal, RevPath),
   reverse(RevPath, Path).

bfs_helper([[Goal|Rest]|_], Goal, [Goal|Rest]).
bfs_helper([CurrentPath|OtherPaths], Goal, Path) :-
   CurrentPath = [Node|_],
   findall([Next|CurrentPath],
         (edge(Node, Next), \+ member(Next, CurrentPath)),
         NewPaths),
   append(OtherPaths, NewPaths, UpdatedPaths),
   bfs_helper(UpdatedPaths, Goal, Path).

% GUI to show BFS path with clickable options
go :-
   findall(Node, (edge(Node,_); edge(_,Node)), NodesDup),
   sort(NodesDup, Nodes),  % Unique nodes
   new(Dialog, dialog('BFS GUI')),
   send(Dialog, append, new(StartMenu, menu(start_node, cycle))),
   send(Dialog, append, new(GoalMenu, menu(goal_node, cycle))),
   add_nodes_to_menu(Nodes, StartMenu),
   add_nodes_to_menu(Nodes, GoalMenu),
   send(Dialog, append, button(run, message(@prolog, run_bfs, StartMenu?selection,
GoalMenu?selection))),
   send(Dialog, open).

add_nodes_to_menu([], _).
add_nodes_to_menu([H|T], Menu) :-
   send(Menu, append, H),
   add_nodes_to_menu(T, Menu).

run_bfs(Start, Goal) :-
   (bfs(Start, Goal, Path) ->
      format('BFS Path: ~w~n', [Path]),
      display_result(Path)
   ;
      send(@display, inform, 'No path found')
```

```
    ).

display_result(Path) :-
    (get(@display, exists) -> true ; new(@display, dialog('BFS Result'))),
    send(@display, clear),
    send(@display, append, text(Path)),
    send(@display, open).
```