FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

EXERCÍCIOS DE PROGRAMAÇÃO EM LÓGICA

LUÍS PAULO REIS
DANIEL CASTRO SILVA

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO

PROGRAMAÇÃO EM LÓGICA - 3º ANO

SETEMBRO DE 2007

Exercício PESQ1. Pesquisa de Ligação num Grafo

Escreva um programa em Prolog que determine um caminho entre dois nós de um Grafo.

a) Utilizando pesquisa em profundidade (evitando ciclos);

b) Utilizando pesquisa em largura.

De forma a experimentar o programa, considere que o grafo é definido pelos seguintes factos:

```
ligado(a,b).
ligado(a,c).
ligado(b,d).
ligado(b,e).
ligado(b,f).
ligado(c,g).
ligado(d,h).
ligado(d,i).
ligado(f,i).
ligado(f,j).
ligado(f,k).
ligado(g,l).
ligado(g,m).
ligado(k,n).
ligado(l,o).
ligado(i,f).
```

Solução:

```
/* Para testar, utilize:
        ?- resolva_larg(No_inicial, No_meta, Solucao).
        ?- resolva_prof(No_inicial, No_meta, Solucao).
  onde No_inicial é o inicio do grafo e No_meta é a meta que se deseja atingir.
  Solucao retorna o caminho entre No_inicial e No_meta, na forma de uma lista.*/

// Utilitários de manipulação de Listas

membro(X, [X|_]):- !.
membro(X, [_|Y]):- membro(X,Y).

concatena([], L, L).
concatena([X|Y], L, [X|Lista]):- concatena(Y, L, Lista).

inverte([X], [X]).
inverte([X|Y], Lista):- inverte(Y, Lista1), concatena(Lista1, [X], Lista).

// a) Pesquisa em Profundidade

// Encontra o caminho Solucao entre No_inicial e No_meta
resolva_prof(No_inicial, No_meta, Solucao):-
      profundidade([], No_inicial, No_meta, Sol_inv),
      inverte(Sol_inv, Solucao).
```

```
// Realiza a pesquisa em profundidade
profundidade(Caminho, No_meta, No_meta, [No_meta|Caminho]).
profundidade(Caminho, No, No_meta, Sol):-
      ligado(No, No1),
      not membro(No1, Caminho),            % previne ciclos
      profundidade([No|Caminho], No1, No_meta, Sol).
```

// b) Pesquisa em Largura

```
// Acha todos os X onde Y esta satisfeito e retorna numa lista Y
ache_todos(X, Y, Z):- bagof(X, Y, Z), !.
ache_todos(_, _, []).

// Estende a fila ate um filho N1 de N, verificando se N1
// não pertence à fila, prevenindo, assim, ciclos
estende_ate_filho([N|Trajectoria], [N1,N|Trajectoria]):-
      ligado(N, N1),
      not membro(N1, Trajectoria).

// Encontra o caminho Solucao entre No_inicial e No_Meta
resolva_larg(No_inicial, No_meta, Solucao):-
      largura([[No_inicial]], No_meta, Sol1),
      inverte(Sol1, Solucao).

// Realiza a pesquisa em largura
largura([[No_meta|T]|_],No_meta,[No_meta|T]).
largura([T|Fila],No_meta,Solucao):-
   ache_todos(ExtensaoAteFilho,estende_ate_filho(T,ExtensaoAteFilho),Extensoes),
   concatena(Fila, Extensoes, FilaExtendida),
   largura(FilaExtendida, No_meta, Solucao).
```

Exercício PESQ2. Pesquisa de Ligação mais Rápida num Grafo

Altera o Programa PESQ1 de forma a que cada ligação tenha um custo. Escreva um programa que lhe permita encontrar o caminho mais rápido (de menor custo) entre dois nós do grafo.

Exercício PESQ3. Pesquisa de Ligação com Visita a uma Lista de Nós

Altera o Programa PESQ1 ou PESQ2 de forma a encontrar um caminho entre dois nós do grafo que visite todos os nós contidos numa lista de nós fornecida como parâmetro de entrada ao algoritmo.

Exercício PESQ4. Todos os Trajectos

Altera o Programa anterior de forma a que retorne todos os trajectos possíveis entre dois nós do grafo

Exercício PESQ5. Disponibilidades para Reuniões

Uma empresa pretende realizar um encontro num determinado mês que reúna alguns responsáveis por várias unidades dessa empresa. As disponibilidades em termos de dias por parte de cada membro a estar presente na reunião são dadas por factos do tipo:

```
disponibilidade(nome, lista_dias_disponíveis).
```

onde cada membro da lista de dias disponíveis é do tipo:

```
disp(primeiro_dia, último_dia).
```

Por exemplo:

```
disponibilidade(pedro, [disp(2,4), disp(12,20), disp(25,28)])
```

indica que Pedro está disponível dos dias 2 a 4, dos dias 12 a 20 e dos dias 25 a 28.

a) Escreva um programa Prolog que receba como argumento um dia e retorne todos os nomes de pessoas disponíveis nesse dia para uma reunião.

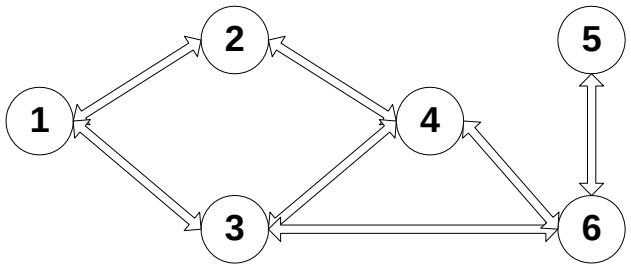
b) Escreva um programa Prolog que receba uma lista de nomes e o número mínimo de dias consecutivos para realizar uma reunião e crie uma lista com elementos do tipo poss(primeiro_dia, último_dia) representando as possibilidades de realização da reunião envolvendo todas as pessoas da lista de nomes.

Exercício PESQ6. Pesquisa de Ligação com Visita a uma Lista de Nós

Considere que um grafo não dirigido é representado por um conjunto de cláusulas unitárias da forma ***ligacao(No1, No2)***.

Exemplo:

ligacao(1, 2).
ligacao(1, 3).
ligacao(2, 4).
ligacao(3, 4).
ligacao(3,6).
ligacao(4, 6).
ligacao(5,6).



a) Escreva um predicado ***caminho(+NoInicio, +NoFim, -Lista),*** que dados dois nós do grafo, calcule um possível caminho (não necessariamente o mais curto) entre esses nós. *Nota: Suponha que a dimensão máxima do caminho é 5*.

Exemplos:

?- caminho(2, 3, Lista).
Lista = [2,4,3] ; Lista = [2,4,6,3] ; Lista = [2,1,3] ; no
?- caminho(1, 5, Lista) *%Uma possível solução é: Vai do nó 1 para o 2, depois para o 4,para o 6 e finalmente para o 5.*
Lista = [1,2,4,6,5] ; Lista = [1,2,4,3,6,5] ; Lista = [1,3,4,6,5] ; Lista = [1,3,6,5] ; no
?- caminho(2, 2, Lista).
Lista = [2,4,2] ; Lista = [2,4,6,3,1,2] ; Lista = [2,4,3,1,2] ; Lista = [2,1,2] ; Lista = [2,1,3,4,2] ; Lista = [2,1,3,6,4,2] ; no

b) Escreva um predicado ***ciclos(+No, +Comp, -Lista),*** que dado um nó, calcule todos os ciclos possíveis, com comprimento inferior a ***Comp***, desse nó. *Sugestão: Utilize o predicado caminho (alínea anterior) como base para a resolução.*

Exemplo:

?- ciclos(4, 3, Lista).
Lista = [[4,3,6], [4,6,3]]
?- ciclos(4, 5, Lista).
Lista = [[4,3,6], [4,6,3], [4,2,1,3], [4,3,1,2]]

Solução:

```
% a) caminho(+NoInicio, +NoFim, -Lista),
ligacao2(X,Y):- ligacao(X,Y).
ligacao2(X,Y):- ligacao(Y,X).
```

```
caminho(NoInicio, NoFim, Lista):-
      caminho(NoInicio, NoFim, [NoInicio], Lista, 5).

caminho(NoInicio, NoFim, Lista, ListaFim,_):-
      ligacao2(NoInicio, NoFim),
      append(Lista, [NoFim], ListaFim).
caminho(NoInicio, NoFim, Lista, ListaFim, N):-
      N>0,
      ligacao2(NoInicio, NoInterm),
      NoInterm \= NoFim,
      \+(member(NoInterm, Lista)),
      append(Lista, [NoInterm], Lista2),
      N2 is N-1,
      caminho(NoInterm, NoFim, Lista2, ListaFim, N2).

% Outra versão. Sera' que funciona? Porque?

caminho2(NoIni, NoFim, Lista):- caminho2(NoIni, NoFim, Lista, 5).

caminho2(_,_,_,0):- !,fail.
caminho2(NoIni, NoFim, [NoIni,NoFim], _):- ligacao2(NoIni,NoFim).
caminho2(NoIni, NoFim, [NoIni|Rest], N):-
      N2 is N-1,
      ligacao2(NoIni, NoInt),
      caminho2(NoInt, NoFim, Rest, N2).

% Ainda outra Versão!

caminho3(NoIni, NoFim, Lista):- caminho3(NoIni, NoFim, Lista, 0).

caminho3(NoIni, NoFim, [H|Rest], N):-
      N < 4,
      ligacao2(NoIni, NoInt),
      (NoInt = NoFim, [H|Rest] = [NoIni, NoFim] ;
      H = NoIni, N2 is N+1, caminho3(NoInt, NoFim, Rest, N2)).

b) ciclos(+No, +Comp, -Lista),

ciclos(No, Comp, Lista):-
      findall(Ciclo, caminho(No, No, [], Ciclo, Comp), Lista).
```