ALGAV
Revisões

Enunciado

Considere uma base de conhecimento de abastecimentos diários, realizados numa planta fabril por vários robots, com factos relativos à rede de postos de fabrico, com a distância em metros entre os vários postos, e factos relativos aos abastecimentos realizados por cada robot.

```
% Rede de postos de fabrico: posto(postA, postB, dist) %
(sentido bidirecional)

posto(pt1, pt2, 21).
posto(pt3, pt4, 55).
posto(pt5, pt3, 30).
posto(pt2, pt5, 19).
posto(pt4, pt2, 13).
posto(pt1, pt6, 28).

% Abastecimentos robots: robot(robot, lista_postos_abastecidos)

robot(r1,[pt4,pt5,pt2,pt6,pt5]).
robot(r2,[pt1,pt3,pt6]).
robot(r3,[pt2,pt5,pt3,pt2,pt5,pt6,pt1,pt5]).
```

Enunciado

```
robot(r1,[pt4,pt5,pt2,pt6,pt5]).
```

Na de lista abastecimentos de um robot, os postos surgem por ordem de abastecimento, tal que o último posto fornecido se encontra na cabeça da lista e coincide com a posição atual do robot, por ex., o robot r1 encontra-se no posto pt4.

O mesmo posto pode surgir várias vezes numa lista de abastecimento se tiver sido abastecido pelo mesmo robot várias vezes nesse dia.

Enunciado

Escreva o predicado **abastposto (P, L)** que, para um determinado **posto** (P), gera uma **lista** (L) com o **número de abastecimentos feitos por cada robot a esse posto**.

Exemplo:

?- abastposto(pt5,L).

L = [(r1,2), (r2,0), (r3,3)].

Resolução

```
% Soluçao 1

abastposto(P,L):-
   listaRobots(LR),
   conta_postos(P,LR,L).

conta_postos(_,[], []).
conta_postos(P,[H|T], [(H,N)|L]):-
   robot(H,LP),
   conta(P,LP,N),
   conta_postos(P,T,L).

listaRobots(L):- findall(R,robot(R,_),L).

conta(_,[],0):-!.
conta(P, [H|T], N):-
   conta(P, T, N1), (P == H, N is N1 + 1; N is N1).
```

```
robot(r1,[pt4,pt5,pt2,pt6,pt5]).
```

L = [(r1,2), (r2,0), (r3,3)].

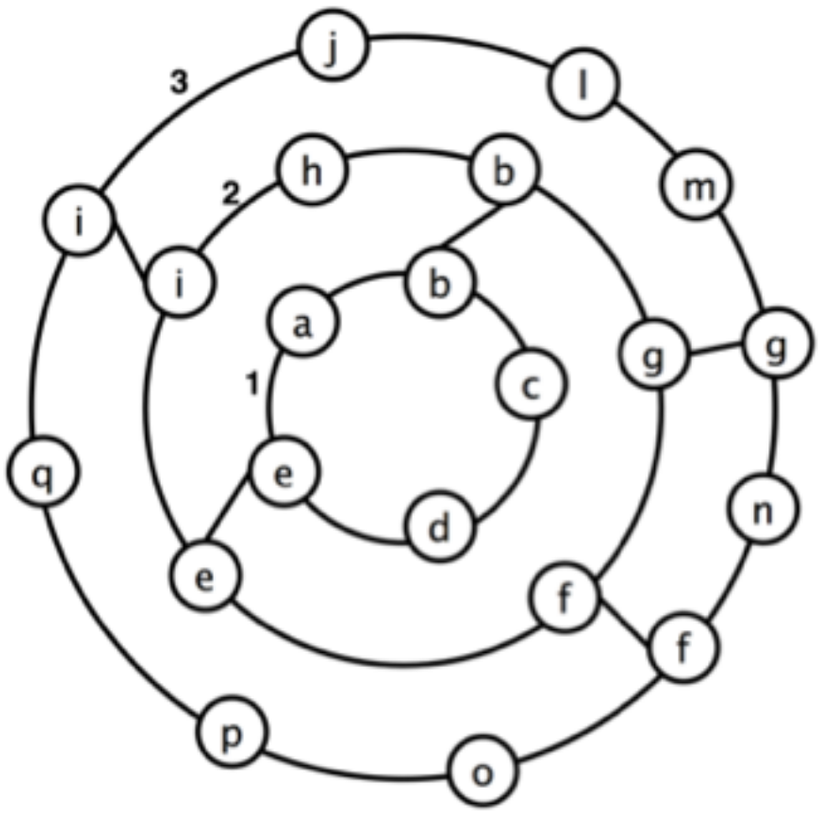
Resolução

```
robot(r1,[pt4,pt5,pt2,pt6,pt5]).

% Soluçao 2

abastposto2(P,L):-
  findall((R,N), (robot(R,LA),
        findall(P,member(P,LA), LP),
                  length(LP,N)), L).
```

Enunciado



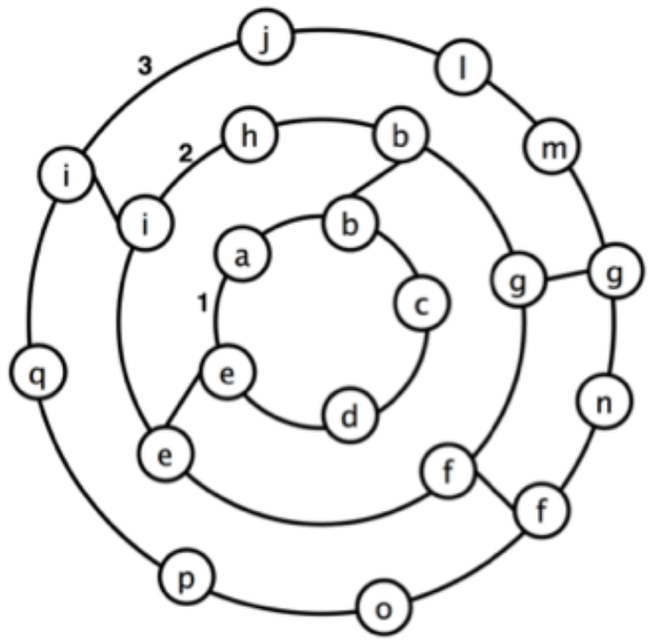
Considere uma rede de linhas de metropolitano organizada através de uma estrutura radial. Algumas estações permitem ligar diferentes linhas. Por ex., a estação **b** permite efetuar a mudança entre a linha **1** e a linha **2**.

A rede de metropolitano é representada por factos **linha(Id,LE)**, em que **Id** é a identificação da linha e **LE** uma lista contendo as estações que integram essa linha. A rede representada na figura é definida pelo seguinte conjunto de factos:

```
linha(1,[a,b,c,d,e]).
linha(2,[b,g,f,e,i,h]).
linha(3,[l,m,g,n,f,o,p,q,i,j]).
```

Enunciado

1. Defina o predicado ***mesma_linha/2*** que deverá ter sucesso no caso dos argumentos estarem instanciados com a identificação de duas estações que pertençam à mesma linha.
   Exemplo:
   `?- mesma_linha(n,j). Yes`




```
mesma_linha(X,Y):-
  linha(_,L), member(X,L), member(Y,L).
```

Enunciado

2. [35%] Defina o predicado ***muda_linha/3*** que verifica se uma estação, identificada através do 1º argumento, permite mudar de linha, garantindo que a nova linha não foi já usada no trajeto e devolvendo no 3º argumento a identificação da nova linha. O 2º argumento recebe uma lista com a identificação das linhas já usadas no percurso.

   Exemplo:

```
?- muda_linha(g,[1,2],NovaLinha).
NovaLinha= 3
```

```
linha(1,[a,b,c,d,e]).
```

```
muda_linha(Est,LE,NL):-
   linha(NL, Lin),
   member(Est, Lin),
   \+ member(NL,LE),
   linha(NL1, Lin1),
   NL1 \= NL,
   member(Est, Lin1).
```

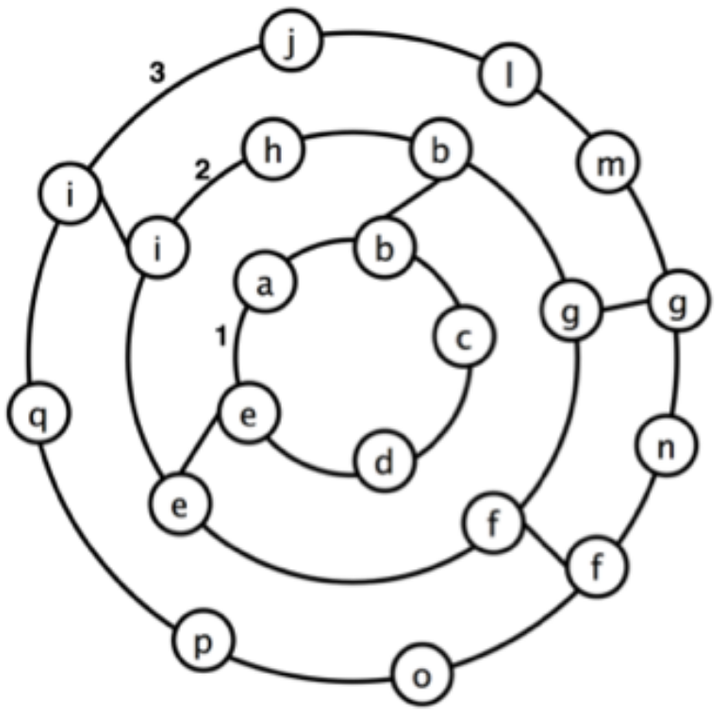
```
muda_linha1(Est,LE,NL):-
   linha(NL, Lin),
   linha(NL1, Lin1),
   NL \= NL1,
   \+ member(NL, LE),
   intersection(Lin, Lin1,LI),
   member(Est,LI).
```

Enunciado

3. [45%] Defina o predicado ***go/3*** que receba nos dois primeiros argumentos as estações de partida e de destino e que devolva através do 3º argumento uma lista de estações que definam um trajeto possível entre as estações de partida e de destino.

Exemplo:

```
?- go(a,p,LE).
LE = [a, b, g, p] ; …
```




```
go(O,D,LE):-
   encontra_linha(O,L),
   go1(O,D,[L],LE).
go1(O,D,_,[O,D]):-
   mesma_linha(O,D).
go1(O,D,LV,[O|LE]):-
   muda_linha(X,LV,NL),
   mesma_linha(O,X),
   \+ member(NL, LV),
   go1(X,D,[NL|LV],LE).

encontra_linha(Est, Linha):-
   linha(Linha,LE), member(Est,LE).
```