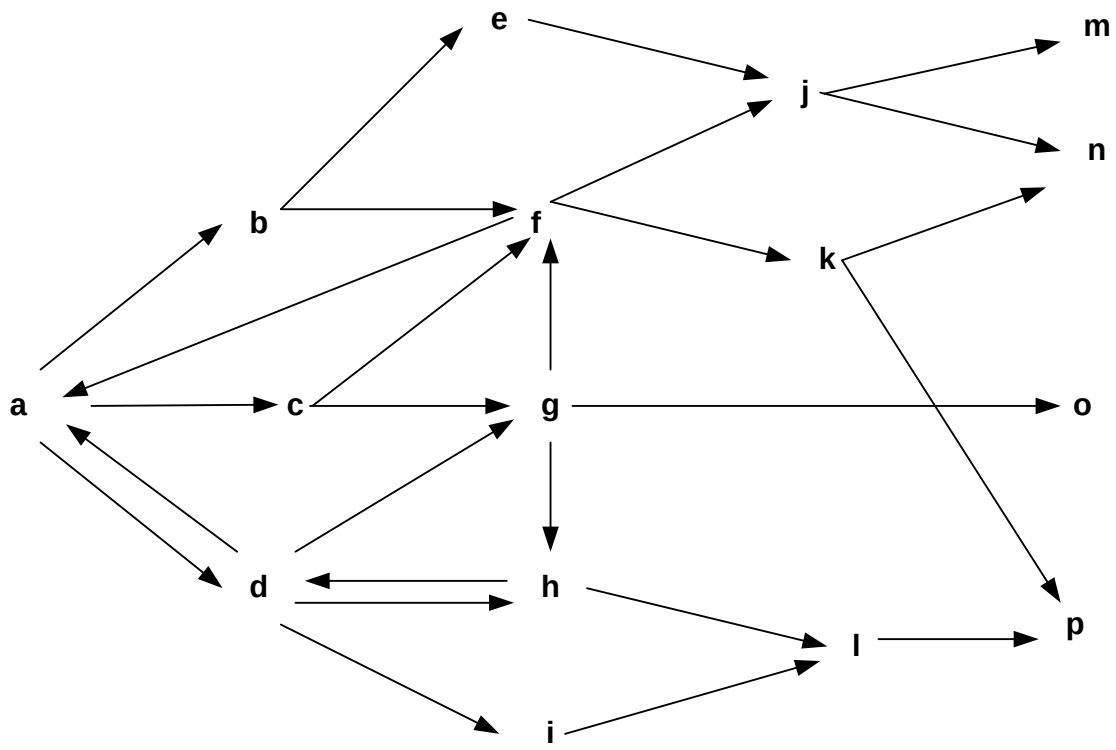
Ficha de Problemas da disciplina de
Algoritmia Avançada
do 3º ano da
Licenciatura em Engenharia Informática da
Instituto Superior de Engenharia do Porto

Ficha nº 6 – Listas e Pesquisa em Grafos

1) Considere o seguinte grafo onde podem ocorrer "ciclos" nos seus caminhos (ou seja, uma vez que se sai de um nó podemos voltar a esse nó).



a) Escreva um programa em PROLOG que dado um nó origem e um nó destino escreva o caminho correspondente. Expresse o conhecimento com factos do tipo *liga(a,b).*

b) Escreva um predicado que seja capaz de criar uma lista com todos os caminhos possíveis entre um nó origem e um nó destino. Cada caminho também está representado através de uma lista

c) Escreva um predicado que seja capaz de encontrar a solução passando pelo menor número de cidades.

2) Considere que uma rede de metro é representada por factos que correspondem às linhas existentes, com factos do tipo *linha(nº_linha, lista_estações)* como, por exemplo, a base de conhecimento que se segue:

```
linha(1,[a,b,c,d,e,f]).
linha(2,[g,b,h,i,j,k]).
linha(3,[l,j,m,n,o,d,p]).
linha(4,[f,q,r,s,t]).
linha(5,[t,u,j,v,a,g]).
```

Considere, ainda, que um programador escreveu os predicados *gera_cruzamentos* e *cruzamento*, que em conjunto visariam gerar factos do tipo *cruza(nº_linhaA ,nº_linhaB, lista_estações_cruzamento)* do modo que se segue:

```
gera_cruzamentos:-findall(_,cruzamento,_).

cruzamento:-    linha(N1,LE1),
                linha(N2,LE2),
                intersecção(LE1,LE2,LI),
                assertz(cruza(N1,N2,LI)).
```

O predicado intersecção(LE1,LE2,LI) corresponde ao dado nas aulas. Os factos do tipo *cruza/3* foram declarados como sendo dinâmicos, ou seja, podem ser criados e removidos durante a execução.

Posteriormente, o programador testa o programa escrito, procurando ver qual é a lista LE das estações que estabelecem o cruzamento entre a linha LA e a linha LB, do seguinte modo:

```
| ?- gera_cruzamentos.
yes

| ?- cruza(LA,LB,LE).

LA = LB = 1, LE = [a,b,c,d,e,f] ;
LA = 1 ,LB = 2 ,LE = [b] ;
LA = 1 ,LB = 3 ,LE = [d] ;
LA = 1 ,LB = 4 ,LE = [f] ;
LA = 1 ,LB = 5 ,LE = [a] ;
LA = 2 ,LB = 1 ,LE = [b] ;
LA = LB = 2 , LE = [g,b,h,i,j,k] ;
LA = 2 ,LB = 3 ,LE = [j] ;
LA = 2 ,LB = 4 ,LE = [] ;
LA = 2 ,LB = 5 ,LE = [g,j] ;
LA = 3 ,LB = 1 ,LE = [d] ;
LA = 3 ,LB = 2 ,LE = [j] ;
LA = LB = 3 ,LE = [l,j,m,n,o,d,p] ;
LA = 3 ,LB = 4 ,LE = [] ;
```

LA = 3 ,LB = 5 ,LE = [j] ;
LA = 4 ,LB = 1 ,LE = [f] ;
LA = 4 ,LB = 2 ,LE = [] ;
LA = 4 ,LB = 3 ,LE = [] ;
LA = LB = 4 ,LE = [f,q,r,s,t] ;
LA = 4 ,LB = 5 ,LE = [t] ;
LA = 5 ,LB = 1 ,LE = [a] ;
LA = 5 ,LB = 2 ,LE = [j,g] ;
LA = 5 ,LB = 3 ,LE = [j] ;
LA = 5 ,LB = 4 ,LE = [t] ;
LA = LB = 5 ,LE = [t,u,j,v,a,g]

Para além das soluções pretendidas pelo programador surgem outras soluções (sublinhadas) que não fazem sentido.

a) Corrija o predicado *cruzamento* de modo que as soluções sem sentido não apareçam.

b) Escreva um predicado *gera_estações* que seja capaz de criar um facto do tipo *estações(lista_todas_estações)* onde na lista não apareçam estações repetidas.

| ?-gera_estações.
yes

| ?- estações(LE).
LE = [c,e,b,h,i,k,l,m,n,o,d,p,f,q,r,s,t,u,j,v,a,g]

c) Escreva um predicado gera_estações_linhas que seja capaz de criar factos do tipo

***estação_linhas(estação,lista_de_todas_linhas_que_passam_pela_estação)*.**

| ?- gera_estações_linhas.
yes

| ?- estação_linhas(E,L).

E = c ,L = [1] ;	**E = n ,L = [3] ;**	**E = t ,L = [4,5] ;**
E = e ,L = [1] ;	**E = o ,L = [3] ;**	**E = u ,L = [5] ;**
E = b ,L = [1,2] ;	**E = d ,L = [1,3] ;**	**E = j ,L = [2,3,5] ;**
E = h ,L = [2] ;	**E = p ,L = [3] ;**	**E = v ,L = [5] ;**
E = i ,L = [2] ;	**E = f ,L = [1,4] ;**	**E = a ,L = [1,5] ;**
E = k ,L = [2] ;	**E = q ,L = [4] ;**	**E = g ,L = [2,5]**
E = l ,L = [3] ;	**E = r ,L = [4] ;**	
E = m ,L = [3] ;	**E = s ,L = [4] ;**	

d) Escreva um predicado *gera_caminho(Eorigem, Edestino, Ltroços)* que dada uma estação de origem e uma estação de destino seja capaz de gerar uma lista com troços do tipo *(estação1,estação2,linha)*. Admita que os factos do tipo *cruza/3, estação_linhas/2* e *estações/1* já estão criados.

Exemplo:

| ?- gera_caminho(q,h,Ltroços).

Ltroços = [(q,f,4),(f,b,1),(b,h,2)]

e) Considerando que o predicado *gera_caminho(Eorigem, Edestino, Ltroços)*, definido em d), já está implementado, escreva um outro predicado *menos_trocas_linha(Eorigem, Edestino, L)* que forneça uma solução *L* com o menor número de trocas de linhas.