<u>Primeiro</u> Trabalho de Técnicas Alternativas de Programação (Prof. Alexandre Direne - 2010/2)

<u>Atenção:</u>

Este trabalho é obrigatório e deverá ser entregue, impreterivelmente, até o dia 06 de outubro de 2010 (quarta-feira). A solução é individual e deverá ser arquivada no diretório "~alexd/TAP/" onde o nome do arquivo terá como prefixo o seu nome-de-usuário no sistema do laboratório e, como extensão, ".pl" para indicar que seu conteúdo possui um programa em Prolog. Assim, por exemplo, se o seu nome de usuário no sistema fosse "grs00" então o nome o arquivo seria "grs00.pl" (dentro do diretório "~alexd/TAP/"). Não se esqueça de proteger completamente o arquivo criado, de maneira a a impedir o acesso a ele por parte de outros! Isso pode ser feito aplicando `chmod og-rwx grs00.pl` antes de efetuar a cópia com a preservação das permissões (`cp -p grs00.pl ~alexd/TAP/`). Não se preocupe com as permissões do professor que irá corrigir o trabalho. A correção dos trabalhos será parcialmente automatizada, sendo assim, é importante que todos os arquivos com as soluções individuais estejam no diretório citado acima, dentro do prazo estipulado. Não será permitida a entrega do arquivo por e-mail.

<u>Enunciado:</u>

O conceito de Ordenação Topológica (*Topological Sorting*) de um grafo direcionado e acíclico (DAG – Directed Acyclic Graph) é associado ao processo de obtenção de conjuntos ordenados a partir dos nodos desse grafo. O critério de ordem faz com que cada nodo deve vir antes de todos os outros para os quais ele tem aresta de saída. Cada grafo direcionado e acíclico tem ao menos uma ordem topologica. A Figura 1 mostra o exemplo de um grafo que possui 150 ordens topológicas diferentes. Algumas delas são:

```
h, f, e, g, d, c, a, b
h, f, e, g, d, c, b, a
h, f, e, g, d, a, c, b
h, f, e, g, d, a, b, c
h, f, e, g, d, b, c, a
h, f, e, g, d, b, a, c
   ...
f, h, e, g, d, c, a, b
f, h, e, g, d, c, b, a
f, h, e, g, d, a, c, b
f, h, e, g, d, a, b, c
f, h, e, g, d, b, c, a
f, h, e, g, d, b, a, c
   ...
g, h, d, f, e, c, a, b
g, h, d, f, e, c, b, a
g, h, d, f, e, a, c, b
g, h, d, f, e, a, b, c
g, h, d, f, e, b, c, a
g, h, d, f, e, b, a, c
   ...
g, f, h, d, e, b, c, a
g, f, h, d, e, b, a, c
```

Implementar um predicado binário em Prolog chamado "`total_de_ordens`" que é responsável por calcular a quantidade de ordens topológicas diferentes de um grafo direcionado e acíclico, e instanciar essa quantidade em uma variável.

Para ilustrar uma aplicação do predicado "`total_de_ordens`", a consulta a seguir mostra o cômputo da quantidade de ordens topológicas para o grafo da Figura 1.

```
?- total_de_ordens([[h,e],[h,d],[f,e],[g,d],[g,c],[e,a],[e,b],[e,c],[d,b]], T).
T = 150 ?
yes
?- total_de_ordens([[a,b],[b,c],[c,d],[d,e]], T).
T = 1 ?
yes
?- total_de_ordens([[a,b],[a,c],[a,d],[a,e],[a,f],[a,g],[a,h]], T).
T = 5040 ?
yes
```

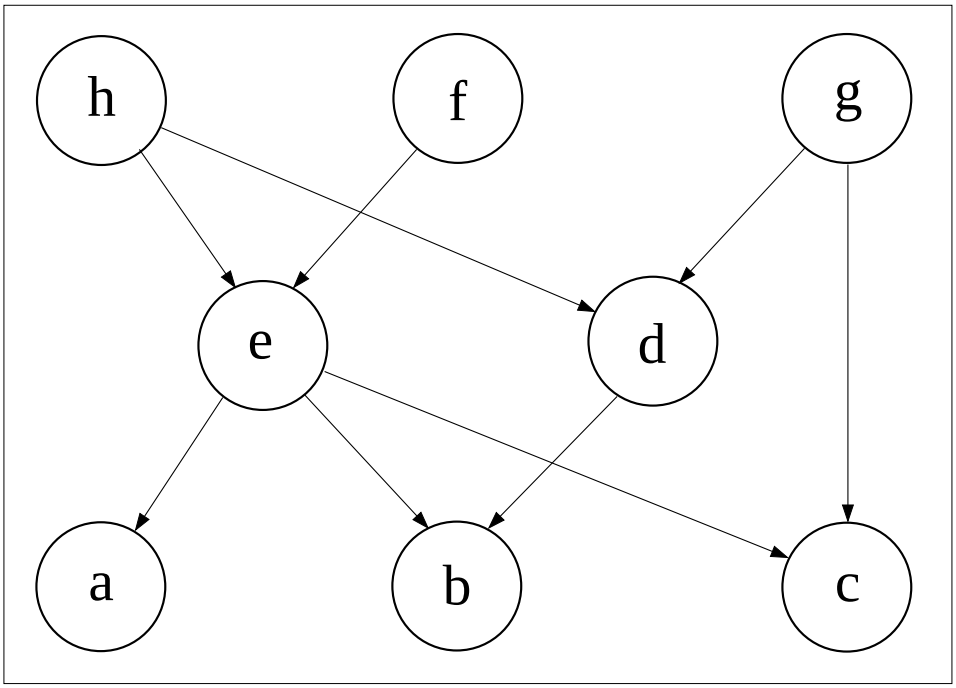

Figura 1: Grafo direcionado e acíclico.

Duas observações importantes cabem no projeto inicial do predicado:

1. a cobertura das ordens deve ser completa para que a contagem seja precisa, o que exigirá um algoritmo de força bruta de complexidade n^n, sendo n a quantidade de vértices do grafo (dica: não use algoritmos clássicos de cálculo da Ordenação Topológica pois eles quase sempre fornecem apenas uma das ordens);
2. projete os predicados principais para serem também retroativos (e não puramente recursivos) pois pode faltar memória para as soluções puramente recursivas, mesmo com $n = 8$;
3. tome cuidado com o tempo de execução pois ele pode ser bem longo, mesmo para programas que estão corretos mas que não levam em consideração fatores mínimos de melhoria de desempenho.

Para servir como apoio inicial, os fatos em Prolog abaixo são uma boa representação do grafo da Figura 1 como base de um cálculo retroativo (use o predicado `assertz` para gerar esses fatos a partir do primeiro termo do predicado `total_de_ordens`).

```
vertice(h).
vertice(e).
vertice(d).
vertice(f).
vertice(g).
vertice(c).
vertice(a).
vertice(b).
aresta(h,e).
aresta(h,d).
aresta(f,e).
aresta(g,d).
aresta(g,c).
aresta(e,a).
aresta(e,b).
aresta(e,c).
aresta(d,b).
```

Obsevações finais:

- Indique, por meio de um comentário de código, qual foi o compilador Prolog que você utilizou (e.g., SWI-Prolog, Poplog-Prolog, GNU-Prolog, etc);
- Não troque o nome do predicado `total_de_ordens` pois isto dificultará a correção do trabalho.