SUJET D'INTELLIGENCE ARTIFICIELLE
PARCOURS DE GRAPHES
CORRIGE

Date : 30 Novembre 1996 Auteur:C.JOUSSELIN

SUJET D'INTELLIGENCE ARTIFICIELLE

PARCOURS DE GRAPHES
CORRIGE
3ème Année SIET CENTRALE 96/97

Introduction

Le but de cet exercice est de vous faire comprendre des mécanismes de parcours de graphes[1].

Sujet

Le sujet consiste à réaliser en PROLOG (Xilog de Bull) un ensemble de prédicats de recherche de chemin dans un graphe et d'arbre de recouvrement.

Définitions

Les définitions et représentations données ci-après n'ont pas pour but d'optimiser les structures de données ni de minimiser la complexité des algorithmes de recherche, mais sont introduites de cette manière pour des raisons pédagogiques.

Représentation d'un graphe

Nous représenterons un graphe G par
G = *graphe*(Sommets, Arcs)
où *graphe* est le foncteur de description du graphe G

Sommets l'ensemble (la liste) de sommets du graphe
Sommets = [a, b, c ...]
Arcs l'ensemble (la liste) d'arcs orientés valués
Arcs = [*arc*(a,b,2), *arc*(b,c,1) ...]
où *arc* est le foncteur de description d'*arc orienté* valué.

Nota : La valeur d'un arc est un nombre associé à cet arc:
valeur(arc(a,b,2)) = 2
Pour ne pas introduire un foncteur supplémentaire, nous appellerons *arc non orienté* entre a et b la réunion des arcs orientés: arc(a,b,x) et arc(b,a,x).

Chemin

Un chemin orienté (respectivement non orienté) est un sous-ensemble (liste) de Sommets, tel qu'il existe un arc orienté (respectivement non orienté) mettant en relation, deux à deux et de proche en proche, les sommets du chemin (sommets adjacents).
Nous ne nous intéresserons qu'aux chemins acycliques (un sommet ne figure qu'une seule fois dans un chemin).

Nota : La valeur d'un chemin est la somme des valeurs des arcs associés à ce chemin.

Chemin(Point1,Point2,Graphe,Chemin,CoûtChemin,Orientation)
où Orientation vaut "orienté" ou "non orienté" est le prédicat de recherche des chemins reliant le point1 au point2 dans le graphe.

1 *"Si R est une relation d'un ensemble E vers un ensemble F, on appelle graphe de R l'ensemble des couples (x,y) (x élément de E et y élément de F) qui vérifient la relation R."*

```
chemin(P1,P2,G,Ch,C,T) :- chemin1(P1,[P2],G,Ch,C,0,T).

chemin1(P1,[P1|T],G,[P1|T],C,C,_).

chemin1(P1,[P3|T],G,Ch,C,C3,O) :- adjacent(P2,P3,G,C2,O),
        not member(P2,T),C4 is C2 + C3,         chemin1(P1,
        [P2,P3|T],G,Ch,C,C4,O).

adjacent(P1,P2,graphe(_,A),C,"orienté") :-
        member(arc(P1,P2,C),A).

adjacent(P1,P2,graphe(_,A),C,"non orienté") :-
        (member(arc(P1,P2,C),A) ; member(arc(P2,P1,C),A)).

member(X,[X|_]).

member(X,[_|T]) :- member(X,T).
```

Graphe connexe

Un graphe est dit connexe s'il existe pour tout sommet un chemin non orienté conduisant à un quelconque autre sommet du graphe.

```
connexe(G) :- not (sommet(P1,G) , sommet(P2,G) ,
         not chemin(P1,P2,G,_,_,"non orienté")).


sommet(P,G) :- adjacent(P,_,G,_,"non orienté").
```

Chemin hamiltonien

Un chemin orienté C est dit hamiltonien pour le graphe G s'il est un chemin acyclique couvrant tous les sommets du graphe G.
Hamiltonien(Graphe,Chemin,CoûtChemin) est le prédicat de recherche des chemins hamiltoniens dans le graphe.

```
hamiltonien(G,Ch,C) :- chemin(_,_,G,Ch,C,"orienté"),
        couvre(G,Ch).

couvre(G,[C|Tc]) :- not var(C),
        not (sommet(P,G) , not member(P,[C|Tc])).
```

Arbre simple héritage

Un graphe est un arbre à simple héritage s'il est orienté, connexe, non cyclique et si aucun sommet du graphe n'a plus d'un père.

```
arbre(Arbre) :- connexe(Arbre),
        not cyclique(Arbre,"non orienté").

cyclique(G,O) :- adjacent(P1,P2,G,_,O),     chemin(P1,P2,G,
        [P1,P3,P4|_],_,O),!.

héritage_simple(Graphe) :-
        not (adjacent(P1,P2,Graphe,_,"orienté") ,
        adjacent(P3,P2,Graphe,_,"orienté") , not P1 == P3).
```

Arbre de recouvrement

Un arbre de recouvrement T pour le graphe G=graphe(S,A) est un arbre à simple héritage T=graphe(S,B) où B est un sous-ensemble de A.
Arbre_de_recouvre(Graphe,Arbre) est le prédicat de recherche des arbres de recouvrement du graphe.

```
arbre_recouvre(graphe(Sg,Aa),graphe(Sg,Ag)) :-
        inclu(Aa,Ag),héritage_simple(graphe(Sg,Aa)),arbre(g
        raphe(Sg,Aa)),not (sommet(P,graphe(Sg,Ag)),
        not sommet(P,graphe(Sg,Aa))).

inclu([],[]) :- !.

inclu(I,[A|T]) :- inclu(T1,T),(I = T1 ; I = [A|T1]).
```

Exemple :

```
graphe([a,b,c,d,e],
    [arc(a,c,1),arc(c,e,1),arc(e,d,1),arc(d,b,1)
    ,arc(d,c,1),arc(c,b,1),arc(b,e,1),arc(b,a,1)])
```

Chemins hamiltoniens:

CH_Hamil = [e,d,b,a,c] Cout = 4

CH_Hamil = [d,b,a,c,e] Cout = 4

CH_Hamil = [b,a,c,e,d] Cout = 4

CH_Hamil = [a,c,b,e,d] Cout = 4

CH_Hamil = [a,c,e,d,b] Cout = 4

CH_Hamil = [c,e,d,b,a] Cout = 4

CH_Hamil = [e,d,c,b,a] Cout = 4

Arbres de recouvrements:

```
Arbre=graphe([a,b,c,d,e],[arc(a,c,1),arc(c,e,1),arc(e,d,1),arc(d,b,1)])
Arbre=graphe([a,b,c,d,e],[arc(a,c,1),arc(c,e,1),arc(e,d,1),arc(c,b,1)])
Arbre=graphe([a,b,c,d,e],[arc(a,c,1),arc(e,d,1),arc(c,b,1),arc(b,e,1)])
Arbre=graphe([a,b,c,d,e],[arc(a,c,1),arc(c,e,1),arc(e,d,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(a,c,1),arc(c,e,1),arc(d,b,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(a,c,1),arc(e,d,1),arc(d,b,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(c,e,1),arc(e,d,1),arc(d,b,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(c,e,1),arc(d,b,1),arc(d,c,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(e,d,1),arc(d,b,1),arc(d,c,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(c,e,1),arc(e,d,1),arc(c,b,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(c,e,1),arc(d,c,1),arc(c,b,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(e,d,1),arc(d,c,1),arc(c,b,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(a,c,1),arc(e,d,1),arc(b,e,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(a,c,1),arc(d,b,1),arc(b,e,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(e,d,1),arc(d,c,1),arc(b,e,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(d,b,1),arc(d,c,1),arc(b,e,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(e,d,1),arc(c,b,1),arc(b,e,1),arc(b,a,1)])
Arbre=graphe([a,b,c,d,e],[arc(d,c,1),arc(c,b,1),arc(b,e,1),arc(b,a,1)])
```