Решения на задачите от контролно 2 по Логическо програмиране

12 януари 2019

1 Първа задача на пролог

Нека $G = \langle V, E \rangle$ е краен неориентиран граф без примки.

I.1 *k-клика в G*, където $k > 2$, наричаме такова k-елементно подмножество W на V, че винаги, когато v_1 и v_2 са различни върхове на W, има ребро в G, което е с краища v_1 и v_2.

I.2 *k-антиклика в G*, където $k > 2$, наричаме такова k-елементно подмножество W на V, че винаги, когато v_1 и v_2 са различни върхове на W, няма ребро в G, което е с краища v_1 и v_2.
За един списък X от двуелементни списъци казваме, че *представлява* G, ако са изпълнени следните условия:

- ако $v_1 \neq v_2$ и G има ребро с краища v_1 и v_2 , то поне един от списъците $[v_1,\ v_2]$, $[v_2,\ v_1]$ е от X;
- ако $[v_1,\ v_2]$ е от X и $v_1 \neq v_2$, то в G има ребро с краища v_1 и v_2;
- $[v,\ v]$ е от X точно тогава, когато v е връх в G, който не е край на ребро от G.

I.1 Да се дефинира на пролог двуместен предикат *cl(K, X)*, който по дадени естествено число K и списък X, представящ граф G, разпознава дали G има *K-клика.*

I.2 Да се дефинира на пролог двуместен предикат *acl(K, X)*, който по дадени естествено число K и списък X, представящ граф G, разпознава дали G има *K-антиклика.*

1.1 Общи предикати

```
length([], 0).
length([_|T], N):- length(T, M), N is M + 1.

append([], L2, L2).
append([H|T], L2, [H|R]):- append(T, L2, R).

member(X, L):- append(_, [X|_], L).

subset([], []).
subset([_|T], R):- subset(T, R).
subset([H|T], [H|R]):- subset(T, R).
```

```
addVertice(V, VL, VR):- not(member(V, VL)),
    append([V], VL, VR).
addVertice(V, VL, VL):- member(V, VL).

extractVertices([], []).
extractVertices([[X, Y]|T], V):- extractVertices(T, TV),
    addVertice(X, TV, TX),
    addVertice(Y, TX, V).
```

1.2 Примерно решение на I.1

```
cl(K, X):- extractVertices(X, V),
    subset(S, V),
    length(S, K),
    isClique(S, X).

isClique(S, X):-
    not((member(A, S), member(B, S),
    A \= B, not(member([A, B], X); member([B, A], X) ) )).
```

1.3 Примерно решение на I.2

```
acl(K, X):- extractVertices(X, V),
    subset(S, V),
    length(S, K),
    isAnticlique(S, X).

isAnticlique(S, X):-
    not((member(A, S), member(B, S),
    A \= B, not(((member([A, B], X); member([B, A], X) ))) )).
```

2 Втора задача на пролог

I.1 Редиците на Фарей, F_n, са редици от двойки естествени числа, които се дефинират рекурсивно за $n \geq 1$ по следния начин:

- $F_1 = \langle(0,1),(1,1)\rangle$;
- F_{n+1} се получава от F_n, като между всеки два последователни члена (a,b) и (c,d) на F_n, за които $b+d = n+1$, се добавя двойката $(a+c, b+d)$.

Да се дефинира на пролог едноместен предикат *farey(F)*, който при преудовлетворяване генерира в F всички редици на Фарей.

I.2 Дървото на Раней, R, е пълно двоично дърво, във върховете на което така са поставени двойките естествени числа, че:

- в корена на R е поставена двойката $(1,1)$;
- ако в един връх v на R е поставена двойката (a,b), то в левия наследник на v е поставена двойката $(a, a+b)$, а в десния - $(a+b, b)$.

Да се дефинира на пролог едноместен предикат *raney(L)*, който при преудовлетворяване генерира в L всички етажи от дървото на Раней.

2.1 Примерно решение на I.1

```
farey(F):- farey(F, _).
farey([[0, 1], [1, 1]], 1).
farey(L, N):- farey(L1, N1), N is N1 + 1, addPairFarey(L1, N, L).

addPairFarey([], _, []).
addPairFarey([X], _, [X]).
addPairFarey([[A, B], [C, D]|T], N, [[A, B], [E, N]|R]):-
    addPairFarey([[C, D]|T], N, R),
    B + D =:= N,
    E is A + C.
addPairFarey([[A, B], [C, D]|T], N, [[A, B]|R]):-
    addPairFarey([[C, D]|T], N, R),
    B + D =\= N.
```

2.2 Примерно решение на I.2

```
raney([[1, 1]]).
raney(L):- raney(L1), addPairRaney(L1, L).

addPairRaney([], []).
addPairRaney([[A, B]|T], [[A, AB], [AB, B]|R]):-
    addPairRaney(T, R),
    AB is A + B.
```