Programming Language HW3 Report – Prolog

資訊109 F74054067 柳孟芸

▷執行環境 ：

VS Code + Terminal

▷執行方法 ：

◎Problem 1

```
$ swipl -q -s problem_1.pl
```

◎Problem 2

```
$ swipl -q -s problem_2.pl
```

◎Problem 3

```
$ swipl -q -s problem_3.pl
```

▷程式碼解說 ：

◎Problem 1 : Goldbach's conjecture

problem_1.pl × problem_2.pl problem_3.pl

```
% problem 1 : Goldbachs conjecture
% Every even integer greater than 2 can be expressed as the sum of two primes.

isPrime(N) :-
    N > 1,
    isPrime(N, 2).

isPrime(N, P) :-
    P =< sqrt(N),
    N mod P =\= 0,
    isPrime(N, P+1).

isPrime(N, P) :-
    P > sqrt(N).

goldbach(Num) :-
    Num > 2,
    Num mod 2 =:= 0,
    NNum1 is 2,
    NNum2 is Num-2,
    goldbach(Num, NNum1, NNum2).

goldbach(Num, Num1, Num2) :-
    isPrime(Num1),
    isPrime(Num2),
    writef('\t'),
    write(Num1),
    write(' '),
    write(Num2),
    nl,
    NNum1 is Num1+1,
    NNum2 is Num2-1,
    goldbach(Num, NNum1, NNum2).

goldbach(Num, Num1, Num2) :-
    Num1 < Num/2,
    NNum1 is Num1+1,
    NNum2 is Num2-1,
    goldbach(Num, NNum1, NNum2).

goldbach(Num, Num1, _) :-
    Num1 >= Num/2.

main :-
    % Input
    write('Input: '),
    readln(Num),
    % Output: summands
    write('Output:'),
    goldbach(Num),
    halt.

:-initialization(main).
```

1. isPrime(N):檢查 N 是否為prime.
-> N 要大於 1
-> N is divisible only by 1 and itself.
2. 檢查 P from 2 to sqrt(N).

1. goldbach():Find Num1 and Num2.
-> 檢查 Num 是否 > 2
-> 檢查 Num 是否為 even
-> Find (Num1, Num2) from (2, Num-2) to (Num/2, Num-Num/2)
-> Check if Num1 and Num2 are both primes

2. 概念：一個數字一個數字找，並利用isPrime函數來判斷一個數字是不是質數，如果不是質數就繼續找下一個，找到是兩個質數相加就write出來，直到找完全部可能的結果。

1. 提供使用者輸入一個 > 2 的 even integer input，並且印出結果。

2. 程式最後用halt.結束。

◎**Problem 2 :** Lowest Common Ancestor

根據Input的關係建立好node的關係，並將這些關係作為fact吃進來。吃進來之後再利用這些parent關係，來判斷是不是ancestor，並且找出lowest的共同祖先。如果想要找關係的兩個node是同一個，就直接給這一個node作為lowest common ancestor。程式最後用halt.結束。

problem_2.pl

```
% problem 2 : Lowest Common Ancestor

ancestor(A,B) :-
    parent(A,B). %if A is B_parent then A is B_ancestor
ancestor(A,B) :-
    parent(X,B),ancestor(A,X). %if X is B_parent and A is X_ancestor then A is B_ancesor

lca(A,B,P) :-
  A==B -> assert(output(P,A));
  ancestor(A,B) -> assert(output(P,A));
  parent(X,A),lca(X,B,P).

lcaloop(P) :-
    P > 0,
    Q is P-1,
    readln([W,Z]),
    lca(W,Z,P),
    lcaloop(Q).
lcaloop(0).

outputloop(R) :-
    R > 0,
    output(R,S),
    write(S), nl,
    T is R - 1,
    outputloop(T),
    !.
outputloop(0).

createNode(N) :-
    N > 0,
    M is N - 1,
    readln([C,D]),
    assert(parent(C,D)),

    createNode(M),
    !.

createNode(0).

main :-
    write('Input : '),
    nl,
    readln([N]),
    M is N - 1,
    createNode(M),
    readln([O]),
    lcaloop(O),
    write('Output :'),
    nl,
    outputloop(O),
    halt.


:- initialization(main).

```

◎**Problem 3 :** Reachable

根據Input來建立edge。在建立edge的時候，就直接建立兩個同一條邊不同方向的兩個edge。在判斷有沒有路徑可以到達目標的node的時候，多用一個visited的list來存有走過的node，這樣就可以避免cycle的循環。最後利用有沒有路徑存在，來判斷是不是reachable。程式最後用halt.結束。

problem_3.pl

```
% problem 3 : Reachable

path(A,B) :-
    walk(A,B,[]).

walk(A,B,Visited) :-
    edge(A,X),
    not(memberchk(X,Visited)),
    (   B = X;
        walk(X,B,[A|Visited]) ).

reachable(G,H,P) :-
    path(G,H) -> assert(output(P, 'Yes'));
    assert(output(P, 'No')).

loop(P) :-
    P>0,
    Q is P-1,
    readln([W,Z]),
    reachable(W,Z,P),
    loop(Q).
loop(0).

outputloop(R) :-
    R > 0,
    output(R,S),
    write(S),
    nl,
    T is R - 1,
    outputloop(T),
    !.
outputloop(0).

createEdge(N) :-
    N > 0,
    M is N - 1,
    readln([C,D]),
    assert(edge(C,D)),
    assert(edge(D,C)),
    createEdge(M),
    !.

createEdge(0).

main :-
    write('Input :'),
    nl,
    readln([_,E]),
    createEdge(E),
    readln([O]),
    loop(O),
    write('Output :'),
    nl,
    outputloop(O),
    halt.

:- initialization(main).

```

▷**執行結果：**

◎**Problem 1**

當Input分別為4與100的結果。

```
liumengyunde-MacBook-Pro:hw3 newmileou$ swipl -q -s problem_1.pl
Input: 4
Output: 2 2
liumengyunde-MacBook-Pro:hw3 newmileou$ swipl -q -s problem_1.pl
Input: 100
Output: 3 97
        11 89
        17 83
        29 71
        41 59
        47 53
```

◎**Problem 2**

```
liumengyunde-MacBook-Pro:hw3 newmileou$ swipl -q -s problem_2.pl
Input :
|: 6
|: 1 2
|: 2 3
|: 1 4
|: 4 5
|: 4 6
|: 3
|: 3 4
|: 5 6
|: 1 2
Output :
1
4
1
```

◎**Problem 3**

```
liumengyunde-MacBook-Pro:hw3 newmileou$ swipl -q -s problem_3.pl
Input :
|: 6 6
|: 1 2
|: 2 3
|: 3 1
|: 4 5
|: 5 6
|: 6 4
|: 2
|: 1 3
|: 1 5
Output :
Yes
No
```