ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Bài Toán: Ba thầy tu ba con quỷ
Môn Học: Nguyên Lý và Phương Pháp Lập Trình

Giáo viên hướng dẫn: thầy Nguyễn Tuấn Đăng
Sinh viên thực hiện: Nguyễn Hoàng Trung
MSSV 17521176

1. Bài Toán

Có 3 thầy tu và 3 con quỷ. Làm sao đưa cả 6 qua sông. Biết rằng có 1 chiếc thuyền, thuyền chở 1 lần nhiều nhất là 2 người, ít nhất là 1 người. Chú ý rằng số thầy tu ở mỗi bờ không ít hơn số quỷ, nếu không quỷ sẽ ăn thịt thầy tu.

2. Code Prolog

```
transition(X,Y):- operation(X,Y).

%A,B: So su, so quy o ben bo trai
%C,D: So su, so quy o ben bo phai
%0->1: di tu trai sang phai.
%1->0: di tu phai sang trai.
goal([0,0,3,3,0]).

%1 su 1 quy tu trai sang phai.
operation([A,B,C,D,1],[X,Y,Z,T,0]):- A>0,B>0,X is A-1,Y is B-1,Z is C+1, T is D+1.
%1 su 1 quy tu phai sang trai.
operation([A,B,C,D,0],[X,Y,Z,T,1]):-C>0,D>0,X is A+1, Y is B+1,Z is C-1, T is D-1.

operation([A,B,C,D,0],[X,B,Z,D,1]):-C>1, X is A+2, Z is C-2.
operation([A,B,C,D,1],[X,B,Z,D,0]):-A>1, X is A-2, Z is C+2.

operation([A,B,C,D,1],[A,Y,C,T,0]):-B>1, Y is B-2, T is D+2.
operation([A,B,C,D,0],[A,Y,C,T,1]):-D>1 ,Y is B+2, T is D-2.

operation([A,B,C,D,0],[X,B,Z,D,1]):-C>0, X is A+1, Z is C-1.
operation([A,B,C,D,1],[X,B,Z,D,0]):-A>0, X is A-1, Z is C+1.

operation([A,B,C,D,0],[A,Y,C,T,1]):-D>0, Y is B+1, T is D-1.
operation([A,B,C,D,1],[A,Y,C,T,0]):-B>0, Y is B-1, T is D+1.

invalid([A,B,C,D,_]):-B>A,D>C.
invalid([A,B,0,_,_]):-B>A.
invalid([0,_,A,B,_]):-B>A.
dfs(CurrentNode, CurrentPath, [CurrentNode, CurrentPath]):- goal(CurrentNode).
dfs(CurrentNode,CurrentPath,Path):-transition(CurrentNode,NextNode),
CurrentNode \= NextNode,
\+member(NextNode,CurrentPath),
\+invalid(CurrentNode),
dfs(NextNode,[CurrentNode|CurrentPath],Path).
```

Đoạn code để giải bài toán ba thầy tu ba con quỷ.

```
[4]  ?- dfs([3,3,0,0,1],[],X).
X = [[0, 0, 3, 3, 0], [[1, 1, 2, 2, 1], [0, 1, 3, 2, 0], [2, 1, 1, 2|...], [1, 0, 2|...], [3, 0|...], [2|...], [...|...]|...]] .
```

Kết quả chương trình.

3. Diễn Giải

Quy ước:

- A,B đại diện cho số thầy tu và số quỷ ở bờ sông bên trái. C,D đại diện cho số thầy tu và số quỷ ở bờ sông bên phải.
- Trạng thái 1 biểu diễn đang ở bờ sông bên trái, trạng thái 0 biểu diễn đang ở bờ sông bên phải.
- F,W,G,C đại diện cho trạng thái của người nông dân, con sói, dê và bắp cải.

- Trạng thái bắt đầu là [3,3,0,0,1], khi mà cả bốn đều ở bờ bên trái.
- Trạng thái kết thúc là [0,0,3,3,0], khi mà cả bốn đề ở bờ bên phải.

Các hành động được thử hiện:

- 1 thầy tu và 1 quỷ sang sông:

```
operation([A,B,C,D,1],[X,Y,Z,T,0]):- A>0,B>0,X is A-1,Y is B-1,Z is C+1, T is D+1.
operation([A,B,C,D,0],[X,Y,Z,T,1]):-C>0,D>0,X is A+1, Y is B+1,Z is C-1, T is D-1.
```

- 2 thầy tu hoặc 2 quỷ cùng sang sông:

```
operation([A,B,C,D,0],[X,B,Z,D,1]):-C>1, X is A+2, Z is C-2.
operation([A,B,C,D,1],[X,B,Z,D,0]):-A>1, X is A-2, Z is C+2.

operation([A,B,C,D,1],[A,Y,C,T,0]):-B>1, Y is B-2, T is D+2.
operation([A,B,C,D,1],[A,Y,C,T,0]):-B>1, Y is B+2, T is D-2.
```

- 1 thầy tu hoặc 1 quỷ qua sông:

```
operation([A,B,C,D,0],[X,B,Z,D,1]):-C>0, X is A+1, Z is C-1.
operation([A,B,C,D,1],[X,B,Z,D,0]):-A>0, X is A-1, Z is C+1.

operation([A,B,C,D,0],[A,Y,C,T,1]):-D>0, Y is B+1, T is D-1.
operation([A,B,C,D,1],[A,Y,C,T,0]):-B>0, Y is B-1, T is
```

- Áp dụng thuật toán DFS để có thể tìm ra tất cả lời giải của bài toán.

```
dfs(CurrentNode, CurrentPath, [CurrentNode, CurrentPath]):- goal(CurrentNode).
dfs(CurrentNode,CurrentPath,Path):-transition(CurrentNode,NextNode),
CurrentNode \= NextNode,
\+member(NextNode,CurrentPath),
\+invalid(CurrentNode),
dfs(NextNode,[CurrentNode|CurrentPath],Path).
```

Trạng thái kết thúc của bài toán:

```
goal([0,0,3,3,0]).
```

Trạng thái bị cấm: số thầy tu ít hơn số quỷ trên bờ sông bất kỳ.

```
invalid([A,B,C,D,_]):-B>A,D>C.
invalid([A,B,0,_,_]):-B>A.
invalid([0,_,A,B,_]):-B>A.
```

Diễn giải:

+ CurrentNode là trạng thái hiện tại, CurrentPath là list gồm các trạng thái đã duyệt (bao gồm Current Node).
+ Nếu CurrentNode không phải là trạng thái đích, ta chuyển sang tìm tất cả các trạng thái có thể nảy sinh từ trạng thái hiện tại; với các ràng buộc như trạng thái kế tiếp (NextNode) phải khác CurrentNode, NextNode không thuộc các trường hợp đã duyệt qua từ CurrentNode và NextNode không thuộc các trường hợp bị cấm.
+ Cho CurrentNode vào list các trạng thái đã duyệt. Tiếp tục gọi hàm dfs bắt đầu từ NextNode. Thực hiện đệ quy cho đến khi tìm được trạng thái kết thúc của bài toán.