Московский Авиационный Институт

(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа**

**по курсу «Логическое Программирование»**

**Тема:**

**Решение задач методом поиска в пространстве состояний.**

|  |  |
| --- | --- |
| Студент: | Суворова С. А. |
| Группа: | М80-206Б-18 |
| Преподаватель: | Сошников Д.В. |
| Преподаватель: | Левинская М. А. |
| Оценка: |  |
| Дата: |  |

Москва

2019

**Введение.**

Задачи искусственного интеллекта наиболее удобным образом решаются методом поиска в пространстве состояний. Будучи логическим языком программирования Prolog при поиске решения составляет дерево, структура в которой поиск производится наиболее оптимально, поэтому ,используя различные методы её обхода, можно быстро найти решения.

**Задание.**

Крестьянину нужно переправить волка, козу и капусту с левого берега реки на правый. Как это сделать за минимальное число шагов, если в распоряжении крестьянина имеется двухместная лодка, и нельзя оставлять волка и козу или козу и капусту вместе без присмотра человека.

**Принцип решения.**

Все способы будут похожи своими используемыми предикатами:

member(X, [X|\_]).

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

move([W, G, C, B], [W1, G, C, B1]) :-

opposite(B, B1),

opposite(W, W1),

opposite(W, B),

opposite(W1, B1),

unsafe([W1, G, C, B1]).

move([W, G, C, B], [W, G1, C, B1]) :-

opposite(B, B1),

opposite(G, G1),

opposite(G, B),

opposite(G1, B1),

unsafe([W, G1, C, B1]).

move([W, G, C, B], [W, G, C1, B1]) :-

opposite(B, B1),

opposite(C, C1),

opposite(C, B),

opposite(C1, B1),

unsafe([W, G, C1, B1]).

move([W, G, C, B], [W, G, C, B1]) :- opposite(B, B1), unsafe([W, G, C, B1]).

opposite(left, right).

opposite(right, left).

unsafe([W, W, C, B]) :-

B \== W.

unsafe([W, G, G, B]) :-

B \== G.

goal([right, right, right, right]).

Отличатся будут только методы обхода:

1.Поиск в глубину.

solve(Node, Solution) :- depth([], Node, Solution).

depth(Path, Node, [Node | Path]) :- goal(Node).

depth(Path, Node, Sol) :-

move(Node, Node1),

member(Node1, Path),

depth([Node | Path], Node1, Sol).

2.Поиск в ширину.

solve\_breadth(Start, Solution) :-

breadth([[Start]], Solution).

breadth([[Node | Path] | \_], [Node | Path]) :-

goal(Node).

breadth([Path | Paths], Solution) :-

extend(Path, NewPaths),

append(Paths, NewPaths, Paths1),

breadth(Paths1, Solution).

extend([Node | Path], NewPaths) :-

bagof([New | [Node | Path]], (move(Node, New), member(New, [Node | Path])), NewPaths), !.

extend(Path, []).

3.Поиск в глубину с итерационным заглублением.

path(Node, Node, [Node]).

path(First, Last, [Last | Path]) :-

path(First, OneButLast, Path),

move(OneButLast, Last),

member(Last, Path).

depth\_iterative(Node, Solution) :-

path(Node, Goal, Solution),

goal(Goal).

**Результаты.**

1.Поиск в глубину.

Длинна пути:5

Время работы: 2 секунды.

2.Поиск в ширину.

Длинна пути:5

Время работы: 1 секунда.

3.Поиск в глубину с итерационным заглублением.

Длинна пути:5

Время работы: 1.5 секунды.

**Вывод.**

При выполнении данной лабораторной работы я лучше поняла каким образом в Прологе строится дерево решений, и как можно обеспечить обход данного дерева для наиболее эффективного способа нахождения решения.