Московский авиационный институт
(Национальный исследовательский университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Лабораторная работа № 3

Решение задач методом поиска в пространстве состояний
по курсу "Логическое программирование"

Студент: Алексюнина Ю.В.
Группа: 80-307Б
Преподаватели:
Сошников Д.В., Левинская М.А.
Оценка:

Москва, 2020

Введение

В современном мире все популярнее становится исскуственный интеллект, который в свою очередь постоянно сталкивается с задачей нахождения решения, имея заданное пространство состояний. Именно для таких задач необходимы алгоритмы поиска в пространствах состояний. Для реализации таких алгоритмов очень удобен язык Пролог, в силу того, что пространство состояний обычно представляется в виде графа, в котором вершинами являются состояния, а ребрами - переходы между ними. И в тоже время поиск решений предикатов в прологе осуществляется поиском по всевозможным вариантам значений аргументов, при котором предикат может быть истинным. Данное сходство между Прологом и алгоритмами поиска помогает легко, в несколько строк реализовать эти алгоритмы в Прологе.

Задание

Написать и отладить Пролог-программу решения задачи искусственного интеллекта, используя технологию поиска в пространстве состояний в соответствии с номером варианта.

2. Три миссионера и три каннибала хотят переправиться с левого берега реки на правый. Как это сделать за минимальное количество шагов, если в их распоряжении 3-ех местная лодка и ни при каких обстоятельствах миссионеры не должны остаться в меньшинстве.

Принцип решения

Принцип работы заключается в алгоритмах поиска решений в пространстве состояний. Предикат move\3 - предикат перехода из одного состояния в другое. 3-ий аргумент предиката указывает на то, где в данный момент находится лодка(1 - левый берег, 2- правый).

```
checkside(X, Y) :- X >= Y, !.
checkside(X, Y) :- X = 0, !.
checkside(_, Y) :- Y = 0, !.

move(Until, After, 1) :-
  type(Between),
  minuslist(Until, Between, After),
  manyelems(After, "каннибал", N2),
```

```
  manyelems(After, "миссионер", M2),
  checkside(M2, N2),
  minuslist(["каннибал", "каннибал", "каннибал", "миссионер", "миссионер", "миссионер"],
After, Another),
  manyelems(Another, "каннибал", N3),
  manyelems(Another, "миссионер", M3),
  checkside(M3, N3).

  move(Until, After, 2) :-
  type(Between),
  minuslist(After, Between, Until),
  manyelems(After, "каннибал", N2),
  manyelems(After, "миссионер", M2),
  checkside(M2, N2),
  minuslist(["каннибал", "каннибал", "каннибал", "миссионер", "миссионер", "миссионер"],
After, Another),
  manyelems(Another, "каннибал", N3),
  manyelems(Another, "миссионер", M3),
  checkside(M3, N3).
```

Здесь type\1 содержит всевозможные варианты размещения в лодке, minuslist\3 осуществляет вычитание из одного списка элементов, составляющих другой список, checkside\2 делает проверку условия безопасности миссионеров в данном состоянии.

Предикат waycreate добавляет в список последовательных состояний, требуемых для достижения цели, очередное состояние, если оно не является членом списка (в целях предотвращения зацикливания).

```
waycreate([st(X, 1)|T], [st(Y, 2), st(X, 1)|T]):-
  move(X, Y, 1),
  not(mymember(st(Y, 2), [st(X, 1)|T])).

waycreate([st(X, 2)|T], [st(Y, 1), st(X, 2)|T]):-
  move(X, Y, 2),
  not(mymember(st(Y, 1), [st(X, 2)|T])).
```

И на основе вышеописанных предикатов реализуются алгоритмы поиска в глубину, в ширину и в глубину с итеративным погружением.

```
% поиск в глубину

dsrch([st(X, 2)|T], X, [st(X, 2)|T]).

dsrch([st(R, N)|P], X, L):-
  waycreate([st(R, N)|P], P1),
  dsrch(P1, X, L).
```

```
dpth_search(Until, After):-
 get_time(Time1),
 dsrch([st(Until, 1)], After, L),
 get_time(Time2),
 T is Time2 - Time1,
 write('Time: '),
 writeln(T),
 prnt(L).

% поиск в ширину

bdth([[st(X, 2)|T]|_], X, [st(X, 2)|T]).

bdth([B|P], X, L):-
  findall(W, waycreate(B, W), Q),
  append(P, Q, QP),
  !, bdth(QP, X, L);
  bdth(P, X, L).

 bdth_search(Until, After):-
  get_time(Time1),
  bdth([[st(Until, 1)]], After, L),
  get_time(Time2),
  T is Time2 - Time1,
  write('Time: '),
  writeln(T),
  prnt(L).

% поиск с итерационным углублением

natural(1).
natural(B) :- natural(A), B is A + 1.

itdpth([st(A, 2)|T], A, [st(A, 2)|T], 0).

itdpth(P, A, L, N) :-
  N > 0,
  waycreate(P, Pl),
  Nl is N - 1,
  itdpth(Pl, A, L, Nl).

id_search(Until, After) :-
  get_time(Time1),
  natural(N),
  itdpth([st(Until, 1)], After, L, N),
  get_time(Time2),
  T is Time2 - Time1,
  write('Time: '),
```

```
  writeln(T),
  prnt(L), !.
```

Здесь предикат print\1 осуществляет более читабельный вывод решения, а get_time замеряет время для последующего измерения времени исполнения программы.

Результат работы программы

dpth_search(["миссионер", "миссионер", "миссионер", "каннибал", "каннибал", "каннибал"], []).

Time: 0.0004401206970214844
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал, каннибал]~~Берег 2:[]
Берег 1:[миссионер, миссионер, каннибал, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал]~~Берег 2:[каннибал]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[каннибал, миссионер, миссионер, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер, миссионер]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[]~~Берег 2:[каннибал, каннибал, каннибал, миссионер, миссионер, миссионер]

Time: 4.369460105895996
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал, каннибал]~~Берег 2:[]
Берег 1:[миссионер, миссионер, каннибал, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал]~~Берег 2:[каннибал]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[каннибал, миссионер, миссионер, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер, миссионер]
Берег 1:[каннибал, каннибал]~~Берег 2:[каннибал, миссионер, миссионер, миссионер]
Берег 1:[]~~Берег 2:[каннибал, каннибал, каннибал, миссионер, миссионер, миссионер]

Time: 7.985560178756714
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал, каннибал]~~Берег 2:[]
Берег 1:[миссионер, миссионер, миссионер, каннибал]~~Берег 2:[каннибал, каннибал]
Берег 1:[каннибал, миссионер, миссионер, миссионер, каннибал]~~Берег 2:[каннибал]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[каннибал, миссионер, миссионер, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер, миссионер]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[]~~Берег 2:[каннибал, каннибал, каннибал, миссионер, миссионер, миссионер]

Time: 11.43206000328064
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал, каннибал]~~Берег 2:[]

Берег 1:[миссионер, миссионер, миссионер, каннибал]~~Берег 2:[каннибал, каннибал]
Берег 1:[каннибал, миссионер, миссионер, миссионер, каннибал]~~Берег 2:[каннибал]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[каннибал, миссионер, миссионер, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер, миссионер]
Берег 1:[каннибал, каннибал]~~Берег 2:[каннибал, миссионер, миссионер, миссионер]
Берег 1:[]~~Берег 2:[каннибал, каннибал, каннибал, миссионер, миссионер, миссионер]

bdth_search(["миссионер", "миссионер", "миссионер", "каннибал", "каннибал", "каннибал"], []).

Time: 0.0018999576568603516
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал, каннибал]~~Берег 2:[]
Берег 1:[миссионер, миссионер, каннибал, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал]~~Берег 2:[каннибал]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[каннибал, миссионер, миссионер, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер, миссионер]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[]~~Берег 2:[каннибал, каннибал, каннибал, миссионер, миссионер, миссионер]

Time: 3.871799945831299
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал, каннибал]~~Берег 2:[]
Берег 1:[миссионер, миссионер, каннибал, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал]~~Берег 2:[каннибал]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[каннибал, миссионер, миссионер, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер, миссионер]
Берег 1:[каннибал, каннибал]~~Берег 2:[каннибал, миссионер, миссионер, миссионер]
Берег 1:[]~~Берег 2:[каннибал, каннибал, каннибал, миссионер, миссионер, миссионер]

Time: 7.0471601486206055
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал, каннибал]~~Берег 2:[]
Берег 1:[миссионер, миссионер, миссионер, каннибал]~~Берег 2:[каннибал, каннибал]
Берег 1:[каннибал, миссионер, миссионер, миссионер, каннибал]~~Берег 2:[каннибал]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[каннибал, миссионер, миссионер, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер, миссионер]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[]~~Берег 2:[каннибал, каннибал, каннибал, миссионер, миссионер, миссионер]

Time: 10.159600019454956
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал, каннибал]~~Берег 2:[]

Берег 1:[миссионер, миссионер, миссионер, каннибал]~~Берег 2:[каннибал, каннибал]
Берег 1:[каннибал, миссионер, миссионер, миссионер, каннибал]~~Берег 2:[каннибал]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[каннибал, миссионер, миссионер, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер, миссионер]
Берег 1:[каннибал, каннибал]~~Берег 2:[каннибал, миссионер, миссионер, миссионер]
Берег 1:[]~~Берег 2:[каннибал, каннибал, каннибал, миссионер, миссионер, миссионер]

id_search(["миссионер", "миссионер", "миссионер", "каннибал", "каннибал", "каннибал"], []).

Time: 0.00635981559753418
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал, каннибал]~~Берег 2:[]
Берег 1:[миссионер, миссионер, каннибал, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[миссионер, миссионер, миссионер, каннибал, каннибал]~~Берег 2:[каннибал]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[каннибал, миссионер, миссионер, каннибал]~~Берег 2:[каннибал, миссионер]
Берег 1:[каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер, миссионер]
Берег 1:[миссионер, каннибал]~~Берег 2:[каннибал, каннибал, миссионер, миссионер]
Берег 1:[]~~Берег 2:[каннибал, каннибал, каннибал, миссионер, миссионер, миссионер]

Вывод

Алгоритм с итерационным погружением позволяет искать наикратчайший путь с относительно небольшим расходом памяти. Такой алгоритм следует использовать, когда стоит задача нахождения наименьшего пути в условиях ограниченной памяти. Вторым алгоритмом, позволяющим искать наикратчайший путь, оказался алгоритм поиска в ширину, использование которого позволяет искать нужный путь чуть быстрее, нежели поиск с итерационным погружением, однако при этом с крупным перерасходом памяти. И самым быстрым, простым в реализации и наиболее пригодным к моей сегодняшней задаче, так как все решения имели длину 7, оказался алгоритм поиска в глубину, который наиболее пригоден в условиях, если не требуется искать наикратчайший путь.