МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное автономное бюджетное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационных систем

Григорьева Мария Георгиевна

Институт информационных технологий и управления в технических системах

курс 4 группа ИС(б) – 43о

09.03.02 Информационные системы (уровень бакалавриата)

ОТЧЁТ

о лабораторном практикуме №5

по дисциплине «Методы и системы искусственного интеллекта»

на тему: «Методы поиска решений задач в пространстве состояний»

Отметка о зачёте \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

Волкова А.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность) (подпись) (инициалы, фамилия)

Севастополь 2018

ЦЕЛЬ

Исследование методов поиска решений задач в пространстве состояний и овладение методологией решения логических задач с применением этих методов.

ВАРИАНТ

6 вариант:

Коммивояжер должен посетить каждый из N заданных городов, изображенных на карте. Между каждой парой городов имеется путь, длина которого указана на этой карте. Нужно, отправляясь из стартового города, найти самый короткий путь, по которому коммивояжер по одному разу проходит через каждый из городов и затем возвращается в стартовый город. N вводится с клавиатуры (N>5). Программа должна генерировать расстояния между городами автоматически и отображать их. Метод поиска в ширину.

ТЕКСТ ПРОГРАММЫ

(defun COMMI\_ROUTE(graph \_path)

(if graph

(cons

(car \_path)

(apply 'COMMI\_ROUTE

(apply

(function

(lambda (\_count)

(list

(COL\_SWAP (mapcar 'cdr (ROW\_SWAP (cdr graph) \_count)) \_count)

(ROW\_SWAP (cdr \_path) \_count))))

(list

(SWAP\_COUNT (cdr (car graph)))))))))

(defun ROW\_SWAP (\_list \_count)

(if (> \_count 0)

(ROW\_SWAP (append (cdr \_list) (list (car \_list))) (1- \_count))

\_list))

(defun COL\_SWAP (\_list \_count)

(if (> \_count 0)

(COL\_SWAP (mapcar 'append (mapcar 'cdr \_list) (mapcar 'list (mapcar 'car \_list))) (1- \_count))

\_list))

(defun SWAP\_COUNT (\_dist\_list)

(if \_dist\_list

(-(length \_dist\_list)(length (member (apply 'min \_dist\_list) \_dist\_list)))

0))

1. РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ

(princ(COMMI\_ROUTE '((2 . (2 6)) (1 . (1 3 4)) (3 . (2 4))

(4 . (2 3 5)) (5 . (4 6)) (6 . (1 5))) '(1 2 3 4 5 6)))

(print(ROW\_SWAP '((1 . (2 6)) (2 . (1 3 4)) (3 . (2 4))

(4 . (2 3 5)) (5 . (4 6)) (6 . (1 5))) '1))

(print(COL\_SWAP '((1 . (2 6)) (2 . (1 3 4)) (3 . (2 4))

(4 . (2 3 5)) (5 . (4 6)) (6 . (1 5))) '1))

(print(SWAP\_COUNT '(4)))

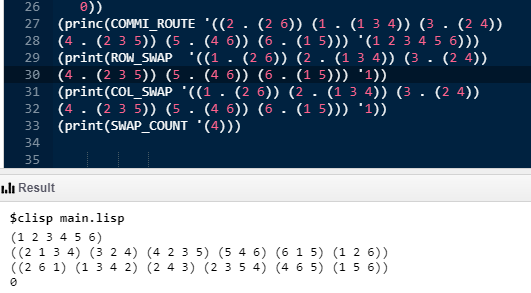


Рисунок 1 – Результат выполнения программы.

ВЫВОДЫ

В ходе данной лабораторной работы были исследованы методы поиска решений задач в пространстве состояний и методологии решения логических задач с применением этих методов. На языке Lisp была решена задача коммивояжера методом поиска в ширину.

Поиск в ширину является оптимальным, если стоимости раскрытия всех вершин равны. Если с каждой вершиной связать стоимость её раскрытия и из списка OPEN выбирать вершину с наименьшей стоимостью, то рассмотренная выше процедура будет обеспечивать нахождение оптимального решения по критерию стоимости (при условии модификации стоимости вершины в списке OPEN, при обнаружении пути с меньшей стоимостью).

Рассмотренная стратегия гарантирует полноту поиска, если стоимость каждого участка пути положительная величина. Так как поиск в этом случае направляется стоимостью путей, то *временная и пространственная сложности* ( при построении дерева поиска) в наихудшем случае будут равны *O(B1+С/с),* где *С* — стоимость оптимального решения, *с* — минимальная стоимость каждого действия. Эта оценка может быть больше *O(Bd)* . Это связано с тем, что процедура поиска по критерию стоимости часто обследует поддеревья поиска, состоящие из мелких этапов небольшой стоимости, прежде чем перейти к исследованию путей, в которые входят крупные, но возможно более полезные этапы.