Занимательная информатика

Пойти туда, зная куда

В.Ф. Очков

«Сообразив, куда прежде, куда
после ехать, чтоб не возвращаться,
Нехлюдов прежде всего направился
в сенат».

Л.Н. Толстой «Воскресенье»

Каждый из нас, отправляясь в путь по мало-мальски сложному маршруту, должен «сообразить, куда прежде ехать, куда после, чтоб не возвращаться». Многие из нас, если и не работали (подрабатывали) курьером, то получали заказанный товар из рук этого человека. Курьеру же, «направляясь в путь», обязательно нужно сообразить «куда прежде, куда позже ехать», чтобы оптимизировать маршрут, то есть попытаться решить *задачу коммивояжера* (странствующего торговца, коробейника). *Задача коммивояжера*: найти маршрут обхода n городов, побывав в каждом городе один раз, вернувшись в исходную точку и сведя к минимуму путь (время в пути, расходы на дорогу и т.д.).

Разработано множество алгоритмов различной сложности для решения этой задачи. Сначала ее несколько упрощают: берут n точек на плоскости с координатами, хранящимися в векторах X и Y, и считают, что путь из i-й точки (i = 1…n) до j-й точки (j=1…n; i ≠ j) лежит по прямой и может быть пройден в двух направлениях (от i к j и от j к i). Один из простейших алгоритмов решения задачи коммивояжера состоит в следующем. Берется случайная точка на плоскости, которая будет первым элементом вектора путь. Затем ищется ближайшая точка от этой стартовой точки. Она становится второй точкой вектора путь. Далее эта операция повторяется до тех пор, пока не будут перебраны все точки. В этом переборе добавляется еще одно условие: нельзя «идти» в город, пардон, в точку, где мы уже были. На рис. 1 показана программа, написанная в среде Mathcad Prime, реализующая этот *алгоритм ближайшего соседа*.

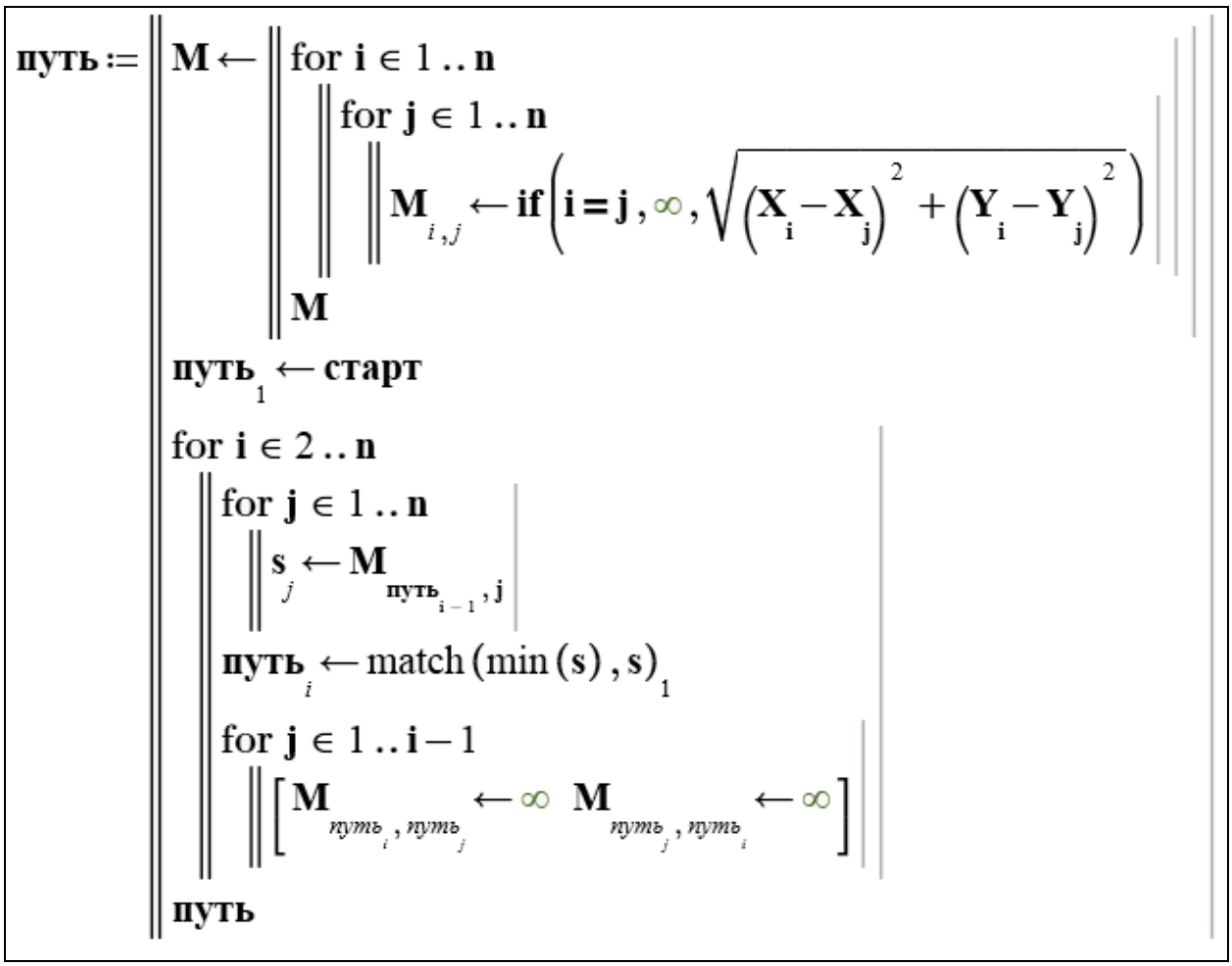

Рис. 1. Программа «Ближайший сосед»

В программе, показанной на рис. 1, сначала заполняется квадратная матрица M, хранящая расстояние от точки i до точки j. Если i = j (диагональ матрицы), то этот элемент матрицы будет хранить значение бесконечности. Это будет знаком того, что по этому маршруту ходить не нужно. Бесконечность будет записываться в элементы матрицы $M_{i, j}$ и $M_{j, i}$, если от точки i к точке j (или наоборот) путь уже пройден. Но еще раньше нужно заполнить векторы X и Y (координаты точек на плоскости) и скаляр старт (номер первой точки маршрута).

Программа, реализующая метод ближайшего соседа (рис. 1), получилась такая компактная благодаря двум встроенным функциям Mathcad: min и match. Первая ищет минимальное значение в векторе или матрице. Вторая определяет координаты элемента вектора или матрицы с заданным значением[1].

[1] Очень часто школьникам дают задание написать на языке Pascal такие программы методом перебора всех элементов вектора или матрицы.

В программе в цикле с параметром i (перебор точек около очередной точки) составляется вектор S, который хранит расстояние от этой i-й точки до всех остальных j-х точек. Далее с помощью функций min и match (см. выше) определяется точка (очередной элемент исходного вектора путь), ближайшая к нашей точке, в которой «мы еще не были».

Все очень просто, но… «иная простота хуже воровства».

На рис. 2 показан предпоследний кадр анимации обхода городов Люксембурга (одной из самых маленьких стран Европы) по алгоритму ближайшего соседа. Этот алгоритм относится к группе *жадных* алгоритмов: бросаясь к ближайшему соседу, мы делаем петли и пропускаем точки, в которые так или иначе придется «зайти», делая при этом броски из одного конца в другой конец «страны».

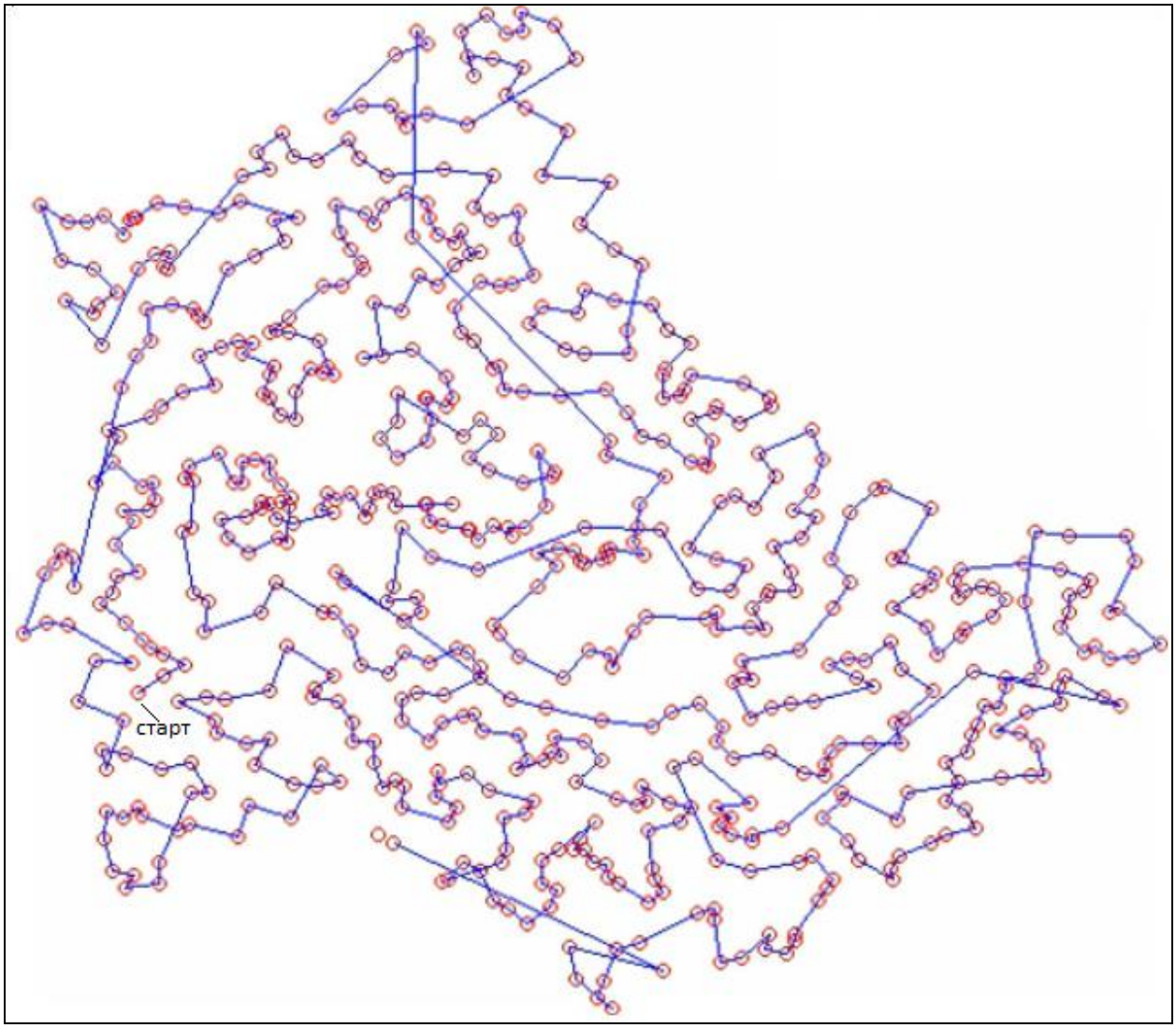

Рис. 2. Путь коммивояжера по Люксембургу

Мы не приводим здесь более сложные оптимальные алгоритмы решения задачи коммивояжера. Мы только ставим перед школьниками очень интересную и занимательную задачу, призывая найти в книгах или Интернете оптимальные алгоритмы решения задачи коммивояжера и реализовать их на уроках информатики.