Глава 1 с Введением:
русская литература, задача коммивояжера, санкции…

В главе описывается урок по технологии STEM на стыке математики, информатики, литературы, москвоведения и даже политологии. Описана методика решения задачи коммивояжера методом ближайшего соседа.

Есть у Чехова довольно забавный рассказ «Новогодняя пытка». Текст короткий – его стоит прочитать [1] или прослушать [2], а потом вернуться к этой главе. В рассказе описано, как жена со скандалом выгоняет из дома мужа делать визиты к родственникам и друзьям – поздравить их с Новым годом: «*Ишь что выдумал: визитов не делать!*». Для посещений нужно было в Москве проехаться по такому маршруту: Зубовский бульвар (старт – место, где жил герой рассказа), Красные казармы в Лефортово (в этом районе находится alma mater четырех авторов данного учебного пособия), Хамовники, Нижегородский вокзал[1], Крестовская застава, Калужские ворота и Сокольницкая роща. А в конце поездки вернуться домой на Зубовский бульвар. Семь памятных мест в Москве как семь древних городов, настаивающих на том что именно у них родился Гомер: Смирна, Хиос, Колофон, Пилос, Аргос, Итака и Афины. Из семи городов авторам известны только три: Смира, Итака и Афины. Три и семь – это красивые числа. Можно также упомянуть семерых древних мудрецов, семь чудес света, семерых гномов, семерых богатырей. Число семь завораживает рассказчиков. Чехов здесь не исключение.

Авторы попробовали заказать такую круговую поездку в интернете через одно из приложений по заказу такси. Вот, что у них получилось – см. рис. 1.1.

[1] Это был временный вокзал для железной дороги Москва – Нижний Новгород. Ее позже продлили до Курского вокзала, а сам Нижегородский вокзал снесли. От него осталось название Нижегородской улицы. Крестовская застава была в районе теперешней станции московского метро Рижская, а Калужские ворота – в районе станции Октябрьская, которая раньше так и называлась – Калужской. Современная станция метро Калужская сейчас находится в другом месте.

Глава 1

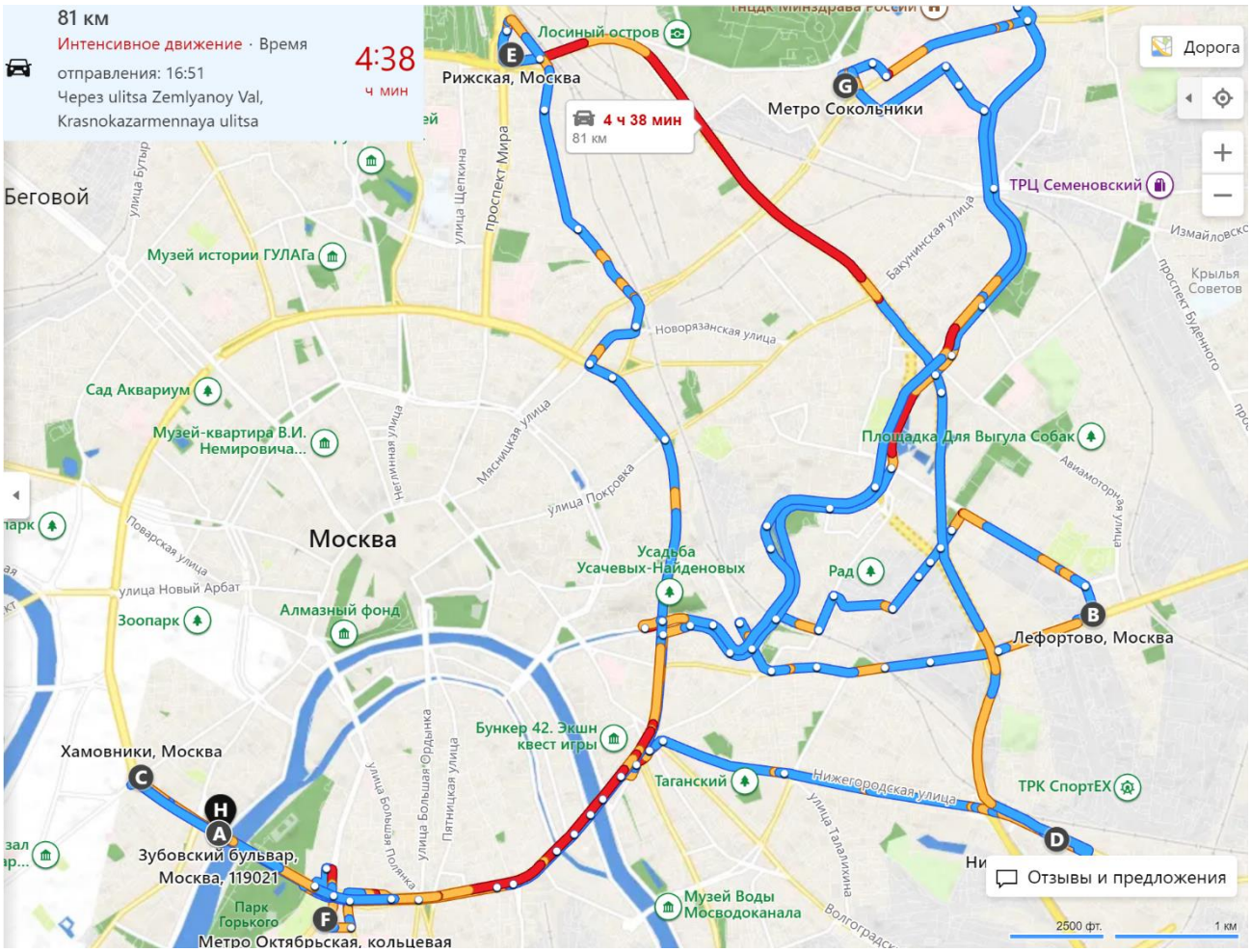

Рис. 1.1. Маршрут такси для новогодних визитов

Буквами A-H на рис.1.1 обозначены места планируемых визитов в порядке их перечисления в рассказе Чехова. Точки A и H – это фактически дна точка – начало и конец маршрута на Зубовском бульваре.

Сине-желто-красные линии на рис. 1.1 – это планируемое движение такси на расстояние 81 км за 4 часа 38 минут (см. левый верхний угол рисунка) со средней скоростью 17.5 км/ч.

Если же отказаться от такси и передвигаться по городу пешком (см. синие пунктирные кружочки на рис. 1.2), то расстояние уменьшиться до 62 км, но на него придется затратить уже 12 часов 19 минут со средней скоростью 5 км в час.

Глава 1

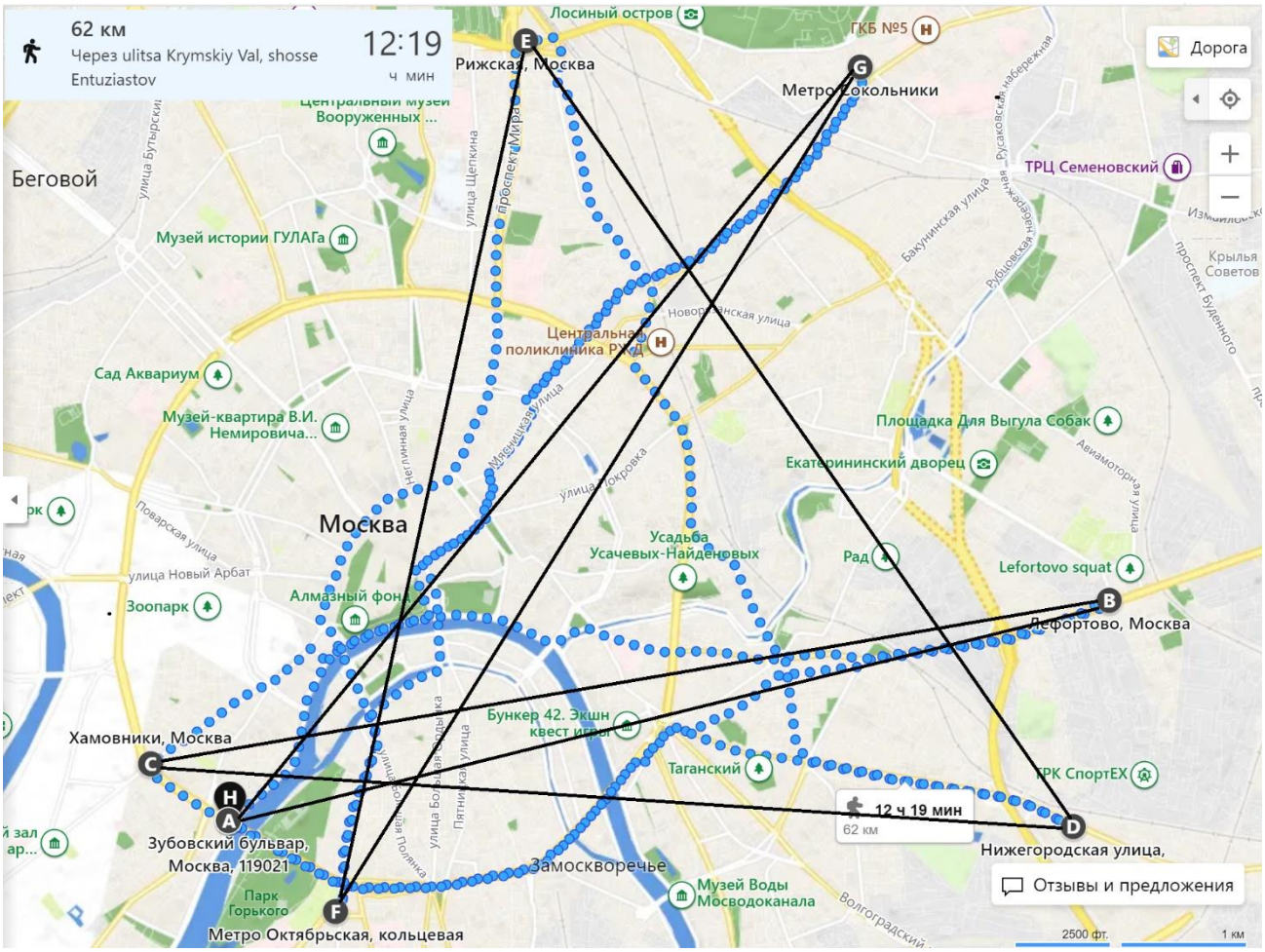

Рис. 1.2. Пеший и вертолетный маршрут для новогодних визитов

Черные прямые линии на рис. 1.2 обозначают следующее. Давно уже ведутся разговоры о том, что в Москве и других крупных городах России вот-вот появится воздушное такси, которому не страшны пробки (см. красные участки линий на рис. 1.1) и которое будет летать по прямым маршрутам. Таким способом передвижения мог бы воспользоваться герой рассказа Чехова, если б дожил до таких времен. А вот еще один не такой уж фантастический сценарий. Герой рассказа помещает свои визитные карточки на квадрокоптер, который опять же по прямым линиям без пробок облетает друзей и знакомых и сбрасывает им новогодние послания. А можно заодно сбросить и небольшой подарок! В старой России по праздникам чиновники приходили к начальнику домой и расписывались в специальном листе у швейцара[2]. Люди, посещавшие с визитом себе равных, но не заставшие хозяина дома, оставляли свои визитки в прихожей. Отсюда, кстати, и идет название этих маленьких белых картонных прямоугольников. Вот, что можно прочесть у Гоголя в «Мертвых душах»: «*Визитная карточка, будь она писана хоть на трефовой двойке или бубновом тузе, но вещь была очень священная. Из-за нее две дамы, большие приятельницы и даже родственницы, перессорились совершенно, именно за то, что одна из них как-то манкировала контрвизитом. И уж как ни*

[2] В рассказе Чехова «Тайна» подпись неизвестного в листе у швейцара имела очень интересные и комичные последствия. Читатель, загляни и этот рассказ [6]!

старались потом мужья и родственники примирить их, но нет, оказалось, что всё можно сделать на свете, одного только нельзя: примирить двух дам, поссорившихся за манкировку визита.»[3]

Сразу видно, что маршруты поездки на такси (рис. 1.1) или пешего путешествия (рис. 1.2) далеко не оптимальные. Можно, конечно, предположить, что время визитов было оговорено заранее. Этим и объясняются такие дальние переезды из одного конца города в другой. Но скорее всего дело обстояло так. Муж с новогодним похмельным синдромом (смотрите или слушайте рассказ Чехова) получил от жены список тех, кого нужно посетить. Супруг, особо не раздумывая, тупо глядел в записку и отдавал нужные приказания извозчикам[4]. Рассказ, кстати, оканчивается описанием еще одного скандала, который жена закатила мужу из-за перерасхода денег на поездку (5 рублей 80 копеек). Поездка на такси в наше время будет стоить по оценке компьютерного приложения (рис. 1.1 и 1.2) порядка пяти тысяч.

А вот как поступил герой романа Л.Н. Толстого «Воскресение» [3]: «*Сообразив, куда прежде, куда после ехать, чтоб не возвращаться, Нехлюдов прежде всего направился в сенат*». Нехлюдов в Петербурге занимался не пустыми визитами – он обивал пороги государственных учреждений в надежде изменить приговор Катюше Масловой – жертвы похоти и подлости самого Нехлюдова, а потом, вдобавок, и судебной ошибки. Нехлюдов, в отличие от героя рассказа Чехова, перед своим «турне» попытался как-то вчерне решить *задачу коммивояжера*. Такую задачу, кстати, в наше время решают доставщики еды на велосипедах с желтыми квадратными сумками за плечами.

Напишем небольшую программу, которая решит зафиксированную в рассказе Чехова задачу коммивояжера – странствующего торговца (TSP – от англ. travelling salesman problem). А какой компьютерный инструмент для этого взять!? Авторы привыкли работать с Mathcad [4], но…

Во время работы над этим учебным пособием с Россией случилось то, что с какой-то роковой периодичностью случается с ней в начале каждого века. Не будем слишком далеко углубляться в историю, но… Начало XVII века – Смутное время и польская интервенция, начало XVIII века – Северная война с ее ключевой Полтавской битвой, начало XIX века – нашествие Наполеона, начало XX века – Первая мировая и гражданская войны, начало XXI века – украинский кризис, который неизвестно еще когда и чем закончится. Но и в серединах веков тоже не все было спокойно: XIX век – Крымская война, XX век – Великая Отечественная… Исторический маятник качается то в одну, то в другую сторону (от войны к миру) с периодом примерно в полвека. Войну с Наполеоном и Крымскую войну мы можем изучать по произведениям упомянутого нами Льва Николаевича. История взаимоотношений России и Запада начала XXI века ждет своего Толстого. Как вам понравится такое название будущего романа: «Гибридная война и цифровой мир»?

Последние события, в частности введенные Западом санкции, исключают возможность скачивания с сайта ptc.com полной версии Mathcad и работы с ней месяц, по истечение которого программа становится урезанной (Mathcad Express), если не куплена лицензия на полную версию. Но сейчас из-за санкций нельзя ничего скачать и невозможно за что-то заплатить. Россию снова в пылу русофобии[5]

---

[3] Обширное цитирование классиков русской литературы можно прокомментировать еще одной цитатой из чеховского водевиля «Свадьба» [8]: «*Они хочут свою образованность показать и всегда говорят о непонятном*». Но авторы заботятся и о гуманитаризации инженерного образования… Ну и эрудицию свою, конечно, тоже показать хочется не только в SMath, но и в литературе.

[4] Сейчас подобные записки жены пишут перед тем, как отправить мужа в магазин. Если это огромный супермаркет, то проблема оптимизации маршрута в нем тоже выливается в задачу коммивояжера, точнее, в задачу странствующего покупателя, а не странствующего продавца.

[5] Русофобия – это хроническая болезнь Европы с периодами рецессий и обострений. Правда, на Западе эту болезнь называют по-другому и считают, что ею болеет не Запад, а Россия. Но истина, как всегда, находится

пытаются загнать в медвежий угол. Да и основа всех основ – операционная система Windows – тоже у нас висит на волоске.

Но, к счастью, у американской программы Mathcad есть вполне достойный русский аналог под названием SMath (www.smath.com), который может быть успешно применен для решения многих математических и инженерных задач в рамках импортозамещения. Плюс еще и в том, что программа SMath работает не только под управлением Windows (как Mathcad), но и под управлением альтернативных бесплатных операционных систем. Да и само ядро – распространяется свободно.

Итак, давайте скачаем SMath, установим его на компьютере (а это делается за пару минут) и решим задачу чеховского новогоднего визитера самым простым способом, а именно методом ближайшего соседа, алгоритм которого относится к группе жадных алгоритмов [5]: из очередной точки человек отправляется в ближайшую точку, где он еще не был. Попутно отметим некоторые отличия SMath и Mathcad. Вернее, не попутно, а главным образом!

Схема центра Москвы[6] на рис. 1.1 и 1.2 была помещена в среду графического редактора Paint для того, чтобы, во-первых, можно было прочертить прямые линии, отмечающие маршрут воздушного такси или квадрокоптера над Москвой[7], и, во-вторых, чтобы оцифровать карту – получить координаты точек в пикселах, намеченных для визитов. Для этого к очередной точке подводится курсор, координаты которого прописываются в левом нижнем углу Paint. Эти координаты переносились в векторы X и Y (см. рис. 1.3, а таже рис. 10.9 в главе 10). От значений элементов вектора Y отнималась константа – число 1456. Это высота в пикселах наших двух картинок, помещенных в среду Paint для оцифровки. Дело в том, что у Paint начало координат находится в левом верхнем углу картинки, а на наших будущих графиках – в левом нижнем. Вычитание константы 1456 возвращала координатам их традиционный формат в математике.

Мы будем реализовывать алгоритм ближайшего соседа следующим образом. Сначала введем номер точки, с которой будет начинаться маршрут (рис. 1.3.). Затем зададим векторы X, Y с координатами точек (рис.1.3) и т.д. Рассмотрим подробно выполнение каждой части программы с помощью SMath.

На рис. 1.14 показано начало расчета – ввод номера точки, с которой будет начинаться маршрут визитера (мы задачу несколько обобщим и будем считать, что поездка может начаться из любой точки, а точек может быть любое количество). Индекс $start$ у переменной i вводится через точку: i.start – точно так, как и у старой доброй[8] версии Mathcad (15 и ниже). Шрифт переменной с прямого на курсив меняется автоматически. Это подчеркивает тот нюанс, что это не встроенный, а пользовательский объект. В Mathcad Prime при вводе текстового индекса (части имени переменной, сдвинутой вниз) от разделительной точки отказались. Там ввод индекса ведется через нажатие специальной кнопки с надписью a_2. Повторное нажатие этой кнопки прекращает ввод индекса. Это позволяет иметь в расчете такие экзотические переменные, как H_2O, H_2SO_4 и др. Желательно иметь такую возможность и в среде SMath.

Сразу отметим, что в среде SMath нет маткадовской системной переменной ORIGIN, которая задает начальный номер элементов в векторе, строк и столбцов в матрице и которая по умолчанию (заводские настройки) равна нулю. В среде SMath нумерация массивов ведется только от единицы, которую

посередине. В произведениях Толстого и Чехова прослеживается взвешенная оценка этого явления, растянутого на века.

[6] Во времена Чехова это была фактически вся Москва.

[7] Заметим вскользь, что полет по некоторым прямым на рис. 1.2 проходит над Кремлем. А это недопустимо по известным причинам, тем более, в наше неспокойное время. Поэтому эти прямые придется несколько изогнуть.

[8] Многие пользователи по ряду причин не переходят на новую версию Mathcad – на Mathcad Prime из-за его необычного интерфейса и отсутствия некоторых возможностей старой версии.

нельзя сменить на нуль или другое целое число – положительное или отрицательное. Хорошо это или плохо – разговор особый. В нашем расчете переменной i_{start} присваивается единица – визитер начинает свои поездки из первой точки – из дома на Зубовском бульваре (литера А на рис. 1.1 и 1.2). Потом мы изменим значение i_{start} с единицы на другое значение и посмотрим, что из этого выйдет.

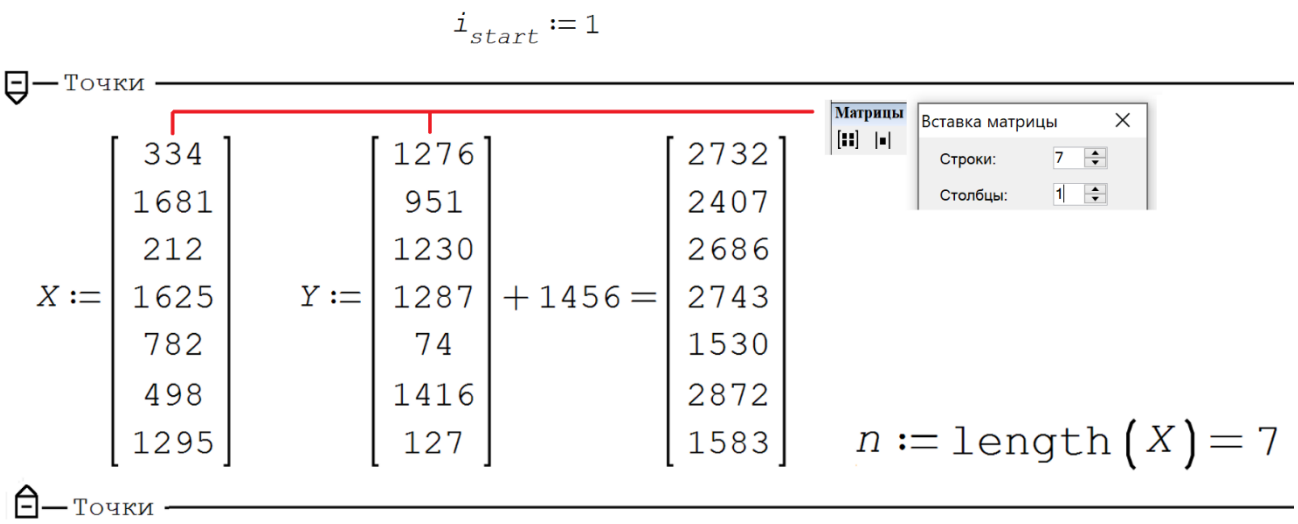

Рис. 1.3. Начало расчета пути коммивояжера – чеховского визитера

Итак, в самом начале расчета мы нажимаем клавишу I на нижнем регистре и получаем то, что показано на рис. 1.4. Пакет SMath в выпадающем списке перечисляет все встроенные и пользовательские конструкции, имена которых начинаются на эту букву. Это очень удобно и этого нет в Mathcad. Эту опцию при желании можно отключить через команду (переключатель) меню «Вид / Динамическая помощь ввода».

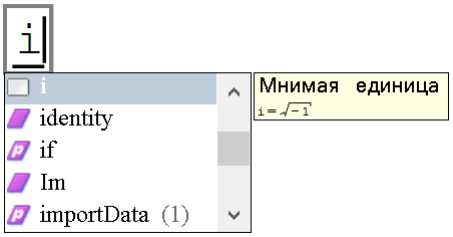

Рис. 1.4. Начало списка переменных и функций, имя которых начинается на букву i

При создании документа рекомендуется сразу работать с так называемыми *областями* (areas), куда помещают часть расчета. Такую область можно свернуть, чтобы одновременно видеть участки документа, далеко отстоящие друг от друга: ввод исходных данных и ответ, например. Без областей для этого пришлось бы крутить колесика мышки, что не очень удобно при объемных расчетах. А так – щелкнул мышкой по квадратику с минусом и область свернулась. Минус превратился в плюс и перед глазами то, что нужно – исходные данные и ответ. Щелкнул еще раз уже по плюсу, а не по минусу – область раскрылась, показывая, что находится там внутри. Областям можно дать название (что мы и сделали) и при необходимости «запаролить»– защитить от посторонних взглядов[9]. В среде SMath в одну область можно вставить другую, вложенную область, что позволяет структурировать расчет.

[9] На форумах пользователей SMath часто спрашивают, как раскрыть «запароленную» область, если забыл пароль.

Этого нет в среде Mathcad. Ниже на рис. 1.14 показана «главная» область с именем `Расчет`, в которую вложены четыре другие именованные области, где хранятся пользовательские функции, о которых речь пойдет ниже.

Авторы немного пофантазировали на своих рисунках. Они нарисовали плюсы и минусы с указателями в квадратиках не только в начале областей, но и в их концах – так, как это делается в Mathcad 15 и чего нет ни в Mathcad Prime, ни в SMath. Кроме того, вложенные области авторы сдвинули вправо для зрительной фиксации структуры расчета. Что-то подобное предусмотрено в пакете Maple. Об этой фантазии (о пожелании) авторы сообщили соавтору – разработчику SMath, который обещал реализовать в очередной версии пакета такую удобную возможность.

В первых двух операторах вложенной области с именем `Точки` на рис. 1.3 в переменные *X* и *Y* вводятся векторы с семью элементами. Это делается так же, как и в Mathcad 15 – через нажатие кнопки с изображением квадратной матрицы два на два элемента на панели **Матрицы**, показанной на рис. 1.5 ниже (эта панель расположена в правом верхнем углу рабочего окна SMath; если подвести курсор мыши к копке, то появится подсказка о ее функции). После этого в появившемся диалоговом окне ведется запрос размеров матрицы – число строк и столбцов. Но, увы, в этом окне нет кнопок для удаления строк и столбцов или для их добавления. Эту недоработку нужно, конечно, исправить, а еще лучше сделать так, как это предусмотрено в Mathcad Prime, где есть возможность выбора размера будущей матрицы или вектора протяжкой мыши (примерно так, как это делается в Word и Excel) и имеются кнопки для удаления лишнего и вставки дополнительного. Строки и столбцы матрицы SMath приходится удалять довольно хитрыми способами. Но добавить элементы в уже существующий вектор или матрицу с одной строкой (горизонтальный вектор) просто – достаточно нажать кнопку запятой или точки запятой в зависимости от того, на что настроен пакет SMath.

Рис. 1.5. Панель операторов Матрицы

Отметим, что пакет SMath позволяет также скачивать данные с файла на диске. И вообще, есть программы, которые по щелчку мыши по картинке записывают на диск координаты точек в виде текстового файла. Потом этот ряд пар чисел можно автоматически перенести в среду SMath и работать дальше с таким массивом данных.

На рисунке 1.6 показана одна из раскрытых областей с функцией пользователя с именем *M*, которая возвращает квадратную матрицу, содержащую расстояния между точками пути коммивояжера. Подобную матрицу (табличку) когда-то давно можно было видеть в загородных автобусах. Пассажир, входящий в автобус, сообщал кондуктору, куда он едет. Кондуктор смотрел на такую табличку и называл пассажиру стоимость проезда. По диагонали такой автобусной матрицы стояли прочерки. У нас же там будет храниться значение бесконечности, запрещающее коммивояжеру идти из точки в ту же самую точку.

В программе на рис. 1.6 можно отметить три атрибута программирования: *программные блоки*, *локальные переменные* и *программные управляющие конструкции*. Программные блоки (конструкции begin-end в языке Pascal) в среде SMath фиксируются вертикальной чертой, вводимой с помощью кнопки line (рис. 1.7). Переменные в программном блоке автоматически становятся локальными (видимыми только в программе), если их вводят в теле программы – справа от вертикальной черты. У нас таких переменных три: `i`, `j` и `M`. Отметим также, что в среде SMath для ввода и глобальных, и локальных переменных используется один и тот же оператор «двоеточие и равно» (:=). При этом нужно нажать только клавишу «двоеточие», а равно допишется автоматически. А еще лучше нажимать не клавишу «двоеточие», а клавишу «равно». Если переменная свободна, то знак «равно» превратиться в знак «двоеточие и равно», и пользователю будет дана возможность ввести в переменную нужное значение. Если же переменная, куда собираются ввести какое-то значение, уже что-то хранит, то нажатие клавиши «равно», приведет к тому, что это что-то будет выведено «на печать». В Mathcad для локального присваивания используется другой оператор – стрелочка влево. А это неудобно и лишено какой-либо логики. Содержимое заполненной матрицы расстояний (округлено до целых) показано вверху на рис. 1.13.

Функция M

$$M(X, Y) := \begin{|l} \textbf{for } i \in [1..n] \\ \quad \textbf{for } j := 1, j \le n, j := j + 1 \\ \quad\quad \textbf{if } i \ne j \\ \quad\quad\quad M_{i\,j} := \sqrt{\left(X_i - X_j\right)^2 + \left(Y_i - Y_j\right)^2} \\ \quad\quad \textbf{else} \\ \quad\quad\quad M_{i\,j} := \infty \\ M \end{|l}$$

Функция M

Рис. 1.6. Функция с именем M, возвращающая квадратную матрицу расстояний

Набор программных управляющих конструкций, изменяющих естественный порядок выполнения операторов (слева-направо и сверху-вниз), в среде SMath примерно такой (рис. 1.7), как и у Mathcad. Нет только оператора `Return`. Оператор же `on error` (обработка ошибки) в среде SMath имеет имя `try` (попытка). Такое же переименование случилось и в среде Mathcad Prime.

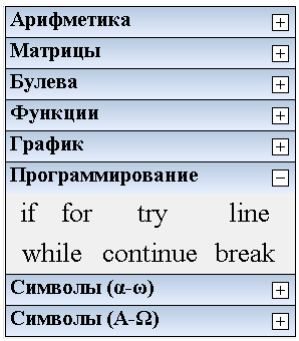

Рис. 1.7. Панель операторов Программирование

Цикл с параметром `for` в среде SMath реализован в двух вариантах – с тремя и с четырьмя операндами – см. рис. 1.8.

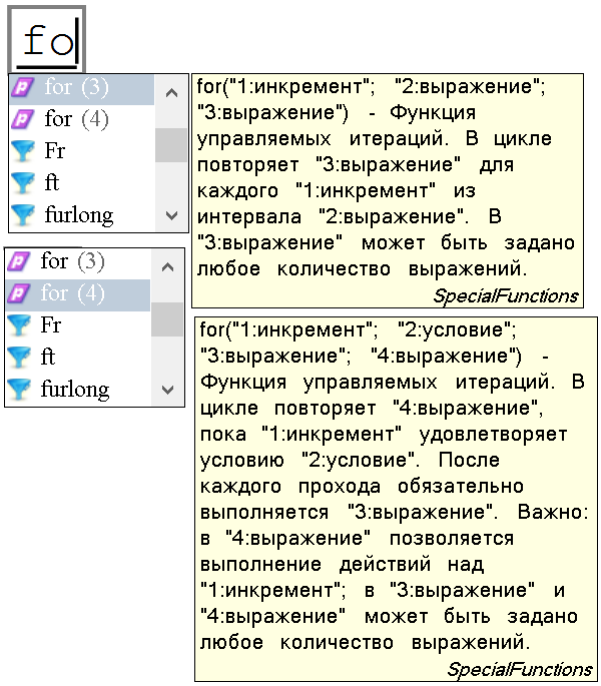

Рис. 1.8. Две формы оператора for

В программе на рис. 1.6 для иллюстрации были задействованы оба варианта цикла `for`, хотя можно было ограничится только первым вариантом с тремя параметрами. Но есть случаи, где второй вариант предпочтительней. Это имеет место тогда, когда переводят на язык SMath программы, написанные на языке C, например[10], где используется четырехаргументная форма записи цикла `for`. Кроме того, такая форма делает легальным прием изменения значения параметра цикла в теле цикла.

В первом (наружном) цикле `for` стоит имя параметра цикла и знак «принадлежит». Затем тут обычно вставляют так называемые переменные диапазона (Range), использующие арифметическую прогрессию. Заготовки этой прогрессии находятся в панели **Матрицы** (рис. 1.5) в двух вариантах – с указанием второго значения переменной диапазона [1.1, 1.2.. *n*] и без указания оной [1.. *n*], как на рис. 1.6. Во втором случае, когда не указывается шаг, второе значение переменной диапазона будет на единицу больше первого (шаг будет единичным). Если первое значение переменной диапазона меньше последнего, шаг будет равен –1.

Сразу подчеркнем, что переменная диапазона в среде SMath – это полноценный вектор, к которому применимы все векторные операции – см. примеры на рис. 1.9. Этого, увы, нет в Mathcad. Там при необходимости придется проделывать особые сложные процедуры, чтобы конвертировать переменную диапазона (Range) в полноценный вектор.

[10] Предлагаем читателю найти в интернете и реализовать в SMath программу на языке C, решающую задачу коммивояжера более совершенными методами – методом отжига, муравьиным, генетическим и др.

$$V := [2; 1{,}5 .. 0] = \begin{bmatrix} 2 \\ 1{,}5 \\ 1 \\ 0{,}5 \\ 0 \end{bmatrix} \qquad V_2 = 1{,}5 \qquad \text{reverse}(V) = \begin{bmatrix} 0 \\ 0{,}5 \\ 1 \\ 1{,}5 \\ 2 \end{bmatrix} \qquad \min(V) = 0 \qquad \max(V) = 2$$

Рис. 1.9. Примеры работы с переменной диапазона как с полноценным вектором

А вот еще одно полезное свойство SMath, о котором, честно говоря, нужно было сказать в самом начале нашего повествования. Из рис. 1.9 видно, что в дробных числах используется запятая, а не точка, а разделителем аргументов в функциях служит не запятая, а точка с запятой. Точку, запятую и другие «знаки препинания» в расчетах можно выбрать, вызвав из меню Сервис диалоговое окно Опции, показанное на рис. 1.10. Тут сразу вспоминается русская версия Excel, с запятой в числах и точкой с запятой в списках. Запятая в числах программы удобна при оформлении статей, когда отечественные редакторы требуют использовать запятую, а не точку в нецелых числах.

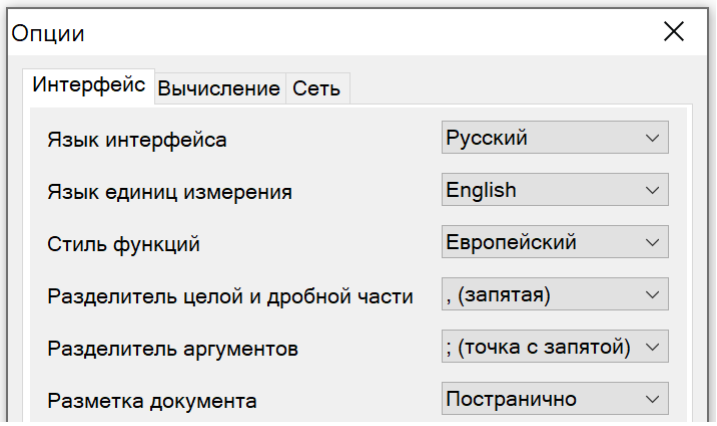

Рис. 1.10. Меню Опции

На рис. 1.11 показана функция пользователя, возвращающая индекс минимального элемента вектора. Используется простой перебор элементов вектора. В Mathcad работу по поиску нужных индексов в массивах (в векторах и матрицах) проводит встроенная функция match с двумя аргументами, которая возвращает искомые массивы значений индексов. Наша же пользовательская функция на рис. 1.11 возвращает скаляр, даже в том случае, когда вектор-аргумент содержит несколько минимальных значений. Нам это подходит, но функция match и другие подобные были бы нелишними и в среде SMath.

На рис. 1.11 после создания функции она сразу вызывается для проверки ее работоспособности. Показано, что в векторе X (абсциссы точек путешествия чеховского визитера – см. рис. 1.1–1.3) минимальный элемент 212 (тут задействована встроенная функция min) стоит на третьем месте. Если б таких элементов было более одного, то встроенная функция Mathcad match вернула бы вектор. У нас же функция i_{min} в такой ситуации вернет скаляр – последнее значение в ряду индексов минимальных значений.

Функция i.min

$$i_{min}(V) := \left| \begin{array}{l} V_{min} := \min(V) \\ \text{for } i \in [1..\text{length}(V)] \\ \quad \text{if } V_i = V_{min} \\ \quad\quad i_{min} := i \\ i_{min} \end{array} \right. \qquad X = \begin{bmatrix} 334 \\ 1681 \\ 212 \\ 1625 \\ 782 \\ 498 \\ 1295 \end{bmatrix} \qquad i_{min}(X) = 3$$

Функция i.min

Рис. 1.11. Функция, возвращающая индекс минимального элемента вектора

Итак, вспомогательные функции созданы и можно вводить в расчет основную функцию – функцию *Way* (путь), возвращающую вектор, содержащий номера точек, по которым должен пройти коммивояжер, руководствуясь алгоритмом ближайшего соседа – см. рис. 1.12. Она довольно проста и не требует особых пояснений. В ней два цикла `for`, второй из которых вложен в первый (см. также рис. 1.6). Во вложенном цикле формируется вектор *S*, хранящий расстояния от текущей *j*-й точки до остальных точек маршрута. Индекс минимального элемента этого вектора, найденный с помощью функции i_{min}, становится номером очередной точки пути коммивояжера, после выполнения оператора $Way_i := i_{min}(S)$. После этого соответствующим элементам матрицы расстояний от *i*-й до *j*-й точки присваиваются значение бесконечности для того, чтобы коммивояжер не прошелся по этому маршруту еще один раз. На рис. 1.13 можно видеть, что содержится в матрице расстояний *M* до и после реализации алгоритма ближайшего соседа.

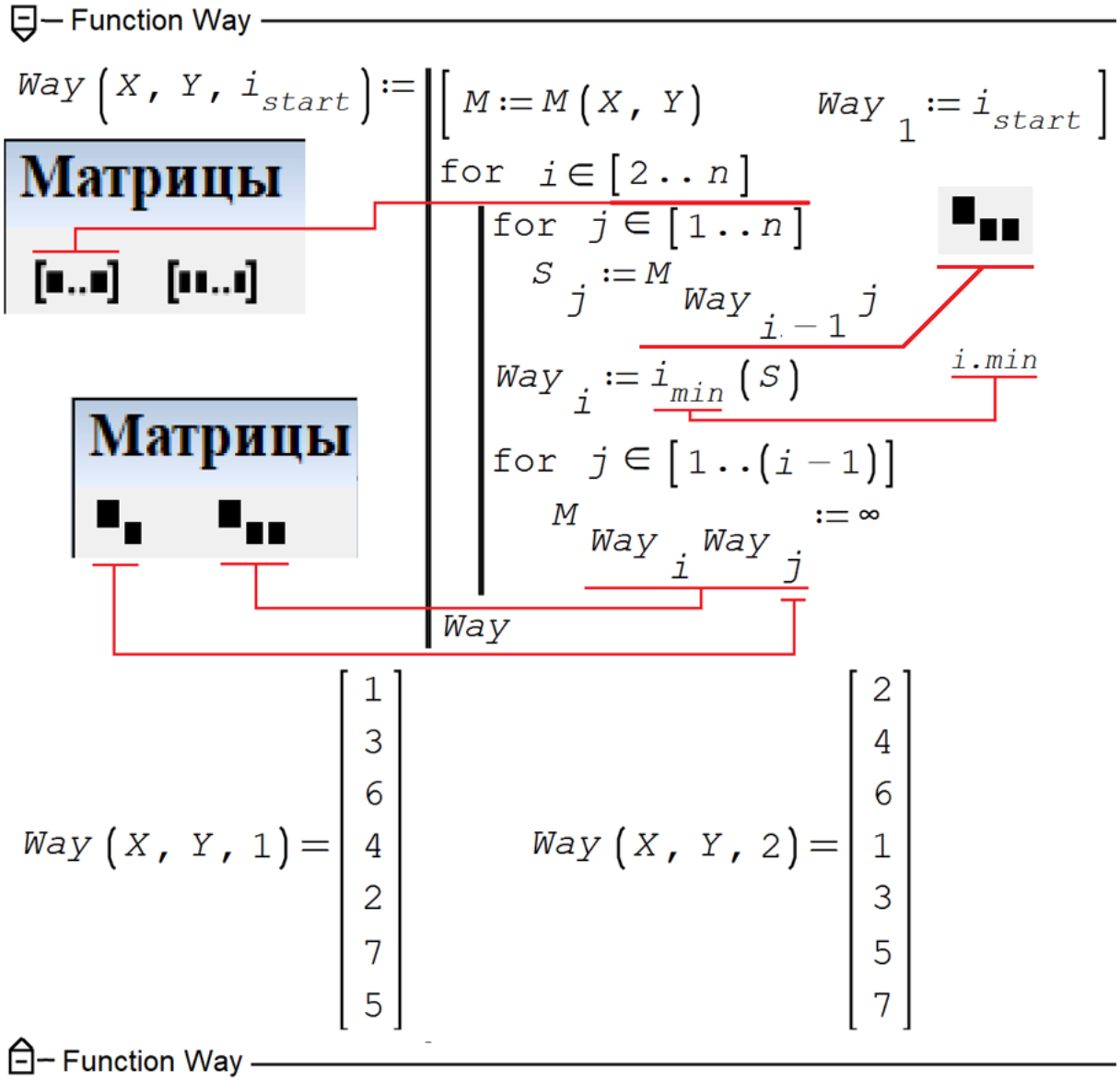

Рис. 1.12. Программа решения задачи коммивояжера методом ближайшего соседа

На рис. 1.12 показаны также примеры вызова функции *Way* при двух разных стартовых точках (см. также рис. 1.15). Если в программе на рис. 1.12 слово *Way* в конце программы заменить на букву M, то будет возвращена матрица с исходными «бесконечностями» по главной диагонали и с добавленными «бесконечностями» после реализации метода ближайшего соседа.

$$\begin{bmatrix} \infty & 1386 & 130 & 1291 & 1283 & 216 & 1498 \\ 1386 & \infty & 1495 & 341 & 1256 & 1271 & 910 \\ 130 & 1495 & \infty & 1414 & 1289 & 341 & 1546 \\ 1291 & 341 & 1414 & \infty & 1477 & 1134 & 1206 \\ 1283 & 1256 & 1289 & 1477 & \infty & 1372 & 516 \\ 216 & 1271 & 341 & 1134 & 1372 & \infty & 1515 \\ 1498 & 910 & 1546 & 1206 & 516 & 1515 & \infty \end{bmatrix}$$

$$\begin{bmatrix} \infty & 1386 & 130 & 1291 & 1283 & 216 & 1498 \\ \infty & \infty & \infty & \infty & 1256 & \infty & 910 \\ \infty & 1495 & \infty & 1414 & 1289 & 341 & 1546 \\ \infty & 341 & \infty & \infty & 1477 & \infty & 1206 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 1271 & \infty & 1134 & 1372 & \infty & 1515 \\ \infty & \infty & \infty & \infty & 516 & \infty & \infty \end{bmatrix}$$

Рис. 1.13. Содержимое матрицы расстояний до (вверху – первая стартовая точка) и после (внизу) реализации алгоритма ближайшего соседа

На рисунке 1.5 на панели **Матрицы** видны две кнопки для обращения к элементам векторов и матриц. В среде Mathcad для этого предназначена одна кнопка, а индексы матрицы отделяются друг от друга не пробелом, а запятой. Запятую в среде SMath тоже можно поставить в этом месте, но она затем пропадет, оставив два индекса матрицы. Функция на рис. 1.12 – это маленький шедевр в плане работы с индексами – текстовыми (i_{min}, i_{start},) и числовыми – с операторами, возвращающими элемент вектора или матрицы. Некоторые операторы у нас «трехэтажные», когда, например, первый индекс матрицы M являются индексом вектора Way. Из-за этого многим студентам долго не удается воспроизвести правильно программу-функцию, показанную на рис. 1.12.

На рис. 1.14 показан созданный расчет с четырьмя вложенными свернутыми областями, хранящими данные по точкам и вышеописанные три пользовательские функции. Остается только построить график и анимировать его. Для этого в расчет вводятся операторы, показанные внизу рис. 1.14. Сразу предупреждаем, что технология построения графиков – плоских и объемных в среде SMath кардинально отличается от того, что заложено в Mathcad.

Рис. 1.14. Операторы, необходимые для создания графика и анимации движения коммивояжера

На рис. 1.14, во-первых, формируется матрица XY с n строками и с такими пятью столбцами: вектор X, вектор Y и три элемента оформления кривой – символ, размер и цвет линии. Это будут точки (".") размером 15 единиц и с красным ("red") цветом. Кроме точек там можно использовать крестики (буква икс или "+"), кружочки (буква о) и звездочки ("*"). Использование других знаков препинания и букв (русских и латинских) приводит к тому, что они будут прописаны немного правее и ниже заданного места на графике. Это очень удобно, и этого нет у пакета Mathcad. Мы этим воспользуемся, когда будем приписывать номер точки, через которую проходит коммивояжер (предпоследний оператор на рис. 1.14).

В среде Mathcad анимацией управляет системная переменная FRAME. В среде же SMath эту работу выполняет переменная t, которую нужно сделать аргументом анимируемых объектов. У нас это $P(t)$ и $WAY(t)$ (не Way!). Номера кадров анимации должны храниться в векторе или в переменной диапазона. У нас это будет переменная $tRange$. Именно она становится таковой после того, как ее выбрали для управления анимацией в диалоговом окне на рис. 1.17. К этому окну можно добраться, нажав правую кнопку мыши на активированном графике. А вставляется график в рабочий документ через команды меню, показанные на рис. 1.16. Частота кадров анимации и другие ее параметры задаются через диалоговое окно, показанное на рис. 1.18.

На рисунке 1.15 показаны пути чеховского визитера из двух разных стартовых точек – из первой (дом визитера на Зубовском бульваре) и из второй (Лефортово – см. также рис. 1.12). При анимации (а ее можно увидеть в пункте 6 здесь https://community.ptc.com/t5/Mathcad/Mathcad-vs-SMath/td-p/802499) синяя прямая линия выходит из начальной точки (первой или второй) и «обходит» другие красные точки, левее и ниже которых будут появляться их номера.

Фигурная скобка, охватывающая параметры графика и находящаяся под графиком (рис. 1.15), вставляется через нажатие соответствующей кнопки в панели инструментов **Функции**. После этого появится требуемая фигурная скобка с двумя аргументами (местодержателями). Чтобы добавить

третий, четвертый и другие дополнительные аргументы, нужно нажать клавишу запятой или точки с запятой – символа для разделения элементов в списке.

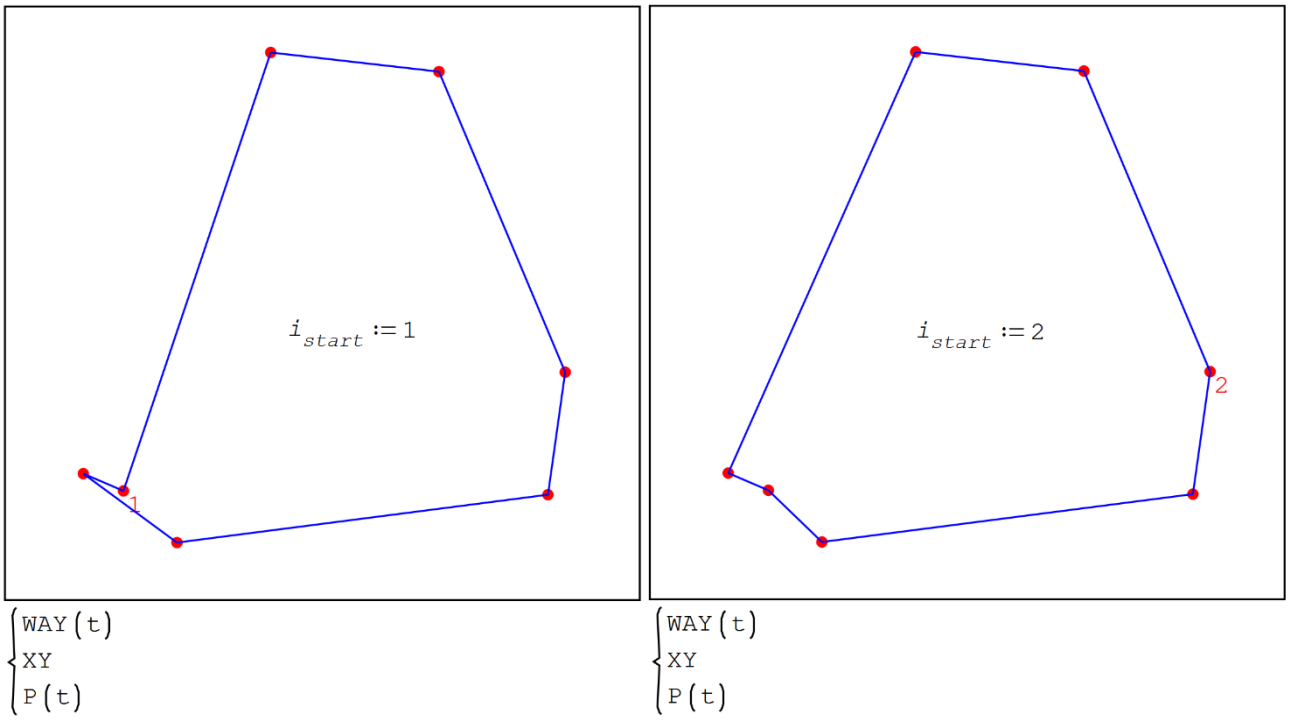

Рис. 1.15. Путь визитера из рассказа Чехова из двух разных стартовых точек

Для минимизации дины маршрута наш чеховский визитер должен был с Зубовского бульвара (точка 1) поехать не в Лефортово (точка 2), а в Хамовники (3), затем к Калужским воротам (6), далее на Нижегородский вокзал (4), а потом уж в Лефортово (2). Далее его маршрут должен был бы пройти через Сокольники (7), Крестовскую заставу (5) и закончится дома на Зубовском бульваре (1). Но правый график на рис. 1.15 показывает более короткий маршрут: в Хамовниках нужно было не жадничать: ехать не к ближайшим Калужским воротам, а к более отдаленной Крестовской заставе. Оптимальное решение задачи коммивояжера в принципе не должно зависеть от начальной точки. В нашем же случае это не так – см. рис. 1.15.

Но все это относится к маршруту, состоящему из отрезков прямых (путешествие на вертолете – см. рис. 1.2). В реальности же приложение на компьютере или смартфоне должно было бы предложить человеку, заказывающему на такси круговой маршрут (см. рис. 1.1), оптимизировать поездку, решив задачу коммивояжера, учитывающую не только местонахождение остановок, но и дорожную обстановку, минимизирующую не только пройденное расстояние, но и стоимость поездки.

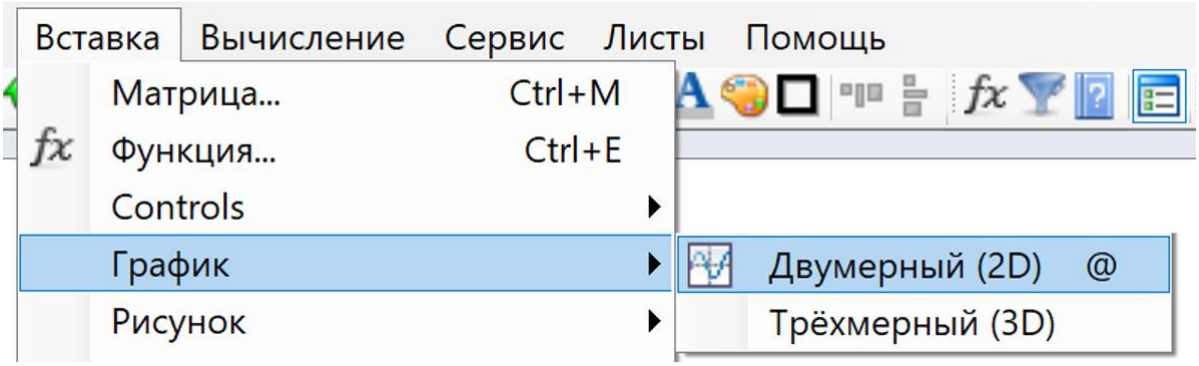

Рис. 1.16. Команды меню вставки двумерного графика

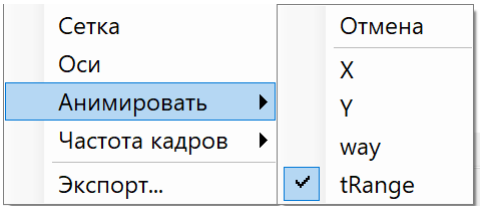

Рис. 1.17. Задание переменной анимирования

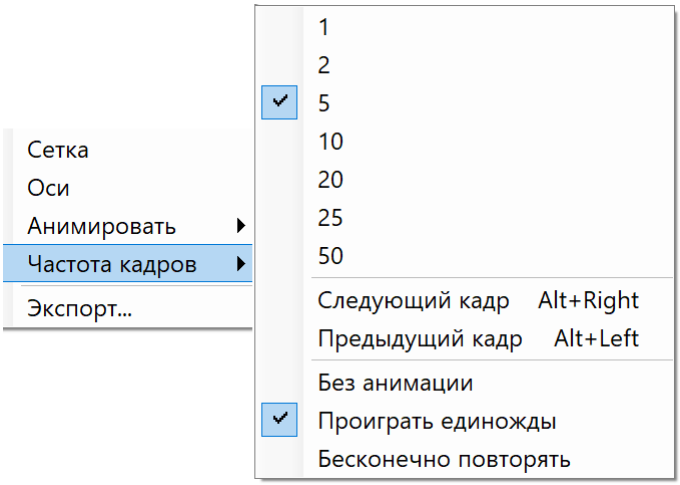

Рис. 1.18. Задание параметров анимации

На рисунке 1.19 показаны три кадра анимации движения коммивояжера по 50 случайным точкам, которые можно ввести в векторы *X* и *Y* через функцию, возвращающую случайные числа.

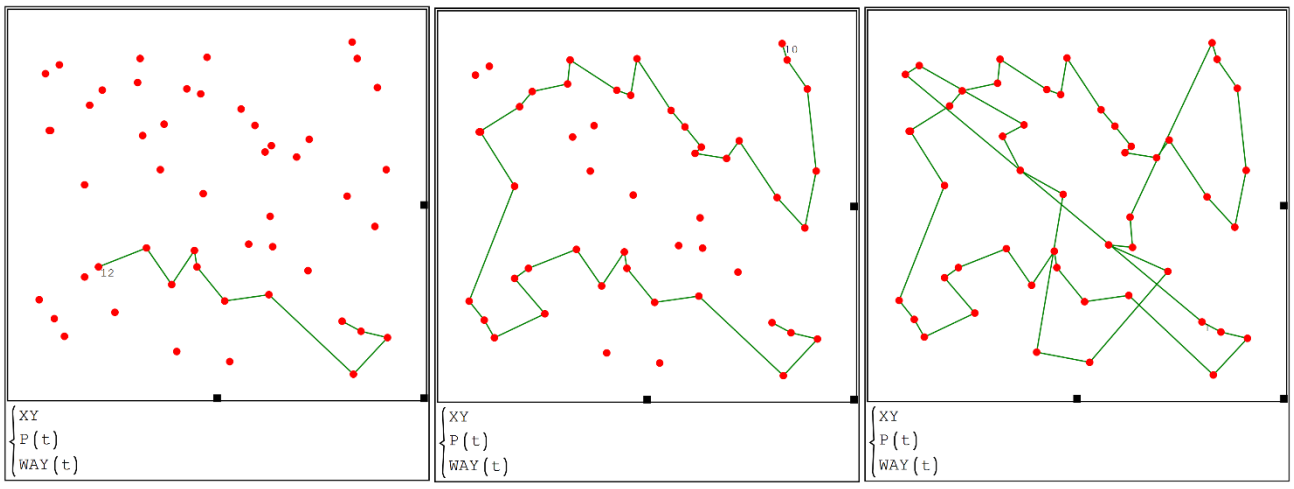

Рис. 1.19. Три кадра анимации движения коммивояжера по пятидесяти случайным точкам

Если в программе на рис. 1.12 функцию `min` заменить на функцию `max`, а знак бесконечности заменить на знак бесконечность с минусом впереди, то будет реализован *алгоритм дальнего соседа* – см. рис. 1.20. Можно перебрать все точки старта и выбрать еще более длинный маршрут, чем тот, по какому жена отправила мужа с визитами в рассказе Чехова.

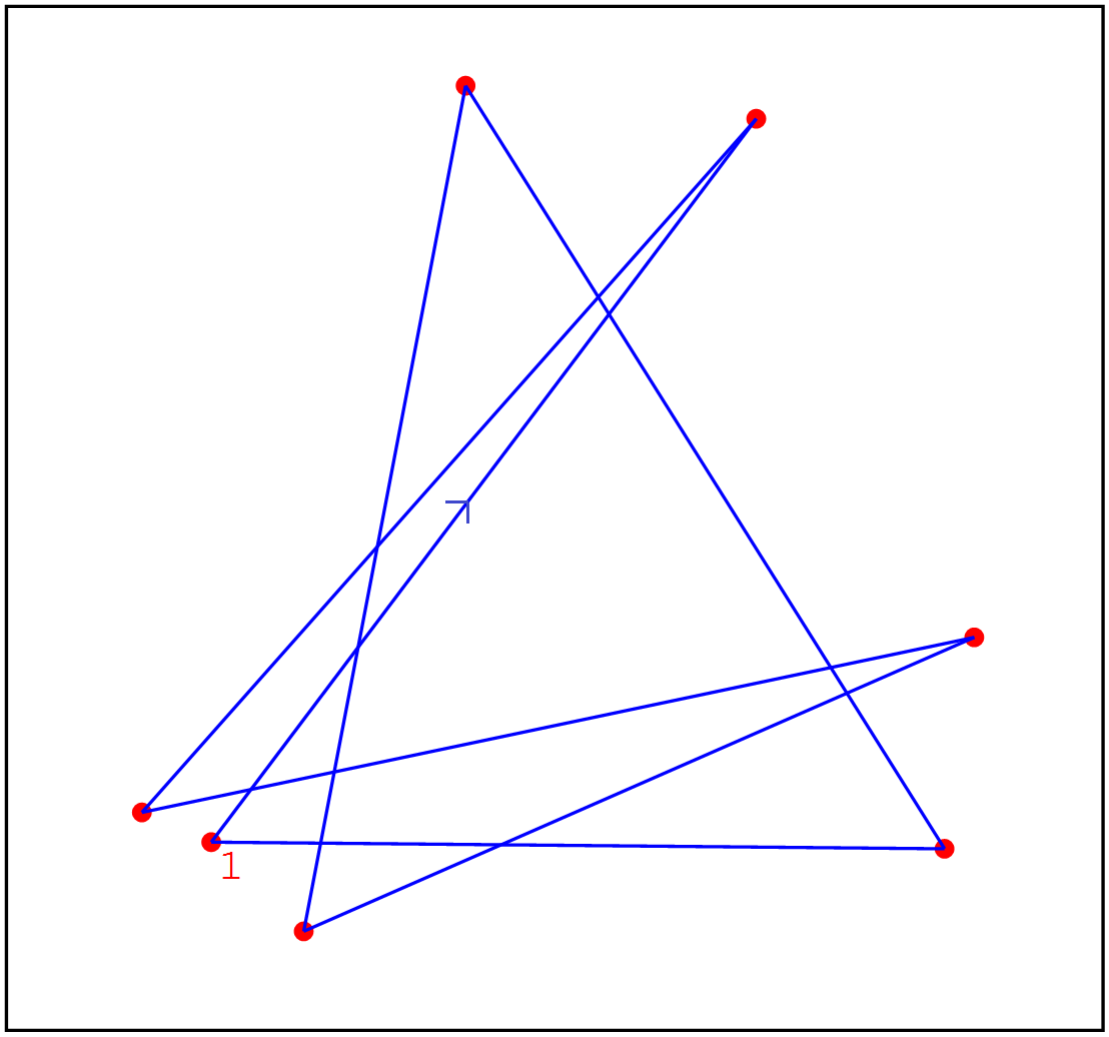

Рис. 1.20. Путь визитера по алгоритму дальнего соседа

На рисунке 1.21 показана картинка из статьи Википедии «Задача коммивояжера» с пояснением того, к чему приводит попытка решения задачи коммивояжера методом прямого перебора. Фрау посылает своего герра облететь на самолете знакомых в 15 городах Германии… Кстати, юго-западнее Штутгарта расположен городок Баденвайер, где умер Чехов – автор рассказа, ставшего стартовой точкой стартовой главы данного учебного пособия.

Это оптимальный маршрут коммивояжёра через 15 крупнейших городов Германии. Указанный маршрут является самым коротким из всех возможных 43 589 145 600 вариантов.

Рис. 1.21. Решение задачи коммивояжера для 15 городов Германии.

На сайте [7] собрана обширная информация о задаче коммивояжера. В частности, там можно найти путь странствующего торговца по всем населенным пунктам Италии (рис. 1.22), других стран и даже всей Земли. Применяются, конечно, более совершенные алгоритмы, исключающие петли и по-настоящему минимизирующие путь. Но опять же не доказано, что это будет самый-самый короткий маршрут. Подобные алгоритмы мы рекомендуем читателям реализовать в среде SMath – создать новую функцию `Way`. Все такие алгоритмы являются эвристическими, использующими практические методы, не являющимися гарантированно точными или оптимальными, но достаточными для решения поставленной задачи. Но для небольшого числа точек (рис. 1.15) можно найти абсолютно точное решение даже без использования сложных алгоритмов.

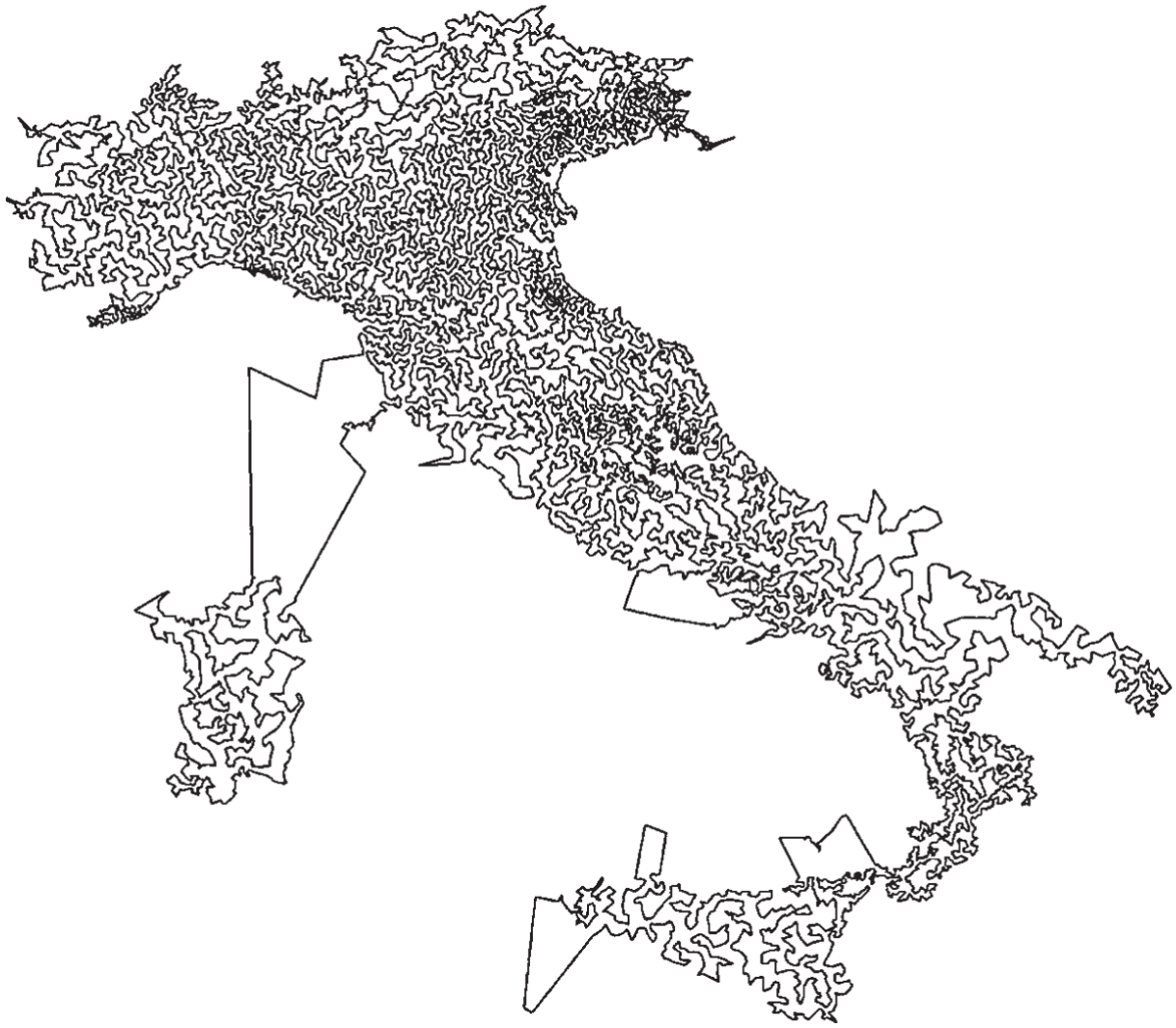

Рис. 1.22. Коммивояжер путешествует по Италии

Послесловие

В феврале 2023 года, когда работа над книгой была почти закончена, первому автору этого учебного пособия пришло сообщение из Канады от разработчика математического пакета Maple – от фирмы MapleSoft о том, что появилась бета-версия этой математической программы, основное новшество которой – это инструменты решения задачи коммивояжера. Автор тут же захотел проверить эти инструменты на задаче о чеховском визитере – см. рис. 1.23-1.25. Вот что у него получилось!

На рисунке 1.23 показано следующее:

- Команда *with*(*GraphTheory*) подгружает инструменты работы с графами в среде Maple.
- В переменную *Points* пользователь записывает матрицу с семью строками и четырьмя столбцами, хранящую информацию о поездке чеховского визитера. Для получения этих данных следует схему поездки (рис. 1.1 или 1.2) поместить, например, в среду графического редактора Paint, а затем подводить курсор к очередной точке и считывать ее координаты в пикселах. Таблица *Points* – это фактически не таблица, а простейшая релятивистская база данных с четырьмя полями (буквенная маркировка мест в городе, название этого места, его абсцисса и ордината) и с семью записями. Координаты мест в городе можно получить и

другим путем – открыть на компьютере карту города, тыкать курсором мыши в нужное место и считывать его координаты: северную широту и восточную долготу. Потом, правда, нужно будет эти углы перевести в километры на плоской карте. Можно ввести в расчет не координаты точек в виде базы данных, а квадратную матрицу с пустой главной диагональю, хранящую расстояния между точками или время (стоимость) поездки между ними (см. рис. 1.13). Если элемент такой матрицы $m_{i, j}$ хранит значение 100, то это означает, что 100 – это расстояние между i-й и j-й точками или время поездки, ее стоимость и др. В такой квадратной матрице пустой может быть не только главная диагональ, но и некоторые «боковые» элементы. Это означает, что прямая поездка между двумя указанными городами невозможна. А нужно ехать через какой-то город.

- Команда *CompleteGraph* информирует пользователя о том, что мы имеем дело с неориентированным графом, у которого 7 вершин (места в Москве) и 21 ребро (пути, соединяющие эти места). По ребру ориентированного графа «проехаться» можно только в одном направлении (улица с односторонним движением.
- Вершины и ребра нашего графа построены командой *DrawGraph*. Получился правильный семиугольник, все вершины которого расположены по кругу (*style=circle*) и соединены прямыми линиями. На этих линиях-ребрах можно при желании отметить расстояния между точками, время поездки или ее стоимость.

```
with(GraphTheory) :

Points := [ "A"  "Зубовский бульвар"   334  1276
            "B"       "Лефортово"      1681  951
            "C"       "Хамовники"      212   1230
            "D"  "Нижегородский вокзал" 1625 1287
            "E"  "Крестовская застава"  782  74
            "F"  "Калужские ворота"     498  1416
            "G"       "Сокольники"      1295 126 ] :

G := CompleteGraph( convert(Points[ .., 1], list), vertexpositions = Points[ .., [3, 4]])

                G := Graph 1: an undirected graph with 7 vertices and 21 edge(s)        (1)

DrawGraph(G, style = circle)
```

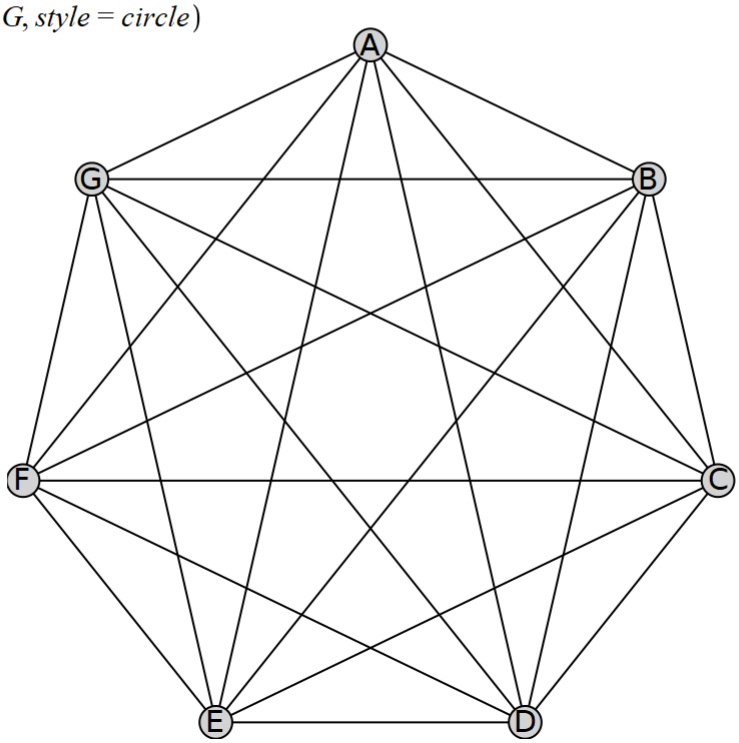

Рис. 1.23. Решение задачи коммивояжера в Maple 2023 (начало)

Функция *TravelingSalesman* (рис. 1.24) вернула, во-первых, расстояние, какое преодолел чеховский визитер (6035 пиксел), и во-вторых, его маршрут, следуя по которому это расстояние окажется минимальным. Два других оператора на рис. 1.24 строят маршрут чеховского визитера, вернее, квадрокоптера, какой он запустил. Точки совпадают с теми, какие показаны на рис. 1.1 и 1.2 с одной лишь разницей – они перевернуты по вертикали и горизонтали. Юг Москвы оказался на севере, а запад на востоке. Но это не так важно. Главное, что задача решена: с Зубовского бульвара (точка A) нужно было ехать (лететь по прямой) либо в Хамовники (C), либо к Калужским воротам (F), а далее «со всеми остановками». А чеховский визитер «поперся» аж в Лефортово (B). Минимальным маршрут будет при отправлении из любой точки в любом направлении. Но последнее утверждение окажется ложным, если по какому-то ребру можно ездить только в одном направлении.

$$W, T := TravelingSalesman(\ G, vertexpositions, startvertex = \text{"A"}\ )$$
$$W, T := 6035.03350860640, [\text{"A"}, \text{"C"}, \text{"E"}, \text{"G"}, \text{"B"}, \text{"D"}, \text{"F"}, \text{"A"}] \quad \textbf{(2)}$$

$$TG := Subgraph(\ G, Trail(T)\ )$$
$$TG := Graph\ 2:\ an\ undirected\ graph\ with\ 7\ vertices\ and\ 7\ edge(s) \quad \textbf{(3)}$$

$$DrawGraph(TG)$$

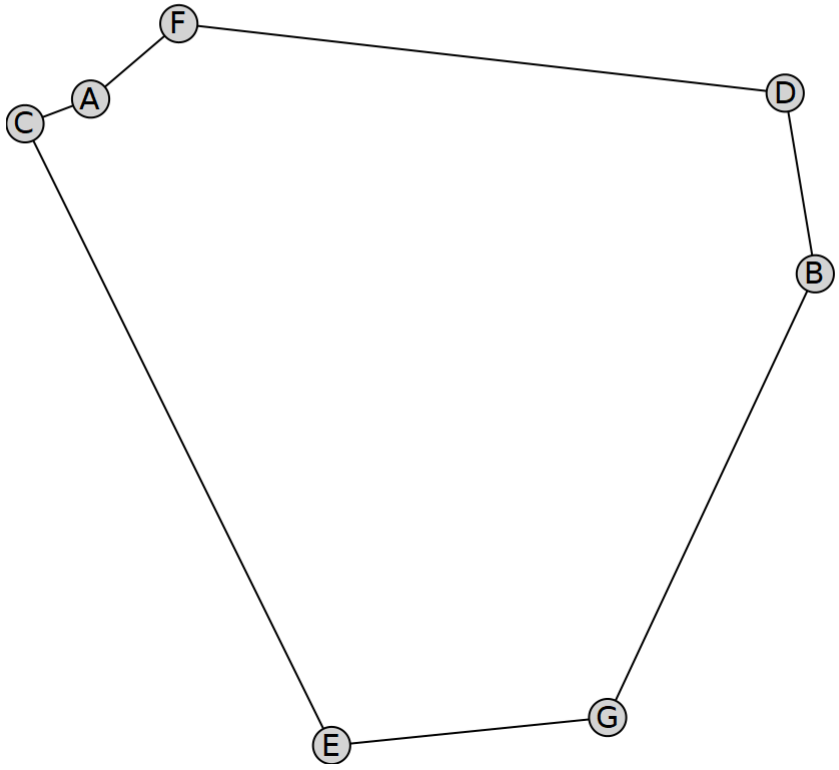

Рис. 1.24. Решение задачи коммивояжера в Maple 2023 (продолжение)

На рисунке 1.25 показан исходный граф, на котором красным жирными линиями дополнительно прорисована так называемая *гамильтонова петля* (цикл) с оптимизированным маршрутом (см. https://ru.wikipedia.org/wiki/Гамильтонов_граф).

$HighlightTrail(G, T, red)$
$DrawGraph(G, style = circle)$

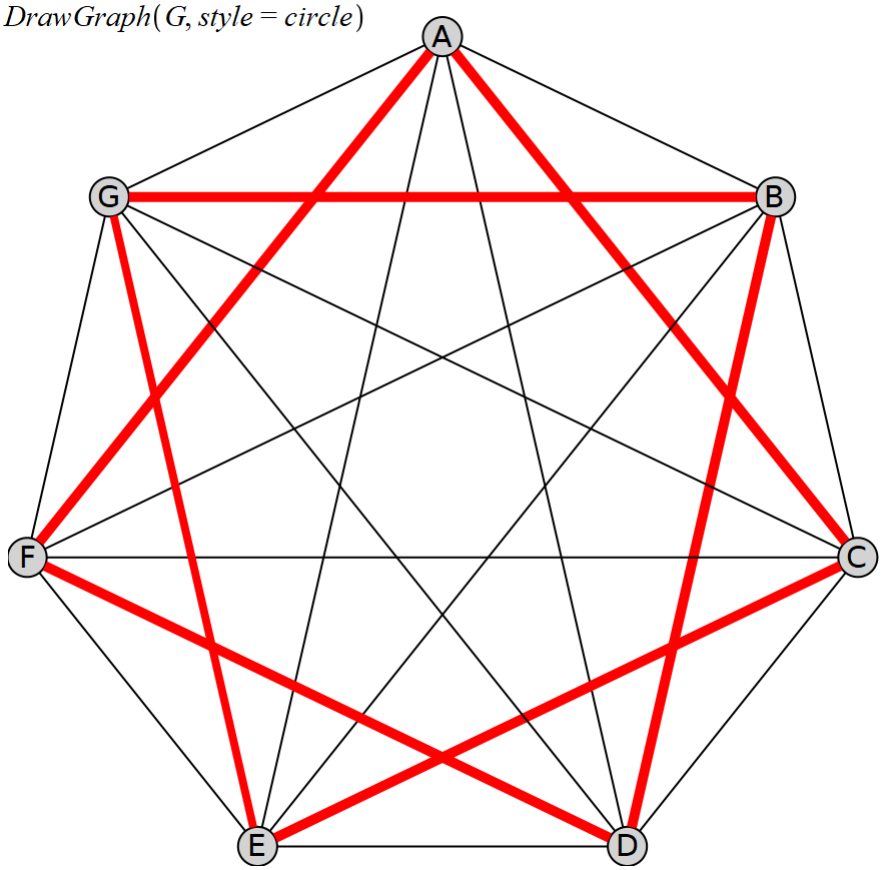

Рис. 1.25. Решение задачи коммивояжера в Maple 2023 (окончание)

Использование пакета Maple для решения задачи из чеховского рассказа – это, конечно, стрельба из пушки по воробьям (по-английски колка орехов кувалдой). Оптимальный маршрут можно было проложить, просто глядя на карту Москвы. Но при увеличении количества точек без компьютера уже не обойтись. Вот как, к примеру, может выглядеть маршрут коммивояжера и его граф при 20 точках – см. рис. 1.26.

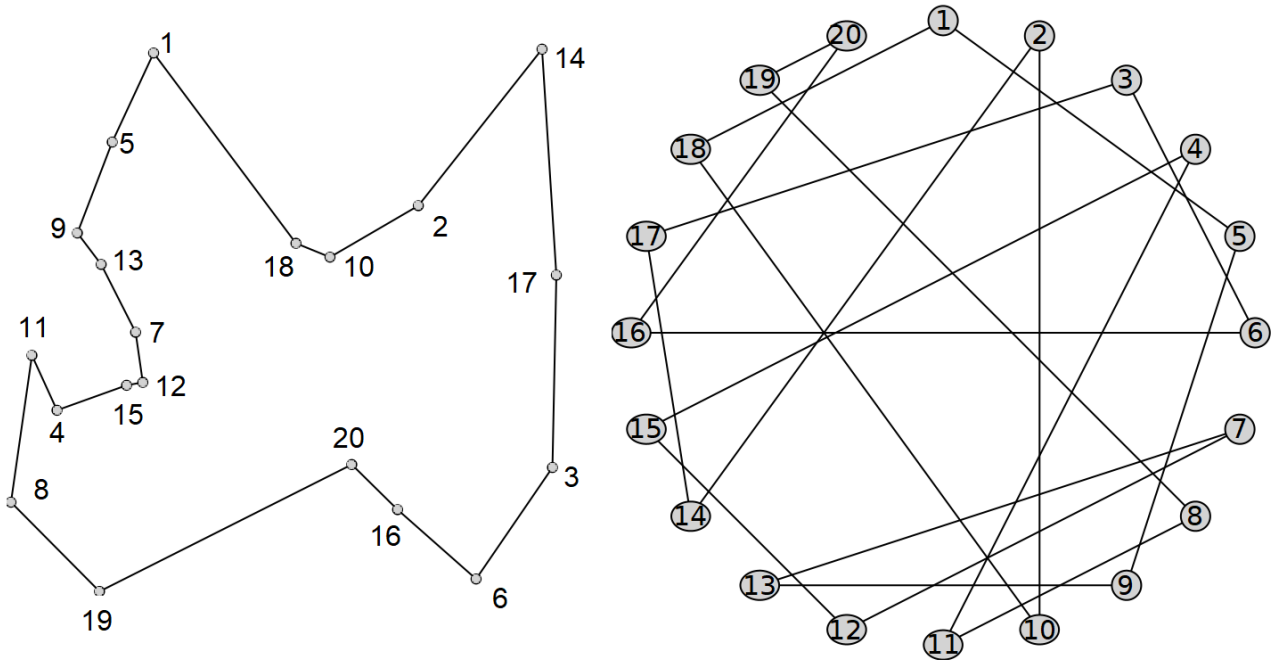

Рис. 1.26. Решение задачи коммивояжера в Maple 2023 (вариант-2)

В анонсе пакета Maple 2023 описана более сложная задача, которую без компьютера тоже не решить – облет на самолете по наикратчайшему маршруту 100 крупнейших городов Африки– см. рис. 1.27.

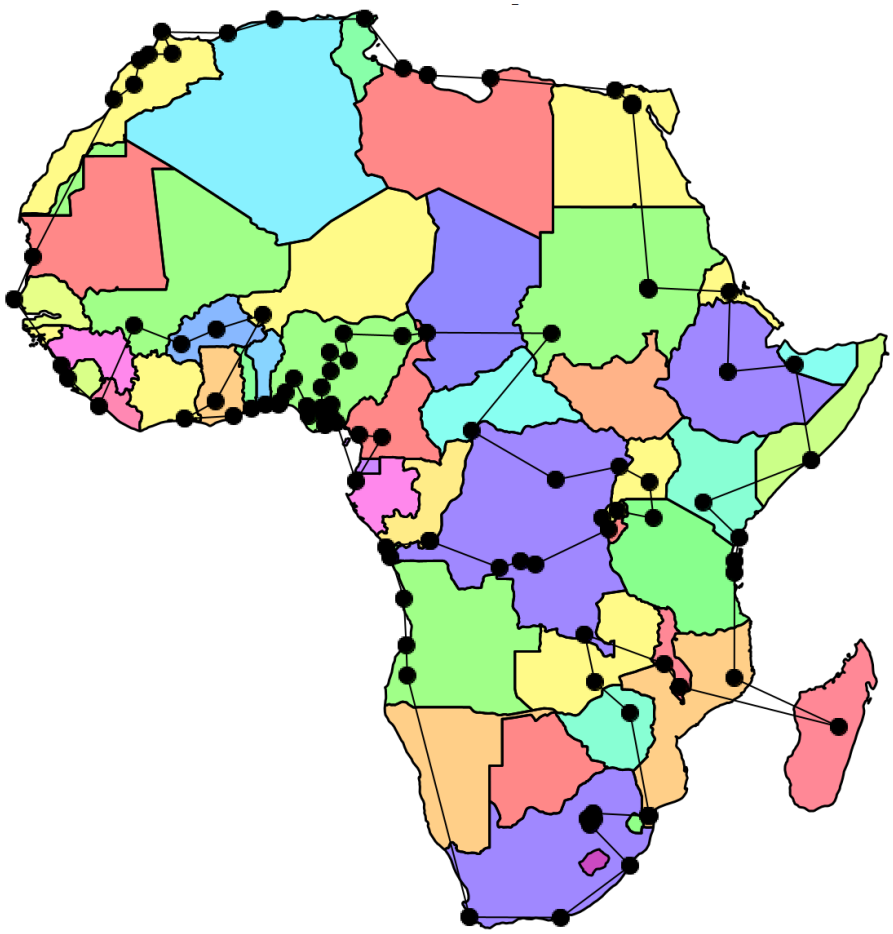

Рис. 1.27. Графическое отображение решение задачи коммивояжера в среде Maple 2023 (сто городов Африки)

Карта Африки, кстати, отсылает нас к еще об одной интересной оптимизационной задаче, связанной с теорией графов. Это задача о минимальном количестве красок, которыми можно раскрасить политическую карту мира. Доказано, что для этого достаточно только четырех цветов, что и показано на рис. 1.27. Но в научно-популярных журналах, подобных тому какой читатель держит в руках (открыл на своем компьютере, планшете, смартфоне), регулярно в первоапрельских выпусках появляются контурные карты с утверждением, что их нельзя раскрасить только в четыре цвета…

Вернемся к задаче коммивояжера.

Доказано, что перемещаться по городу с минимизацией расстояния, времени или денег, следует, комбинируя виды транспорта. Нужно ездить на велосипеде, периодически спешиваясь или закатывая велосипед в общественный транспорт (в трамвай-автобус-метро) или даже в такси. Мы это сейчас наблюдаем воочию, например, в Москве, где на бешенной скорости гоняют велосипедисты с желтыми квадратными сумками за плечами. Они развозят заказчикам продукты и готовую еду, на которую москвичи «подсели» после пандемии. В руках этих коммерческих байкеров можно увидеть смартфон, который не только показывает дорогу, но и оптимизирует маршрут, решая довольно сложную задачу коммивояжера…

Другой пример. Одна всемирно известная курьерская служба, перевозящая грузы по всему миру на кораблях, самолетах, поездах, автомашинах, велосипедах и т.д., наняла математиков и программистов

для оптимизации перевозок. Математики с программистами создали программу, которая позволила этой службе (компании) сэкономить почти два миллиарда евро в год.

В вагоне метро можно видеть своеобразный граф – схему линий со станциями – вершинами графа и с перегонами между станциями – ребрами графа. Такую схему можно открыть и на смартфоне, указав стартовую и финишную станции. Смартфон решит несложную задачу коммивояжера и найдет такую схему пересадок, при которой время в пути будет минимальным. Сюда добавляются и пешие переходы. Поэтому смартфон укажет, в какой вагон поезда метро нужно войти, чтобы пересадка заняла меньше времени. Компьютерному приложению по заказу такси (рис. 1 и 2) не хватает кнопки, нажатие на которую должно оптимизировать круговой маршрут через решение задачи коммивояжера.

Раскрытие генома живого существа требует решения довольно сложной задачи коммивояжера. С другой стороны, один из самых эффективных алгоритмов решения задачи коммивояжера – это генетический алгоритм.

Задания читателям.

1. Воссоздать в среде SMath расчет, описанный в статье.
2. Ввести в расчет 50, 100 и более случайных точек, выбрать начальную и проложить от нее маршрут коммивояжера, использующий алгоритм ближайшего соседа. Затем перебрать все точки в качестве стартовых и найти минимальный маршрут.
3. Реализовать в среде SMath более оптимальные маршруты коммивояжера, найдя их алгоритмы в интернете или в других источниках.

Литература и ссылки:

1. chehov-lit.ru/chehov/text/novogodnyaya-pytka.htm
2. https://knigavuhe.org/book/novogodnjaja-pytka/
3. http://az.lib.ru/t/tolstoj_lew_nikolaewich/text_0090.shtml
4. Очков В.Ф., Богомолова Е.П., Иванов Д.А. Физико-математические этюды с Mathcad и Интернет: учеб. пособие. 2-е изд. СПб.: Лань, 2018. 560 с. URL: http://twt.mpei.ac.ru/ochkov/T-2018/PhysMathStudies.pdf
5. Очков В.Ф., Иванова А.О., Алексеев М.Д. Три жадных алгоритма // Информатика в школе. 2018. № 9. С. 34–42. URL: http://twt.mpei.ac.ru/ochkov/3-Problem.pdf
6. http://chehov-lit.ru/chehov/text/tajna.htm
7. http://www.math.uwaterloo.ca/tsp
8. https://ilibrary.ru/text/967/p.1/index.html