**Графовые модели. Остов минимального веса**

Основные понятия теории графов. Инструментальные программные средства. Блок-схема алгоритма моделирования. Операционная среда моделирования. Контрольная задача моделирования

**Введение**

В настоящее время исследования в областях, традиционно относящихся к математике, занимают все более заметное место. Проблема выбора оптимального варианта решения относится к числу наиболее актуальных технико-экономических проблем.

Развитие теории графов в основном обязано большому числу всевозможных приложений. По-видимому, из всех математических объектов графы занимают одно из первых мест в качестве формальных моделей реальных систем.

Графы нашли применение практически во всех отраслях научных знаний: физике, биологии, химии, математике, истории, лингвистике, социальных науках, технике и т.п. Наибольшей популярностью теоретико-графовые модели используются при исследовании коммуникационных сетей, систем информатики, химических и генетических структур, электрических цепей и других систем сетевой структуры.

Цель курсового проекта заключается в закреплении практических умений и навыков в нахождении остова минимального веса с помощью алгоритма Краскала, в разработке программы на языке Delphi для аналитического и графического решений нашей поставленной задачи. Использование компьютерных технологий для решения данных задач сокращает усилия и время человека, а это не мало важно в настоящие время.

В курсовом проекте в разделе «Постановка задачи» рассматривается теоретический материал по теме «Графовые модели. Остов минимального веса», в разделе «Алгоритм нахождения» рассматриваются алгоритмы нахождения «Остова минимального веса», в разделе «Инструментальные программные средства» выбираются инструментальные средства для разработки программного продукта, в разделе «Операционная среда моделирования» определятся интерфейс программного продукта, в разделе «Контрольная задача моделирования» формулируется задача для ее решения вручную и с помощью программного продукта.

1 Постановка задачи моделирования

1.1Основные понятия теории графов

Графом называется алгоритмическая модель, состоящая из множества вершин (узлов) v и соединяющих их ребер e. Ребро - неупорядоченная пара вершин графа.

Ребра называются смежными, если они имеют общую вершину. Вершины называются смежными, если есть ребро их соединяющее. Ребро, которое соединяет вершины, называется инцидентным этим вершинам, а вершины – инцидентные этому ребру.

Граф G’(v’,e’) является остовом минимального веса графа G(v,e), если v’=v и e’ – есть подмножество ребер минимального веса и количества, соединяющих все вершины. Остовом минимального веса может являться сеть минимальной стоимости, в матрице соединения которой cij=cji.

Граф G=<v, e> называется ориентированным (орграфом), если найдется дуга (a,b)∈R такая, что (b,a)∉R.

Если же отношение R симметрично, то есть из (a,b)∈R следует (b,a)∈R, то граф G называется неориентированным (неорграфом).

Связный граф - граф, у которого для любых двух различных вершин существует цепь (последовательность смежных вершин), соединяющая их.

Для решения данной задачи моделирования будет использован неориентированный связный граф.

1.2 Представление графов

Существует четыре базовых представлений графов в памяти машины: матрица инцидентности, матрица смежности, матрица списков смежности и связанные списки смежности. Они различаются скоростью выполнения операций над элементами представления и объемом занимаемой памяти.

Для решения поставленной задачи моделирования больше всего подходит представление графов в памяти машины в виде матрицы смежности, а именно матрица весов.

Матрица смежности - матрица размером , элементы которой равны 1, если i-я вершина смежна с j-ой, и 0 в противном случае.



Матрица смежности является симметричной и достаточно просто может использоваться для ввода и обработки на ЭВМ. Для случая взвешенного графа вместо 1 используется значение весовой функции, и такая матрица называется матрицей весов. В поставленной задаче будем использовать матрицу весов.

1.3 Алгоритм нахождения остова минимального

веса во взвешенном графе

Для нахождения остова минимального веса с помощью метода Краскала нужно выполнить следующие шаги:

1.Помечают вершину начала построения остова. Среди ребер, инцидентных помеченной вершине, находят ребро минимального веса для остова. Помечают новую вершину, инцидентную этому ребру. Вершина новая, если к ней ранее не обращались.

2.Смотрят, все ли вершины помечены. Если да, то заканчивают поиск, если нет, то переходят к шагу 3.

3.Среди ребер, инцидентных помеченным вершинам, за исключением ребер остова и ребер, образующих в остове цикл, находят ребро минимального веса для остова. Помечают новую вершину, инцидентную этому ребру, и переходят к шагу 2.

4.При изменении вершины начала конфигурация остова минимального веса не измениться. Она может измениться при наличии в графе ребер одинакового минимального веса.

2 Инструментальные программные средства

2.1 Обоснование выбора инструментальных средств

При выборе программных средств для разработки программы «Поиск остова минимального веса во взвешенном графе» необходимо учитывать возможности графического отображения графа и остова в программе, определение модулей в программе и связи между ними, оценки развитости аппарата структур и типов данных, наличие специальных функций осуществления вычислений, возможность сохранения промежуточных результатов. Кроме того, программа должна позволять пользователю возвращаться к предыдущим этапам вычислений и сохранять выходные данные на жестком диске.

Учет этих возможностей позволит реализовать основные функции разрабатываемой программы, сделать ее легкодоступной для использования, предупредить возникновение логических ошибок, обеспечить надежность программного обеспечения и его модифицируемость.

Выбор того или иного программного средства определяется как спецификой разработки программного обеспечения и его популярностью, так и финансовыми возможностями разработчика.

В настоящее время наиболее распространенной средой является Delphi.

Delphi – пакет средств быстрой разработки приложений. К достоинствам относятся удобный интерфейс, высокая скорость работы, большое количество библиотек компонентов, эффективность создаваемых программ. Кроме того, строгая типизированность языка Object Pascal позволяет компилятору уже на этапе компиляции обнаружить многие ошибки, а также возможность работать с указателями.

По сравнению с другими системами визуального проектирования среда Delphi проста и эффективна, а написанные с помощью нее программы имеют небольшие размеры и высокую производительность. Так же в Delphi существует большая библиотека компонентов (командные кнопки, поля редактирования, переключатели и т.д.). С помощью компонентов обеспечиваются удобство интерфейса, наглядность работы программ, работа по созданию интерфейса сокращается до расстановки компонентов на форме.

Кроме того, в Delphi имеются развитые средства для работы с графическими возможностями Windows. В стандартном графическом интерфейсе Windows основой для рисования служит дескриптор контекста устройства нос и связанные с ним шрифт, перо и кисть. В состав входят объектно-ориентированные надстройки над последними, назначением которых является удобный доступ к свойствам инструментов и прозрачная для пользователя обработка всех их изменений. Поэтому использование класса TCanvas, являющегося основой графической системы Delphi, позволяет выполнить одну из основных функций разрабатываемой программы – наглядное представление графа. Delphi также дает возможность использовать традиционный набор функций работы с файлами, унаследованный от Turbo Pascal. Что позволяет сохранять результаты работы программы в файлы на жестком диске. Кроме того, в данной среде имеется возможность, наряду с обычными массивами, создавать динамические массивы, которые будут играть роль матрицы весов ребер графа. Хотя по большей части на представление графа в памяти машины выбор инструментальных средств особого значения не имеет.

Программа CorelDRAW 11, составляющая основу современного набора программных средств фирмы Corel, была выпущена в августе 2002 г. Она представляет собой результат двенадцатилетней эволюции, обладает удивительной универсальностью и мощностью, будучи в равной степени полезной и в промышленном дизайне, и в разработке рекламной продукции, и в подготовке публикаций, и в создании изображений для web-страниц, также в создании блок-схем алгоритмов. Несмотря на то, что мировым лидером программ для работы с векторной графикой сегодня является другая программа — Adobe Illustrator, CorelDRAW 11 ни в чем не уступает ей, а по многим параметрам и превосходит, и у нее — огромная армия пользователей-профессионалов, считающих CorelDRAW своим основным рабочим инструментом.

Пользовательский интерфейс CorelDRAW 11 построен очень рационально, с высокой степенью унификации и последовательным проведением простой идеи: если пользователю не нужны те или иные средства и возможности программы, он может не затрачивать время и усилия на их изучение. Это делает программу весьма привлекательной в качестве первого программного средства для приступающих к изучению машинной графики в целом или векторной графики в частности.

Таким образом, данная среда разработки программных продуктов позволяет выполнить основные функции данной задачи.

3 Блок-схема алгоритма задачи моделирования

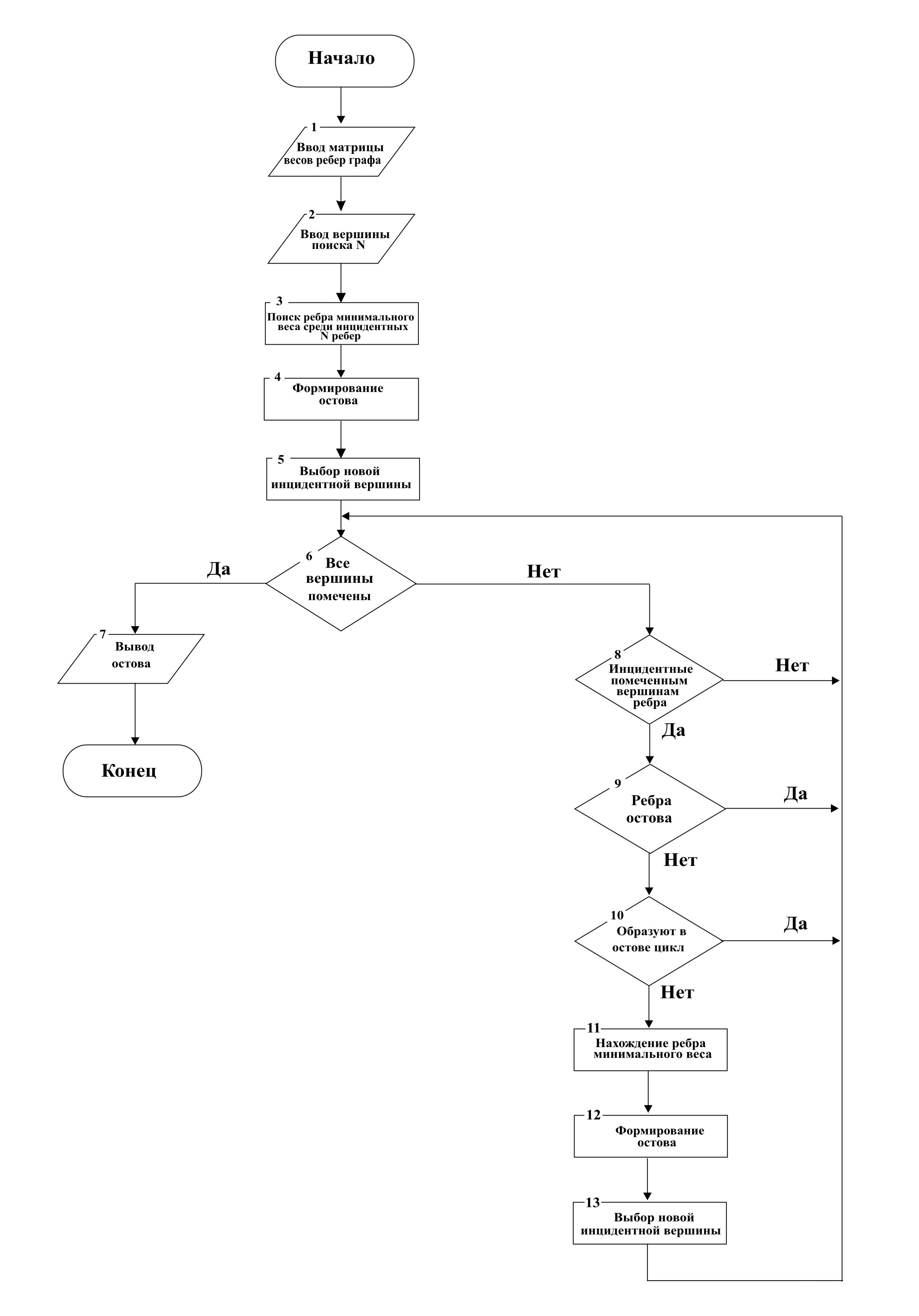


Рисунок 1.Блок-схема алгоритма задачи моделирования

3.1 Описание блок-схемы алгоритма задачи моделирования

Блок 1. Ввод матрицы весов ребер графа. Запись графа в память компьютера осуществляется при помощи двумерного массива, который служит матрицей весов ребер графа.

Блок 2. Ввод вершины поиска. После заполнения матрицы весов пользователем программа автоматически определяет вершину начала построения остова.

Блок 3. Поиск ребра минимального веса среди инцидентных n ребер. Программа анализирует матрицу весов и находит ребро с минимальным весом. Найденное ребро сохраняется в переменную min.

Блок 4. Формирование остова. Формируется остов.

Блок 5.Выбор новой инцидентной вершины. Помечается новая вершина, инцидентная ребру, - переменная m.

Блок 6. Все вершины графа помечены. Если все вершины графа помечены, то поиск остова заканчивается. Если нет, то среди инцидентных помеченным вершинам ребер, за исключением ребер остова и ребер, образующих в остов цикл, происходит поиск ребра минимального веса min и построение остова.

Блок 7. Вывод остова. После того как все вершины графа помечены, на монитор пользователя выводится остов минимального веса.

Блок 8. Инцидентные помеченным вершинам ребра. Если есть такие ребра, то программа анализирует найденные ребра, если нет инцидентных ребер, то программа переходит к Блоку 6.

Блок 9. Ребра остова. Найденное ребро не используется в остове, то программа переходит к Блоку 10, а если используется, то переходит к Блоку 6.

Блок 10. Образует ребро в остове цикл, если да то программа переходит к Блоку 6. Если ребро не образует в остове цикл, то программа переходит к Блоку11.

Блок 11. Нахождение ребра минимального веса. Программа анализирует оставшиеся инцидентные ребра выбранной вершине и переходит к Блоку 12.

Блок 12. Формирование остова. Программа формирует полученный остов, проверяется связанность ребер с вершинами графа, за это отвечает массив связанности ar[jmin, imin], если он равен единицам, то все ребра связаны с вершинами, если он не равен единице, то продолжается формирование остова.

Блок 13. Выбор новой инцидентной вершины. Помечается новая вершина графа, программа переходит к Блоку 6.

Блоки блок-схемы во многом повторяют шаги теоретического решения, лишь незначительно конкретизируясь на привязке к конкретному языку программирования (в данном случае Delphi).

В отличие от блок-схемы задачи моделирования здесь невозможно описать многие производимые операции (например, представление графа в виде графического образа, прорисовка остова и др.) в виде связанной структуры шагов решения задачи. Поскольку эти операции описываются множеством процедур и функций, присущим данной среде программирования.

4 Операционная среда моделирования

4.1 Описание операционной среды моделирования

Операционная система компьютера представляет собой комплекс взаимосвязанных программ, который действует как интерфейс между приложениями и пользователями с одной стороны, и аппаратурой компьютера с другой стороны. Операционная система также является механизмом, распределяющим ресурсы компьютера.

Программа, решающая данную задачу моделирования, должна обеспечивать удобный графический интерфейс для лучшего понимания модели. Широкий круг возможностей графического вывода и представления информации предоставляет разработанная фирмой Microsoft операционная система Windows.

Простота Windows достигнута за счет применения графического интерфейса пользователя, обеспечивающего удобную работу.

Широчайшее распространение Windows сделало ее фактическим стандартом для IBM PC - совместимых компьютеров. Подавляющее большинство пользователей таких компьютеров работают в Windows, поэтому в наше время большинство новых программ разрабатывается именно для эксплуатации их в среде Windows.

Windows не только обеспечивает удобный и наглядный интерфейс для операций с файлами, дисками и т.д., но и предоставляет новые возможности для запускаемых в среде Windows программ. Разумеется, для использования этих возможностей программы должны быть спроектированы по требованиям Windows.

Windows имеет разные версии, смотря, для кого предназначена операционная система для сервера или для клиента. Для разработки курсового проекта я выбрал операционную систему под названием Windows XP Professional, так как я считаю её наиболее подходящей. Данная версия Windows XP Professional наиболее распространена среди всех версий Windows. Для этой версии Windows написано большое количество программ, а это означает, что этой версией Windows пользуется большое количество пользователей, а если пользуются значит она работает корректно.

4.2 Аппаратная среда моделирования

Основные аппаратные затраты приходятся на среду проектирования данной программной модели (в данном случае Delphi). Минимальные требования, предъявляемые к оборудованию, при работе в данной среде программирования следующие:

-Процессор Intel Pentium с тактовой частотой 166 МГц и выше;

-128 МБ оперативной памяти;

-свободное пространство на жестком диске для полной установки 5 МБ;

-дисковод для компакт-дисков;

-VGA или SVGA монитор;

-стандартный манипулятор мышь и клавиатура;

-операционная система Windows 98/2000/XP.

Программная модель требует гораздо меньше аппаратных средств. Для ее работы достаточно стандартного набора оборудования: монитор типа VGA/SVGA, клавиатура, мышь. Программа занимает 568 КБ свободного пространства на диске и 12 МБ оперативной памяти. Программа может больше занимать пространства на жестком диске это связанно с тем, что матрица весов занесенная пользователем перед поиском минимального веса записывается в файл, и соответственно чем больше матриц весов будет занесено тем больше будет вес файла. После закрытия программы файл, в который записывались матрицы весов, он удаляется и пространство на жестком диске освобождается – это сделано для того чтобы не «засорять» свободное место на жестком диске. Особых требований к видеоадаптеру программа не имеет, но желательно 16 МБ и выше.

4.3 Руководство оператора

В данном подразделе представлен, алгоритм и правило работы с программой; функции программы.

Для запуска программы необходимо активировать exe – файл с названием «Краскал.exe» запустится программа. Рисунок главной формы изображен на рисунке1.

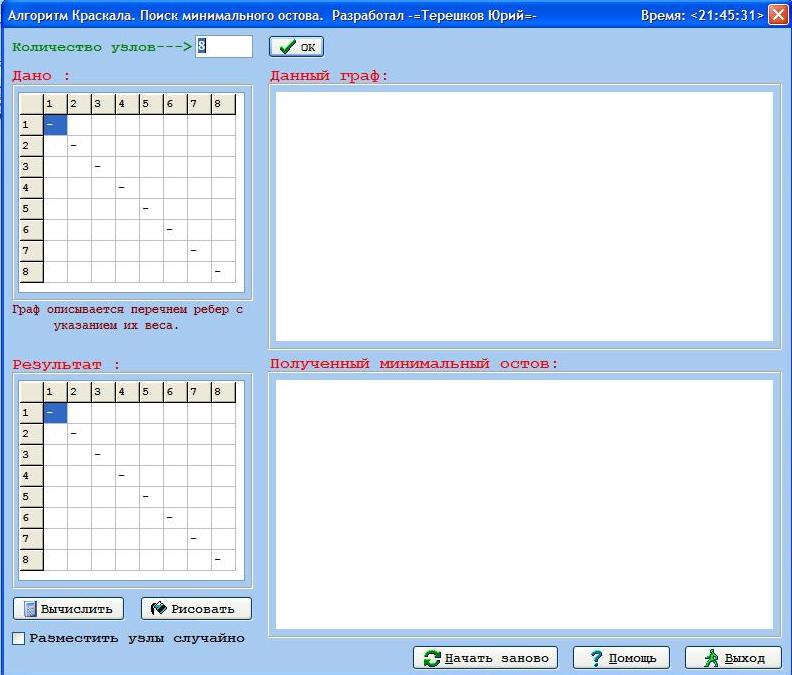


Рисунок 2.Главная форма программы.

На главной форме программы изображены: текстовое поле необходимое для ввода количество узлов графа, для которого нужно будет найти остов минимального веса, затем нужно нажать кнопку «ОК». Далее нужно занести веса в матрицу весов «Дано» вводить нужно только по горизонтали, а по вертикали программа заполнит поля автоматически. Далее нужно расставить узлы нашего графа, для этого одним щелчком по полю «Данный граф» создастся узел, он будет помечен синей точкой аналогично выполнить для остальных вершин графа. Также узлы можно расставить случайным образом, для этого нужно пометить флажок «Разместить узлы случайно» и нажать кнопку «Рисовать» при каждом нажатии на кнопку вершины будут размещаться случайно. Пример графа изображен на рисунке 2.

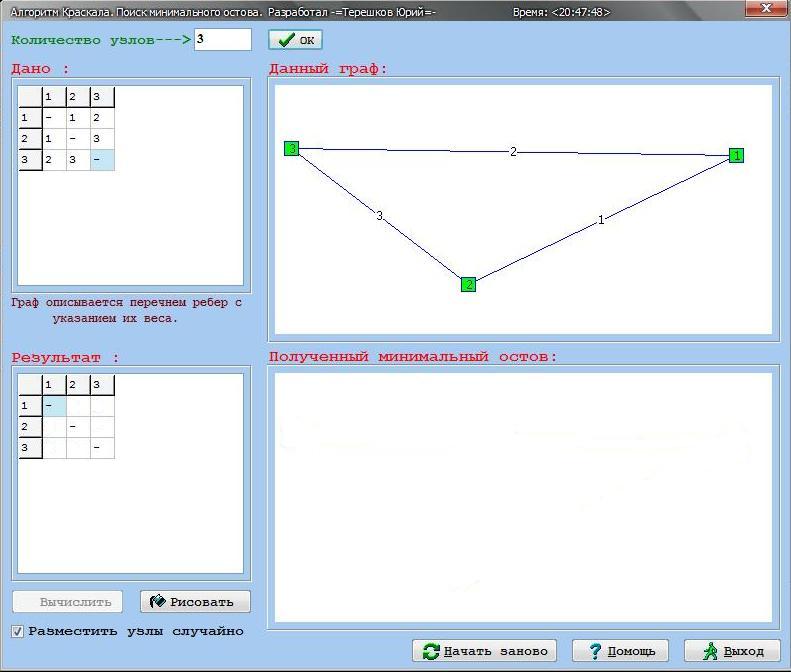


Рисунок 3.Графическое изображение графа.

После того, когда граф на рисован необходимо найти «Остов минимального веса» с помощью алгоритма Краскала, для этого нажимать кнопку «Вычислить». Остов минимального веса будет изображен в поле «Полученный минимальный остов» и в поле «Результат» будет показан результат виде матрицы весов. Результат решения на рисунке 3.

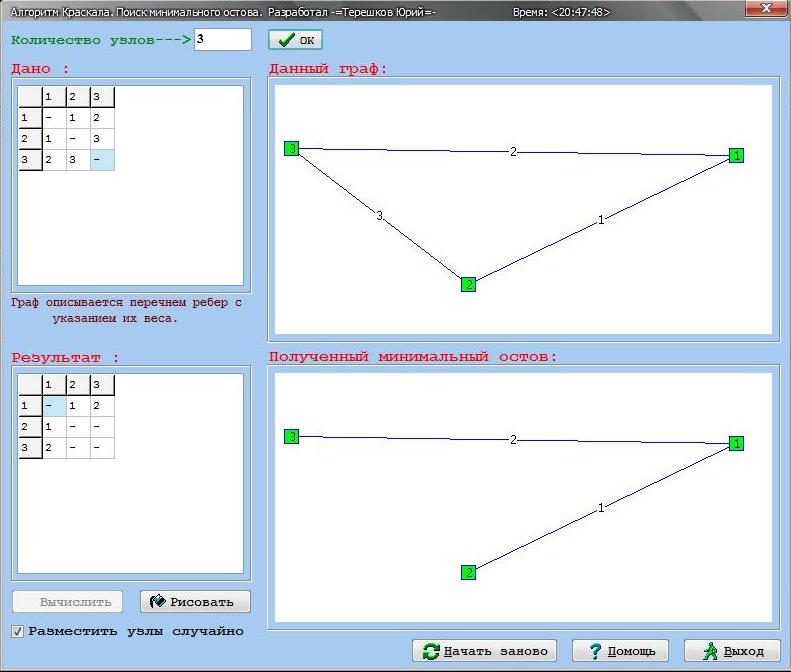


Рисунок 4.Найденный остов минимального веса.

На форме размещены еще три кнопки:

-«Начать заново» при нажатии на эту кнопку все поля очищаются и главная форма принимает первоначальный вид.

-«Помощь» при нажатии на эту кнопку вызывает помощь для пользователя. Помощь для пользователя изображена на рисунке 4.

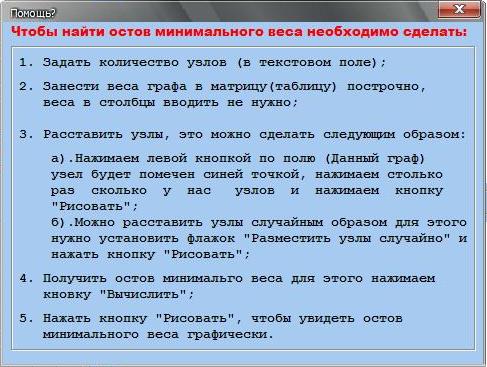


Рисунок 5. Помощь для пользователя.

Последняя кнопка, которая размещена на форме «Выход», при нажатии на кнопку приложение будет закрыто.

4.4 Лицензионное соглашение

Алгоритм Краскала (версия 1.0)

1) Всеми авторскими правами на "Алгоритм Краскала" эксклюзивно владеет автор программы – Терешков Юрий Игоревич.

2) " Алгоритм Краскала " могут распространяться только в том виде, в котором они поставляются автором.

3) " Алгоритм Краскала " распространяются по принципу "как есть". При этом не предусматривается никаких гарантий, явных или подразумеваемых. Вы используете его на свой собственный риск. Автор не отвечает за потери данных, повреждения, потери прибыли или любые другие виды потерь, связанные с использованием (правильным или неправильным) этой программы.

4) Вы не можете эмулировать, клонировать, сдавать в аренду, давать напрокат, продавать, изменять, декомпилировать, дизассемблировать " Алгоритм Краскала". Любое подобное неавторизованное использование приводит к немедленному и автоматическому прекращению действия этой лицензии и может повлечь за собой уголовное и/или гражданское преследование.

5) Все права, явно не представленные здесь, принадлежат автору программы.

6) Запуск и использование " Алгоритм Краскала " свидетельствует о согласии с условиями данной лицензии.

7) Если вы не согласны с условиями данной лицензии, то должны удалить файлы " Алгоритм Краскала " со своих устройств хранения информации и отказаться от их использования.

Спасибо за использование " Алгоритм Краскала "!

Автор программы: Терешков Юрий Игоревич.

5 Контрольная задача моделирования

В данном разделе решено две контрольные задачи:

-вручную;

-с помощью программной модели.

После решения контрольных задач проведено сравнение полученных минимальных остовов.

Задача №1. Дан взвешенный связный неориентированный граф, состоящий из пяти вершин. Необходимо найти остов минимального веса с помощью алгоритма Краскала.

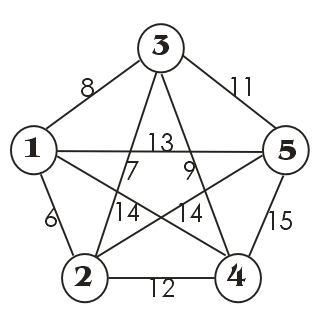


Рисунок 6. Исходный граф.

Выбираем вершину начала построения остова минимального веса, например, первую вершину.

Шаг 1. Найдено ребро минимального веса: 1-2=6. Полученный остов на рисунок 7.

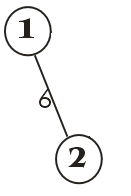


Рисунок 7. Полученный оостов на шаге 1

Шаг 2. Найдено ребро минимального веса: 2-3=7. Полученный остов на рисунок 8.

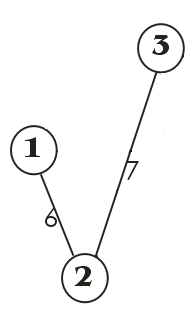


Рисунок 8.Полученный остов на шаге 2

Шаг 3. Найдено ребро минимального веса: 3-4=9. Полученный остов на рисунок 9.

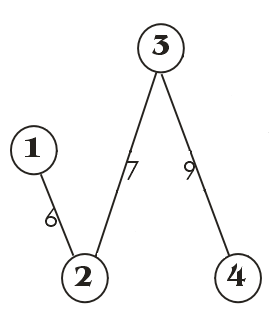


Рисунок 9.Полученный остов на шаге 3

Шаг 4. Найдено ребро минимального веса: 3-5=11.

Рассмотрены все вершины и инцидентные ребра этим вершинам, оставшиеся образуют цикл в полученном минимальном остове. А это не удовлетворяет условиям поставленной задачи.

На четвертом шаге получили окончательный остов минимального веса, который представлен на рисунке 10.

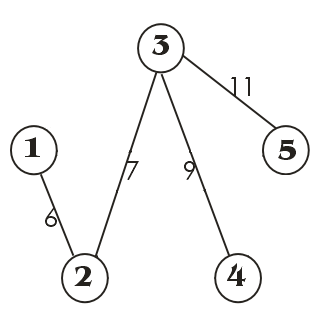


Рисунок 10. Остов минимального веса

При изменении вершины начала построения конфигурация остова минимального веса не измениться, а измениться лишь последовательность построения ребер остова.

Например, если в качестве начальной вершины выбрать четвертую вершину, то последовательность этапов построения остова минимального веса будет выглядеть следующим образом:

Шаг 1. 4-3=9;

Шаг 2. 3-2=7;

Шаг 3. 2-1=6;

Шаг 4. 3-5=11;

При этом конфигурация остова останется прежней. Решим данную задачу с помощью программной модели. Чтобы решить данную задачу необходимо построить матрицу весов, матрица представлена на рисунке 11.



Рисунок 11. Матрица весов

Полученный минимальный остов с помощью программной модели изображен на рисунке 12.

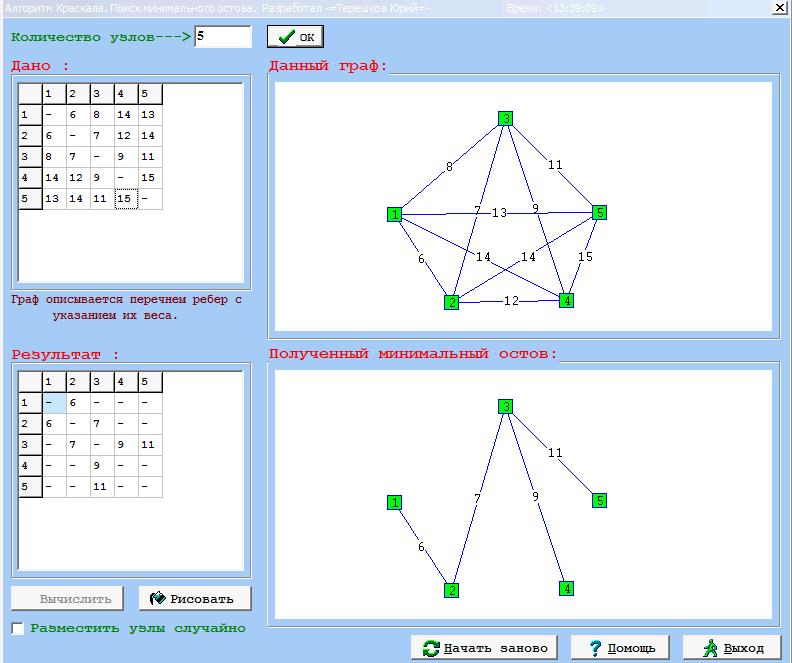


Рисунок 12. Полученный минимальный остов

После проверки вычислений вручную и программной модели результат одинаковый, это означает, что программная модель работает и функционирует верно.

Задача №2. Дан взвешенный, связный, неориентированный граф, состоящий из девяти вершин. Необходимо найти остов минимального веса с помощью алгоритма Краскала. Исходный граф на рисунке 13.

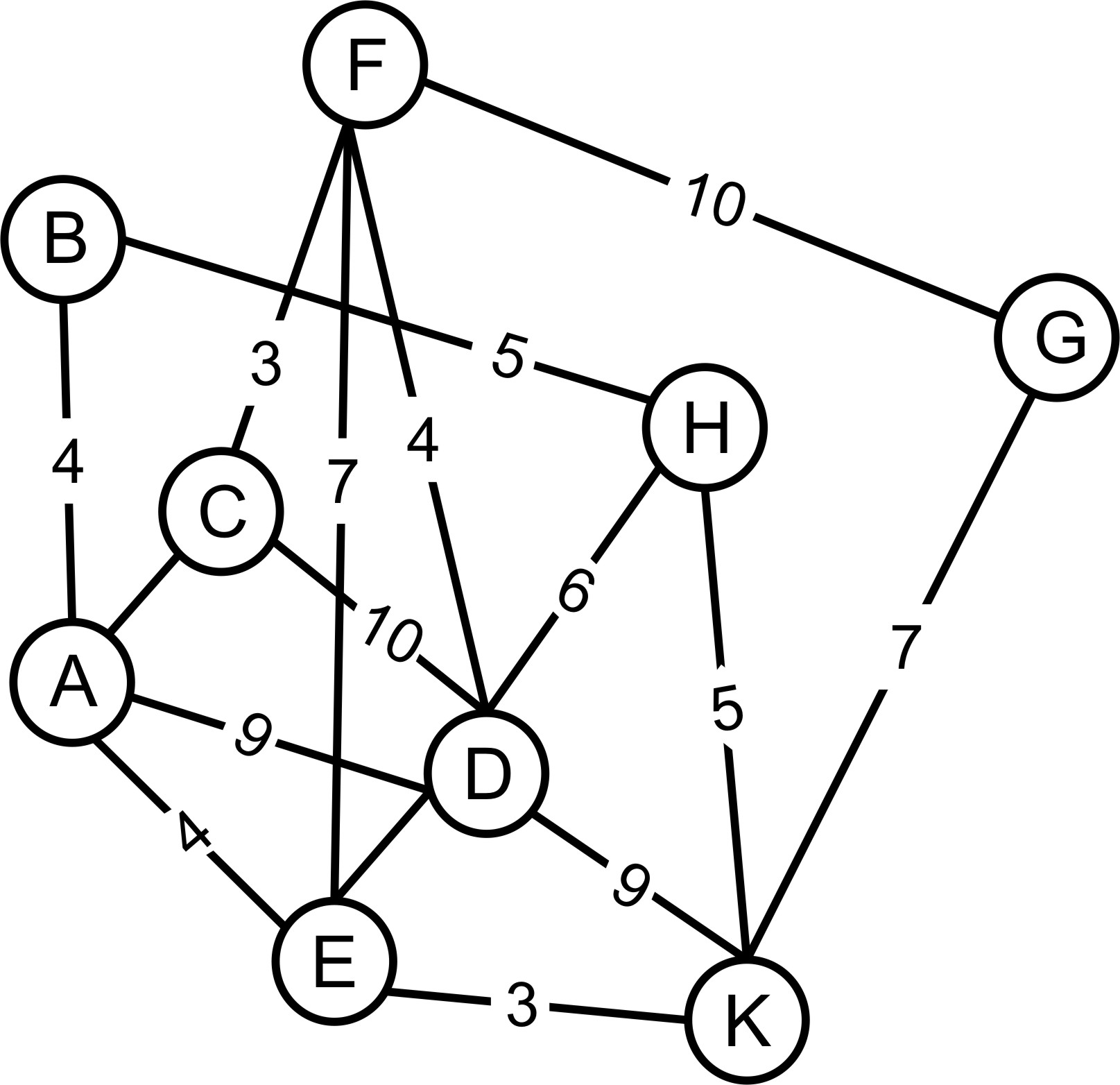


Рисунок 13. Исходный граф

Выбираем вершину начала построения остова минимального веса, например, первую вершину.

Шаг 1. Найдено ребро минимального веса: AC=1. Полученный остов на рисунок 14.

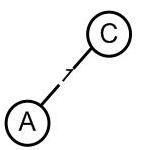


Рисунок 13. Полученный остов на шаге 1

Шаг 2. Найдено ребро минимального веса: CF=3, AB=4, AC=4. Полученный остов на рисунок 15.

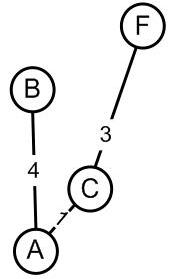


Рисунок 14. Полученный остов на шаге 2

Шаг 3. Найдено ребро минимального веса: FD=4, EK=3, AE=4. Полученный остов на рисунок 15.

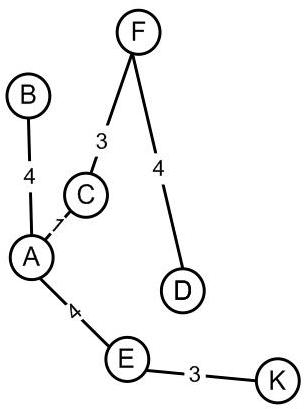


Рисунок 15. Полученный остов на шаге 3

Шаг 4. Найдено ребро минимального веса: KH=5, KG=4. Рассмотрены все вершины и инцидентные ребра этим вершинам, оставшиеся ребра образуют цикл в полученном минимальном остове. А это не удовлетворяет условиям поставленной задачи.

На четвертом шаге получен окончательный остов минимального веса, который представлен на рисунке 16.

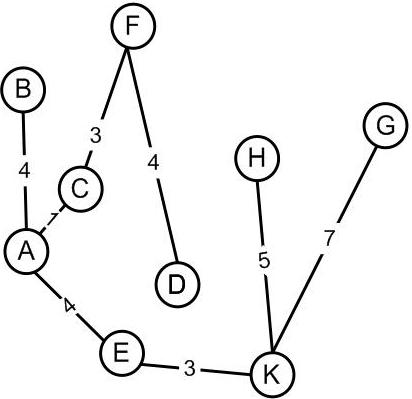


Рисунок 16. Остов минимального веса

Решим данную задачу с помощью программной модели. Чтобы решить данную задачу необходимо построить матрицу весов.

Таблица 1. Матрица весов

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H | K |
| A | - | 4 | 1 | 9 | 4 | - | - | - | - |
| B | 4 | - | - | - | - | - | - | 5 | - |
| C | 1 | - | - | 10 | - | 3 | - | - | - |
| D | 9 | - | 10 | - | 5 | 4 | - | 6 | 9 |
| E | 4 | - | 3 | 5 | - | 7 | - | - | 3 |
| F | - | - | - | 4 | 7 | - | 10 | - | - |
| G | - | - | - | - | - | 10 | - | - | 7 |
| H | - | 5 | - | 6 | - | - | - | - | 5 |
| K | - | - | - | 9 | 3 | - | 7 | 5 | - |

Полученный минимальный остов с помощью программной модели изображен на рисунке 17.

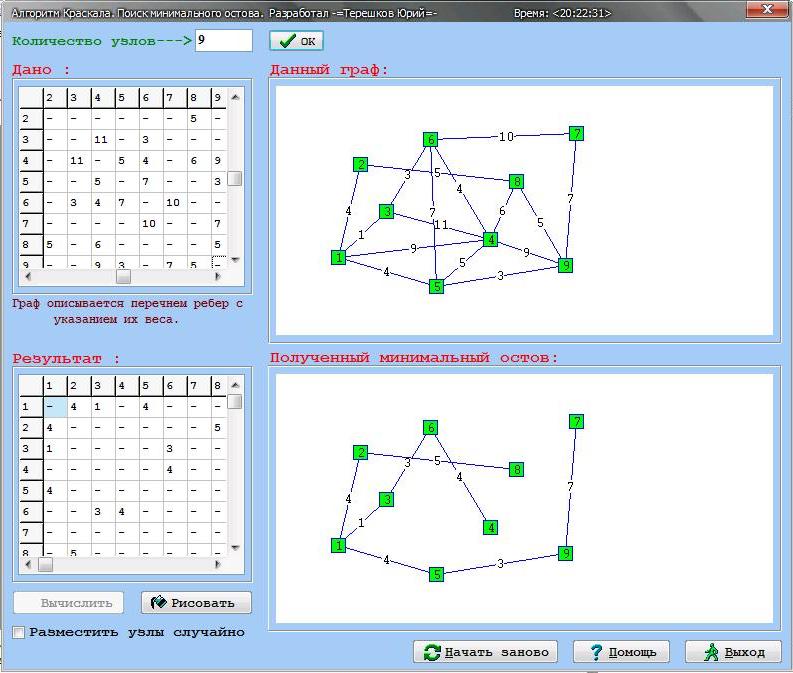


Рисунок 17. Остов минимального веса

После проверки вычислений вручную и программной модели полученные остовы минимального веса различаются, но они построены верно. Это связано с тем, что программа выбирает другую вершину начала. После решения двух контрольных задач стало ясно, что разработанная программная модель работает верно.

Заключение

Целью данного курсового проекта была задача нахождения остова минимального веса во взвешенном графе с помощью алгоритма Краскала. Есть много способов создания модели, решающей эту задачу. Могут существовать различные алгоритмы обработки графов с разными представлениями: в виде матрицы инцидентности, матрицы смежности, матрицы весов. При решении данной задачи можно изменять вершину начала поиска остова минимального веса, при этом конфигурация остова не измениться. Она может измениться при наличии ребер одинакового минимального веса.

Контрольная задача показала, что данная программная модель функционирует верно, и поэтому она может быть успешно использована в качестве наглядного пособия для изучения задачи нахождения остова минимального веса. Для эффективности изучения в программе создана подсказка для пользователя, позволяющая быстро изучить назначение компонентов. Для наглядности представления метода в программе имеется графическое изображение графа.

**Список литературы**

Судоплатов С.В., Овчинникова Е. В. Элементы дискретной математики: Учебник. – М.: ИНФРА-М, Новосибирск: Изд-во НГТУ,2002. – 208 с.

Кандзюба С.П., Громов В.Н. Delphi 7. Базы данных и приложения. Лекции и упражнения. – СПб: ООО «ДиаСофтЮП», 2005. – 576 с.

Богумирский Б. А. Энциклопедия Windows 98. 2-е изд. – СПб.: Питер, 2003–896 с.

Липский С.Г. «Комбинаторика для программистов»

Васильков Ю.В., Н.Н. Василькова «Компьютерные технологии вычислений в математическом моделировании», М. Финансы и Статистика, 1999

Культин Н.Б. Delphi 7 Программирование на Object Pascal. – СПб.: БХВ – Петербург, 2005. – 528 с.

Приложение А: листинг программы

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Buttons, Grids, StdCtrls, ExtCtrls, ComCtrls, XPMan, Menus;

type

TForm1 = class(TForm)

sg: TStringGrid;

SpeedButton3: TSpeedButton;

sr: TStringGrid;

SpeedButton4: TSpeedButton;

SpeedButton5: TSpeedButton;

Edit1: TEdit;

Label1: TLabel;

SpeedButton7: TSpeedButton;

Image1: TImage;

Image2: TImage;

Label2: TLabel;

Label3: TLabel;

Timer1: TTimer;

SpeedButton8: TSpeedButton;

CheckBox1: TCheckBox;

Bevel1: TBevel;

Bevel2: TBevel;

Bevel3: TBevel;

Bevel4: TBevel;

Bevel5: TBevel;

Bevel6: TBevel;

Bevel7: TBevel;

Bevel8: TBevel;

Bevel9: TBevel;

Bevel10: TBevel;

BitBtn1: TBitBtn;

BitBtn2: TBitBtn;

Vremya: TTimer;

XPManifest1: TXPManifest;

Label4: TLabel;

BitBtn3: TBitBtn;

BitBtn4: TBitBtn;

Label5: TLabel;

Label6: TLabel;

procedure FormCreate(Sender: TObject);

procedure SpeedButton3Click(Sender: TObject);

procedure SpeedButton4Click(Sender: TObject);

procedure SpeedButton5Click(Sender: TObject);

procedure SpeedButton7Click(Sender: TObject);

procedure Image2DblClick(Sender: TObject);

procedure SpeedButton8Click(Sender: TObject);

procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure Image2Click(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure BitBtn2Click(Sender: TObject);

procedure VremyaTimer(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure BitBtn4Click(Sender: TObject);

procedure sgClick(Sender: TObject);

procedure srClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

f:file of integer;

idown,n,wrt,i,j:integer;

a,ar:array[1..10,1..10] of integer;

m:array[1..10] of integer;

vx:array[1..10] of integer;

vy:array[1..10] of integer;

implementation

uses Unit2;

{$R \*.dfm}

procedure TForm1.FormCreate(Sender: TObject);

var

t,i:integer;

begin

n:=8; {изначально число вершин=8}

SpeedButton3.Enabled:=True;

idown:=1;

speedbutton7.click;

image1.Canvas.brush.color:= clwhite;

image1.Canvas.pen.Color:=clwhite;

image2.Canvas.brush.color:= clwhite;

image2.Canvas.pen.Color:=clwhite;

image1.Canvas.Rectangle(0,0,image1.Width,image1.Height);

image2.Canvas.Rectangle(0,0,image1.Width,image1.Height);

for i:=1 to sr.ColCount do

for j:=1 to sr.Rowcount do begin

sr.Cells[i,j]:='';

end;

for i:=1 to sg.ColCount do

for j:=1 to sg.Rowcount do begin

sg.Cells[i,j]:='';

edit1.Text:= inttostr(t);

for t:=1 to n do begin

sg.Cells[t,t]:='-';

sr.Cells[t,t]:='-';

end; end;

edit1.Text:= inttostr(n);

end;

procedure TForm1.SpeedButton3Click(Sender: TObject);

var

o,min,imin,jmin:integer;

begin

SpeedButton3.Enabled:=false;

assignfile(f,extractfilepath(application.ExeName)+'\in.krs');

rewrite(f);

for i:= 1 to n do

for j:= 1 to n do

begin

if sg.cells[i,j]='-' then

wrt:=999

else

wrt:=strtoint(sg.cells[i,j]);

write(f,wrt);

end;

closefile(f);

assignfile(f,extractfilepath(application.exename)+'\in.krs');

reset(f);

for i:= 1 to n do

for j:= 1 to n do

begin

read(f,wrt);

if wrt=999 then

sg.cells[i,j]:='-'

else

sg.cells[i,j]:= inttostr(wrt);

a[i,j]:=wrt;

end;

closefile(f);

for i:=1 to n do

m[i]:=0;

m[1]:=1;

repeat

o:=0;

min:=100; imin:=1; jmin:=1;

for i:= 1 to n do

if m[i]=1 then

for j:= 1 to n do

if (a[i,j]<>0) and (a[i,j]<900) and (m[j]<>1) then

begin

if a[i,j]<min then

begin

min:= a[i,j];

imin:=i;

jmin:=j;

o:=1;

end; end;

if o=1 then

begin

ar[imin,jmin]:=min;

ar[jmin,imin]:=min;

m[jmin]:=1;

end;

until o=0;

speedbutton4.Click;

end;

procedure TForm1.SpeedButton4Click(Sender: TObject);

begin

for i:= 1 to n do

for j:= 1 to n do

begin

if ar[i,j]=0 then

sr.cells[i,j]:='-'

else

sr.cells[i,j]:=inttostr(ar[i,j]);

end;

end;

procedure TForm1.SpeedButton5Click(Sender: TObject);

var

i,x,y:integer;

begin

idown:=1;

form1.canvas.Refresh;

if checkbox1.Checked then speedbutton8.Click;

image1.Canvas.brush.color:= clwhite;

image1.Canvas.pen.Color:=clwhite;

image2.Canvas.brush.color:= clwhite;

image2.Canvas.pen.Color:=clwhite;

image1.Canvas.Rectangle(0,0,image1.Width,image1.Height);

image2.Canvas.Rectangle(0,0,image1.Width,image1.Height);

with image1.Canvas do

begin

brush.color:= cllime;

pen.Color:=clblue;

font.Name:='Courier';

font.Size:=8;

for i:= 1 to n do

for j:=1 to n do

if (a[i,j]<>0) and (a[i,j]<900) then

begin

pen.Width:=1;

moveto(vx[i]+7,vy[i]+7);

lineto(vx[j]+7,vy[j]+7);

brush.color:= clwhite;

textout(round((vx[i]+vx[j]+4)/2),round((vy[i]+vy[j]+1)/2),inttostr(a[i,j]));

end;

brush.color:= cllime;

for i:= 1 to n do

begin

font.Size:=1;

rectangle(vx[i],vy[i],vx[i]+15,vy[i]+15);

textout(vx[i]+4,vy[i]+1,inttostr(i));

end;

end;

with image2.Canvas do

begin

brush.color:= clLime;

pen.Color:=clblue;

font.Name:='Courier';

font.Size:=8;

for i:= 1 to n do

for j:=1 to n do

if (ar[i,j]<>0) and (ar[i,j]<900) then

begin

pen.Width:=1;

moveto(vx[i]+7,vy[i]+7);

lineto(vx[j]+7,vy[j]+7);

brush.color:= clwhite;

textout(round((vx[i]+vx[j]+4)/2),round((vy[i]+vy[j]+1)/2),inttostr(ar[i,j]));

end;

brush.color:= cllime;

for i:= 1 to n do

begin

font.Size:=1;

rectangle(vx[i],vy[i],vx[i]+15,vy[i]+15);

textout(vx[i]+4,vy[i]+1,inttostr(i));

end; end; end;

procedure TForm1.SpeedButton7Click(Sender: TObject);

var

i:integer;

begin

for i:= 1 to n do

begin

sg.ColCount:=n+1;

sg.Rowcount:=n+1;

sr.ColCount:=n+1;

sr.Rowcount:=n+1;

sg.Cells[0,i]:=inttostr(i);

sr.Cells[0,i]:=inttostr(i);

sg.Cells[i,0]:=inttostr(i);

sr.Cells[i,0]:=inttostr(i); end;

for i:= 1 to n do

for j:= 1 to n do

begin

ar[i,j]:=0;

end;

end;

procedure TForm1.Image2DblClick(Sender: TObject);

begin

timer1.Enabled:=true;

end;

procedure TForm1.SpeedButton8Click(Sender: TObject);

begin

for i:= 1 to n do

begin

vx[i]:=random(470);

vy[i]:=random(230);

end; end;

procedure TForm1.Image1MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

begin

vx[idown]:=x;

vy[idown]:=y;

idown:=idown+1;

image1.Canvas.brush.color:= cllime;

image1.Canvas.pen.Color:=clblue;

image1.Canvas.Rectangle(x-1,y-1,x+1,y+1);

end;

procedure TForm1.Image2Click(Sender: TObject);

begin

image1.Canvas.brush.color:= clwhite;

image1.Canvas.pen.Color:=clwhite;

image2.Canvas.brush.color:= clwhite;

image2.Canvas.pen.Color:=clwhite;

image1.Canvas.Rectangle(0,0,image1.Width,image1.Height);

image2.Canvas.Rectangle(0,0,image1.Width,image1.Height); timer1.Enabled:=false;

end;

procedure TForm1.BitBtn1Click(Sender: TObject);

begin

n:= strtoint(edit1.text);

speedbutton7.Click;

end;

procedure TForm1.BitBtn2Click(Sender: TObject);

var i:integer;

begin

if MessageDlg('Завершить работу',mtConfirmation,[mbyes,mbno],0) = mrYes then begin

AlphaBlend:=true;

i:=255;

while i>0 do begin

AlphaBlendValue:=i;

Application.ProcessMessages;

dec(i,1);

end;close;end;end;

procedure TForm1.FormShow(Sender: TObject);

begin

Edit1.SetFocus;

end;

procedure TForm1.BitBtn4Click(Sender: TObject);

begin

form2.show;

end;

procedure TForm1.sgClick(Sender: TObject);

var i,j:integer;

begin

for i:= 1 to n do

for j:= 1 to n do

begin

sg.Cells[i,j]:=sg.Cells[j,i];

end;

end;

procedure TForm1.srClick(Sender: TObject);

var i,j:integer;

begin

for i:= 1 to n do

for j:= 1 to n do

begin

sr.Cells[i,j]:=sr.Cells[j,i];

end; end; end.

Приложение Б: исходные файлы