

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РФ
ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Дорофеева В.И.

Алгоритмы на графах и их приложения

Учебно-методическое пособие
по дисциплине «Дискретная математика»
(раздел «Элементы теории графов»)

Орел, 2002

Печатается по решению редакционно-издательского совета Орловского государственного университета (Протокол № 4 от 26 декабря 2002 г.)
УДК 519.178

Автор: Дорощеева В.И., канд. физ.-мат. наук, и.о. доцента кафедры информатики ОГУ

Рецензенты: Мешков А.Г., докт. физ.мат. наук, профессор,
зав. каф. информатики ОГУ
Коротченкова А.А., канд пед. наук, доцент,
зав. каф. информатики Орловского коммерческого института

Ответственный за выпуск: Веркеенко Г.П., к.п.н., профессор, проректор по НИР ОГУ

Дорощеева В.И. Алгоритмы на графах и их приложения. Учебно-методическое пособие по дисциплине «Дискретная математика» (раздел «Элементы теории графов»). – Орел: ОГУ, 2002.- 48 с.

Данное пособие содержит основные сведения по теории графов, описание некоторых алгоритмов на графах и учебно-методические рекомендации по решению задач с помощью специально разработанных программ на языке Delphi. В пособии приводится ряд примеров для разных типов задач, возникающих в дискретной математике, прикладной комбинаторике и теории алгоритмов.

Пособие предназначено для студентов естественно-научных специальностей, изучающих дискретную математику.

Библиогр.: 11

Содержание

Введение	4
1 Некоторые сведения из теории графов, необходимые для решения задач оптимизации	5
2 Оптимизация потока в сети	7
2.1 Формулировка задачи нахождения максимального потока в транспортной сети в терминах теории графов	7
2.2 Алгоритм Форда-Фалкерсона нахождения максимального потока в сети	11
2.3 Приложение алгоритма Форда-Фалкерсона к решению задачи нахождения максимального потока	13
2.4. Численная и программная реализации решения задачи оптимизации потока в сети	17
2.4.1 Преимущества реализации в среде Delphi	17
2.4.2 Описание этапов работы программы	17
2.5 Дополнительные возможности редактирования	22
2.6 Применение программы для решения некоторых других задач	24
2.7 Результаты решения некоторых задач	27
3 Кратчайшие пути	32
3.1 Формулировка задачи нахождения кратчайшего расстояния в терминах теории графов	32
3.2 Алгоритм Дейкстры для нахождения минимального пути в графе	32
3.3 Описание структуры программы	35
3.4 Описание этапов работы программы	41
3.5 Дополнительные возможности редактирования.	43
3.6 Результаты решения задач	45
Литература	47

Введение

Методы решения прикладных задач с недавних пор выходят за рамки классической математики. Специалисту в области прикладной математики в значительной степени приходится оперировать понятиями математики дискретной. Различные направления дискретной математики дают возможность решать широкие классы практических задач. Среди таких задач существенный интерес вызывают задачи и алгоритмы их решения, позволяющие использовать теорию графов для получения искомого результата.

Проблема решения задач дискретной математики была весьма актуальна до недавнего времени, и некоторые типы задач оставались просто нерешенными. Прежде всего, это было связано с тем, что большинство таких задач требовали значительного числа операций перебора, предполагали умение работать с огромными объемами информации. Аналитически для реальных задач это было выполнить невозможно. И только с возникновением ЭВМ появилась возможность организовать перебор на компьютере со значительной скоростью. Однако, даже и при наличии ЭВМ реализация алгоритмов, использующих понятия из теории графов, представляет существенную сложность из-за трудоемкости самих алгоритмов, большого числа операций, ограничений машинной памяти и различных других условий на используемые ресурсы.

Наряду с этим, в последнее десятилетие, с развитием средств коммуникации с помощью компьютеров (локальных и глобальных компьютерных сетей), актуальность решения задач на графах невероятно возросла, поскольку, в частности, теоретической основой передачи информации в сетях являются понятия и определения теории графов.

Данное пособие содержит основные сведения по теории графов, описание некоторых алгоритмов на графах и учебно-методические рекомендации по решению задач с помощью программ на языке Delphi, разработанных в рамках написания дипломных работ по кафедре информатики ОГУ. В пособии приводятся ряд примеров для разных типов задач, возникающих в дискретной математике, прикладной комбинаторике и теории алгоритмов.

1 Некоторые сведения из теории графов, необходимые для решения задач оптимизации

Задачи дискретной математики, конкретные типы которых будут рассмотрены далее, часто приводят к проблеме оптимизации. Пусть дан критерий, позволяющий выделить решения с некоторыми заданными свойствами. Требуется найти множество таких решений, называемых «оптимальными». В большинстве задач этот критерий имеет числовую природу, а множество всех задач разбивается на вполне упорядоченные классы. Как правило, рассматривают задачи на максимум и минимум.

В основе почти всех комбинаторных методов оптимизации лежит следующий принцип. Множество всех решений разбивается некоторым образом на два подмножества: подмножество, содержащее все оптимальные решения, и подмножество, не содержащее их. Если первое из подмножеств не есть оптимальное, то стремятся получать подобные разбиения, в которых подмножества, содержащие все оптимальные решения, имеют все меньшую мощность. Действуют так до тех пор, пока не приходят к оптимальному подмножеству. На этом же принципе основаны также некоторые способы, позволяющие получать, по крайней мере, одно оптимальное решение.

Одними из наиболее удобных способов решения подобных задач стали способы, в основе которых лежит интерпретация задачи в терминах теории графов. Введем некоторые известные понятия из теории графов, которые будут использованы в дальнейшем.

Представим себе на плоскости некоторое конечное множество точек N и конечный набор A линий, соединяющих некоторые пары точек из N . Указанной геометрической конфигурацией описывается, например, схема автомобильных дорог, связывающих города некоторой области.

Итак, ориентированный граф $G=(N,A)$ (далее граф) состоит из:

1. Множества узлов (или вершин) $N=\{n_1, n_2, \dots, n_p\}$.
2. Множества A упорядоченных пар (n_i, n_j) элементов N , элементы A называются дугами.

Граф может быть графически представлен диаграммой, на которой узлам соответствуют точки плоскости, а каждая дуга (n_i, n_j) изображается стрелкой, направленной из точки n_i в точку n_j . Так на рисунке 1 граф представлен четырьмя узлами n_1 – n_4 и шестью дугами (n_1, n_2) , (n_1, n_4) , (n_2, n_4) , (n_3, n_2) , (n_4, n_1) и (n_4, n_3) .

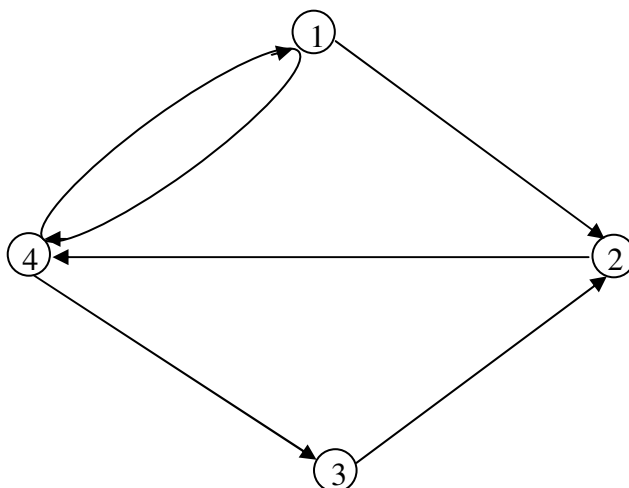


Рисунок 1

Многие физические структуры удобно представить в виде графа такого рода (например, систему городских улиц, если каждую улицу с двухсторонним движением рассматривать как пару дуг с одинаковыми конечными узлами, но с противоположной ориентацией).

Будем говорить, что на графе $G=(N, A)$ дуга (n_i, n_j) направлена от n_i к n_j , а точки n_i и n_j называть соответственно начальным и конечным узлом дуги (n_i, n_j) . Последовательность дуг, в которой конечный узел каждой дуги является в то же время начальным узлом следующей дуги, называется путем. Так на графе, представленном на рисунке 1, последовательность (n_1, n_2) , (n_2, n_4) , (n_4, n_3) – путь из n_1 в n_3 .

Для графов также существует такое понятие, как изоморфизм. Понятие изоморфизма для графов имеет наглядное толкование. Представим рёбра графов эластичными нитями, связывающими узлы – вершины. Тогда, изоморфизм можно представить как перемещение узлов и растяжение нитей.

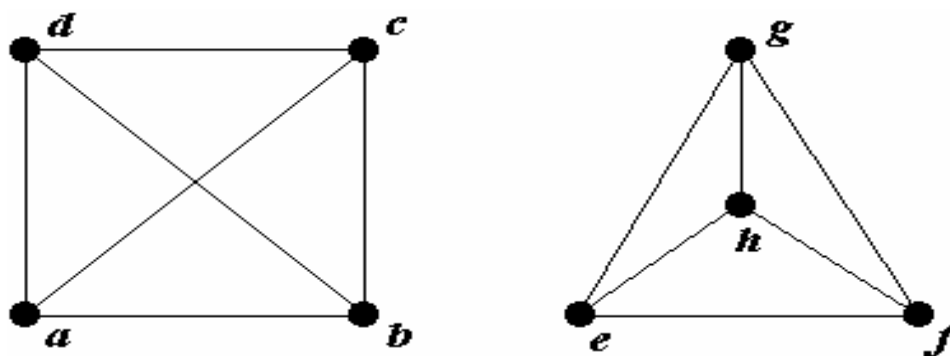


Рисунок 2

Два графа на рисунке 2 изоморфны.

Цепью называется последовательность дуг, в которой каждая промежуточная дуга соединена с предшествующим концом или началом. Например, на рисунке 1 последовательность $(n_1, n_4), (n_2, n_4), (n_3, n_2)$ – цепь из n_1 в n_3 . Двигаясь вдоль цепи, можно пройти дугу в направлении, противоположном ее ориентации. Дуги, проходимые в направлении ориентации, называются прямыми дугами цепи, остальные – обратными дугами.

Граф называется *вырожденным*, если он не имеет ребер. *Параллельными* ребрами графа называются такие, которые имеют общие узлы начала и конца. Неориентированный граф $G=(N, A)$ называется *связанным*, если для любых двух узлов n_i, n_j принадлежащих N существует последовательность ребер из набора A , соединяющий n_i и n_j .

Пусть задан граф $G=(N,A)$. Говорят, что на дугах графа реализуется *числовая* функция, если каждой дуге $(n_i, n_j) \in G$ ставится в соответствие число $\mu_{ij}=f(n_i, n_j)$ из некоторого множества M .

2 Оптимизация потока в сети

2.1 Формулировка задачи нахождения максимального потока в терминах теории графов

Изучение сетевых моделей началось еще в 40-х годах в связи с транспортными задачами, то есть задачами о прикреплении поставщиков к потреби-

телям, минимизирующими суммарные расходы по перевозке. Однако скоро выяснилось, что методы, развитые при анализе такого рода задач, приложимы и к другим сетевым проблемам, например к задачам об информационных потоках в сетях связи или к задачам о дорожных транспортных потоках. Более того, обнаружилось, что целый ряд прикладных комбинаторных задач, не связанных с реально существующими сетями, допускает, тем не менее, изящное математическое решение на языке сетевых моделей.

На практике часто возникает задача максимизации потока некоего продукта между двумя заданными узлами сети при условии, что поток вдоль каждой дуги ограничен. Очевидный пример – поток городского транспорта.

Приведем основные определения из теории потоков в сетях.

Транспортной сетью называется ориентированный граф $G=(E,U)$ с множеством вершин E и множеством дуг U , для которого выполняются условия:

- 1) существует одна и только одна вершина X_0 , в которую не заходит ни одна дуга из множества U ;
- 2) существует одна и только одна вершина X_N , из которой не исходит ни одной дуги;
- 3) каждой дуге $u \in U$ поставлено в соответствие целое число $c(u) \geq 0$, называемое пропускной способностью дуги.

X_0 называют входом (*источником*) сети, X_N – выходом (*стоком*), а $c(u)$ – пропускной способностью дуги u . На рисунке 3 изображен граф, являющийся транспортной сетью

Поток в транспортной сети. Функцию $\varphi(u)$, определенную на множестве U , называют *потоком* в транспортной сети, если:

1. $(\forall u \in U) \varphi(u) \geq 0$,
2. $(\forall X_i \neq 0 \text{ и } \neq X_N) \sum_{u \in U^-_{x_i}} \varphi(u) = \sum_{u \in U^+_{x_i}} \varphi(u)$

где $U_{x_i}^-$ – множество дуг, заходящих в X_i ;

$U_{x_i}^+$ – множество дуг, исходящих из X_i ,

3. $(\forall u \in U) \varphi(u) \leq 0$,

Поток уподобляется количеству вещества, протекающему по дуге.

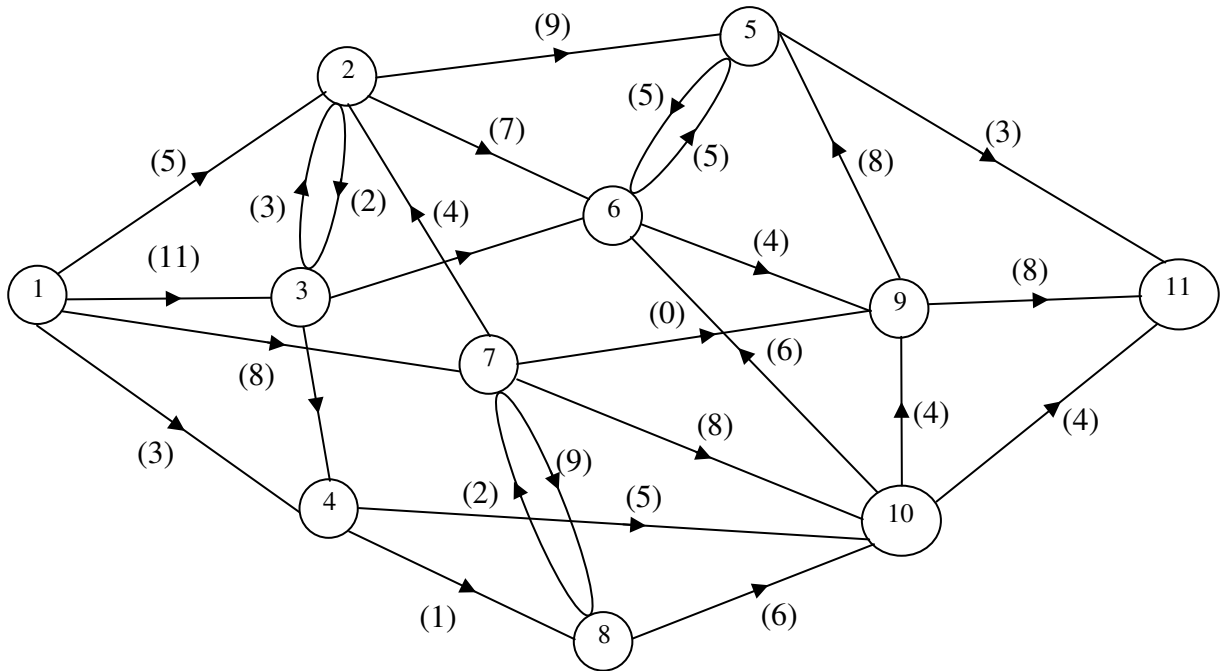


Рисунок 3

В силу условия 2 из определения потока имеем

$$\left(\varphi_{x_0} = \sum_{u \in U_{x_i}^+} \varphi(u) \text{ и } \varphi_{x_N} = \sum_{u \in U_{x_i}^-} \varphi(u) \right) \Rightarrow (\varphi_{x_0} = \varphi_{x_N}).$$

Разрез. Для подмножества $A \subset E$ такого, что $X_0 \notin A$, а $X_N \in A$, разрезом сети $G=(E,U)$ называют множество U_A^- дуг, заходящих в A .

Пропускная способность разреза. Число

$$c(U_A^-) = \sum_{u \in U_A^-} c(u)$$

называют пропускной способностью разреза U_A^- .

Например для сети на рисунке 4 $c(U_A^-) = 8 + 2 + 5 + 6 + 9 + 5 + 8 + 8 = 51$.

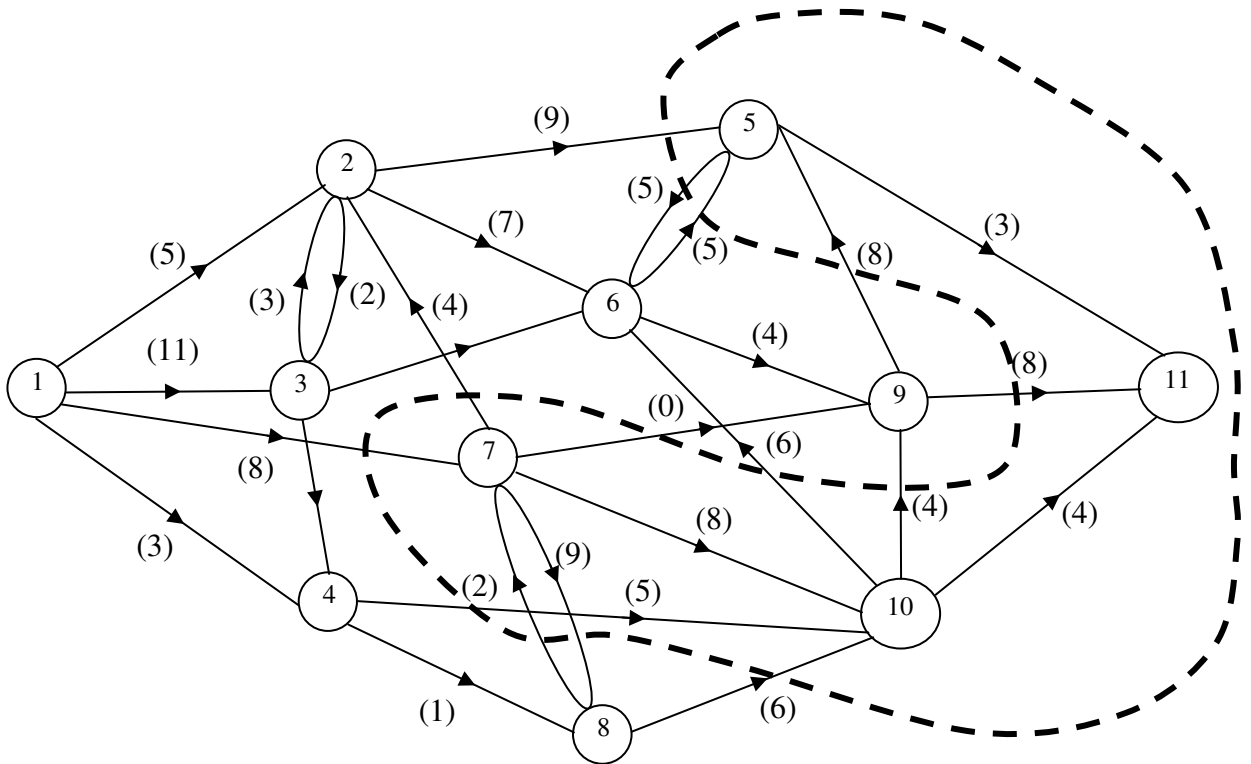


Рисунок 4

Насыщенные дуги. Дугу $u \in U$ называют насыщенной, если $\varphi(u) = c(u)$.

Пример потока в сети дан на рисунке 5, на котором первое число, поставленное в соответствие каждой дуге – ее пропускная способность, а второе – поток по дуге. Жирными линиями выделены насыщенные дуги, т.е. те, поток по которым равен их пропускной способности. Поток в этой сети не является максимальным.

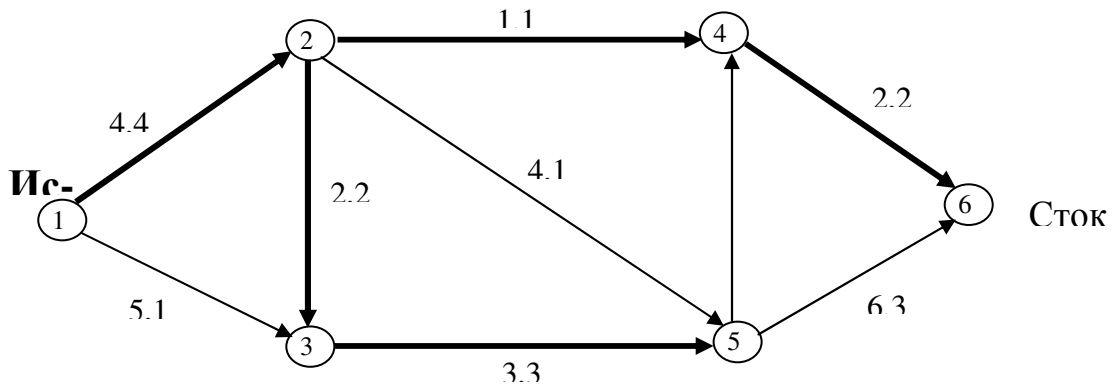


Рисунок 5
10

Матрицей пропускных способностей сети $G=(E,U)$ называется квадратная матрица C размерности $n \times n$, где $n=|E|$, каждому элементу C_{ij} которой соответствует величина пропускной способности дуги (X_i, X_j) .

Матрицей потоков сети $G=(E,\Gamma)$ называется квадратная матрица F размерности $n \times n$, где $n=|E|$, каждому элементу F_{ij} которой соответствует величина потока, протекающего по дуге (X_i, X_j) .

Существенным моментом в решении многих задач является нахождение пропускной способности графа. Например, в транспортной задаче, необходимо найти, сколько единиц транспорта проследует через сеть дорог в некоторой области.

Таким образом, математически задача максимального потока формулируется следующим образом:

Задача о максимальном потоке. Требуется определить такую функцию $\varphi(u)$, $u \in U$, что φ_{X_N} принимает максимальное значение.

2.2 Алгоритм Форда-Фалкерсона нахождения максимального потока в сети

В качестве теоретической базы алгоритма для решения указанной задачи использованы следующие теоремы. Для простоты записи положим $\varphi = \varphi_{X_N}$.

Теорема 1. Пусть $\mu=(X_0, X_{i_1}, \dots, X_{i_n}, X_N)$ – путь, соединяющий вход X_0 с выходом X_N . Если все дуги этого пути ненасыщенные, то можно увеличить φ , не нарушая соотношения.

Действительно, рассмотрим разности

$$\delta(u)=c(u)-\varphi(u)>0$$

и выберем среди них наименьшую δ^* . Увеличивая на δ^* поток в каждой дуге приходим к потоку $\varphi + \delta^*$.

Пусть $v=(X_0, X_{i_1}, \dots, X_{i_n}, X_n)$ – цепь, соединяющая вход X_0 с выходом X_n . Предположим, что она не является путем. Если ориентация дуги совпадает с направлением прохождения цепи, то будем обозначать ее через u^{\rightarrow} , в противном случае – через u^{\leftarrow} .

Положим

$$\delta(u^{\rightarrow})=c(u^{\rightarrow})-\varphi(u^{\rightarrow}),$$

$$\varphi^*=\min \varphi(u^{\leftarrow}),$$

$$\delta^*=\min \delta(u^{\leftarrow}),$$

$$\varepsilon^*=\min[\varphi^*,\delta^*].$$

Теорема 2. Если $\varepsilon^*>0$, то, увеличивая на ε^* поток на каждой дуге u^{\rightarrow} и уменьшая на ε^* поток на каждой дуге u^{\leftarrow} цепи v , мы увеличиваем поток φ также на ε^* .

Полный поток. Поток φ транспортной сети назовем *полным*, если любой путь, соединяющий X_0 с X_n , содержит по крайней мере одну насыщенную дугу.

Иначе говоря, нельзя увеличить φ , рассматривая только пути; хотя это иногда можно сделать, рассматривая цепи, соединяющие X_0 с X_n .

Теорема 3. Если не существует цепи v от X_0 до X_n с $\varepsilon^*>0$, то поток φ нельзя больше увеличить, то есть он максимальный.

Действительно, исключая из транспортной сети все дуги, для которых $\delta(u^{\rightarrow})=0$ или $\varphi(u^{\leftarrow})=0$, получаем по крайней мере два несвязных подграфа (в противном случае существовала бы ненасыщенная цепь, соединяющая X_0 с X_n). Множество вершин того из них, среди вершин которого содержится X_n , обозначим через A .

Очевидно, что A определяет разрез U_A^- и

$$\varphi_{X_N}^0 = \sum_{u \in U_A^-} \varphi(u) - \sum_{u \in U_A^+} \varphi(u) = \sum_{u \in U_A^-} c(u) + 0 = c(U_A^-).$$

Отсюда и из того, что общее количество вещества, входящего в A , не превосходит $c(U_A^-)$, заключаем, что все дуги $u \in U_A^-$ насыщенные, то есть $\varphi_{X_N}^0$ – максимальный поток. Из теор. 1,2 и 3 вытекает следующая основная теорема.

Теорема 4 (теорема Форда-Фалкерсона). Для заданной транспортной сети максимальное значение потока равно минимальной пропускной способности разреза, т. е.

$$\max \varphi_{X_N} = \min_{A \mid X_0 \notin A, X_N \in A} c(U_A^-).$$

2.3 Приложение алгоритма Форда-Фалкерсона к решению задачи нахождения максимального потока

Опишем алгоритм Форда-Фалкерсона в приложении к конкретной задаче.

Пусть необходимо найти максимальный поток транспортной сети, графическое изображение которой представлено на Рисунке 6.

Пусть узлом-источником является вершина 1, узлом-стоком – вершина 6. Построим максимальный поток (F) между этими вершинами. Начальное значение F нулевое. Очевидно, что структурой данных для описания F является матрица того же типа, что и матрица C, в которой определены пропускные способности дуг.

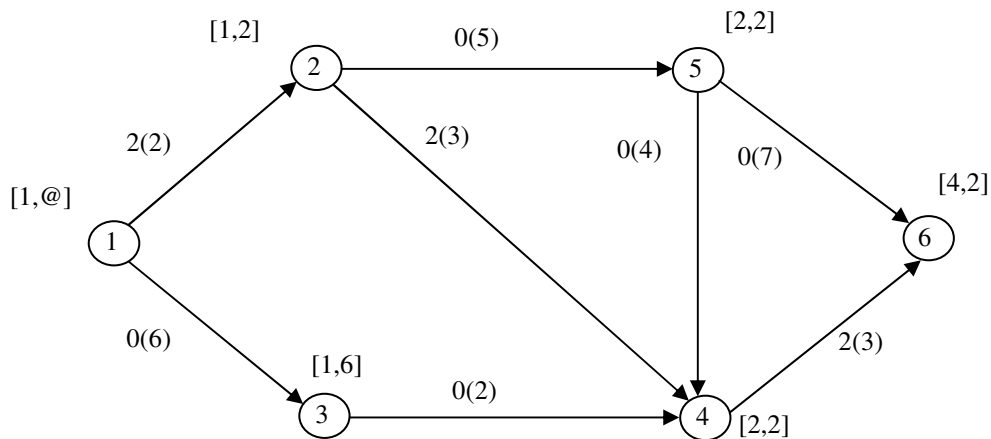


Рисунок 6

1. Первая итерация.

Первый шаг. Присвоим вершине 1 метку [1,@]. Рассмотрим дуги, началом которых является вершина 1 – дуги (1,2) и (1,3). Вершины 2 и 3 не помечены, по этому присваиваем им метки, для 2-й – [1,2] и 3-й – [1,6]. Что представляют из себя метки? Первая цифра – номер вершины, из которой идет поток, вторая цифра – численное значение потока, который можно передать по этой дуге.

Второй шаг. Выберем помеченную, но не просмотренную вершину. Первой в соответствующей структуре данных записана вершина 2. Рассмотрим дуги, для которых она является началом – дуги (2,4) и (2,5). Вершины 4 и 5 не помечены. Присвоим им метки [2,2] и [2,2]. Итак, на втором шаге вершина 2 просмотрена, вершины 3,4,5 помечены, но не просмотрены, остальные вершины не помечены.

Третий шаг. Выбираем вершину 3. Рассмотрим дугу (3,4). Вершина 4 помечена. Переходим к следующей вершине – четвертой, соответствующая дуга – (4,6). Вершина 6 не помечена. Присваиваем ей метку [4,2]. Мы достигли вершины-стока, тем самым, найдя путь (последовательность дуг), поток по которым можно увеличить. Информацию об этом пути содержат метки вершин. В данном случае путь $1 \rightarrow 2 \rightarrow 4 \rightarrow 6$. Максимально возможный поток, который можно передать по дугам этого пути, определяется второй цифрой метки вершины стока, то есть 2. Поток в сети стал равным 2.

2. Вторая итерация.

Первый шаг. Присвоим вершине 1 метку [1,@]. Рассмотрим дуги, началом которых является помеченная вершина 1. Это дуги (1,2) и (1,3). Вершина 2 не может быть помечена, так как пропускная способность дуги (1,2) исчерпана. Вершине 3 присваиваем метку (1,6).

Второй шаг. Выберем помеченную, но не просмотренную вершину. Это вершина 3. Повторяем действия. В результате вершина 4 получает метку [3,2].

Третий шаг. Выбираем вершину 4, только она помечена и не просмотрена. Вершине 6 присваиваем метку [4,1]. Почему только одна единица потока?

На предыдущей итерации израсходованы две единицы пропускной способности данной дуги, осталась только одна. Вершина-сток достигнут. Найден увеличивающий поток путь. Это путь $1 \rightarrow 3 \rightarrow 4 \rightarrow 6$, по которому можно передать единичный поток. Результирующий поток в сети равен трем. Изменение состояния сети на этой итерации показано на рисунке 7

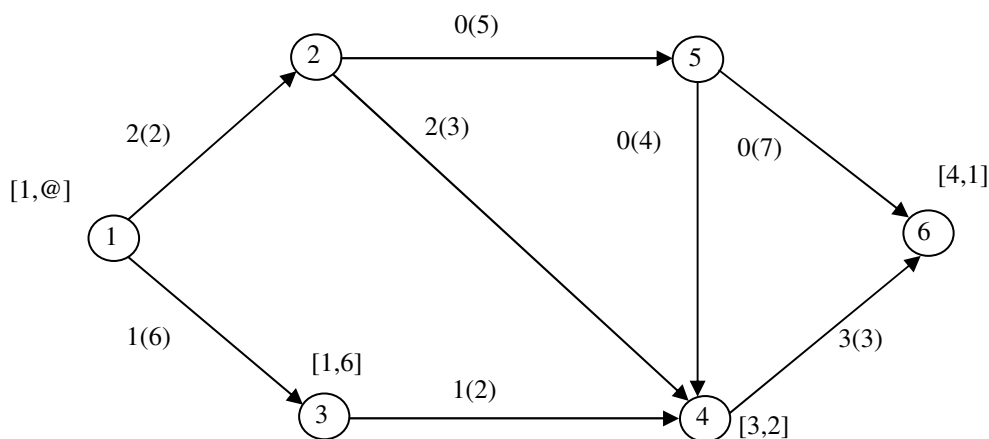


Рисунок 7

3. Третья итерация (рисунок 8). Вершине 1 присваиваем метку [1, @].

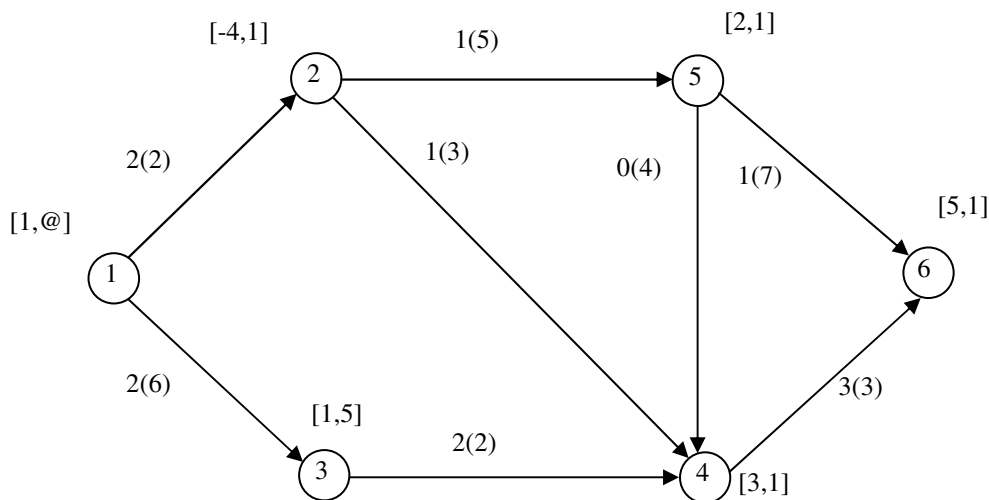


Рисунок 8

Первый шаг. Результат метка [1,5] у вершины 3. Второй шаг – метка [3,1] у вершины 4. Пропускная способность дуги (4,6) израсходована полностью. Однако есть обратная дуга (2,4), по которой передается поток, не равный нулю.

Попробуем перераспределить поток. Нам необходимо передать из вершины 4 поток, равный единице (зафиксирован в метке вершины). Задержим единицу потока в вершине 2, то есть вернем единицу потока из вершины 4 в вершину 2. Эту особенность зафиксируем в метке вершины 2 – $[-4,1]$. Тогда единицу потока из вершины 4 мы передадим по сети вместо той, которая задержана в вершине 2, а единицу потока из вершины 2 попытаемся передать по сети, используя другие дуги. Итак, вершина 4 просмотрена, вершина 2 помечена, вершины 5 и 6 не помечены. Четвертый и пятый шаги очевидны. Передаем единицу потока из вершины 2 в вершину 6 через вершину 5. Вершина-сток достигнута, найдена цепочка $1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 6$, по которой можно передать поток равный единице. При этом по прямым дугам поток увеличивается на единицу, а по обратным – уменьшается. Суммарный поток в сети – 4 единицы.

4. Четвертая итерация (рисунок 9). Вершине 1 присваиваем метку $[1, @]$. Первый шаг. Помечаем вершину 3 – $[1,4]$. Второй шаг. Рассматриваем помеченную, но не просмотренную вершину 3. Из нее выходит одна дуга – $(3,4)$. Вершину 4 пометить не можем – пропускная способность дуги исчерпана.

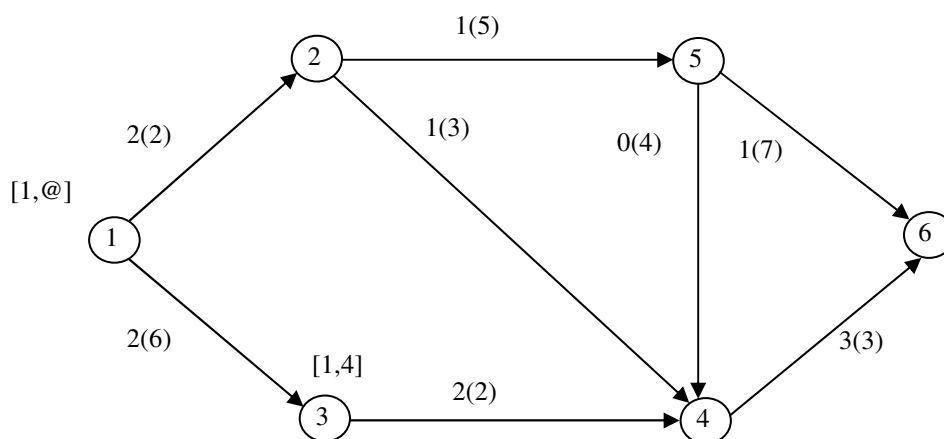


Рисунок 9

Помеченных вершин больше нет, и вершина сток не достигнута. Увеличивающую поток цепочку построить не можем. Найден максимальный поток в сети, равный 4. Можно заканчивать работу.

2.4. Численная и программная реализации решения задачи оптимизации потока в сети

2.4.1 Преимущества реализации в среде Delphi

Воспользоваться программной средой Delphi для решения задач прикладной комбинаторики оказывается предпочтительнее, чем другими программными средами. Это программное обеспечение является весьма мощным продуктом. Оно позволяет наиболее полно использовать все возможности весьма популярной сегодня операционной системы Windows. Выполненная в этой среде программа имеет все достоинства и такой же внешний вид, как и продукция компании Microsoft. Быстрая компиляция, удобное построение программы, визуальная компоновка интерфейса создаваемой программы. Эти и большое количество других преимуществ перед другими программными средами позволяют остановить свой выбор в создании собственного программного продукта на Delphi. Стоит отметить, что Delphi написана на основе уже многим известного Borland Pascal, поэтому переход на этот более удобный в пользовании язык не составит труда для большей части программистов.

2.4.2 Описание этапов работы программы

Программа была создана для того, что бы находить максимальную пропускную способность сети. Кроме того, она может решать еще и некоторые дополнительные задачи. Стоит отметить несомненное преимущество этой программы перед другими, что она может применяться не только в качестве прикладного пакета, но и с большим успехом использоваться в качестве обучающей программы. Программе присущ весьма удобный и интуитивно понятный пользовательский интерфейс, который доступен всем, не знакомым с основами программирования. Кроме того имеется поэтапная визуализация, как самого задания графа, так и процессов распределения потоков в нем.

2.4.2.1 Запуск программы

Для того, что бы запустить программу нужно найти на диске файл Graph.exe и запустить его.

2.4.2.2 Структура окна

После запуска программы на экране будет отображено следующее окно (рисунок 10).

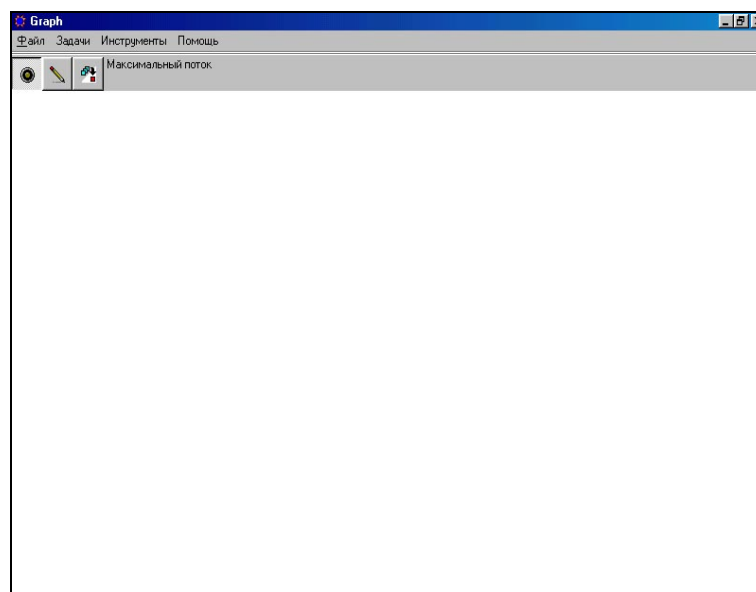


Рисунок 10

Строка меню состоит из четырех пунктов **Файл**, **Задачи**, **Инструменты**, **Помощь**. Более подробно о каждом пункте будет написано позже.

Под строкой меню располагается панель инструментов. На панели располагается три кнопки слева направо: Расстановка вершин, Расстановка связей, Источник и сток. Справа от кнопок находится надпись «Максимальный поток» возле нее выдается максимальная пропускная способность сети после ее расчета. Белая область называется рабочей областью (используется для отображения сети и распределения потоков в ней).

2.4.2.3 Ввод вершин графа

По умолчанию первая кнопка слева, находящаяся на панели инструментов нажата. Это означает, что программа находится в режиме «расстановка вершин». В данном режиме при клике по рабочей области (белое поле) на экране будут отображаться вершины графа, пронумерованные в порядке возрастания (рисунок 11).



Рисунок 11

Для того чтобы переключаться между режимами, можно использовать и меню. Пункт меню **Инструменты** → **Расстановка вершин**.

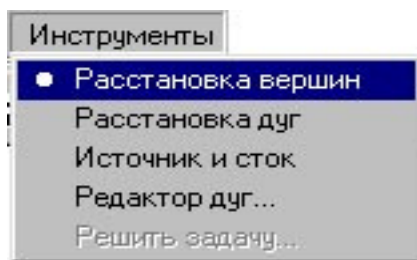


Рисунок 12

2.4.2.4 Ввод дуг

После расстановки вершин следует задать дуги. Для этого требуется либо нажать соответствующую кнопку на панели инструментов либо воспользоваться меню **Инструменты** → **Расстановка дуг**. После этого для задания дуги между парой вершин нужно последовательно кликнуть вначале по одной затем по другой. При щелчке по второй вершине на экран выводится запрос (рисунок 13).

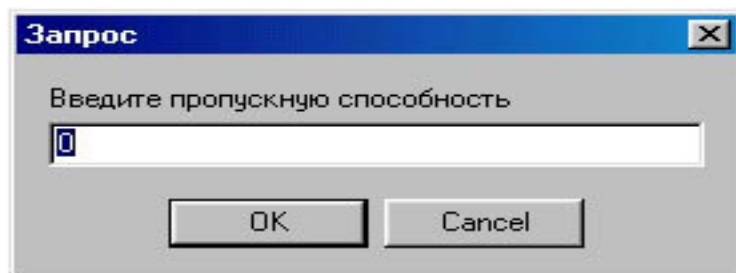


Рисунок 13

В нем следует указать ту пропускную способность, которую вы хотите присвоить данной дуге, после чего следует нажать ОК. В том случае если будет введен ноль, то дуга, отображаться не будет (то же произойдет при нажатии кнопки Cancel). После задания всех дуг на экране будет отображен граф (рисунок 14)

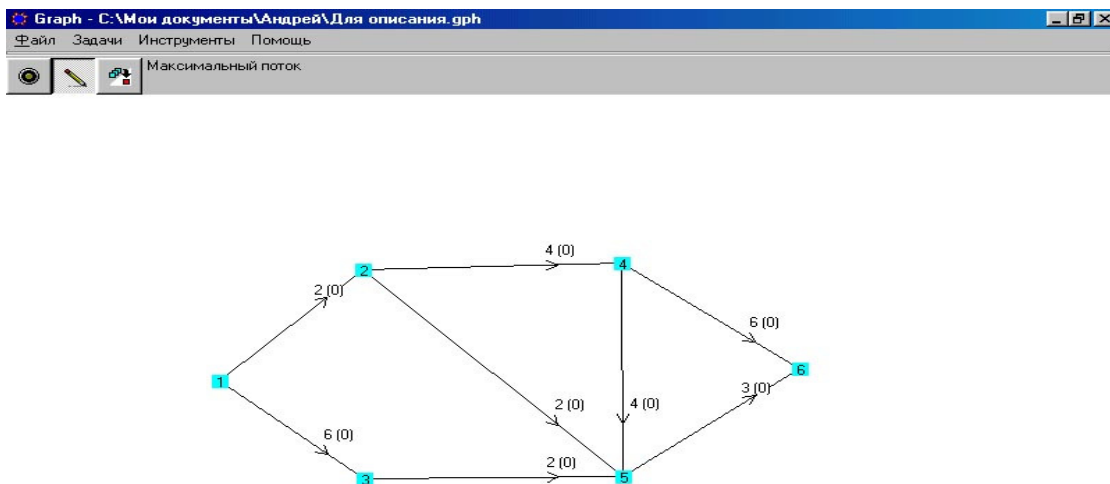


Рисунок 14

В изображении графа направления дуг указаны стрелками, а цифрами возле дуги указаны пропускные способности дуг.

2.4.2.5 Задание источника и стока

Как видно на рисунке 12, пункт меню **Решить задачу** не активен. Это связано с тем, что в сети не заданы вершина источник и вершина сток. Для того чтобы расставить эти вершины нужно нажать кнопку или в пункте меню **Инструменты** → **Источник и сток**. Затем последовательно щелкнуть вначале по вершине которую следует сделать источником, а затем по той которую нужно сделать стоком (при этом источник будет отображаться синим цветом, а сток красным).

2.4.2.6 Решение задачи

После этого можно в пункте меню **Инструменты** выбрать подпункт **Решить задачу**. На экран будет выведено сообщение, в котором будет указана максимальная пропускная способность сети (рисунок 15).

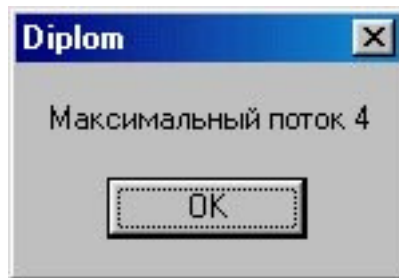


Рисунок 15

После нажатия ОК на экране будет отображен граф следующего вида (рисунок 16).

Здесь толстыми линиями показаны те дуги, пропускная способность которых используется полностью. Возле значений пропускных способностей, в скобках, указана величина потока проходящего по данной дуге.

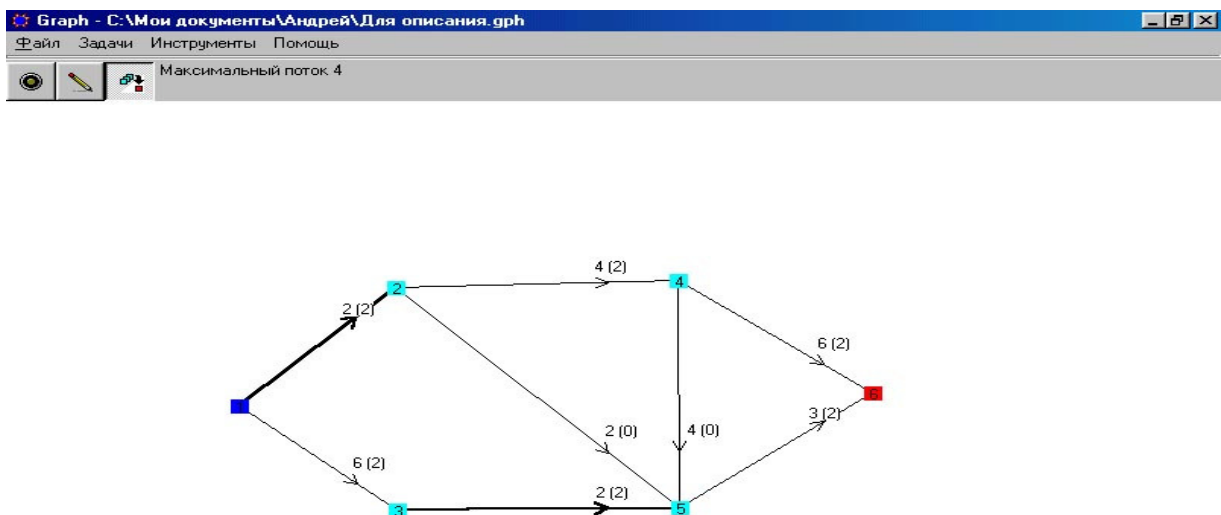


Рисунок 16

2.5 Дополнительные возможности редактирования.

2.5.1 Удаление дуг

Если во время задания сети нечаянно введена не та дуга, то следует воспользоваться редактором дуг **Инструменты** → **Редактор дуг**. На экране отобразится следующее (рисунок 17).

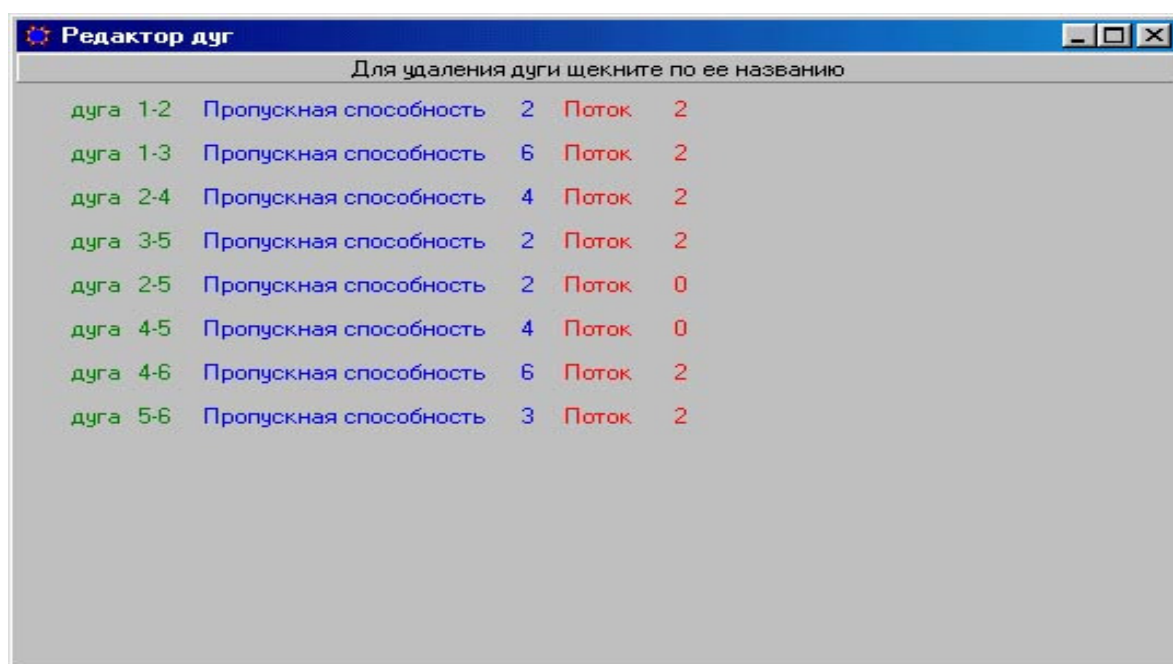


Рисунок 17

Для того, что бы удалить дугу нужно кликнуть по ее цифровому обозначению. Например, для удаления дуги 2-4 нужно нажать на цифры 2-4 (при этом цвет их изменится на серый). После закрытия редактора на экране будет отображен граф без дуги 2-4.

2.5.2 Удаление вершин

Для удаления неправильно введенной вершины требуется щелкнуть по ней правой кнопкой мыши. Поле этого появится контекстное меню (рисунок 18).

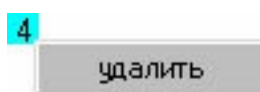


Рисунок 18

Если нажать на слово удалить, то вершина будет удалена (вместе с вершиной удаляются все дуги, которые связаны с этой вершиной).

2.5.3 Удаление всего графа

Для того, что удалить весь граф и вернуть программу в исходное состояние можно воспользоваться меню **Файл** → **Очистить** (рисунок 19).

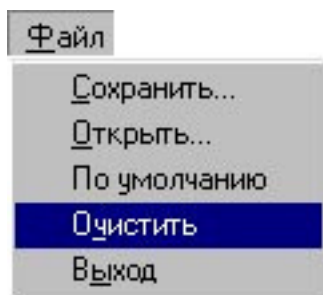


Рисунок 19

2.5.4 Сохранение в файл и открытие из файла

Если требуется продолжить работу с графом через некоторое время, то можно воспользоваться возможностью сохранения графа в файл. Для этого надо нажать в строке меню **Файл** → **Сохранить**. На экране появится диалог сохранения файла (рисунок 20).

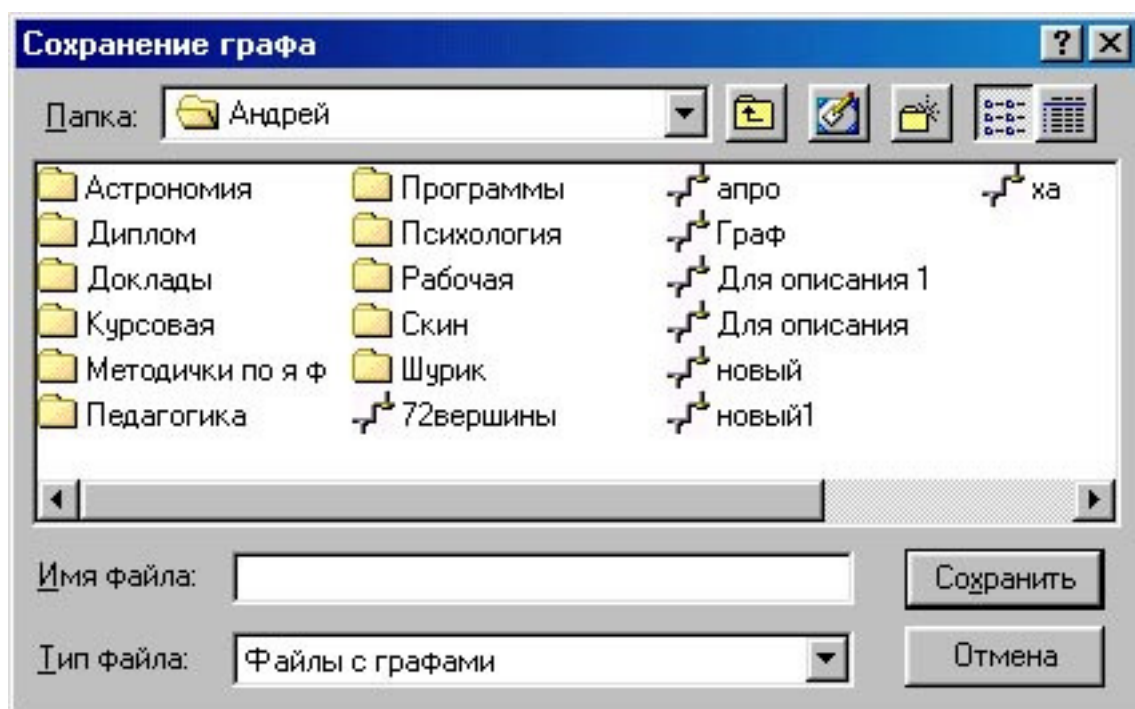


Рисунок 20

В разделе имя файла вводится то имя, под которым нужно сохранить граф, после чего надо нажать кнопку сохранить. Программа автоматически присваивает сохраняемым файлам расширение *.grh. Для открытия графа нужно нажать **Файл** → **Открыть**, появляется диалог открытия файла, выбрав в нем нужный и нажав кнопку открыть на экране можно увидеть сохраненный граф.

2.5.5 Задание пропускной способности по умолчанию

Если во вводимом графе большое число дуг имеет одинаковую пропускную способность, то можно воспользоваться заданием пропускной способности дуги по умолчанию. Для этого в строке меню выбрать **Файл** → **По умолчанию** и программа выдаст запрос (рисунок 21).

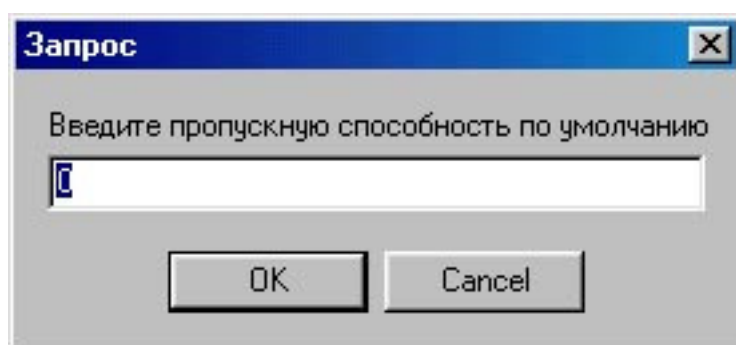


Рисунок 21

После введения нужной цифры следует нажать ОК. После этой операции при задании дуги она будет отображаться без запроса пропускной способности (ее пропускная способность будет всегда равна заданной).

2.6 Применение программы для решения некоторых других задач

Кроме задачи о нахождении максимального потока в программе также предоставлена возможность решения двух других задач. Это задача о нахождении максимального паросочетания и транспортная задача. Переключение между задачами происходит в пункте меню задачи (рисунок 22).

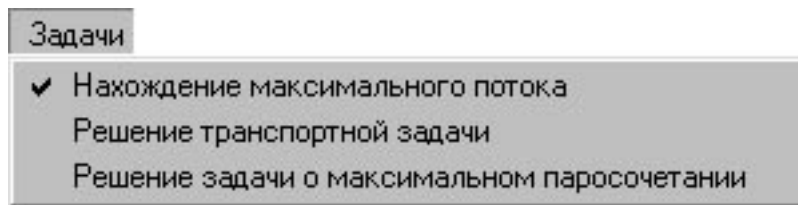


Рисунок 22

2.6.1 Транспортная задача

Переключаться на решение транспортной задачи следует до начала введения графа. Это связано с тем, что при задании транспортной сети появляется еще один параметр — стоимость пути. Запрос этого параметра происходит сразу после введения пропускной способности и выглядит следующим образом (рисунок 23)

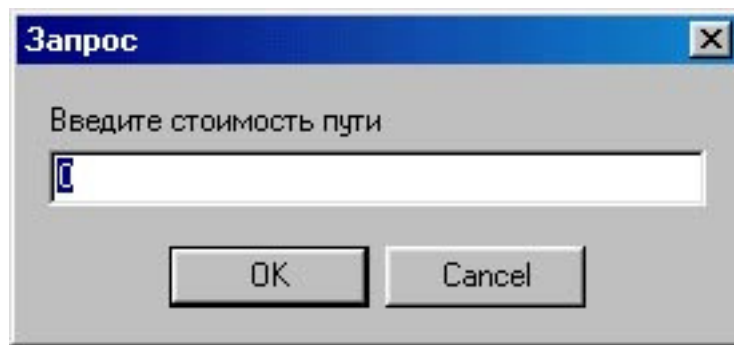


Рисунок 23

Также после решения этой задачи, наряду с вычислением пропускной способности, появляется второе сообщение о величине стоимости пути, которое следует сразу после сообщения о величине максимального потока (рисунок 24).

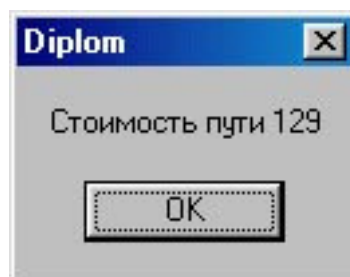


Рисунок 24

После решения данной задачи граф выглядит следующим образом (рисунок 25). Появилась стоимость пути. Она отображается в фигурных скобках.

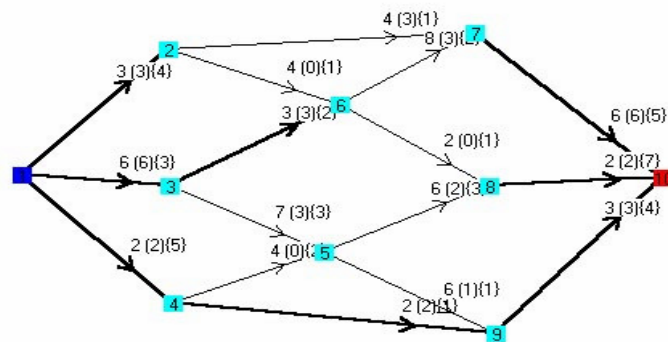


Рисунок 25

2.6.2 Задача о максимальном паросочетании

При решении задачи о максимальном паросочетании граф строится особым образом (рисунок 26):

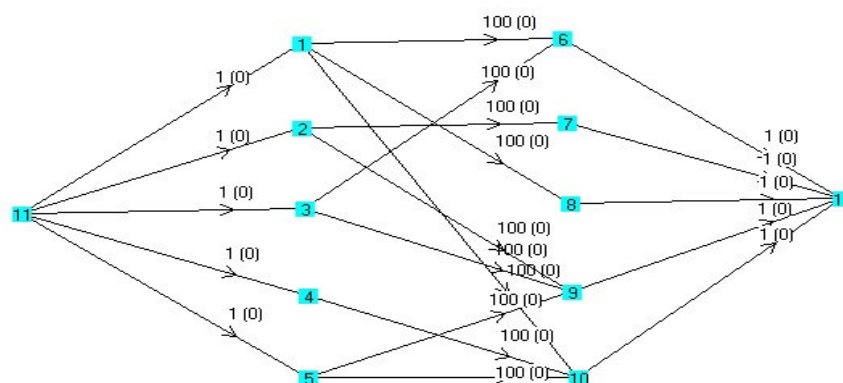


Рисунок 26

Между вершинами, располагающимися в два столбца, дуги имеют очень большую пропускную способность, а оставшиеся дуги имеют пропускную способность равную единице. После решения задачи граф имеет следующий вид (рисунок 27).

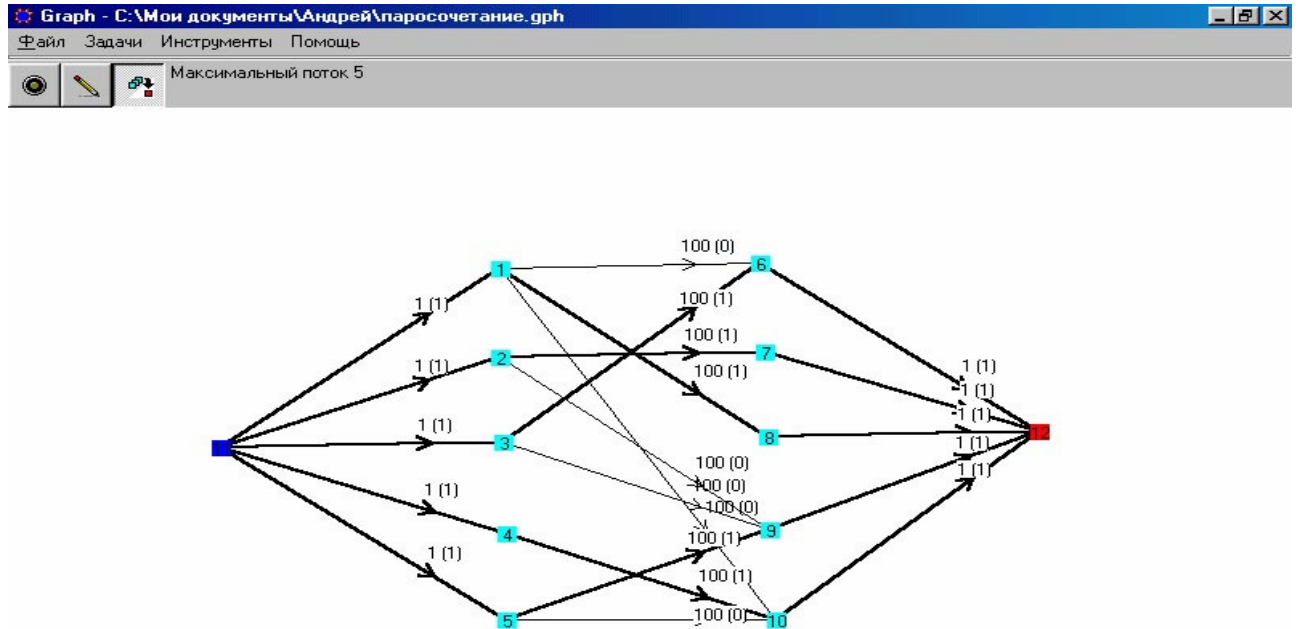


Рисунок 27

Если представить, что один столбец это служащие, другой столбец вакансии, а дуги между ними это всевозможные назначения на должности, то дуги выделенные толстыми линиями представляют те назначения, которые нужно сделать, что бы всех распределить.

2.7 Результаты решения некоторых задач

В качестве демонстрации возможностей работы программы, приведем примеры решения некоторых задач. Далее рассмотрим три типа решаемых задач, причем для каждого из типов укажем задачи различной сложности.

2.7.1 Задачи нахождение максимальной пропускной способности сети

Задача №1. Эта задача демонстрирует решение достаточно простого графа, для которого можно найти максимальную пропускную способность аналитически, используя алгоритм Форда-Фалкерсона. Она равна для данного гра-

фа, состоящего из 10 вершин и 20 дуг, 40 единиц переносимого вещества(рисунок 28) .

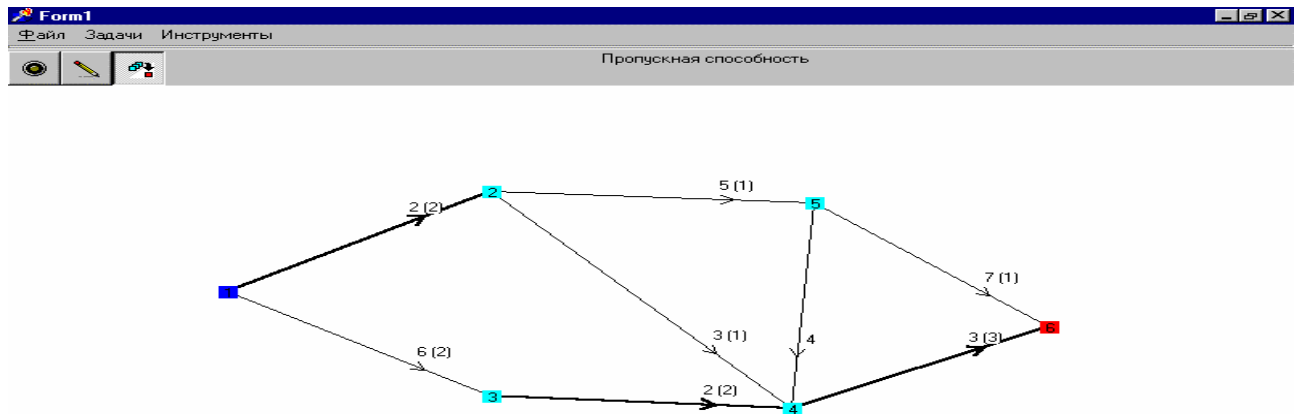


Рисунок 28

Задача №2. Демонстрируется нахождение максимальной пропускной способности для графа, аналитическое решение которого с помощью техники меток Форда-Фалкерсона представляет большую проблему.

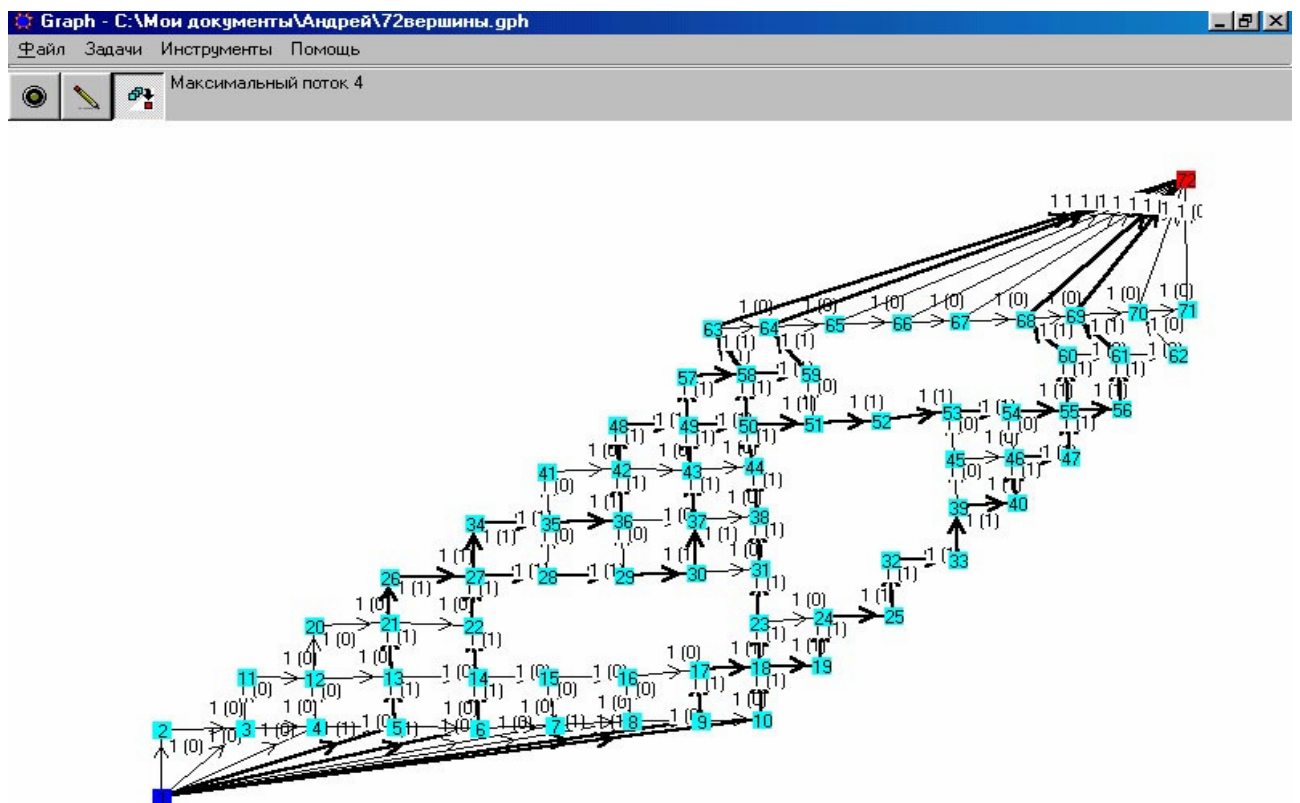


Рисунок 29

Эта сеть состоит из 72 узлов и около 120 дуг. Величина протекающего по сети потока равна 4 (рисунок 29).

2.7.2 Транспортная задача

Рассмотрим теперь следующий тип задач – транспортные, в которых кроме нахождения потока определяется еще и стоимость пути для этого потока.

Задача №3. Рассмотрим вначале задачу с небольшим количеством узлов. Узлов здесь 7, дуг 10, поток равен 4, стоимость пути 37 (рисунок 30).

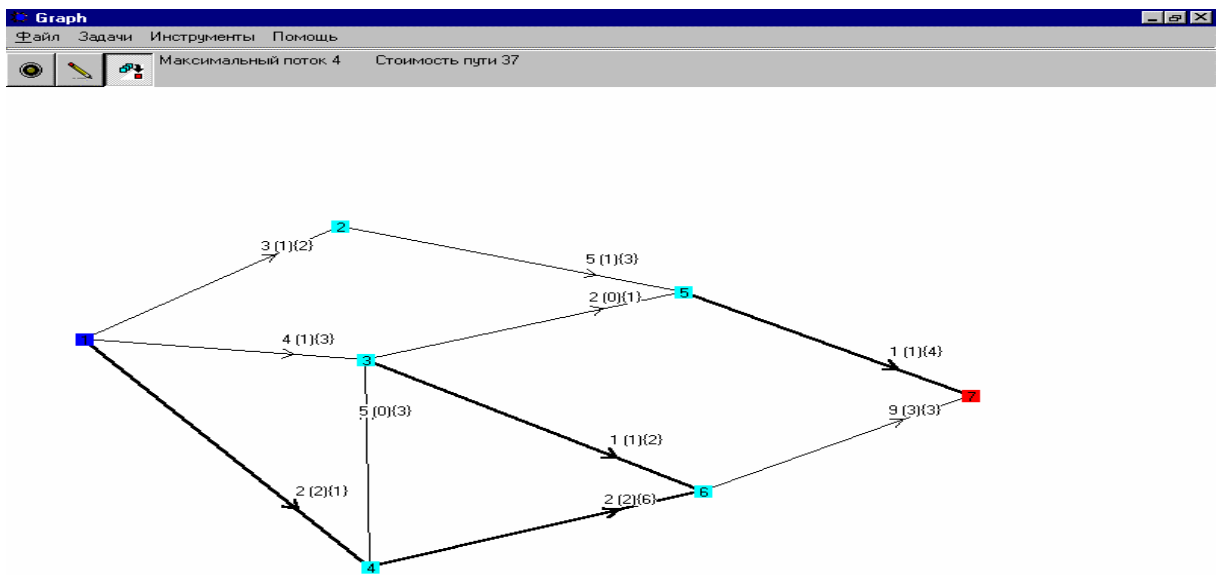


Рисунок 30

Задача №4. Следующая задача состоит из 30 узлов и 50 дуг

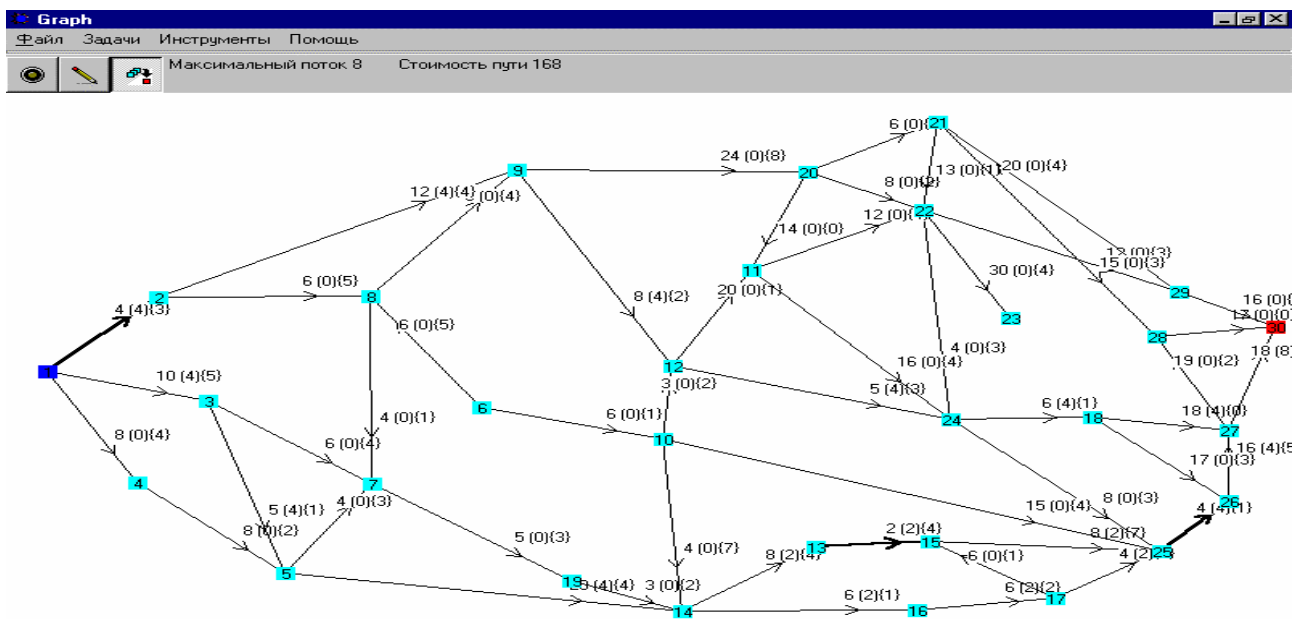


Рисунок 31

2.7.3 Задача о построении максимального паросочетания

в двудольном графе

Решение подобного рода задач используется для моделирования процессов, происходящих в жизни. Например, при распределении заданного количества людей по определенному количеству должностей. Граф при этом строится следующим образом. В два столбца выстраиваются вершины: первый столбец – служащие, второй столбец – должности, и вершины-служащие соединяются с вершинами-должностями в направлении, обратном тому, которое задано в исходной задаче (исходное направление соответствует должностям, на которые в принципе могут быть назначены служащие). Затем ставятся две вершины слева и справа от этих столбцов, они будут являться источником и стоком. Дуги, идущие от источника и к стоку имеют пропускную способность равную единице, а дуги, соединяющие вершины в столбцах имеют пропускную способность больше, чем по ним может пройти в данном графе. После решения задачи толстыми линиями между столбцами указаны те пары, которые соответствуют друг другу.

Задача №5. Задача решается для двудольного графа $G=(X,Y,U)$, где X – множество вершин, из которых исходят дуги, а Y – множество вершин, в которые заходят дуги, $|X|=5$, $|Y|=5$. Жирными линиями, соединяющими вершины исходного двудольного графа, на рисунке 32 указаны дуги, входящие в максимальное паросочетание.

Задача №6. Задача решается для двудольного графа $G=(X,Y,U)$, где X – множество вершин, из которых исходят дуги, а Y – множество вершин, в которые заходят дуги, $|X|=8$, $|Y|=9$. Жирными линиями, соединяющими вершины исходного двудольного графа, на рисунке 33 указаны дуги, входящие в максимальное паросочетание.

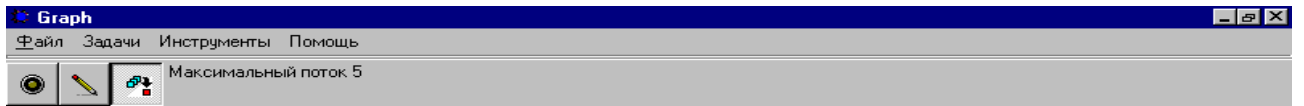


Рисунок 32

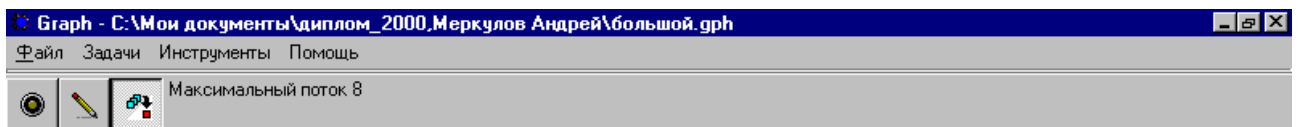


Рисунок 33

3 Кратчайшие пути

3.1 Формулировка задачи нахождения кратчайшего расстояния в терминах теории графов

На практике часто возникает задача минимизации расстояния между двумя заданными точками при условии, что существует не менее одного возможного пути. Очевидный пример – маршрут движения по городу.

Существенным моментом в решении многих задач является нахождение пути (маршрута) минимальной длины в графе. Например, в транспортной задаче, необходимо найти, кратчайший путь по дорогам в некоторой области.

Таким образом, математически задача о поиске кратчайшего расстояния формулируется следующим образом:

Задача о минимальном пути. Требуется определить такую функцию $\varphi(u)$, $u \in U$, что φ_{x_N} принимает минимальное значение.

3.2 Алгоритм Дейкстры для нахождения минимального пути в графе

Рассмотрим способ задания взвешенного графа, т.е. графа, каждому ребру которого соответствует некий параметр - вес. Для определения такого графа используется матрица весов W , размер которой $n \times n$, где n - количество вершин графа. При этом элемент w_{ij} равен весу ребра, соединяющего i -ю и j -ю вершины. Если такого ребра нет, то w_{ij} полагаем равным бесконечности (на практике это максимально число, возможное на данном языке программирования). Этот способ задания используется, например, в алгоритмах поиска пути во взвешенном графе.

Рассмотрим поиск пути минимальной суммарной длины во взвешенном графе с неотрицательными весами, который реализуется с помощью алгоритма Дейкстры. Процедура находит путь минимального веса в графе $G=(V,E)$, заданном весовой матрицей W , у которой элемент w_{ij} равен весу ребра, соединяющего i -ю и j -ю вершины. При этом предполагается, что все элементы w_{ij} неотрица-

тельны. Путь ищется из вершины номер u_1 к вершине номер u_2 . Процедура использует алгоритм Дейкстры. Для бесконечности используется число 0.

Алгоритм, по которому происходит поиск, заключается в следующем:

1. всем вершинам приписывается вес - вещественное число, $d(i)=GM$ для всех вершин кроме вершины с номером u_1 , а $d(u_1)=0$;
 2. всем вершинам приписывается метка $m(i)=0$;
 3. вершина u_1 объявляется текущей - $t=u_1$;
 4. для всех вершин, у которых $m(i)=0$, пересчитываем вес по формуле:
 $d(i):=\min\{d(i), d(t)+W[t,i]\}$;
 5. среди вершин, для которых выполнено $m(i)=0$ ищем t_u , для которой $d(i)$ минимальна, если минимум не найден, т.е. вес всех не "помеченных" вершин равен бесконечности (GM), то путь не существует; Выход;
1. иначе найденную вершину с минимальным весом полагаем текущей и помечаем ($m(t)=1$)
 2. если $t=u_2$, то найден путь веса $d(t)$, Выход;
 3. переходим на шаг 4.

На выходе имеем переменную $length$, которая определяет длину пути ($length=-1$ если пути не существует, $length=0$, если $u_1=u_2$), переменную $Weight$ - вес пути и массив $Path$, содержащий последовательность номеров вершин, определяющих путь. В алгоритме не упомянуто, как же определить сам путь, но это легко выяснить, если посмотреть блок-схему.

Опишем алгоритм Дейкстры в приложении к конкретной задаче.

Пусть необходимо найти минимальный путь в транспортной сети, графическое изображение которой представлено на Рисунке 34. Пусть узлом-источником является вершина 1, узлом-стоком – вершина 6. Найдем минимальный путь (F) между этими вершинами. Начальное значение F нулевое.

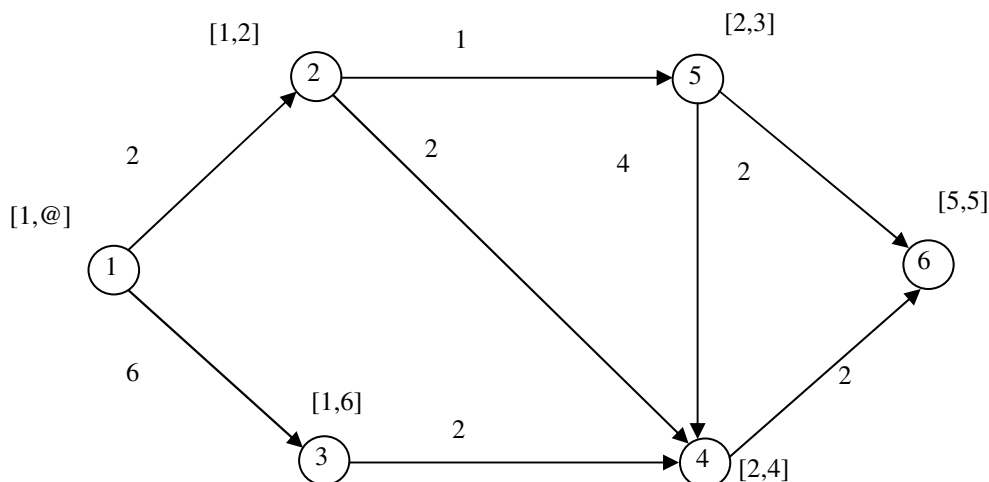


Рисунок 34

Первая итерация.

Первый шаг. Присвоим вершине 1 метку $[1, @]$. Рассмотрим дуги, началом которых является вершина 1 – дуги (1,2) и (1,3). Вершины 2 и 3 не помечены, по этому присваиваем им метки, для 2-й – $[1,2]$ и 3-й – $[1,6]$. Что представляют из собой метки? Первая цифра – номер вершины, из которой идет ребро, вторая цифра – численное значение длины пути пройденного по графу до данной вершины.

Второй шаг. Выберем помеченную, но не просмотренную вершину. Первой в соответствующей структуре данных записана вершина 2. Рассмотрим дуги, для которых она является началом – дуги (2,4) и (2,5). Вершины 4 и 5 не помечены. Присвоим им метки $[2,4]$ и $[2,3]$. Итак, на втором шаге вершина 2 просмотрена, вершины 3,4,5 помечены, но не просмотрены, остальные вершины не помечены.

Третий шаг. Выбираем вершину 3. Рассмотрим дугу (3,4). Вершина 4 помечена. Сравниваем, как на данном шаге мы можем пометить данную вершину, $[3,8]$ из 3-й вершины пройдя 8 единиц пути. Это длиннее чем 4 поэтому оставляем метку без изменения. Переходим к следующей вершине – четвертой, соответствующая дуга – (4,6). Вершина 6 не помечена. Присваиваем ей метку $[4,6]$. Далее переходим к следующей вершине. Вершина 5 помечена как $[4,6]$ сравниваем как на данном этапе можно пометить вершину, $[5,5]$ т.е. найден более короткий путь. Поэтому помечаем вершину заново как $[5,5]$ Мы достигли верши-

ны-стока, тем самым, найдя путь (последовательность дуг), прохождение по которой есть кратчайший путь. Информацию об этом пути содержат метки вершин. В данном случае путь $1 \rightarrow 2 \rightarrow 5 \rightarrow 6$. Минимально возможный путь, который существует в графе, определяется второй цифрой метки вершины стока, то есть 5. Кратчайший путь в графе имеет длину 5.

3.3 Описание структуры программы

Программа была создана для того, что бы находить минимальный (максимальный) путь в графе. Кроме того, она может решать еще и некоторые дополнительные задачи. Программе присущ весьма удобный и интуитивно понятный пользовательский интерфейс, который доступен всем, не знакомым с основами программирования.

3.3.1 Запуск программы

Для того, что бы запустить программу нужно найти в меню **Пуск** иконку с названием **Graph1** и нажать на нее.

3.3.2 Вид окна

После запуска программы на экране будет отображено следующее окно (Рисунок 35).

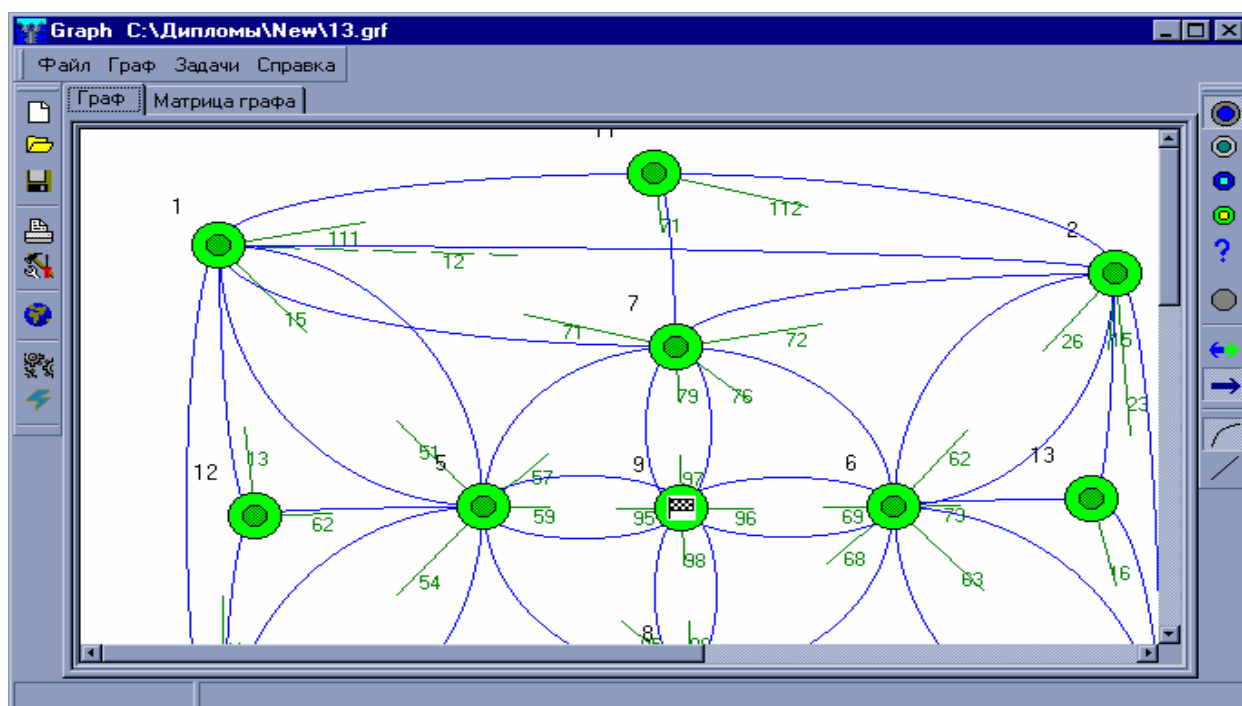


Рисунок 35

Строка меню состоит из четырех пунктов **Файл, Граф Задачи, Справка**. Более подробно о каждом пункте будет написано позже.

Главное окно программы состоит из:

- **Рабочей области**, которая служит для введения и редактирования графа.
- **Стандартной панели инструментов**, предназначенной для работы с файлами.
- **Панели настройки**, предназначенной для настройки компонент графа.
- **Панели поиска решения с помощью**, которой можно отображать найденные решения.

Также на вкладке "Матрица графа" имеются:

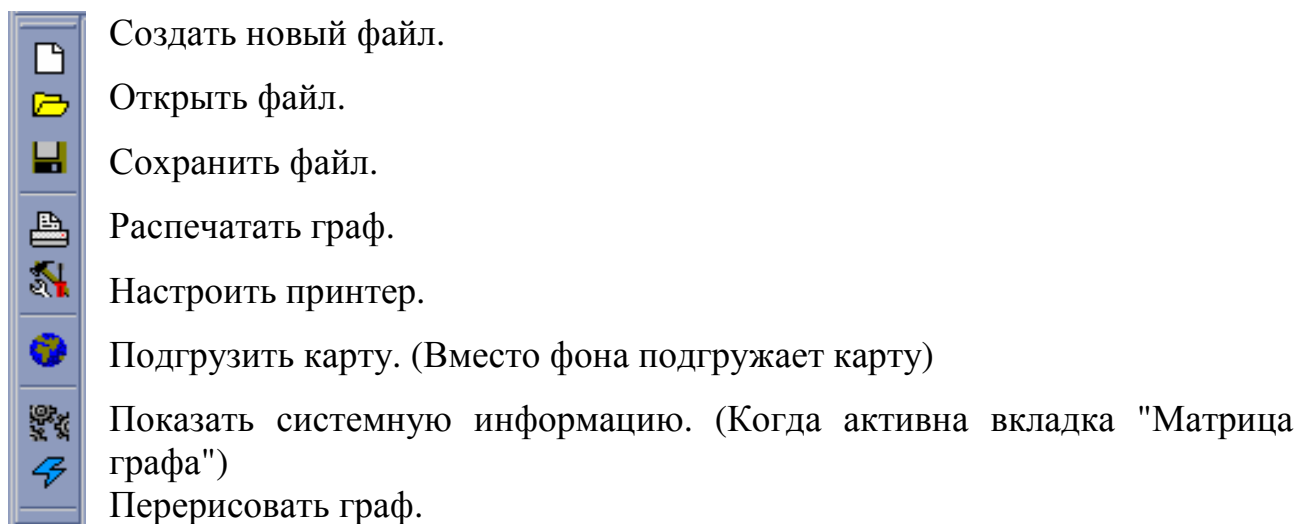
- **Матрица весов** графа, отображающая веса ребер графа.
- **Матрица системных данных**, отображающая системную информацию о графе.

3.3.3 Рабочая область

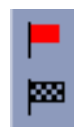

Рабочая область приложения предназначена для ввода и редактирования графа. Что производится с помощью мыши. Выше приведен вид главной формы с введенным графом на рабочей области.

3.3.4 Стандартная панель инструментов











Данная панель предназначена для работы с файлами, и обслуживание графа. Часть из этих кнопок повторяют элементы выбора из главного меню Файл. Назначение кнопок приведено дальше.



А также имеется еще две кнопки, которые появляются при выборе задачи требующей ввода начальной и конечной вершины.

-  Пометить начало пути. (Появляется, если этого требует задача)
-  Пометить конец пути. (Тоже)

3.3.5 Панель настройки

-  Вершина типа Мегополис.
-  Вершина типа Областной центр.
-  Вершина типа Районный центр.
-  Вершина типа Город.
-  Невидимая вершина. (Предназначена для ввода графа на карте)
-  Безтипная вершина.
-  Вводится неориентированный граф.
-  Вводится ориентированный граф.
-  Ребро графа выводится как дуга.
-  Ребро графа выводится как линии.

В главном меню, в пункте **Граф** имеются элементы выбора, предназначенные для настройки режима ввода весов ребер, имени вершины и места, где будет выводиться имя вершины. А также будет ли вообще вершина иметь имя.

Здесь надо сделать небольшое замечание: дело в том, что сами типы вершин никакой смысловой нагрузки не несут. Они нужны для того, чтобы удобнее было вводить граф, т.е. с помощью данных типов вершин можно нарисовать некое подобие географической карты тем самым, приближая понятие графа к более знакомым, а не абстрактным вещам. При работе с настоящей картой удобнее всего использовать невидимый тип вершины.

3.3.6 Панель поиска решения

Панель настройки отображается после того, как будет найдено решение. Ее изображение представлено на следующем рисунке 36.



Рисунок 36

Панель состоит из 3-х элементов. Кнопка, выводящая максимальный путь, минимальный путь и окно, в котором можно выбрать длину пути.

Здесь надо напомнить, что в программе используется модифицированный алгоритм Дейкстры. Поэтому программа ищет все пути. И среди них может оказаться несколько путей одинаковой длины. Которые можно отобразить с помощью окна находящегося на описываемой панели.

3.3.7 Матрица весов

В этой матрице отображаются веса ребер графа. Эти веса можно редактировать непосредственно в самой таблице. Матрица может быть заполнена нулями, если не стоит режим запрашивания веса ребра сразу же после введения ребра. По данным, содержащимся в этой матрице, решается выбранная вами задача. Вид этой матрицы представлен на рисунке 37. Эта матрица весов для графа, содержащего 12 точек. Цифры в пересечении столбцов и строк обозначают вес ребра между соответствующими вершинами.

Граф	Матрица графа											
Имена	1	2	3	4	5	6	7	8	9	10	11	12
1				41	51		71					13
2	12					62	72				112	
3		23				63	83		30			
4			34		54		84					
5	15			45			85	95				62
6		26	36				76	96				
7					57			97	71			
8						68		98	90			
9					59	69	79	89				
10				50								
11	111											
12				52								

Рисунок 37

3.3.8 Системная матрица

Данная матрица содержит системную информацию, по которой перерисовывается граф. Эти данные не подлежат изменению и вводятся в процессе ввода графа. Матрица содержит тип вершины, ее координаты на рисунке 38, имя и порядковый номер, то есть какой по счету эта вершина была введена. На рисунке показана системная матрица графа с 12 вершинами.

Номер	Имя	Тип вершины	Координата X	Координата Y	Радиус
1	1	Поселок	76	74	15
2	2	Поселок	570	92	15
3	3	Поселок	596	401	15
4	4	Поселок	78	410	15
5	5	Поселок	222	241	15
6	6	Поселок	448	241	15
7	7	Поселок	328	139	15
8	8	Поселок	336	350	15
9	9	Поселок	331	242	15
10	10	Поселок	322	450	15
11	11	Поселок	316	28	15
12	12	Поселок	96	247	15

Рисунок 38

3.3.9 Форма настройки

Данная форма предназначена для настройки описанных выше компонент графа. Она содержит следующие закладки.

- **Ребра.** На данной закладке находятся элементы настройки формы отображаемого ребра в виде дуги или в виде линии. Здесь же настраивается цвет и стиль выводимого ребра, цвет, которым будет выводиться найденное решение, а также режим запроса веса ребра (длины). Ее вид представлен на рисунке 39. Данную форму можно вызвать из главного меню Граф\ Общие настройки.

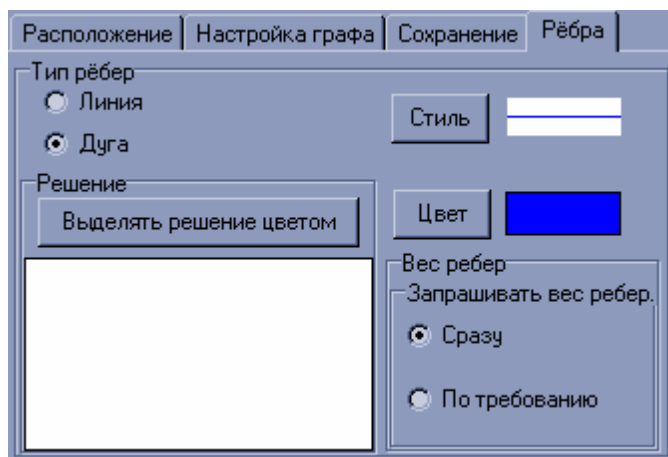


Рисунок 39

- **Настройка графа.** На данной закладке находятся компоненты для настройки типов используемых вершин (рисунок 40). Здесь же можно изменить соответствующий тип вершины с помощью кнопки настройка. Режима запроса имени вершины.

Присваивать имена Места, где будет выводится имя вершины Выводить имя. Типа графа. При направленном, все ребра проходимы только в одном направлении. При ненаправленном, ребра проходимы в обоих направлениях. А также максимально возможный радиус вершины.



Рисунок 40

Расположение. На данной закладке расположена информация об используемых в программе директориях (рисунок 41).

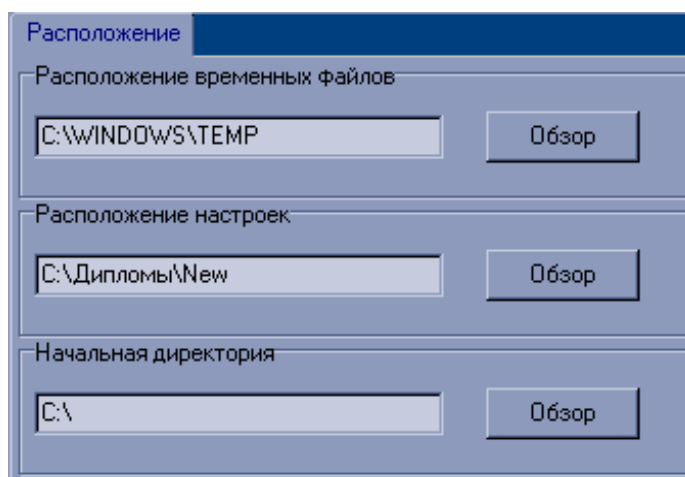


Рисунок 41

- а также закладка **Сохранение**, на которой устанавливается промежуток автосохранения. Эта закладка представлена на рисунке 42, на котором также отображается сама форма настройки.

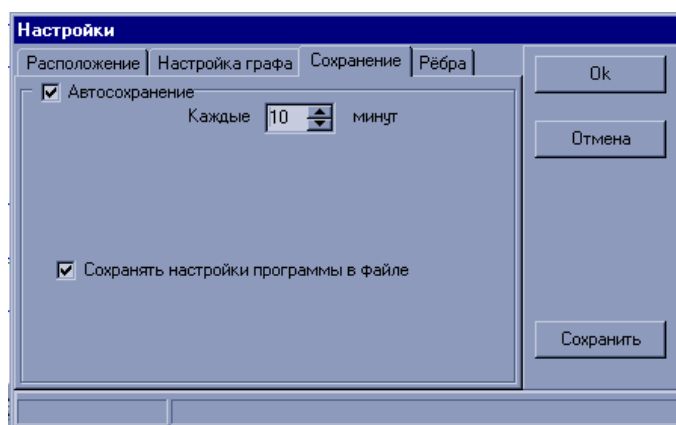


Рисунок 42

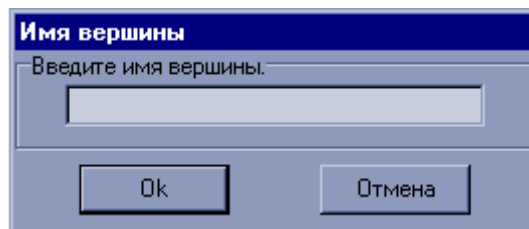
По остальным формам и функциям, которые используются в данной программе, справка не требуется, так как работа с ними интуитивно понятна.

3.4 Описание этапов работы программы

3.4.1 Ввод вершин графа.

Для того чтобы ввести вершину графа необходимо щелкнуть по рабочей области. Щелчок левой кнопки мыши по рабочей кнопке выводит вершину текущего типа (кнопка такой вершины нажата на панели настройки).

Рисунок 43



После ввода вершины, если стоит режим запроса имени вершины сразу, появится запрос на ввод имени вершины (рисунок 43). Здесь надо ввести имя новой вершины и нажать **Ок**.

3.4.2 Ввод дуг

Ребро графа вводится передвижением мыши с нажатой левой кнопкой от одной вершины графа, до другой. Причем выводится тот тип ребра, кнопка которого нажата на Панели настройки. Граф использует только один тип ребер, и поэтому смена типа ребер приводит к перерисовке всех существующих ребер данным типом ребра. После ввода ребра, дуги графа, если установлен режим ввода веса ребра сразу после ее ввода появится следующее окно (рисунок 44):

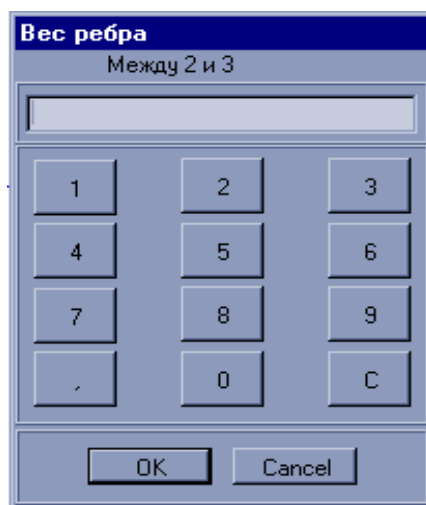




Рисунок 44

В нем следует указать ту пропускную способность, которую вы хотите присвоить данной дуге, после чего следует нажать **ОК** или **Enter**. Если будет нажата кнопка **Cancel**, то ребру будет присвоен вес равный 0 и перед решением задачи программа запросит заново вес этого ребра.

В изображении графа направления дуг указаны прямыми, выходящими из одной вершины и указывающими на другую. Причем вершина, из которой выходит прямая, является началом дуги. Цифрами указаны веса дуг.

3.4.3 Задание начальной и конечной вершины.

Для того чтобы найти кратчайший путь, надо указать из какой вершины, в какую надо попасть. То есть ввести начальную и конечную вершину. Для того чтобы пометить начальную вершину, надо нажать на кнопку  и затем щелкнуть по вершине, которая должна быть начальной. Для того чтобы пометить вершину как конечную, надо нажать на кнопку  и повторить все, что приведено выше для начальной вершины.

3.4.4 Решение задачи

После этого можно в пункте меню **Задачи** выбрать подпункт **Решить** или нажать F9. После этого рядом с главным меню появится панель решения, с помощью которой можно будет вывести соответствующий путь. Цвет, которым будет отображаться найденное решение, можно установить на форме настройки, на закладке **Ребра**.

Здесь линиями выбранного вами цвета будет показан найденный путь. В окошке на панели решения задач будет стоять цифра, обозначающая длину нарисованного пути.

3.5 Дополнительные возможности редактирования.

3.5.1 Удаление дуг.

Для того чтобы удалить дугу надо перейти на матрицу графа и удалить значение, стоящее на пересечении имен вершин.

3.5.2 Удаление вершин

Для удаления неправильно введенной вершины требуется щелкнуть по ней правой кнопкой мыши. Появится контекстное меню (рисунок 45).

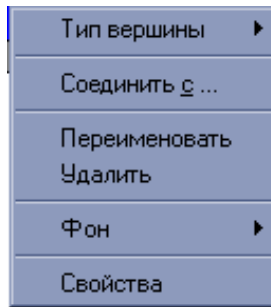


Рисунок 45

Если нажать на слово **Удалить**, то вершина будет удалена (вместе с вершиной удаляются все дуги, которые связаны с этой вершиной).

3.5.3 Удаление всего графа

Для того, что удалить весь граф и вернуть программу в исходное состояние можно воспользоваться меню **Файл** → **Новый**, или нажать на кнопку с листом, находящуюся на панели стандартных инструментов.

3.5.4 Сохранение в файл и открытие из файла

Можно сохранить граф в файл. Для этого надо нажать в строке меню **Файл** → **Сохранить**. На экране появится диалог сохранения (рисунок 46).

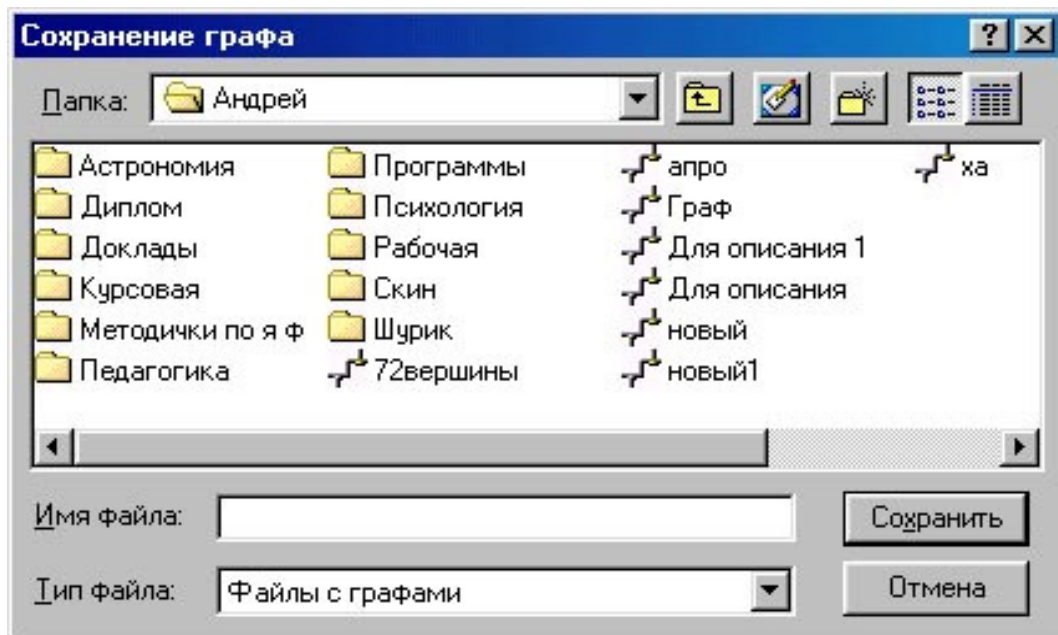


Рисунок 46

В разделе **Имя файла** вводится то имя, под которым нужно сохранить граф, после чего надо нажать кнопку сохранить. Программа автоматически присваивает сохраняемым файлам расширение *.grf. Для открытия графа нуж-

но нажать **Файл** → **Открыть**, появляется диалог открытия файла, выбрать в нем нужный и, нажав кнопку **Открыть**, на экране можно увидеть сохраненный граф.

3.6 Результаты решения задач

Поскольку изначально графы разрабатывались как способ построения различных моделей, с их помощью можно решать различные прикладные задачи. В частности, это может быть нахождение кратчайшего расстояния на местности, решение некоторых сетевых проблем (использование в Internet) и др.

В качестве демонстрации возможностей работы программы, приведем примеры решения некоторых задач

3.6.1 Задача нахождения кратчайшего расстояния в графе

Задача №1. Эта задача демонстрирует решение для достаточно простого графа, для которого можно найти минимальное расстояние аналитически, используя алгоритм Дейкстры. Она равна для данного графа, состоящего из 5 вершин и 9 дуг, 48, 85 ед. пути (рисунок 47).

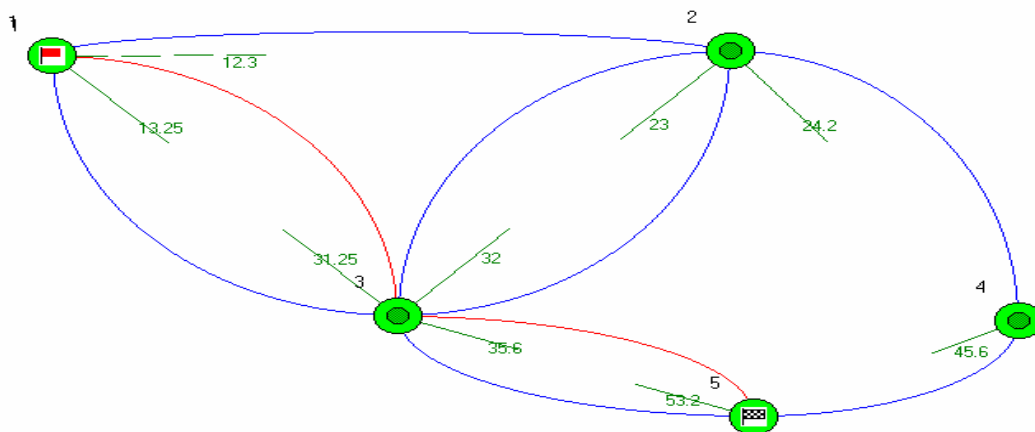


Рисунок 47

Задача №2. Эта задача демонстрирует решение более сложного графа, для которого можно найти максимальную пропускную способность аналитически, посидев немного с калькулятором. Этот граф имеет 20 вершин. Минимальным

Литература

1. "Комбинаторный анализ: задачи и упражнения" под редакцией К. А. Рыбникова М. Наука 1982г.
2. Кофман А. "Введение в прикладную комбинаторику" М. Наука 1975г.
3. Липский В. "Комбинаторика для программистов" М. Мир, 1988г.
4. Окулов С. М. "Конспекты занятий по информатике, алгоритмы на графах" Киров, 1996г.
5. Пападимитриу Х. Стайглиц К. "Комбинаторная оптимизация: алгоритмы и сложность" М. Мир, 1985г.
6. Рейнгольд Э. Нивергельт Ю. Део Н. "Комбинаторные алгоритмы: теория и практика" М. Мир, 1980г.
7. Харари Ф. "Теория графов" М. Мир, 1973г.
8. Яблонский С. В. "Введение в дискретную математику" М. Наука 1979г.
9. П. Дарахвелидзе, Е. Марков, Delphi 4 СПб. 1999г.
10. Кушнеренко В. Delphi "Тонкости программирования" М. "Познавательная книга плюс" 2000г.
11. Фаронов В. В. " Delphi 3 учебный курс" М. Нолидж 1998г.

Отпечатано с готового оригинал-макета
в типографии «Труд». г. Орел, ул. Ленина, 1.

Заказ № 2583 Тираж 250 экз.