Московский авиационный институт
(Национальный исследовательский университет)
факультет “Информационные технологии и прикладная математика”
кафедра “Математическая кибернетика”

КУРСОВАЯ РАБОТА

по курсу «Дискретная математика»
2-й семестр

Тема: Теория графов, алгебраические структуры, теория алгоритмов. Реализация алгоритма обхода в ширину для построения BFS дерева, поиска расстояний от заданной вершины, компонент связности

Студент: Дегтярев М.Э.

Группа: М8О-112Б

Руководитель: Алексеев Н.С.

Оценка: _____

Дата: _____

Москва
2018

Введение

В настоящем отчете по курсовой работе приведены результаты, полученные в рамках изучения курса “Дискретная математика” во втором учебном семестре.

В части I приведены решения типовых задач по теории графов, а также представлена реализация алгоритма построения BFS дерева, вычисления расстояний от заданной вершины до всех остальных вершин, нахождения компонент связности в неориентированном графе на основе поиска в ширину. Текст программы и тестовые примеры вынесены в Приложение.

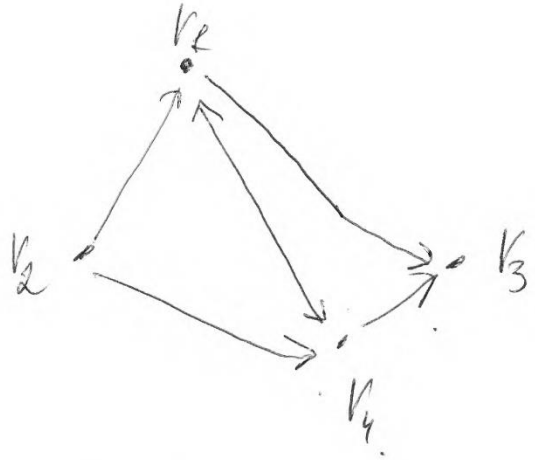
В части II приведены решения типовых задач по темам “Алгебраические структуры” и “Теория алгоритмов”.

Содержание работы

Часть первая – теория графов

Задача №7

	1	2	3	4
1	0	0	1	1
2	1	0	0	1
3	0	0	0	0
4	1	0	1	0



Найти:

а) Маке. внутр. уст. множества.

б) Мин. внешн. уст. мн-ва

в) Ядро

а) Найдем Маке. внутр уст. множества методом МАГУ

Имеем формулу.

$$(\bar{Y}_1 \vee \bar{Y}_3) \& (\bar{Y}_1 \vee \bar{Y}_4) \& (\bar{Y}_2 \bar{Y}_1) \& (\bar{Y}_2 \vee \bar{Y}_4) \& (\bar{Y}_4 \vee \bar{Y}_1) \& (\bar{Y}_4 \vee \bar{Y}_3) = (\bar{Y}_1 \vee (\bar{Y}_3 \& \bar{Y}_4 \& Y_2)) \& (\bar{Y}_4 \vee (\bar{Y}_2 \& \bar{Y}_3))$$

$$= \bar{Y}_1 \& \bar{Y}_4 \vee \bar{Y}_2 \& \bar{Y}_3 \& \bar{Y}_4 \vee \bar{Y}_1 \bar{Y}_2 \bar{Y}_3 \Rightarrow$$

$\{v_2, v_3\}$ $\{v_1\}$ $\{v_4\}$

б) $(\gamma_1 \vee \gamma_3 \vee \gamma_4)(\gamma_2 \vee \gamma_1 \vee \gamma_4)(\gamma_4 \vee \gamma_2 \vee \gamma_3) \equiv (\gamma_1\gamma_2 \vee \gamma_1\gamma_1 \vee \gamma_1\gamma_4 \vee \gamma_3\gamma_2 \vee \gamma_3\gamma_1 \vee \gamma_3\gamma_4 \vee \gamma_4\gamma_2 \vee \gamma_4\gamma_1 \vee \gamma_4\gamma_4) \& (\gamma_4 \vee \gamma_1 \vee \gamma_3) \equiv (\gamma_1 \vee \gamma_4 \vee \gamma_3\gamma_2) \& (\gamma_4 \vee \gamma_1 \vee \gamma_3) \equiv$

$\equiv \gamma_1 \vee \gamma_2\gamma_3 \vee \gamma_4$

$\{\sigma_1\}$ $\{\sigma_2, \sigma_3\}$ $\{\sigma_4\}$

Здесь в основании и выбр и вн. уст. являются все полученные мн-ва $\Rightarrow$ они ядра.

Ответ:

а) $\Rightarrow \{\sigma_1\}, \{\sigma_2, \sigma_3\}, \{\sigma_4\}$

б)

в)

Задача №9

$$A=\begin{pmatrix}0&0&1&1\\1&0&0&1\\0&0&0&0\\1&0&1&0\end{pmatrix}$$

а) $$A^2=\begin{pmatrix}0&0&1&1\\1&0&0&1\\0&0&0&0\\1&0&1&0\end{pmatrix}\begin{pmatrix}0&0&1&1\\1&0&0&1\\0&0&0&0\\1&0&1&0\end{pmatrix}=\begin{pmatrix}1&0&1&0\\1&0&?&1\\0&0&0&0\\0&0&1&1\end{pmatrix}$$

На главной диагонали имеются ненулевые элементы => контуры существуют

б) Найдем матрицу односторонней связности по формуле $T=E\vee A\vee(A^2)\vee(A^3)$

1) $$A^2=\begin{pmatrix}1&0&1&0\\1&0&1&1\\0&0&0&0\\0&0&1&1\end{pmatrix}$$

2) $$A^3=\begin{pmatrix}0&0&1&1\\1&0&1&1\\0&0&0&0\\1&0&1&0\end{pmatrix}$$

3) $$T=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\vee\begin{pmatrix}0&0&1&1\\1&0&0&1\\0&0&0&0\\1&0&1&0\end{pmatrix}\vee\begin{pmatrix}1&0&1&0\\1&0&1&1\\0&0&0&0\\0&0&1&1\end{pmatrix}\vee\begin{pmatrix}0&0&1&1\\1&0&1&1\\0&0&0&0\\1&0&1&0\end{pmatrix}=$$

$$=\underline{\begin{pmatrix}1&0&1&1\\1&1&1&1\\0&0&1&0\\1&0&1&1\end{pmatrix}}$$

в) По формуле $S=T\,\&\,T^T$ найдем матрицу сильной связности

$$S=\begin{pmatrix}1&0&1&1\\1&1&1&1\\0&0&1&0\\1&0&1&1\end{pmatrix}\,\&\,\begin{pmatrix}1&1&0&1\\0&1&0&0\\1&1&1&1\\1&1&0&1\end{pmatrix}=\underline{\begin{pmatrix}1&0&0&1\\0&1&0&0\\0&0&1&0\\1&0&0&1\end{pmatrix}}$$

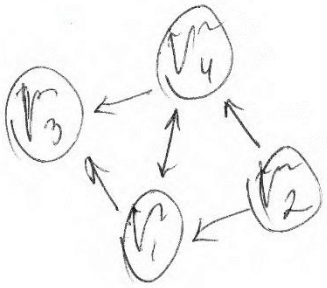
$V_1 = \{v_1, v_4\}$

$V_2 = \{v_2\}$

$V_3 = \{v_3\}$

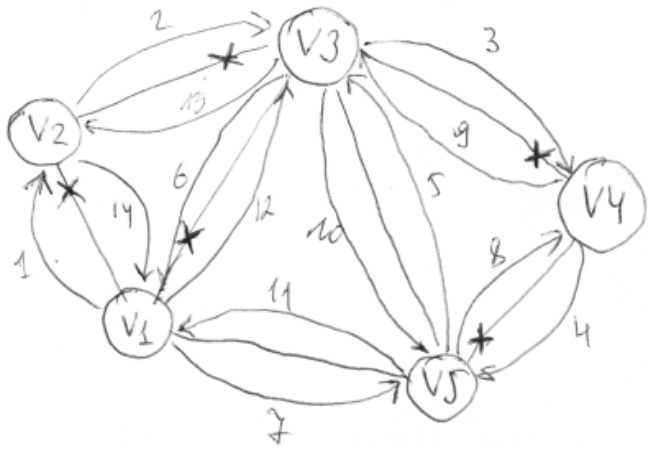
сильная св-ть графа.

д)



Построим изобр-е графа.

Задача №10



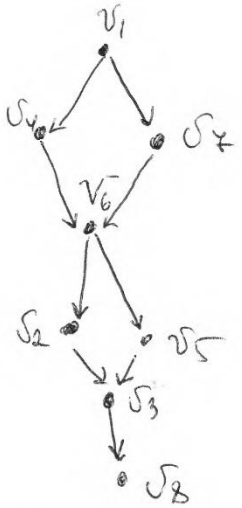
Воспользуемся Алгоритмом Терри.

Маршрут обхода:

$V_1 V_2 V_3 V_4 V_5 V_3 V_1 V_5 V_4 V_3 V_5 V_1 V_3 V_2 V_1$

Задача №11

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1	0	0	0	1	0	0	1	0
v_2	1	0	1	0	1	0	1	0
v_3	0	1	0	1	0	1	0	1
v_4	1	0	0	0	0	1	1	0
v_5	1	0	1	1	0	0	1	0
v_6	0	1	0	1	1	0	1	0
v_7	0	0	0	1	0	1	0	0
v_8	1	1	1	1	0	0	0	0



Пометим вершину v_1 индексом 0.

$v_1 \in W_0(v_1)$

$k=0$, $FW_0 = \{v_1\}$

$k=1$, $FW_1 = \{v_4, v_7\}$

$k=2$, $FW_2 = \{v_6\}$

$k=3$, $FW_3 = \{v_2, v_5\}$

$k=4$, $FW_4 = \{v_3\}$

$k=5$, $FW_5 = \{v_8\}$

Мин. пути

1) $v_1 v_4 v_6 v_2 v_3 v_8$
2) $v_1 v_4 v_6 v_5 v_3 v_8$
3) $v_1 v_7 v_6 v_2 v_3 v_8$
4) $v_1 v_7 v_6 v_5 v_3 v_8$

Длины всех путей минимальной и равной между собой $\Rightarrow$ ответом являются 4 пути

Задача №12

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1	∞	5	2	7	∞	∞	∞	∞
v_2	3	∞	2	3	∞	∞	∞	∞
v_3	∞	2	∞	∞	3	∞	∞	∞
v_4	5	∞	∞	∞	1	4	∞	9
v_5	4	∞	∞	1	∞	∞	2	∞
v_6	6	∞	∞	∞	∞	∞	4	5
v_7	∞	∞	∞	∞	∞	4	∞	9
v_8	8	∞	∞	∞	∞	∞	5	∞

$\lambda_i^{(0)}$	$\lambda_i^{(1)}$	$\lambda_i^{(2)}$	$\lambda_i^{(3)}$	$\lambda_i^{(4)}$	$\lambda_i^{(5)}$
(0)	0	0	0	0	0
∞	5	4	4	4	4
∞	2	2	2	2	2
∞	7	7	6	6	6
∞	∞	(5)	5	5	5
∞	∞	11	11	10	10
∞	∞	∞	7	7	7
∞	∞	16	16	15	15

1) Составим таблицу итераций.

2) Длина min путей из v_1 во все остальные вершины определена в последнем столбце таблицы.

3)

① $v_1 - v_2$: $v_1 v_3 v_2 = 4$

② $v_1 - v_3$: $v_1 v_3 = 2$

③ $v_1 - v_4$: $v_1 v_3 v_2 v_4 = 6$

④ $v_1 - v_5$: $v_1 v_3 v_5 = 5$

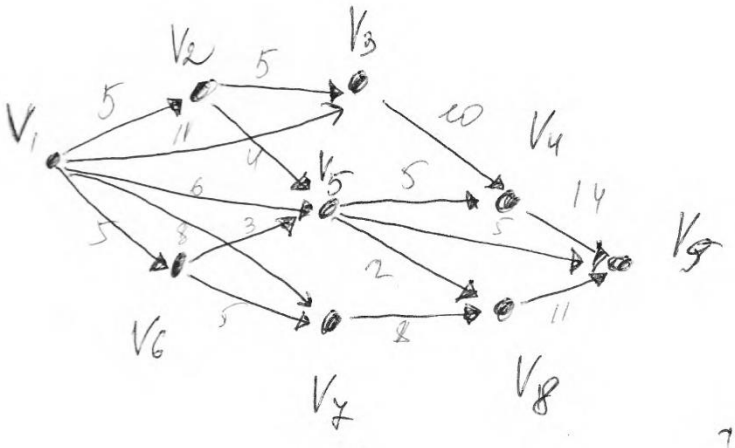
⑤ $v_1 - v_6$: $v_1 v_3 v_2 v_4 v_6 = 10$

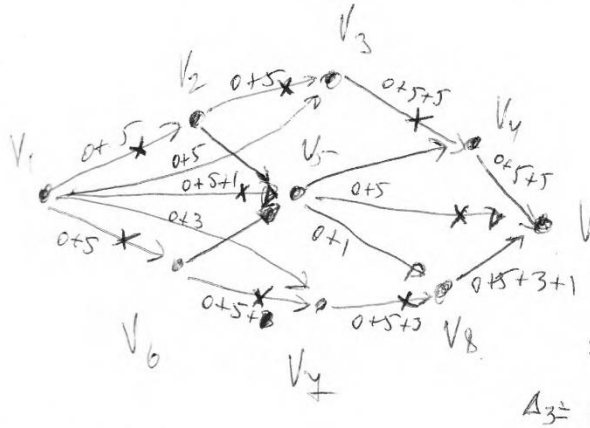
⑥ $v_1 - v_7$: $v_1 v_3 v_2 v_7 = 7$.

⑦ $v_1 - v_8$: $v_1 v_3 v_2 v_4 v_8 = 15$.

Задача №13

a	b	c	d	e	f	g
5	5	5	10	4	8	2





I. Полный поток

1. $V_1 V_2 V_3 V_4 V_9$

$\Delta_1 = \min(5-0, 5-0, 10-0, 14-0) = \boxed{5}$

2. $V_1 V_6 V_7 V_8 V_9$

$\Delta_2 = (5-0, 5-0, 8-0, 11-0) = \boxed{5}$

3. $V_1 V_5 V_9$

$\Delta_3 = (8-0, 5-0) = \boxed{5}$

4. $V_1 V_3 V_4 V_9$

$\Delta_4 = \min(14-0, 10-5, 14-5) = \boxed{5}$

5. $V_1 V_7 V_8 V_9$

$\Delta_5 = \min(8-0, 8-5, 11-5) = \boxed{3}$

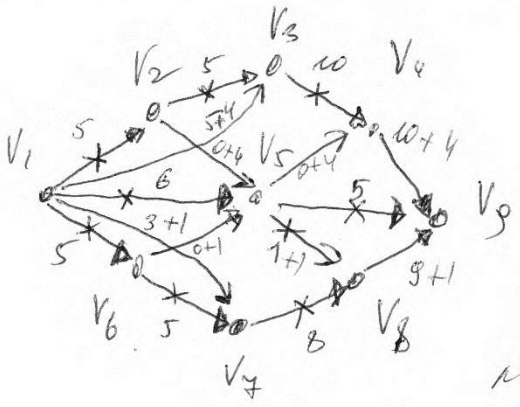
6. $V_1 V_5 V_8 V_9$

$\Delta_6 = \min(6-5, 2-0, 11-8) = \boxed{1}$

$C_{пол} = 10 + 5 + 9 = \underline{24} = 5 + 5 + 6 + 8 = 14 + 10 = 24$

II. Максимального потока поиск. Найдем ув. цепи

7. $V_1 V_3 V_2 V_7 V_4 V_9$ $\Rightarrow \Delta_7 = \min(11-5, 5, 4-0, 5-0, 14-10) = 4$



8. $V_1 V_7 V_6 V_5 V_8 V_9$

$1_8 = \min(8-3, 5, 3-0, 2-1, 11) = 1$

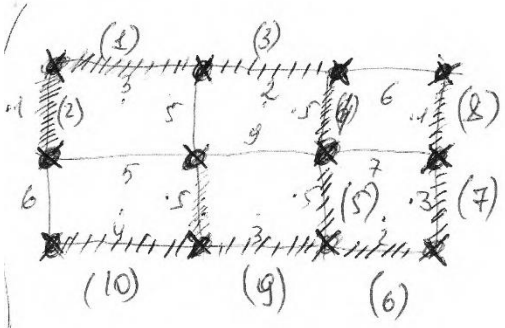
Величина максимального потока.

$$\varphi_{max} = 14+5+10 = \underline{29} = \underbrace{5+9+6+4}_{10}+5 = 29$$

Ответ: $\varphi_{пол} = 24$

$\varphi_{max} = 29$

Задача №1



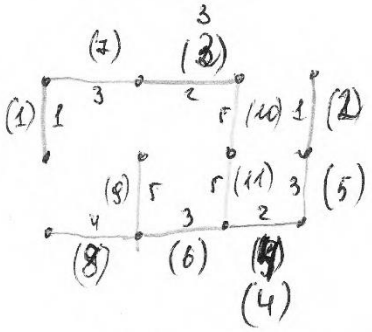
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
6	1	3	5	4	3	9	2	6	7

x_{11}	x_{12}	x_{13}
2	3	1

$1+3+5+5+4+2+5+3+2+3+1=34$

1) Воспользуемся алгоритмом Прима (выбором ребра цвета

2) Воспользуемся алгоритмом Краскала

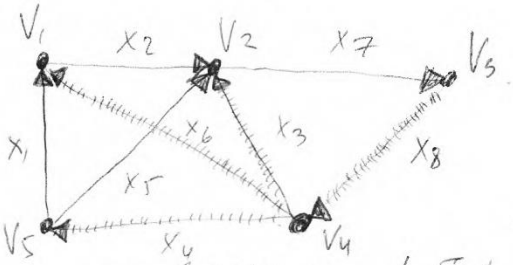


$1+1+2+2+3+3+3+4+5+5+\ldots =$

$= 34$

Циклов нет. Результат совпадает с результатом, полученным при помощи алгоритма Прима.

Задача №2



1. Выделим основное дерево (каркас)
2. Выбираем произвольно направления ребер.
3. Уравнений системы Кирхгофа для напряжений

а) Составим унимодальную матрицу.

$x_1 : x_1 x_6 x_4$
$x_2 : x_2 x_3 x_6$
$x_5 : x_5 x_3 x_4$
$x_7 : x_7 x_8 x_3$

	ξ_1				ξ_2			
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	1	0	0	1	0	-1	0	0
x_2	0	+1	-1	0	0	1	0	0
x_5	0	0	-1	1	1	0	0	0
x_7	0	0	1	0	0	0	1	1

Составим СЛАУ

$CU = 0$

$$\begin{cases} \xi_1 + U_4 - U_6 = 0 \\ U_2 - U_3 + U_6 = 0 \\ -U_3 + U_4 + \xi_2 = 0 \\ U_3 + U_7 + U_1 = 0 \end{cases}$$

Уравнение Кирхгофа для тока 3

Матрица инцидентности

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
V_1	1	-1	0	0	0	1	0	0
V_2	0	1	1	0	1	0	-1	0
V_3	0	0	0	0	0	0	1	-1
V_4	0	0	-1	-1	0	-1	0	1
V_5	-1	0	0	1	-1	0	0	1

Составляем СЛАУ

$$\begin{cases} I_1 - I_2 + I_6 \\ I_2 + I_3 + I_5 - I_7 = 0 \\ I_7 - I_8 = 0 \\ ~~-I_3 + I_4 - I_6 + I_8 = 0~~ \\ -I_1 + I_4 - I_5 = 0 \end{cases}$$

По закону Ома $U_i = I_i R_i$

$$\begin{cases} I_1 - I_2 + I_6 = 0 \\ I_2 + I_3 + I_5 - I_7 = 0 \\ I_7 - I_8 = 0 \\ -I_1 + I_4 - I_5 = 0 \\ \mathcal{E}_1 + I_4 R_4 - I_6 R_6 = 0 \\ -I_3 R_3 + I_4 R_4 + \mathcal{E}_2 = 0 \\ I_3 R_3 + I_7 R_7 + I_8 R_8 = 0 \\ I_2 R_2 - I_3 R_3 + I_6 R_6 = 0 \end{cases}$$

ответ

Часть первая – задача №8. Реализация программы

Алгоритм обхода графа в ширину

Формулировка задачи:

Построение BFS-дерева, вычисление расстояний от заданной вершины до всех остальных вершин, нахождение компонент связности на основе поиска в ширину

Основные понятия:

Обход в ширину (BFS – Breadht-first search) – один из простейших алгоритмов обхода графа, являющийся основной для алгоритмов, решающих поставленные задачи в программе.

Расстояние – число, являющееся суммой чисел длин рёбер в нагруженном графе от исходной вершины до результирующей вершины. В нашем случае расстояние от вершины V1 до V2 является кратчайшим

BFS-дерево – Изначально оно состоит из одного корня (в нашем случае, заданной в программе нами вершины). Когда мы добавляем непосещенную вершину в очередь, то добавляем ее и ребро, по которому мы до нее дошли, в дерево. Поскольку каждая вершина может быть посещена не более одного раза, она имеет не более одного родителя. После окончания работы алгоритма для каждой достижимой из корня вершины T, путь в дереве поиска в ширину соответствует кратчайшему пути от корня S до T в графе G.

Компоненты связности – граф G является связным, если для двух любых вершин этого графа существует путь между ними. Если же граф G не является связным, то его можно разбить на непересекающиеся связные подмножества, которые называются компонентами связности.

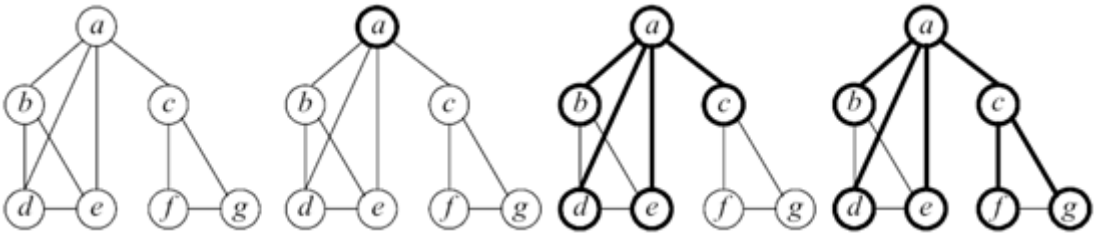
Описание алгоритма обхода в ширину

При поиске в ширину, после посещения первой вершины, посещаются все соседние с ней вершины. Потом посещаются все вершины, находящиеся на расстоянии двух ребер от начальной. При каждом новом шаге посещаются вершины, расстояние от которых до начальной на единицу больше предыдущего. Чтобы предотвратить повторное посещение вершин, необходимо вести список посещенных вершин. Для хранения временных данных, необходимых для работы алгоритма, используется очередь – упорядоченная последовательность элементов, в которой новые элементы добавляются в конец, а старые удаляются из начала.

Реализация алгоритма обхода в ширину

Шаг 1. Все вершины заданного графа помечаются непосещёнными. В очередь заносится вершина, с которой начнётся обход.

Шаг 2. Происходит помечание первой вершины как посещённая, все смежные вершины помечаются как обнаруженные. Обнаруженные вершины добавляются в очередь. После этого из очереди берётся следующая вершина, выполняются аналогичные действия.



Шаг 3. Пока очередь не будет пустой, происходит выполнение шага 2.

Реализация алгоритма поиска расстояний от заданной вершины

Шаг 1. Заводим массив расстояний до вершин. Расстояния изначально делаем неизмеримо большие.

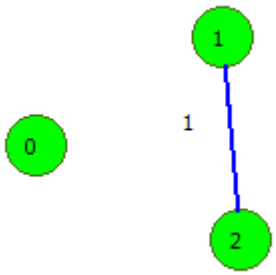
Шаг 2. Изменив основные условия будем добавлять в очередь те вершины, расстояние до которых можно уменьшить. В текущей вершине проверяем расстояние, при возможности уменьшаем его.

Шаг 3. Пока очередь не будет пустой, выполняем шаг 2.

Реализация алгоритма поиска компонент связности

Шаг 1. Заводим массив вершин, счётчик. Производим обнуление.

Шаг 2. На k-й , происходит обход графа в ширину из k-й вершины (при условии того, что она ещё не была посещена). В массив вершин на k-м индексе указываем значение счётчика. Инкрементируем счётчик. Например, задан граф

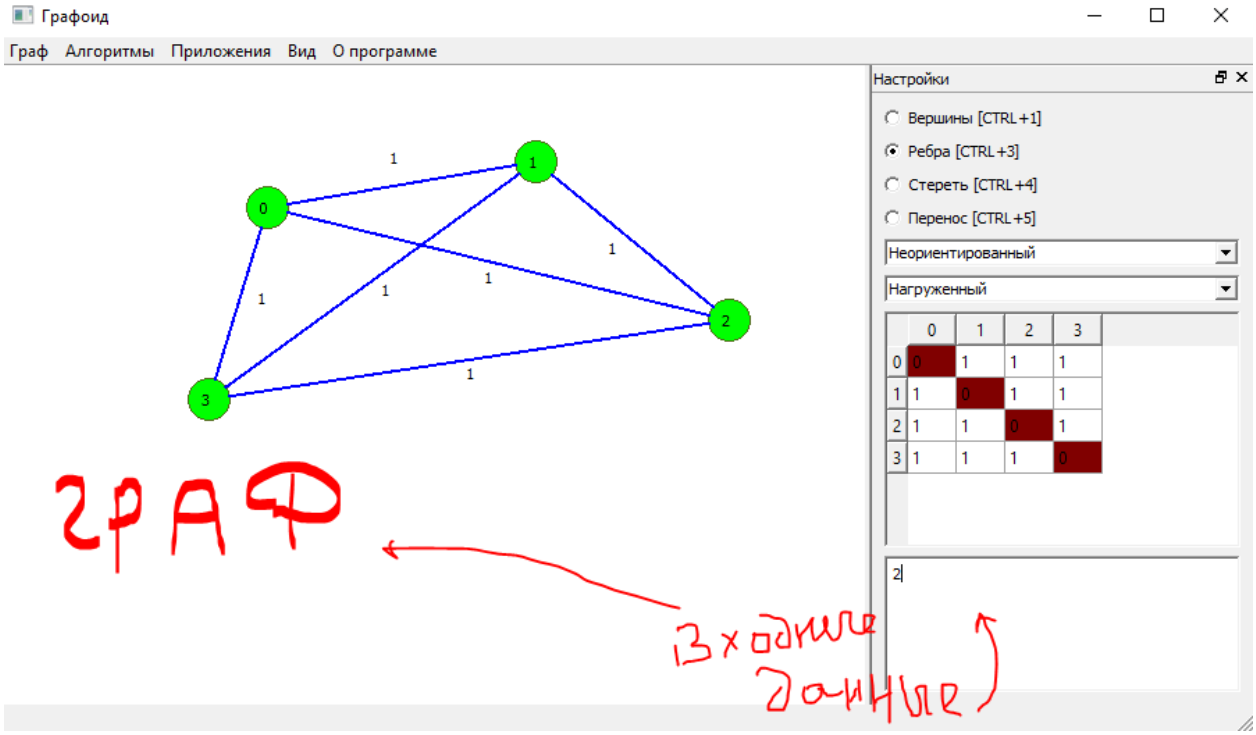


После его обработки массив вершин будет выглядеть вот так
Versh[0] = 1
Versh[1] = 2
Versh[2] = 2

В обычном представлении для пользователя: {0}, {1,2}

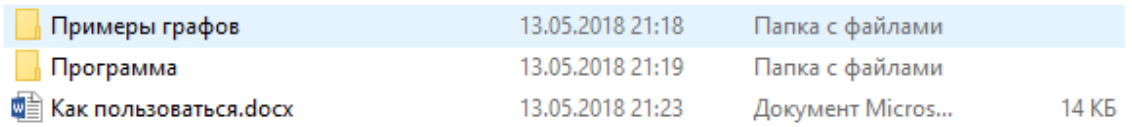
Описание программы

Программа представляет собой один исполняемый файл формата .exe, который в качестве входных данных принимает информацию из входного потока, генерирующимся программой ГРАФОИД. Она выбирается в качестве исполняемого файла в графойде, принимает на вход неориентированный граф, а также вершину, из которой необходимо начать обход.



Инструкция по использованию программы

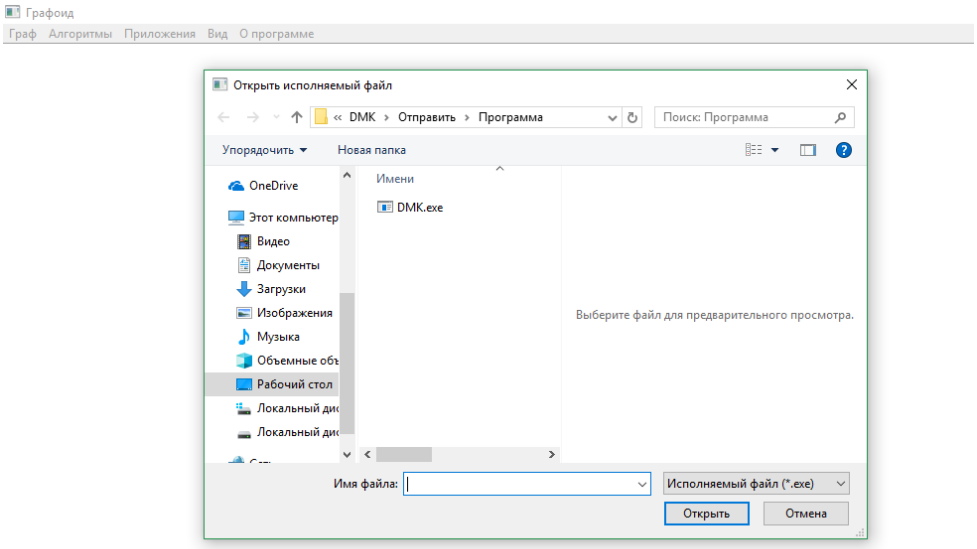
1. Содержание папки с программой
    a. Основная папка с решением содержит два каталога:
        - Программа – здесь находится исполняемый файл, необходимый для открытия в Графойде (По умолчанию, DMK.exe)
        - Примеры графов – данный каталог содержит основные примеры графов, которые можно задавать программе



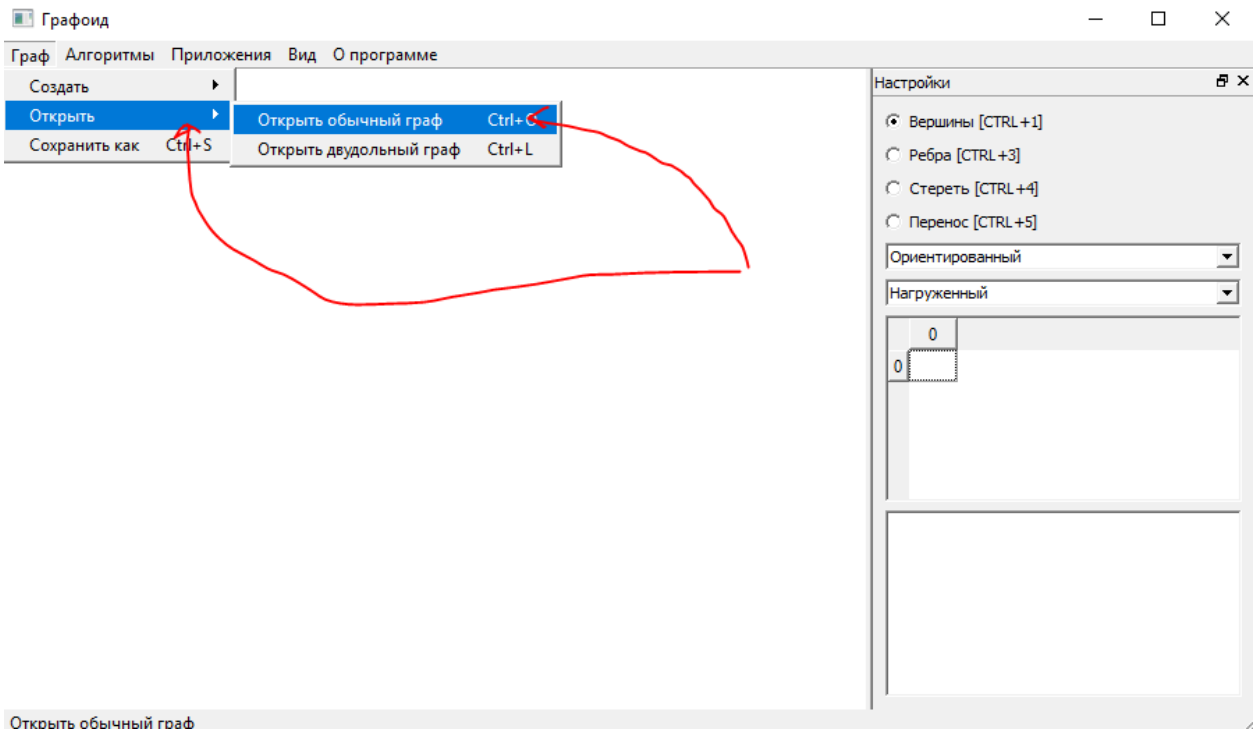
2. Использование программы

Шаг 1. Откройте программу «Графойд»

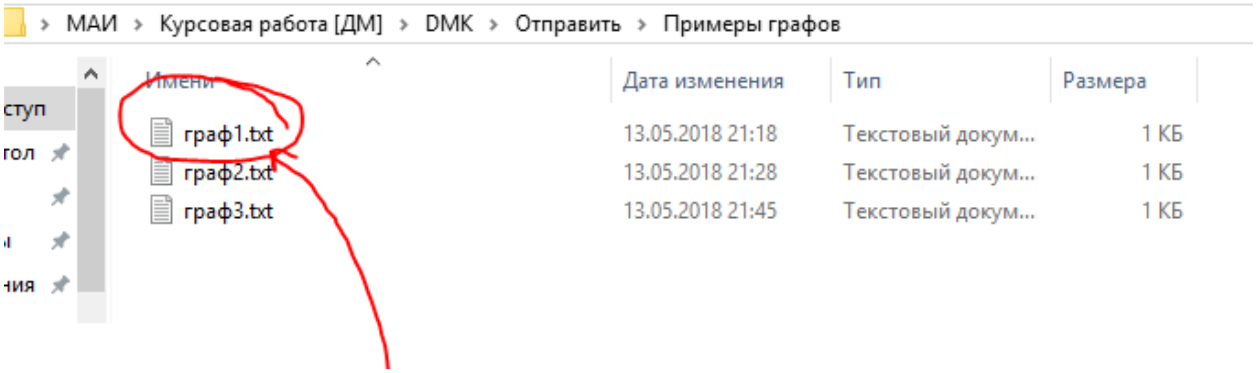
Шаг 2. В качестве исполняемого файла выберите **DMK.exe**, находящийся в папке «Программа»



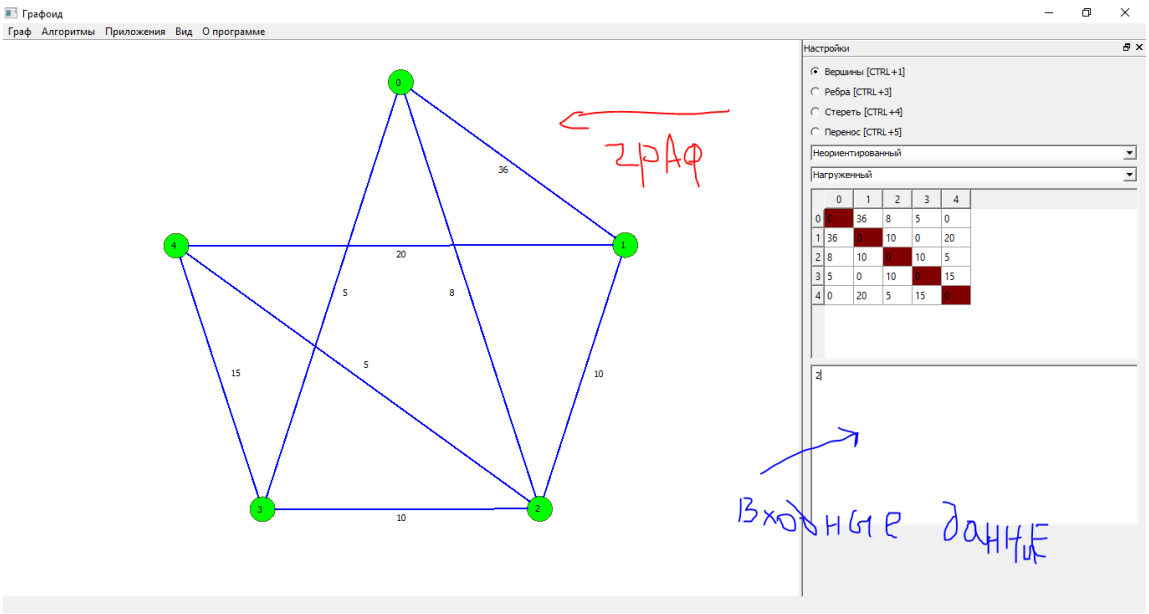
Шаг 3. Откройте пример графа для программы. Для этого левой кнопкой мыши кликните по пункту меню: ГРАФ->ОТКРЫТЬ->ОТКРЫТЬ ОБЫЧНЫЙ ГРАФ



Появится окно выбора файла. Выберите каталог «Примеры графов», откройте текстовый файл «граф1.txt»



После двойного клика будет открыт следующий граф, похожий на звезду:

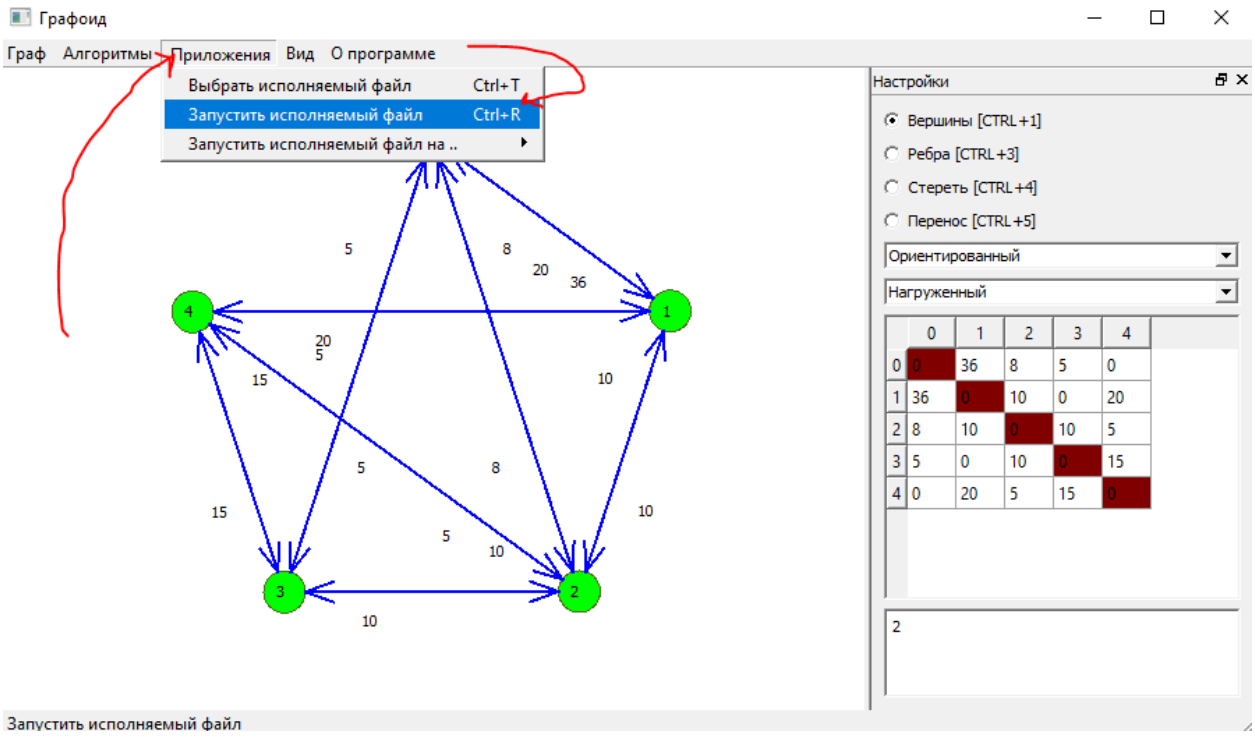


Здесь, в текстовом поле вы увидите цифру «2». **Именно в нём необходимо указать исходную вершину, от которой мы будем искать расстояния до всех остальных вершин. Также она будет являться корнем BFS дерева.**

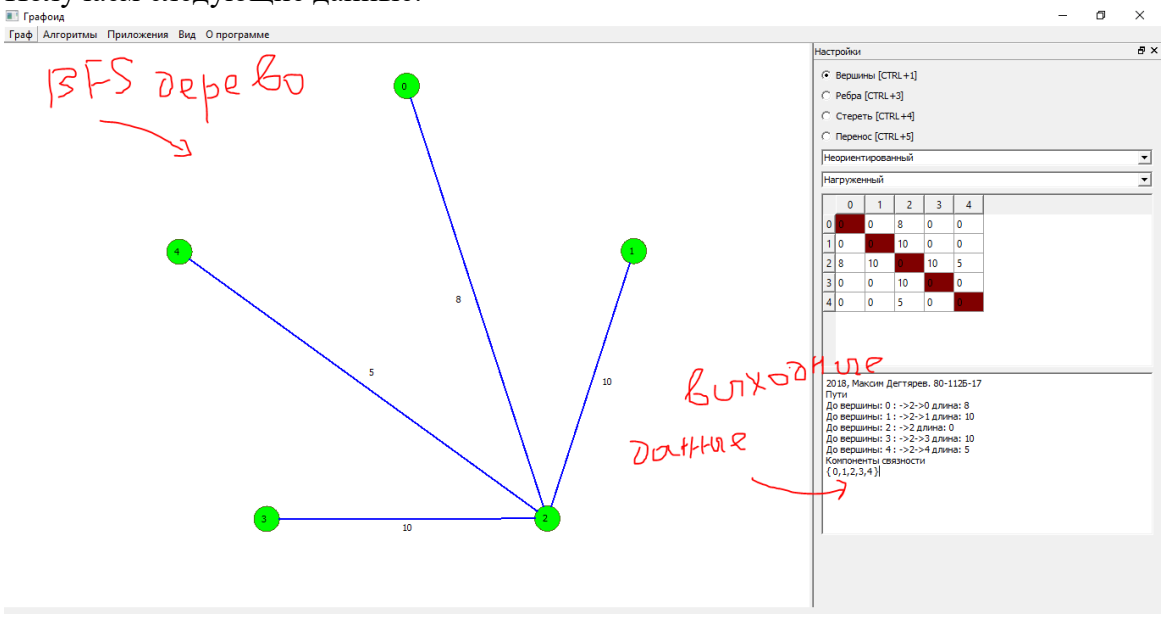
На пустой области строится неориентированный граф

Шаг 4.

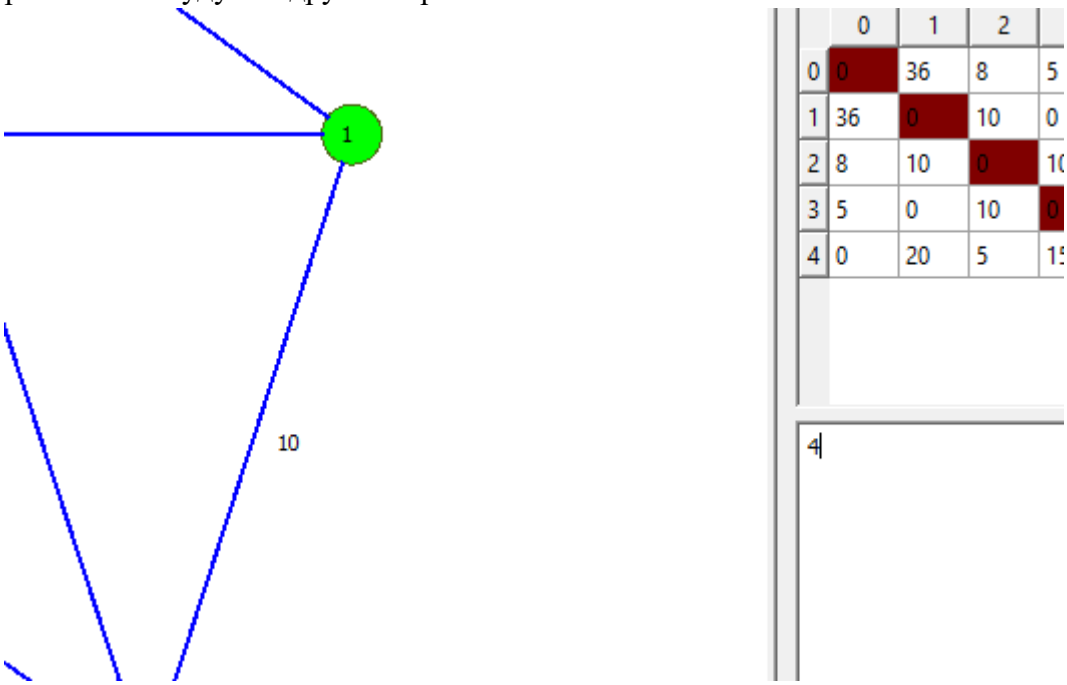
Приступим к запуску программы. Выбираем вкладку ПРИЛОЖЕНИЯ->ЗАПУСТИТЬ ИСПОЛНЯЕМЫЙ ФАЙЛ



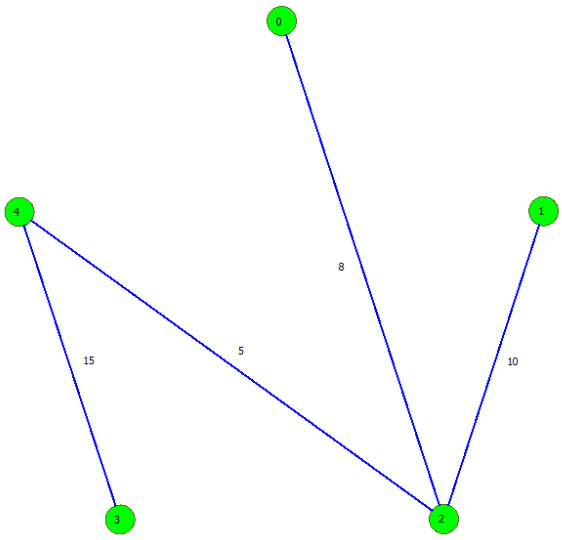
Получаем следующие данные:

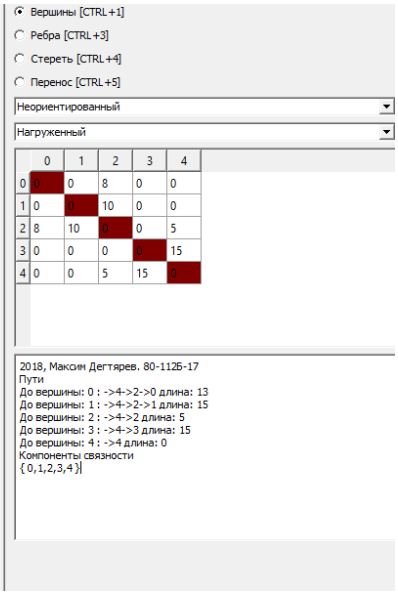


На месте рисования графа появляется BFS дерево. В текстовом поле выводится фамилия автора программы, минимальные расстояния до вершин, пути до них, компоненты связности. Можно также попробовать открыть этот же граф и посмотреть, какие расстояния будут из другой вершины:



Имеем:





Стоит заметить, что программа не удаляет вершины при построении дерева. Это обусловлено тем, что будет потеряна их нумерация, а значит пользоваться программой будет неудобно при их удалении.

Оценка вычислительной сложности алгоритма

Самый затратный по времени алгоритм программы – поиск минимальных расстояний от исходной вершины s до всех остальных вершин. Его сложность есть O(n * m).

Тестовые примеры

Пример 1.

5

0 36 8 5 0

36 0 10 0 20

8 10 0 10 5

5 0 10 0 15

0 20 5 15 0

Text:

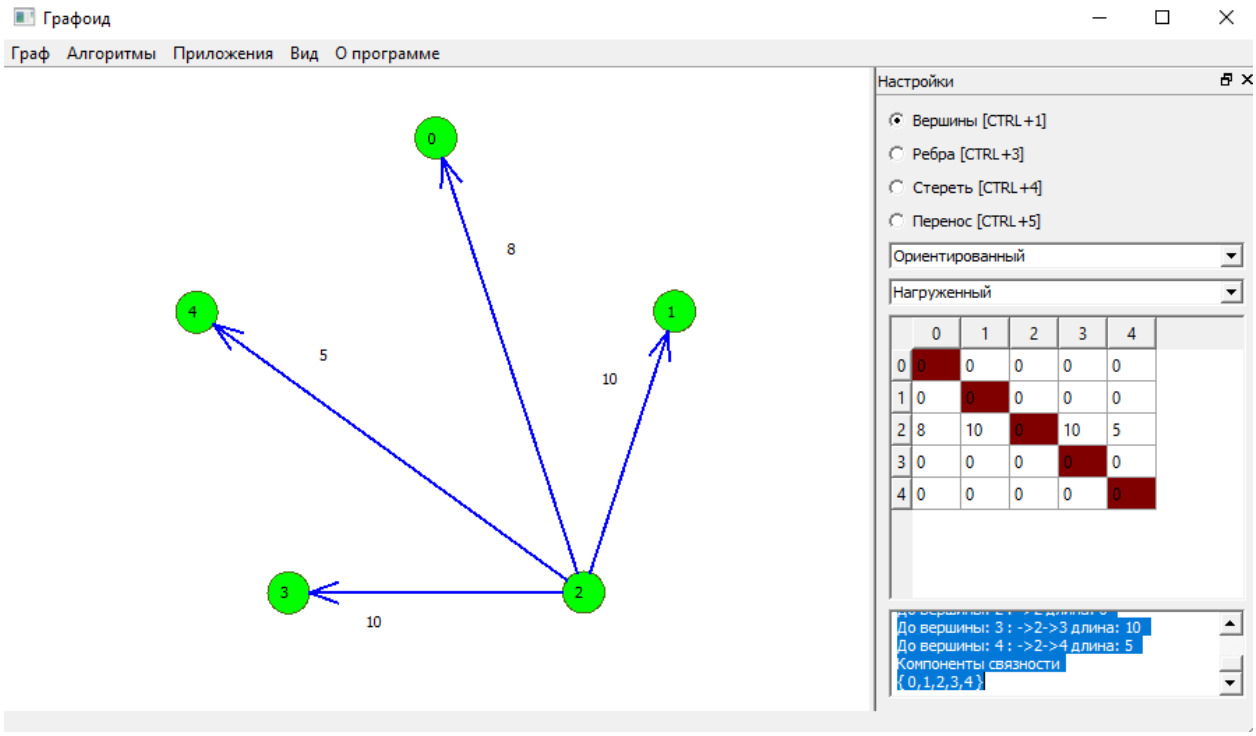
2

Решение:

Пути
До вершины: 0 : ->2->0 длина: 8
До вершины: 1 : ->2->1 длина: 10
До вершины: 2 : ->2 длина: 0

До вершины: 3 : ->2->3 длина: 10
До вершины: 4 : ->2->4 длина: 5
Компоненты связности
{ 0,1,2,3,4 }

Скриншот:



Пример 2.

8

0 0 1 0 1 0 0 1

0 0 1 0 1 0 0 1

1 1 0 1 0 0 0 0

0 0 1 0 1 0 0 0

1 1 0 1 0 1 0 0

0 0 0 0 1 0 1 0

0 0 0 0 0 1 0 1

1 1 0 0 0 0 1 0

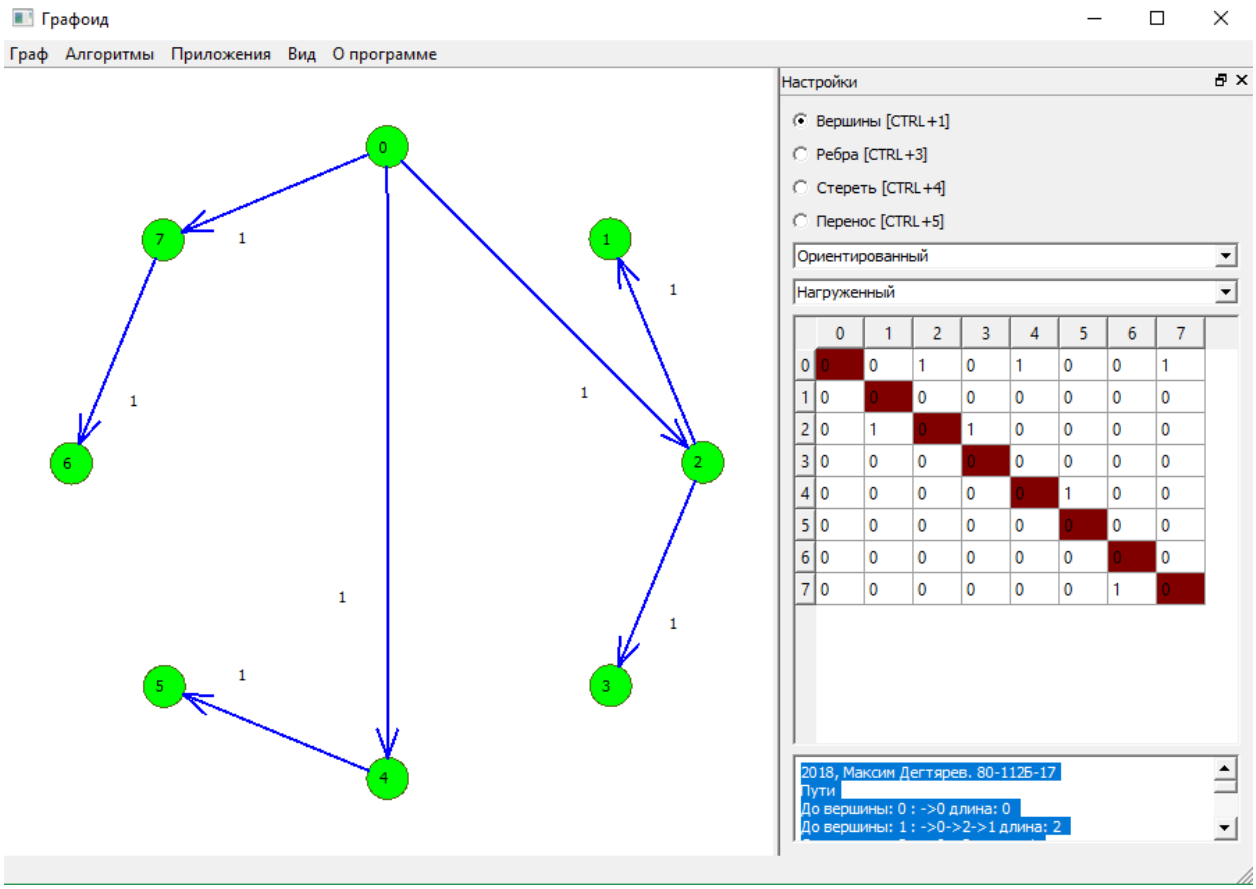
Text:

0

Решение:

Пути
До вершины: 0 : ->0 длина: 0
До вершины: 1 : ->0->2->1 длина: 2
До вершины: 2 : ->0->2 длина: 1
До вершины: 3 : ->0->2->3 длина: 2
До вершины: 4 : ->0->4 длина: 1
До вершины: 5 : ->0->4->5 длина: 2
До вершины: 6 : ->0->7->6 длина: 2
До вершины: 7 : ->0->7 длина: 1
Компоненты связности
{ 0,2,4,7,1,3,5,6 }

Скриншот



Пример 3.

7

0 360 0 0 0 0 0

360 0 0 0 0 0 0

0 0 0 640 52 0 0

0 0 640 0 44 0 0

0 0 52 44 0 0 0

0 0 0 0 0 0 15

0 0 0 0 0 15 0

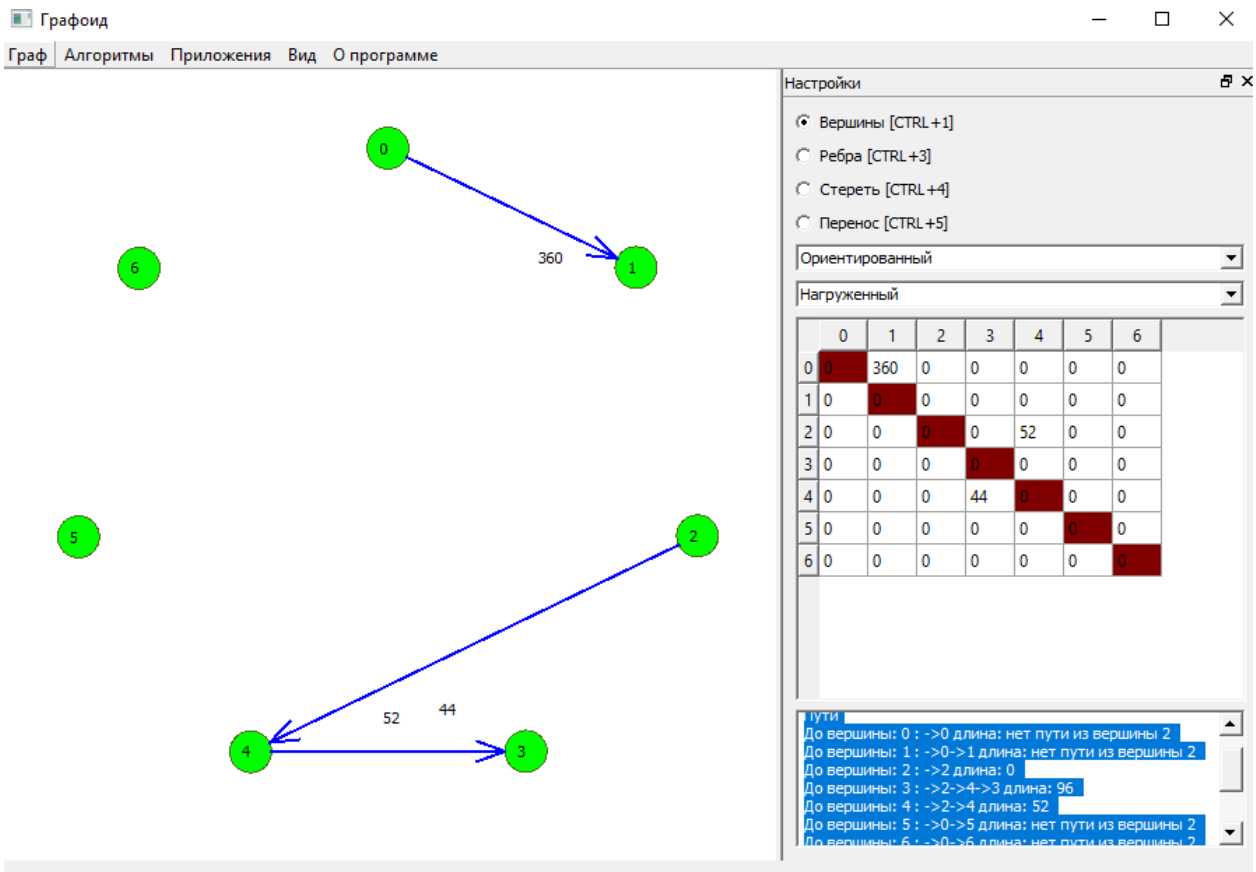
Text:

2

Решение:

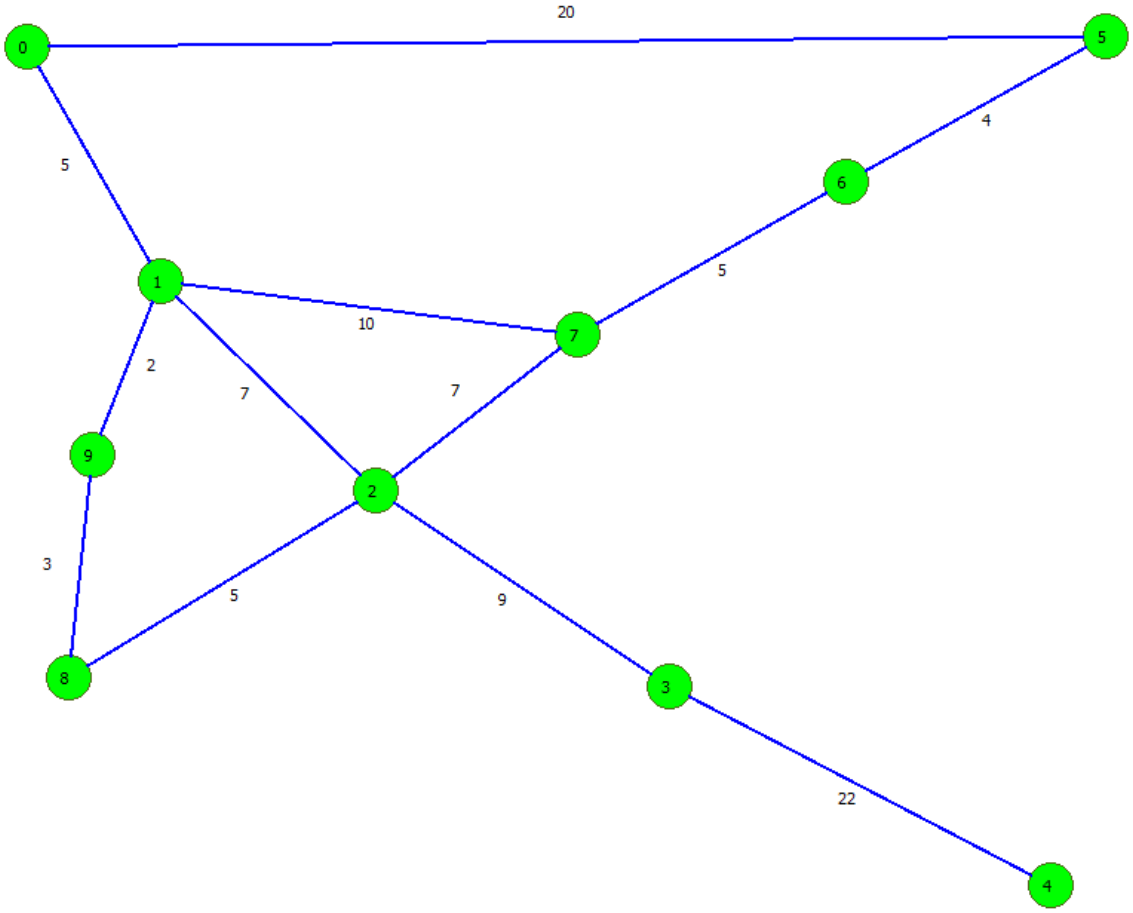
2018, Максим Дегтярев. 80-112Б-17
Пути
До вершины: 0 : ->0 длина: нет пути из вершины 2
До вершины: 1 : ->0->1 длина: нет пути из вершины 2
До вершины: 2 : ->2 длина: 0
До вершины: 3 : ->2->4->3 длина: 96
До вершины: 4 : ->2->4 длина: 52
До вершины: 5 : ->0->5 длина: нет пути из вершины 2
До вершины: 6 : ->0->6 длина: нет пути из вершины 2
Компоненты связности
{ 0,1 }
{ 2,3,4 }
{ 5,6 }

Скриншот

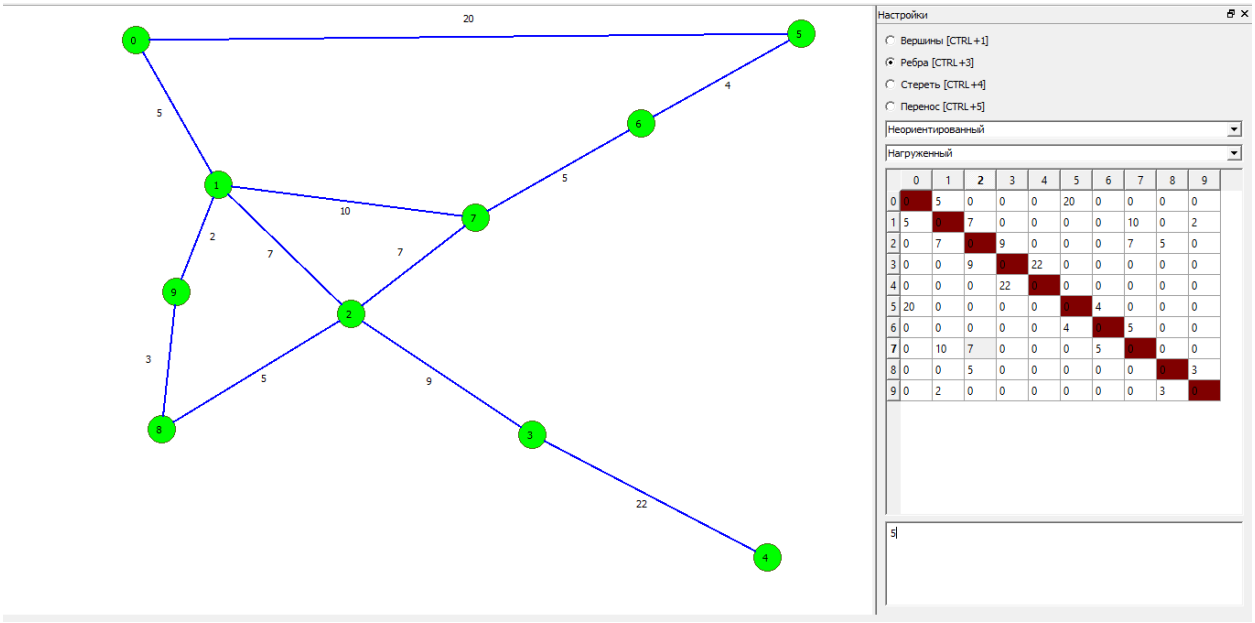


Пример прикладной задачи

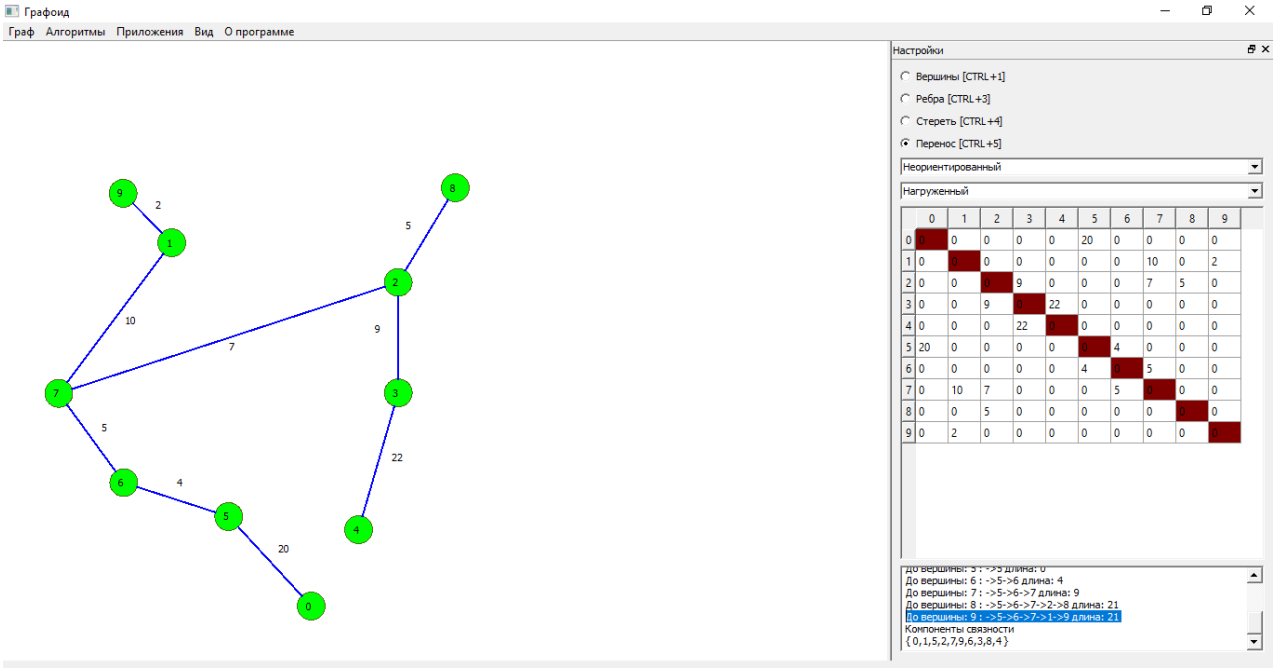
Пётр – вечно опаздывающий молодой человек, поэтому он пользуется метрополитеном. Ему необходимо добраться от станции 5 до станции 9 за минимальное количество времени. Схему нескольких участков метрополитена можно задать неориентированным нагруженным графом



Где вершины – станции метро, а длины ребёр – время, которое необходимо затратить на перемещение между станциями. Для этого Пётр может запустить написанную программу через Графойд, ввести данный граф, затем указать в качестве входных данных станцию №5



После выполнения программа не только построит карту самых быстрых путей от станции №5, но и путь до нужной ему станции №9, а также время, равное 21 минуте



Часть вторая – теория алгоритмов, алгебраические структуры

Задача №3

$f(x,y) = 2y(x+2)$

$S(x) = x+1$

$0(x) = 0$

$I_n^m(x_1 \dots x_n) = x_m (1 \le m \le n)$, $\Sigma(x_1, x_2) = x_1 + x_2$

Решение: Воспользуемся операторами примитивной рекурсии

$f(x,0) = 0$

$f(x, y+1) = 2(y+1)(x+2) = (2y+2)(x+2) =$

$= 2yx + 4y + 2x + 4 = 2y(x+2) + 2(x+2) =$

$= f(x,y) + 2(x+2)$

$\Psi(x,y,z) = z + 2(x+2) = z + 2(x+1+1) =$

$0(x) = f(x,0) = 0$ $= z + 2(S(x)+1) =$

$= z + 2S(S(x)) = z + S(S(x)) + S(S(x)) =$

$= \Sigma(z, S(S(x))) + S(S(x)) = \Sigma(\Sigma(z, S(S(x))), S(S(x)))$

Ответ: $\Sigma(\Sigma(z, S(S(x))), S(S(x)))$

Задание №4

Дано:

$[(6817)(274)(15)(678)]^{-111}$ в S_8

Найти

а) Разложение на независимые циклы:

$[(6817)(274)(15)(678)]^{-111} = [(14285)]^{-111}$

б) $-111 = -110 - 1 = 5(-22) - 1$, $[(14285)]^{-1} =$ (15824)

т.е порядок есть 5

в) Представив в виде пр-я транспозиций

в) Представив в виде пр-я транспозиций получим:

$(15)(18)(12)(14)$

г) Четное;

Задача №5

Дано

$H = \langle (1\,2), (3\,4) \rangle$ группа S_4

Найти

а) элементы из H

б) ЛСК группы S_4 по H

в) ПСК группы S_4 по H

г) Является ли H нормальным делителем?

а)

1) $(1\,2)(1\,2) = (1)(2) = e$

2) $(1\,2)(3\,4) = (1\,2)(3\,4)$

3) $(3\,4)(3\,4) = (3)(4) = e$

4) $(1\,2)(3\,4)(1\,2) = (1)(2)(3\,4)$ дальше

5) $(1\,2)(3\,4)(3\,4) = (1\,2)(3)(4) = (1\,2)$ дальше

6) $(1\,2)(3\,4)(1\,2)(3\,4) = (1)(2)(3)(4) = e$

$H = \{e\}$

б) $eH = e \cdot e = e$

в) $He = e \cdot e = e$

г) ПСК = ЛСК => Нормальный делитель

Таблица Кэли:

	e	$(1\,2)$	$(3\,4)$	$(1\,2)(3\,4)$
e	e	$(1\,2)$	$(3\,4)$	$(1\,2)(3\,4)$
$(1\,2)$	$(1\,2)$	e	$(1\,2)(3\,4)$	$(3\,4)$
$(3\,4)$	$(3\,4)$	$(1\,2)(3\,4)$	e	$(1\,2)$
$(1\,2)(3\,4)$	$(1\,2)(3\,4)$	$(3\,4)$	$(1\,2)$	e

Задача №6

Дано.

$a = 1011$

$c' = 0100101$

$c'' = 0001110$

Найти b - ?

а) Определим соотв. кодовое слово

$b = b_1 b_2 1 b_4 0 1 1$

$b_4 + 1 + 1 = 0 \Rightarrow b_4 = 0$

$b_2 + 1 + 1 + 1 = 1 \Rightarrow b_2 = 1$

$b_1 + 1 + 1 = 0 \Rightarrow b_1 = 0$

$\begin{pmatrix} 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{pmatrix}$

Значит $b = 0110011$

б) найдем ошибки

$c'M = (0\ 0\ 0) \Rightarrow$ Ошибок нет

$b = 0110011$

слово : 0101

$\begin{pmatrix} 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{pmatrix}$

в) $c''M = (1\ 1\ 1) = 7_{10} \Rightarrow$ Ошибка в 7 позиции $\Rightarrow$ слово $b = 0001111$

$a = 0111$

Заключение

В ходе выполнения курсовой работы были решены 12 задач по различным разделам курса “Дискретная математика”.

Кроме того был изучен вопрос о различных методах поиска путей, вычисления расстояний и компонент связности в графах. Была написана и отлажена программа, реализующая алгоритм обхода в ширину. Программа написана на языке программирования C++. Программа обеспечивает связь по установленному формату с системой ГРАФОИД, разработанной на кафедре 805, что дает возможность обеспечить графический интерфейс при ее использовании. Эта программа является основным результатом курсового проектирования.

Список источников

1. В.А. Таланов, В.Е. Алексеев. Графы и алгоритмы. https://www.intuit.ru/studies/courses/648/504/lecture/11474?page=2 Кофман А. Введение в прикладную комбинаторику.
2. Липский В. Комбинаторика для программистов.
3. Нефедов В.Н., Осипова В.А. Курс дискретной математики.
4. Нефедов В.Н. Дискретные задачи оптимизации. https://goo.gl/faUEEU

Приложение

Исходный код программы: Язык прораммирования C++

```
#define _CRT_SECURE_NO_WARNINGS

#include <cstdlib>
#include <iostream>
#include <fstream>
#include <string>
#include <queue>

#define MAX 500
using namespace std;

int graph[MAX][MAX];

int bfstree[MAX][MAX]; //BFS дерево

int vertex[MAX] = { 0 };
int lenght[MAX] = { 0 };
bool inq[MAX];
int p[MAX] = { 0 };
int marsh[MAX];
int R = 0;
int NUM = 0;
int N = 0;
int global[MAX] = { 0 };

string timepute = "";

/*
Открытие файла -> Запись матрицы
*/
void GetData(char *argv[])
{
	string BUFFER;
	ifstream in(argv[1]);
	in >> N;
	int i, j = 0;

	for (i = 0; i < N; i++)
	{
```

```
		for (j = 0; j < N; j++)
		{
			in >> graph[i][j];
		}
	}

	while ((BUFFER != "Text:"))
	{
		in >> BUFFER;
	}

	//Дошли до числа
	if (in.eof())
		NUM = 0;
	else
		in >> NUM;

	in.close();
}

void SendData(char *argv[], string Text)
{
	int i, j = 0;
	ofstream out;
	out.open(argv[1]);
	out.clear();
	char buffer[120];
	_itoa(N, buffer, 10);
	out << buffer << "\n";

	for (i = 0; i < N; i++) {
		for (j = 0; j < N; j++) {
			out << bfstree[i][j];
			if (j != N) out << " ";
		}
		if (i != N) out << "\n";
	}
	out << "\nText:\n";
	out << Text;
	out.close();
}

void print_way(int u) {
	if (p[u] != u) {
		print_way(p[u]);
	}
	timepute = timepute + "->" + to_string(u);
}

void init_pute(int u) {
	if (p[u] != u) {
		init_pute(p[u]);
	}
	marsh[R] = u; //Записали предка
```

```
	R++;
}

void zapolnen()
{
	if (R > 1) {
		for (int i = 0; i < R - 1; i++)
		{
			bfstree[marsh[i]][marsh[i + 1]] = graph[marsh[i]][marsh[i + 1]];
		}
	}
	else if (R == 1)
	{
		bfstree[marsh[0]][marsh[0]] = graph[marsh[0]][marsh[0]];
	}
}

string BFS(int from)
{
	string res = "";
	queue<int> Queue;
	res+= "{ ";

	int nodes[MAX]; // вершины графа
	for (int i = 0; i < N; i++)
		nodes[i] = 0; // исходно все вершины равны 0
	Queue.push(from); // помещаем в очередь первую вершину
	while (!Queue.empty())
	{ // пока очередь не пуста
		int node = Queue.front(); // извлекаем вершину
		Queue.pop();
		nodes[node] = 2; // отмечаем ее как посещенную
		global[node] = 1;
		res+= to_string(node) + ",";
		for (int j = 0; j < N; j++)
		{ // проверяем для нее все смежные вершины
			if (graph[node][j] > 0 && nodes[j] == 0)
			{ // если вершина смежная и не обнаружена
				Queue.push(j); // добавляем ее в очередь
				nodes[j] = 1; // отмечаем вершину как обнаруженную
			}
		}
	}
	res.erase(res.length() - 1, 1);
	res += " }\n";
	return res;
}

int main(int argc, char *argv[])
{
	//Первая часть задачи - BFS дерево и расстояния
	string rez = "";
	rez += "2018, Максим Дегтярев. 80-112Б-17\n";
	GetData(argv);
```

```
	for (int i = 0; i < N; i++)
	{
		lenght[i] = 1e+9;
	}

	int u = NUM;
	lenght[u] = 0;
	p[u] = u;
	queue<int> q;
	q.push(u);
	inq[u] = true;
	while (!q.empty())
	{
		int z = q.front();
		q.pop();
		inq[z] = false;
		for (int i = 0; i < N; i++)
		{
			if (graph[z][i] > 0) {
				int v = i;
				int len = graph[z][i];
				if (lenght[v] > lenght[z] + len)
				{
					p[v] = z;
					lenght[v] = lenght[z] + len; //Взяли наименьшую длину
					if (!inq[v])
					{
						q.push(v);
						inq[v] = true;
					}
				}
			}
		}
	}

	rez += "Пути\n";
	for (int i = 0; i < N; i++)
	{
		print_way(i);
		rez += "До вершины: " + to_string(i) + " : "+ timepute + " длина: " + ((lenght[i] == 1e+9)
? ("нет пути из вершины " + to_string(NUM)) : (to_string(lenght[i]))) + " \n";
		timepute = "";
	}
	//Нашли расстояния - строим дерево. BFS дерево - дерево минимальных расстояний
между вершинами

	//Строим BFS дерево
	for (int i = 0; i < N; i++)
	{
		init_pute(i);
		zapolnen();
		R = 0;
	}
```

```
	//Достраиваем то, что потеряли при постройке
	//Вторая часть задачи - компоненты связности

	rez += "Компоненты связности\n";

	for (int i = 0; i < N; i++)
	{
		if (global[i] == 0)
		{
			rez += BFS(i);
		}
	}

	SendData(argv, rez);

	return 0;
}
```