Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Омский государственный технический университет»

Факультет (институт) *Информационных технологий и компьютерных систем*

Кафедра *Прикладная математика и фундаментальная информатика*

Расчетно-графическая работа

по дисциплине ***Дискретная математика***

на тему Применение теории графов

Пояснительная записка

Шифр проекта 020-РГР-02.03.02-№ 14- ПЗ

Студента Курпенова Куата Ибраимовича
фамилия, имя, отчество полностью

Курс *1* Группа ФИТ-212

Направление (специальность) ***02.03.02***
Фундаментальная информатика и информационные технологии
код, наименование

Руководитель ***ст. преподаватель***
ученая степень, звание

Федотова И.В.
фамилия, инициалы

Выполнил 04.06.2022
дата, подпись студента

Работа защищена с количеством баллов	10 баллов

04.06.2022
дата, подпись руководителя

Омск 2022

СОДЕРЖАНИЕ

1 Теоретический анализ

Теоретический анализ задания состоит в ознакомлении с основными понятиями, вводимыми и используемыми при рассмотрении данного задания.

1.1 Основные понятия теории графов

Графом называется любая пара (V,E), где V– непустое множество элементов любой природы, $E = \{e_1,e_2,...\}$– семейство пар $e_i = (u, v)$ элементов из V произвольной кратности и упорядоченности. Обозначают граф G или $G\,(V, E)$.

Элемент множества Vназывается *вершиной*.

Элемент множества E называется *ребром*.

Число вершин графа называется его *порядком* и обозначается $|V|$.

Если вершины v_1 и v_2 соединены ребром $e = (v_1, v_2)$, то говорят, что вершины v_1 и v_2 смежные, а ребро $e = (v_1, v_2)$ инцидентно вершинам v_1 и v_2.

Множество всех вершин графа G смежных с некоторой вершиной v, называется *окружением вершины* v и обозначается как $U(v)$.

Два ребра называются *смежными*, если они имеют общую вершину.

Матрица смежности графа G с конечным числом вершин n – это квадратная матрица A размера $n \times n$, в которой значение элемента a_{ij}равно числу рёбер из *i*-й вершины графа в *j*-ю вершину.

Если E множество упорядоченных пар элементов из V, то граф $G = (V,E)$ называется *ориентированным графом (орграфом).*

В этом случае элементы множества E называются *дугами*.

При этом дуга $e = (v_1, v_2)$ называется исходящей из вершины v_1и заходящей в вершинуv_2. На диаграмме графа дуга изображается линией со стрелкой из вершины v_1 в вершинуv_2.

Если в графе хотя бы одна пара вершин соединена более чем одной ребром, то такой граф называется *мультиграфом*, а ребра называются *кратными.*

Дуги, имеющие одинаковые концевые вершины и одинаково направленные называются *параллельными* или *кратными*, анаправленные противоположно – *противоположно-направленными*.

Кроме того, элементами множества E могут быть пары (v, v) , $v \subset V$, то они называются *петлями*, а графG называется *псевдографом*. Обычно петля считается неориентированной.

Число ребер, инцидентных вершине v, называется *степенью вершины*v и обозначается $deg(v)$ или $d(v)$.

Для ориентированного графа число дуг, исходящих из вершины v, называется *полустепенью исхода* и обозначается через $dev^{+}(v)$, а число дуг, входящих в вершину υ , – *полустепенью захода* и обозначается $dev^{-}(v)$.

Маршрутом в графе$G = (V,E)$ называется чередующаяся последовательность вершин и ребер $\{v_0, e_1,v_1,....e_k,v_k ,... \}$,в которой любые два соседних элемента инцидентны.

Маршрут называется *цепью*, если все его ребра различные. Цепь, соединяющая вершины u и v,обозначается $[u,v]$, и тогда вершина v называется *достижимой* из вершиныu.

Цепь называется *простой*, если все вершины различны.

Для ориентированных графов цепь называется *путем*.

Путь называется *простым*, если все вершины различны.

Граф G называется *связным*, если для любых двух его вершин u и v существует соединяющий их маршрут$[u,v]$.

Вес ребра – числовое значение, поставленное в соответствие данному ребру взвешенного графа.

Взвешенным графом (или нагруженным) называется граф G (E,V) если на нём определена любая функция F: $V \rightarrow R$ (функция на множестве ребер со значениями во множестве вещественных чисел)[1].

2 Решение практической задачи

Далее будет рассмотрена практическая задача и описаны решения ручным и программным способом.

2.1 Постановка задачи

Постановка задачи следующая: «Имеется сеть железнодорожных станций, соединяющих пункты между собой. Каждая линия характеризуется протяженностью в километрах. Определить минимальные расстояния между пунктами.

Формат входных данных

Во входном файле записано сначала число N (1<=N<=100), определявшее количество пунктов. Затем идет описание соединений, где каждое соединение задается тремя числами - номерами узлов, которые она соединяет и протяженностью в километрах. Все соединения строго ориентированы.

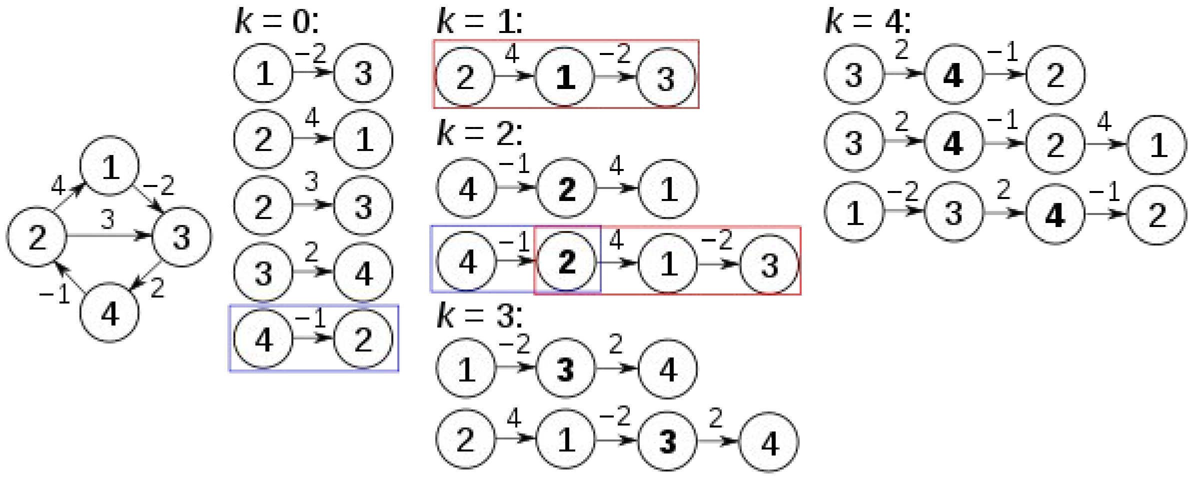
Формат выходных данных

На экран выводятся значения минимальных расстояний между пунктами.

2.2 Выбор метода решения

Для получения ответа нужно применить алгоритм Флойда-Уоршелла, так как именно он позволяет найти самую длинную цепь в графе.

2.3. Описание ручной реализации алгоритма



До первой рекурсии внешнего цикла, обозначенного выше k = 0, единственные известные пути соответствуют отдельным ребрам в графе. При k = 1 находятся пути, проходящие через вершину 1: в частности, найден путь [2,1,3], заменяющий путь [2,3], который имеет меньше ребер, но длиннее (с точки зрения веса). При k = 2 находятся пути, проходящие через вершины 1,2. Красные и синие прямоугольники показывают, как путь [4,2,1,3] собирается из двух известных путей [4,2] и [2,1,3], встреченных в предыдущих итерациях, с 2 на пересечении. Путь [4,2,3] не рассматривается, потому что [2,1,3] - это кратчайший путь, встреченный до сих пор от 2 до 3. При k = 3 пути, проходящие через вершины 1,2,3 найдены. Наконец, при k = 4 находятся все кратчайшие пути.

Матрица расстояний на каждой итерации k, обновленные расстояния выделены **жирным шрифтом**, будет иметь вид:

k = 0			*j*		
		1	**2**	**3**	**4**
	1	0	∞	−2	∞
	2	4	0	3	∞
i	**3**	∞	∞	0	2
	4	∞	−1	∞	0

k = 1			*j*		
		1	**2**	**3**	**4**
	1	0	∞	−2	∞
i	**2**	4	0	**2**	∞

	3	∞	∞	0	2
	4	∞	−1	∞	0

$k = 2$			j		
		1	**2**	**3**	**4**
	1	0	∞	−2	∞
	2	4	0	2	∞
i	**3**	∞	∞	0	2
	4	**3**	−1	**1**	0

$k = 3$			j		
		1	**2**	**3**	**4**
	1	0	∞	−2	**0**
	2	4	0	2	**4**
i	**3**	∞	∞	0	2
	4	3	−1	1	0

$k = 4$			j		
		1	**2**	**3**	**4**
	1	0	**−1**	−2	0
	2	4	0	2	4
i	**3**	**5**	**1**	0	2
	4	3	−1	1	0

2.4 Описания программной реализации алгоритма

Необходимо выполнить программную реализацию алгоритма (Приложение А) и проверить на том же примере. Скриншот работы в Приложении А.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Конова Е.А., Поллак Г.А. Алгоритмы и программы. Язык С++: –издательство «Лань», 2017 – 384 с.

2) Мазалов В.В. Математическая теория игр и приложения: – издательство «Лань», 2016 – 446 с.

3) Омельченко А.В. Теория графов: – Москва: издательство МЦНМО 2018. – 415 с.

4) Скотт Мейерс. Эффективный и современный С++: 42 рекомендации по использованию С++: Пер. с англ. – Вильямс, 2016. – 304 с.

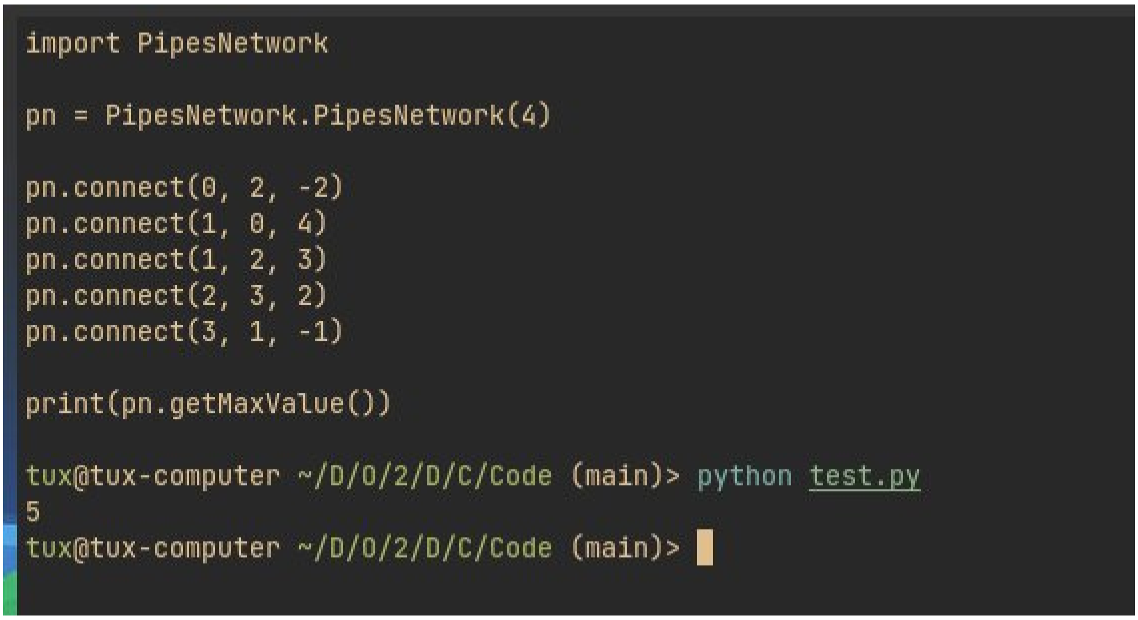
5) Уилсон Р. Введение в теорию графов, 5-е изд: Пер. с англ. – издательство «Диалектика», 2018 – 240 с.

ПРИЛОЖЕНИЕ А

Исходный код

PipesNetwork.py

Скриншот работы программы



```
import PipesNetwork

pn = PipesNetwork.PipesNetwork(4)

pn.connect(0, 2, -2)
pn.connect(1, 0, 4)
pn.connect(1, 2, 3)
pn.connect(2, 3, 2)
pn.connect(3, 1, -1)

print(pn.getMaxValue())

tux@tux-computer ~/D/O/2/D/C/Code (main)> python test.py
5
tux@tux-computer ~/D/O/2/D/C/Code (main)>
```