Министерство образования Российской Федерации  
Пензенский государственный университет  
Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

К курсовому проектированию  
по курсу «Логика и основы алгоритмизации  
в инженерных задачах»

на тему «Реализация алгоритма Дейкстры»

Выполнил:

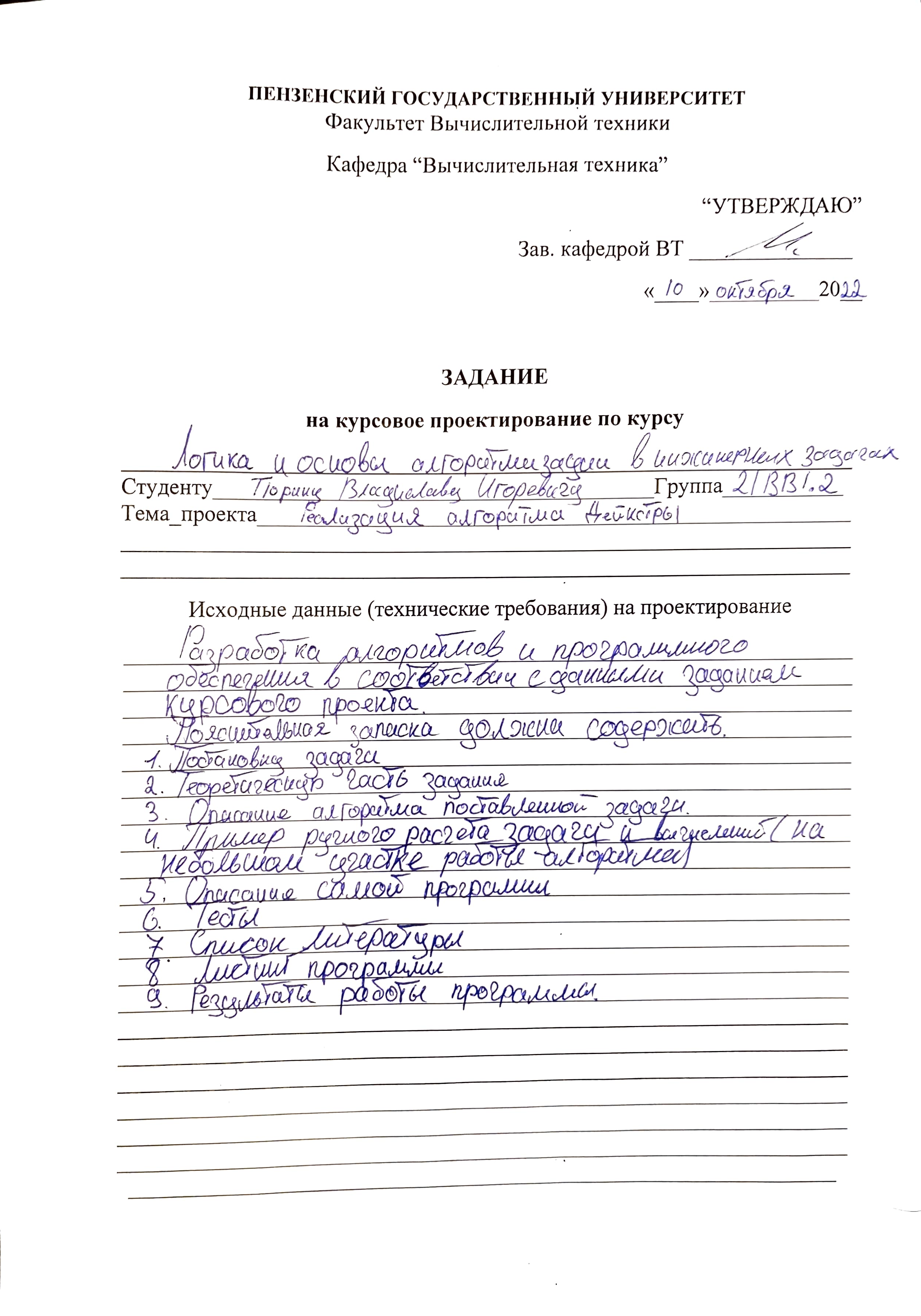
студент группы 21ВВ2

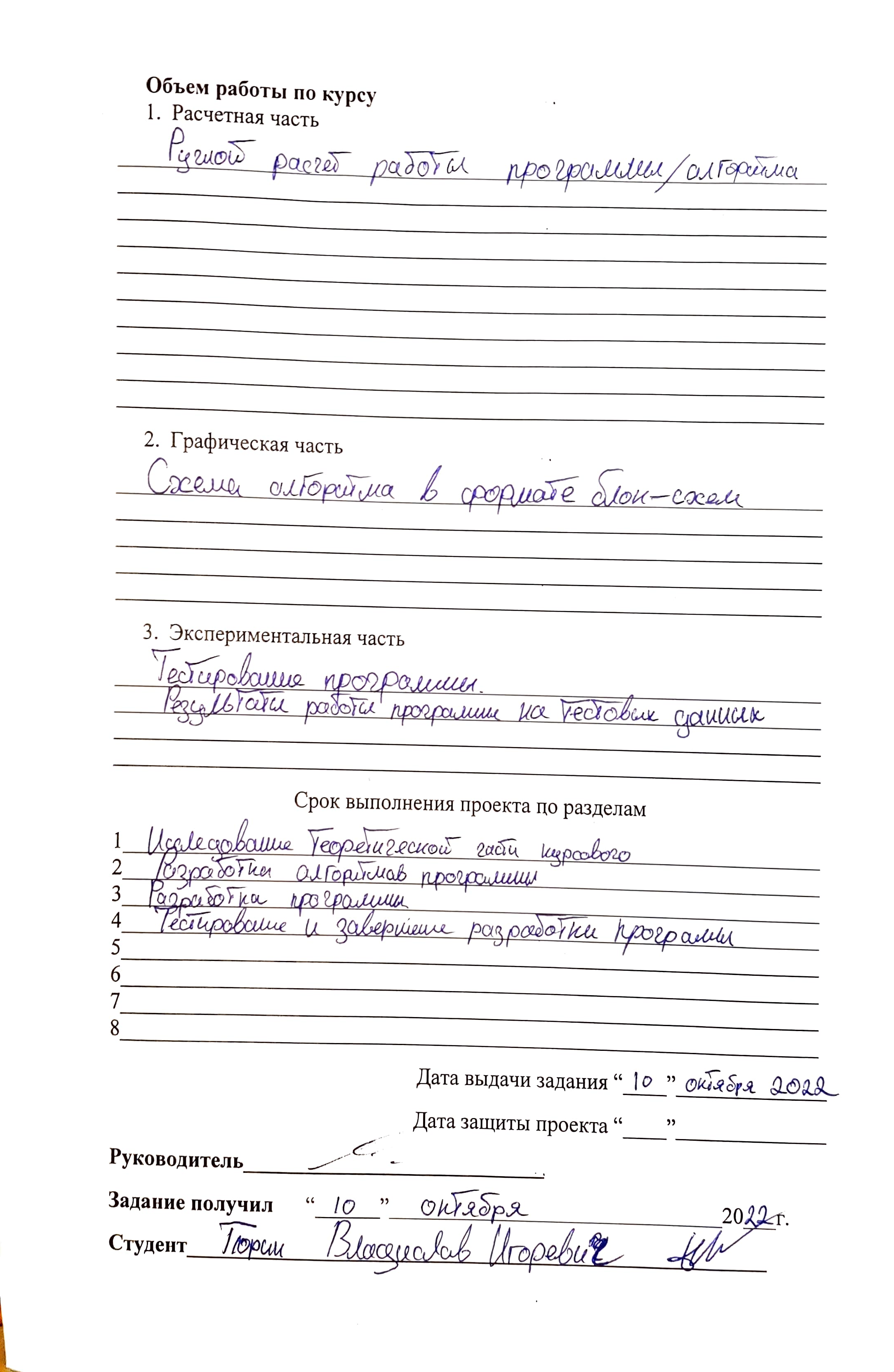
Тюрин В.И.

Принял:  
 Митрохин М.А.

Юрова О.В.

Пенза 2022





**Содержание**

[Введение 5](#_Toc121497678)

[Постановка задачи 6](#_Toc121497679)

[Теоретическая часть задания 7](#_Toc121497680)

[Описание алгоритма программы 8](#_Toc121497681)

[Описание программы 10](#_Toc121497682)

[Тестирование 14](#_Toc121497683)

[Ручной расчет задачи 17](#_Toc121497684)

[Заключение 18](#_Toc121497685)

[Список литературы 19](#_Toc121497686)

[Приложение А. 20](#_Toc121497687)

[Листинг программы. 20](#_Toc121497688)

Введение

Алгоритм Дейкстры – алгоритм на графах, изобретенный нидерландским ученым Эдсгером Дейкстрой в 1959 году. Алгоритм находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании, например, его используют протоколы маршрутизации OSPF и IS-IS.

В качестве среды разработки мною была выбрана среда PyCharm, язык программирования – Python.

Целью данной курсовой работы является разработка программы на языке Python, который является современным и широко используемым. Именно с его помощью в данной курсовом проекте реализуется алгоритм Дейкстры, осуществляющий поиск кратчайшего пути от одной вершины графа до всех остальных.

Постановка задачи

Требуется разработать программу, которая выявит кратчайшие пути от одной вершины графа до других, используя алгоритм Дейкстры.

Исходный граф в программе должен задаваться матрицей смежности, причем при генерации данных должны быть предусмотрены граничные условия. Программа должна работать так, чтобы пользователь вводил количество вершин для генерации матрицы смежности. После обработки данных на экран должна выводиться матрица смежности графа. Необходимо предусмотреть различные исходы работы алгоритма, чтобы программа не выдавала ошибок.

Устройство ввода – клавиатура и мышь.

Задания выполняются в соответствии с вариантом №13.

Теоретическая часть задания

Граф G (рисунок 1) задается множеством вершин X1, X2, ..., Xn. и множеством ребер, соединяющих между собой определенные вершины. Ребра из множества А ориентированы, что показывается линией, которая указывает достижимость данной вершины, граф с такими ребрами называется неориентированным.

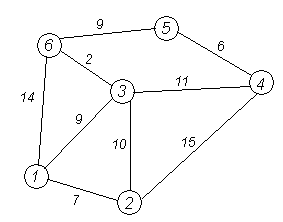


Рисунок 1 - Пример графа

При представлении графа матрицей смежности информация о ребрах графа хранится в квадратной матрице, где присутствие пути из одной вершины в другую обозначается единицей, иначе нулем. После этого по данной матрице строится еще одна матрица смежности, которая показывает вес дуги.

Каждой вершине из *V* сопоставим метку — минимальное известное расстояние от этой вершины до *a*. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Описание алгоритма программы

Для программной реализации алгоритма понадобится один массив: T – массив, в который записываются минимальные пути. Словарь S, куда записываются посещенные вершины. Имеется граф G=(I, J). Каждая из вершин графа пока не занесена в словарь с посещенными вершинами.

В качестве исходного пункта выбирается вершина 1. В массиве T задается нулевое значение для начальной вершины. Пока не просмотрены все вершины выполняется цикл: с помощью функции getlink() и цикла for мы получаем все связанные вершины с текущей, и если вершина не записана в словарь S, то есть не посещена, то в переменную w записывается ее вес(сумма путей), если текущий вес меньше чем тот что уже записан, то в этом случае сохраняем новые значения связи. Далее выбирается вершина с минимальным весом с помощью функции arg\_min(), если данная вершина больше нуля, то мы ее добавляем во множество S. После завершения цикла в массиве T будут записаны кратчайшие пути от начальной вершины ко всем остальным.

Ниже представлен псевдокод алгоритма и функций getlink() и arg\_min().

getlink()

1. цикл: i и weight в enumerate(gg[v])
   1. если weigth>0, то
      1. Вернуть i

arg\_min()

1. amin = -1
2. m= максимальное значение из массива T
3. цикл: i и t в enumerate(T)
   1. если t < m & I не находится в S, то
      1. m = t
      2. amin = i
4. вернуть amin

Сам алгоритм

S = {v}

T[v] = 0

1. пока v не равно -1 делать:
   1. цикл: j в getlink(v,gg)
      1. Если j не в S, то
         1. w = T[v] + gg[v][j]
         2. если w < T[j], то
            1. T[j] = w
   2. v = arg\_min(T, S)
   3. Если v >= 0, то
      1. v добавить в S

Полный код можно увидеть в Приложении А.

Описание программы

Для написания данной программы использовался язык программирования Python. Это универсальный и современный язык программирования, который завоевал свою популярность у программистов, благодаря богатой стандартной библиотеке и комфортным порогом входа.

Проект был создан в виде консольного приложения.

Данная программа является многомодульной, поскольку состоит из нескольких функций: getlink, arg\_min, creatematrix, start, printm, print2file, print2secfile, printresult.

Работа начинается с выбора генерации графа. На выбор пользователю предлагается 2 варианта: Случайная генерация с вводом только количества вершин или прочитать матрицу смежности из файла, в который пользователь ввел ее заранее. Если пользователь выбрал случайную генерацию, то программа запросит ввести количество вершин, после чего программа сгенерирует граф заданного размера.

if wi==1 :  
 while True:  
 try:  
 n = int(input("Введите размер матрицы: "))  
 except:  
 print("Введите число")  
 else:  
 break  
 gg = creatematrix(n)  
 break

def creatematrix(value):  
 matrx = [[0] \* (value) for \_ in range(value)]  
 i = 0  
 while (i < len(matrx)):  
 j = i  
 while (j < len(matrx[i])):  
 if (i == j):  
 matrx[i][j] = 0  
 else:  
 matrx[i][j] = int((randint(0, 1)))  
 if (matrx[i][j] == 1):  
 matrx[i][j] = int((randint(0, 99)))  
 matrx[j][i] = matrx[i][j]  
 j += 1  
 i += 1  
 return matrx

Далее выводятся элементы массива, а именно матрица смежности сгенерированного графа и предложит выбрать вершину, для которой будет выполняться алгоритм. (рис.2).

def printm(mtx, n):  
 print()  
 print(' ', end=' ')  
 [print("%2d" % i, end=' ') for i in range(0, len(mtx))]  
 print()  
 print('-' \* (len(mtx[0]) \* 3 + 4))  
  
 for i, row in enumerate(mtx):  
 print("%2d |" % (i), end=' ')  
 for col in row:  
 print("%2d" % col, end=' ')  
 print()

def start():  
 while True:  
 try:  
 print()  
 v=int(input("Введите стартовую вершину: "))   
 except:  
 print("Введите число")  
 else:  
 break  
 return v

Если же пользователь выбрал ввод матрицы смежности из файла, то программа сразу выведет данную матрицу и предложит выбрать вершину, для которой будет выполняться алгоритм. (рис.3).

gg = []  
file = open("matrix.txt", "r")  
with open('matrix.txt') as f:  
#data = file.read()  
 for line in f:  
 n = len(line)  
 gg.append([float(x) for x in line.split()])

def start():  
 while True:  
 try:  
 print()  
 v=int(input("Введите стартовую вершину: "))   
 except:  
 print("Введите число")  
 else:  
 break  
 return v

Ниже можно увидеть оформление начального запроса и дальнейшее действия с ним.

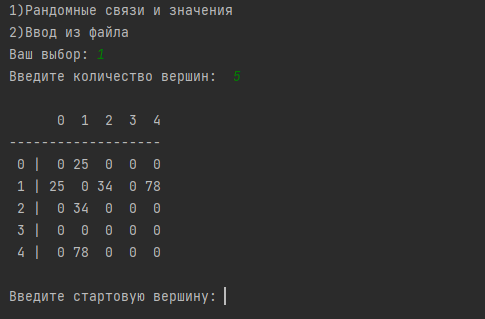


Рисунок 2 - Случайная генерация матрицы

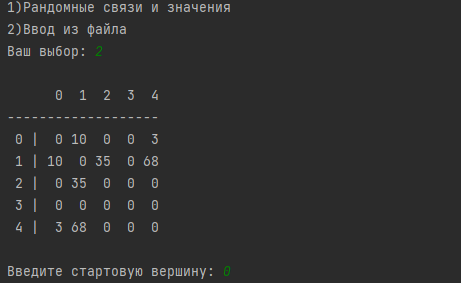


Рисунок 3 - Ввод матрицы смежности из файла

После ввода вершины, программа выводит ответ и записывает его в файл(рис.4)(рис.5).

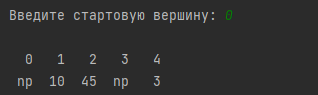


Рисунок 4 - вывод ответа

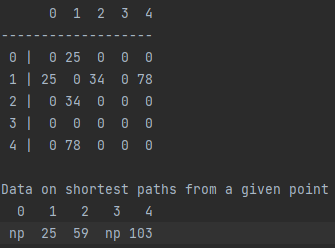


Рисунок 5 - вывод ответа и матрицы в файл

Тестирование

Среда разработки PyCharm предоставляет все средства, необходимые при разработке и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом данных, изменением дизайна выводимых данных, взаимодействием функций.

Ниже продемонстрирован результат тестирования программы при вводе пользователем количества вершин графа и вводе матрицы смежности из файла.

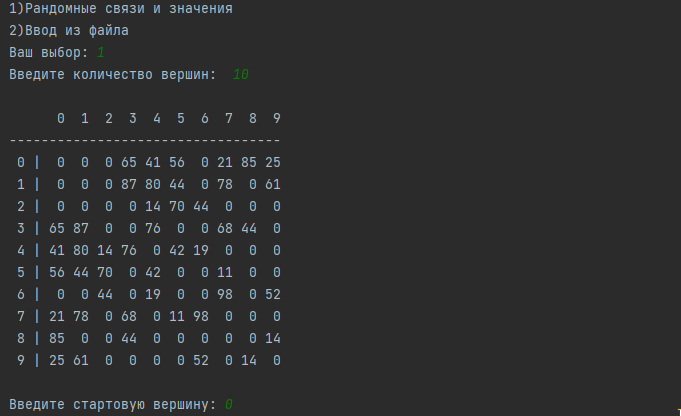


Рисунок 6 - Тестирование при вводе количества вершин = 10

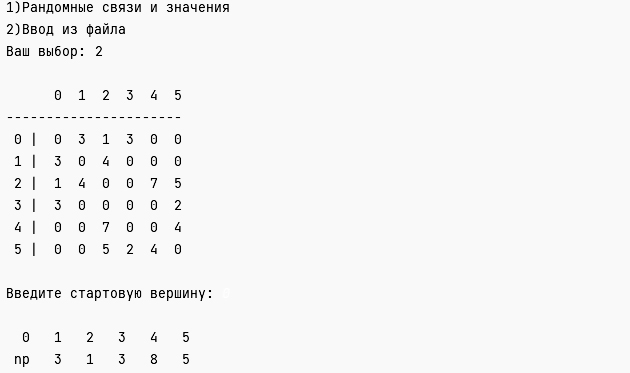


Рисунок 7 - Тестирование при вводе матрицы смежности из файла

Таблица 1 - Описание поведения программы при тестировании

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Вывод сообщения о выборе: сгенерировать матрицу или вывести матрицу из файла | Верно |
| Выбор генерации матрицы | Вывод сообщения о количестве вершин в орграфе | Верно |
| Выбор вывода матрицы из  файла | Вывод матрицы | Верно |
| Выбор стартовой вершины | Вывод сообщения о выборе стартовой вершины | Верно |
| Вывод ответа | Вывод вершин графа и под ними кратчайшие пути из выбранной вершины | Верно |

В результате тестирования было выявлено, что программа успешно находит кратчайшие пути, использую алгоритм Дейкстры.

Ручной расчет задачи

Проведем проверку программы посредством ручных вычислений на примере графа с 6 вершинами (рисунок 7).

Начинаем алгоритм из вершины 0. Проверяем, если есть путь из вершины 0 в другие вершины, то идем дальше. Далее вычисления будут представлены в виде таблицы (таблица 2).

Таблица 2 - Ручной расчет программы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| 2 |  | 3 | 1 | 3 | ∞ | ∞ |
| 3 |  | 3 |  | 3 | 8 | 6 |
| 4 |  |  |  | 3 | 8 | 6 |
| 5 |  |  |  |  | 8 | 5 |
| 6 |  |  |  |  | 8 |  |

Таким образом, можно сделать вывод, что программа работает верно.

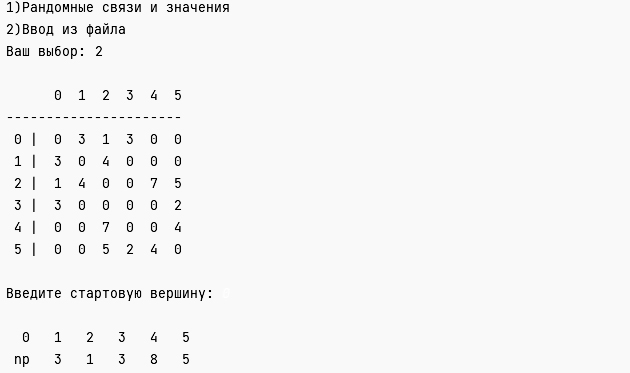


Рисунок 8 - Тестирование работы программы

Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм Дейкстры по поиску кратчайших путей из одной вершины ко всем остальным в PyCharm.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежностей. Приобретены навыки по осуществлению алгоритма Дейкстры. Углублены знания языка программирования Python.

Недостатком разработанной программы является примитивный пользовательский интерфейс. Потому что программа работает в консольном режиме, не добавляющем к сложности языка сложность программного оконного интерфейса.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

Список литературы

1. Алгоритм Дейкстры — Википедия (wikipedia.org)
2. Онлайн-школа Фоксфорд (foxford.ru)
3. Ананий В. Левитин Глава 9. Жадные методы: Алгоритм Дейкстры
4. Марк Лутц. Изучаем Python

Приложение А.

Листинг программы.

from random import randint  
import math  
  
  
def getlink(v, gg):  
 for i, weight in enumerate(gg[v]):  
 if weight > 0:  
 yield i  
  
  
def arg\_min(T, S):  
 amin = -1  
 m = max(T) # максимальное значение  
 for i, t in enumerate(T):  
 if t < m and i not in S:  
 m = t  
 amin = i  
 return amin  
  
  
def creatematrix(value):  
 matrx = [[0] \* (value) for \_ in range(value)]  
 i = 0  
 while (i < len(matrx)):  
 j = i  
 while (j < len(matrx[i])):  
 if (i == j):  
 matrx[i][j] = 0  
 else:  
 matrx[i][j] = int((randint(0, 1)))  
 if (matrx[i][j] == 1):  
 matrx[i][j] = int((randint(0, 99)))  
 matrx[j][i] = matrx[i][j]  
 j += 1  
 i += 1  
 return matrx  
  
  
def start():  
 while True:  
 try:  
 print()  
 v=int(input("Введите стартовую вершину: ")) # стартовая вершина (нумерация с нуля)  
 except:  
 print("Введите число")  
 else:  
 break  
 return v  
  
"""""  
def printm(mat,n):  
 for i in range(len(mat)):  
 print('',i,end=" ")  
 print("\n",'-'\*len(mat)\*3)  
 for row in mat:  
 print(' '.join(s.rjust(len(str(n \* n)), ' ')  
 for s in map(str, row)))  
 #print()  
 #for mat in mat:  
 #print(\*mat)  
"""""  
def printm(mtx, n):  
 print()  
 print(' ', end=' ')  
 [print("%2d" % i, end=' ') for i in range(0, len(mtx))]  
 print()  
 print('-' \* (len(mtx[0]) \* 3 + 4))  
  
 for i, row in enumerate(mtx):  
 print("%2d |" % (i), end=' ')  
 for col in row:  
 print("%2d" % col, end=' ')  
 print()  
  
  
def print2file(gg, T):  
 fout = open('Dxtra.txt', 'w') # 'w' - это режим "запись" ("write")  
 #for i in range(len(gg)):  
 #print(gg[i],file=fout)  
 print(file=fout)  
 print(' ', end=' ',file=fout)  
 [print("%2d" % i, end=' ',file=fout) for i in range(0, len(gg))]  
 print(file=fout)  
 print('-' \* (len(gg[0]) \* 3 + 4),file=fout)  
 for i, row in enumerate(gg):  
 print("%2d |" % (i), end=' ',file=fout )  
 for col in row:  
 print("%2d" % col, end=' ',file=fout)  
 print(file=fout)  
 print(file=fout)  
 print("Data on shortest paths from a given point",file=fout)  
 for i in range(N):  
 print("%3d" % i, end=" " , file=fout)  
 print(file=fout)  
 for i in range(len(T)):  
 if (T[i]== math.inf) or (T[i]==0):  
 print("","np",end=" ",file=fout)  
 else:  
 print("%3d" % T[i],end=" ",file=fout)  
 fout.close()  
 return  
  
  
def print2secfile(gg, T):  
 fout = open('matrix.txt', 'w') # 'w' - это режим "запись" ("write")  
 for i, row in enumerate(gg):  
 for col in row:  
 print("%2d" % col, end=' ',file=fout)  
 print(file=fout)  
 return  
  
  
def printresult(N, T):  
 print()  
 for i in range(N):  
 print("%3d" % i, end=" ")  
  
 print()  
 for i in range(len(T)):  
 if (T[i]== math.inf) or (T[i]==0):  
 print("","np",end=" ")  
 else:  
 print("%3d" % T[i],end=" ")  
 return  
  
  
while True:  
 try:  
 wi = int(input("1)Рандомные связи и значения\n2)Ввод из файла\nВаш выбор: "))  
 except:  
 print("Введите число")  
 else:  
 break  
  
  
while True:  
 if wi==1 :  
 while True:  
 try:  
 n = int(input("Введите количество вершин: "))  
 except:  
 print("Введите число")  
 else:  
 break  
 gg = creatematrix(n)  
 break  
 elif wi==2 :  
 gg = []  
 file = open("matrix.txt", "r")  
 with open('matrix.txt') as f:  
 #data = file.read()  
 for line in f:  
 n = len(line)  
 gg.append([float(x) for x in line.split()])  
 break  
 elif wi!=1 or wi!=2:  
 while True:  
 try:  
 print()  
 wi = int(input("1)Рандомные связи и значения\n2)Ввод из файла\nВаш выбор: "))  
 except:  
 print("Введите число 1 или 2 ")  
 else:  
 break  
 else:  
 break  
  
#gg=createMatrix(n)  
printm(gg, n)  
N=len(gg)  
T = [math.inf]\*N  
v=start()  
  
  
while v>n or v==n :  
 print("Введите существующую вершину!!! ")  
 v=start()  
  
S = {v} # просмотренные вершины  
T[v] = 0 # нулевой вес для стартовой вершины  
  
  
while v != -1: # цикл, пока не просмотрим все вершины  
 for j in getlink(v, gg): # перебираем все связанные вершины с вершиной v  
 if j not in S: # если вершина еще не просмотрена  
 w = T[v] + gg[v][j]  
 if w < T[j]:  
 T[j] = w  
 v = arg\_min(T, S) # выбираем следующий узел с наименьшим весом  
 if v >= 0: # выбрана очередная вершина  
 S.add(v) # добавляем новую вершину в рассмотрение  
  
  
printresult(N, T)  
if wi == 1:  
 print2file(gg, T)  
print2secfile(gg, T)