Министерство образования и науки РФ

Федеральное государственное автономное 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 | Разработка программы |

Пояснительная записка

|  |  |
| --- | --- |
| **Шифр проекта** | 020-РГР-02.03.02-№ 22-ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Студента** | | Фельде Сергея Дмитриевича | | | | | |
|  |  |  |  | | фамилия, имя, отчество полностью | | | | | |
|  |  |  | Курс | *1* |  | Группа | | ФИТ-**231** | | |
|  |  |  |  |  |  | |  |  | |  |
|  | | | **Направление (специальность)** | | | | | ***02.03.02*** | | |
|  | | | *Фундаментальная информатика и информационные технологии* | | | | | | | |
|  |  |  | код, наименование | | | | | | | |
|  |  |  | Руководитель | | ***ст. преподаватель*** | | | | | |
|  |  |  | ученая степень, звание | | | | | |
|  |  |  | ***Федотова И.В.*** | | | | | | | |
|  |  |  | фамилия, инициалы | | | | | | | |
|  |  |  | Выполнил | |  | | | | | |
|  |  |  | дата, подпись студента | | | | | |
|  |  |  | **Работа защищена с количеством баллов** | | | | | | | |
|  |  |  |  | | | | | |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | дата, подпись руководителя |  |  |  |

Омск 2024

**Содержание**

[Введение 3](#_Toc167034402)

[Постановка задачи 4](#_Toc167034403)

[Теоретическая часть 5](#_Toc167034404)

[Программная реализация 7](#_Toc167034405)

[Результаты 11](#_Toc167034406)

[Заключение 12](#_Toc167034407)

[Список использованной литературы 13](#_Toc167034408)

Введение

На сегодняшний день информационные технологии играют большую роль в жизни общества. Прогресс не стоит на месте, и теперь эти технологии применяются практически во всех сферах жизни человечества. Но для решения многих задач могут быть очень полезны знания дискретной математики.

Постановка задачи

Условие задачи (№22).

В стране Угандии построили аттракцион "Лабиринт знаний". Лабиринт представляет собой N комнат, пронумерованных от 1 до N, между некоторыми из которых есть двери. Когда человек проходит через дверь, показатель его знаний изменяется на определенную величину, фиксированную для данной двери. Вход в лабиринт находится в комнате 1, выход – в комнате N. Каждый человек проходит лабиринт ровно один раз. Определить наилучший результат, который можно набрать при прохождение лабиринта.

Формат входных данных.

Первая строка входного файла содержит целые числа N (1 ≤ N ≤ 2000) – количество n комнат и M (0 ≤ M ≤ 10000) – количество дверей. В каждой из следующих M строк содержится описание двери – номера комнат, из которой она ведет и в которую она ведет (через дверь можно ходить только в одном направлении), а также целое число, которое прибавляется к количеству знаний при прохождении через дверь (это число по модулю не превышает 10000).

Формат выходных данных.

В выходной файл вывести сообщение если лабиринт пройти нельзя или максимальное количество набранных знаний в противном случае.

Теоретическая часть

Задача была решена с применением алгоритма Форда – Беллмана. Но для получения максимального количества набранных знаний в алгоритме использовались числа с противоположным знаком.

Рассмотрим пример с графом, заданным матрицей весов (рисунок 1), где вершины V являются комнатами, а дуги с весами – дверями с числами прибавляемых знаний.

Так как мы ищем максимальный путь от первой вершины до 4, то необходимо заменить числа в матрице на числа с противоположным знаком (рисунок 2).

Дальше строим новую таблицу (рисунок 3), где i – это номер итерации, а число за чертой – это вершина, из которой пришли. Так на итерации 0 путь до начальной вершины 1 равен 0.

На итерации 1 только в вершину 2 можно попасть из вершины 1, поэтому записываем соответствующее значение.

На итерации 2 из вершины 2 можно попасть в вершину 3 и вершину 4. Поэтому записываем соответствующие длины путей до этих вершин, которые являются суммами длины пути от вершины 1 до вершины 2 и длины пути от 2 до 3 и 4 соответственно.

На итерации 3 из вершины 3 можно попасть в вершину 2 и 4. Так как длина пути 1 – 2 меньше длины пути 1 – 2 – 3 – 2, то переходим к сравнению путей 1 – 2 – 4 и 1 – 2 – 3 – 4. Здесь уже длина пути 1 – 2 – 3 – 4 меньше и мы выбираем её.

Теперь для получения максимального пути необходимо изменить знаки у чисел на противоположные.

В результате ручного вычисления с помощью данного алгоритма мы получили максимальный путь до вершины 4, который равен 17.

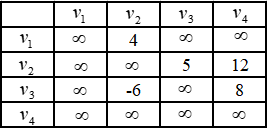


Рисунок 1 – Матрица весов

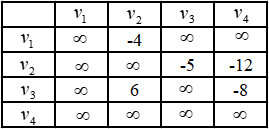


Рисунок 2 – Матрица весов с противоположными знаками

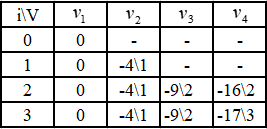


Рисунок 3 – Таблица работы алгоритма

Программная реализация

**Структура**

Код выполнен в среде PyCharm Community Edition на языке Python.

Программа состоит из трёх основных функций: Menu – функция, вызывающая меню, Alg – функция, выполняющая алгоритм, Create – функция, создающая выходной файл.

**Код программы**

def Menu():  
 print("1) Начать \n2) Условия задачи \n3) Об авторе \n4) Выход")  
 m = input("Выберите действие: ")  
 match m:  
 case "1":  
 print(Alg())  
 Create()  
 case "2":  
 print("В стране Угандии построили аттракцион 'Лабиринт знаний'.\nЛабиринт представляет собой N комнат, пронумерованных от\n1 до N, между некоторыми из которых есть двери.\nКогда человек проходит через дверь, показатель его знаний\nизменяется на определенную величину, фиксированную для данной двери.\nВход в лабиринт находится в комнате 1, выход - в комнате N.\nКаждый человек проходит лабиринт ровно один раз.\nОпределить наилучший результат, который можно набрать при прохождение лабиринта.")  
 case "3":  
 print("Фельде Сергей Дмитриевич\nСтудент группы ФИТ-231\nОМГТУ 2024")  
 case "4":  
 exit()  
def Alg():  
 def get\_connects(vershina):  
 lst = []  
 for elem in data:  
 if elem[0][1] == vershina:  
 lst.append(elem)  
 return lst  
 with open('rgr\_test.txt') as f:  
 try:  
 N = int(f.readline())  
 M = int(f.readline())  
 except ValueError:  
 output = 'Неправильный ввод'  
 return output  
 if N > 2000 or M > 10000 or N < 1 or M < 0:  
 output = 'Неправильный ввод'  
 return output  
 data = []  
 cur\_v = set()  
 for i in range(M):  
 try:  
 nums = [int(x) for x in f.readline().split()]  
 except ValueError:  
 output = 'Неправильный ввод'  
 return output  
 cur\_v.add(nums[0])  
 cur\_v.add(nums[1])  
 w = -int(nums[2])  
 if abs(w) > 10000:  
 output = 'Неправильный ввод'  
 return output  
 data.append((nums, w))  
 if f.readline() != '':  
 output = 'Неправильный ввод'  
 return output  
 if M == 0:  
 output = 'Лабиринт нельзя пройти'  
 return output  
 prev\_list, cur\_list = [], []  
 for i in range(N):  
 prev\_list.append('inf')  
 cur\_list.append('inf')  
 cur\_v = sorted(list(cur\_v))  
 o = min(cur\_v)  
 start\_v = 1  
 cur\_list[start\_v - o] = 0  
 dests = {}  
 for v in cur\_v:  
 dests[v] = 'inf'  
 if v == start\_v:  
 dests[v] = 0  
 iteration = 0  
 no\_otriz = True  
 while True:  
 has\_changed = False  
 for ver in cur\_v:  
 if ver == start\_v:  
 continue  
 v\_list = get\_connects(ver)  
 if v\_list:  
 for elem in v\_list:  
 if dests[elem[0][0]] != 'inf':  
 if cur\_list[ver - o] == 'inf':  
 way\_len = elem[1] + dests[elem[0][0]]  
 cur\_list[ver - o] = way\_len, elem[0][0]  
 dests[ver] = way\_len  
 else:  
 way\_len = elem[1] + dests[elem[0][0]]  
 if way\_len < cur\_list[ver - o][0]:  
 cur\_list[ver - o] = way\_len, elem[0][0]  
 dests[ver - 0] = way\_len  
 if prev\_list != cur\_list:  
 has\_changed = True  
 if has\_changed and iteration == N:  
 output = 'Бесконечное количество набранных знаний'  
 no\_otriz = False  
 break  
 if not has\_changed:  
 break  
 iteration += 1  
 prev\_list = cur\_list[:]  
 if no\_otriz:  
 if N in dests:  
 output = f'Максимальное количество набранных знаний - {-dests[N]}'  
 else:  
 output = 'Лабиринт нельзя пройти'  
 return output  
def Create():  
 rgr\_output = open("rgr\_output.txt", "w+")  
 rgr\_output.write(Alg())  
 rgr\_output.close()  
while True:  
 Menu()

Результаты

На основе выбранного пользователем пункта из отображаемого меню можно получить: результаты работы алгоритма (рисунок 4), условия задачи (рисунок 5), информацию об авторе программы (рисунок 6).

Результаты, полученные в результате выполнения программы и ручного вычисления на основе алгоритма Форда – Беллмана, полностью совпадают. Это означает исправную работу алгоритма программы.

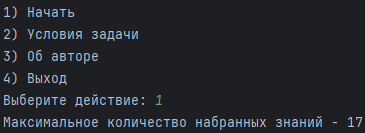


Рисунок 4 – Результаты выполнения программы

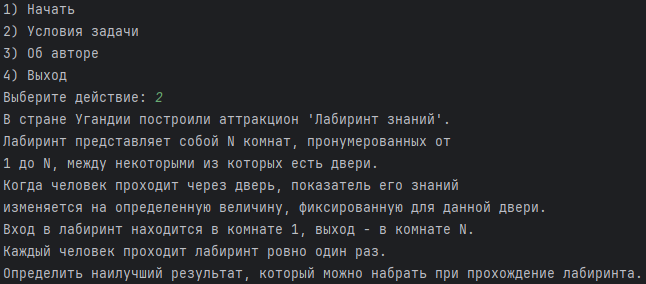


Рисунок 5 – Условия задачи

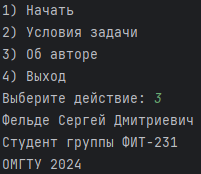


Рисунок 6 – Информация об авторе

Заключение

В ходе выполнения расчётно-графической работы была разработана и реализована программа для вычисления максимального пути на основе уже существующего алгоритма Форда – Беллмана.

Список использованной литературы

1. METANIT. https://metanit.com/python/ (Дата обращения: 19.04.24)
2. Python Lessons. https://pylessons.readthedocs.io/ru/latest/contents.html# (Дата обращения: 19.04.24)