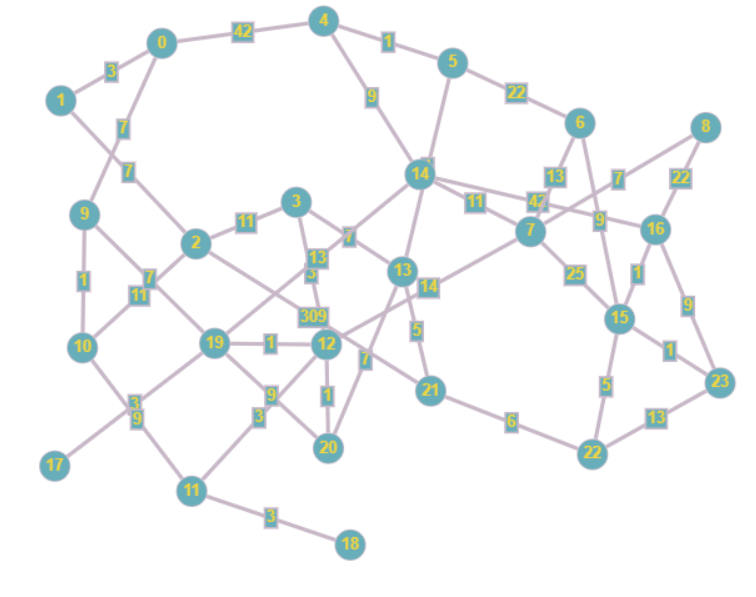
Отчет по лабораторной работе 3

В данной работе была реализована программа на языке Python для нахождения кратчайшего пути в графе по алгоритму Флойда-Уоршелла. На выходе алгоритм выдаёт на основе входного графа другой граф – граф наименьших расстояний между вершинами (без пройденного пути). В данном случае алгоритм был расширен и в выходных данных выводится также граф предыдущих «вершин» - таким образом возможно вывести пройденный пут из любой вершины до любой вершины (если это возможно).

В качестве входных данных задается матрица смежности, например:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 3 | 10 | 18 | 25 | 24 | 42 | 29 | 36 | 7 | 8 | 17 | 15 | 23 | 27 | 39 | 40 | 17 | 20 | 14 | 16 | 28 | 34 | 40 |  |
| 3 | 0 | 7 | 18 | 27 | 26 | 45 | 32 | 39 | 10 | 11 | 20 | 18 | 25 | 30 | 41 | 42 | 20 | 23 | 17 | 19 | 30 | 36 | 42 |  |
| 10 | 7 | 0 | 11 | 20 | 19 | 41 | 28 | 35 | 12 | 11 | 17 | 14 | 18 | 28 | 34 | 35 | 18 | 20 | 15 | 15 | 23 | 29 | 35 |  |
| 18 | 18 | 11 | 0 | 9 | 8 | 30 | 17 | 24 | 11 | 12 | 6 | 3 | 7 | 17 | 23 | 24 | 7 | 9 | 4 | 4 | 12 | 18 | 24 |  |
| 25 | 27 | 20 | 9 | 0 | 1 | 23 | 20 | 27 | 18 | 19 | 13 | 10 | 2 | 9 | 18 | 19 | 14 | 16 | 11 | 9 | 7 | 13 | 19 |  |
| 24 | 26 | 19 | 8 | 1 | 0 | 22 | 21 | 28 | 17 | 18 | 12 | 9 | 1 | 10 | 17 | 18 | 13 | 15 | 10 | 8 | 6 | 12 | 18 |  |
| 42 | 45 | 41 | 30 | 23 | 22 | 0 | 13 | 20 | 35 | 36 | 30 | 27 | 23 | 24 | 9 | 10 | 31 | 33 | 28 | 28 | 20 | 14 | 10 |  |
| 29 | 32 | 28 | 17 | 20 | 21 | 13 | 0 | 7 | 22 | 23 | 17 | 14 | 22 | 11 | 22 | 23 | 18 | 20 | 15 | 15 | 27 | 27 | 23 |  |
| 36 | 39 | 35 | 24 | 27 | 28 | 20 | 7 | 0 | 29 | 30 | 24 | 21 | 29 | 18 | 23 | 22 | 25 | 27 | 22 | 22 | 34 | 28 | 24 |  |
| 7 | 10 | 12 | 11 | 18 | 17 | 35 | 22 | 29 | 0 | 1 | 10 | 8 | 16 | 20 | 32 | 33 | 10 | 13 | 7 | 9 | 21 | 27 | 33 |  |
| 8 | 11 | 11 | 12 | 19 | 18 | 36 | 23 | 30 | 1 | 0 | 9 | 9 | 17 | 21 | 33 | 34 | 11 | 12 | 8 | 10 | 22 | 28 | 34 |  |
| 17 | 20 | 17 | 6 | 13 | 12 | 30 | 17 | 24 | 10 | 9 | 0 | 3 | 11 | 17 | 27 | 28 | 7 | 3 | 4 | 4 | 16 | 22 | 28 |  |
| 15 | 18 | 14 | 3 | 10 | 9 | 27 | 14 | 21 | 8 | 9 | 3 | 0 | 8 | 14 | 24 | 25 | 4 | 6 | 1 | 1 | 13 | 19 | 25 |  |
| 23 | 25 | 18 | 7 | 2 | 1 | 23 | 22 | 29 | 16 | 17 | 11 | 8 | 0 | 11 | 16 | 17 | 12 | 14 | 9 | 7 | 5 | 11 | 17 |  |
| 27 | 30 | 28 | 17 | 9 | 10 | 24 | 11 | 18 | 20 | 21 | 17 | 14 | 11 | 0 | 27 | 28 | 16 | 20 | 13 | 15 | 16 | 22 | 28 |  |
| 39 | 41 | 34 | 23 | 18 | 17 | 9 | 22 | 23 | 32 | 33 | 27 | 24 | 16 | 27 | 0 | 1 | 28 | 30 | 25 | 23 | 11 | 5 | 1 |  |
| 40 | 42 | 35 | 24 | 19 | 18 | 10 | 23 | 22 | 33 | 34 | 28 | 25 | 17 | 28 | 1 | 0 | 29 | 31 | 26 | 24 | 12 | 6 | 2 |  |
| 17 | 20 | 18 | 7 | 14 | 13 | 31 | 18 | 25 | 10 | 11 | 7 | 4 | 12 | 16 | 28 | 29 | 0 | 10 | 3 | 5 | 17 | 23 | 29 |  |
| 20 | 23 | 20 | 9 | 16 | 15 | 33 | 20 | 27 | 13 | 12 | 3 | 6 | 14 | 20 | 30 | 31 | 10 | 0 | 7 | 7 | 19 | 25 | 31 |  |
| 14 | 17 | 15 | 4 | 11 | 10 | 28 | 15 | 22 | 7 | 8 | 4 | 1 | 9 | 13 | 25 | 26 | 3 | 7 | 0 | 2 | 14 | 20 | 26 |  |
| 16 | 19 | 15 | 4 | 9 | 8 | 28 | 15 | 22 | 9 | 10 | 4 | 1 | 7 | 15 | 23 | 24 | 5 | 7 | 2 | 0 | 12 | 18 | 24 |  |
| 28 | 30 | 23 | 12 | 7 | 6 | 20 | 27 | 34 | 21 | 22 | 16 | 13 | 5 | 16 | 11 | 12 | 17 | 19 | 14 | 12 | 0 | 6 | 12 |  |
| 34 | 36 | 29 | 18 | 13 | 12 | 14 | 27 | 28 | 27 | 28 | 22 | 19 | 11 | 22 | 5 | 6 | 23 | 25 | 20 | 18 | 6 | 0 | 6 |  |
| 40 | 42 | 35 | 24 | 19 | 18 | 10 | 23 | 24 | 33 | 34 | 28 | 25 | 17 | 28 | 1 | 2 | 29 | 31 | 26 | 24 | 12 | 6 | 0 |  |

На основе её можно построить граф и/или проводить вычисления:



Код представлен по URL: <https://replit.com/@AlieksandrAnan2/lab3-1>

На выходе была получена матрица наименьших расстояний вида:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 3 | 10 | 18 | 25 | 24 | 42 | 29 | 36 | 7 | 8 | 17 | 15 | 23 | 27 | 39 | 40 | 17 | 20 | 14 | 16 | 28 | 34 | 40 |  |
| 3 | 0 | 7 | 18 | 27 | 26 | 45 | 32 | 39 | 10 | 11 | 20 | 18 | 25 | 30 | 41 | 42 | 20 | 23 | 17 | 19 | 30 | 36 | 42 |  |
| 10 | 7 | 0 | 11 | 20 | 19 | 41 | 28 | 35 | 12 | 11 | 17 | 14 | 18 | 28 | 34 | 35 | 18 | 20 | 15 | 15 | 23 | 29 | 35 |  |
| 18 | 18 | 11 | 0 | 9 | 8 | 30 | 17 | 24 | 11 | 12 | 6 | 3 | 7 | 17 | 23 | 24 | 7 | 9 | 4 | 4 | 12 | 18 | 24 |  |
| 25 | 27 | 20 | 9 | 0 | 1 | 23 | 20 | 27 | 18 | 19 | 13 | 10 | 2 | 9 | 18 | 19 | 14 | 16 | 11 | 9 | 7 | 13 | 19 |  |
| 24 | 26 | 19 | 8 | 1 | 0 | 22 | 21 | 28 | 17 | 18 | 12 | 9 | 1 | 10 | 17 | 18 | 13 | 15 | 10 | 8 | 6 | 12 | 18 |  |
| 42 | 45 | 41 | 30 | 23 | 22 | 0 | 13 | 20 | 35 | 36 | 30 | 27 | 23 | 24 | 9 | 10 | 31 | 33 | 28 | 28 | 20 | 14 | 10 |  |
| 29 | 32 | 28 | 17 | 20 | 21 | 13 | 0 | 7 | 22 | 23 | 17 | 14 | 22 | 11 | 22 | 23 | 18 | 20 | 15 | 15 | 27 | 27 | 23 |  |
| 36 | 39 | 35 | 24 | 27 | 28 | 20 | 7 | 0 | 29 | 30 | 24 | 21 | 29 | 18 | 23 | 22 | 25 | 27 | 22 | 22 | 34 | 28 | 24 |  |
| 7 | 10 | 12 | 11 | 18 | 17 | 35 | 22 | 29 | 0 | 1 | 10 | 8 | 16 | 20 | 32 | 33 | 10 | 13 | 7 | 9 | 21 | 27 | 33 |  |
| 8 | 11 | 11 | 12 | 19 | 18 | 36 | 23 | 30 | 1 | 0 | 9 | 9 | 17 | 21 | 33 | 34 | 11 | 12 | 8 | 10 | 22 | 28 | 34 |  |
| 17 | 20 | 17 | 6 | 13 | 12 | 30 | 17 | 24 | 10 | 9 | 0 | 3 | 11 | 17 | 27 | 28 | 7 | 3 | 4 | 4 | 16 | 22 | 28 |  |
| 15 | 18 | 14 | 3 | 10 | 9 | 27 | 14 | 21 | 8 | 9 | 3 | 0 | 8 | 14 | 24 | 25 | 4 | 6 | 1 | 1 | 13 | 19 | 25 |  |
| 23 | 25 | 18 | 7 | 2 | 1 | 23 | 22 | 29 | 16 | 17 | 11 | 8 | 0 | 11 | 16 | 17 | 12 | 14 | 9 | 7 | 5 | 11 | 17 |  |
| 27 | 30 | 28 | 17 | 9 | 10 | 24 | 11 | 18 | 20 | 21 | 17 | 14 | 11 | 0 | 27 | 28 | 16 | 20 | 13 | 15 | 16 | 22 | 28 |  |
| 39 | 41 | 34 | 23 | 18 | 17 | 9 | 22 | 23 | 32 | 33 | 27 | 24 | 16 | 27 | 0 | 1 | 28 | 30 | 25 | 23 | 11 | 5 | 1 |  |
| 40 | 42 | 35 | 24 | 19 | 18 | 10 | 23 | 22 | 33 | 34 | 28 | 25 | 17 | 28 | 1 | 0 | 29 | 31 | 26 | 24 | 12 | 6 | 2 |  |
| 17 | 20 | 18 | 7 | 14 | 13 | 31 | 18 | 25 | 10 | 11 | 7 | 4 | 12 | 16 | 28 | 29 | 0 | 10 | 3 | 5 | 17 | 23 | 29 |  |
| 20 | 23 | 20 | 9 | 16 | 15 | 33 | 20 | 27 | 13 | 12 | 3 | 6 | 14 | 20 | 30 | 31 | 10 | 0 | 7 | 7 | 19 | 25 | 31 |  |
| 14 | 17 | 15 | 4 | 11 | 10 | 28 | 15 | 22 | 7 | 8 | 4 | 1 | 9 | 13 | 25 | 26 | 3 | 7 | 0 | 2 | 14 | 20 | 26 |  |
| 16 | 19 | 15 | 4 | 9 | 8 | 28 | 15 | 22 | 9 | 10 | 4 | 1 | 7 | 15 | 23 | 24 | 5 | 7 | 2 | 0 | 12 | 18 | 24 |  |
| 28 | 30 | 23 | 12 | 7 | 6 | 20 | 27 | 34 | 21 | 22 | 16 | 13 | 5 | 16 | 11 | 12 | 17 | 19 | 14 | 12 | 0 | 6 | 12 |  |
| 34 | 36 | 29 | 18 | 13 | 12 | 14 | 27 | 28 | 27 | 28 | 22 | 19 | 11 | 22 | 5 | 6 | 23 | 25 | 20 | 18 | 6 | 0 | 6 |  |
| 40 | 42 | 35 | 24 | 19 | 18 | 10 | 23 | 24 | 33 | 34 | 28 | 25 | 17 | 28 | 1 | 2 | 29 | 31 | 26 | 24 | 12 | 6 | 0 |  |

Сложность алгоритма составляет O(n)=n3, так как алгоритм основан на 3 вложенных циклах вычислений.

Для уменьшения сложности алгоритма была применена сортировка данных с помощью рекурсионного алгоритма быстрой сортировки. Уменьшения сложности не произошло, что вполне объяснимо тем фактом, что задача решается полным перебором. Тем не менее, при замере количества вложенных действий, удалось отметить некоторое снижение сложности за счет сортировки благодаря пропуску части итераций цикла 2-го уровня вложенности. То есть для оптимизации алгоритма сортировка не требуется и не имеет никакого эффекта, так как алгоритм Флойда-Уоршелла решается полным перебором элементов матрицы графа.

**import random  
from collections import OrderedDict  
  
  
# merge\_dicts  
def merge\_dicts(d\_list):  
 if len(d\_list)==0:return False  
 if len(d\_list)==1:return d\_list[0]  
 for d in (range(1,len(d\_list))):  
 for k in d\_list[d]:  
 d\_list[0][k]=d\_list[d][k]  
 return d\_list[0]  
  
# Быстрая сортировка словаря  
def quickSort\_rec(g\_dict):  
 if len(g\_dict) <= 1:  
 return g\_dict  
  
 keys=list(g\_dict.keys())  
 q = random.choice(keys)  
  
 l\_nums = OrderedDict([(k,g\_dict[k]) for k in keys if g\_dict[k] < g\_dict[q]])  
 b\_nums = OrderedDict([(k,g\_dict[k]) for k in keys if g\_dict[k] > g\_dict[q]])  
  
  
 e\_nums = OrderedDict([(k,g\_dict[k]) for k in keys if g\_dict[k] == g\_dict[q]])#  
  
 return merge\_dicts([quickSort\_rec(l\_nums), e\_nums , quickSort\_rec(b\_nums)])  
  
  
def floydwarshall(graph, alg=False):  
 # Заполняем результирующих граы  
 # Добавляем бесконечный путь там, где о не обозначен  
 # no edge, и еще 0 поглавной диагонали (т.к. отрабатываем без петель)  
 dist = {}  
 pred = {}  
 count=0  
  
  
 for u in graph:  
 dist[u] = {}  
 pred[u] = {}  
 for v in graph:  
 dist[u][v] = float("inf")  
 pred[u][v] = -1  
 dist[u][u] = 0  
 for neighbor in graph[u]:  
 dist[u][neighbor] = graph[u][neighbor]  
 pred[u][neighbor] = u  
  
 #Сортировка графа  
 for i in dist:  
 dist[i]=quickSort\_rec(dist[i])**

**for t in graph:  
 for u in graph:  
 if (alg) and (dist[u][t] == float('inf')):  
 pass  
 continue  
 dist[u] = quickSort\_rec(dist[u])  
  
 for v in graph:  
 count=count+1  
 newdist = dist[u][t] + dist[t][v]  
 if newdist < dist[u][v]:  
 dist[u][v] = newdist  
 pred[u][v] = pred[t][v]  
 elif newdist > dist[u][v]:  
 pass  
  
  
 elif newdist > dist[u][v] :  
 pass  
 # break  
  
 print(f"счетчик проходов (с сортировкой = {alg}) {count}")  
 return dist, pred  
  
matrx= [ #матрица смежности  
 [0, 3, 0, 0, 42, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],  
 [3, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],  
 [0, 7, 0, 11, 0, 0, 0, 0, 0, 0, 11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 309, 0, 0, ],  
 [0, 0, 11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],  
 [42, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],  
 [0, 0, 0, 0, 1, 0, 22, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],  
 [0, 0, 0, 0, 0, 22, 0, 13, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, ],  
 [0, 0, 0, 0, 0, 0, 13, 0, 7, 0, 0, 0, 14, 0, 11, 25, 0, 0, 0, 0, 0, 0, 0, 0, ],  
 [0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 22, 0, 0, 0, 0, 0, 0, 0, ],  
 [7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, ],  
 [0, 0, 11, 0, 0, 0, 0, 0, 0, 1, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 3, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, ],  
 [0, 0, 0, 3, 0, 0, 0, 14, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, ],  
 [0, 0, 0, 7, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 5, 0, 0, ],  
 [0, 0, 0, 0, 9, 0, 0, 11, 0, 0, 0, 0, 0, 0, 0, 0, 42, 0, 0, 13, 0, 0, 0, 0, ],  
 [0, 0, 0, 0, 0, 0, 9, 25, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 5, 1, ],  
 [0, 0, 0, 0, 0, 0, 0, 0, 22, 0, 0, 0, 0, 0, 42, 1, 0, 0, 0, 0, 0, 0, 0, 9, ],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, ],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 1, 0, 13, 0, 0, 3, 0, 0, 9, 0, 0, 0, ],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 7, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, ],  
 [0, 0, 309, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, ],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 6, 0, 13, ],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 9, 0, 0, 0, 0, 0, 13, 0, ],  
  
]  
  
graph={}  
for idx, row in enumerate(matrx):  
 graph[idx]={}  
 for idx2, vertex in enumerate(row):  
 if vertex>0:  
 graph[idx][idx2]=vertex  
  
  
dist, pred = floydwarshall(graph,True)  
dist2, pred2 = floydwarshall(graph)  
print ( "Матрица минимальных растояний (c сортировкой): ")  
for v in dist:  
 dist[v]= [(key,dist[v][key]) for key in dist[v].keys()]  
 print (" %s: %s " % (v, dist[v]))  
  
  
print ( "Предыдущие точки (запоминание маршрута): ")  
  
for v in pred:  
 pred[v]= [[key,pred[v][key]] for key in pred[v].keys()]  
 print (" %s: %s " % (v, pred[v]))**