САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМЕНИ ПРОФЕССОРА М.А БОНЧ-БРУЕВИЧА

**Курсовая работа**

**Математические модели в сетях связи (ММвСС)**

На тему:

**«Расчет пропускной способности линий связи»**

Вариант 8527

Факультет: ИКСС

Специальность: Программная инженерия

Выполнил: Чепуренко В.А. ИКПИ-85.

Преподаватель: Лобастова М.В.

Санкт-Петербург, 2019

**Содержание**

[Цель работы 3](#_Toc532461870)

[Параметры сети 3](#_Toc532461871)

[Требования к качеству обслуживания 3](#_Toc532461872)

[Значения исходных данных 4](#_Toc532461873)

[Матрица интенсивности трафика в направлениях 4](#_Toc532461874)

[Матрица кратчайших маршрутов между вершинами графа 5](#_Toc532461875)

[Матрица интенсивностей нагрузок на линии связи 5](#_Toc532461876)

[Матрица потоков 6](#_Toc532461877)

[Интенсивность трафика ПД в линиях связи 6](#_Toc532461878)

[Пропускная способность линий связи 7](#_Toc532461879)

[Оптимизация пропускной способности линии связи 7](#_Toc532461880)  
 [Выводы 9](#_Toc532461879)

[Список литературы](#_Toc532461880) 9

[Приложение: код](#_Toc532461880) 9

**Цель работы**

Применение знаний в области моделирования сетей связи с целью расчета параметров сетевых элементов. Целью выполнения задания является расчет требуемых пропускных способностей линий связи в сети с заданными структурными параметрами и требованиями к качеству обслуживания.

**Параметры сети:**

* количество узлов в сети связи *n* (20),
* интенсивность удельной абонентской нагрузки *y*0 (0,1 Эрл),
* тип кодека (G.711),
* скорость потока для данного типа кодека *a*o (85,6 Кбит/с),
* длина пакета *L* (200 байт),
* количество абонентов в узлах связи (Распределение абонентов по узлам сети - индивидуально),
* матрица расстояний между узлами связи (Матрица расстояний – индивидуальна).

**Требования к качеству обслуживания:**

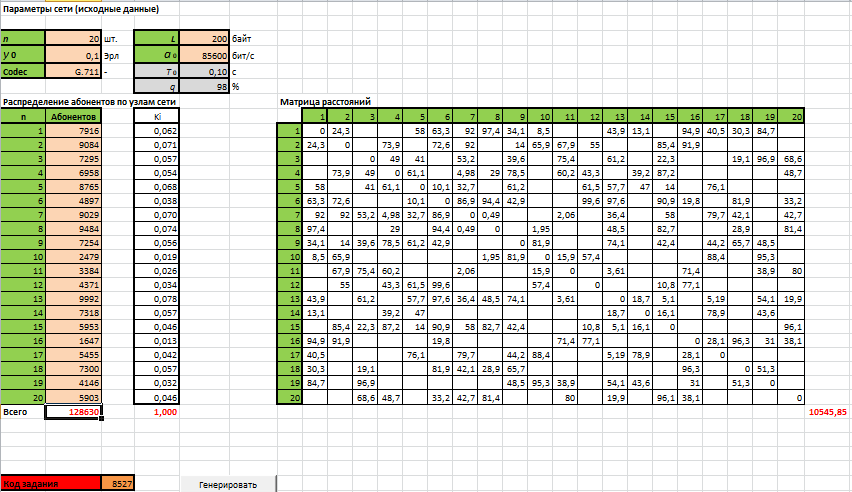
* начальное требование к величине задержки *T*о (100 мс=0,1с),
* доля вызовов, обслуженных с гарантированным качеством *q* (98%). (Данные могу быть получены многократно.)
* Структура сети задана неориентированным графом. Вершинами графа являются узлы сети, а ребрами линии связи.
* Граф задан матрицей (матрицей расстояний между вершинами).   
  D = [dij]; i,j = 1..n
* *n* – количество узлов сети.  
  Узлы сети заданы вершинами графа *n*1…*n*20.  
  Каждый из узлов сети может производить «*собственный*» трафик и может выполнять функции маршрутизации трафика других узлов сети.
* «Собственный» трафик узла связи производят абоненты, подключенные к этому узлу. Количество абонентов определено индивидуальным заданием.
* Линии связи в сети являются *двунаправленными* (т.е. если скорость передачи равна *b* Мбит/с, то данные по ней могут передаваться независимо в двух направлениях со скоростями *b* Мбит/с).
* Трафик между узлами сети распределен *пропорционально их абонентской емкости*.
* Маршрутизация трафика производится по *кратчайшему маршруту*, при этом кратчайшим маршрутом является маршрут с *минимальной длиной*, вычисленной согласно заданной *матрице расстояний* между узлами.
* Все абоненты сети используют только услугу IP телефонии (VoIP).
* Удельная интенсивность абонентской нагрузки определена заданием (0,1 Эрл).
* Число абонентов, включенных в узлы, задается индивидуально (Распределение абонентов по узлам сети).

**Задание**

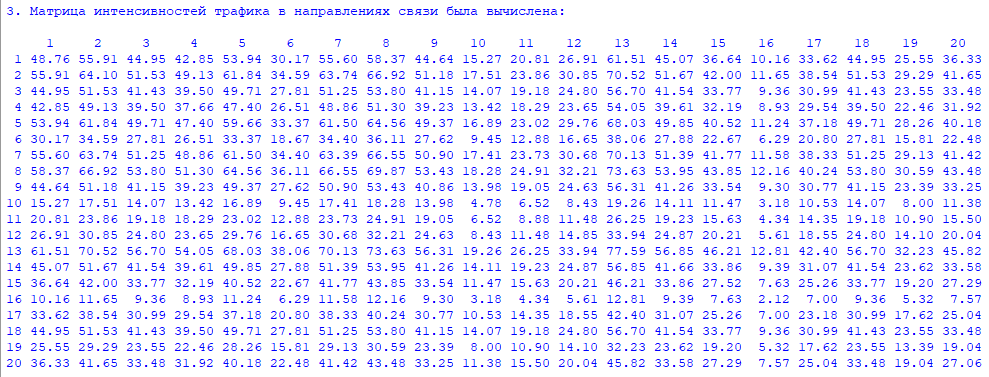
1. Требуется выполнить расчет величины пропускной способности линий связи, для обеспечения нормы качества обслуживания (q и T0).

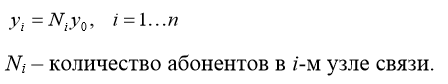
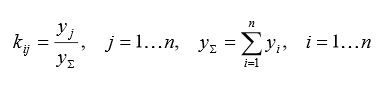
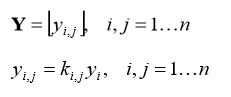
2. Оптимизировать величины пропускной способности линий связи в сети, при целевом значении «сквозной» задержки абонент-абонент T0/2=50 мс.

**Значения исходных данных**



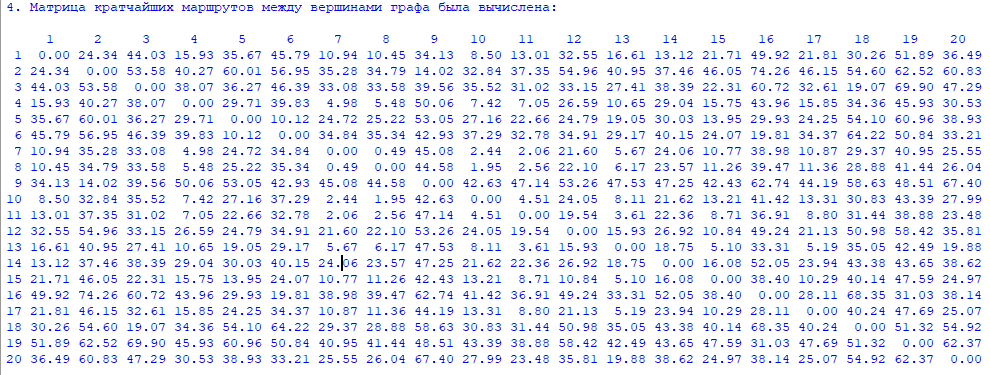
# 1. Матрица интенсивности трафика в направлениях связи (Y)



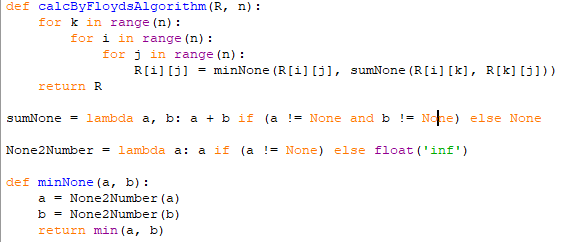
  

**2. Матрица кратчайших маршрутов между вершинами графа (R)**

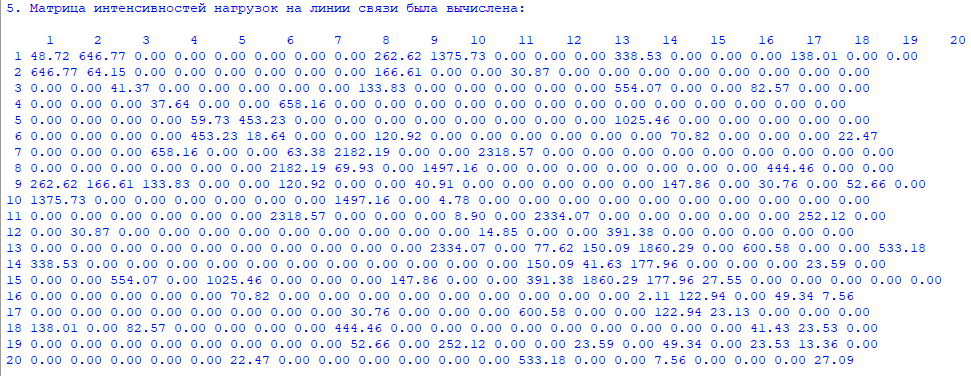
На основе матрицы расстояний между вершинами графа, используя алгоритм Флойда.  
Данная матрица содержит информацию о кротчайшем пути из вершины i в вершину j.



Реализация алгоритма Флойда:



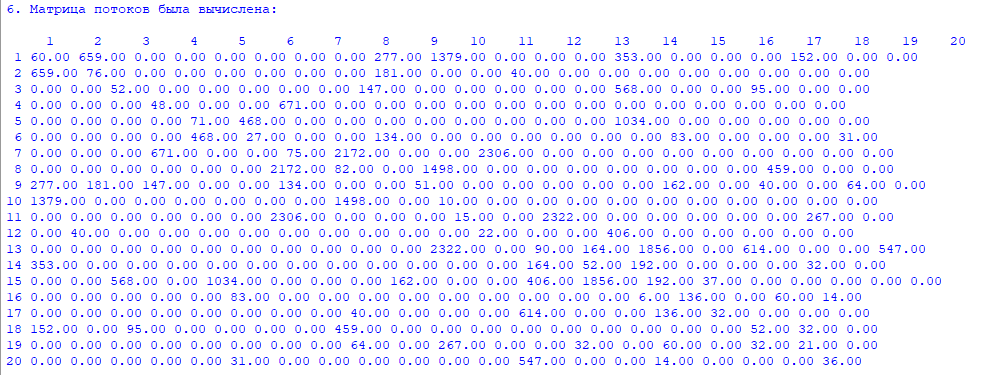
**3. Матрица интенсивностей нагрузок на линии связи (~Y)**

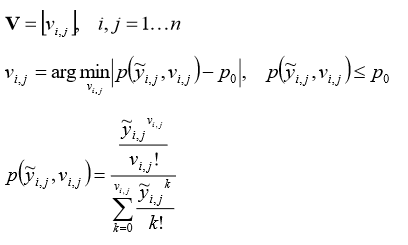


При помощи матрицы R найдем матрицу ~Y [~yi j]; i, j = 1..n

Если i=j, то ~Yi j = Yi j; есть i не = j, то ~Yi j = 0.

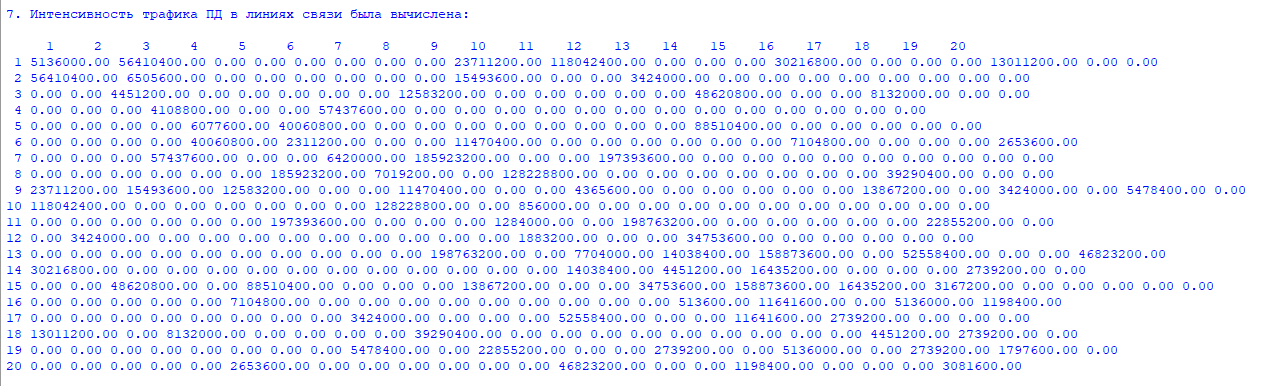
**4. Матрица потоков (V)**

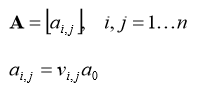




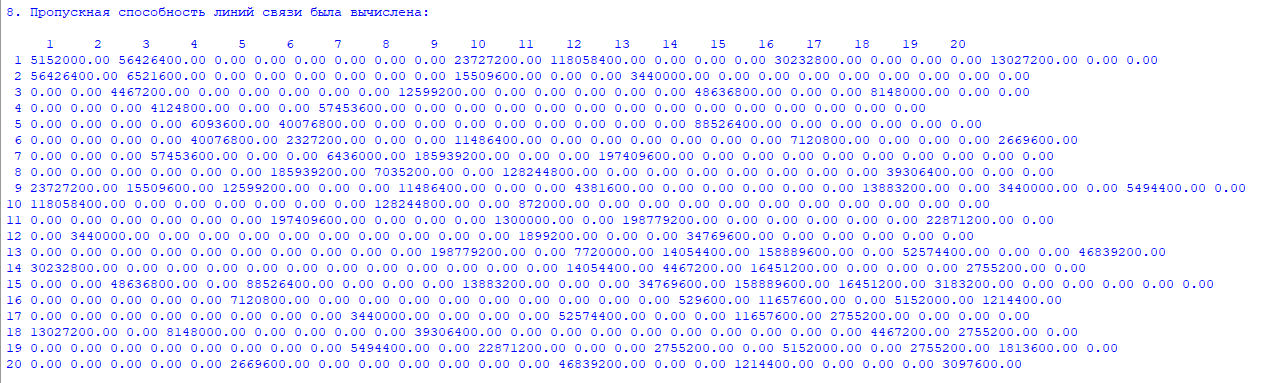
На основе матрицы интенсивностей нагрузок на линии связи Y и требований к качеству обслуживания p0, найдем матрицу потоков V = [vi, j]; i, j =1..n; p0=1-q/100, где q 98 %, т.е p0 = 1-0,98=0,02

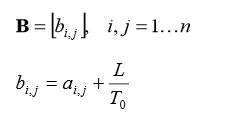
**5. Интенсивность трафика ПД в линиях связи (А)**



На основе матрицы потоков V и данных о типе кодека (скорости одного потока) вычислим матрицу. Для кодека G.711 примем a0=85,6 Кбит/c =85600Бит/с;  


**6. Пропускная способность линий связи (В)**



На основе матрицы интенсивности трафика ПД A и данных и требований к величине задержки To, вычислим матрицу следующей формулой:   
L = 200Байт \* 8 = 1600Бит;  
To =0,1;  
  
**Оптимизация пропускной способности линий связи**

Основная задача состоит в приближении значения к Topt =To/2=0,05  
Bo=B;

dc=10000;

mo= ∞;

Oprev = ∞;

**НАЧАЛО ЦИКЛА**

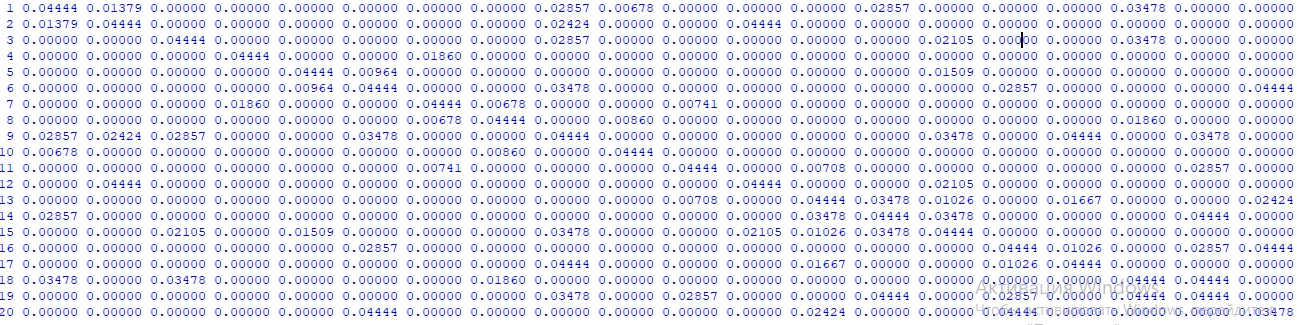
Oprev = mo;

for (i( 1..20 ))  
 for (j( 1..20 ))  
 for (x( 1..20 ))  
 for (y( 1..20 ))  
 Считаем задержки для каждого:  
 Bm=Bxy;  
 Если x,y = i,j , то Вm +=dc;  
 DELxy = L/( bm – axy);  
Далее находим значения матрицы задержек (DLxy) из DEL на маршруте из x в y путем суммирования задержек;  
Затем находим Oij , для этого суммируем элементы из таблицы DL:  
 for (x( 1..20 ))  
 for (y( 1..20 ))  
 Oij+ = (DLij - Topt)^2  
После – находим минимальное значение из таблицы O:

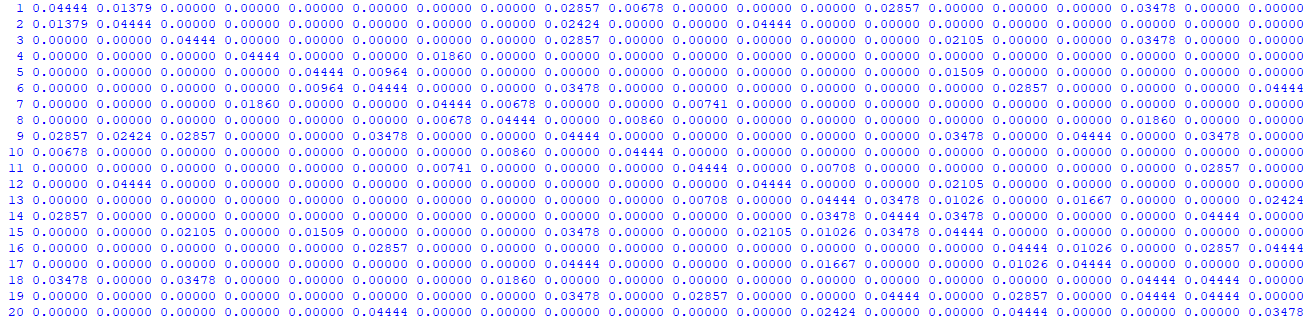
m0 = min {O};  
 B0mimj += dc

Если m0 < Oprev, то переходим к **НАЧАЛУ ЦИКЛА**

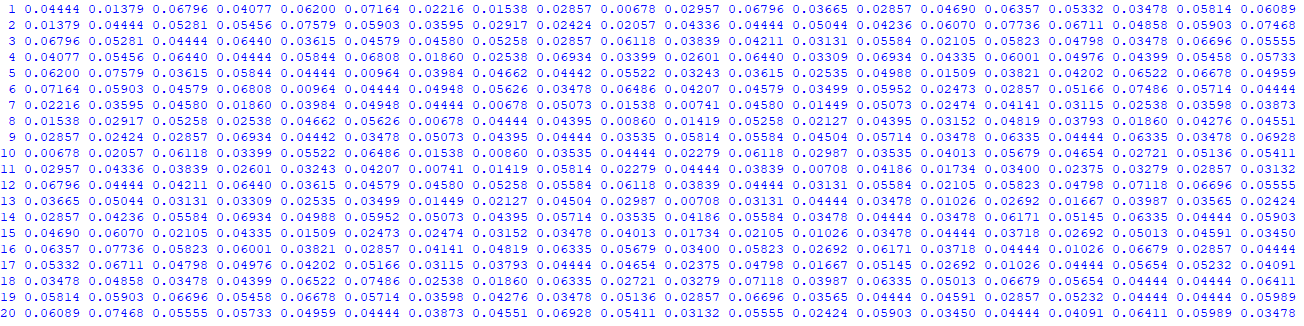
**7. Bo**



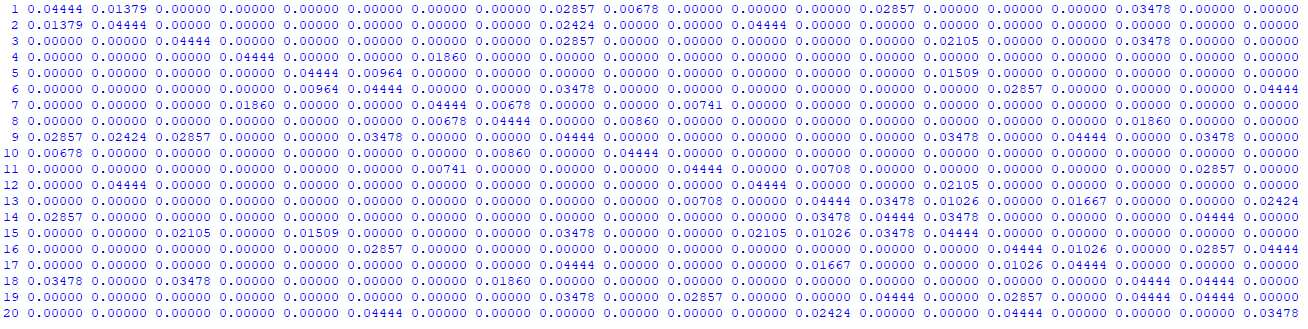
**8. DEL**

****

**9. DL**



**10. O**

****

# Выводы

В данной курсовой работе были рассчитаны требуемые пропускные способности линий связи в сети с заданными структурными параметрами и требованиями к качеству обслуживания.

Вычислив необходимые параметры сети, появляется возможность оптимизации её пропускной способности, для обслуживания клиентов с задержкой равной T0/2.

# Список литературы

1. 3.8.1 Documentation [Электронный ресурс] /. — Электрон. журн. — Режим доступа: https://docs.python.org/3/, свободный. — Документация по языку программирования Python 3.

**Приложение: код**

**MAIN.PY:**

import calculation as C

import optimization as O

import output as OUT

def main():

# --- Входные данные:

# Матрица расстояний

# None - нет пути

D = [[0, 24.33963418, None, None, 57.96404481, 63.28330636, 92.04159379, 97.41273522, 34.13305879, 8.501392603, None, None, 43.92023683, 13.11889291, None, 94.92898583, 40.45057893, 30.26080728, 84.720999, None],

[24.33963418, 0, None, 73.85107875, None, 72.56256938, 92.0316875, None, 14.01893497, 65.85347056, 67.89705157, 54.96413112, None, None, 85.38839221, 91.94399714, None, None, None, None],

[None, None, 0, 49.04459119, 40.96282125, None, 53.19741368, None, 39.55920339, None, 75.43881536, None, 61.22687459, None, 22.31400609, None, None, 19.07089353, 96.94798589, 68.55219007],

[None, 73.85107875, 49.04459119, 0, 61.08250022, None, 4.982894659, 29.04109359, 78.52806449, None, 60.23363471, 43.33114028, None, 39.18160796, 87.18847632, None, None, None, None, 48.74228835],

[57.96404481, None, 40.96282125, 61.08250022, 0, 10.12077928, 32.69773126, None, 61.20582223, None, None, 61.52334809, 57.69343972, 47.02911973, 13.95421624, None, 76.08047128, None, None, None],

[63.28330636, 72.56256938, None, None, 10.12077928, 0, 86.94083095, 94.44400668, 42.9277122, None, None, 99.59403872, 97.64439464, None, 90.90544581, 19.80754733, None, 81.85216784, None, 33.2095325],

[92.04159379, 92.0316875, 53.19741368, 4.982894659, 32.69773126, 86.94083095, 0, 0.494223833, None, None, 2.063733339, None, 36.42662168, None, 58.00264478, None, 79.68242764, 42.0504868, None, 42.74333119],

[97.41273522, None, None, 29.04109359, None, 94.44400668, 0.494223833, 0, None, 1.947158575, None, None, 48.47936034, None, 82.70263076, None, None, 28.87930274, None, 81.40396476],

[34.13305879, 14.01893497, 39.55920339, 78.52806449, 61.20582223, 42.9277122, None, None, 0, 81.9386065, None, None, 74.06826615, None, 42.4292624, None, 44.19479966, 65.73415399, 48.50550294, None],

[8.501392603, 65.85347056, None, None, None, None, None, 1.947158575, 81.9386065, 0, 15.85895419, 57.43348002, None, None, None, None, 88.36366534, None, 95.32405734, None],

[None, 67.89705157, 75.43881536, 60.23363471, None, None, 2.063733339, None, None, 15.85895419, 0, None, 3.607875109, None, None, 71.37306333, None, None, 38.88342977, 79.96090055],

[None, 54.96413112, None, 43.33114028, 61.52334809, 99.59403872, None, None, None, 57.43348002, None, 0, None, None, 10.83549857, 77.0598352, None, None, None, None],

[43.92023683, None, 61.22687459, None, 57.69343972, 97.64439464, 36.42662168, 48.47936034, 74.06826615, None, 3.607875109, None, 0, 18.74819398, 5.097728968, None, 5.193954706, None, 54.12440896, 19.87610459],

[13.11889291, None, None, 39.18160796, 47.02911973, None, None, None, None, None, None, None, 18.74819398, 0, 16.07987285, None, 78.85996699, None, 43.64811778, None],

[None, 85.38839221, 22.31400609, 87.18847632, 13.95421624, 90.90544581, 58.00264478, 82.70263076, 42.4292624, None, None, 10.83549857, 5.097728968, 16.07987285, 0, None, None, None, None, 96.07403874],

[94.92898583, 91.94399714, None, None, None, 19.80754733, None, None, None, None, 71.37306333, 77.0598352, None, None, None, 0, 28.11259627, 96.32515311, 31.03093505, 38.13814521],

[40.45057893, None, None, None, 76.08047128, None, 79.68242764, None, 44.19479966, 88.36366534, None, None, 5.193954706, 78.85996699, None, 28.11259627, 0, None, None, None],

[30.26080728, None, 19.07089353, None, None, 81.85216784, 42.0504868, 28.87930274, 65.73415399, None, None, None, None, None, None, 96.32515311, None, 0, 51.32429004, None],

[84.720999, None, 96.94798589, None, None, None, None, None, 48.50550294, 95.32405734, 38.88342977, None, 54.12440896, 43.64811778, None, 31.03093505, None, 51.32429004, 0, None],

[None, None, 68.55219007, 48.74228835, None, 33.2095325, 42.74333119, 81.40396476, None, None, 79.96090055, None, 19.87610459, None, 96.07403874, 38.13814521, None, None, None, 0]]

# Захадкодить матрицу смежностей

S = [[48.71573972, 55.90371142, 44.89405271, 42.82012594, 53.94055819, 30.13655601, 55.56523673, 58.36534556, 44.64173521, 15.25597761, 20.82542486, 26.89950711, 61.49162093, 45.03559667, 36.63527015, 10.13577859, 33.57053564, 44.92482314, 25.51483791, 36.32756589],

[55.90371142, 64.15226308, 51.51813729, 49.13820415, 61.89944803, 34.58318277, 63.76384669, 66.97710954, 51.22859053, 17.50698593, 23.89820104, 30.86850968, 70.56466454, 51.68056596, 42.04077742, 11.63130529, 38.52384358, 51.55344787, 29.27953355, 41.68767162],

[44.89405271, 51.51813729, 41.37217212, 39.46094224, 49.7089909, 27.77238203, 51.2062155, 53.78665941, 41.1396486, 14.05916582, 19.19169712, 24.78927544, 56.6676825, 41.50261214, 33.76128042, 9.34063982, 30.93697038, 41.40052865, 23.51323175, 33.47771515],

[42.82012594, 49.13820415, 39.46094224, 37.63800358, 47.41263313, 26.48940838, 48.84069191, 51.30192957, 39.23916038, 13.40968825, 18.305117, 23.64410946, 54.04986084, 39.58535645, 32.20164347, 8.909139392, 29.50780533, 39.48798881, 22.42701392, 31.9311778],

[53.94055819, 61.89944803, 49.7089909, 47.41263313, 59.72574438, 33.36873591, 61.52467154, 64.62509523, 49.42961207, 16.89219855, 23.05897536, 29.78450983, 68.08666718, 49.86571562, 40.56444453, 11.22285237, 37.17101376, 49.74306149, 28.25133328, 40.22373863],

[30.13655601, 34.58318277, 27.77238203, 26.48940838, 33.36873591, 18.64309181, 34.37379538, 36.10600016, 27.61629324, 9.437660732, 12.88303506, 16.64058695, 38.03997823, 27.85994403, 22.66332971, 6.270200575, 20.76742206, 27.79141724, 15.78400218, 22.47297753],

[55.56523673, 63.76384669, 51.2062155, 48.84069191, 61.52467154, 34.37379538, 63.37778201, 66.57158983, 50.91842183, 17.40098811, 23.75350696, 30.68161315, 70.13742362, 51.36766073, 41.78623727, 11.56088238, 38.29059706, 51.24131229, 29.10225764, 41.43526938],

[58.36534556, 66.97710954, 53.78665941, 51.30192957, 64.62509523, 36.10600016, 66.57158983, 69.92634378, 53.4843629, 18.27787919, 24.95052165, 32.22775713, 73.67187126, 53.95624038, 43.89197854, 12.14347197, 40.22018192, 53.82352484, 30.56881287, 43.52332426],

[44.64173521, 51.22859053, 41.1396486, 39.23916038, 49.42961207, 27.61629324, 50.91842183, 53.4843629, 40.90843194, 13.98014927, 19.08383425, 24.64995258, 56.34919381, 41.26935552, 33.5715323, 9.288142735, 30.7630957, 41.16784576, 23.38108062, 33.28956076],

[15.25597761, 17.50698593, 14.05916582, 13.40968825, 16.89219855, 9.437660732, 17.40098811, 18.27787919, 13.98014927, 4.777610977, 6.521756977, 8.423936096, 19.25691363, 14.10349219, 11.47281894, 3.174152997, 10.51305683, 14.06880199, 7.990308637, 11.37645728],

[20.82542486, 23.89820104, 19.19169712, 18.305117, 23.05897536, 12.88303506, 23.75350696, 24.95052165, 19.08383425, 6.521756977, 8.902632356, 11.49923346, 26.28696883, 19.25220555, 15.66116147, 4.33293011, 14.35102231, 19.20485112, 10.90730312, 15.52962139],

[26.89950711, 30.86850968, 24.78927544, 23.64410946, 29.78450983, 16.64058695, 30.68161315, 32.22775713, 24.64995258, 8.423936096, 11.49923346, 14.85317655, 33.9540014, 24.86743217, 20.22900023, 5.596701392, 18.53673715, 24.80626603, 14.08859986, 20.0590943],

[61.49162093, 70.56466454, 56.6676825, 54.04986084, 68.08666718, 38.03997823, 70.13742362, 73.67187126, 56.34919381, 19.25691363, 26.28696883, 33.9540014, 77.61802379, 56.84634689, 46.24300396, 12.79392366, 42.3745316, 56.70652258, 32.20619762, 45.85460313],

[45.03559667, 51.68056596, 41.50261214, 39.58535645, 49.86571562, 27.85994403, 51.36766073, 53.95624038, 41.26935552, 14.10349219, 19.25220555, 24.86743217, 56.84634689, 41.63346342, 33.86772448, 9.370089404, 31.03450983, 41.53105807, 23.58736531, 33.58326518],

[36.63527015, 42.04077742, 33.76128042, 32.20164347, 40.56444453, 22.66332971, 41.78623727, 43.89197854, 33.5715323, 11.47281894, 15.66116147, 20.22900023, 46.24300396, 33.86772448, 27.55050066, 7.62232061, 25.24575527, 33.78442043, 19.1876996, 27.31910052],

[10.13577859, 11.63130529, 9.34063982, 8.909139392, 11.22285237, 6.270200575, 11.56088238, 12.14347197, 9.288142735, 3.174152997, 4.33293011, 5.596701392, 12.79392366, 9.370089404, 7.62232061, 2.108846303, 6.984673093, 9.347041903, 5.308607634, 7.558299775],

[33.57053564, 38.52384358, 30.93697038, 29.50780533, 37.17101376, 20.76742206, 38.29059706, 40.22018192, 30.7630957, 10.51305683, 14.35102231, 18.53673715, 42.3745316, 31.03450983, 25.24575527, 6.984673093, 23.13381404, 30.95817461, 17.58254684, 25.03371298],

[44.92482314, 51.55344787, 41.40052865, 39.48798881, 49.74306149, 27.79141724, 51.24131229, 53.82352484, 41.16784576, 14.06880199, 19.20485112, 24.80626603, 56.70652258, 41.53105807, 33.78442043, 9.347041903, 30.95817461, 41.42890461, 23.52934774, 33.50066081],

[25.51483791, 29.27953355, 23.51323175, 22.42701392, 28.25133328, 15.78400218, 29.10225764, 30.56881287, 23.38108062, 7.990308637, 10.90730312, 14.08859986, 32.20619762, 23.58736531, 19.1876996, 5.308607634, 17.58254684, 23.52934774, 13.36338024, 19.02653969],

[36.32756589, 41.68767162, 33.47771515, 31.9311778, 40.22373863, 22.47297753, 41.43526938, 43.52332426, 33.28956076, 11.37645728, 15.52962139, 20.0590943, 45.85460313, 33.58326518, 27.31910052, 7.558299775, 25.03371298, 33.50066081, 19.02653969, 27.08964394]]

# Матрица путей, считается в C++

P = [[1, 1, 15, 7, 15, 5, 8, 10, 1, 1, 7, 15, 11, 1, 13, 17, 13, 1, 11, 13],

[2, 2, 9, 7, 15, 9, 8, 10, 2, 1, 7, 2, 11, 1, 13, 17, 13, 1, 9, 13],

[10, 9, 3, 7, 15, 5, 11, 7, 3, 8, 13, 15, 15, 15, 3, 17, 13, 3, 11, 13],

[10, 1, 15, 4, 15, 5, 4, 7, 1, 8, 7, 15, 11, 1, 13, 17, 13, 8, 11, 13],

[10, 1, 15, 7, 5, 5, 11, 7, 6, 8, 13, 15, 15, 15, 5, 6, 13, 8, 16, 13],

[10, 9, 15, 7, 6, 6, 11, 7, 6, 8, 13, 15, 15, 15, 5, 6, 13, 8, 16, 6],

[10, 1, 15, 7, 15, 5, 7, 7, 1, 8, 7, 15, 11, 1, 13, 17, 13, 8, 11, 13],

[10, 1, 15, 7, 15, 5, 8, 8, 1, 8, 7, 15, 11, 1, 13, 17, 13, 8, 11, 13],

[9, 9, 9, 7, 6, 9, 8, 10, 9, 1, 7, 15, 15, 1, 9, 6, 9, 3, 9, 13],

[10, 1, 15, 7, 15, 5, 8, 10, 1, 10, 7, 15, 11, 1, 13, 17, 13, 8, 11, 13],

[10, 1, 15, 7, 15, 5, 11, 7, 1, 8, 11, 15, 11, 13, 13, 17, 13, 8, 11, 13],

[10, 12, 15, 7, 15, 5, 11, 7, 15, 8, 13, 12, 15, 15, 12, 17, 13, 8, 11, 13],

[10, 1, 15, 7, 15, 5, 11, 7, 15, 8, 13, 15, 13, 13, 13, 17, 13, 8, 11, 13],

[14, 1, 15, 7, 15, 5, 8, 10, 1, 1, 13, 15, 14, 14, 14, 17, 13, 1, 14, 13],

[10, 1, 15, 7, 15, 5, 11, 7, 15, 8, 13, 15, 15, 15, 15, 17, 13, 8, 11, 13],

[10, 1, 15, 7, 6, 16, 11, 7, 6, 8, 13, 15, 17, 13, 13, 16, 16, 8, 16, 16],

[10, 1, 15, 7, 15, 5, 11, 7, 17, 8, 13, 15, 17, 13, 13, 17, 17, 8, 11, 13],

[18, 1, 18, 7, 15, 5, 8, 18, 3, 8, 7, 15, 11, 1, 13, 17, 13, 18, 18, 13],

[10, 9, 15, 7, 6, 16, 11, 7, 19, 8, 19, 15, 11, 19, 13, 19, 13, 19, 19, 13],

[10, 1, 15, 7, 15, 20, 11, 7, 15, 8, 13, 15, 20, 13, 13, 20, 13, 8, 11, 20]]

# Смежности почему то Y

# В восстановление путей передавать ее и D

# Проверить на точность переноса алгоритма дейкстры

# Матрицу посчитать в екселе

# Количество узлов связи

n = 20

# Длина пакета

L = 200 # байт

# Codec

Codec = 'G.711'

# Интенсивность удельной абонентской нагрузки

y0 = 0.1 # Эрл

# --- Требования к качеству обслуживания:

# Начальное требование к величине задержки T0

T0 = 0.1 # 100ms = 0.1s

# Доля вызовов, обслуженных с гарантированным качеством

q = 98 # 98%

# Распределение абонентов по узлам связи (количество)

N = [7916, 9084, 7295, 6958, 8765,

4897, 9029, 9484, 7254, 2479,

3384, 4371, 9992, 7318, 5953,

1647, 5455, 7300, 4146, 5903]

# --- Рассчеты:

y = C.calcTrafficNodeIntensity(N, y0, n)

print("1. Интенсивность исходящего трафика от каждого из узлов сети была вычислена.")

OUT.printArr(y)

k = C.calcTrafficRatio(y, n)

print("2. Коэффициенты распределения трафика по направлениям связи были вычислены.")

OUT.printMatrix(k, n)

Y = C.calcTrafficMatrixIntensity(k, y, n)

print("3. Матрица интенсивностей трафика в направлениях связи была вычислена:")

OUT.printMatrix(Y, n)

R = C.calcByFloydsAlgorithm(D, n)

print("4. Матрица кратчайших маршрутов между вершинами графа была вычислена:")

OUT.printMatrix(D, n)

\_next = C.calcPath(P, S, n)

OUT.printMatrix(\_next, n)

TNEXT = [list(x) for x in zip(\*\_next)]

OUT.printMatrix(TNEXT, n)

# Матрица интенсивностей нагрузок на линии связи:

print("5. Матрица интенсивностей нагрузок на линии связи была вычислена:")

Ytilda = [[48.72, 646.77, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 262.62, 1375.73, 0.00, 0.00, 0.00, 338.53, 0.00, 0.00, 0.00, 138.01, 0.00, 0.00],

[646.77, 64.15, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 166.61, 0.00, 0.00, 30.87, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00],

[0.00, 0.00, 41.37, 0.00, 0.00, 0.00, 0.00, 0.00, 133.83, 0.00, 0.00, 0.00, 0.00, 0.00, 554.07, 0.00, 0.00, 82.57, 0.00, 0.00],

[0.00, 0.00, 0.00, 37.64, 0.00, 0.00, 658.16, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00],

[0.00, 0.00, 0.00, 0.00, 59.73, 453.23, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1025.46, 0.00, 0.00, 0.00, 0.00, 0.00],

[0.00, 0.00, 0.00, 0.00, 453.23, 18.64, 0.00, 0.00, 120.92, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 70.82, 0.00, 0.00, 0.00, 22.47],

[0.00, 0.00, 0.00, 658.16, 0.00, 0.00, 63.38, 2182.19, 0.00, 0.00, 2318.57, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00],

[0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2182.19, 69.93, 0.00, 1497.16, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 444.46, 0.00, 0.00],

[262.62, 166.61, 133.83, 0.00, 0.00, 120.92, 0.00, 0.00, 40.91, 0.00, 0.00, 0.00, 0.00, 0.00, 147.86, 0.00, 30.76, 0.00, 52.66, 0.00],

[1375.73, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1497.16, 0.00, 4.78, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00],

[0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2318.57, 0.00, 0.00, 0.00, 8.90, 0.00, 2334.07, 0.00, 0.00, 0.00, 0.00, 0.00, 252.12, 0.00],

[0.00, 30.87, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 14.85, 0.00, 0.00, 391.38, 0.00, 0.00, 0.00, 0.00, 0.00],

[0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2334.07, 0.00, 77.62, 150.09, 1860.29, 0.00, 600.58, 0.00, 0.00, 533.18],

[338.53, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 150.09, 41.63, 177.96, 0.00, 0.00, 0.00, 23.59, 0.00],

[0.00, 0.00, 554.07, 0.00, 1025.46, 0.00, 0.00, 0.00, 147.86, 0.00, 0.00, 391.38, 1860.29, 177.96, 27.55, 0.00, 0.00, 0.00, 0.00, 0.00],

[0.00, 0.00, 0.00, 0.00, 0.00, 70.82, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.11, 122.94, 0.00, 49.34, 7.56],

[0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 30.76, 0.00, 0.00, 0.00, 600.58, 0.00, 0.00, 122.94, 23.13, 0.00, 0.00, 0.00],

[138.01, 0.00, 82.57, 0.00, 0.00, 0.00, 0.00, 444.46, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 41.43, 23.53, 0.00],

[0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 52.66, 0.00, 252.12, 0.00, 0.00, 23.59, 0.00, 49.34, 0.00, 23.53, 13.36, 0.00],

[0.00, 0.00, 0.00, 0.00, 0.00, 22.47, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 533.18, 0.00, 0.00, 7.56, 0.00, 0.00, 0.00, 27.09]]

OUT.printMatrix(Ytilda, n)

# Матрица потоков:

print("6. Матрица потоков была вычислена:")

V = C.calcStreamMatrix(Ytilda, q, n)

OUT.printMatrix(V, n)

# Интенсивность трафика ПД в линиях связи:

print("7. Интенсивность трафика ПД в линиях связи была вычислена:")

A = C.calcTrafficLineIntensity(V, Codec, n)

OUT.printMatrix(A, n)

# Пропускная способность линий связи:

print("8. Пропускная способность линий связи была вычислена:")

B = C.calcLinesCapacity(A, L, T0, n)

OUT.printMatrix(B, n)

summa = 0

for i in range(n):

for j in range(n):

summa += B[i][j]

print("Сумма: " + str(summa))

# --- Оптимизация:

print("9. Оптимизация пропускной способности линий связи:")

O.optimization(B, A, L, TNEXT, n)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**CALCULATION.PY:**

import math

# 1. Интенсивность исходящего трафика от каждого из узлов сети:

# y[i]=N[i]\*y[0], i = 1..n

def calcTrafficNodeIntensity(N, y0, n):

y = []

for i in range(n):

y.append(round(N[i] \* y0))

return y

# 2. Коэффициенты распределения трафика по направлениям связи:

# k[i][j] = y[i]/sum(y), j = 1..n, i = 1..n

def calcTrafficRatio(y, n):

k = []

for i in range(n):

row = []

for j in range(n):

value = y[j]/sum(y)

row.append(value)

k.append(row)

return k

# 3. Матрица интенсивностей трафика в направлениях связи

# Y = |y[i][j]|, i = 1..n, j = 1..n

# y[i][j] = k[i][j] \* y[i], i = 1..n, j = 1..n

def calcTrafficMatrixIntensity(k, y, n):

Y = []

for i in range(n):

row = []

for j in range(n):

value = k[i][j]\*y[i]

row.append(value)

Y.append(row)

return Y

# 4. Матрица кратчайших маршрутов между вершинами графа

# Алгоритм Флойда

# Принимает матрицу смежности как аргумент

# Элемент матрицы смежности может быть None

def calcByFloydsAlgorithm(R, n):

for k in range(n):

for i in range(n):

for j in range(n):

R[i][j] = minNone(R[i][j], sumNone(R[i][k], R[k][j]))

return R

sumNone = lambda a, b: a + b if (a != None and b != None) else None

None2Number = lambda a: a if (a != None) else float('inf')

def minNone(a, b):

a = None2Number(a)

b = None2Number(b)

return min(a, b)

# 4.5. Матрица восстановленных путей

# По S и D

ar3 = [[0] \* 20 for i in range(20)]

ar2 = [[0] \* 20 for i in range(20)]

ar1 = [[0] \* 20 for i in range(20)]

def calcPath(\_ar1, \_ar2, n):

ar1 = \_ar1

ar2 = \_ar2

for i in range(n):

for j in range(n):

calcDeykstra(i, j)

return ar1

def calcDeykstra(k, l):

for j in range(20):

if ar1[k][j] == ((l+1)):

ar3[k][l] += ar2[k][j]

if ar1[l][j] == ((k+1)):

ar3[k][l] += ar2[k][j]

# 5. Матрица интенсивностей нагрузок на линии связи

# Правильно ли я считаю?!!

# j = 1..n, i = 1..n

def calcIntensity(Y, R, n):

Ytilda = []

for i in range(n):

row = []

for j in range(n):

if i == j:

row.append(Y[i][j])

else:

row.append(0)

Ytilda.append(row)

return Ytilda

# 6. Матрица потоков

# j = 1..n, i = 1..n

def calcStreamMatrix(Ytilda, q, n):

V = [[0] \* 20 for i in range(20)]

p0 = 1 - q / 100

v = 0

p = 1

for i in range(n):

for j in range(n):

while p0 <= p:

#print(str(p0) + " <= " + str(p))

v += 1

p = fErlang(Ytilda[i][j], v, n)

V[i][j] = v - 1

p = 1

v = 0

return V

def fErlang(yt, v, n):

p = 1

for i in range(v):

if yt != 0:

p = 1 + p \* i / yt # Деление на 0

else:

return 0

return 1 / p

# 7. Интенсивность трафика ПД в линиях связи

# j = 1..n, i = 1..n

def calcTrafficLineIntensity(V, Codec, n):

if Codec == 'G.711':

a0 = 85600 # бит/с

A = []

for i in range(n):

row = []

for j in range(n):

row.append(V[i][j]\*a0)

A.append(row)

return A

# 8. Пропускнаяспособность линий связи

# j = 1..n, i = 1..n

def calcLinesCapacity(A, L, T0, n):

B = []

for i in range(n):

row = []

for j in range(n):

if (A[i][j] != 0):

row.append(A[i][j]+(L\*8/T0))

else:

row.append(A[i][j])

B.append(row)

return B

**OPTIMIZATION.PY**

import random

import output as OUT

def optimization(B, A, L, \_next, n):

# Инициализируем необходимые матрицы

DEL = [[0] \* n for i in range(n)]

DL = [[0] \* n for i in range(n)]

O = [[0] \* n for i in range(n)]

Bo = B

dc = 10000 # Шаг

m0 = float('inf')

Oprev = float('inf')

cycles = 0

while True: # Базовый цикл

Oprev = m0

cycles += 1

for i in range(n):

for j in range(n):

if Bo[i][j] != 0:

L = 200 \* 8

for x in range(n):

for y in range(n):

if Bo[x][y] != 0:

bm = Bo[x][y]

am = A[x][y]

if (i == x) and (j == y):

bm += dc;

DEL[x][y] = L / (bm - am)

for x in range(1, n+1):

for y in range(1, n+1):

\_sum = 0

if x == y:

\_sum = DEL[x-1][y-1]

else:

k = x

nextK = y

while (k != y):

nextK = \_next[k-1][y-1]

\_sum += DEL[k-1][int(nextK-1)]

k = nextK

DL[x-1][y-1] = \_sum

Topt = 0.05

sumO = 0

for x in range(n):

for y in range(n):

sumO += (DL[x][y] - Topt) \* (DL[x][y] - Topt)

O[i][j] = sumO

mi = 0

mj = 0

m0 = O[mi][mj]

for i in range(n):

for j in range(n):

if O[i][j] != 0:

tmp = O[i][j]

if tmp < m0:

m0 = tmp

mi = i

mj = j

b = Bo[mi][mj]

Bo[mi][mj] = b + dc

if m0 < Oprev:

pass

else:

break

print("Оптимизация выполнена.") # Спорное утверждение

print("DEL: ")

OUT.printMatrix(DEL, n)

print("DL: ")

OUT.printMatrix(DL, n)

print("O: ")

OUT.printMatrix(DEL, n)

print("Bo: ")

OUT.printMatrix(DEL, n)

**OUTPUT.PY**

def printMatrix(M, n):

print('\n ', end = '')

for i in range(n):

print(' {0:5.0f} '.format(i+1), end = '')

print()

for i in range(n):

print('{0:5.0f} '.format(i+1), end = '')

for j in range(n):

print('{0:7.5f}'.format(M[i][j]), end = ' ')

print()

print()

def printArr(A):

print()

for i in range(len(A)):

print('{0:6.0f}'.format(i+1), end = ' ')

print()

for i in range(len(A)):

print('{0:6.2f}'.format(A[i]) if A[i] != None else 'None', end = ' ')

print('\n')