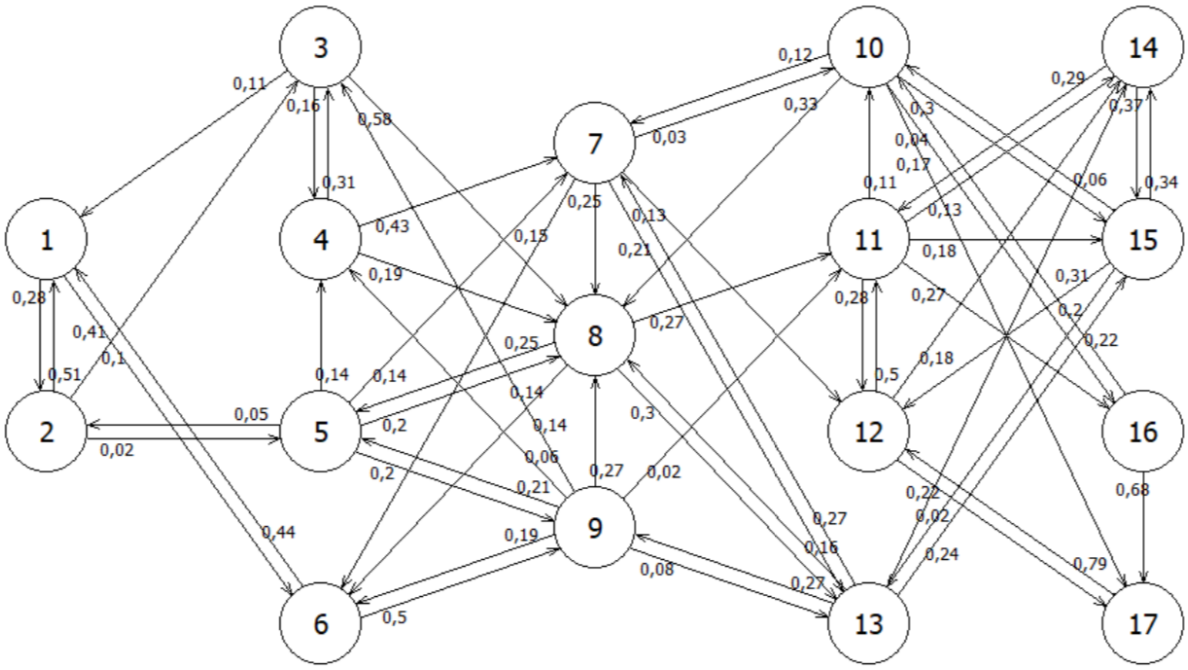
Выполнил студент Деменчук Г.М.
Группа ПИ19-4
Вариант 81

Контрольная работа №1

Система имеет 17 дискретных состояний. Изменение состояний происходит в дискретные моменты времени с заданной вероятность. Схема марковского процесса изображена на рисунке.



Инициализация матрицы

Импортируем Numpy

In [1]:
```
import copy
import numpy as np
from numpy.linalg import matrix_power
from numpy.matrixlib import matrix
```

Добавляем элементы матрицы

In [2]:
```
P = np.matrix([
    [0.31, 0.28, 0, 0, 0, 0.41, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0.51, 0.37, 0.1, 0, 0.02, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0.11, 0, 0.15, 0.16, 0, 0, 0, 0.58, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0.31, 0.07, 0, 0, 0.43, 0.19, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0.05, 0, 0.14, 0.27, 0, 0.14, 0.2, 0.2, 0, 0, 0, 0, 0, 0, 0, 0],
    [0.44, 0, 0, 0, 0, 0.06, 0, 0, 0.5, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0.15, 0.23, 0.25, 0, 0.03, 0, 0.13, 0.21, 0, 0, 0, 0],
    [0, 0, 0, 0, 0.25, 0.14, 0, 0.04, 0, 0, 0.27, 0, 0.3, 0, 0, 0, 0],
    [0, 0, 0.14, 0.06, 0.21, 0.19, 0, 0.27, 0.03, 0, 0.02, 0, 0.08, 0, 0, 0, 0],
```

```
    [0, 0, 0, 0, 0, 0, 0.12, 0.33, 0, 0.04, 0, 0, 0, 0, 0.3, 0.04, 0.17],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0.11, 0.03, 0.28, 0, 0.13, 0.18, 0.27, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.5, 0.1, 0, 0.18, 0, 0, 0.22],
    [0, 0, 0, 0, 0, 0, 0.27, 0.16, 0.27, 0, 0, 0, 0.04, 0.02, 0.24, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.29, 0, 0, 0.34, 0.37, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0.06, 0, 0.31, 0.2, 0.34, 0.09, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0.22, 0, 0, 0, 0, 0, 0.1, 0.68],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.79, 0, 0, 0, 0, 0.21]
])

print(P)
```

```
[[0.31 0.28 0.   0.   0.   0.41 0.   0.   0.   0.   0.   0.   0.   0.
  0.   0.   0.  ]
 [0.51 0.37 0.1  0.   0.02 0.   0.   0.   0.   0.   0.   0.   0.   0.
  0.   0.   0.  ]
 [0.11 0.   0.15 0.16 0.   0.   0.   0.58 0.   0.   0.   0.   0.   0.
  0.   0.   0.  ]
 [0.   0.   0.31 0.07 0.   0.   0.43 0.19 0.   0.   0.   0.   0.   0.
  0.   0.   0.  ]
 [0.   0.05 0.   0.14 0.27 0.   0.14 0.2  0.2  0.   0.   0.   0.   0.
  0.   0.   0.  ]
 [0.44 0.   0.   0.   0.   0.06 0.   0.   0.5  0.   0.   0.   0.   0.
  0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.15 0.23 0.25 0.   0.03 0.   0.13 0.21 0.
  0.   0.   0.  ]
 [0.   0.   0.   0.   0.25 0.14 0.   0.04 0.   0.   0.27 0.   0.3  0.
  0.   0.   0.  ]
 [0.   0.   0.14 0.06 0.21 0.19 0.   0.27 0.03 0.   0.02 0.   0.08 0.
  0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.12 0.33 0.   0.04 0.   0.   0.   0.
  0.3  0.04 0.17]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.11 0.03 0.28 0.   0.13
  0.18 0.27 0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.5  0.1  0.   0.18
  0.   0.   0.22]
 [0.   0.   0.   0.   0.   0.   0.27 0.16 0.27 0.   0.   0.   0.04 0.02
  0.24 0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.29 0.   0.   0.34
  0.37 0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.06 0.   0.31 0.2  0.34
  0.09 0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.22 0.   0.   0.   0.
  0.   0.1  0.68]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.79 0.   0.
  0.   0.   0.21]]
```

Задание 1

Найдите вероятность того, что за 10 шагов система перейдет из состояния 1 в состояние 12;

$$\left\| p_{ij}^{(k)} \right\| = \left\| p_{ij}^{(k-1)} \right\| \times P = P \times P \times \cdots \times P = P^k$$

Возводим матрицу P в степень 10

In [3]:
```
new_matrix = matrix_power(P, 10)
print(new_matrix)
```

```
[[0.15806223 0.08528433 0.03508663 0.02202248 0.06170556 0.11461612
  0.04415639 0.09053492 0.09070655 0.01277155 0.05936269 0.05188577
  0.05344336 0.03854974 0.04033735 0.01610312 0.0253712 ]
 [0.16253543 0.0881126  0.03553747 0.02199401 0.06138168 0.11686047
  0.04319036 0.08996192 0.09122713 0.0123647  0.05772414 0.05028476
  0.05244757 0.03715754 0.03911844 0.01568885 0.02441294]
 [0.08233368 0.04233103 0.02229025 0.01590878 0.04801564 0.06946442
  0.04245077 0.07546182 0.0628904  0.02384549 0.10146485 0.11020817
  0.05491547 0.08489492 0.07451677 0.03039366 0.05861387]
 [0.07463233 0.03774509 0.02093456 0.01533594 0.04666936 0.06452958
  0.04250019 0.07406647 0.06052152 0.02495969 0.1055867  0.11606161
  0.05507339 0.08947437 0.07818616 0.03190066 0.06182238]
 [0.08339603 0.04254473 0.02298762 0.0166257  0.05004293 0.07092473
  0.04419911 0.07806925 0.0650403  0.02328146 0.09923316 0.10654984
  0.05625249 0.08200129 0.07292855 0.02960791 0.05631492]
 [0.12750173 0.06777618 0.03046333 0.02012516 0.05788464 0.09747109
  0.04498762 0.08667629 0.08082305 0.01684654 0.07462576 0.07300967
  0.05522509 0.05510375 0.05311996 0.02137623 0.03698393]
 [0.0691588  0.03510178 0.01916624 0.01393158 0.04285667 0.05977757
  0.03977229 0.0692702  0.0560087  0.02641202 0.11155404 0.12416577
  0.05319187 0.09641561 0.08223877 0.0337327  0.06724538]
 [0.06629391 0.03335612 0.01855245 0.01364333 0.04211418 0.05767923
  0.03963867 0.06843526 0.05509397 0.02684628 0.11337338 0.12680382
  0.05302858 0.0983372  0.0837763  0.03440379 0.06862352]
 [0.0862643  0.04416046 0.02327045 0.01667745 0.05000552 0.07216764
  0.04381822 0.07793009 0.06577534 0.02298404 0.09829125 0.10503748
  0.05584108 0.08095108 0.07194834 0.02918499 0.05569228]
 [0.04131735 0.02020733 0.01282979 0.00985075 0.03236195 0.04027436
  0.03429033 0.05669834 0.04116652 0.03170393 0.13130457 0.15453462
  0.0494638  0.11982402 0.09868469 0.04115306 0.08433458]
 [0.0183626  0.00808491 0.00697753 0.00583386 0.0217633  0.02289701
  0.02788799 0.04366633 0.02727936 0.03655467 0.15090305 0.1812912
  0.04521554 0.14181718 0.1125123  0.04738788 0.1015653 ]
 [0.01411811 0.00578804 0.00575458 0.00501319 0.01949766 0.01931535
  0.02666488 0.04100432 0.02481763 0.03743676 0.15402642 0.18750988
  0.04400121 0.14588944 0.11582404 0.04890144 0.10443704]
 [0.06224486 0.03122583 0.01807755 0.01343409 0.04180421 0.05589785
  0.03985145 0.06820825 0.05362833 0.02735667 0.11455215 0.12960533
  0.05338216 0.10034472 0.08571426 0.03519838 0.06947391]
 [0.01941409 0.00845268 0.00743755 0.00627438 0.0230261  0.0240372
  0.02914988 0.04544773 0.02893722 0.03608511 0.1483874  0.17885564
  0.04612652 0.14014306 0.11241526 0.04692954 0.09888064]
 [0.02648052 0.01215191 0.00920248 0.00747025 0.02617465 0.0292499
  0.03084326 0.04919574 0.03296332 0.03470307 0.14366746 0.16973674
  0.04757666 0.13378679 0.10736806 0.04460116 0.09482803]
 [0.01599533 0.00684782 0.00602783 0.00513256 0.0196524  0.02024756
  0.02647555 0.04110448 0.02532174 0.03729723 0.15328397 0.18658761
  0.04359903 0.14459898 0.11473146 0.04879582 0.10430062]
 [0.01078468 0.00421267 0.00485258 0.00428522 0.01757635 0.01680135
  0.02515248 0.03853175 0.02205418 0.03832614 0.15766951 0.19165877
  0.04318421 0.14961152 0.11748195 0.04974248 0.10807415]]
```

Обращаемся к 1 строке 12 столбцу

In [4]:
```
result_1 = new_matrix[1-1, 12-1]
print(f"Вероятность перехода: {result_1}")
```

```
Вероятность перехода: 0.051885771452628654
```

Задание 2

Найдите вероятности состояний системы спустя 10 шагов, если в начальный момент

вероятность состояний были следующими
A=(0,01 ; 0,09 ; 0,03 ; 0,04 ; 0,03 ; 0,11 ; 0,09 ; 0,1 ; 0,09 ; 0,03 ; 0,09 ; 0,06 ; 0,05 ; 0,01 ; 0,02 ; 0,1 ; 0,05);

$$A(k) = A(k-1) \times P = A(0) \times P \times \cdots \times P = A(0) \times P^k$$

Объявляем вектор вероятностей состояний

In [5]:
```
a0 = np.array([0.01,0.09,0.03,0.04,0.03,0.11,0.09,0.1,0.09,0.03,0.09,0.06,0.05,0
```

Возводим матрицу в степень 10

In [6]:
```
p_power = matrix_power(P, 10)
```

Умножаем полученную матрицу на вектор вероятностей и получаем результат:

In [7]:
```
#Умножение матриц
result_2 = np.dot(a0, p_power)
print(result_2)
```

```
[[0.06852266 0.03522906 0.01813633 0.0128548  0.0396737  0.05773424
  0.03700849 0.06517348 0.05318092 0.02719215 0.11452415 0.12963753
  0.05077768 0.10032018 0.08451306 0.03502499 0.07049658]]
```

Задание 3

Найдите вероятность первого перехода за 9 шагов из состояния 1 в состояние 9;

$$\hat{p}_{ij}^{(k)} = \sum_{m \neq j} p_{im} \hat{p}_{mj}^{(k-1)}$$

Объявляем функцию для того, чтоб совершать переход

In [8]:
```
def task3_def(P, buffer_matrix):
    """
    Совершает переход в матрице
    P - основная матрица
    buffer - буфераная матрица
    """

    #Инициализация нулями
    temp_matrix = np.zeros(P.shape)

    #Цикл по размерности матрицы
    for i in range(P.shape[0]):
        for j in range(P.shape[1]):
```

```
            buffer_list = []
            #Цикл по m
            for m in range(P.shape[0]):

                #Если m == j
                buf_result = 0
                #Если m!= j
                if m != j:
                    buf_result = P[i, m] * buffer_matrix[m, j]

                buffer_list.append(buf_result)

            #Выставляем новый элемент в матрице, сумма buffer_list
            temp_matrix[i, j] = sum(buffer_list)

    return temp_matrix
```

In [9]:

```
def first_transition_p(start, end, n):
    """
    Вероятность первого перехода
    """
    #Копируем матрицу P
    result_matrix = P.copy()

    #От 2 до n +1
    for i in range(2, n + 1):
        result_matrix = task3_def(P, result_matrix)

    # Получаем элемент результирующей матрицы
    result = result_matrix[start - 1, end - 1]
    return result
```

In [10]:

```
start = 1
end = 9
n = 9

result_3 = first_transition_p(1,9,9)
print(f"Вероятность первого перехода за {n} шагов из состояния {start} в состояние {end}: {
```

```
Вероятность первого перехода за 9 шагов из состояния 1 в состояние 9: 0.041984906975047254
```

Задание 4

Найти вероятность перехода из состояния 2 в состояние 1 не позднее чем за 6 шагов

- $p_{ij}^{(t\leq k)} = \sum_{t=1}^{k} \hat{p}_{ij}^{(t)}$ - вероятность перехода из состояния i в состояние j не позднее чем за k шагов.

In [11]:

```
def task3_def_modified(current_m, previous_m):
    """
    Модифицированная версия перехода в матрице.
    Совершает переход в матрице, но при этом учитывает предыдущее значение в previous_m, а
    """
    current_len = current_m.shape[0]
```

```
    result_m = np.zeros((current_len, current_len))

    for i in range(current_len):
        for j in range(current_len):

            buf_list = []
            for m in range(current_len):
                if m != j:
                    buf = current_m[i, m] * previous_m[m, j]
                    buf_list.append(buf)

            result_m[i, j] = sum(buf_list)

    return result_m
```

In [12]:

```
start = 2
end = 1
n = 6

#Делаем копии матриц
previous_m = np.copy(P)
result = np.copy(P)
matrix = np.copy(P)

for i in range(1, n):
    previous_m = task3_def_modified(matrix, previous_m)
    result+= previous_m

result_4 = result[start-1][end-1]
print(f"Вероятность перехода из состояния {start} в состояние {end} не позднее чем за {n} ш
```

```
Вероятность перехода из состояния 2 в состояние 1 не позднее чем за 6 шагов = 0.844640532913
```

Задание 5

Найдите среднее количество шагов для перехода из состояния 4 в состояние 3;

- $\bar{t}_{i,j} = \sum_{t=1}^{\infty} t \cdot \hat{p}_{ij}^{(t)}$ - среднее количество шагов, необходимых для первого перехода из состояния i в состояние j

In [13]:

```
start = 4 - 1
end = 3 - 1

result_matrix = P.copy()
result_5 = result_matrix[start, end]

#Тут надо чуть подождать, пока компутер 10тыс итераций сделает
for i in range(2, 10000):
    result_matrix = task3_def(P, result_matrix)
    buffer = i * result_matrix[start, end]
    result_5 += buffer

print(result_5)
```

```
67.18936874533797
```

Задание 6

Найдите вероятность первого возвращения в состояние 11 за 9 шагов;

$$f_{jj}^{(k)} = p_{jj}^{(k)} - \sum_{m=1}^{k-1} f_{jj}^{(m)} p_{jj}^{(k-m)}$$

Вводим функцию first_return_p т.к. вычисление вероятности первого возвращаения нужно еще и для заданий 7 и 8

In [14]:
```
def first_return_p(P, start, n):
    """
    Вероятность первого возвращения
    """

    #Заполняем нулями
    result_matrix = np.zeros(P.shape)

    for i in range(1, n):

        #Вероятность первого возвращения * матрица вероятностей ^ n-i
        buffer = first_return_p(P,start, i) * matrix_power(P, n - i)
        result_matrix += buffer

    result_matrix = matrix_power(P, n) - result_matrix
    result = result_matrix[start - 1, start - 1]
    return result
```

In [15]:
```
result_6 = first_return_p(P, 11, 9)
print(f"6. Вероятность первого возвращения в состояние 11 за 9 шагов: {result_6}")
```

```
6. Вероятность первого возвращения в состояние 11 за 9 шагов: 0.03491918711359146
```

Задание 7

Найдите вероятность возвращения в состояние 16 не позднее чем за 9 шагов;

- вероятность возвращения не позднее чем за k шагов.

$$f_{jj}^{(t\leq k)} = \sum_{t=1}^{k} f_{jj}^{(t)}$$

In [16]:
```
start = 16
n = 9
```

```
result_list = []
for step in range(1, n + 1):
    buffer = first_return_p(P, start - 1, step)
    result_list.append(buffer)

result_7 = sum(result_list)
print(result_7)
```

```
0.6680329333785395
```

Задание 8

Найдите среднее время возвращения в состояние 3

среднее время возвращения

$$\bar{\mu}_{j,j} = \sum_{t=1}^{\infty} t \cdot f_{jj}^{(t)}$$

In [17]:

```
start = 3

result_list = []
#Все еще максимально плохой способ вычислений
for i in range(1, 20):
    buffer = i * first_return_p(P, start - 1, i)
    result_list.append(buffer)

result_8 = sum(result_list)
print(result_8)
print(43.991976134208066)
```

```
2.2004889285844165
43.991976134208066
```

Задание 9

Найдите установившиеся вероятности

Вычисялем M по формуле ниже:

$$M = \begin{bmatrix} p_{11} - 1 & p_{21} & p_{31} \\ p_{12} & p_{22} - 1 & p_{32} \\ p_{13} & p_{23} & p_{33} - 1 \end{bmatrix} = P^T - E, \qquad X = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$MX = B$$

Создаем единичную матрицу E

In [18]:

```
E = np.eye(P.shape[0])
print("Матрица E с единицами на главной диагонали")
print(E)
```

Матрица E с единицами на главной диагонали

```
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

Вычисляем матрицу M

In [19]:

```
M = P.T - E
print("M")
print(M)
```

```
M
[[-0.69  0.51  0.11  0.    0.    0.44  0.    0.    0.    0.    0.    0.
   0.    0.    0.    0.    0.  ]
 [ 0.28 -0.63  0.    0.    0.05  0.    0.    0.    0.    0.    0.    0.
   0.    0.    0.    0.    0.  ]
 [ 0.    0.1  -0.85  0.31  0.    0.    0.    0.    0.14  0.    0.    0.
   0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.16 -0.93  0.14  0.    0.    0.    0.06  0.    0.    0.
   0.    0.    0.    0.    0.  ]
 [ 0.    0.02  0.    0.   -0.73  0.    0.    0.25  0.21  0.    0.    0.
   0.    0.    0.    0.    0.  ]
 [ 0.41  0.    0.    0.    0.   -0.94  0.15  0.14  0.19  0.    0.    0.
   0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.43  0.14  0.   -0.77  0.    0.    0.12  0.    0.
   0.27  0.    0.    0.    0.  ]
 [ 0.    0.    0.58  0.19  0.2   0.    0.25 -0.96  0.27  0.33  0.    0.
   0.16  0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.2   0.5   0.    0.   -0.97  0.    0.    0.
   0.27  0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.03  0.    0.   -0.96  0.11  0.
   0.    0.    0.06  0.22  0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.27  0.02  0.   -0.97  0.5
   0.    0.29  0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.13  0.    0.    0.    0.28 -0.9
   0.    0.    0.31  0.    0.79]
 [ 0.    0.    0.    0.    0.    0.    0.21  0.3   0.08  0.    0.    0.
  -0.96  0.    0.2   0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.13  0.18
   0.02 -0.66  0.34  0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.3   0.18  0.
   0.24  0.37 -0.91  0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.04  0.27  0.
   0.    0.    0.   -0.9   0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.17  0.    0.22
   0.    0.    0.    0.68 -0.79]]
```

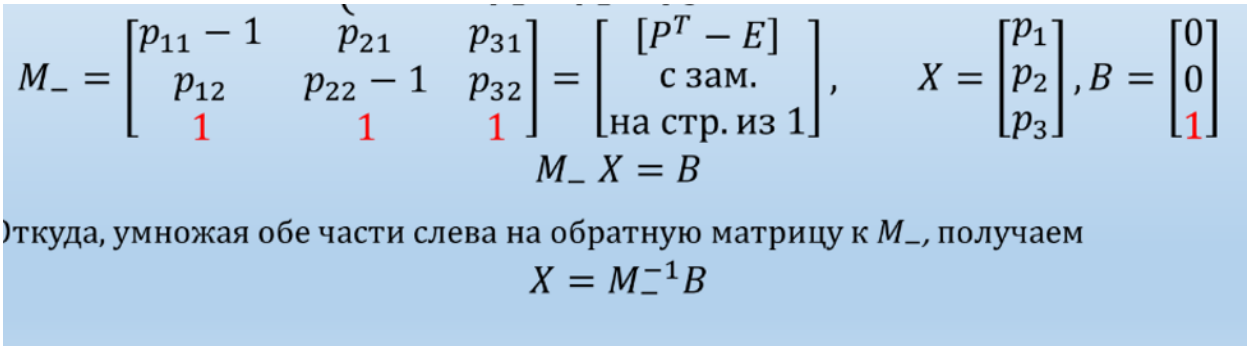
Инициализируем матрицу B

In [20]:

```
B = np.zeros(M.shape[0])
print(B)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Элементы последних строк матриц M_ и B меняем на 1



Добавляем единицы матрице M

In [21]:

```
M_ = copy.deepcopy(M)
M_[-1, :] = 1
print("Матрица M_ после замены на единицы")
print(M_)
```

```
Матрица M_ после замены на единицы
[[-0.69  0.51  0.11  0.    0.    0.44  0.    0.    0.    0.    0.    0.
   0.    0.    0.    0.    0.  ]
 [ 0.28 -0.63  0.    0.    0.05  0.    0.    0.    0.    0.    0.    0.
   0.    0.    0.    0.    0.  ]
 [ 0.    0.1  -0.85  0.31  0.    0.    0.    0.    0.14  0.    0.    0.
   0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.16 -0.93  0.14  0.    0.    0.    0.06  0.    0.    0.
   0.    0.    0.    0.    0.  ]
 [ 0.    0.02  0.    0.   -0.73  0.    0.    0.25  0.21  0.    0.    0.
   0.    0.    0.    0.    0.  ]
 [ 0.41  0.    0.    0.    0.   -0.94  0.15  0.14  0.19  0.    0.    0.
   0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.43  0.14  0.   -0.77  0.    0.    0.12  0.    0.
   0.27  0.    0.    0.    0.  ]
 [ 0.    0.    0.58  0.19  0.2   0.    0.25 -0.96  0.27  0.33  0.    0.
   0.16  0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.2   0.5   0.    0.   -0.97  0.    0.    0.
   0.27  0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.03  0.    0.   -0.96  0.11  0.
   0.    0.    0.06  0.22  0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.27  0.02  0.   -0.97  0.5
   0.    0.29  0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.13  0.    0.    0.    0.28 -0.9
   0.    0.    0.31  0.    0.79]
 [ 0.    0.    0.    0.    0.    0.    0.21  0.3   0.08  0.    0.    0.
  -0.96  0.    0.2   0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.13  0.18
   0.02 -0.66  0.34  0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.3   0.18  0.
   0.24  0.37 -0.91  0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.04  0.27  0.
   0.    0.    0.   -0.9   0.  ]
 [ 1.    1.    1.    1.    1.    1.    1.    1.    1.    1.    1.    1.
   1.    1.    1.    1.    1.  ]]
```

Добавляем единицы матрице B

In [22]:
```
B[-1] = 1
print("Матрица B после замены на единицы")
print(B)
```

```
Матрица B после замены на единицы
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```

Находим обратную матрицу M_

In [23]:
```
M_T = np.linalg.inv(M_)
print(M_T)
```

```
[[-4.97274277e+00 -4.25695005e+00 -1.61888384e+00 -1.29703321e+00
  -1.47008291e+00 -3.13355322e+00 -1.23578115e+00 -1.15005960e+00
  -1.60608706e+00 -5.97335418e-01 -1.38834553e-01 -5.76080038e-02
  -1.01954676e+00 -1.55223610e-01 -2.91069959e-01 -9.54482988e-02
   4.55103230e-02]
 [-2.28131134e+00 -3.54925505e+00 -7.87052330e-01 -6.35485798e-01
  -8.29404134e-01 -1.47084104e+00 -6.01537721e-01 -5.84661938e-01
  -8.03090019e-01 -2.99739269e-01 -6.94478201e-02 -2.87739469e-02
  -5.06623957e-01 -7.72447788e-02 -1.44792173e-01 -4.80124680e-02
   2.27314180e-02]
 [-6.68660673e-01 -7.89107777e-01 -1.59532256e+00 -7.19606216e-01
  -4.81112800e-01 -6.18193273e-01 -2.99142456e-01 -3.10979411e-01
  -5.99849071e-01 -1.59349227e-01 -3.88878765e-02 -1.64981710e-02
  -3.14162683e-01 -4.68776151e-02 -8.83656672e-02 -2.44703053e-02
   1.30335551e-02]
 [-3.56023028e-01 -3.62272643e-01 -4.64086795e-01 -1.36816794e+00
  -4.96038979e-01 -3.75338993e-01 -2.10109186e-01 -2.51851398e-01
  -4.11675918e-01 -1.23372797e-01 -2.92341150e-02 -1.22397950e-02
  -2.23321217e-01 -3.37122006e-02 -6.33556678e-02 -1.94139746e-02
   9.66943808e-03]
 [-8.97163323e-01 -8.81693301e-01 -8.51109875e-01 -7.43735099e-01
  -2.21802783e+00 -9.84699090e-01 -6.59000862e-01 -9.26406675e-01
  -1.12484668e+00 -4.31636455e-01 -9.75690358e-02 -3.99469098e-02
  -6.73999994e-01 -1.04031998e-01 -1.94389614e-01 -7.04466237e-02
   3.15580587e-02]
 [-2.71402513e+00 -2.36448547e+00 -1.22760836e+00 -1.11749653e+00
  -1.22372429e+00 -3.05459439e+00 -1.16590701e+00 -1.04808136e+00
  -1.43781992e+00 -5.49468172e-01 -1.27499516e-01 -5.28636611e-02
  -9.33070801e-01 -1.42165718e-01 -2.66531909e-01 -8.79118952e-02
   4.17622922e-02]
 [-3.48514941e-01 -3.42251269e-01 -5.47179916e-01 -1.05082587e+00
  -8.07047531e-01 -4.33239553e-01 -1.73278072e+00 -5.02788636e-01
  -5.73763738e-01 -4.24870098e-01 -1.00020421e-01 -4.17509657e-02
  -7.54131398e-01 -1.14160674e-01 -2.14387529e-01 -6.72090652e-02
   3.29832629e-02]
 [-1.05803220e+00 -1.06953452e+00 -1.62156445e+00 -1.37695977e+00
  -1.37688862e+00 -1.18550352e+00 -1.10225869e+00 -1.89158030e+00
  -1.40864607e+00 -8.33035214e-01 -1.77061735e-01 -7.02327757e-02
  -1.04467222e+00 -1.67569575e-01 -3.10086242e-01 -1.41982060e-01
   5.54838928e-02]
 [-1.64188071e+00 -1.45365419e+00 -9.53228803e-01 -8.85604591e-01
  -1.23023845e+00 -1.87160779e+00 -9.21310484e-01 -9.12802660e-01
  -2.15673695e+00 -4.80195349e-01 -1.21766696e-01 -5.25122443e-02
  -1.05104505e+00 -1.54790806e-01 -2.92795306e-01 -7.12870042e-02
   4.14846730e-02]
 [ 5.54294386e-01  5.59533276e-01  3.16582953e-01  2.90336482e-01
   3.35480383e-01  4.74014353e-01  2.27361780e-01  2.24519597e-01
   3.40472076e-01 -9.07853817e-01 -1.32793837e-01 -3.98074990e-02
   2.32377066e-01 -4.87641647e-03  1.03887934e-02 -1.86977684e-01
   3.14479242e-02]
 [ 2.50283040e+00  2.52349377e+00  1.49959336e+00  1.46403380e+00
   1.62362012e+00  2.16938753e+00  1.29768776e+00  1.09339961e+00
```

```
  1.61410613e+00  8.43912174e-01 -6.31640987e-01 -1.65728903e-01
  1.26553858e+00  1.98548041e-01  4.95383136e-01  3.51762791e-01
  1.30925833e-01]
[ 4.08302317e+00  4.10923459e+00  2.93161403e+00  2.77514951e+00
  3.00559086e+00  3.69145744e+00  2.43106028e+00  2.49226328e+00
  3.04047409e+00  1.73453004e+00  8.71501939e-01 -1.93927535e-01
  2.45287573e+00  1.31395275e+00  1.24667704e+00  5.94221513e-01
  1.53202753e-01]
[-2.08070607e-01 -1.90604338e-01 -5.20762167e-01 -5.61263594e-01
 -5.10605881e-01 -3.37861255e-01 -6.62657385e-01 -6.52209502e-01
 -5.48724248e-01 -3.87881893e-01 -1.29074554e-01 -6.11691276e-02
 -1.54880853e+00 -2.15769715e-01 -4.14324330e-01 -4.11226729e-02
  4.83236108e-02]
[ 2.43013865e+00  2.45178989e+00  1.53684664e+00  1.45527332e+00
  1.61545098e+00  2.12446651e+00  1.23883808e+00  1.18430591e+00
  1.61694856e+00  6.40873113e-01  6.20439907e-03 -1.50966117e-01
  1.04902399e+00 -1.43463854e+00 -1.83578223e-01  2.89172574e-01
  1.19263232e-01]
[ 1.61100236e+00  1.63022646e+00  8.88518711e-01  8.28983401e-01
  9.53919987e-01  1.36006590e+00  6.60576536e-01  5.99813972e-01
  9.44239424e-01  2.59114764e-02 -2.00237136e-01 -1.23419237e-01
  3.44983399e-01 -6.02555239e-01 -1.18140239e+00  1.14668580e-01
  9.75011975e-02]
[ 7.75484425e-01  7.81916276e-01  4.63948362e-01  4.52113985e-01
  5.01996276e-01  6.71883562e-01  3.99411295e-01  3.37998531e-01
  4.99363932e-01  2.12824594e-01 -1.95394244e-01 -5.14878930e-02
  3.89989443e-01  5.93476828e-02  1.49076665e-01 -1.01389239e+00
  4.06754355e-02]
[ 3.18965133e+00  3.20361435e+00  2.54969503e+00  2.49028811e+00
  2.60711281e+00  2.97415683e+00  2.33554993e+00  2.29912058e+00
  2.61563548e+00  1.73668631e+00  1.31175619e+00  1.15893278e+00
  2.33459441e+00  1.68176840e+00  1.74355339e+00  5.18348986e-01
  8.44430999e-02]]
```

Перемножаем матрицы M_T и B

In [24]:

```
#Перемножаем матрицы
result_9 = np.dot(M_T, B)

print("Устновившиеся вероятности:")
print(result_9)

print("Сумма:", result_9.sum())
```

```
Устновившиеся вероятности:
[[0.04551032 0.02273142 0.01303356 0.00966944 0.03155806 0.04176229
  0.03298326 0.05548389 0.04148467 0.03144792 0.13092583 0.15320275
  0.04832361 0.11926323 0.0975012  0.04067544 0.0844431 ]]
Сумма: 1.0
```

Контрольная работа №2

In [25]:

```
import numpy as np
import copy
```

Задана система массового обслуживания со следующими характеристиками:

- интенсивность поступления λ=29
- каналов обслуживания m=7

- интенсивность обслуживания μ=6
- максимальный размер очереди n=6

Изначально требований в системе нет.

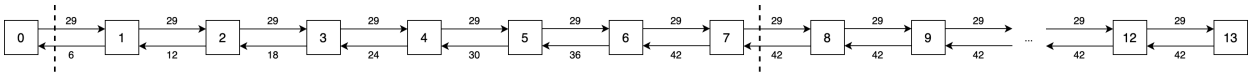
Задаем входные данные

In [26]:
```
L = 29
u = 6
m = 7
n = 6
```

Задание А

Составьте граф марковского процесса, запишите систему уравнений Колмогорова и найдите установившиеся вероятности состояний.

Граф марковского процесса



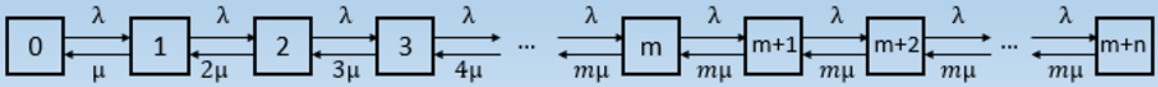
Система уравнений Колмогорова

$$\begin{cases} P_0 = 6P_1 - 29P_0 \\ P_1 = 29P_0 + 12P_2 - P_1(6 + 29) \\ P_2 = 29P_1 + 18P_3 - P_2(12 + 29) \\ P_3 = 29P_2 + 24P_4 - P_3(18 + 29) \\ P_4 = 29P_3 + 30P_5 - P_4(24 + 29) \\ P_5 = 29P_4 + 36P_6 - P_5(30 + 29) \\ P_6 = 29P_5 + 42P_7 - P_6(36 + 29) \\ \dots \\ P_{12} = 29P_{11} + 42P_{13} - P_{12}(42 + 29) \\ P_{13} = 29P_{12} - 42P_{13} \end{cases}$$

$$P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + \dots + P_{12} + P_{13} = 1$$

Общее описание входных данных

- Интенсивность поступления требований - λ
- Интенсивность обслуживания требований одним каналом - μ
- Количество каналов - m
- Количество мест в очереди - n



Составляем матрицу интенсивностей переходов

Руководствуемся формулой ниже:

$$\Lambda(t) = \|\lambda_{ij}(t)\| = \begin{Vmatrix} 0 & \lambda_{12}(t) & \dots & \lambda_{1n}(t) \\ \lambda_{21}(t) & 0 & \dots & \lambda_{1n}(t) \\ \dots & \dots & \dots & \dots \\ \lambda_{n1}(t) & \lambda_{n2}(t) & \dots & 0 \end{Vmatrix}$$

In [27]:
```
p_matrix = np.zeros((m + n + 1, m + n + 1))
for i in range(m + n):
    p_matrix[i, i + 1] = L
    if i < m:
        p_matrix[i + 1, i] = (u * (i + 1))
    else:
        p_matrix[i + 1, i] = u*m

print("Матрица интенсивности переходов")
print(p_matrix)
```

```
Матрица интенсивности переходов
[[ 0. 29.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 6.  0. 29.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. 12.  0. 29.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. 18.  0. 29.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. 24.  0. 29.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. 30.  0. 29.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. 36.  0. 29.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0. 42.  0. 29.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. 42.  0. 29.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. 42.  0. 29.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0. 42.  0. 29.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. 42.  0. 29.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. 42.  0. 29.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. 42.  0.]]
```

Рассчитываем установившиеся вероятности

Формула, по которой считаю:

$$M = \begin{bmatrix} -(\lambda_{12} + \lambda_{13}) & \lambda_{21} & \lambda_{31} \\ \lambda_{12} & -(\lambda_{21} + \lambda_{23}) & \lambda_{32} \\ \lambda_{13} & \lambda_{23} & -(\lambda_{31} + \lambda_{32}) \end{bmatrix}, X = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$M = \Lambda^T - D$, где $D = diag(\sum_j \lambda_{1j}, \sum_j \lambda_{2j}, \sum_j \lambda_{3j})$ – диагональная матрица из сумм строк матрицы интенсивностей переходов Λ

Определяем диагональную матрицу D

In [28]:
```
print("Диагональная матрица из сумм строк матрицы интенсивности переходов")
result = []
for i in range(p_matrix.shape[0]):
    result.append(p_matrix[i, :].sum())
D = np.diag(result)
print(D)
```

```
Диагональная матрица из сумм строк матрицы интенсивности переходов
[[29.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. 35.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. 41.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. 47.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. 53.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. 59.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0. 65.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. 71.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. 71.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0. 71.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. 71.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. 71.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. 71.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. 42.]]
```

Определяем M

In [29]:

```
M = p_matrix.T - D
print(M)
```

```
[[-29.   6.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [ 29. -35.  12.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.  29. -41.  18.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.  29. -47.  24.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.  29. -53.  30.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.  29. -59.  36.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.  29. -65.  42.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.  29. -71.  42.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.  29. -71.  42.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.  29. -71.  42.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.   0.  29. -71.  42.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  29. -71.  42.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  29. -71.  42.]
 [  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  29. -42.]]
```

Определяем M_

Работаем по формуле ниже, согласно которой, элементы последних строк матриц M_ и B меняем на 1

$$M\_ = \begin{bmatrix} -(\lambda_{12}+\lambda_{13}) & \lambda_{21} & \lambda_{31} \\ \lambda_{12} & -(\lambda_{21}+\lambda_{23}) & \lambda_{32} \\ 1 & 1 & 1 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$M\_ X = B$$

In [30]:

```
M_ = copy.deepcopy(M)
M_[-1, :] = 1
print("После замены на единицы")
print(M_)
```

```
После замены на единицы
[[-29.   6.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [ 29. -35.  12.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.  29. -41.  18.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.  29. -47.  24.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
```

```
[  0.   0.   0.  29. -53.  30.   0.   0.   0.   0.   0.   0.   0.]
[  0.   0.   0.   0.  29. -59.  36.   0.   0.   0.   0.   0.   0.]
[  0.   0.   0.   0.   0.  29. -65.  42.   0.   0.   0.   0.   0.]
[  0.   0.   0.   0.   0.   0.  29. -71.  42.   0.   0.   0.   0.]
[  0.   0.   0.   0.   0.   0.   0.  29. -71.  42.   0.   0.   0.]
[  0.   0.   0.   0.   0.   0.   0.   0.  29. -71.  42.   0.   0.]
[  0.   0.   0.   0.   0.   0.   0.   0.   0.  29. -71.  42.   0.]
[  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  29. -71.  42.]
[  1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.]]
```

Определяем B

In [31]:

```
b_matrix = np.zeros(M_.shape[0])
b_matrix[-1] = 1
print(b_matrix)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```

Вычисляем результирующий X по формуле:

$$X = M\_^{-1}B$$

In [32]:

```
X = np.linalg.inv(M_).dot(b_matrix)
print("Установившиеся вероятности:\n\n"+"\n".join(map(str,X)))
```

```
Установившиеся вероятности:

0.007390705126183501
0.03572174144322026
0.08632754182111563
0.1390832618229085
0.16805894136934776
0.16245697665703618
0.1308681200848347
0.09036132101095729
0.06239234069804194
0.04308042572007658
0.029746008235290968
0.020538910448177093
0.014181628642788949
0.009792076920020943
```

Проверяем корректность вероятностей

In [33]:

```
print("Сумма вероятностей:", X.sum(),"если = 1, то все правильно")
```

```
Сумма вероятностей: 1.0000000000000002 если = 1, то все правильно
```

Задание B

Найдите вероятность отказа в обслуживании.

b) • $p_{\text{отказа}} = p_{n+m}$ - вероятность отказа в обслуживании (новая заявка вынуждена будет покинуть систему необслуженной)

In [34]:
```
print("Вероятность отказа в обслуживании", X[-1])
```

```
Вероятность отказа в обслуживании 0.009792076920020943
```

Задание C

Найдите относительную и абсолютную интенсивность обслуживания.

c) • $q = 1 - p_{n+m}$ - относительная пропускная способность (доля от всех поступающих заявок)
• $A = (1 - p_{n+m}) * \lambda$ – абсолютная пропускная способность

In [35]:
```
print("Относительная пропускная способность = ", 1 - X[-1])
print("Абсолютная пропускная способность = ", (1 - X[-1]) * L)
```

```
Относительная пропускная способность =  0.9902079230799791
Абсолютная пропускная способность =  28.716029769319395
```

Задание D

Найдите среднюю длину в очереди.

d) $L_{\text{оч}} = \sum_{i=1}^{n} i p_{m+i}$ - средняя длина очереди

In [36]:
```
results = []

for i in range(1, n + 1):
    buf = i * X[m + i]
    results.append(buf)
d_result = sum(results)

print("d) Средняя длина очереди:", d_result)
```

```
d) Средняя длина очереди: 0.4496074633708468
```

Задание E

Найдите среднее время в очереди.

e) $T_{\text{оч}} = \sum_{i=0}^{n-1} \frac{i+1}{m\mu} p_{m+i}$ - среднее время в очереди

In [37]:
```
results = []
```

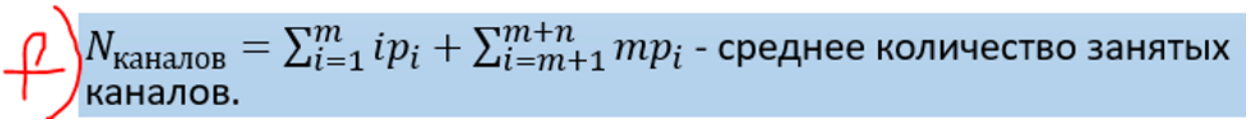
```
for i in range(n):
    buf = (i + 1)/ (m * u)* X[m + i]
    results.append(buf)

e_result = sum(results)
print("Среднее время в очереди:", e_result)
```

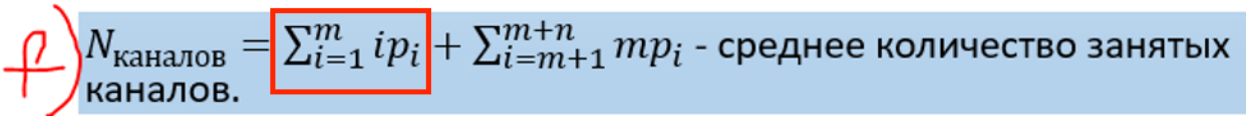
```
Среднее время в очереди: 0.015503705633477473
```

Задание F

Найдите среднее число занятых каналов.



Находим сумму левой части



In [38]:

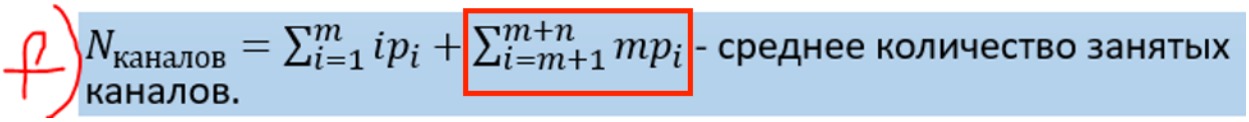
```
results_1 = []

for i in range(1, m + 1):
    buf = i * X[i]
    results_1.append(buf)

print("Сумма левой части:",sum(results_1))
```

```
Сумма левой части: 3.527885226902458
```

Находим сумму правой части



In [39]:

```
results_2 = []

for i in range(m+ 1, m + n + 1):
    buf = m * X[i]
    results_2.append(buf)

print("Сумма правой части:",sum(results_2))
```

```
Сумма правой части: 1.258119734650775
```

Получаем среднее число занятых каналов из двух сумм:

In [40]:

```
f_result = sum(results_1)+ sum(results_2)
print("Среднее число занятых каналов:",f_result)
```

```
Среднее число занятых каналов: 4.786004961553234
```

Задание G

Найдите вероятность того, что поступающая заявка не будет ждать в очереди.

Сумма установившихся вероятностей от 0 до m-1

In [41]:
```
print(m)
print(X[:m-1])

g_result = sum(X[:m-1])
print("Вероятность того, что поступающая заявка не будет ждать в очереди:", g_result)
```

```
7
[0.00739071 0.03572174 0.08632754 0.13908326 0.16805894 0.16245698]
Вероятность того, что поступающая заявка не будет ждать в очереди: 0.5990391682398118
```

Задание H

Найти среднее время простоя системы массового обслуживания.

h) $$\bar{t}_i = \frac{1}{\sum_j \lambda_{ij}}$$

In [42]:
```
h_result = 1/L
print("Среднее время простоя системы массового обслуживания:",h_result)
```

```
Среднее время простоя системы массового обслуживания: 0.034482758620689655
```