Варіант 6

In [7]:
```
import pandas as pd
import numpy as np
import copy
from IPython.display import display
```

Зчитуємо файли prob_06.csv та table_06.csv

In [8]:
```
prob = pd.read_csv("var_06/prob_06.csv",header=None)

prob.columns=[ i for i in range(0,20)]

prob_temp=(copy.deepcopy(prob))
prob_temp.index=['Probability of text','Probability of key']

display(prob_temp)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Probability of text	0.24	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
Probability of key	0.24	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04

In [9]:
```
table = pd.read_csv("var_06/table_06.csv",header=None)

table_temp=(copy.deepcopy(table))
table_temp.index=['K_'+str(i) for i in range(0,20)]
table_temp.columns=['M_'+str(i) for i in range(0,20)]

display(table_temp)
```

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_10	M_11	M_12	M_13	M_14	M_15	M_16	M_17	M_18	M_19
K_0	14	12	1	2	8	17	7	0	9	18	4	19	15	3	13	10	11	6	16	5
K_1	13	12	11	0	15	10	5	4	16	17	7	18	19	2	8	1	6	3	14	9
K_2	17	12	19	8	11	1	5	0	16	3	15	6	13	10	9	2	18	14	4	7
K_3	18	6	7	19	13	0	4	10	9	17	16	14	2	8	1	5	15	12	3	11
K_4	6	0	16	14	3	4	11	2	10	18	15	17	5	9	13	19	1	8	7	12
K_5	17	9	11	10	7	3	5	8	16	4	18	13	15	0	19	1	14	12	2	6
K_6	7	18	15	10	11	8	9	13	1	14	4	2	17	12	3	5	6	19	0	16
K_7	11	10	14	19	8	1	4	5	7	16	12	9	0	2	6	17	18	3	15	13
K_8	11	7	3	1	19	10	9	6	4	12	0	15	18	14	5	8	13	16	17	2
K_9	3	16	15	13	19	18	11	1	9	2	8	10	4	14	6	17	12	5	7	0
K_10	15	9	3	18	8	10	5	14	6	0	16	12	2	19	13	11	7	4	17	1
K_11	6	4	19	12	15	17	10	14	3	11	18	8	7	9	2	13	0	1	5	16
K_12	19	13	18	1	10	3	9	2	17	14	12	7	16	15	0	6	11	4	5	8
K_13	12	18	3	17	11	15	9	6	14	1	4	13	2	0	16	8	19	7	5	10
K_14	11	19	18	16	2	12	4	15	1	6	7	5	13	9	17	0	8	3	14	10
K_15	2	10	4	14	7	19	18	3	15	11	17	0	6	9	5	1	13	12	16	8
K_16	6	10	3	9	13	7	16	19	5	18	4	17	11	15	8	2	1	14	12	0
K_17	17	12	7	11	15	13	18	3	0	4	19	9	8	2	16	1	6	10	5	14
K_18	10	5	9	2	13	19	1	15	6	16	3	18	17	11	8	12	14	4	0	7
K_19	5	2	18	11	10	17	15	1	12	19	14	0	4	3	7	9	6	8	16	13

Обрахунок розподілу P(C)

In [10]:
```
c_probab = []    #порожній список для ймовірностей шифротексту
j = 0            #індекс шифротексти, ймовірність якого обчислюємо
for c in range(0,20):
    total =           0 #загальна ймовірність
    occurens = np.where(table == c)    #всі входження шифротексту "c"
    c_rows = occurens[0]               #номер ключів для цього ШТ
    c_columns = occurens[1]            #номер ВТ для цього ШТ

    for i in range(0,20):
        total += prob[c_rows[i]][1]*prob[c_columns[i]][0]

    c_probab.insert(j, round(total, 3))
    j+=1
```

```
c_probab = np.array(c_probab)
probability_c = pd.DataFrame(data=c_probab.reshape(1, 20))
probability_c

probability_c_temp=copy.deepcopy(probability_c)

probability_c_temp.index=['Probability cipher']

display(probability_c_temp)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Probability cipher	0.04	0.04	0.048	0.048	0.04	0.048	0.064	0.048	0.04	0.04	0.048	0.064	0.048	0.048	0.088	0.048	0.04	0.064	0.048	0.048

Обрахунок розподілу P(M,C)

In [11]:
```
temp=0

probability_m_c=[]

for c in range(0,20):
    for m in range(0,20):
        for k in range(0,20):
            if(table[m][k]==c):
                temp+=prob[m][0]*prob[k][1]

        probability_m_c.append(temp)
        temp=0



probability_m_c=np.array(probability_m_c)

probability_m_c=pd.DataFrame(data=probability_m_c.reshape(20,20))

probability_m_c_temp=copy.deepcopy(probability_m_c)
probability_m_c_temp.index=['C_'+str(i) for i in range(0,20)]
probability_m_c_temp.columns=['M_'+str(i) for i in range(0,20)]


display(probability_m_c_temp)
```

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_10	M_11	M_12	M_13	M_14	M_15	M_16	M_17
C_0	0.0000	0.0016	0.0000	0.0016	0.0000	0.0016	0.0000	0.0112	0.0016	0.0016	0.0016	0.0032	0.0016	0.0032	0.0016	0.0016	0.0016	0.0000
C_1	0.0000	0.0000	0.0096	0.0032	0.0000	0.0032	0.0016	0.0032	0.0032	0.0016	0.0000	0.0000	0.0000	0.0000	0.0016	0.0064	0.0032	0.0016
C_2	0.0096	0.0016	0.0000	0.0112	0.0016	0.0000	0.0000	0.0032	0.0000	0.0016	0.0000	0.0016	0.0048	0.0048	0.0016	0.0032	0.0000	0.0000
C_3	0.0096	0.0000	0.0064	0.0000	0.0016	0.0032	0.0000	0.0032	0.0016	0.0016	0.0016	0.0000	0.0000	0.0112	0.0016	0.0000	0.0000	0.0048
C_4	0.0000	0.0016	0.0016	0.0000	0.0000	0.0016	0.0048	0.0016	0.0016	0.0032	0.0144	0.0000	0.0032	0.0000	0.0000	0.0000	0.0000	0.0048
C_5	0.0096	0.0016	0.0000	0.0000	0.0000	0.0000	0.0064	0.0016	0.0016	0.0000	0.0000	0.0016	0.0016	0.0000	0.0032	0.0032	0.0000	0.0016
C_6	0.0288	0.0016	0.0000	0.0000	0.0000	0.0000	0.0000	0.0032	0.0032	0.0016	0.0000	0.0016	0.0016	0.0000	0.0032	0.0016	0.0064	0.0096
C_7	0.0096	0.0016	0.0032	0.0000	0.0032	0.0016	0.0096	0.0000	0.0016	0.0000	0.0032	0.0016	0.0016	0.0000	0.0016	0.0000	0.0016	0.0016
C_8	0.0000	0.0000	0.0000	0.0016	0.0128	0.0016	0.0000	0.0016	0.0000	0.0000	0.0016	0.0016	0.0016	0.0016	0.0048	0.0032	0.0016	0.0032
C_9	0.0000	0.0032	0.0016	0.0016	0.0000	0.0000	0.0064	0.0000	0.0128	0.0000	0.0000	0.0032	0.0000	0.0064	0.0016	0.0016	0.0000	0.0000
C_10	0.0096	0.0048	0.0000	0.0032	0.0032	0.0048	0.0016	0.0016	0.0016	0.0000	0.0000	0.0016	0.0000	0.0016	0.0000	0.0096	0.0000	0.0016
C_11	0.0288	0.0000	0.0032	0.0032	0.0048	0.0000	0.0032	0.0000	0.0000	0.0032	0.0000	0.0000	0.0016	0.0016	0.0000	0.0016	0.0112	0.0000
C_12	0.0096	0.0144	0.0000	0.0016	0.0000	0.0016	0.0000	0.0000	0.0016	0.0016	0.0032	0.0016	0.0000	0.0016	0.0000	0.0016	0.0016	0.0048
C_13	0.0096	0.0016	0.0000	0.0016	0.0048	0.0016	0.0000	0.0016	0.0000	0.0000	0.0000	0.0032	0.0032	0.0000	0.0128	0.0016	0.0032	0.0000
C_14	0.0576	0.0000	0.0016	0.0032	0.0000	0.0000	0.0000	0.0032	0.0016	0.0032	0.0016	0.0016	0.0000	0.0032	0.0000	0.0000	0.0032	0.0032
C_15	0.0096	0.0000	0.0032	0.0000	0.0048	0.0016	0.0016	0.0032	0.0016	0.0000	0.0032	0.0016	0.0112	0.0032	0.0000	0.0000	0.0016	0.0000
C_16	0.0000	0.0016	0.0016	0.0016	0.0000	0.0000	0.0016	0.0000	0.0048	0.0032	0.0032	0.0000	0.0016	0.0000	0.0032	0.0000	0.0000	0.0016
C_17	0.0288	0.0000	0.0000	0.0016	0.0000	0.0128	0.0000	0.0000	0.0016	0.0032	0.0016	0.0032	0.0032	0.0000	0.0016	0.0032	0.0000	0.0000
C_18	0.0096	0.0032	0.0048	0.0016	0.0000	0.0016	0.0032	0.0000	0.0000	0.0128	0.0032	0.0032	0.0016	0.0000	0.0000	0.0000	0.0032	0.0000
C_19	0.0096	0.0016	0.0032	0.0032	0.0032	0.0032	0.0000	0.0016	0.0000	0.0016	0.0016	0.0096	0.0016	0.0016	0.0016	0.0016	0.0016	0.0016

Обрахунок розподілу P(M|C)

In [12]:
```
conditional_probability_m_c = []

for m in range(0,20):
```

```
    for c in range(0,20):
        temp = probability_m_c[m][c]/probability_c[c]
        conditional_probability_m_c.append(temp)

conditional_probability_m_c=np.array(conditional_probability_m_c)

conditional_probability_m_c=pd.DataFrame(data=conditional_probability_m_c.reshape(20,20))


conditional_probability_m_c=conditional_probability_m_c.transpose()

conditional_probability_m_c_temp=copy.deepcopy(conditional_probability_m_c)
conditional_probability_m_c_temp.index=['C_'+str(i) for i in range(0,20)]
conditional_probability_m_c_temp.columns=['M_'+str(i) for i in range(0,20)]

display(conditional_probability_m_c_temp)
```

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_10	M_11	M_12	M_13
C_0	0.000000	0.040000	0.000000	0.040000	0.000000	0.040000	0.000000	0.280000	0.040000	0.040000	0.040000	0.080000	0.040000	0.080000
C_1	0.000000	0.000000	0.240000	0.080000	0.000000	0.080000	0.040000	0.080000	0.080000	0.040000	0.000000	0.000000	0.000000	0.000000
C_2	0.200000	0.033333	0.000000	0.233333	0.033333	0.000000	0.000000	0.066667	0.000000	0.033333	0.000000	0.033333	0.100000	0.100000
C_3	0.200000	0.000000	0.133333	0.000000	0.033333	0.066667	0.000000	0.066667	0.033333	0.033333	0.033333	0.000000	0.000000	0.233333
C_4	0.000000	0.040000	0.040000	0.000000	0.000000	0.040000	0.120000	0.040000	0.040000	0.080000	0.360000	0.000000	0.080000	0.000000
C_5	0.200000	0.033333	0.000000	0.000000	0.000000	0.000000	0.133333	0.033333	0.033333	0.000000	0.000000	0.033333	0.033333	0.000000
C_6	0.450000	0.025000	0.000000	0.000000	0.000000	0.000000	0.000000	0.050000	0.050000	0.025000	0.000000	0.025000	0.025000	0.000000
C_7	0.200000	0.033333	0.066667	0.000000	0.066667	0.033333	0.200000	0.000000	0.033333	0.000000	0.066667	0.033333	0.033333	0.000000
C_8	0.000000	0.000000	0.000000	0.040000	0.320000	0.040000	0.000000	0.040000	0.000000	0.000000	0.040000	0.040000	0.040000	0.040000
C_9	0.000000	0.080000	0.040000	0.040000	0.000000	0.000000	0.160000	0.000000	0.320000	0.000000	0.000000	0.080000	0.000000	0.160000
C_10	0.200000	0.100000	0.000000	0.066667	0.066667	0.100000	0.033333	0.033333	0.033333	0.000000	0.000000	0.033333	0.000000	0.033333
C_11	0.450000	0.000000	0.050000	0.050000	0.075000	0.000000	0.050000	0.000000	0.000000	0.050000	0.000000	0.000000	0.025000	0.025000
C_12	0.200000	0.300000	0.000000	0.033333	0.000000	0.033333	0.000000	0.000000	0.033333	0.033333	0.066667	0.033333	0.000000	0.033333
C_13	0.200000	0.033333	0.000000	0.033333	0.100000	0.033333	0.000000	0.033333	0.000000	0.000000	0.000000	0.066667	0.066667	0.000000
C_14	0.654545	0.000000	0.018182	0.036364	0.000000	0.000000	0.000000	0.036364	0.018182	0.036364	0.018182	0.018182	0.000000	0.036364
C_15	0.200000	0.000000	0.066667	0.000000	0.100000	0.033333	0.033333	0.066667	0.033333	0.000000	0.066667	0.033333	0.233333	0.066667
C_16	0.000000	0.040000	0.040000	0.040000	0.000000	0.000000	0.040000	0.000000	0.120000	0.080000	0.080000	0.000000	0.040000	0.000000
C_17	0.450000	0.000000	0.000000	0.025000	0.000000	0.200000	0.000000	0.000000	0.025000	0.050000	0.025000	0.050000	0.050000	0.000000
C_18	0.200000	0.066667	0.100000	0.033333	0.000000	0.033333	0.066667	0.000000	0.000000	0.266667	0.066667	0.066667	0.033333	0.000000
C_19	0.200000	0.033333	0.066667	0.066667	0.066667	0.066667	0.000000	0.033333	0.000000	0.033333	0.033333	0.200000	0.033333	0.033333

Детерміністична вирішуюча функція

In [13]:
```
def deterministic_solving_functions(matrix_M_I_C):
    deterministic_matrix=[]
    var_c=0
    while var_c!=20:
        temp=max([ matrix_M_I_C[x][var_c] for x in range(0,20)])
        position = np.where(matrix_M_I_C.iloc[var_c] == temp)
        deterministic_matrix.append(position[0][0])

        var_c+=1

    return deterministic_matrix

any_data_t=copy.deepcopy(deterministic_solving_functions(conditional_probability_m_c))
any_data=pd.DataFrame(data=deterministic_solving_functions(conditional_probability_m_c)).transpose()

any_data_temp=copy.deepcopy(any_data)
# display(any_data)
any_data_temp.index=['Deterministic function']
any_data_temp.columns=['C_'+str(i) for i in range(0,20)]
display(any_data_temp)
```

	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_10	C_11	C_12	C_13	C_14	C_15	C_16	C_17	C_18	C_19
Deterministic function	7	2	3	13	10	0	0	0	4	8	0	0	1	14	0	12	18	0	9	0

Стохастична вирішуюча функція

In [14]:
```
def stochastic_solving_functions(matrix_M_I_C):

    stochastic_matrix=np.zeros((20, 20))
```

```
    var_c=0

    while var_c!=20:
        temp=max([ matrix_M_I_C[x][var_c] for x in range(0,20)])
        for m in range(0,20):
                stochastic_matrix[var_c][m]=1 if temp==matrix_M_I_C[m][var_c] else 0

        sumation=sum([stochastic_matrix[var_c][x] for x in range(0,20)])
        if 1<sumation:
            for m in range(0,20):
                stochastic_matrix[var_c][m]= stochastic_matrix[var_c][m]/sumation

        var_c+=1

    temp=pd.DataFrame(data=stochastic_matrix)


    return temp

stochastic_matrix=stochastic_solving_functions(conditional_probability_m_c)

stochastic_matrix_t=copy.deepcopy(stochastic_matrix)

stochastic_matrix_temp=copy.deepcopy(stochastic_matrix)

stochastic_matrix_temp.index=['C_'+str(i) for i in range(0,20)]
stochastic_matrix_temp.columns=['M_'+str(i) for i in range(0,20)]

display(stochastic_matrix_temp)
```

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_10	M_11	M_12	M_13	M_14	M_15	M_16	M_17	M_18	M_19
C_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	0.0	0.0	0.0	0.0	0.0
C_1	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C_2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C_3	0.0	0.0	0.0	0.0	0.0	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
C_4	0.0	0.0	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	0.0	0.0
C_5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5
C_6	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C_7	0.5	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C_8	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C_9	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	0.0	0.0	0.0	0.0
C_10	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0
C_11	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C_12	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C_13	0.0	0.0	0.0	0.0	0.0	0.0	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
C_14	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C_15	0.0	0.0	0.0	0.0	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
C_16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
C_17	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C_18	0.0	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	0.0	0.0	0.0
C_19	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Обчислення втрат для детерміністичної функції

In [15]:
```
def losses_deterministic_functions(m_and_k,deterministic__matrix):
    # display(m_and_k)
    # display(deterministic__matrix)
    # print(m_and_k[0][2])
    # print(m_and_k[0][1])
    # print(deterministic__matrix[2])

    matrix_r=np.zeros((20,20))

    for k in range(0,20):
        for m in range(0,20):
            if m==deterministic__matrix[m_and_k[m][k]]:
                matrix_r[m_and_k[m][k]][m]=0
            else:
                matrix_r[m_and_k[m][k]][m]=1

    return pd.DataFrame(data=matrix_r)


losses_deterministic=losses_deterministic_functions(table,any_data_t)
losses_deterministic_temp=copy.deepcopy(losses_deterministic)
```

```
losses_deterministic_temp.index=['C_'+str(i) for i in range(0,20)]
losses_deterministic_temp.columns=['M_'+str(i) for i in range(0,20)]

display(losses_deterministic_temp)
```

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_10	M_11	M_12	M_13	M_14	M_15	M_16	M_17	M_18	M_19
C_0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0
C_1	0.0	0.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0
C_2	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0
C_3	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0
C_4	0.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0
C_5	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0
C_6	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0
C_7	0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0
C_8	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0
C_9	0.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C_10	0.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
C_11	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0
C_12	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0
C_13	1.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0
C_14	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0
C_15	1.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0
C_16	0.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0
C_17	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0
C_18	1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
C_19	0.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0

Обчислення втрат для стохастичної функції

In [16]:
```
# Стох функ втрат - L(Ci, Mj) = сума(стох_функ(Ci, Мл)), де л = 0, ...19; л!= j

def losses_stochastic_functions(stochastic_matrix_temp):
    # display(stochastic_matrix_temp) # reversed
    matrixes=np.zeros((20,20))
    for c in range(0,20):
        for m in range(0,20):
            matrixes[c][m]=sum([stochastic_matrix_temp[x][c] for x in range(0,20) if m!=x])

    return pd.DataFrame(data=matrixes)

t=losses_stochastic_functions(stochastic_matrix_t)

t_temp=copy.deepcopy(t)

t_temp.index=['C_'+str(i) for i in range(0,20)]
t_temp.columns=['M_'+str(i) for i in range(0,20)]

display(t_temp)
```

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_10	M_11	M_12	M_13	M_14	M_15	M_16	M_17	M_18	M_19
C_0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_1	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_2	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0
C_4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_5	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5
C_6	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_7	0.5	1.0	1.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_8	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_10	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0
C_11	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_12	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_13	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0
C_14	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_15	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_16	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0
C_17	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_18	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C_19	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Середні втрати для детерміністичної функції

In [17]:
```
def mean_losses_determinating(M_C,losses_determin):
    temp=0
    for c in range(0,20):
        for m in range(0,20):
            temp=temp+M_C[c][m]*losses_determin[c][m]

    return temp

print(mean_losses_determinating(probability_m_c,losses_deterministic))
```

```
0.6703999999999999
```

Середні втрати для стохастичної функції

In [18]:
```
def mean_losses_stochastic(M_C,losses_stochast):
    temp=0
    for c in range(0,20):
        for m in range(0,20):
            temp=temp+M_C[c][m]*losses_stochast[c][m]

    return temp

print(mean_losses_determinating(probability_m_c,losses_deterministic))
```

```
0.6703999999999999
```