```
In [ ]:
```

In [1]:
```
hideMe="Yes" # hide this cell from show in Jupyter notebook
%load_ext tikzmagic
#from __future__ import print_function
import tikzmagic

from IPython.display import display, Math, Markdown, Latex
import numpy as np

def stable_matrix(P0, last, printYes = None):
    P_next = np.copy(P0)
    for i in range(1,last+1):
        P_next = P_next.dot(P0)
    return P_next

def printMatrixs(matrixS, width=8, prec=6):
    raw_counts = max([len(a) for a in matrixS])
    for i in range(0,raw_counts):
        for A in matrixS:
            if len(A)>=i: printMatrix(A,i, width, prec)
        print()

def printMatrix(matrixA,raw_current, width, prec):
    print("[", end = "") #spec = "{:<"+str(col_width)+"G}"
    for j in range (len(matrixA[raw_current])):
        col_width = max([len("{:g}".format(a)) for a in  matrixA[:,j]])
        if col_width <= width:
            if j>0: print(" ",end = "")
            print( ("{:<"+str(col_width)+"g}").format(matrixA[raw_current][j]), end = "")
        else:
            if j>0: print(" ",end = "")
            print( ("{:"+str(width)+"."+str(prec)+"f}").format(matrixA[raw_current][j]), e
    print("]", end = "")
####################################################
import notebook
from jupyter_core.paths import jupyter_config_dir, jupyter_config_path
print(jupyter_config_dir())
print(jupyter_config_path())
#help("modules")
import sys
import os
print('\n'.join(sys.path), "\ncurrent folder ==",os.getcwd())
#https://sites.google.com/site/kochiuyu/Tikz
#%https://share.cocalc.com/share/96fd2324ae3de4c1f97ef1a116a87fd0839c3c2b/tikzimpatient.ip
```

```
C:\Users\polit\.jupyter
['C:\\Users\\polit\\.jupyter', 'd:\\html_doc\\math\\probability\\markov_chains\\env\\e
tc\\jupyter', 'C:\\ProgramData\\jupyter']
D:\HTML_DOC\Math\Probability\Markov_Chains\env
C:\Program Files\Python38\python38.zip
C:\Program Files\Python38\DLLs
C:\Program Files\Python38\lib
C:\Program Files\Python38
d:\html_doc\math\probability\markov_chains\env

d:\html_doc\math\probability\markov_chains\env\lib\site-packages
d:\html_doc\math\probability\markov_chains\env\lib\site-packages\pip-20.2b1-py3.8.egg
d:\html_doc\math\probability\markov_chains\env\lib\site-packages\win32
d:\html_doc\math\probability\markov_chains\env\lib\site-packages\win32\lib
```

```
d:\html_doc\math\probability\markov_chains\env\lib\site-packages\Pythonwin
d:\html_doc\math\probability\markov_chains\env\lib\site-packages\IPython\extensions
C:\Users\polit\.ipython
current folder == D:\HTML_DOC\Math\Probability\Markov_Chains\env
```

Invertible matrix (https://en.wikipedia.org/wiki/Invertible_matrix)

Обратная матрица (https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0)

Обратная матрица - такая матрица A^{-1}, при умножении на которую, исходная матрица A даёт в результате единичную матрицу E

$$A \cdot A^{-1} = A^{-1} \cdot A = E$$

Как найти обратную матрицу? (http://www.mathprofi.ru/kak_naiti_obratnuyu_matricu.html)

ПРИСОЕДИНЕННАЯ МАТРИЦА (взаимная матрица) к квадратной матрице A^*— матрица, в которой вместо каждого элемента a_{ij} поставлено его алгебраическое дополнение a_{ij}, а затем матрица транспонирована. Для матрицы n-го порядка:

$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$ Присоединённая матрица будет :

$A^* = \begin{bmatrix} A_{\color{red}{11}} & A_{\color{red}{21}} & A_{\color{red}{31}} & \dots & A_{\color{red}{n1}} \\ A_{\color{red}{12}} & A_{22} & A_{32} & \dots & A_{n2} \\ A_{\color{red}{13}} & A_{23} & A_{33} & \dots & A_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{\color{red}{1n}} & A_{2n} & A_{3n} & \dots & A_{nn} \end{bmatrix}$

Произведение матрицы A на A^* равно скалярной матрице, у которой главной диагонали стоит определитель $D = det A$:

$$A^* \cdot A = A \cdot A^* = \begin{bmatrix} D & 0 & 0 & \dots & 0 \\ 0 & D & 0 & \dots & 0 \\ 0 & 0 & D & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & D \end{bmatrix}$$

Поэтому обратная матрица A^{-1} к невырожденной матрице A выражается так:

$$A^{-1} = D^{-1} \cdot A^* = \frac{A^*}{det A}$$

In [2]:
```python
P0 = np.array([[1, 0, 7],
               [5, 6, 8],
               [2, 3, 4]])
print(P0)
print("{:G}".format(np.linalg.det(P0)))
```

```
[[1 0 7]
 [5 6 8]
 [2 3 4]]
21
```

[Как вычислить определитель? (http://www.mathprofi.ru/kak_vychislit_opredelitel.html)](http://www.mathprofi.ru/kak_vychislit_opredelitel.html)

$$\det A = \Delta = \begin{vmatrix} 1 & 0 & 7 \\ 5 & 6 & 8 \\ 2 & 3 & 4 \end{vmatrix} = ???$$

Разложение по строке:

$$\det A = \Delta = \begin{vmatrix} 1 & 0 & 7 \\ 5 & 6 & 8 \\ 2 & 3 & 4 \end{vmatrix} = 1 \cdot (-1)^{1+1} \begin{vmatrix} 6 & 8 \\ 3 & 4 \end{vmatrix} - 0 \cdot (-1)^{1+2} \begin{vmatrix} 5 & 8 \\ 2 & 4 \end{vmatrix} + 7 \cdot (-1)^{1+3} \begin{vmatrix} 5 & 6 \\ 2 & 3 \end{vmatrix} =$$

$$= 1(24 - 24) - 0(20 - 16) + 7(15 - 12) = 0 - 0 + 21 = 21$$

Правило Треугольника:

$$\det A = \Delta = \begin{vmatrix} 1 & 0 & 7 \\ 5 & 6 & 8 \\ 2 & 3 & 4 \end{vmatrix} = (1 \cdot 6 \cdot 4) + (2 \cdot 0 \cdot 8) + (7 \cdot 3 \cdot 5) - (2 \cdot 6 \cdot 7) - (1 \cdot 8 \cdot 3) - (4 \cdot 0 \cdot 5) =$$

$$= 24 + 0 + 105 - 84 - 24 - 0 = 21$$

$$\det A = \Delta = \begin{vmatrix} 1 & 0 & 7 \\ 5 & 6 & 8 \\ 2 & 3 & 4 \end{vmatrix} = 21$$

In [3]:
```python
P0 = np.array([[ 1,-2, 3],
               [ 4, 0, 6],
               [-7, 8, 9]])
print(P0)
print("{:G}".format(np.linalg.det(P0)))
```

```
[[ 1 -2  3]
 [ 4  0  6]
 [-7  8  9]]
204
```

$$\det A = \Delta = \begin{vmatrix} 1 & -2 & 3 \\ 4 & 0 & 6 \\ -7 & 8 & 9 \end{vmatrix} = ???$$

Разложение по строке:

Итак, определитель «три на три» сводится к решению трёх маленьких определителей - **МИНОРОВ**.

Коль скоро выбран способ разложения определителя по первой строке, очевидно, что всё вращается вокруг неё:

$$\begin{vmatrix} 1 & -2 & 3 \\ 4 & 0 & 6 \\ -7 & 8 & 9 \end{vmatrix}$$

1) Из матрицы знаков выписываем соответствующий знак "+": $\begin{vmatrix} + & - & + \\ - & + & - \\ + & - & + \end{vmatrix}$

2) Затем записываем сам элемент: $\begin{vmatrix} 1 & -2 & 3 \\ 4 & 0 & 6 \\ -7 & 8 & 9 \end{vmatrix} = +1$

3) МЫСЛЕННО вычеркиваем строку и столбец, в котором стоит первый элемент. Оставшиеся четыре числа и образуют определитель «два на два», который называется **МИНОРОМ** данного элемента $a_{11} = 1$(единицы):

$$\begin{vmatrix} 1 & -2 & 3 \\ 4 & 0 & 6 \\ -7 & 8 & 9 \end{vmatrix} = +1 \begin{vmatrix} 0 & 6 \\ 8 & 9 \end{vmatrix}$$

4) Из матрицы знаков выписываем соответствующий знак "-": $\begin{vmatrix} + & - & + \\ - & + & - \\ + & - & + \end{vmatrix}$

5) Затем записываем второй элемент: $\begin{vmatrix} 1 & -2 & 3 \\ 4 & 0 & 6 \\ -7 & 8 & 9 \end{vmatrix} = +1 \begin{vmatrix} 0 & 6 \\ 8 & 9 \end{vmatrix} - (-2)$

6) МЫСЛЕННО вычеркиваем строку и столбец, в котором стоит второй элемент. Оставшиеся четыре числа записываем в маленький определитель.

$$\begin{vmatrix} 1 & -2 & 3 \\ 4 & 0 & 6 \\ -7 & 8 & 9 \end{vmatrix} = +1 \begin{vmatrix} 0 & 6 \\ 8 & 9 \end{vmatrix} - (-2) \begin{vmatrix} 4 & 6 \\ -7 & 9 \end{vmatrix}$$

7) Третий элемент первой строки по подобию предыдущих. Из матрицы знаков выписываем соответствующий знак: $\begin{vmatrix} + & - & + \\ - & + & - \\ + & - & + \end{vmatrix}$

8) Записываем третий элемент:

$$\begin{vmatrix} 1 & -2 & \color{red}{3} \\ 4 & 0 & 6 \\ -7 & 8 & 9 \end{vmatrix} = +1\begin{vmatrix} 0 & 6 \\ 8 & 9 \end{vmatrix} - (-2)\begin{vmatrix} 4 & 6 \\ -7 & 9 \end{vmatrix} + \color{red}{3}$$

9) МЫСЛЕННО вычеркиваем строку и столбец, в котором стоит третий элемент.Оставшиеся четыре числа записываем в маленький определитель:

$$\begin{vmatrix} \color{purple}{1} & \color{purple}{-2} & \color{red}{3} \\ 4 & 0 & \color{purple}{6} \\ -7 & 8 & \color{purple}{9} \end{vmatrix} = +1\begin{vmatrix} 0 & 6 \\ 8 & 9 \end{vmatrix} - (-2)\begin{vmatrix} 4 & 6 \\ -7 & 9 \end{vmatrix} + \color{red}{3}\begin{vmatrix} 4 & 0 \\ -7 & 8 \end{vmatrix}$$

In [4]:
```python
def matrix_cofactor(matrix):
    return np.linalg.inv(matrix).T * np.linalg.det(matrix)

P0 = np.array([[ 1,-2, 3],
               [ 4, 0, 6],
               [-7, 8, 9]])
P_M = matrix_cofactor(P0)
print(P_M)
```

```
[[-48. -78.  32.]
 [ 42.  30.   6.]
 [-12.   6.   8.]]
```

[Минор и алгебраическое дополнение матрицы. (https://ru.onlinemschool.com/math/library/matrix/minors/)](https://ru.onlinemschool.com/math/library/matrix/minors/)

Минором M_{ij} к элементу a_{ij} определителя n-го порядка называется определитель $(n-1)$-го порядка, полученный из исходного определителя вычеркиванием i-той строки и j-того столбца.

Найти миноры матрицы $A = \begin{vmatrix} 1 & -2 & 3 \\ 4 & 0 & 6 \\ -7 & 8 & 9 \end{vmatrix}$

$$M_{11} = \begin{vmatrix} 0 & 6 \\ 8 & 9 \end{vmatrix} = 0 \cdot 9 - 6 \cdot 8 = 0 - 48 = -48$$

$$M_{12} = \begin{vmatrix} 4 & 6 \\ -7 & 9 \end{vmatrix} = 4 \cdot 9 - 6 \cdot (-7) = 36 + 42 = 78$$

$$M_{13} = \begin{vmatrix} 4 & 0 \\ -7 & 8 \end{vmatrix} = 4 \cdot 8 - 0 \cdot (-7) = 32 - 0 = 32$$

$$M_{21} = \begin{vmatrix} -2 & 3 \\ 8 & 9 \end{vmatrix} = (-2) \cdot 9 - 3 \cdot 8 = -18 - 24 = -42$$

$$M_{22} = \begin{vmatrix} 1 & 3 \\ -7 & 9 \end{vmatrix} = 1 \cdot 9 - 3 \cdot (-7) = 9 + 21 = 30$$

$$M_{23} = \begin{vmatrix} 1 & -2 \\ -7 & 8 \end{vmatrix} = 1 \cdot 8 - (-2) \cdot (-7) = 8 - 14 = -6$$

$$M_{31} = \begin{vmatrix} -2 & 3 \\ 0 & 6 \end{vmatrix} = (-2) \cdot 6 - 3 \cdot 0 = -12 - 0 = -12$$

$$M_{32} = \begin{vmatrix} 1 & 3 \\ 4 & 6 \end{vmatrix} = 1 \cdot 6 - 3 \cdot 4 = 6 - 12 = -6$$

$$M_{33} = \begin{vmatrix} 1 & -2 \\ 4 & 0 \end{vmatrix} = 1 \cdot 0 - (-2) \cdot 4 = 0 + 8 = 8$$

Матрица Миноров:

$$M = \begin{vmatrix} -48 & 78 & 32 \\ -42 & 30 & -6 \\ -12 & -6 & 8 \end{vmatrix}$$

In [5]:
```
def matrix_cofactor(matrix):
    return np.linalg.inv(matrix).T * np.linalg.det(matrix)

P0 = np.array([[ 1,-2, 3],
               [ 4, 0, 6],
               [-7, 8, 9]])
print("Determinant == {:G}".format(np.linalg.det(P0)))

P_M = matrix_cofactor(P0)
printMatrixs([P0,P_M], width=3, prec=0)
print("---raw---------")
for i in range(P0.shape[1]):
    print(P0[i,:],P_M[i,:] )
    print(" =={:G}".format(np.dot(P_M[:,i],P0[:,i])))
print("---column------")
for i in range(P0.shape[0]):
    print(P0[:,i],P_M[:,i] )
    print(" =={:G}".format(np.dot(P_M[:,i],P0[:,i])))

for i,j in [[0,1],[0,2],[1,2],]:
    print("raw[{}]*raw[{}]      ==".format(i,j), end ="")
    print("{:0.0f}".format(np.dot(P_M[i,:],P0[j,:])))
    print(P0[i,:],P_M[j,:] );
for i,j in [[0,1],[0,2],[1,2],]:
    print("column[{}]*column[{}]==".format(i,j), end="")
    print("{:0.0f}".format(np.dot(P_M[:,i],P0[:,j])))
    print(P0[:,i],P_M[:,j]);
```

```
Determinant == 204
[1  -2 3][-48 -78 32]
[4  0  6][42  30  6 ]
[-7 8  9][-12 6   8 ]
---raw---------
[ 1 -2  3] [-48. -78.  32.]
 ==204
[4 0 6] [42. 30.  6.]
 ==204
[-7  8  9] [-12.   6.   8.]
 ==204
---column------
[ 1  4 -7] [-48.  42. -12.]
 ==204
[-2  0  8] [-78.  30.   6.]
 ==204
[3 6 9] [32.  6.  8.]
 ==204
raw[0]*raw[1]      ==0
[ 1 -2  3] [42. 30.  6.]
raw[0]*raw[2]      ==-0
[ 1 -2  3] [-12.   6.   8.]
raw[1]*raw[2]      ==0
[4 0 6] [-12.   6.   8.]
column[0]*column[1]==0
[ 1  4 -7] [-78.  30.   6.]
column[0]*column[2]==0
[ 1  4 -7] [32.  6.  8.]
column[1]*column[2]==0
[-2  0  8] [32.  6.  8.]
```

[Алгебраическое дополнение (https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%)

Нахождение дополнительного минора и алгебраического дополнения

Алгебраическим дополнением элемента a_{ij} матрицы $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$

называется число: $A_{ij} = (-1)^{i+j} M_{ij}$,

где M_{ij} — **дополнительный минор/определитель** матрицы, получающейся из исходной матрицы A путём вычёркивания i-й строки и j-го столбца.

Свойства

Алгебраическое дополнение элемента — это коэффициент, с которым этот самый элемент входит в определитель матрицы. Это утверждается следующей теоремой:

Теорема (о разложении определителя по строке/столбцу). Определитель матрицы A может быть представлен в виде суммы:

$$\det A = \sum_{j=1}^{n} a_{ij} A_{ij} = \sum_{i=1}^{n} a_{ij} A_{ij}$$

Для алгебраического дополнения справедливо следующее утверждение:

Лемма о фальшивом разложении определителя. Сумма произведений элементов одной строки (столбца) на соответствующие алгебраические дополнения элементов другой строки (соответственно столбца) равна нулю, то есть $\sum_{j=1}^{n} a_{i_1 j} A_{i_2 j} = \sum_{i=1}^{n} a_{i j_1} A_{i j_2} = 0$ при $i_1 \neq i_2$ и $j_1 \neq j_2$.

Из этих утверждений следует алгоритм нахождения **обратной** матрицы:

- заменить каждый элемент исходной матрицы на его алгебраическое дополнение,
- транспонировать полученную матрицу - в результате будет получена **союзная матрица**,
- разделить каждый элемент союзной матрицы на определитель исходной матрицы.

$$A = \begin{vmatrix} 1 & -2 & 3 \\ 4 & 0 & 6 \\ -7 & 8 & 9 \end{vmatrix}$$

Матрица Миноров:

$$M = \begin{vmatrix} -48 & 78 & 32 \\ -42 & 30 & -6 \\ -12 & -6 & 8 \end{vmatrix}$$

Матрица **Алгебраических** дополнений $A_{ij} = (-1)^{i+j} M_{ij}$. где M_{ij} — **дополнительный минор/определитель** матрицы.

$$\begin{aligned}
A_{11} &= (-1)^{1+1} \cdot M_{11} &&= (-1)^2 \cdot \begin{vmatrix} 0 & 6 \\ 8 & 9 \end{vmatrix} &&= (+1)(0-48) &&= -48 \\
A_{12} &= (-1)^{1+2} \cdot M_{12} &&= (-1)^3 \cdot \begin{vmatrix} 4 & 6 \\ -7 & 9 \end{vmatrix} &&= (-1)(36+42) &&= -78 \\
A_{13} &= (-1)^{1+3} \cdot M_{13} &&= (-1)^4 \cdot \begin{vmatrix} 4 & 0 \\ -7 & 8 \end{vmatrix} &&= (+1)(32-0) &&= 32 \\
A_{21} &= (-1)^{2+1} \cdot M_{21} &&= (-1)^3 \cdot \begin{vmatrix} -2 & 3 \\ 8 & 9 \end{vmatrix} &&= (-1)(-18-24) &&= 42 \\
A_{22} &= (-1)^{2+2} \cdot M_{22} &&= (-1)^4 \cdot \begin{vmatrix} 1 & 3 \\ -7 & 9 \end{vmatrix} &&= (+1)(9+21) &&= 30 \\
A_{23} &= (-1)^{2+3} \cdot M_{23} &&= (-1)^5 \cdot \begin{vmatrix} 1 & -2 \\ -7 & 8 \end{vmatrix} &&= (-1)(8-14) &&= 6 \\
A_{31} &= (-1)^{3+1} \cdot M_{31} &&= (-1)^4 \cdot \begin{vmatrix} -2 & 3 \\ 0 & 6 \end{vmatrix} &&= (+1)(-12-0) &&= -12 \\
A_{32} &= (-1)^{3+2} \cdot M_{32} &&= (-1)^5 \cdot \begin{vmatrix} 1 & 3 \\ 4 & 6 \end{vmatrix} &&= (-1)(6-12) &&= 6 \\
A_{33} &= (-1)^{3+3} \cdot M_{33} &&= (-1)^6 \cdot \begin{vmatrix} 1 & -2 \\ 4 & 0 \end{vmatrix} &&= (+1)(0+8) &&= 8
\end{aligned}$$

[Cofactor matrix (https://en.wikipedia.org/wiki/Minor_(linear_algebra)#Applications_of_minors_an](https://en.wikipedia.org/wiki/Minor_(linear_algebra)#Applications_of_minors_an)

[Adjugate matrix (https://en.wikipedia.org/wiki/Adjugate_matrix#3_%C3%97_3_generic_matrix)](https://en.wikipedia.org/wiki/Adjugate_matrix#3_%C3%97_3_generic_matrix)

Cofactor matrix: $C = \begin{vmatrix} -48 & -78 & 32 \\ 42 & 30 & 6 \\ -12 & 6 & 8 \end{vmatrix}$

In [6]:
```
P0 = np.array([[ 1,-2, 3],
               [ 4, 0, 6],
               [-7, 8, 9]])
print("Determinant == {:G}".format(np.linalg.det(P0)))

P_M = matrix_cofactor(P0)
printMatrixs([P_M], width=3, prec=0)
```

```
Determinant == 204
[-48 -78 32]
[42  30  6 ]
[-12 6   8 ]
```

[*Находим транспонированную матрицу алгебраических дополнений (http://www.mathprofi.ru/kak_naiti_obratnuyu_matricu.html)*](http://www.mathprofi.ru/kak_naiti_obratnuyu_matricu.html)

[*Inverse of a matrix (https://en.wikipedia.org/wiki/Minor_(linear_algebra)#Cofactor_expansion_of_the_determinant##Inverse_*](https://en.wikipedia.org/wiki/Minor_(linear_algebra)#Cofactor_expansion_of_the_determinant##Inverse_)

[*Adjugate matrix (https://en.wikipedia.org/wiki/Adjugate_matrix#3_%C3%97_3_generic_matrix)*](https://en.wikipedia.org/wiki/Adjugate_matrix#3_%C3%97_3_generic_matrix)

[Присоединённая матрица (https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D1%81%D0%BE%D0%E](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D1%81%D0%BE%D0%E)

Присоединённая (союзная, взаимная) матрица — матрица ${displaystyle {C}^{*}}$, составленная из алгебраических дополнений для соответствующих элементов транспонированной матрицы. Из определения следует, что присоединённая матрица рассматривается только для квадратных матриц и сама является квадратной, так как понятие алгебраического дополнения вводится для квадратных матриц.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

Присоединённая матрица будет : $A^* = \begin{bmatrix} A_{11} & A_{21} & A_{31} & \dots & A_{n1} \\ A_{12} & A_{22} & A_{32} & \dots & A_{n2} \\ A_{13} & A_{23} & A_{33} & \dots & A_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{1n} & A_{2n} & A_{3n} & \dots & A_{nn} \end{bmatrix}$

$$C^* = \begin{vmatrix} -48 & -78 & 32 \\ 42 & 30 & 6 \\ -12 & 6 & 8 \end{vmatrix}^T = \begin{vmatrix} -48 & 42 & -12 \\ -78 & 30 & 6 \\ 32 & 6 & 8 \end{vmatrix}$$

In [7]:
```
P0 = np.array([[ 1,-2, 3],
               [ 4, 0, 6],
               [-7, 8, 9]])

printMatrixs([P0], width=2, prec=0)
print("Determinant == {:G}".format(np.linalg.det(P0)))

P_M = matrix_cofactor(P0)
print("matrix Cofactor:"); printMatrixs([P_M], width=3, prec=0)

P_MT = P_M.T
print("Присоединённая (союзная, взаимная) матрица \nAdjugate matrix:"); printMatrixs([P_MT
```

```
[1  -2 3]
[4  0  6]
[-7 8  9]
Determinant == 204
matrix Cofactor:
[-48 -78 32]
[42  30  6 ]
[-12 6   8 ]
Присоединённая (союзная, взаимная) матрица
Adjugate matrix:
[-48 42 -12]
[-78 30 6  ]
[32  6  8  ]
```

Произведение матрицы A на A^* равно скалярной матрице, у которой главной диагонали стоит определитель $D = det A$:

$$A^* \cdot A = A \cdot A^* = \begin{bmatrix} D & 0 & 0 & \dots & 0 \\ 0 & D & 0 & \dots & 0 \\ 0 & 0 & D & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & D \end{bmatrix}$$

In [8]:
```
P0 = np.array([[ 1,-2, 3],
               [ 4, 0, 6],
               [-7, 8, 9]])

printMatrixs([P0], width=2, prec=0)
print("Determinant == {:G}".format(np.linalg.det(P0)))

P_M = matrix_cofactor(P0)
P_MT = P_M.T
print("Присоединённая (союзная, взаимная) матрица \nAdjugate matrix:"); printMatrixs([P_MT

print("A * A'=")
P_D = P0.dot(P_MT)
printMatrixs([P_D], width=3, prec=0)
```

```
[1  -2 3]
[4  0  6]
[-7 8  9]
Determinant == 204
Присоединённая (союзная, взаимная) матрица
Adjugate matrix:
[-48 42 -12]
[-78 30 6  ]
[32  6  8  ]
A * A'=
[204  -0  -0]
[  0 204   0]
[ -0   0 204]
```

$$A \cdot A^* = \begin{vmatrix} 1 & -2 & 3 \\ 4 & 0 & 6 \\ -7 & 8 & 9 \end{vmatrix} \cdot \begin{vmatrix} -48 & 42 & -12 \\ -78 & 30 & 6 \\ 32 & 6 & 8 \end{vmatrix} = \begin{vmatrix} 204 & 0 & 0 \\ 0 & 204 & 0 \\ 0 & 0 & 204 \end{vmatrix}$$

Поэтому обратная матрица A^{-1} к невырожденной матрице A выражается так:

$$A^{-1} = D^{-1} \cdot A^* = \frac{A^*}{det A}$$

$$A^* = \begin{bmatrix} A_{\color{red}{11}} & A_{\color{red}{21}} & A_{\color{red}{31}} & \ldots & A_{\color{red}{n1}} \\ A_{\color{red}{12}} & A_{22} & A_{32} & \ldots & A_{n2} \\ A_{\color{red}{13}} & A_{23} & A_{33} & \ldots & A_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{\color{red}{1n}} & A_{2n} & A_{3n} & \ldots & A_{nn} \end{bmatrix}$$

$$A^* = \begin{vmatrix} -48 & 42 & -12 \\ -78 & 30 & 6 \\ 32 & 6 & 8 \end{vmatrix}$$

$$A^{-1} = 204^{-1} \cdot \begin{vmatrix} -48 & 42 & -12 \\ -78 & 30 & 6 \\ 32 & 6 & 8 \end{vmatrix} = \frac{\begin{vmatrix} -48 & 42 & -12 \\ -78 & 30 & 6 \\ 32 & 6 & 8 \end{vmatrix}}{204} =$$

$$A^{-1} = \begin{vmatrix} -\frac{4}{17} & \frac{7}{34} & -\frac{1}{17} \\ -\frac{13}{34} & \frac{5}{34} & \frac{1}{34} \\ \frac{8}{51} & \frac{1}{34} & \frac{2}{51} \end{vmatrix} = \begin{vmatrix} -0.235 & 0.206 & -0.059 \\ -0.382 & 0.147 & 0.029 \\ 0.157 & 0.029 & 0.039 \end{vmatrix}$$

In [9]:
```
P0 = np.array([[ 1,-2, 3],
               [ 4, 0, 6],
               [-7, 8, 9]])
printMatrixs([P0], width=2, prec=0)
Determinant = np.linalg.det(P0)
P_M = matrix_cofactor(P0)
P_MT = P_M.T

A_inv = P_MT/Determinant
printMatrixs([A_inv], width=6, prec=3)
A_inv_frac = np.array([[-4/17,  7/34,-1/17],
                       [-13/34, 5/34,  1/34],
                       [8/51,   1/34,  2/51]])
printMatrixs([A_inv_frac, A_inv_frac-A_inv], width=8, prec=5)
```

```
[1  -2 3]
[4  0  6]
[-7 8  9]
[-0.235  0.206 -0.059]
[-0.382  0.147  0.029]
[ 0.157  0.029  0.039]
[-0.23529  0.20588 -0.05882][-0.00000 0 -0.00000]
[-0.38235  0.14706  0.02941][ 0.00000 0 -0.00000]
[ 0.15686  0.02941  0.03922][ 0.00000 0  0.00000]
```

Проверка: $A \cdot A^{-1} = \begin{vmatrix} 1 & -2 & 3 \\ 4 & 0 & 6 \\ -7 & 8 & 9 \end{vmatrix} \cdot \begin{vmatrix} -\frac{4}{17} & \frac{7}{34} & -\frac{1}{17} \\ -\frac{13}{34} & \frac{5}{34} & \frac{1}{34} \\ \frac{8}{51} & \frac{1}{34} & \frac{2}{51} \end{vmatrix} =$

$$= \left| \begin{array}{c:c:c} 1 \cdot (-\frac{4}{17}) - 2 \cdot (-\frac{13}{34}) + 3 \cdot \frac{8}{51} & 1 \cdot \frac{7}{34} - 2 \cdot \frac{5}{34} + 3 \cdot \frac{1}{34} & 1 \cdot (-\frac{1}{17}) - 2 \cdot \frac{1}{34} + 3 \cdot \frac{2}{51} \\ 4 \cdot (-\frac{4}{17}) + 0 \cdot (-\frac{13}{34}) + 6 \cdot \frac{8}{51} & 4 \cdot \frac{7}{34} + 0 \cdot \frac{5}{34} + 6 \cdot \frac{1}{34} & 4 \cdot (-\frac{1}{17}) + 0 \cdot \frac{1}{34} + 6 \cdot \frac{2}{51} \\ (-7) \cdot (-\frac{4}{17}) + 8 \cdot (-\frac{13}{34}) + 9 \cdot \frac{8}{51} & (-7) \cdot \frac{7}{34} + 8 \cdot \frac{5}{34} + 9 \cdot \frac{1}{34} & (-7) \cdot (-\frac{1}{17}) + 8 \cdot \frac{1}{34} + 9 \cdot \frac{2}{51} \end{array} \right|$$

$$= \left| \begin{array}{c:c:c} -\frac{4}{17} + \frac{13}{17} + \frac{8}{17} & \frac{7}{34} - \frac{10}{34} + \frac{3}{34} & -\frac{1}{17} - \frac{1}{17} + \frac{2}{17} \\ -\frac{16}{17} + \frac{16}{17} & \frac{14}{17} + \frac{3}{17} & -\frac{4}{17} + \frac{4}{17} \\ \frac{28}{17} - \frac{52}{17} + \frac{24}{17} & -\frac{49}{34} + \frac{40}{34} + \frac{9}{34} & \frac{7}{17} + \frac{4}{17} + \frac{6}{17} \end{array} \right| = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

$$A \cdot A^{-1} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

In [10]:
```
P0 = np.array([[ 1,-2, 3],
               [ 4, 0, 6],
               [-7, 8, 9]])
printMatrixs([P0], width=2, prec=0)
Determinant = np.linalg.det(P0)
P_M = matrix_cofactor(P0)
P_MT = P_M.T

A_inv = P_MT/Determinant
printMatrixs([A_inv], width=6, prec=3)
A_inv_frac = np.array([[-4/17,  7/34,-1/17],
                       [-13/34, 5/34,  1/34],
                       [8/51,   1/34,  2/51]])
printMatrixs([P0.dot(A_inv_frac)], width=2, prec=0)
printMatrixs([P0.dot(A_inv)], width=2, prec=0)
```

```
[1  -2 3]
[4  0  6]
[-7 8  9]
[-0.235  0.206 -0.059]
[-0.382  0.147  0.029]
[ 0.157  0.029  0.039]
[ 1 -0 0]
[ 0  1 0]
[ 0  0 1]
[ 1 -0  0]
[ 0  1  0]
[-0  0  1]
```

In [ ]: