1. Let C_n be the graph consisting of n vertices all arranged in an n-gon. So C_3 is a triangle, C_4 is a square, C_5 is a pentagon, etc.

(a) For $n = 3, 4, 5$ find the eigenvalues of the adjacency matrix of Cn and their multiplicities.

In [2]:
```python
import numpy as np
import sympy as sy
```

In [3]:
```python
A = np.array([[0, 1, 1],
              [1, 0, 1],
              [1, 1, 0]])
A = sy.Matrix(A)
# A.jordan_form()
A.eigenvals()
```

Out[3]:
```
{2: 1, -1: 2}
```

So eigenvalue 2 appears once, eigenvalue -1 appears twice. I also calculated this by hand, and used sympy to view the jordan form.

In [7]:
```python
A = np.array([[0, 1, 1, 0],
              [1, 0, 0, 1],
              [1, 0, 0, 1],
              [0, 1, 1, 0]])
A = sy.Matrix(A)
# A.jordan_form()
A. eigenvals()
```

Out[7]:
```
{-2: 1, 2: 1, 0: 2}
```

Again eigenvalue -2 apears once. Eigenvalue 2 apears once. Eigenvalue 0 apears twice.

In [9]:
```python
A = np.array([[0, 1, 0, 0, 1],
              [1, 0, 1, 0, 0],
              [0, 1, 0, 1, 0],
              [0, 0, 1, 0, 1],
              [1, 0, 0, 1, 0]])
A = sy.Matrix(A)
# A.jordan_form()
A.eigenvals()
```

Out[9]:
```
{2: 1, -sqrt(5)/2 - 1/2: 2, -1/2 + sqrt(5)/2: 2}
```

Eigenvalue 2 apears once. Eigenvalue $\frac{\sqrt{5}}{2} - \frac{1}{2}$ apears twice. Eigenvalue $-\frac{1}{2} + \frac{\sqrt{5}}{2}$ apears twice.

(b) For which values of n will -2 be an eigenvalue of the adjacency matrix of C_n? Explain your answer.

Well, from the answer above I expect -2 to apear for even values of n in 2-regular graphs with n vertices. These n-gons are 2-regular graphs. I do not have an explination yet.

After reading the notes I've fouynd that a -k eigenvalue apears when there is a bipartite component. See question 5.

(c) For $n = 4$, how many paths of length 100 are there that start and end at vertex 1?

In [10]:
```
A = np.array([[0, 1, 0, 1],
              [1, 0, 1, 0],
              [0, 1, 0, 1],
              [1, 0, 1, 0]])
A = sy.Matrix(A)
P, D = A.jordan_form()

A100 = A**100
A100[1,1]
```

Out[10]: $633825300114114700748351602688$

I've diagonalized this matrix as a first step.If I was doing this by hand I might want to raise the diagonalized matrix to the 100'th power. But with the computer I can caclulate A^{100} directly. Each element of A^{100} represents the paths from i to j, where i, j are the rows and columns of A^{100} respectivly.

(d) For $n = 4$, what is the probability that a random path of length 100 that starts at vertex 1 will end at vertex 3? If the length of the path gets very big, what number do you expect this probability to approach? Explain.

In [16]:
```
A = np.array([[0, 1, 0, 1],
              [1, 0, 1, 0],
              [0, 1, 0, 1],
              [1, 0, 1, 0]])
A = sy.Matrix(A)
toss, D = A.jordan_form()

A100 = A**100
p = A100[1,3] / sum(A100)
p = sy.Float(p)
# convert to percent
p * 100
```

Out[16]: 12.5

I find that there is a 12% chance. Maybe I'm doing this wrong but I do not expect this value to change as the power of A ingreases.

2. Consider the following two graphs: G_1 is the graph consisting of four vertices with edges forming a square (so, so far we have four vertices and four edges), together with one more vertex in the middle of the square that is connected to nothing. The graph G_2 is on the same vertices, and it has 4 edges: one from the middle vertex to each of the

other vertices. So - it has one central vertex connected to four other vertices, and no further edges. This is called a star graph. These two graphs are very different. Show that their adjacency matrices have the same eigenvalues with the same multiplicities.

Here are the adjacency matirces for teh graphs. Instead of solving for their charaacteristic polynomial by had ie. calculating $det(xI - A) = 0$, I put them into jordan normal form using the sympy package, and view the eigenvalues along the diagonal.

In [34]:
```
G1 = np.array([[0, 1, 0, 1, 0],
               [1, 0, 1, 0, 0],
               [0, 1, 0, 1, 0],
               [1, 0, 1, 0, 0],
               [0, 0, 0, 0, 0]])

G2 = np.array([[0, 0, 0, 0, 1],
               [0, 0, 0, 0, 1],
               [0, 0, 0, 0, 1],
               [0, 0, 0, 0, 1],
               [1, 1, 1, 1, 0]])

G1 = sy.Matrix(G1)
G2 = sy.Matrix(G2)
J1 = G1.jordan_form()
J2 = G2.jordan_form()
```

In [31]:
```
J1[1]
```

Out[31]:

$$\begin{bmatrix} -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

In [33]:
```
J2[1]
```

Out[33]:

$$\begin{bmatrix} -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Above are the jordan form for the adjacency matrixes of G^1 and G^2. as we can see the eigenvalue -2 shpows up once. The eigenvalue 2 shows up once. The eigenvalue 0 shows up three times in both matrices. So while the graphs are very different the eigenvalues are the same.

3 . Let $N \in M_4(R)$ be the matrix whose entries are all 1's. Let $A = N - I$.
(a) The matrix A is an adjacency matrix of a graph. Which graph is it?
(b) Show that $A^2 - 2A - 3I = 0$.
(c) Show that 3 is an eigenvalue of A of multiplicity 1.

(d) Are there any other eigenvalues? If so, what are their multiplicities?

(e) What is the determinant of A?

In [12]:
```
N = np.ones((4,4))
I = np.array([[1, 0, 0, 0],
              [0, 1, 0, 0],
              [0, 0, 1, 0],
              [0, 0, 0, 1],
             ])
I = sy.Matrix(I)
N = sy.Matrix(N)

A = N - I
A
```

Out[12]:

$$\begin{bmatrix} 0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 0 \end{bmatrix}$$

The graph is a square with all the cross vectors con

In [13]:
```
A**2 - 2*A - 3*I
```

Out[13]:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

In [14]:
```
A.jordan_form()
```

Out[14]:
```
[Matrix([
 [-1.0, -1.0, -1.0, 1.0],
 [ 1.0,    0,    0, 1.0],
 [   0,  1.0,    0, 1.0],
 [   0,    0,  1.0, 1.0]]),
 Matrix([
 [-1.0,    0,    0,   0],
 [   0, -1.0,    0,   0],
 [   0,    0, -1.0,   0],
 [   0,    0,    0, 3.0]])]
```

In [15]:
```
A.det()
```

Out[15]: -3.0

This can be acheaved by multiplying the eigenvalues.

4. Let $A \in M_3(R)$ be an orthogonal 3×3 matrix. (a) Show that $P_A(x)$ has a real root.

(b) Show that any real eigenvalues of A have to be either 1 or -1.

(c) Show that if det $A = 1$, then 1 must be an eigenvalue of A.

(d) Show that if det $A = 1$ then A is a rotation about some axis.

(e) Show that the product of two rotations about the origin in R_3 is again a rotation.

5. Explain why the adjacency matrix of a k-regular graph has $-k$ as an eigenvalue if and only if the graph has a bi-partite component.

6. Give examples of 4-regular graphs where the eigenvalues of the adjacency matrix have the following properties.

(a) 4 is an eigenvalue of multiplicity 1, and -4 is not an eigenvalue.

(b) 4 is an eigenvalue of multiplicity 3 and -4 is an eigenvalue of multiplicity 2.

This one hurt, but I've done it.

In [ ]:
```
G13 = np.array([[0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
                 [0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
                 [0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
                 [0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
                 [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
                 [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
                 [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
                 [1, 1, 1, 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, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
                 [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
                 [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
                 [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0
                 [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 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, 1, 1, 1, 1
                 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1
                 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1
                 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1
                 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0
                 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0
                 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0
                 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 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, 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, 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, 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, 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

G13 = sy.Matrix(G13)
G13.eigenvals()
```