In [2]:

```
#1
import numpy as np

# From the Figure like the Hint
#      4p1 - p2 - p3 =0
#      5p2 - p1 - p3 - p4 - p5 = 0
#      .
#      .
#      .

A = np.array([
    [4, -1, -1, 0, 0, 0, 0, 0, 0, 0],
    [-1, 5, -1, -1, -1, 0, 0, 0, 0, 0],
    [-1, -1 , 5, 0, -1, -1, 0, 0, 0, 0],
    [0, -1, 0, 5, -1, 0, -1, -1, 0, 0],
    [0, -1, -1, -1, 6, -1, 0, -1, -1, 0],
    [0, 0, -1, 0, -1, 5, 0, 0, -1, -1],
    [0, 0, 0, -1, 0, 0, 4, -1, 0, 0],
    [0, 0, 0, -1, -1, 0, -1, 5, -1, 0],
    [0, 0, 0, 0, -1, -1, 0, -1, 5, -1],
    [0, 0, 0, 0, 0, -1, 0, 0, -1, 4]
])

b = np.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1])
```

In [3]:

```
#Jacobi's Method

from numpy.linalg import inv, solve, norm

def jacobi(A, b, tolerance):
    xk_1 = np.zeros_like(b)
    D = np.diag(A)
    LplusU = A - np.diag(D)
    x_k = (b - (LplusU @ xk_1)) / D
    while (norm(x_k - xk_1, 2)/norm(x_k, 2)) > tolerance:
        xk_1 = x_k
        x_k = (b - (LplusU @ xk_1)) / D

    return x_k

print(jacobi(A, b, 1e-8))
```

```
[0.09019607 0.18039214 0.18039214 0.2980392  0.33333332 0.2980392
 0.45490195 0.52156861 0.52156861 0.45490195]
```

In [4]:

```
#Gauss-Siedel Method

from numpy.linalg import inv

def siedel(A, b, N):
    x = np.zeros_like(b)
    LD = np.tril(A)
    U = A - LD
    LDinv = inv(LD)
    for i in range(N):
        x = LDinv @ (b - U@x)
    return x



siedel(A, b, 9)
```

Out[4]:

```
array([0.08372205, 0.17174901, 0.17305007, 0.29069653, 0.3253404 ,
       0.29270444, 0.45090528, 0.51623368, 0.51703681, 0.45243531])
```

In [17]:

```
#2
#The Power Method

from numpy.linalg import eig

A1 = np.array([
    [0, 1, 2],
    [.5, 0, 0],
    [0, .25, 0]
])

l_1, v_1 = eig(A1)
print(l_1)
print(v_1)

print("------------------------------------------------------------------------------")
x = np.random.rand(3)
for i in range(50):
    x = (A1 @ x) / norm(x,2)

lmbda_1 = ((A1@x) / x)[0]
v_1 = x / lmbda
print(lmbda_1)
print(v_1)
```

```
[ 0.88464618+0.j         -0.44232309+0.2948714j -0.44232309-0.2948714j]
[[-0.86227396+0.j          0.69332593+0.j          0.69332593-0.j        ]
 [-0.48735527+0.j         -0.54259608-0.36171765j -0.54259608+0.36171765j]
 [-0.13772604+0.j          0.11796101+0.28307982j  0.11796101-0.28307982j]]
------------------------------------------------------------------------------
0.8846461771191109
[0.86227396 0.48735527 0.13772604]
```

In [19]:

```
#2
#The Power Method

from numpy.linalg import eig

A2 = np.array([
    [0, 6, 8],
    [.5, 0, 0],
    [0, .5, 0]
])

l_2, v_2 = eig(A2)
print(l_2)
print(v_2)

print("------------------------------------------------------------------------------")
x = np.random.rand(3)
for i in range(50):
    x = (A2 @ x) / norm(x,2)

lmbda_2 = ((A2@x) / x)[0]
v_2 = x / lmbda
print(lmbda_2)
print(v_2)
```

```
[ 2.         -0.99999998 -1.00000002]
[[-0.96836405  0.87287156 -0.87287156]
 [-0.24209101 -0.43643579  0.43643578]
 [-0.06052275  0.2182179  -0.21821788]]
------------------------------------------------------------------------------
2.0000000000000355
[2.18926861 0.54731715 0.13682929]
```

In [ ]:

In [ ]:

In [ ]: