In [ ]:
```
# directed graph
graph = [[ 0, 1,  0,  1, 1,  0],
       [1,  0,  1, 1,  0,  1],
       [ 0,  1,  0,  0, 1,  1],
       [ 1, 1,  0,  0,  1, 1],
       [1,  0, 1,  1,  0,  1],
       [ 0,  1,  1, 1,  1,  0]]
```

In [ ]:
```
print(graph)
```

```
[[0, 1, 0, 1, 1, 0], [1, 0, 1, 1, 0, 1], [0, 1, 0, 0, 1, 1], [1, 1, 0, 0, 1, 1], [1,
0, 1, 1, 0, 1], [0, 1, 1, 1, 1, 0]]
```

Degree Centrality

In [ ]:
```
class DegreeCentrality:
    @staticmethod
    def degreeCentralityIn(graph:list,node:int):
        return sum(graph[node-1])

    @staticmethod
    def degreeCentralityOut(graph:list,node:int):
        sum =0
        for row in graph:
            sum+=row[node-1]
        return sum

    @staticmethod
    def degreeCentrality(graph:list,node:int):
        return DegreeCentrality.degreeCentralityIn(graph,node)+DegreeCentrality.degree
```

In [ ]:
```
print(DegreeCentrality.degreeCentralityIn(graph,4))
print(DegreeCentrality.degreeCentralityOut(graph,4))
print(DegreeCentrality.degreeCentrality(graph,4))
```

```
4
4
8
```

EigenVector Centrality

In [ ]:
```
import numpy as np

def EigenVectorCentraility(graph:list, node:int):
    array = np.array(graph)
    eigenvalue , eigenvectors = np.linalg.eig(array)
    max_i = 0
    max_val = eigenvalue[0]

    for i in range(len(eigenvalue)):
        if max_val > eigenvalue[i]:
            max_val = eigenvalue[i]
            max_i = i

    return eigenvectors[node][max_i]
```

In [ ]:
```
EigenVectorCentraility(graph,4)
```

Out[ ]:
```
0.5465327677026017
```

In [ ]: