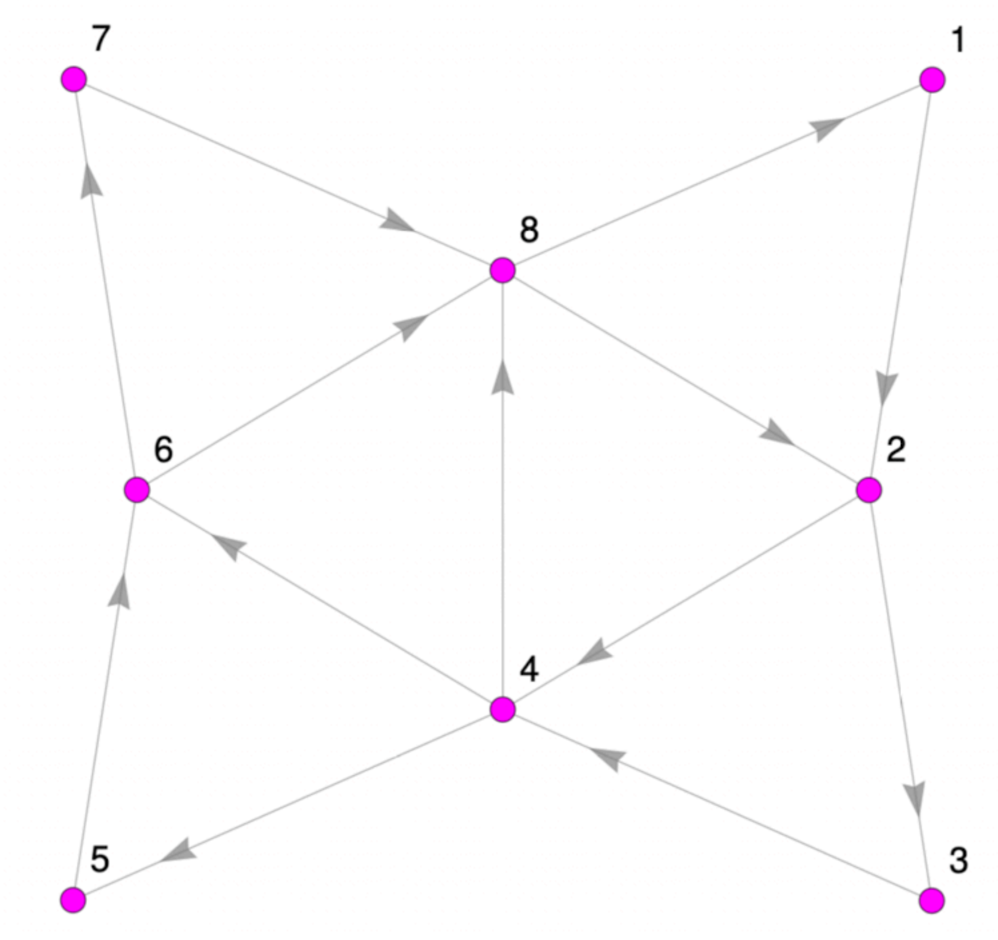
22_P3_Assessment_GionRubitschung

October 28, 2023

[1]:
```python
import numpy as np
from numpy.linalg import matrix_power
```



1 a. Define the adjacency matrix A of this directed graph

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

[2]:
```
A = np.array(
    [
        [0, 0, 0, 0, 0, 0, 0, 1],
        [1, 0, 0, 0, 0, 0, 0, 1],
        [0, 1, 0, 0, 0, 0, 0, 0],
        [0, 1, 1, 0, 0, 0, 0, 0],
        [0, 0, 0, 1, 0, 0, 0, 0],
        [0, 0, 0, 1, 1, 0, 0, 0],
        [0, 0, 0, 0, 0, 1, 0, 0],
        [0, 0, 0, 1, 0, 1, 1, 0],
    ]
)
A
```

[2]:
```
array([[0, 0, 0, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 0, 1],
       [0, 1, 0, 0, 0, 0, 0, 0],
       [0, 1, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 1, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0, 1, 1, 0]])
```

2 b. Compute A^3

[3]:
```
A_pow3 = matrix_power(A, 3)
A_pow3
```

[3]:
```
array([[0, 1, 1, 1, 1, 1, 0, 0],
       [0, 1, 1, 2, 1, 2, 1, 0],
       [0, 0, 0, 1, 0, 1, 1, 1],
       [1, 0, 0, 1, 0, 1, 1, 2],
       [1, 1, 0, 0, 0, 0, 0, 1],
       [1, 2, 1, 0, 0, 0, 0, 1],
       [0, 1, 1, 1, 0, 0, 0, 0],
       [1, 2, 1, 2, 1, 0, 0, 1]])
```

3 c. Compute $tr(A^4)$, the number of all closed walks of length 4

[4]:
```python
A_pow4 = matrix_power(A, 4)
trace_A_pow4 = np.trace(A_pow4)
print(f"A^4:\n{A_pow4}\nTrace of A^4: {trace_A_pow4}")
```

```
A^4:
[[1 2 1 2 1 0 0 1]
 [1 3 2 3 2 1 0 1]
 [0 1 1 2 1 2 1 0]
 [0 1 1 3 1 3 2 1]
 [1 0 0 1 0 1 1 2]
 [2 1 0 1 0 1 1 3]
 [1 2 1 0 0 0 0 1]
 [2 3 2 2 0 1 1 3]]
Trace of A^4: 12
```

$$A^4 = \begin{pmatrix} 1 & 2 & 1 & 2 & 1 & 0 & 0 & 1 \\ 1 & 3 & 2 & 3 & 2 & 1 & 0 & 1 \\ 0 & 1 & 1 & 2 & 1 & 2 & 1 & 0 \\ 0 & 1 & 1 & 3 & 1 & 3 & 2 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 0 & 1 & 1 & 3 \\ 1 & 2 & 1 & 0 & 0 & 0 & 0 & 1 \\ 2 & 3 & 2 & 2 & 0 & 1 & 1 & 3 \end{pmatrix}$$

$$tr(A^4) = 1 + 3 + 1 + 3 + 0 + 1 + 0 + 3 = 12$$

4 d. Compute the number of walks of length 10 from vertex 1 to vertex 1

[5]:
```python
A_pow10 = matrix_power(A, 10)
print(f"A^10:\n{A_pow10}\nNumber of walks from vertex 1 to vertex 1:␣
↪{A_pow10[0,0]}")
```

```
A^10:
[[29 49 30 49 17 30 20 43]
 [43 78 49 79 29 47 30 63]
 [20 43 29 49 20 29 17 30]
 [30 63 43 78 32 49 29 47]
 [17 30 20 43 17 32 20 29]
 [29 47 30 63 23 49 32 49]
 [20 29 17 30  8 23 17 32]
 [49 79 49 90 30 63 43 78]]
Number of walks from vertex 1 to vertex 1: 29
```

$A^{10}_{11} = 29$

5 e. Compute the number of walks of length 10 from vertex 1 to vertex 1 that never use the 4 → 8 edge

In order to achieve this, I am altering the entry of A_{84} from 1 to 0. This eliminates the $4 \to 8$ edge. If we then compute A^{10} we get all walks from vertex 1 to vertex 1 without the $4 \to 8$ edge.

[6]:
```python
A_modified = A
A_modified[7,3] = 0
A_modified_pow10 = matrix_power(A_modified, 10)
print(f"A_modified^10:\n{A_modified_pow10}\nNumber of walks from vertex 1 to vertex 1 without using the 4 to 8 edge: {A_modified_pow10[0,0]}")
```

```
A_modified^10:
[[ 7 21 15 20 10  6  2  8]
 [ 8 28 21 35 20 16  6 10]
 [ 2  8  7 21 15 20 10  6]
 [ 6 10  8 28 21 35 20 16]
 [10  6  2  8  7 21 15 20]
 [20 16  6 10  8 28 21 35]
 [15 20 10  6  2  8  7 21]
 [21 35 20 16  6 10  8 28]]
Number of walks from vertex 1 to vertex 1 without using the 4 to 8 edge: 7
```

$A^{10}_{11} = 7$ (without the $4 \to 8$ edge)

6 f. Compute the number of walks of length 10 from vertex 1 to vertex 1 that use the 4 → 8 edge at least once

We can build on the same principle as in e. We know that A to the power of 10 contains all walks of length 10. In case of vertex 1 to vertex 1 this is 29. What we don't know is the walks that use the $4 \to 8$ edge at least once. But from e. we know the number of walks of length 10 from vertex 1 to vertex 1 that never use the $4 \to 8$ edge. By this logic we can subtract the walks that never use the $4 \to 8$ edge and we get all walks that use the $4 \to 8$ edge at least once.

[7]:
```python
A_with_4_to_8_pow10 = A_pow10 - A_modified_pow10
print(f"A_with_4_to_8_pow10^10:\n{A_with_4_to_8_pow10}\nNumber of walks from vertex 1 to vertex 1 using the 4 to 8 edge at least once: {A_with_4_to_8_pow10[0,0]}")
```

```
A_with_4_to_8_pow10^10:
[[22 28 15 29  7 24 18 35]
 [35 50 28 44  9 31 24 53]
 [18 35 22 28  5  9  7 24]
 [24 53 35 50 11 14  9 31]
 [ 7 24 18 35 10 11  5  9]
 [ 9 31 24 53 15 21 11 14]
 [ 5  9  7 24  6 15 10 11]
 [28 44 29 74 24 53 35 50]]
```

```
Number of walks from vertex 1 to vertex 1 using the 4 to 8 edge at least once:
22
```

$A^{10}_{11} = 22$ (with using the $4 \to 8$ edge at least once)