In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
# add here the libraries you need
```

Discrete Time Markov Chains

This is an exercise notebook on DTMCs.

Remember to revise of the lecture on DTMC before attempting to solve it!

Models

Consider few DTMCs, those seen in the lectures and few more:

1. A student goes to a concert and alternates between being "dancing" and being "at the bar" until, at a certain point, it goes home which is an absorbing state. Model this scenario as a DTMC, draw the diagram and define the respective transition matrix, called `transition_concert` .

Give names to states and use a dictionary to match names with indexes in the matrix.

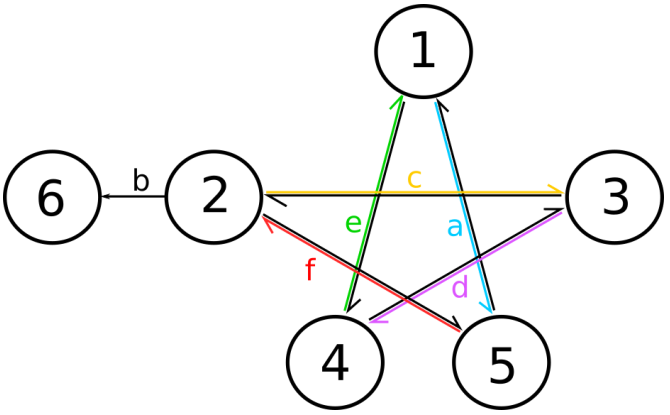
Always check that the defined transition matrix is **well-defined**.

In [ ]:

```

```

2. Consider a walker moving randomly along the star graph shown below. The graph is composed of six states. State 6 is the terminal point.

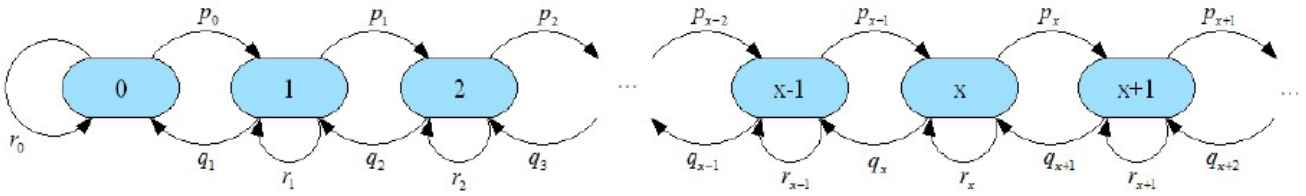


Model the movement of the walker as a DTMC. Choose the parameters $a, b, c, d, e, f, e \in (0, 1)$ so that the transition matrix, `transition_star` , is well-defined. Keep in mind that, ingoing and outgoing transitions can happen with different probabilities.

In [ ]:

```
```

3. A general birth-death chain, `transition_birth_death()` . Define it writing a function that takes `N` as input (the maximum population size) and vectors `p` (the birth probability) and `q` (the death probability) of length N and returns the transition matrix `transition_birth_death` .



In [ ]:

```
```

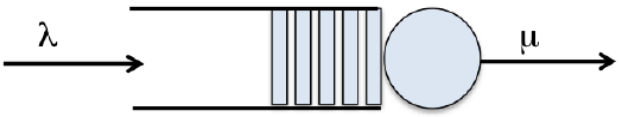
In [ ]:

```
```

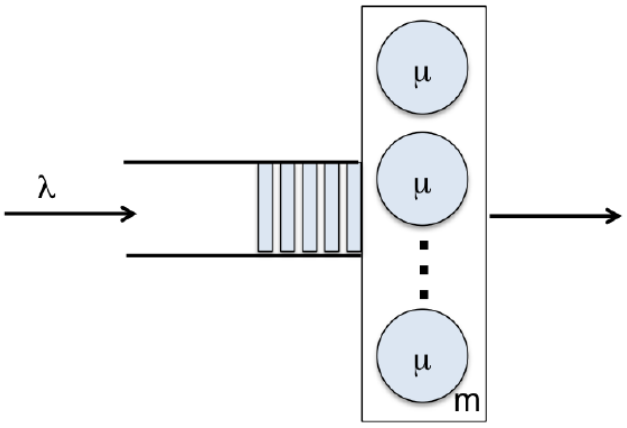
4. **Queue model**. Queuing chains model the number of customers in the system as a function of time, and in particular, whether the server(s) can adequately handle the flow of customers. Let X_n be the number of custumers in the system at time $n \in \mathbb{N}$. Define a DTMC that models a queueing system where the costumer propensity to join the system, i.e. the arrival rate, is inversely proportional to the current length of the queue.

Draw the transitions graph, the transition matrix and explain the details of your model in the two following scenarios:

4.1. **Single server**: the service is provided by a single server with service rate μ



4.2. m **servers**: the service is provided by m servers each with a service rate μ



In [ ]:

Transient probability

Write a function that takes a DTMC as input (both the transition matrix `transition_model` and the initial probability `prob_init_model`) and the number of steps `n`, and returns the probability `prob_model` of being in each state after `n` steps. Plot the output as an histogram. *Extra*: compute the transient probabilities from time 0 to time `n` and visualize this output as a 2d colormap, with one axis representing states, and the other time.

In [ ]:

Reachability (absorption probability) and hitting times.

Write a function that takes as input:

- a DTMC (the transition matrix `transition_model` is enough),
- a subset of states `target_region` (either specified by their name or by their indices). The function should compute a vector `prob_absorption_model`, containing the **absorption probability** for each state. Use the iterative method discussed in the lecture.

Implement also a method with the same input, returning the **expected hitting times** `exp_hitting_time_model`.

In [ ]:

Steady state probability

Write a function taking as input a DTMC and computes its steady state probability, assuming the chain is irreducible, aperiodic, and finite. *Extra*: explore Python libraries and find one implementing graph algorithms. Use a method to compute strongly connected components to check if a chain is irreducible.

If a chain is not irreducible, feel free to make it reducible adding transitions where needed.

In [ ]:

```
```

Model class

Implement a class having all the methods to initialize a DTMC, to compute transient behaviour, steady state behaviour, and reachability. Consider also defining a map linking state numbers to reference meaningful names.

In [ ]:

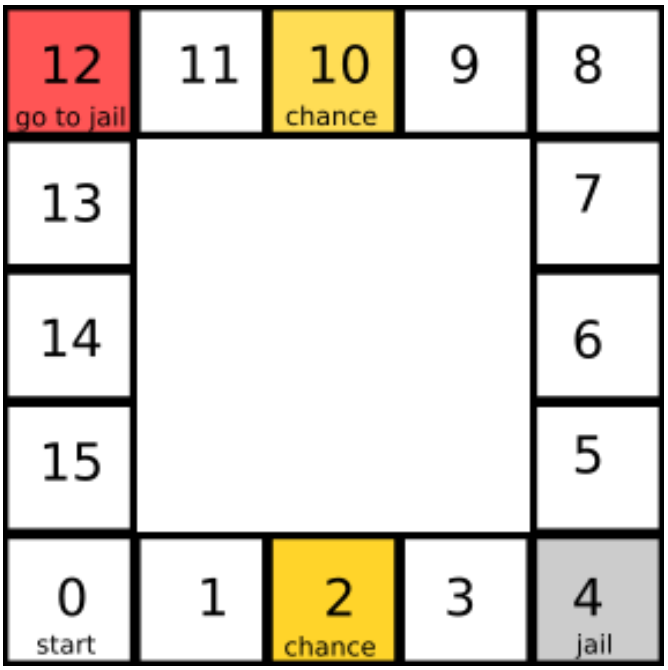
```
```

Summary exercise

Small Monopoly:

The game Small Monopoly is played on a game board that has 16 spaces arranged around the outside of a square. The squares have names like Reading Railroad and Park Place but we will number the squares 0 (Go), 1 (Baltic Avenue), ..., 15 (Boardwalk). You roll two dice and move forward a number of spaces equal to the sum. The game ends when you reach or pass Boardwalk.

- Square 12 is “Go to Jail,” which sends you to square 4. If you roll a double, you get out.
- There are two Chance squares at 2 and 10 (diamonds on the graph) where you draw a card, which can send you to another square. The card is picked (with replacement) from a deck containing all the 16 squares.
- Described this game as a directed graph, i.e. define its transition matrix.
- What is the probability of ending the game?
- How long does a game take on average?


12
go to jail
11
10
chance
9
8
13
7
14
6
15
5
0
start
1
2
chance
3
4
jail

In [ ]: