Stanford CME 241 (Winter 2021) - Assignment 2

Frog Problem

Chih-Hsuan 'Carolyn' Kao (chkao831@stanford.edu)

Feb 3rd, 2021

In [1]:

```
from IPython.display import Image
import numpy as np
```

A frog is on one side of the pond and wants to get on the other side. There are n lily leaves ahead in a line -- the n-th leave lies on the other end of the pond and is the destination.

Rule:

Whatever the position the frog is at any time, it will only go ahead and the probability to land on one of the leaves left in front of it (including the destination) is uniform.

Question:

What is the expected value for the number of jumps it will take the frog to arrive to the destination leaf?

My Analyses:

Theoretically speaking, we could set up the problem as a Markov chain: Having states enumerating from 1 to n. $s = 0$ implies the start (at river bank) and $s = n$ at terminal state. The process is a strictly monotone discrete Markov chain with the $(n+1) * (n+1)$ transition probability matrix as below,

In [2]:

```
Image(filename = "frog_transition_matrix.png", width = 400, height = 400)
```

Out[2]:

$$\mathbf{P} = \begin{bmatrix} 0 & 1/n & 1/n & \cdots & 1/n & 1/n & 1/n \\ 0 & 0 & 1/(n-1) & \cdots & 1/(n-1) & 1/(n-1) & 1/(n-1) \\ 0 & 0 & 0 & \cdots & 1/(n-2) & 1/(n-2) & 1/(n-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

From above, we could see that this problem can be framed as a problem of finding the distribution of the "hitting time" for the absorbing state of a discrete Markov chain, from a perspective of the Markov Process. On the other hand, if we aim at just calculating the expected value of jumps, we could take advantage of the uniform distribution and use the formula of the expected value to obtain it.

In [3]:

```
def frog_problem(n: int) -> float:
    """Solver of the Frog Problem.
    """

    summation = dict()
    for i in reversed(range(n)):
        summation[i] = (1 + sum(1 + summation[j] for j in range(i + 1, n))) / (n - i)
    return summation[0]
```

In [4]:

```
frog_problem(10)
```

Out[4]:

```
2.9289682539682538
```

From this example, it is shown that for a frog to get to another side of pond with 10 lily leaves in line, it takes around 2.93 jumps to arrive to the destination.

In [ ]:

```

```