Transition matrix of Markov Chain for n=4:

	0000	0001	0010	0100	1000	1100	1010	1001	0110	0101	0011	1110	1101	1011	0111	1111
0000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0001	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0010	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0100	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1000	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1100	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1010	0	0	0	0	0	1/6	2/6	1/6	1/6	0	1/6	0	0	0	0	0
1001	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0110	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0101	0	0	0	0	0	1/6	0	1/6	1/6	2/6	1/6	0	0	0	0	0
0011	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1110	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1101	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1011	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

States are labelled with a bit string with each bit representing an agent in the cycle; agents adjacent in the string are adjacent in the cycle, and the first and last agents are also adjacent. Each bit is labelled either 0 or 1 depending on the type of agent in the corresponding position in the cycle.

For each row, i, in the matrix, any elements in columns corresponding to states that cannot be reached from the state denoted by i are set to 0. Each distinct pair of elements has a $\frac{1}{\binom{4}{2}}$ chance of being selected for possible swapping as there are $\binom{4}{2} = 6$ total pairs that can be selected. For each pair, there is a column with a corresponding state, j, such that $i \to j$ if the pair swaps which is guaranteed to happen if and only if at least one agent is happier, and neither is unhappier. Thus, $(i,j) = \frac{1}{6} \times 1\ (if\ swap\ is\ agreed\ to)\ OR\ \frac{1}{6} \times 0\ (if\ swap\ is\ not\ agreed\ to)$.

Finally, $(i,i) = 1 - (sum\ of\ every\ other\ entry\ in\ the\ row)$

Canonical Form for n=4:

	1010	0101	0000	0001	0010	0100	1000	1100	1001	0110	0011	1110	1101	1011	0111	1111
1010	2/6	0	0	0	0	0	0	1/6	1/6	1/6	1/6	0	0	0	0	0
0101	0	2/6	0	0	0	0	0	1/6	1/6	1/6	1/6	0	0	0	0	0
0000	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0001	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0010	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0100	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1000	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1100	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1001	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0110	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0011	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1110	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1101	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1011	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1010 and 0101 are the only two transient states so the rows and columns pertaining to them have been reordered to be at the top and left ends of the matrix respectively.

Montecarlo Simulations

The following is a sample output from the script in q1a_script.py:

```
For n=4, the average absorbing time is 1.542
For n=5, the average absorbing time is 2.572
For n=6, the average absorbing time is 5.047
For n=7, the average absorbing time is 8.138
For n=8, the average absorbing time is 12.123
For n=9, the average absorbing time is 14.826
For n=10, the average absorbing time is 20.623
```

The average absorbing time was found using 1000 randomly sampled simulations for each value of n. As shown, absorbing time increased significantly as n went up.

Numerical Approximations

From q1a_script.py:

```
Numerically, absorption time for n=4 is: [ 1.5  1.5]
Numerically, absorption time for n=5 is: [ 2.5  2.5  2.5  2.5  2.5]
```

The values here are quite close to the values found by averaging the simulations for n=4 (1.542) and n=5 (2.572).