MoveMaster

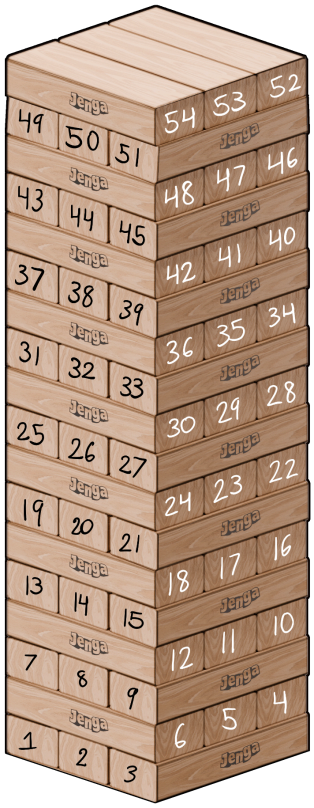
Modeling The Game Mathematically

In a classic game of Jenga, there are 54 wooden blocks. Assuming we can have less than 54 blocks in play at any point in the game (i.e., assuming we break the official rules of Jenga), this means that there are 2^{54} = 18,014,398,509,000,000 (roughly 18 quadrillion) unique configurations of jenga blocks achievable by simply removing pieces from the original tower. In order to differentiate these game states from one another, let's represent a game configuration as a 54-bit long sequence of binary:

111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111

In our model, the left-most bits represent the blocks at the very bottom of the tower, and the right-most bits represent the blocks at the very top of the tower. Additionally, "**1**" bits indicate the presence of a block in that position, while "**0**" bits indicate the absence of a block. Furthermore, every **tribit** represents a layer in the Jenga tower, with the bits progressing from furthest from the camera → closest to the camera when being read from left → right.

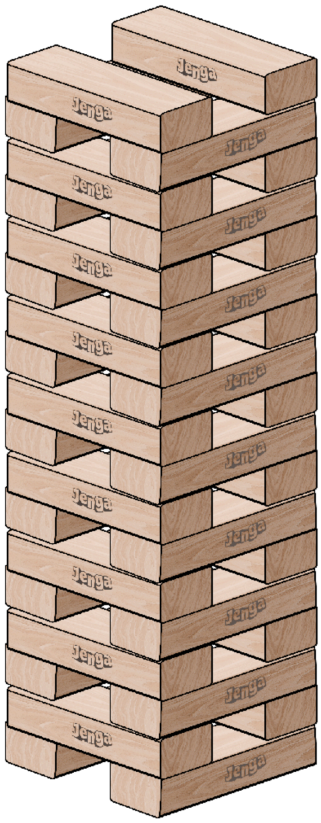
Since this manner of representing the configuration of a Jenga game is not particularly intuitive, we provide a labeled diagram to the right:



Let's consider a different sequence:

101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101 101

Visualizing this sequence, we have the following game configuration:

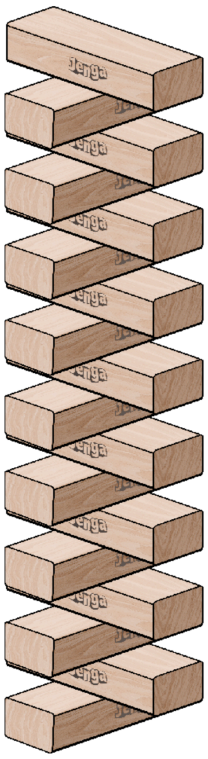


This would be considered a “**success state**” because this tower is structurally sound and would not collapse.

Let’s consider another sequence:

010 010 010 010 010 010 010 010 010 010 010 010 010 010 010 010 010 010 010

If we were to visualize the above sequence:

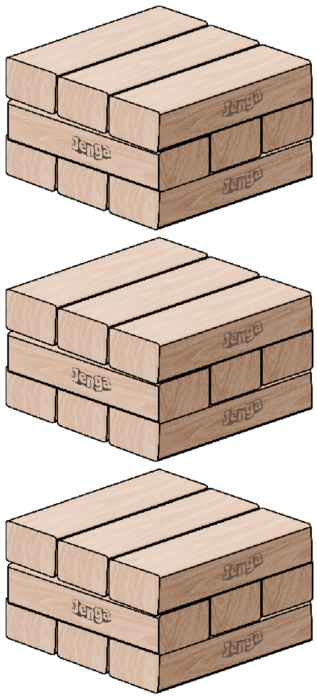


Similar to the previous configuration, this would also be considered a “success state” because this tower is structurally sound and would not collapse.

On the other hand, if we were to visualize a sequence like:

111 111 111 000 000 000 111 111 111 000 000 000 111 111 111 000 000 000

This would result in the following game configuration:

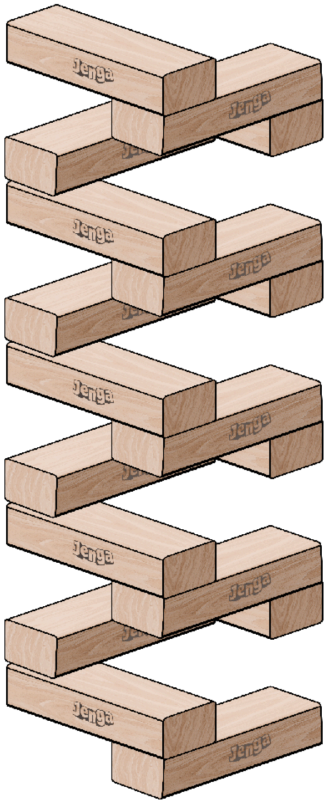


This would immediately be ruled out as a “**fail state**” because the tower is (clearly) unstable and would collapse.

Similarly, if we were to visualize a sequence like:

001 001 100 100 001 001 100 100 001 001 100 100 001 001 100 100 001 001

This would result in the following game configuration:

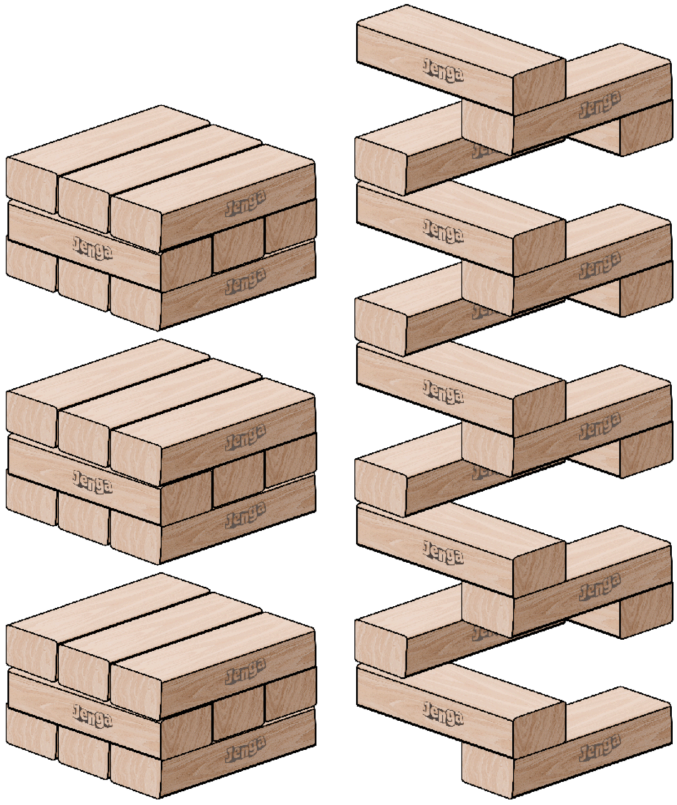


Again, this would be considered a “fail state” because the tower is highly unstable and would collapse.

Although we could simply use the physics engine to determine which states are success and fail states (i.e., whether or not the tower falls over), if we assume that running a single simulation takes 0.1 seconds, this means that simulating all 2^{54} possible configurations would take approximately 57,123,283 years (this is unreasonably long!). Thus, we must rely on analysis techniques to eliminate as many fail states as we can.

Eliminating “Fail States” Through Analysis

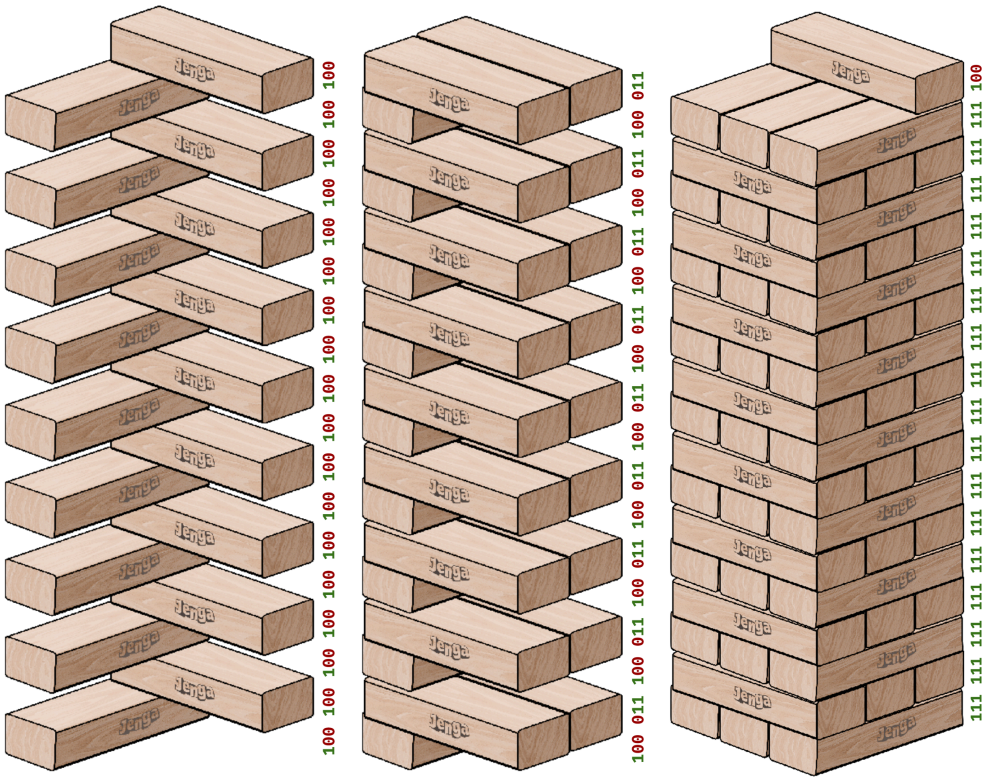
As demonstrated earlier, not all 2^{54} of these configurations are physically possible, as some of these combinations would result in floating pieces, or pieces whose center of mass is not supported, immediately constituting a fail state. In order to determine what fraction of these configurations result in fail states, let us reconsider 2 of the above examples:



On the **left** is `111 111 111 000 000 000 111 111 111 000 000 000 111 111 111 000 000 000`
On the **right** is `001 001 100 100 001 001 100 100 001 001 100 100 001 001 100 100 001 001`

A similarity that exists between these two fail configurations is the presence of **two or more consecutive zeros within a single tribit**. Let's visualize some more sequences to determine if this is a consistent pattern:

`111 111 111 111 111 111 111 111 111 111 111 111  111 111 100`


100 100 100 100 100 100 100 100 100 100
100 011 100 011 100 011 100 011 100 011 100 011
111 111 111 111 111 111 111 111 111 111 100