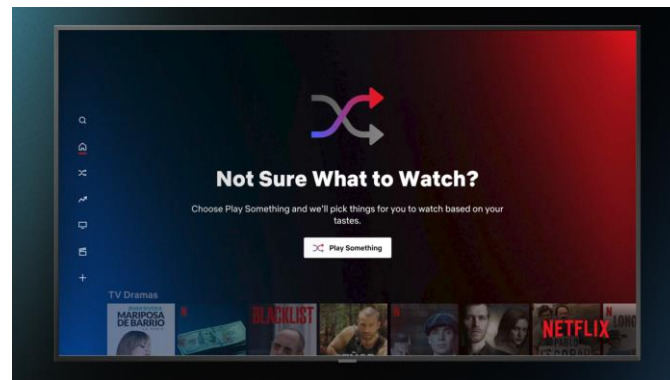
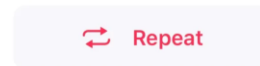
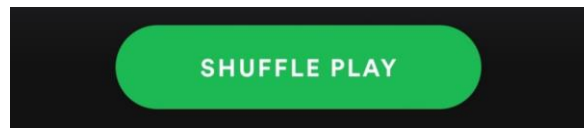


Is Random Really Random?

Giovi Moriarty

Overview

- Monte Carlo Fallacy
- PRNGs vs TRNGs
- Common algorithms used to generate randomness
- Spotify's approach
- Example



How Humans Perceive Randomness

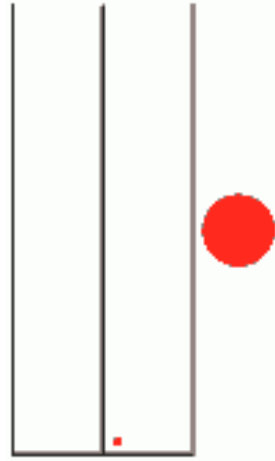
- When we shuffle a playlist and see the same artist/album multiple times in a row, we tend to think this pattern is not truly random
 - Each song being played is an independent event
 - An entire album could play in order and still represent a random sequence of events
- Users started to criticize streaming platforms' algorithms for not being truly random



Possible outcomes of a truly random shuffle that don't actually appear random

Monte Carlo Fallacy (AKA Gambler's Fallacy)

- Belief that if something happens more frequently than normal in the past then it is less likely to happen again in the future (or vice versa)
 - Coin flip lands on heads 3 times in a row so next flip must come up heads
 - Actually, tails is not more likely than heads
- Only for independent events



The Problem

- The probability of each song playing is equal
 - Independent events
- If you have 10 tracks of genre A, 11 of genre B, and 11 of genre C, a truly random shuffle algorithm could generate a pattern such as:
 - AACBBCBACABBCCACCCCABBACBACABABB
 - Does this look random?

The Problem

- The probability of each song playing is equal
 - Independent events
- If you have 10 tracks of genre A, 11 of genre B, and 11 of genre C, a truly random shuffle algorithm could generate a pattern such as:
 - AACBBCBACABBCCA**CCCC**ABBACBACABABB
 - Does this look random?

The Problem

- The probability of each song playing is equal
 - Independent events
- If you have 10 tracks of genre A, 11 of genre B, and 11 of genre C, a truly random shuffle algorithm could generate a pattern such as:
 - AACBBCBACABBCCACCCCABBACBACABABB
 - Does this look random?

The Problem

- The probability of each song playing is equal
 - Independent events
- If you have 10 tracks of genre A, 11 of genre B, and 11 of genre C, a truly random shuffle algorithm could generate a pattern such as:
 - AACBBCBACABBCCACCCCABBACBACABABB
 - Does this look random?
- Other equally likely outcomes:
 - AAAAAAAAAABBBBBBBBBBCCCCCCCCCCC
 - BCABCABCABCABCABCABCABCABCABC

PRNGs and TRNGs

Pseudorandom number generators vs True random number generators

PRNGs vs TRNGs

PRNG (pseudorandom number generator)

- Computer algorithm used to generate random number
 - Starts with key/seed and then calculates number
- Useful for simulation and modeling
- Efficient, deterministic, and periodic
 - Able to be easily reproduced

TRNG (true random number)

- Based on random events in nature
 - Climate changes, the cosmic microwave background
- Very expensive to get data
- Slow
- Good for data encryption and gambling

Early PRNGs

- Middle square method and Lehmer generator used in the 1940s and 50s

Middle Square Method

- Invented by John von Neumann in 1949
- Generally not a good method due to short period
- Seed is n-digit number. Square seed and take n digits in the middle of the squared number to use as produced random number and next seed
- All numbers produced have same number of digits

Lehmer Generator

- Published in 1951
- Predecessor to Linear Congruential Generator
- $X_n = (a * X_{n-1}) \bmod m$
 - m is a prime number

Linear Congruential Generator

- One of the oldest and best-known PRNG algorithms

$$X_n = (a * X_{n-1} + b) \text{ mod } m$$

- Requirements:
 - $m > 0$, usually prime
 - $0 < a < m$
 - $0 \leq b < m$
 - $0 \leq X_0 < m$
- Fast and requires little memory
- Easy for computers to use

Other Currently Used PRNGs

Xoroshiro128+

- XOR, rotate, shift, rotate
- Uses a shift/rotate-based linear transformation
- 64-bit
- Period of $2^{128}-1$
- Can jump in increments of 2^{64}

Squares RNG

- New implementation of Middle Square method
- Several rounds of squaring are applied to a counter to produce an output

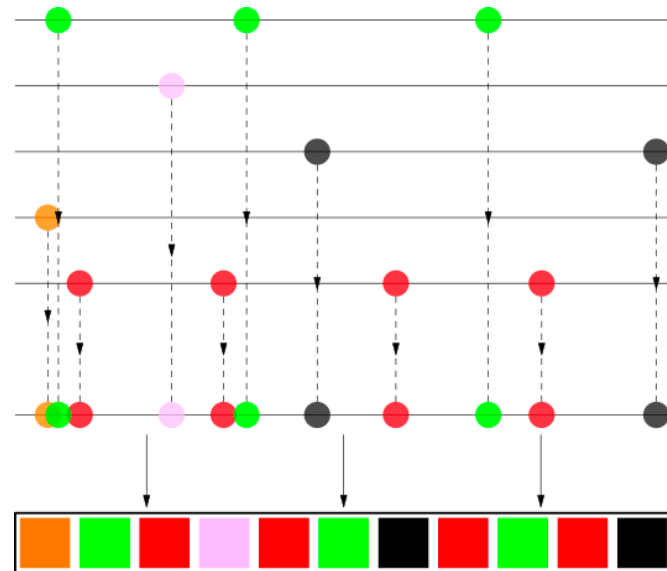
Applications of PRNGs

Fisher-Yates Shuffle

- First described in 1938 by Ronald Fisher and Frank Yates was done manually, but has been adapted for computer use
- Original algorithm used by Spotify to shuffle songs
- Produces an unbiased permutation
- Essentially shuffles an array
- Users claimed shuffle wasn't random enough

Less Random Approach to Randomizing

- Spotify developed new algorithm that places songs in specific order rather than completely randomizing
- Anti-clustering algorithm similar to dithering
 - Try to stretch artists out as evenly as possible
- Uses Fisher-Yates shuffle within artists/albums

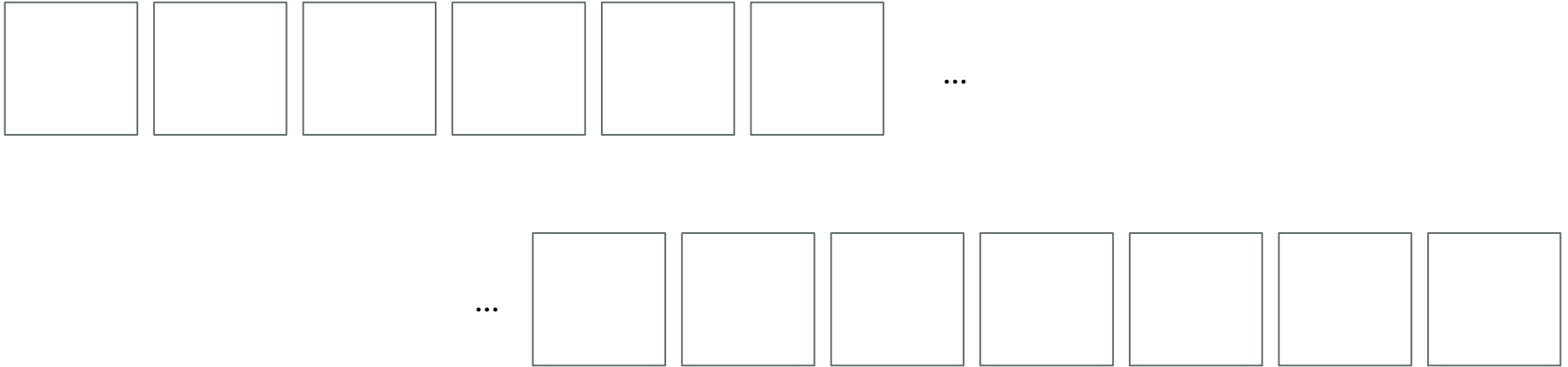


Example



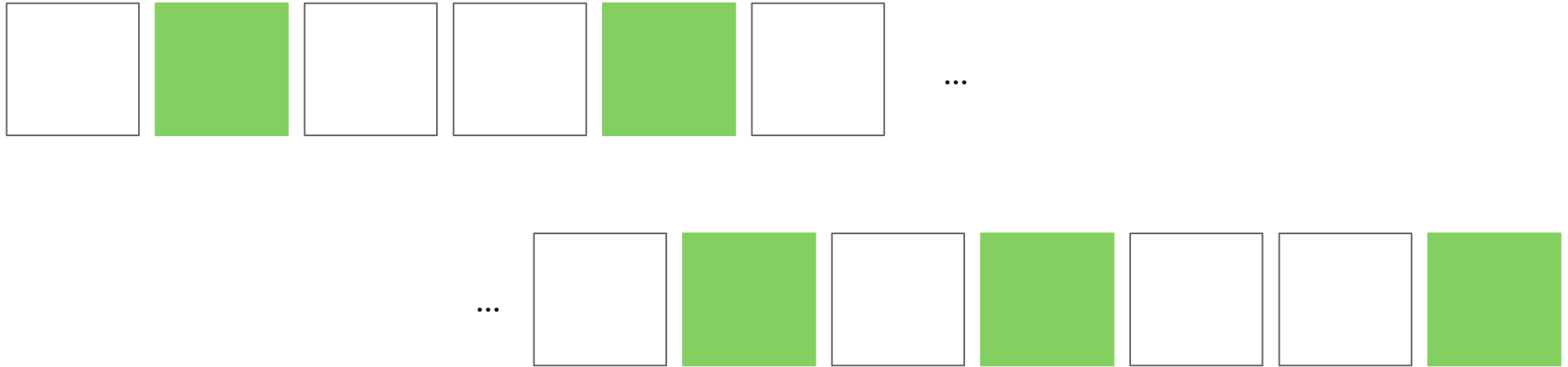
Total: 13
Pop: 5
Rock: 4
Country: 3
Jazz: 1

Sequence Length: 13



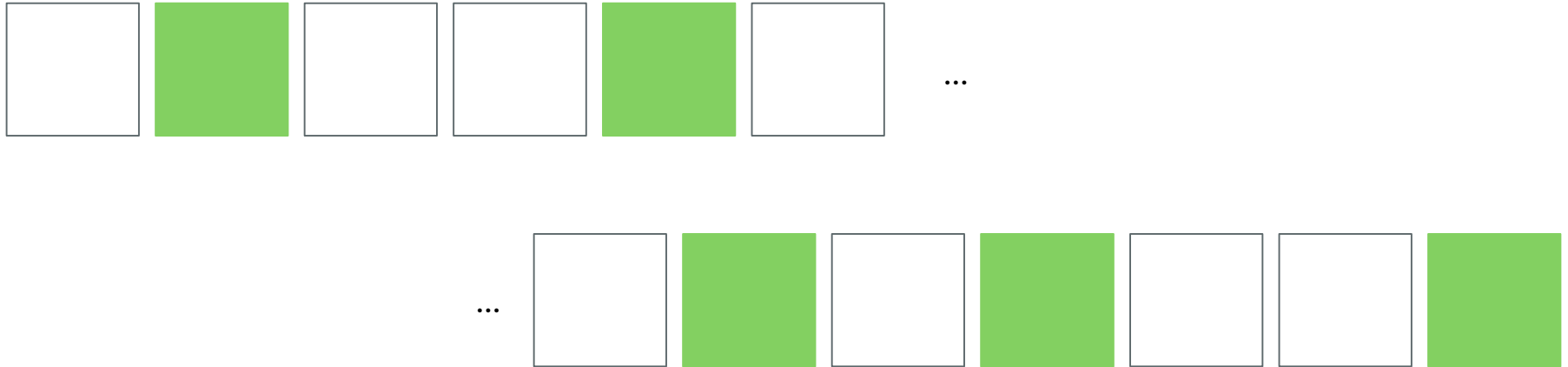
5 pop songs: should appear every ~20% of sequence
20% of 13 \rightarrow 2.6

First, Place Pop Songs



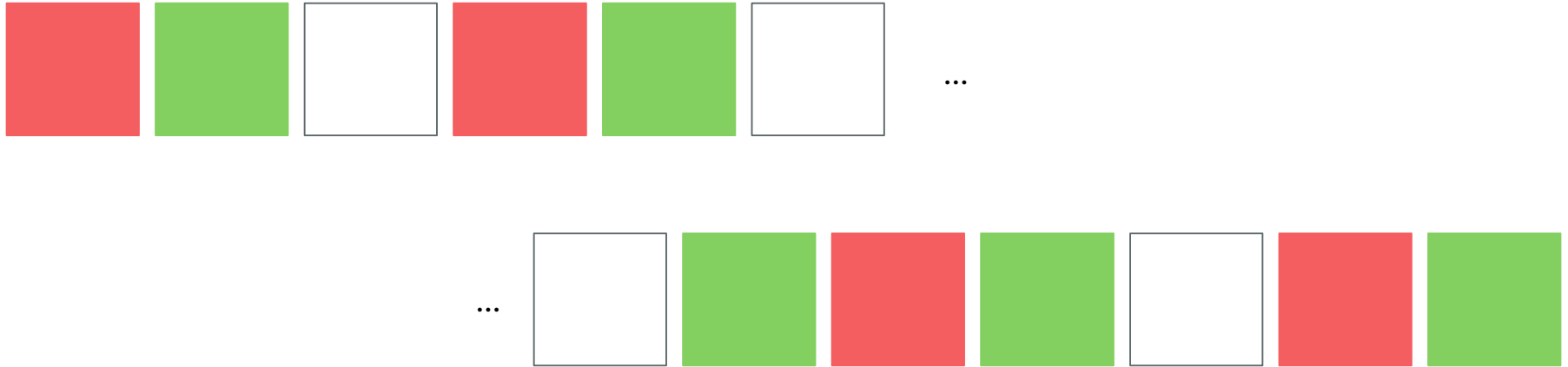
5 pop songs: should appear every ~20% of sequence
20% of 13 \rightarrow 2.6

First, Place Pop Songs



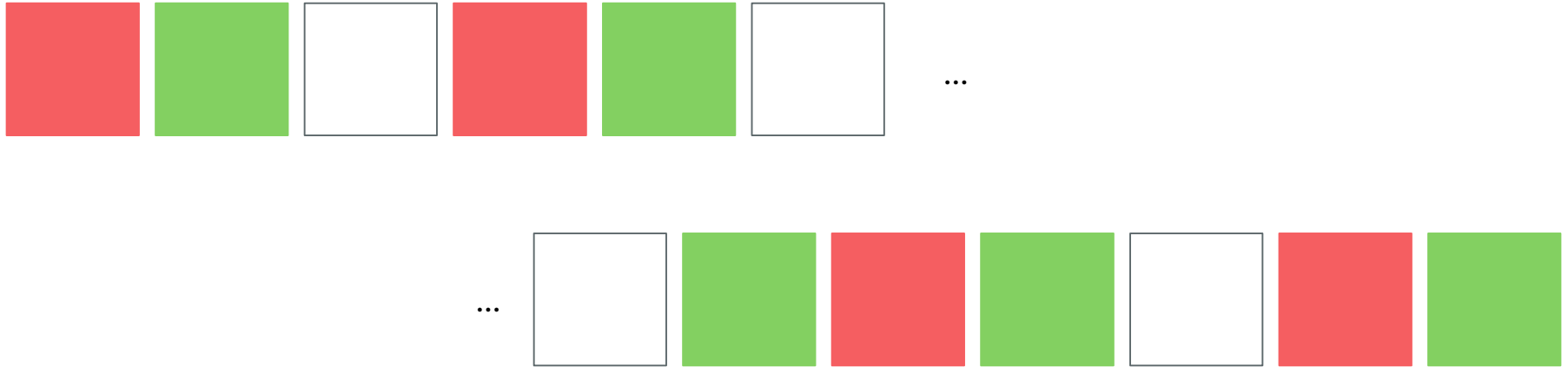
4 rock songs: should appear every ~25% of sequence
25% of 13 \rightarrow 3.25

Next, Order Rock Songs



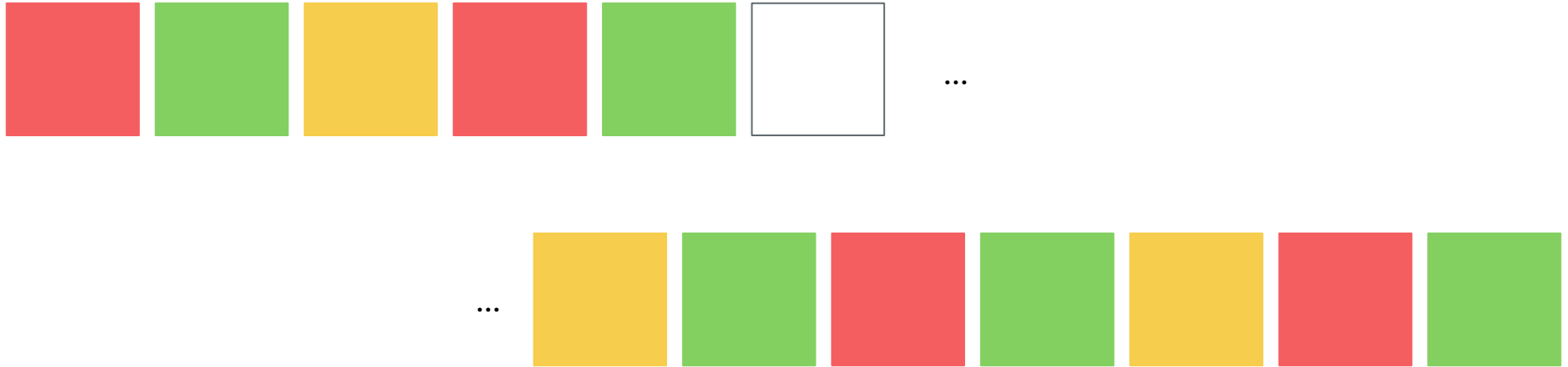
4 rock songs: should appear every ~25% of sequence
25% of 13 \rightarrow 3.25

Next, Order Rock Songs



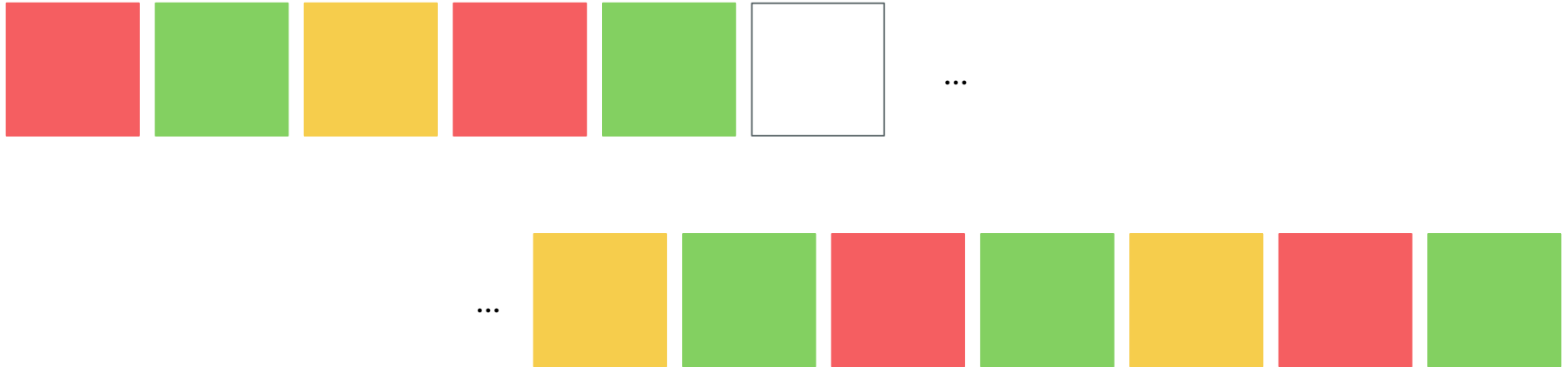
3 country songs: should appear every ~33% of sequence
33% of 13 \rightarrow 4.29

Add in Country Songs



3 country songs: should appear every ~33% of sequence
33% of 13 → 4.29

Add in Country Songs



1 jazz song: can go in last remaining spot

Place Remaining Jazz Song



1 jazz song: can go in last remaining spot

Within Each Genre, Use Fisher-Yates Shuffle to Order Specific Songs

Say our pop songs are:

good 4 u (Olivia Rodrigo) = 0

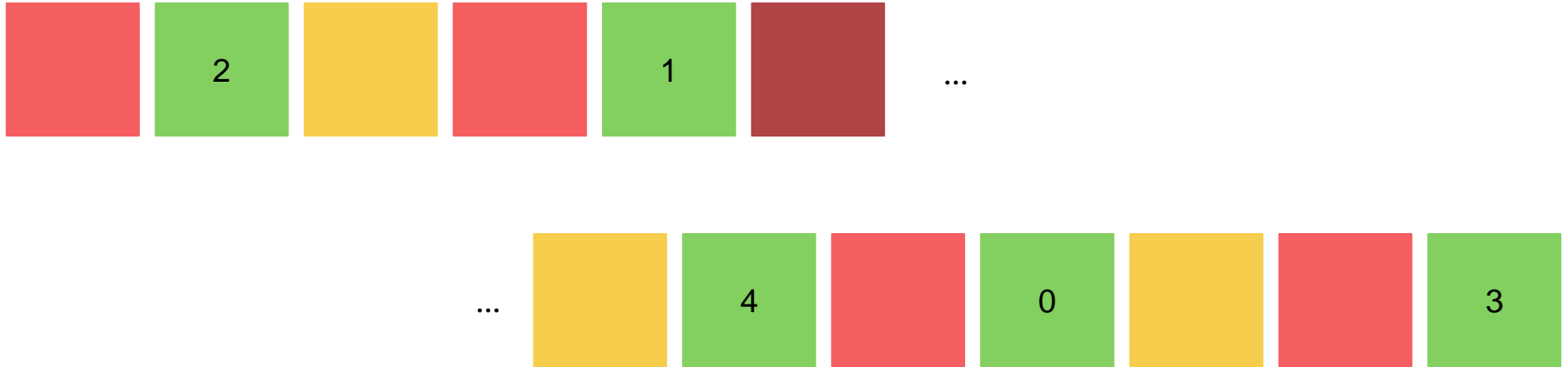
Happier Than Ever (Billie Eilish) = 1

STAY (The Kid LAROI) = 2

Shivers (Ed Sheeran) = 3

and Save Your Tears (The Weeknd) = 4

Use Fisher-Yates to Shuffle Within Subgroups (Example):



And there you have your non-
random shuffled playlist!

Bibliography

<https://uxdesign.cc/randomly-not-random-2fd53536513c>

<https://towardsdatascience.com/building-a-pseudorandom-number-generator-9bc37d3a87d5>

<https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/>

<https://www.freecodecamp.org/news/random-number-generator/>

<https://engineering.atspotify.com/2014/02/how-to-shuffle-songs/>

<http://keyj.emphy.de/balanced-shuffle/>

https://rosettacode.org/wiki/Linear_congruential_generator

https://en.wikipedia.org/wiki/Gambler%27s_fallacy

https://en.wikipedia.org/wiki/Middle-square_method

https://en.wikipedia.org/wiki/Lehmer_random_number_generator

https://bashtage.github.io/randomgen/bit_generators/xoroshiro128.html

<https://cs.paperswithcode.com/paper/squares-a-fast-counter-based-rng>