Darts, Dice, and Coins

Sampling from a Discrete Distribution

The Problem Statement

- You are given an n-sided loaded die with probabilities p_1, p_2, ..., p_n of sides 1, 2, ..., n coming up.
- What is the most efficient algorithm for simulating rolls of the die?

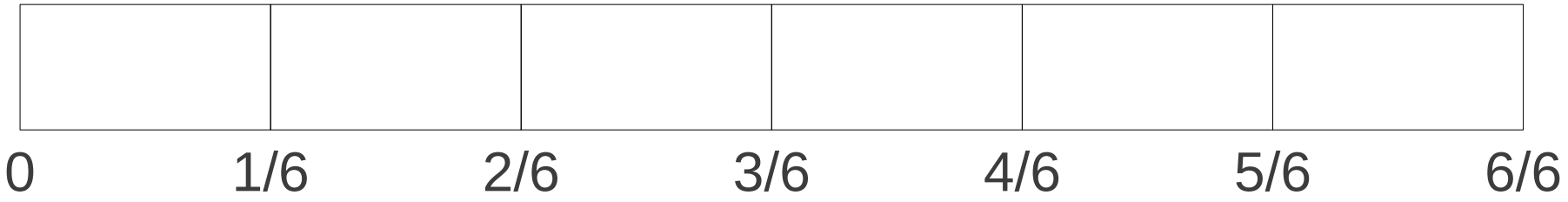
The Computational Model

- Assume that the following can be done in O(1) time:
  - Generation of a uniformly-random real number in the interval [0, 1)
  - Addition, subtraction, multiplication, division, and comparisons of real numbers.
  - Floor and ceiling of real numbers.

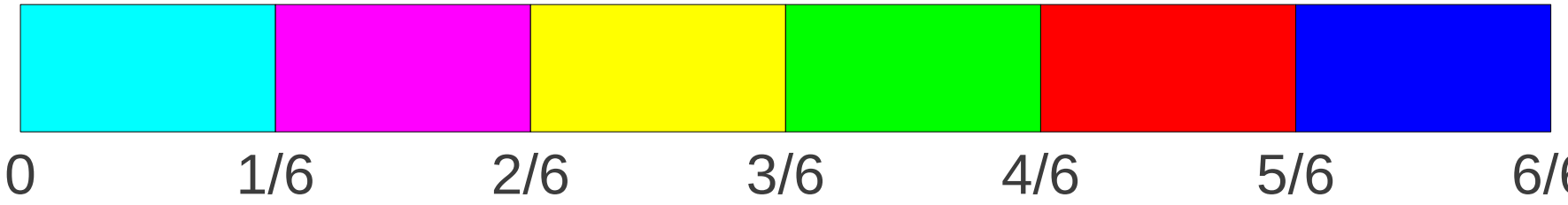
Simulating a Fair Die

Simulating a Fair Six-Sided Die

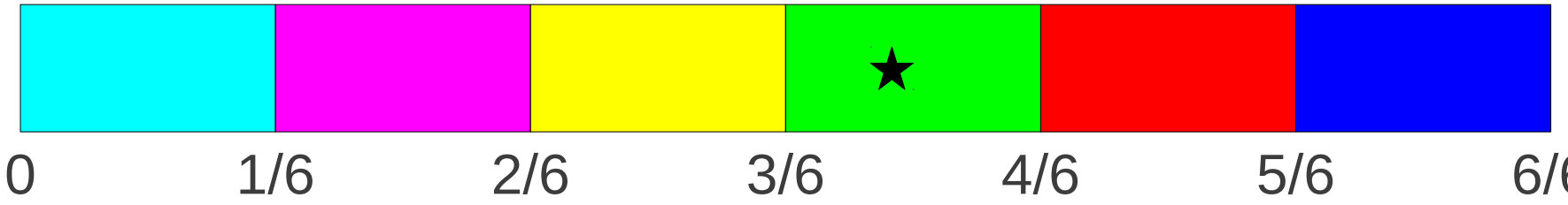
Simulating a Fair Six-Sided Die



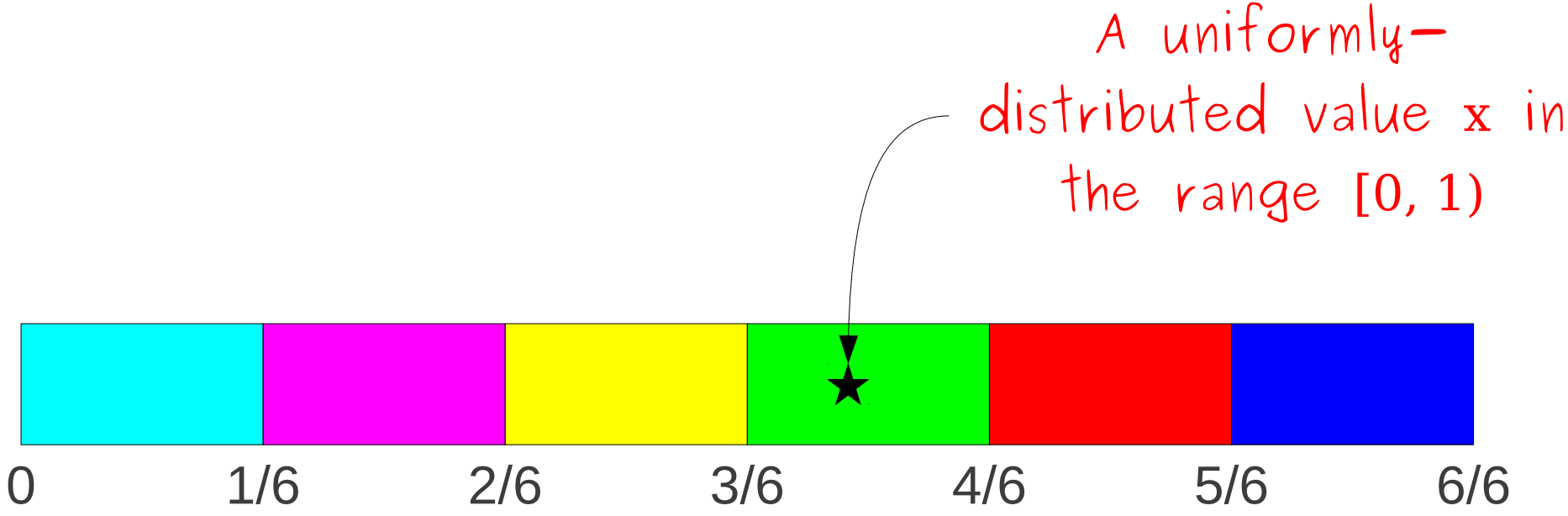
Simulating a Fair Six-Sided Die



Simulating a Fair Six-Sided Die

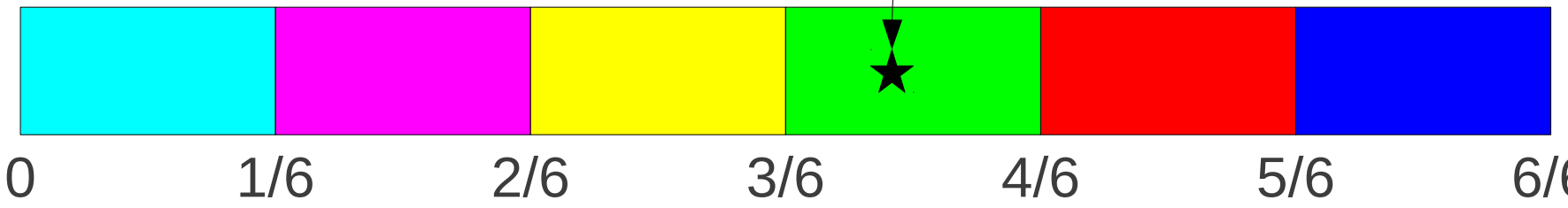


Simulating a Fair Six-Sided Die



Simulating a Fair Six-Sided Die

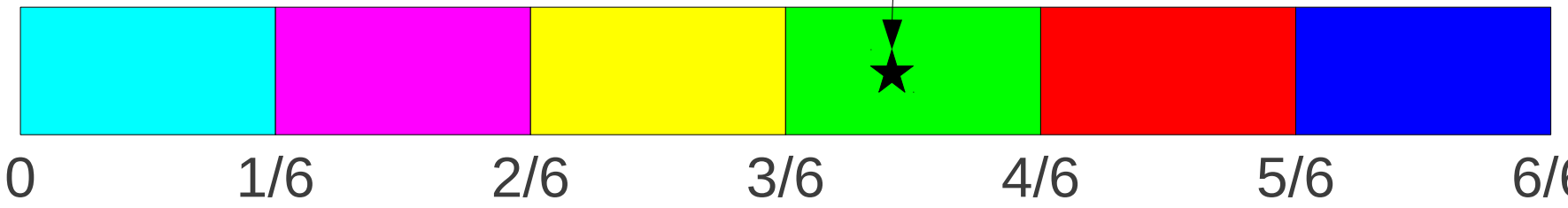
A uniformly-distributed value x in the range $[0, 1)$



What is the largest integer k such that $k/6 < x$?

Simulating a Fair Six-Sided Die

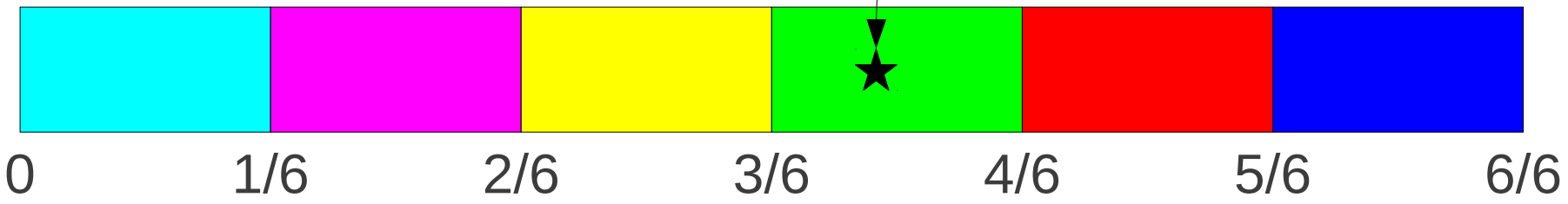
A uniformly-distributed value x in the range $[0, 1)$



What is the largest integer k such that $k < 6x$?

Simulating a Fair Six-Sided Die

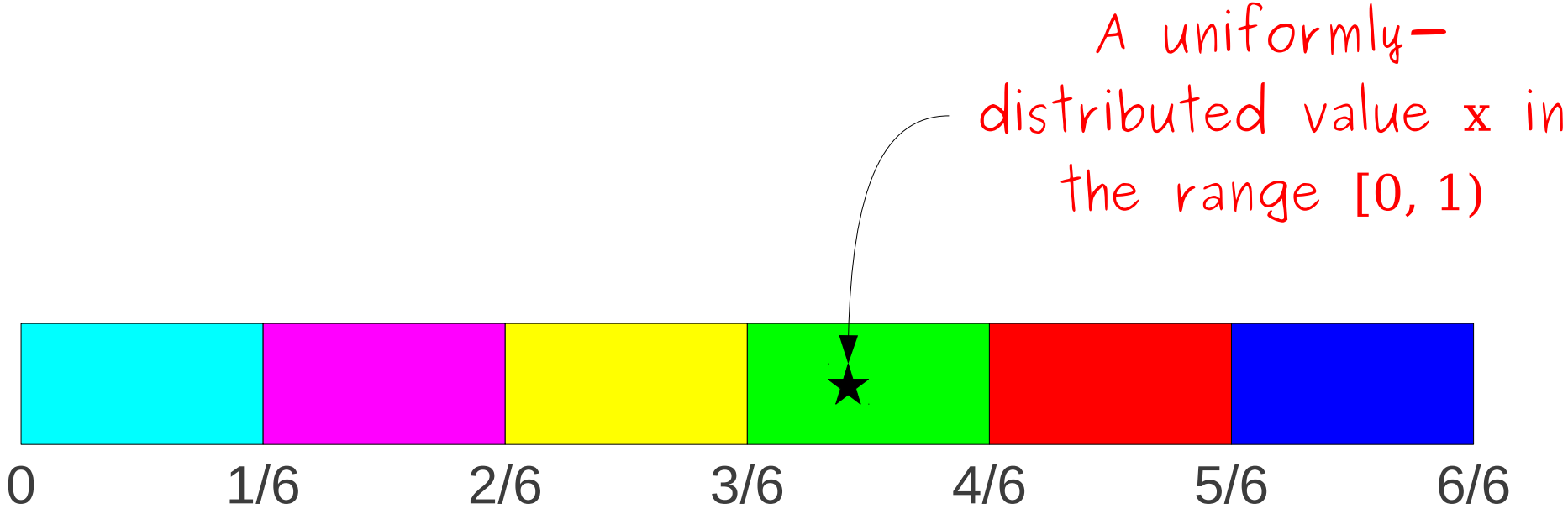
A uniformly-distributed value x in the range $[0, 1)$



What is the largest integer k such that $k < 6x$?

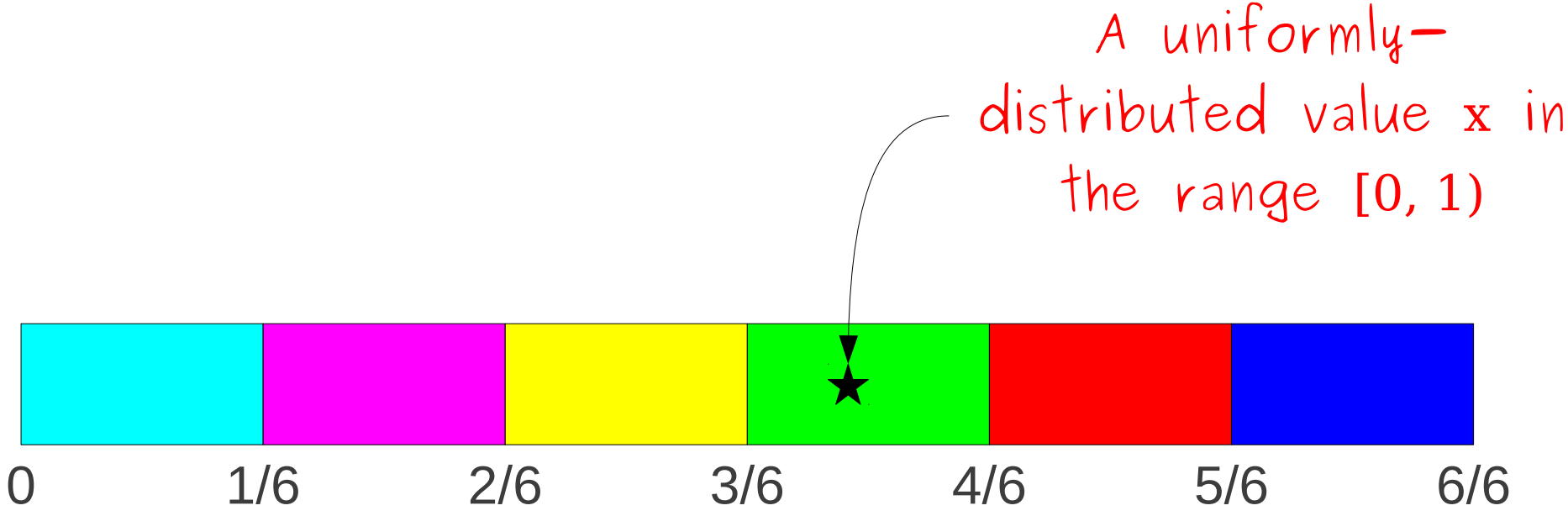
$$k = \lfloor 6x \rfloor$$

Simulating a Fair Six-Sided Die



1. Generate a random value $x \in [0, 1)$
2. Output $k = \lfloor 6x \rfloor$

Simulating a Fair Six-Sided Die

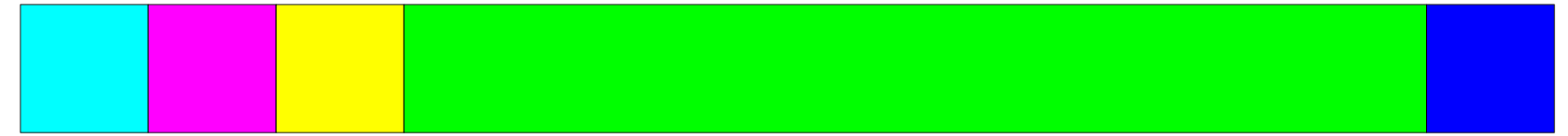


1. Generate a random value $x \in [0, 1)$
2. Output $k = \lfloor 6x \rfloor$

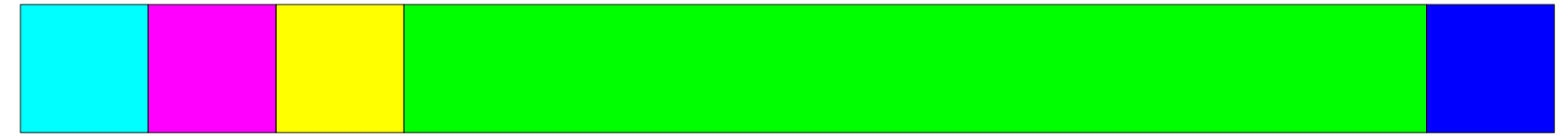
Time required: **O(1)**

Simulating a Fair n-Sided Die

Simulating a Fair n-Sided Die



Simulating a Fair n-Sided Die



1. Generate a random value $x \in [0, 1)$
2. Output $k = \lfloor nx \rfloor$

Time required: **O(1)**

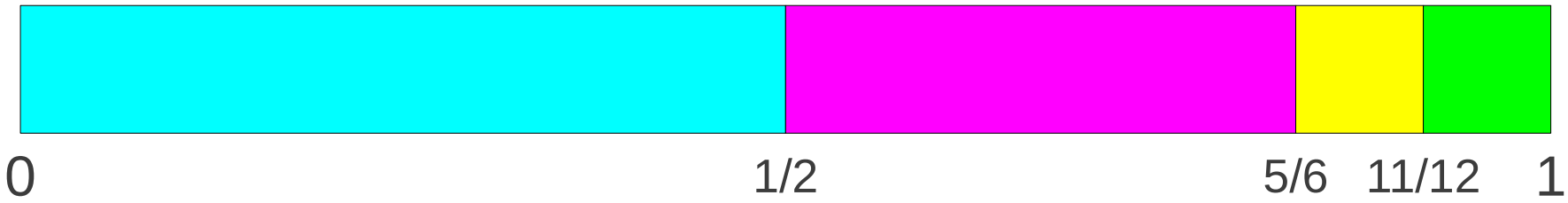
Can we simulate a loaded die with a fair die?

A Sample Loaded Die

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

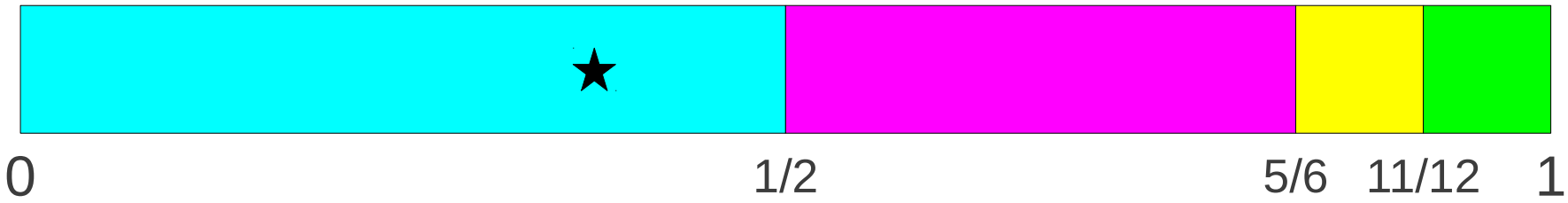
A Sample Loaded Die

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



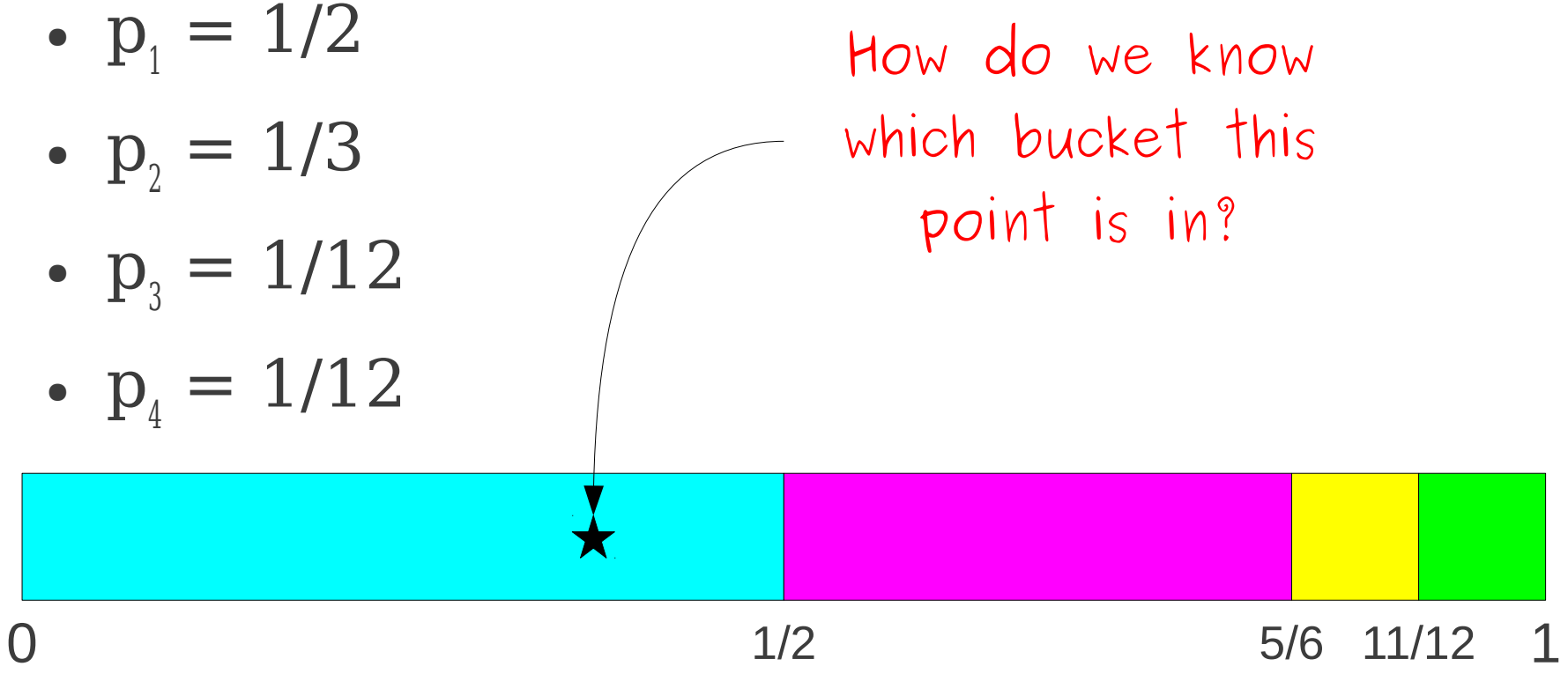
A Sample Loaded Die

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



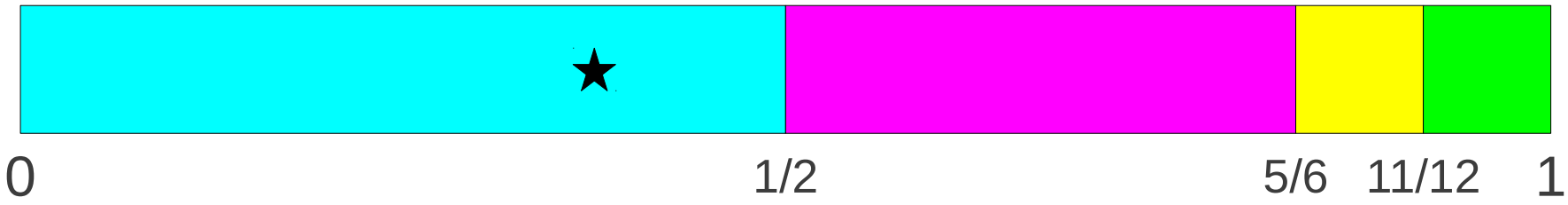
A Sample Loaded Die

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



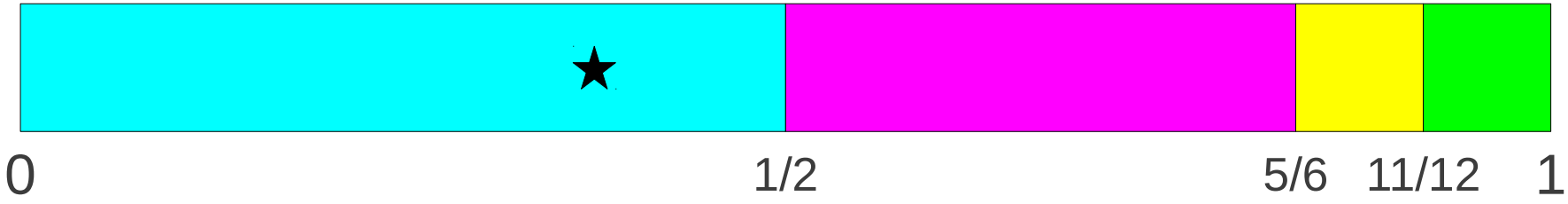
A Sample Loaded Die

- A four-sided die:
  - $p_1 = 6/12$
  - $p_2 = 4/12$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



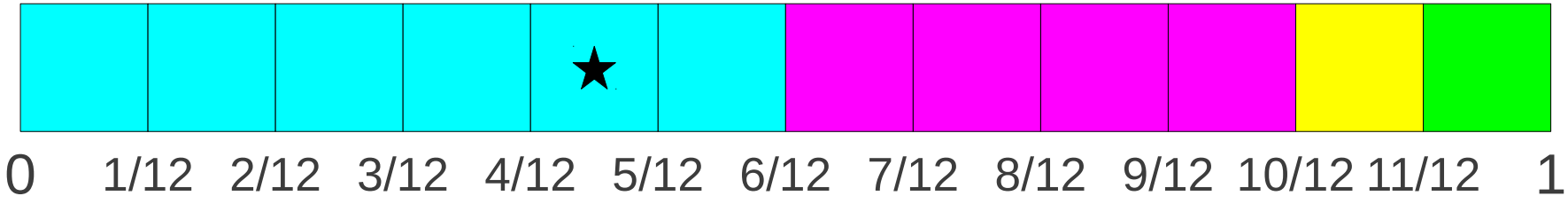
A Sample Loaded Die

- A four-sided die:
  - $p_1 = 6/12$
  - $p_2 = 4/12$
  - $p_3 = 1/12$
  - $p_4 = 1/12$
- A 12-sided fair die:
  - Six sides labeled 1
  - Four sides labeled 2
  - One side labeled 3
  - One side labeled 4



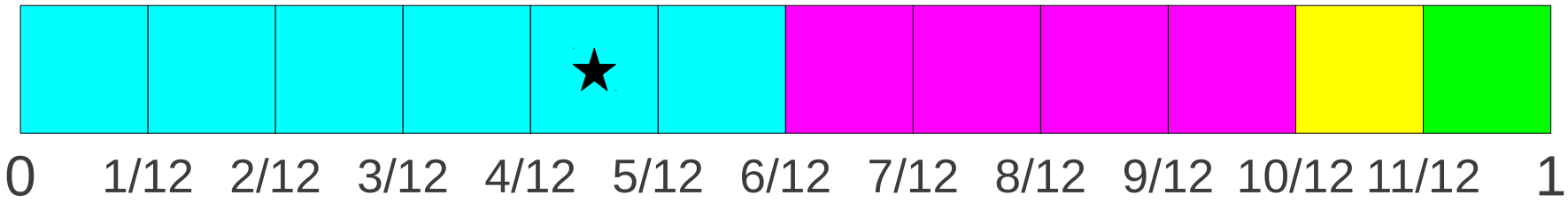
A Sample Loaded Die

- A four-sided die:
  - $p_1 = 6/12$
  - $p_2 = 4/12$
  - $p_3 = 1/12$
  - $p_4 = 1/12$
- A 12-sided fair die:
  - Six sides labeled 1
  - Four sides labeled 2
  - One side labeled 3
  - One side labeled 4



A Sample Loaded Die

- A four-sided die:
  - $p_1 = 6/12$
  - $p_2 = 4/12$
  - $p_3 = 1/12$
  - $p_4 = 1/12$
- A 12-sided fair die:
  - Six sides labeled 1
  - Four sides labeled 2
  - One side labeled 3
  - One side labeled 4



1	1	1	1	1	1	2	2	2	2	3	4

Loaded Dice from Fair Dice

- Normalize the probabilities so that they have the same denominator d.
- Create a fair d-sided die, and label the sides with a frequency that guarantees the original probabilities.
- Roll the fair die, then report the result.

The Catch

- The common denominator can be *huge*!
- Example: 1/1,000,000 and 999,999/1,000,000 requires a one-million-sided die to simulate a two-sided die.
  - Can require arbitrarily many sides with just two outcomes.
- Storing the table necessary to map back to the original distribution is completely infeasible here.
- Algorithm impractical unless we know something about the inputs in advance.

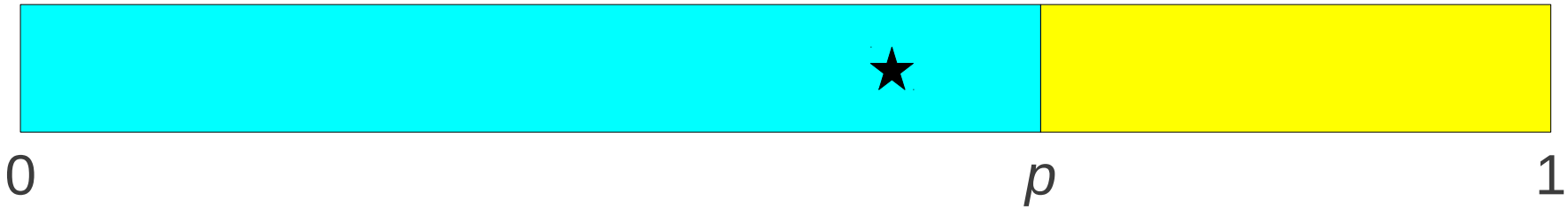
Simulating a Biased Coin

Simulating a Biased Coin

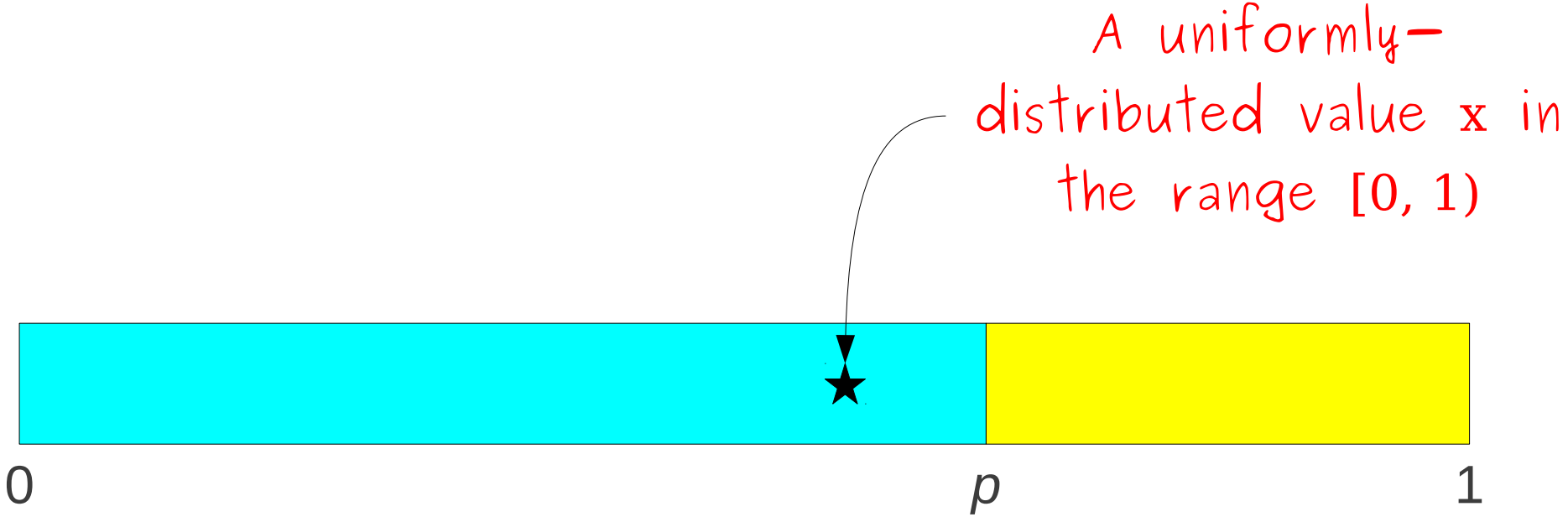
Simulating a Biased Coin

Simulating a Biased Coin

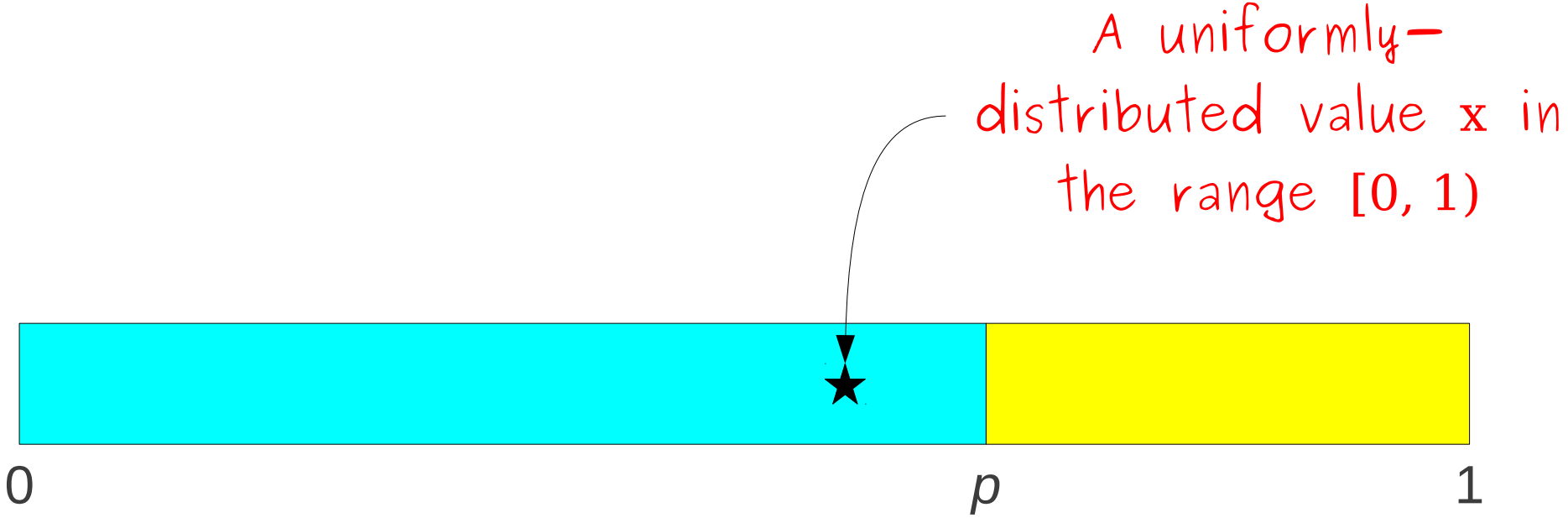
Simulating a Biased Coin



Simulating a Biased Coin

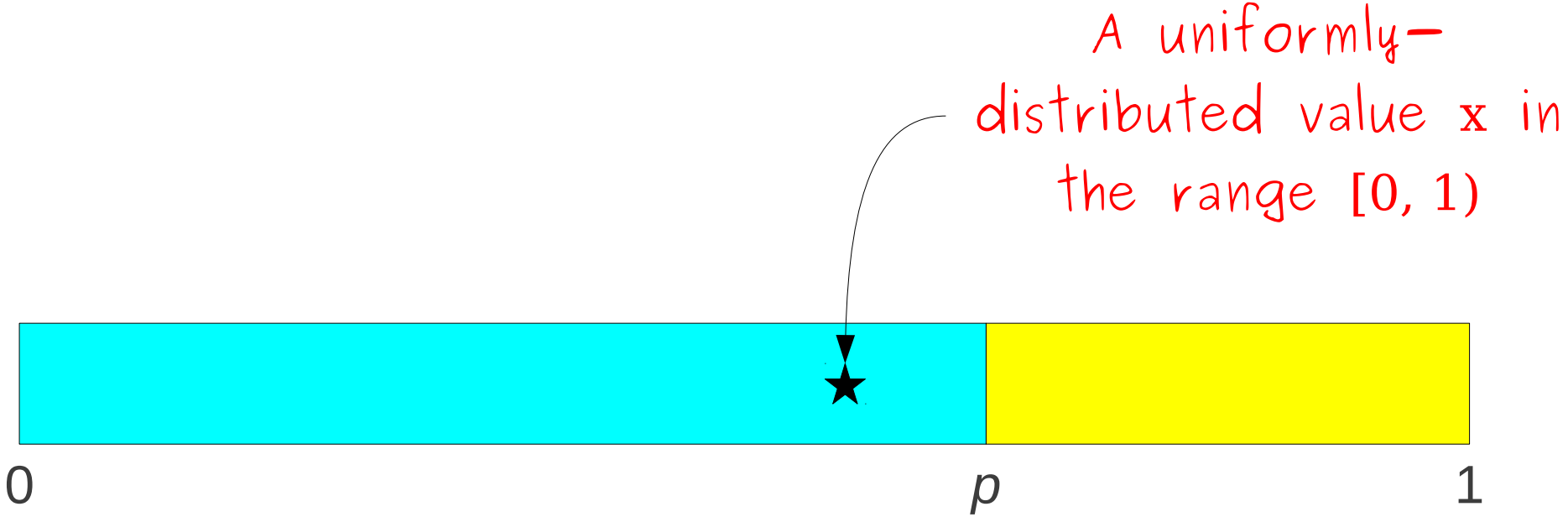


Simulating a Biased Coin



What bucket does x belong to?

Simulating a Biased Coin



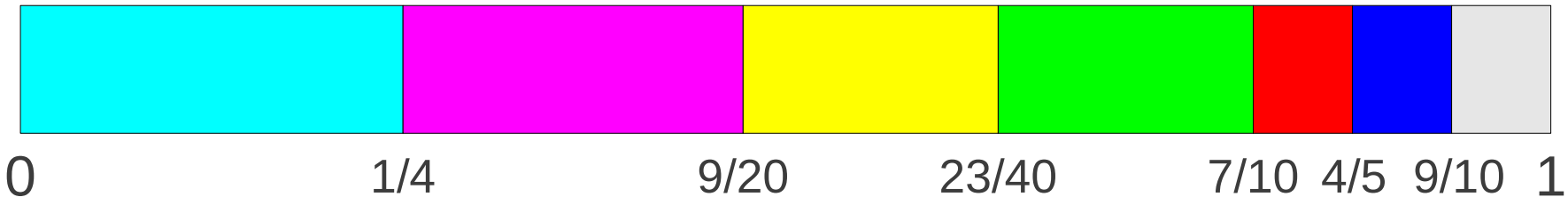
If x < p, output “heads”
Otherwise output “tails.”

Generalizing This Idea

- A 7-sided die:
  - $p_1 = 1/4$
  - $p_2 = 1/5$
  - $p_3 = 1/8$
  - $p_4 = 1/8$
  - $p_5 = 1/10$
  - $p_6 = 1/10$
  - $p_7 = 1/10$

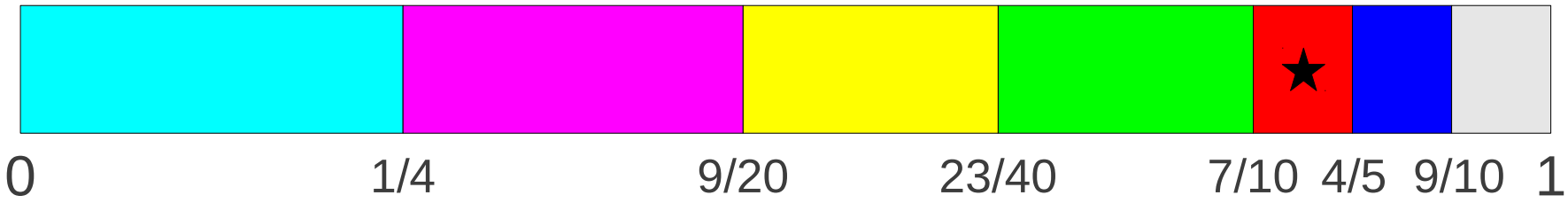
Generalizing This Idea

- A 7-sided die:
  - $p_1 = 1/4$
  - $p_2 = 1/5$
  - $p_3 = 1/8$
  - $p_4 = 1/8$
  - $p_5 = 1/10$
  - $p_6 = 1/10$
  - $p_7 = 1/10$



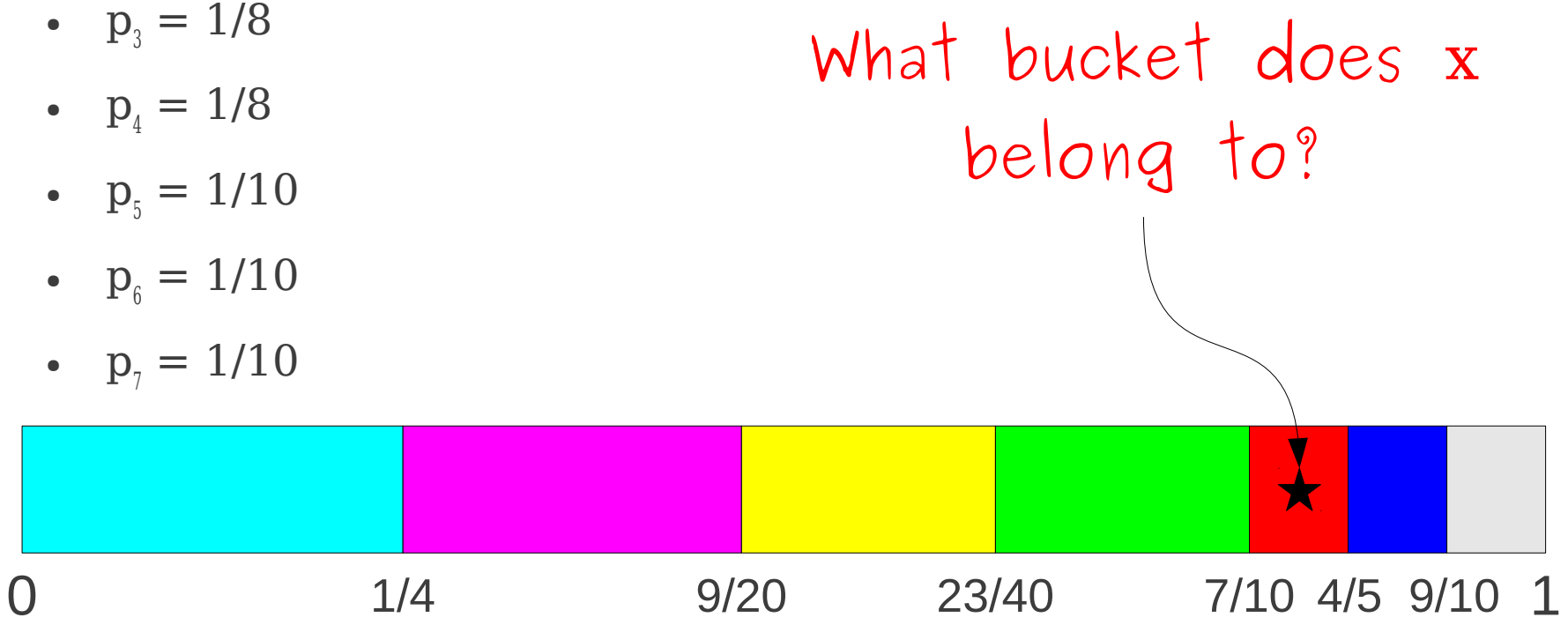
Generalizing This Idea

- A 7-sided die:
  - $p_1 = 1/4$
  - $p_2 = 1/5$
  - $p_3 = 1/8$
  - $p_4 = 1/8$
  - $p_5 = 1/10$
  - $p_6 = 1/10$
  - $p_7 = 1/10$



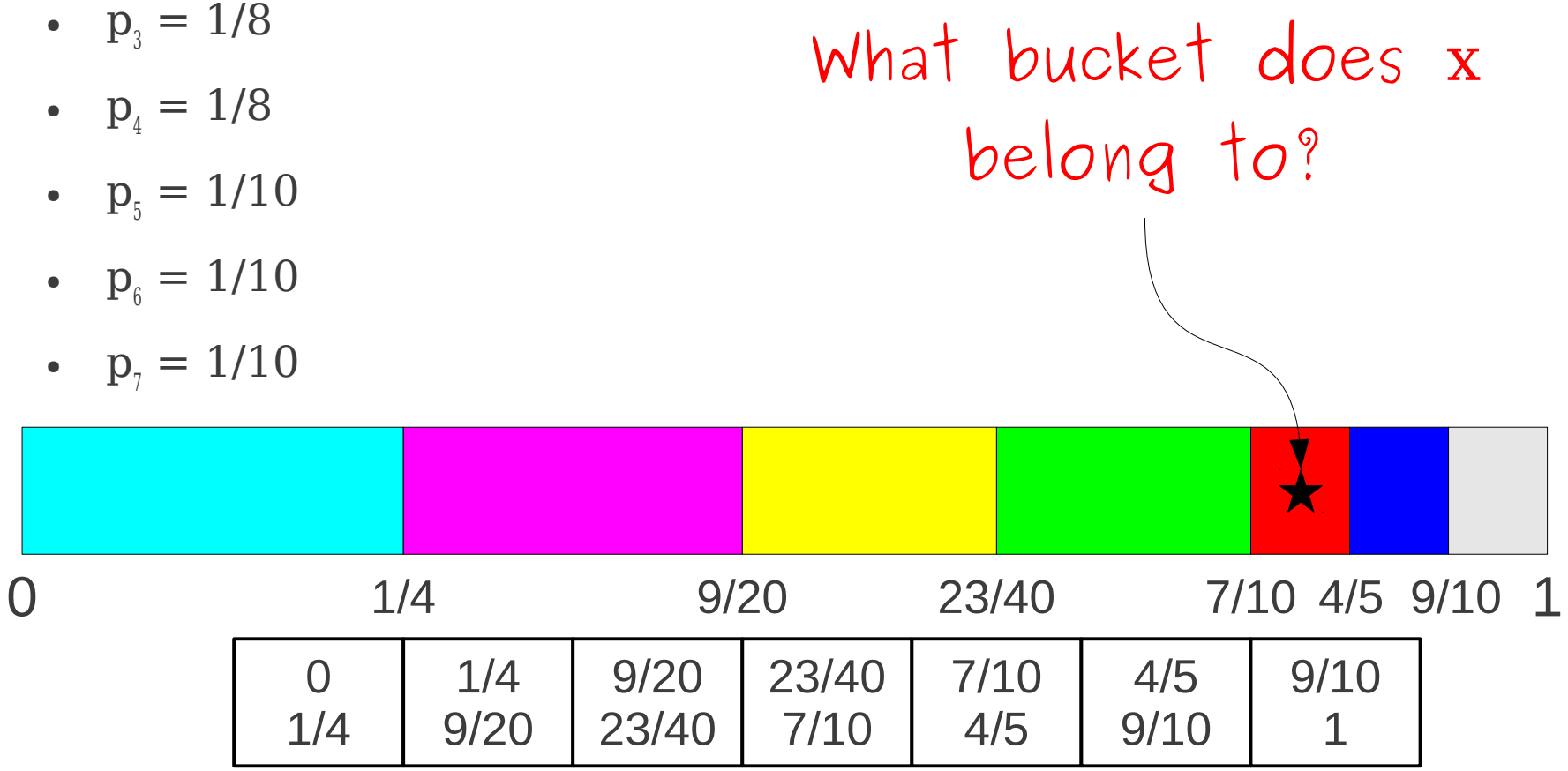
Generalizing This Idea

- A 7-sided die:
  - $p_1 = 1/4$
  - $p_2 = 1/5$
  - $p_3 = 1/8$
  - $p_4 = 1/8$
  - $p_5 = 1/10$
  - $p_6 = 1/10$
  - $p_7 = 1/10$



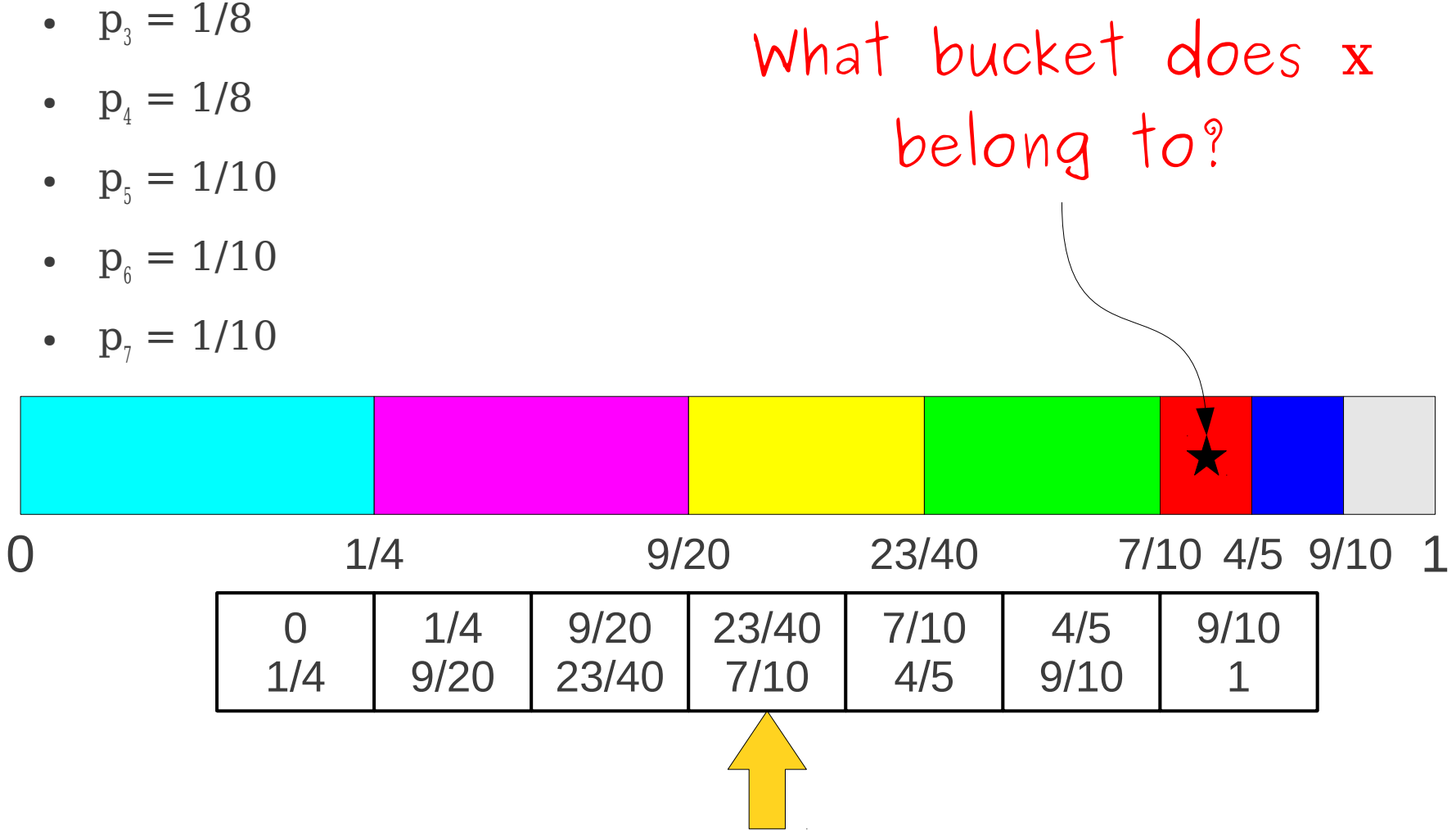
Generalizing This Idea

- A 7-sided die:
  - $p_1 = 1/4$
  - $p_2 = 1/5$
  - $p_3 = 1/8$
  - $p_4 = 1/8$
  - $p_5 = 1/10$
  - $p_6 = 1/10$
  - $p_7 = 1/10$



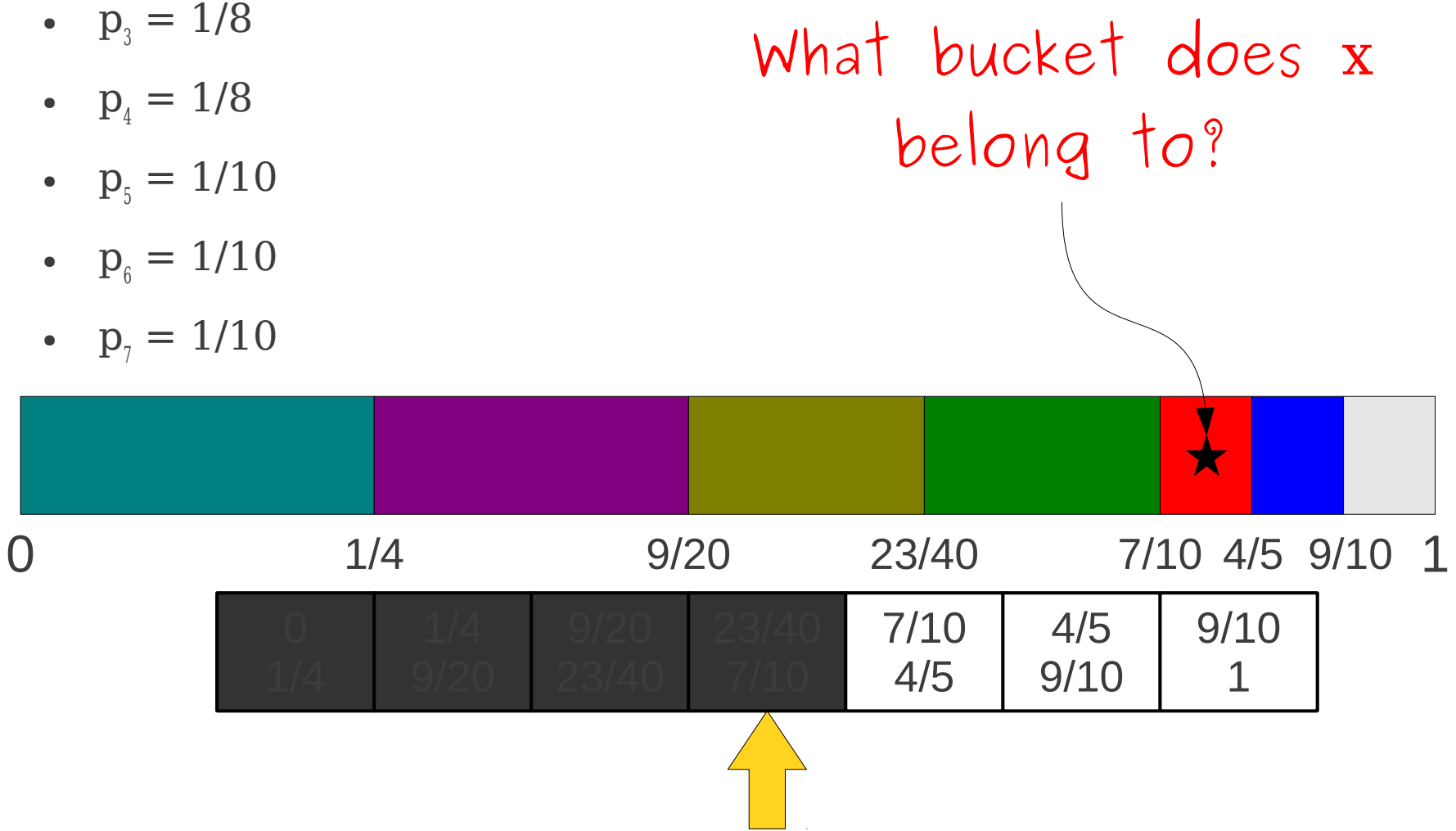
Generalizing This Idea

- A 7-sided die:
  - $p_1 = 1/4$
  - $p_2 = 1/5$
  - $p_3 = 1/8$
  - $p_4 = 1/8$
  - $p_5 = 1/10$
  - $p_6 = 1/10$
  - $p_7 = 1/10$



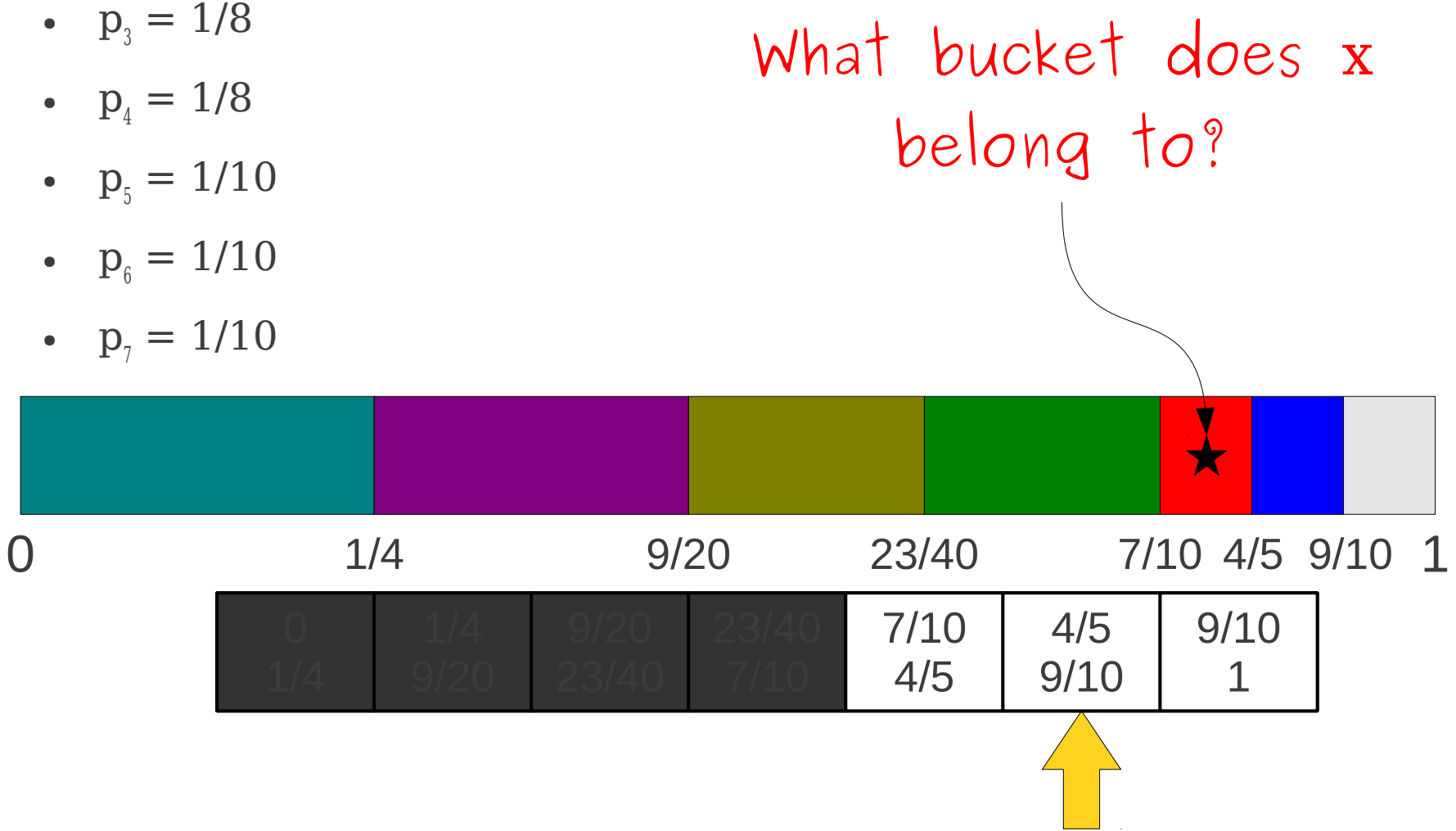
Generalizing This Idea

- A 7-sided die:
  - $p_1 = 1/4$
  - $p_2 = 1/5$
  - $p_3 = 1/8$
  - $p_4 = 1/8$
  - $p_5 = 1/10$
  - $p_6 = 1/10$
  - $p_7 = 1/10$



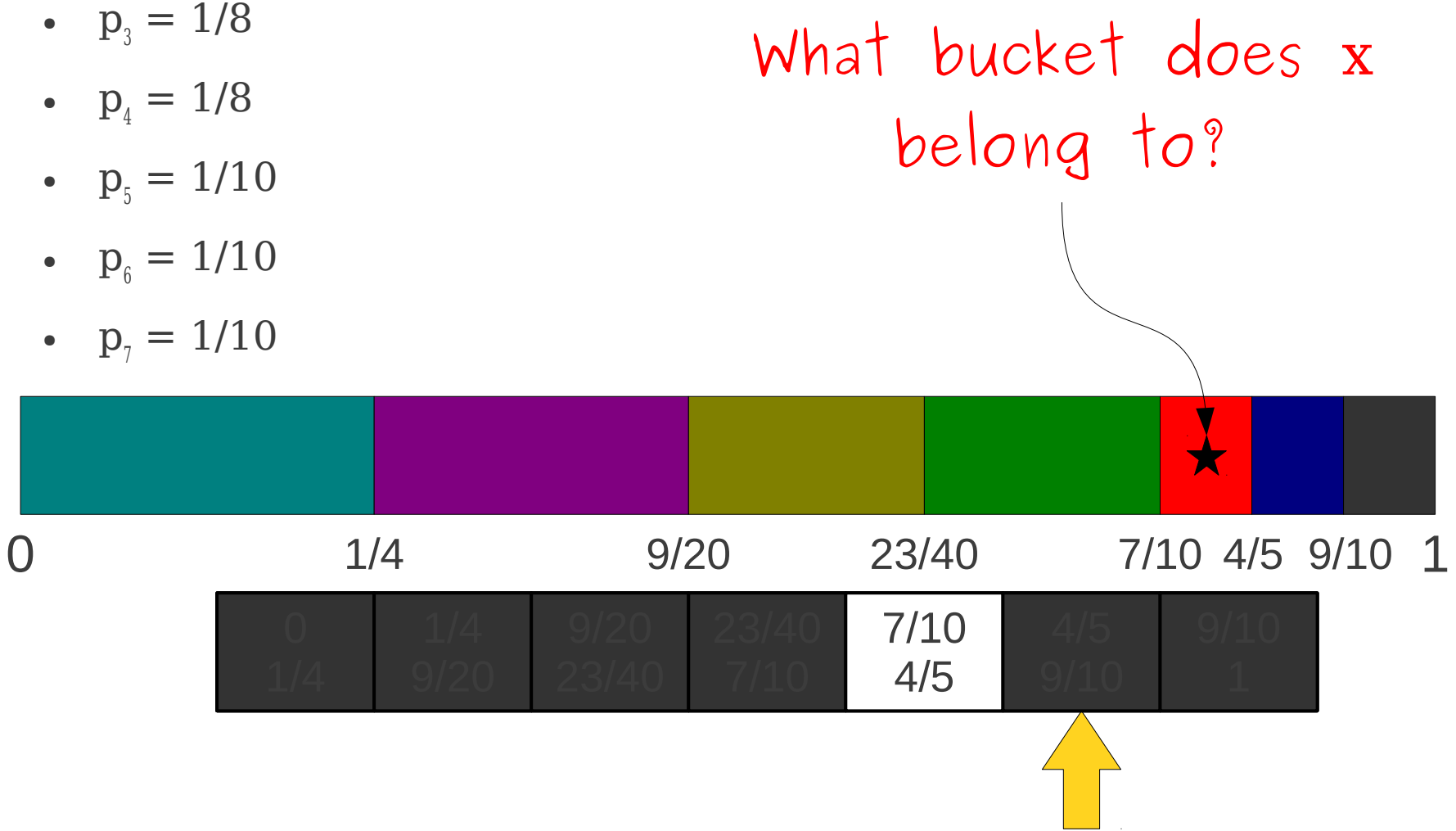
Generalizing This Idea

- A 7-sided die:
  - $p_1 = 1/4$
  - $p_2 = 1/5$
  - $p_3 = 1/8$
  - $p_4 = 1/8$
  - $p_5 = 1/10$
  - $p_6 = 1/10$
  - $p_7 = 1/10$



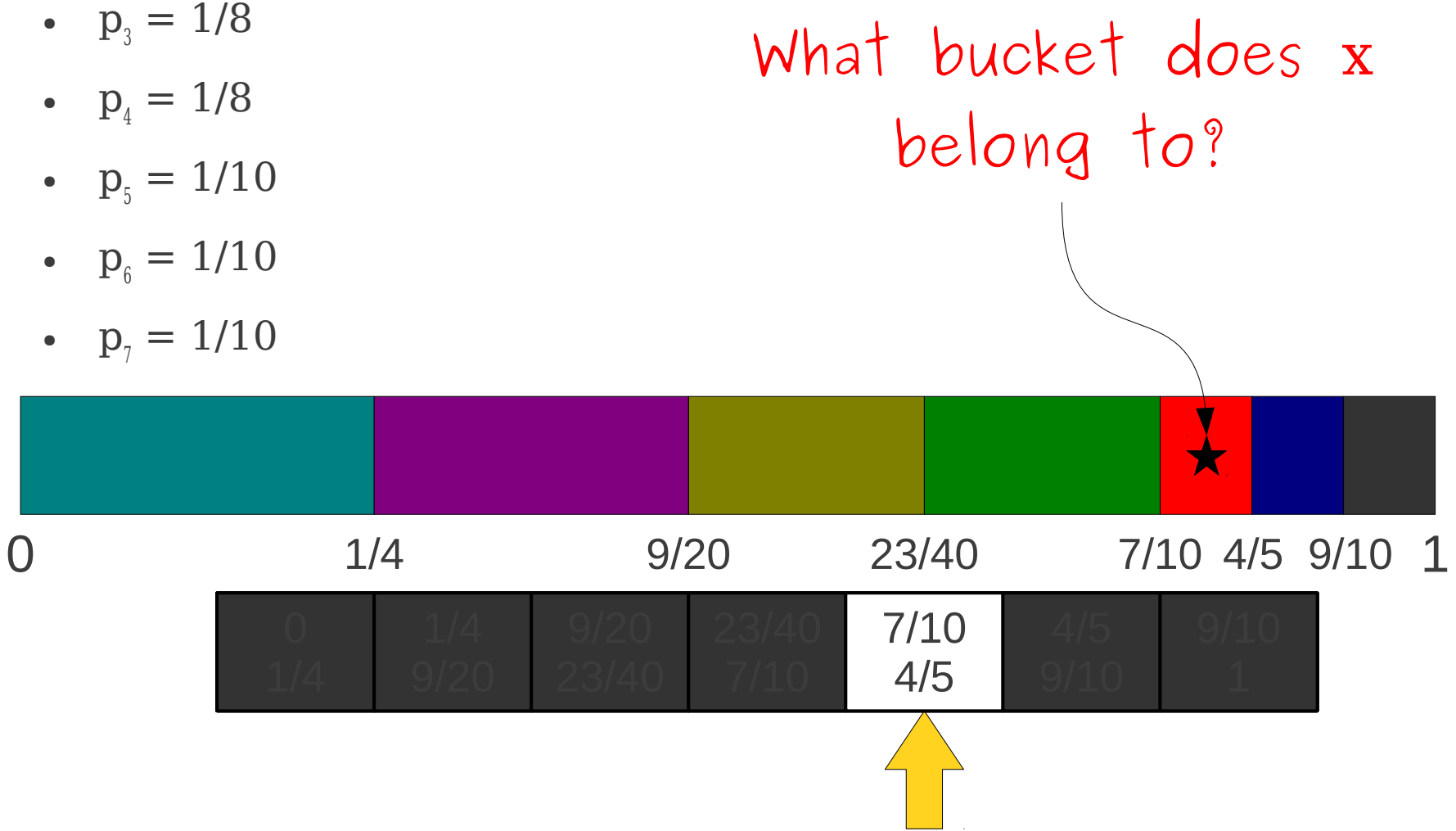
Generalizing This Idea

- A 7-sided die:
  - $p_1 = 1/4$
  - $p_2 = 1/5$
  - $p_3 = 1/8$
  - $p_4 = 1/8$
  - $p_5 = 1/10$
  - $p_6 = 1/10$
  - $p_7 = 1/10$



Generalizing This Idea

- A 7-sided die:
  - $p_1 = 1/4$
  - $p_2 = 1/5$
  - $p_3 = 1/8$
  - $p_4 = 1/8$
  - $p_5 = 1/10$
  - $p_6 = 1/10$
  - $p_7 = 1/10$



Roulette Wheel Selection

- Construct a **cumulative probability table** containing a running total of the probabilities so far.
- To simulate the loaded die:
  - Generate a random value in [0, 1).
  - Do a binary search on the table for the bucket.
- Initialization time: $\Theta(n)$
- Generation time: $O(\log n)$

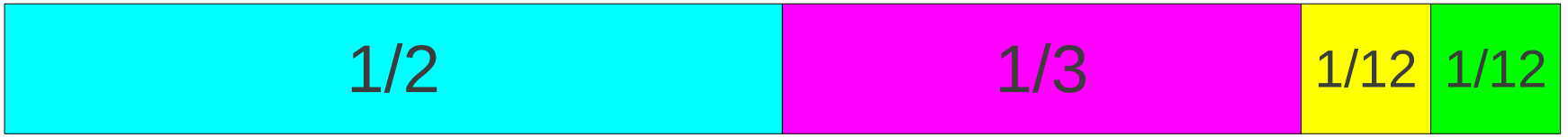
Throwing Darts

A Sample Loaded Die

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

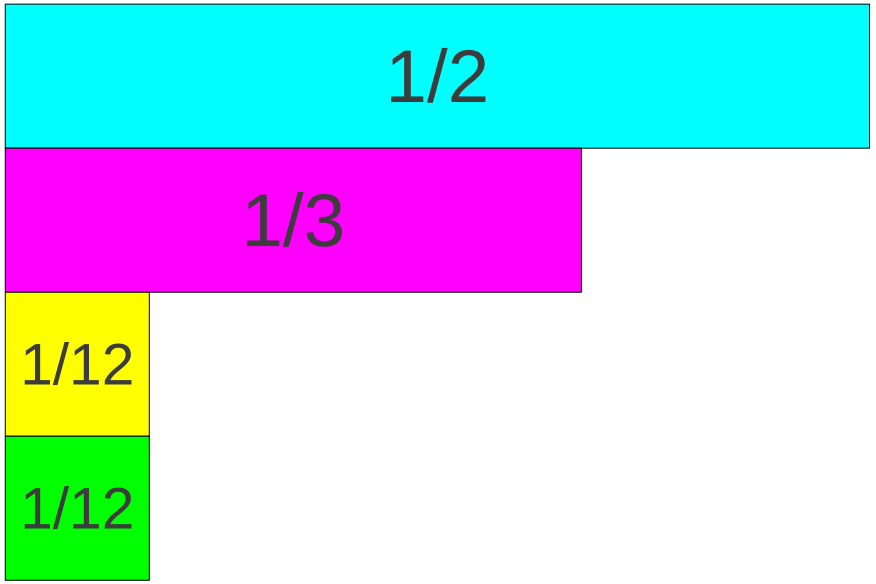
A Sample Loaded Die

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



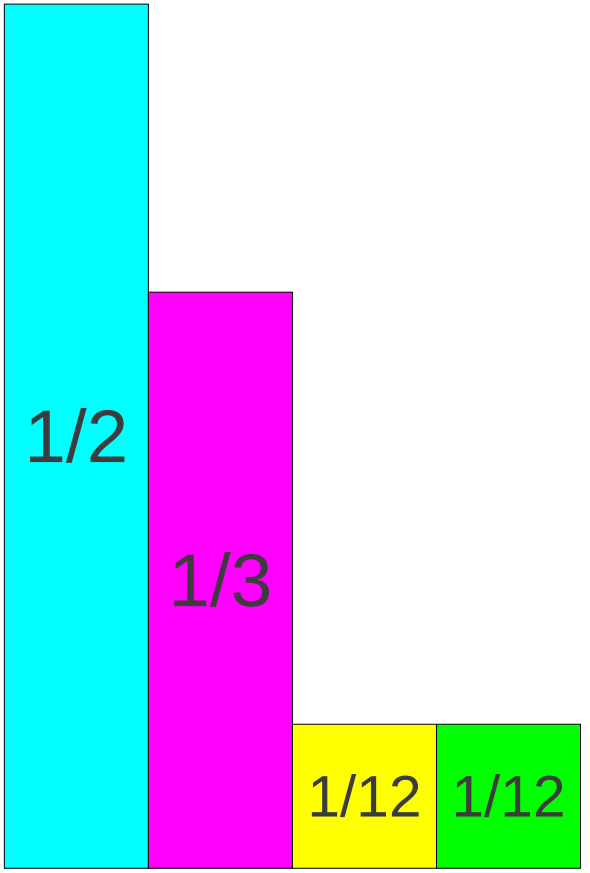
A Sample Loaded Die

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



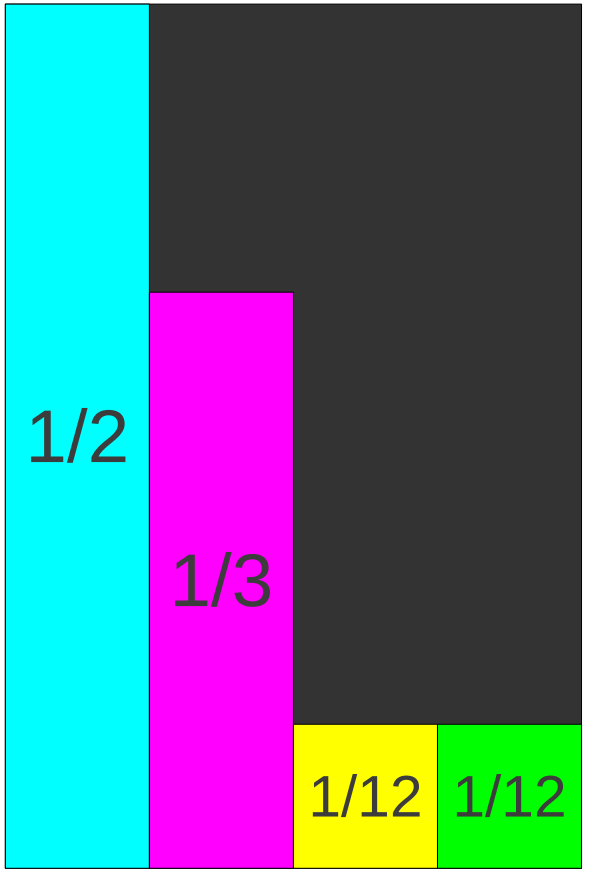
A Sample Loaded Die

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



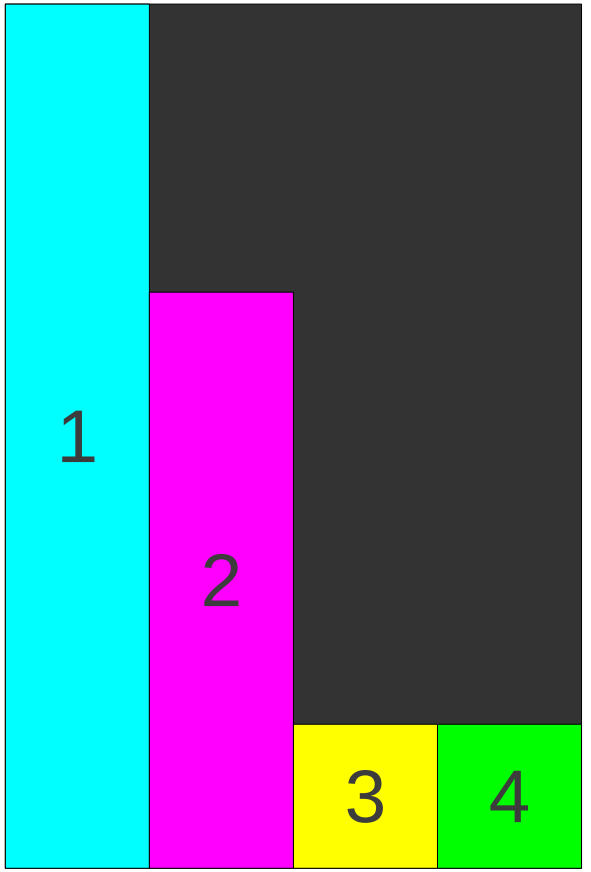
A Sample Loaded Die

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



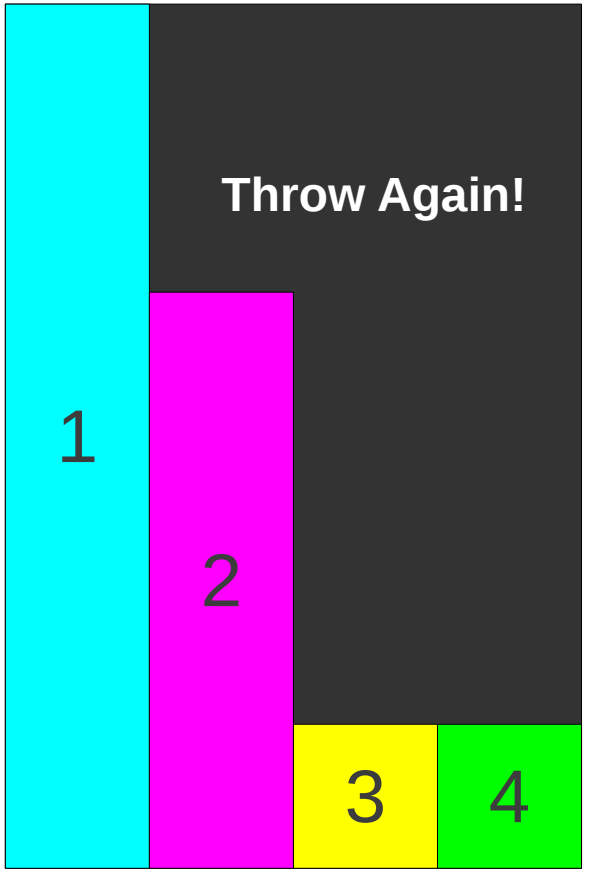
A Sample Loaded Die

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



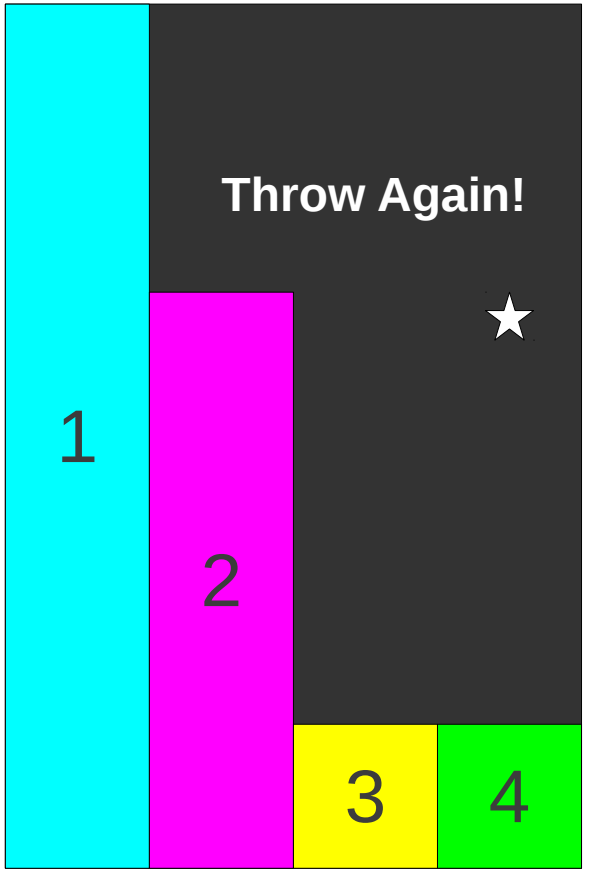
A Sample Loaded Die

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



A Sample Loaded Die

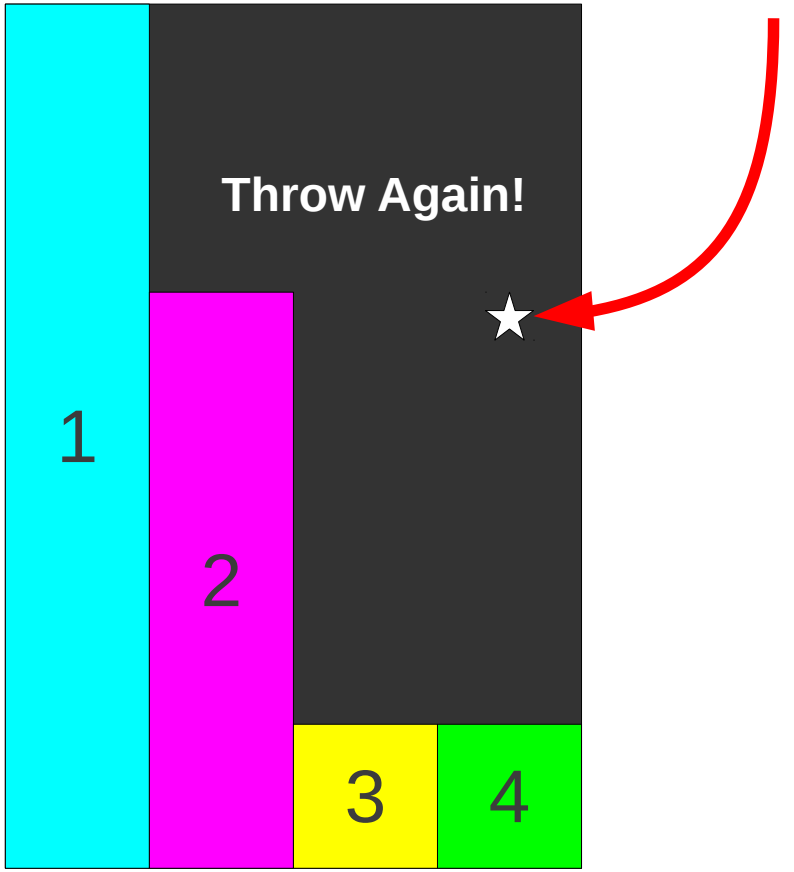
- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



A Sample Loaded Die

How do we know what we hit?

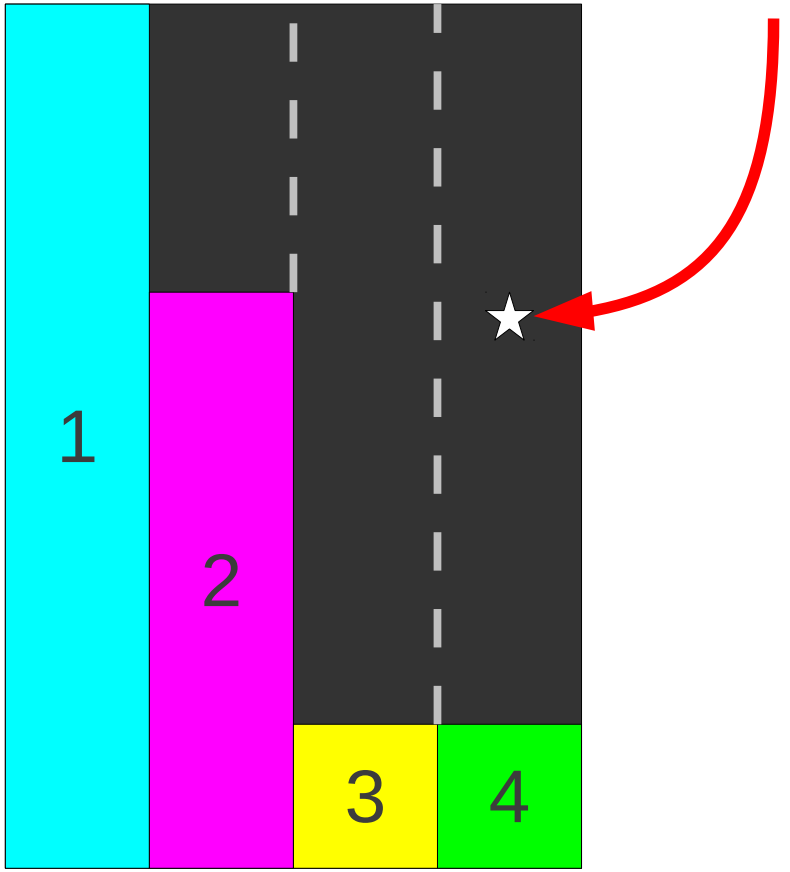
- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



A Sample Loaded Die

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

How do we know what we hit?

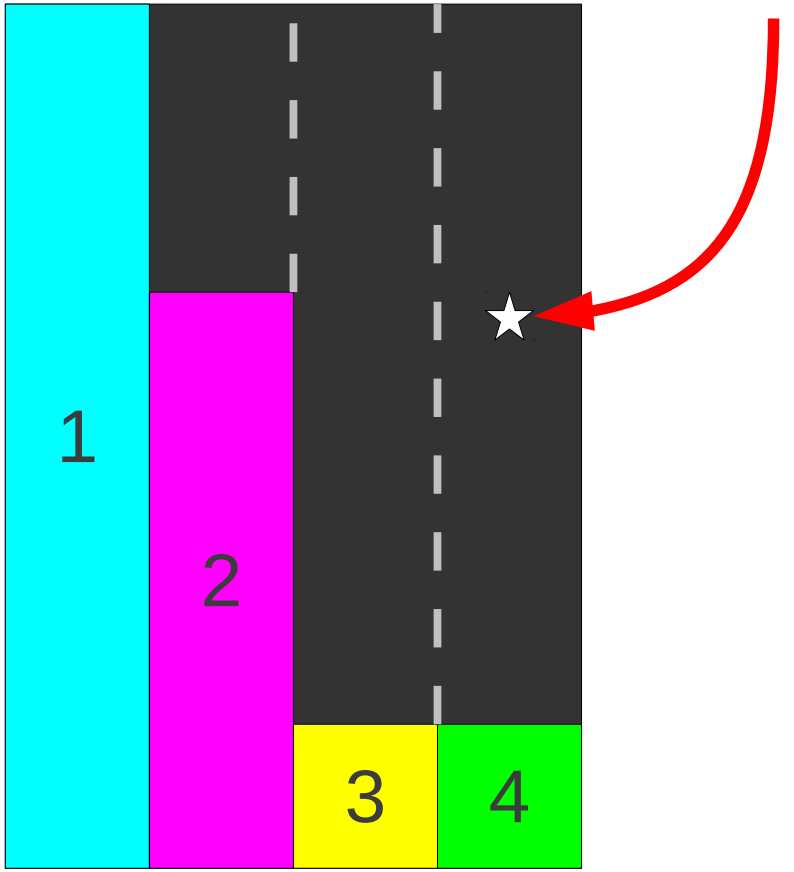


A Sample Loaded Die

How do we know what we hit?

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

Each column has the same width.

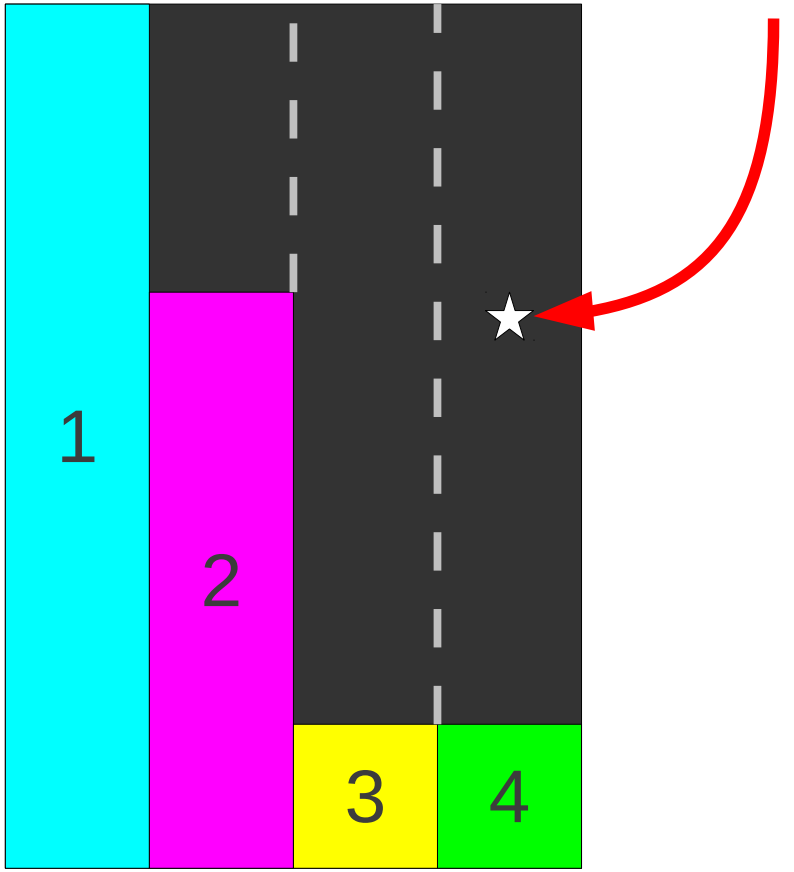


A Sample Loaded Die

How do we know what we hit?

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

Choose the column by rolling a fair die.



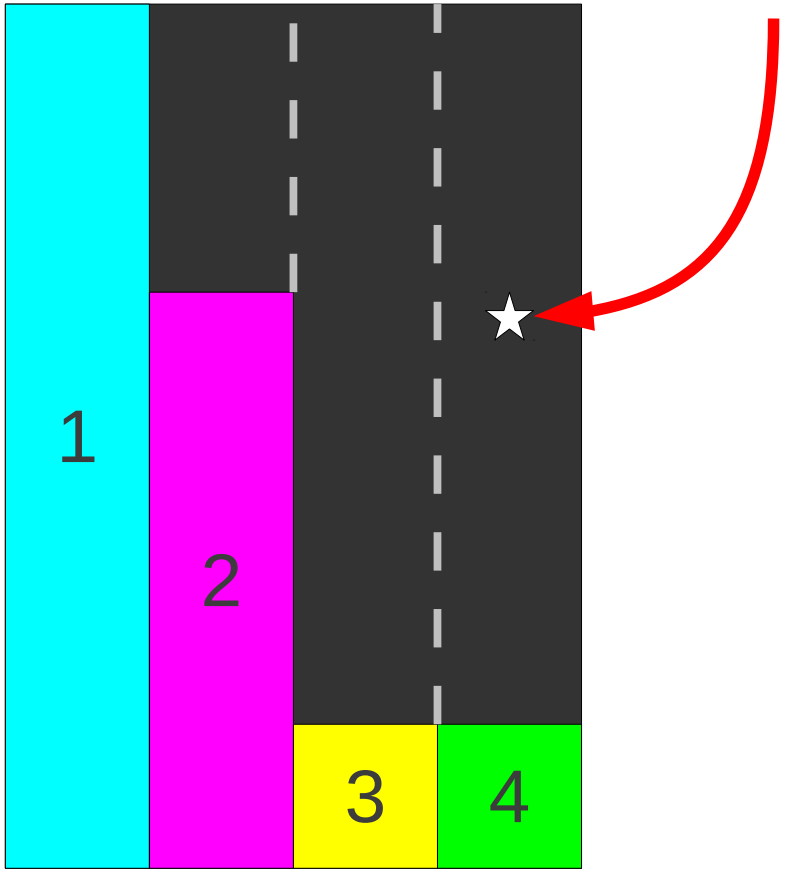
A Sample Loaded Die

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

Choose the column by rolling a fair die.

Each column has only two options.

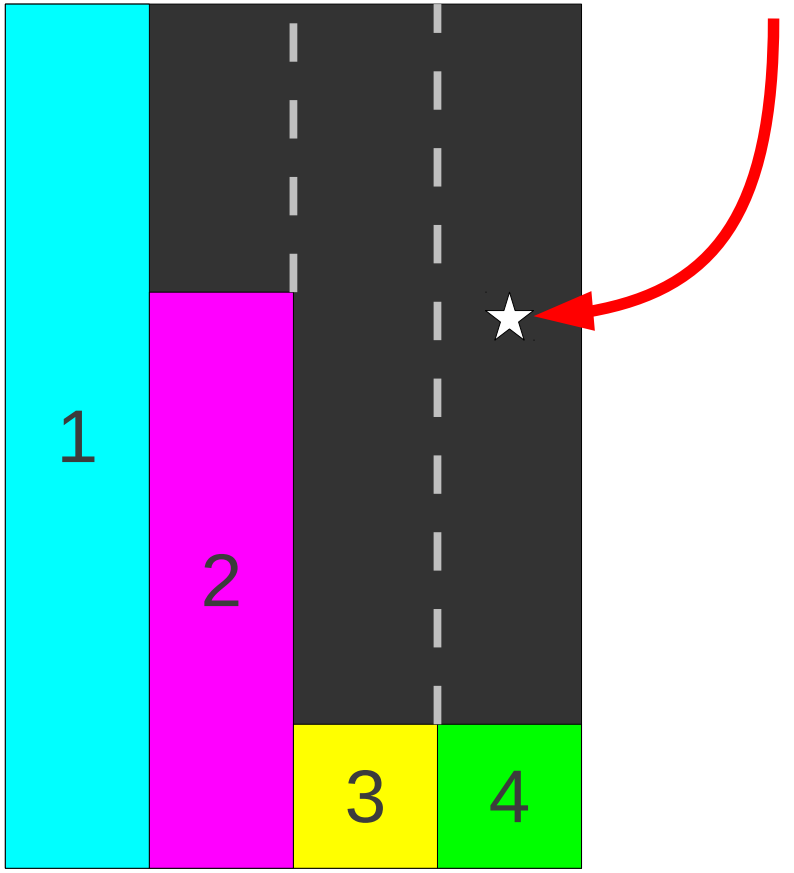
How do we know what we hit?



A Sample Loaded Die

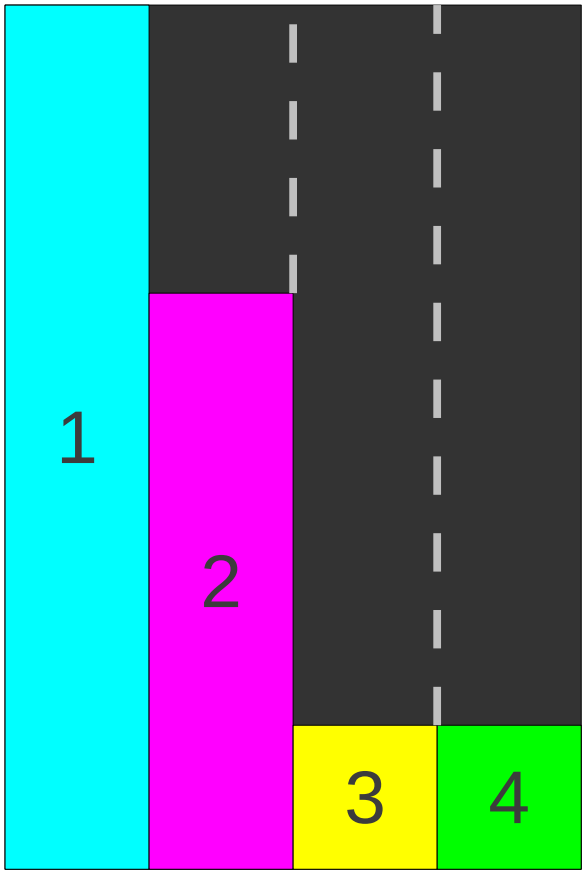
How do we know what we hit?

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

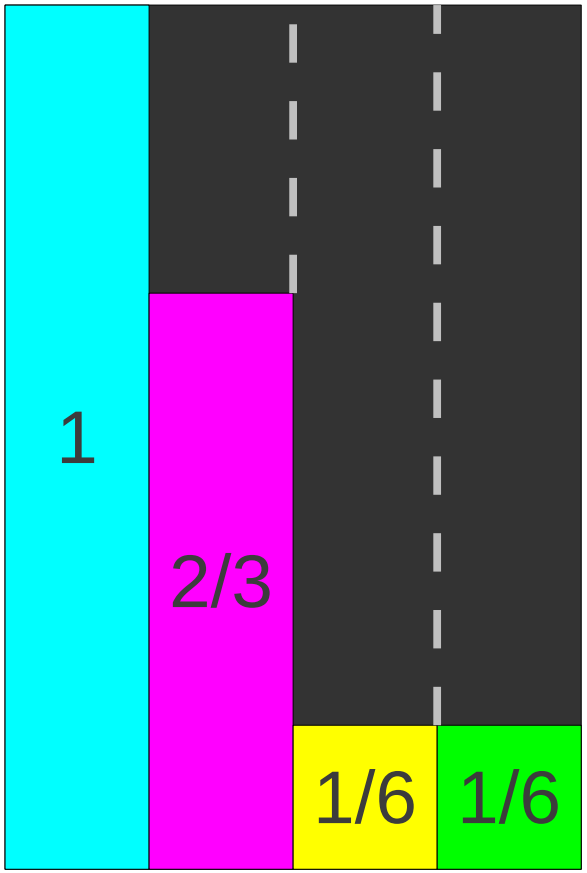


Choose the column by rolling a fair die.

Choose which part of the column by flipping a biased coin.

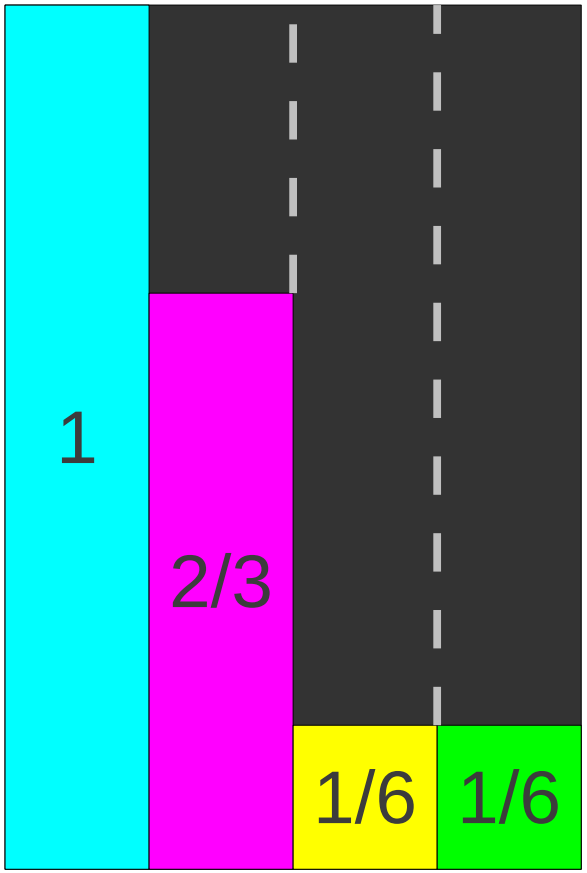


Throwing a dart at this board is equivalent to rolling a fair die, then flipping one of many biased coins.



Throwing a dart at this board is equivalent to rolling a fair die, then flipping one of many biased coins.

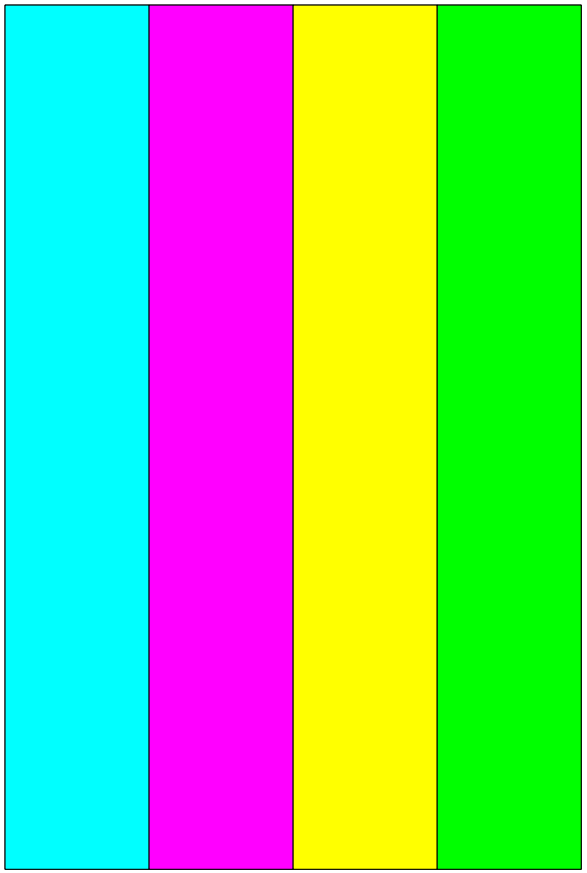
The Dartboard Algorithm



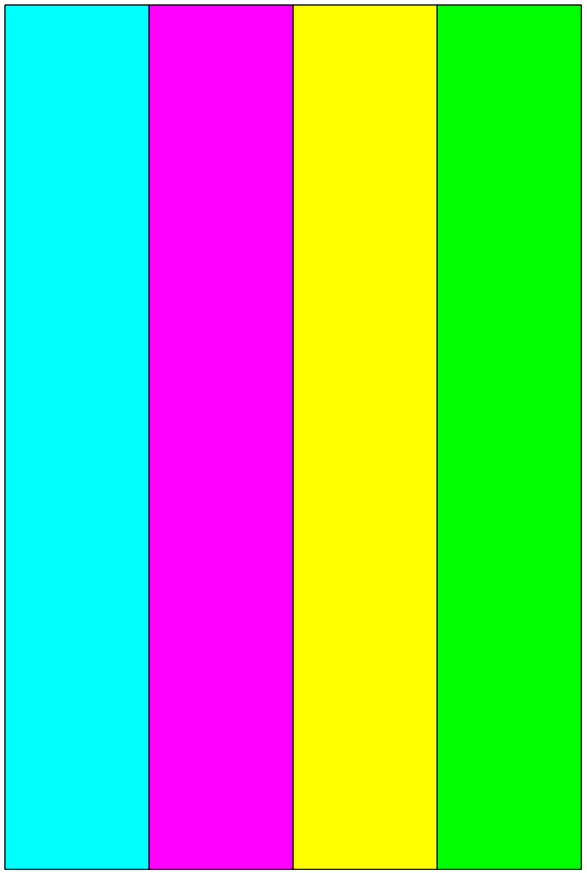
- Build the target by dividing all probabilities by the largest probability.
- Until you choose a side:
  - Roll a fair die to choose a column.
  - Flip the coin in that column.
  - If it's heads, output the column.

Not All Dartboards are Equal

Not All Dartboards are Equal

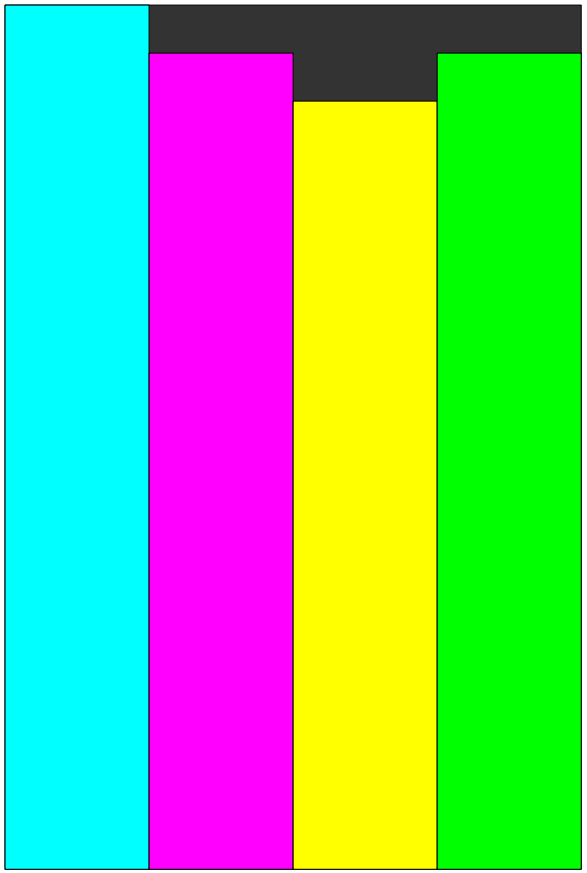


Not All Dartboards are Equal



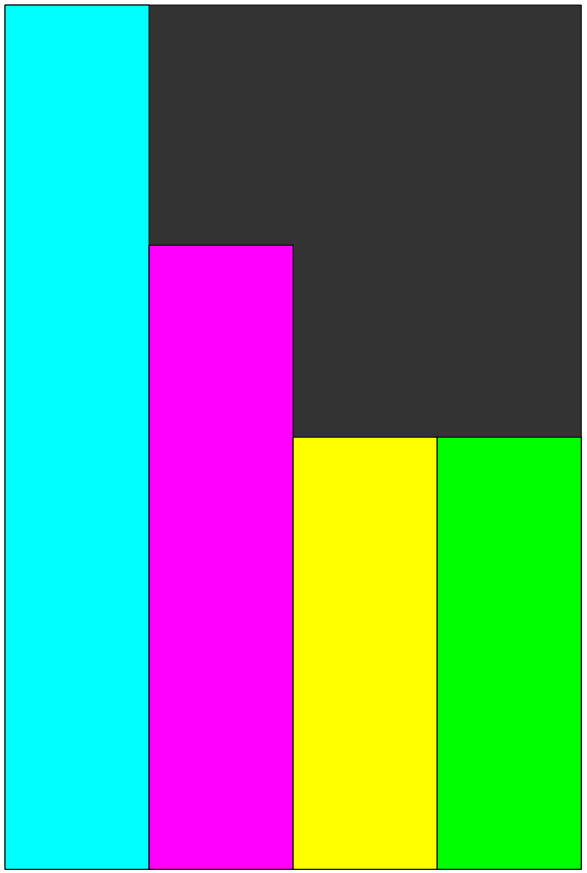
- If the die is close to being fair, the probability of missing is very low.

Not All Dartboards are Equal



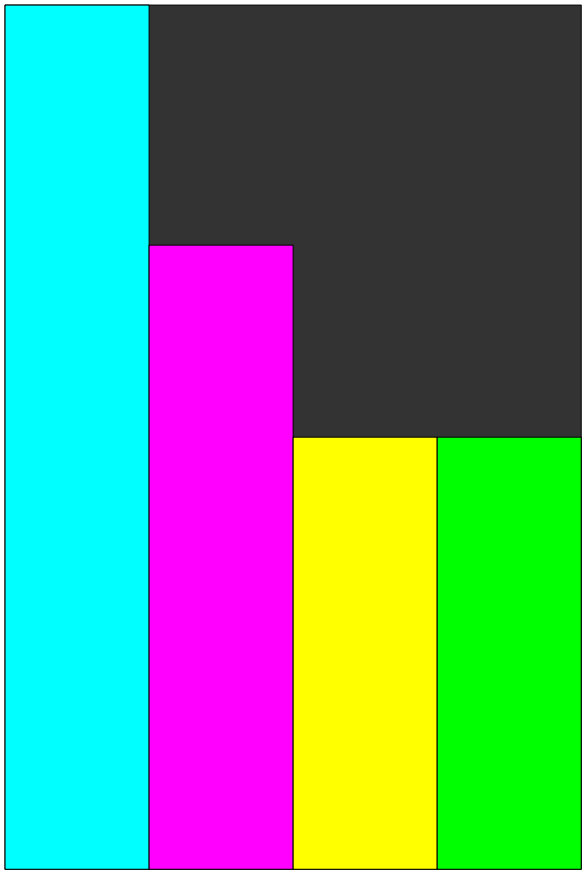
- If the die is close to being fair, the probability of missing is very low.

Not All Dartboards are Equal



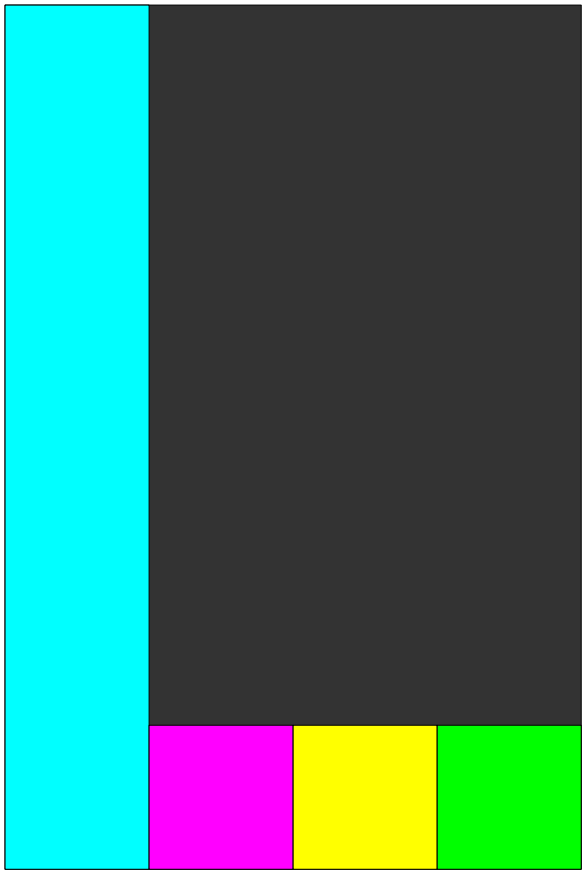
- If the die is close to being fair, the probability of missing is very low.

Not All Dartboards are Equal



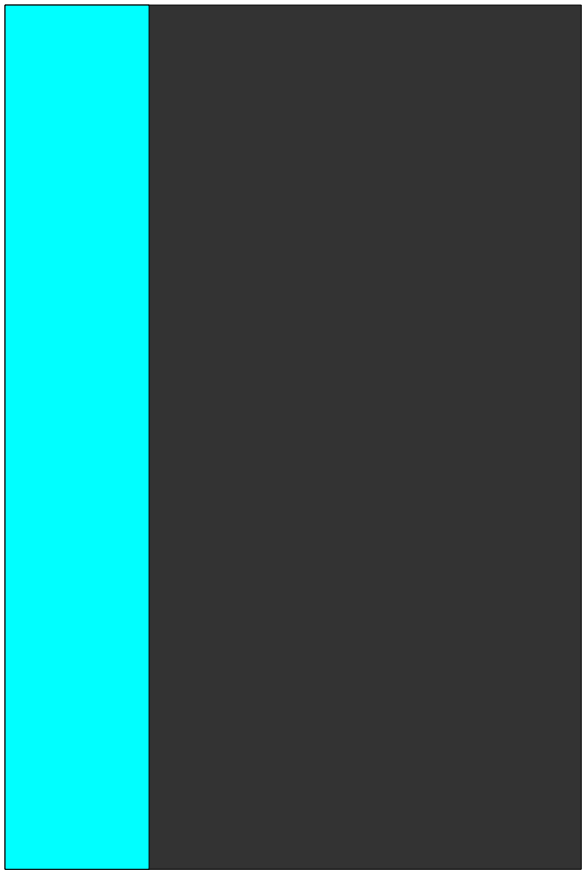
- If the die is close to being fair, the probability of missing is very low.
- As the die becomes more unfair, the probability of missing keeps increasing.

Not All Dartboards are Equal



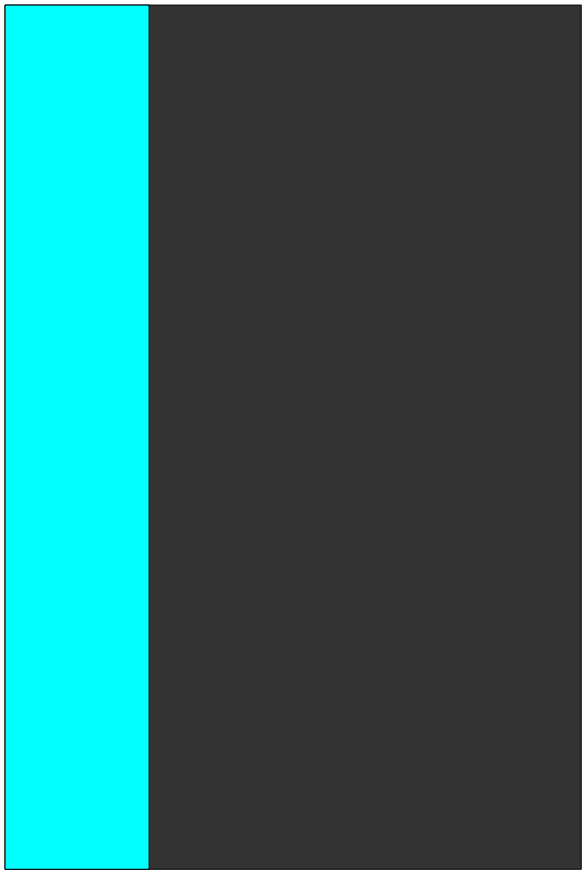
- If the die is close to being fair, the probability of missing is very low.
- As the die becomes more unfair, the probability of missing keeps increasing.

Not All Dartboards are Equal



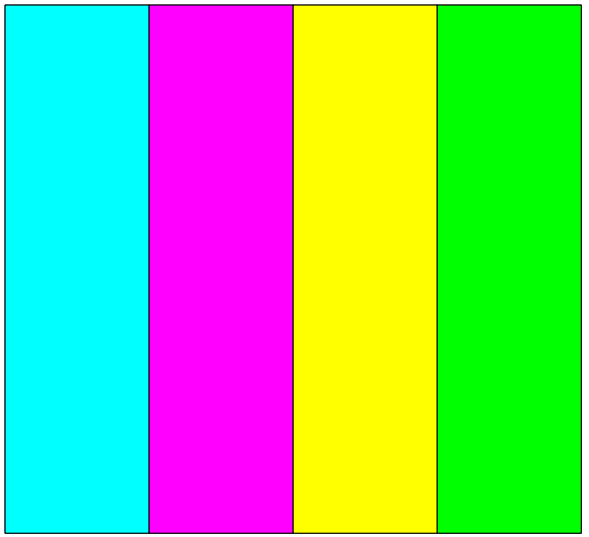
- If the die is close to being fair, the probability of missing is very low.
- As the die becomes more unfair, the probability of missing keeps increasing.

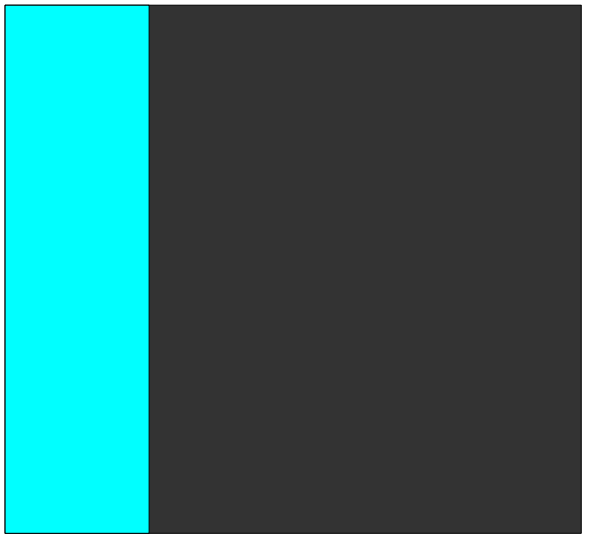
Not All Dartboards are Equal



- If the die is close to being fair, the probability of missing is very low.
- As the die becomes more unfair, the probability of missing keeps increasing.
- In the limit, the dartboard might have n – 1 out of n columns always fail.

Not All Dartboards are Equal



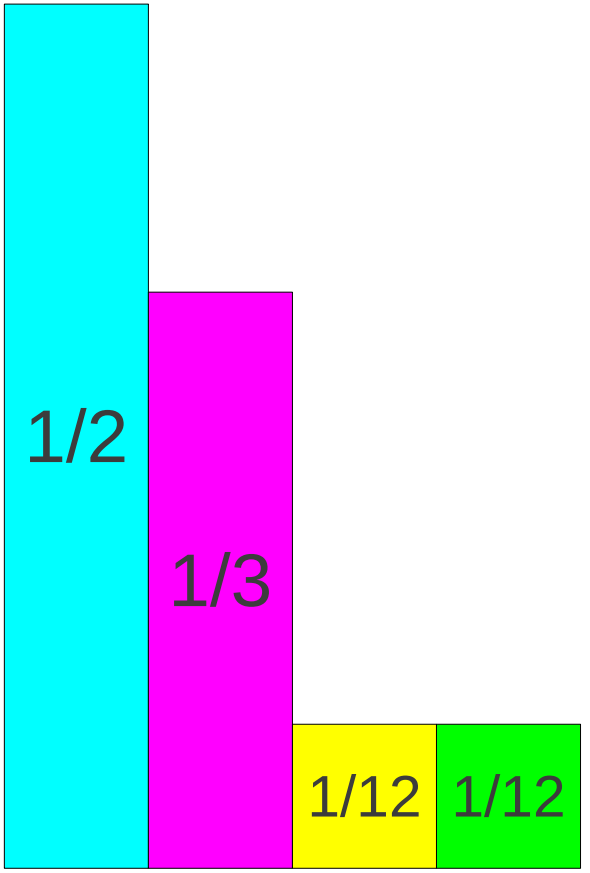


- Perfectly fair die: O(1) dart tosses required.
- Perfectly biased die: O(n) dart tosses required on expectation.
- General expected number of dart tosses: $O(n\ p_{max})$

Can we get rid of the empty space?

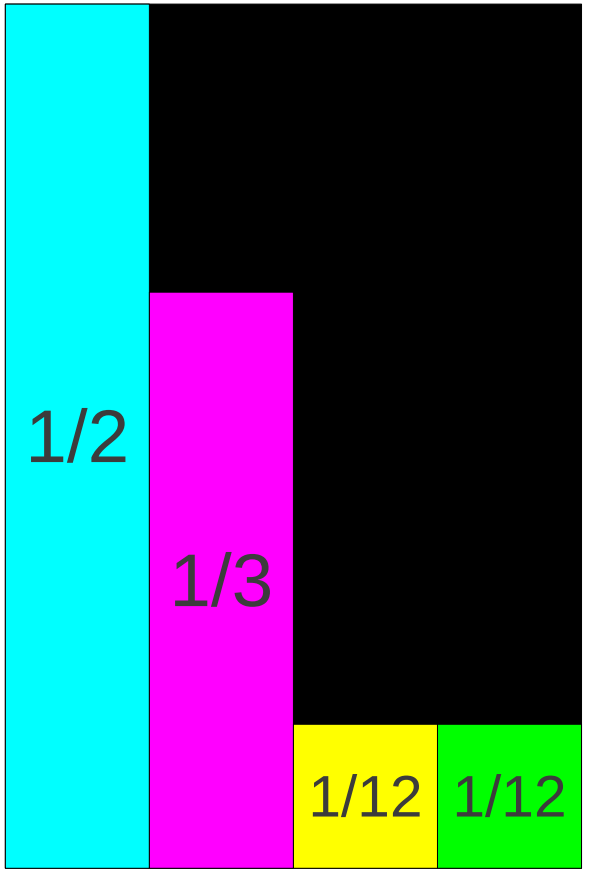
Getting Rid of Spaces

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



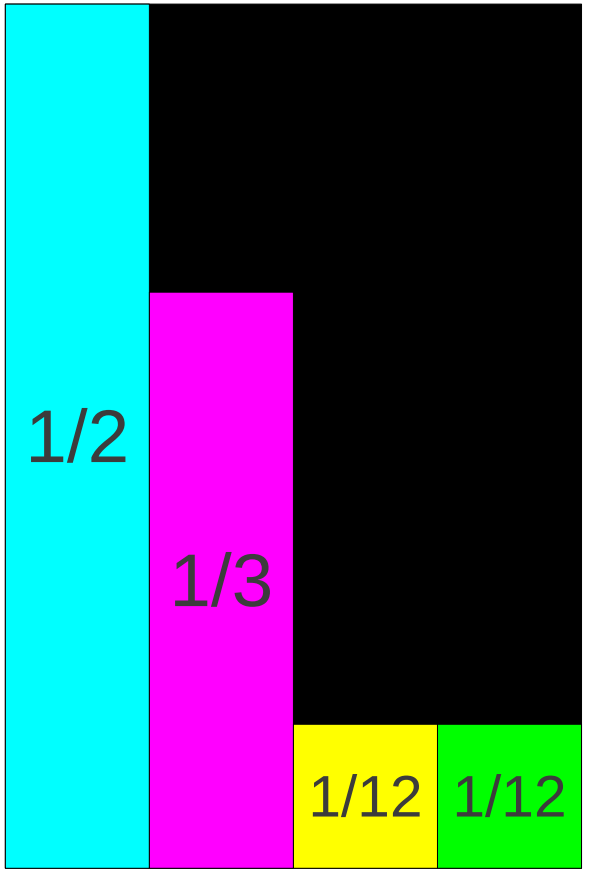
Getting Rid of Spaces

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



Getting Rid of Spaces

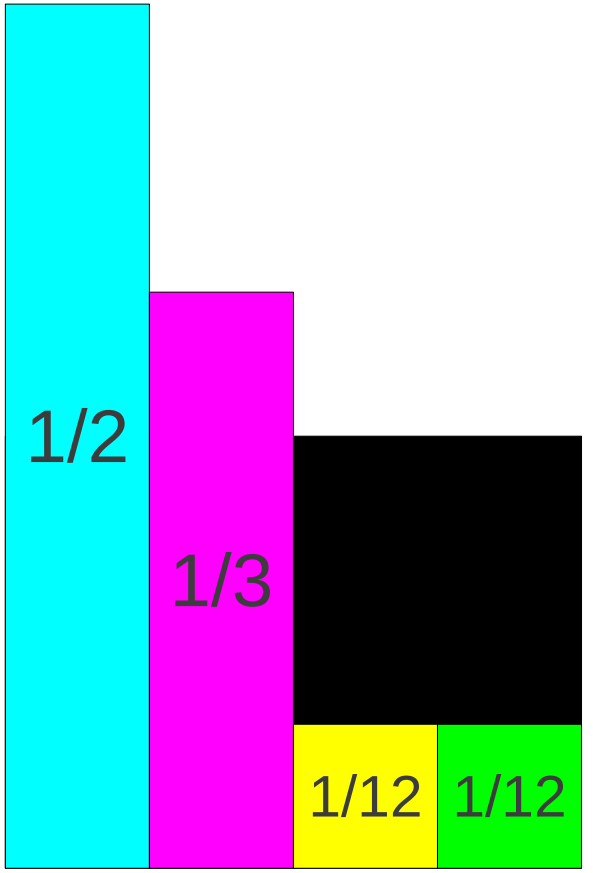
- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



Height is ½, the maximum of these values.

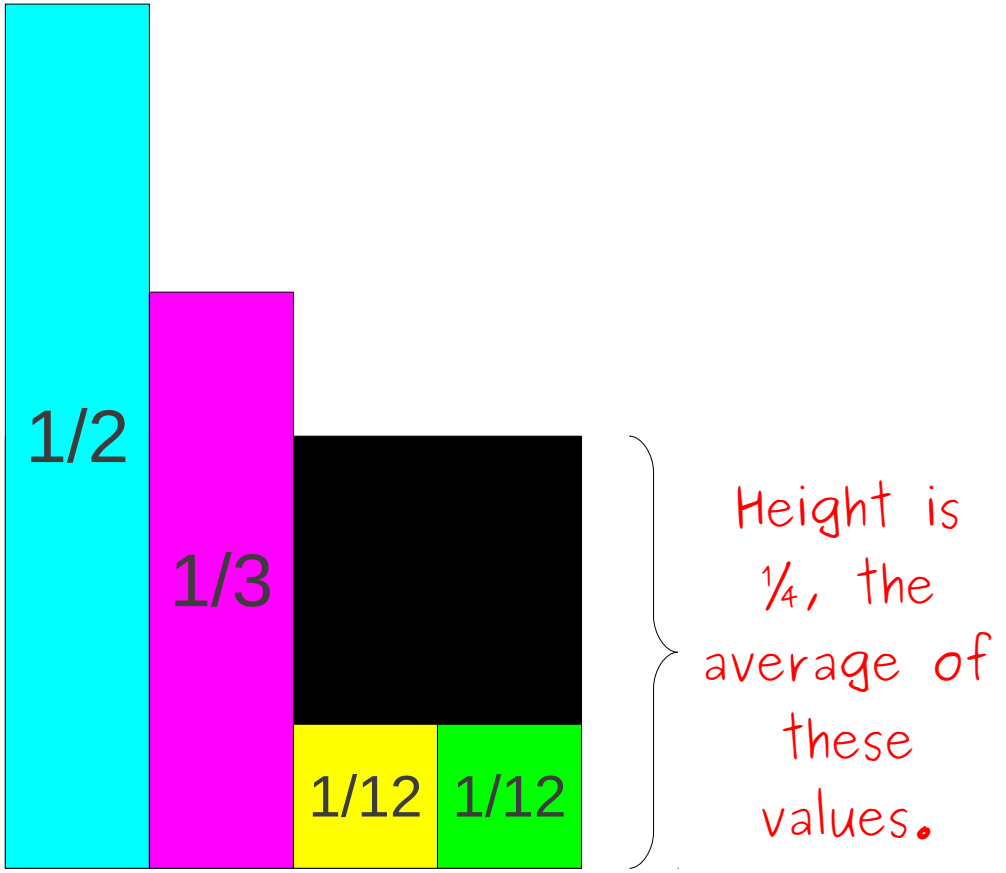
Getting Rid of Spaces

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



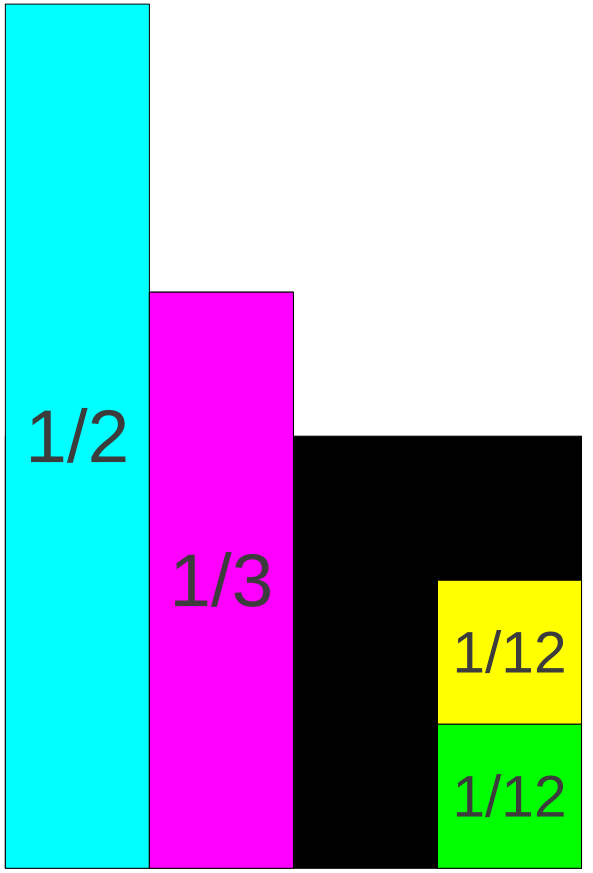
Getting Rid of Spaces

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



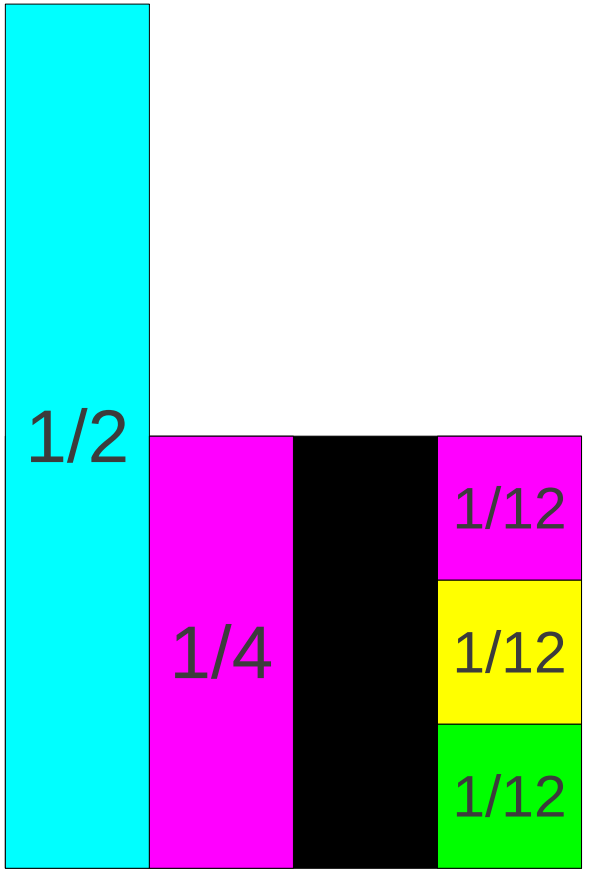
Getting Rid of Spaces

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



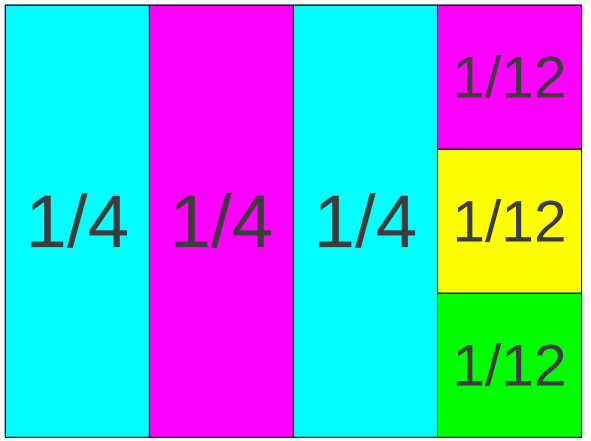
Getting Rid of Spaces

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



Getting Rid of Spaces

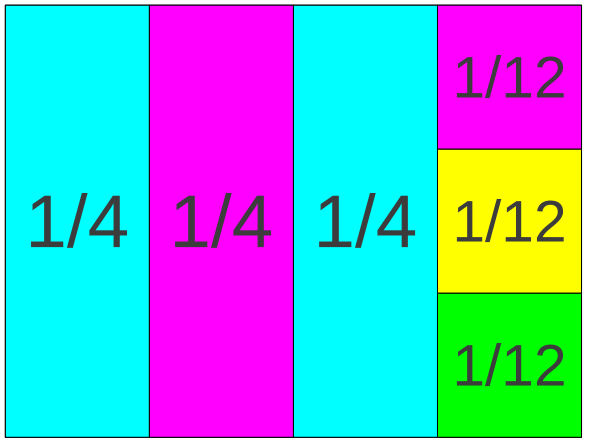
- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



Getting Rid of Spaces

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

We have gotten rid of the blank spaces. Every dart tossed will hit something!

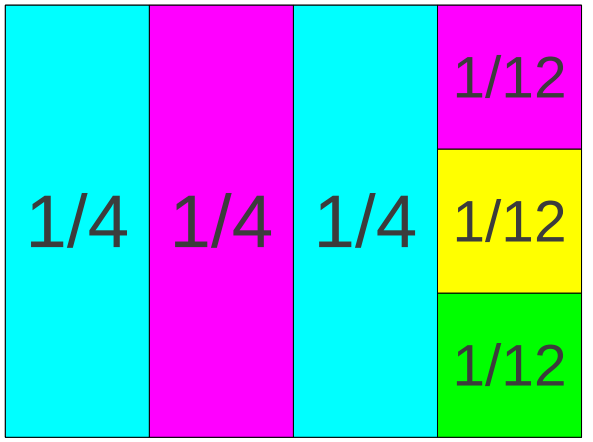


Getting Rid of Spaces

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

We have gotten rid of the blank spaces. Every dart tossed will hit something!

But now, each column isn't a biased coin.

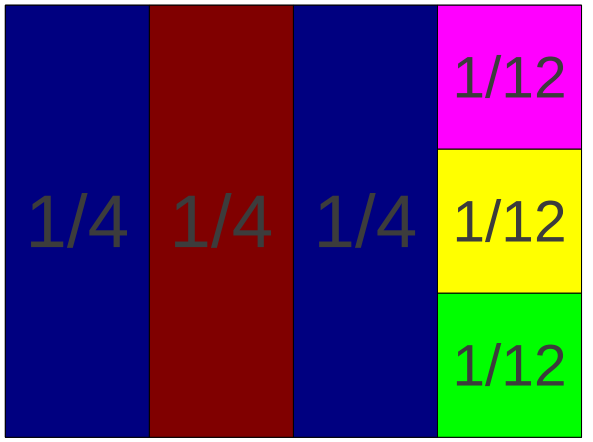


Getting Rid of Spaces

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

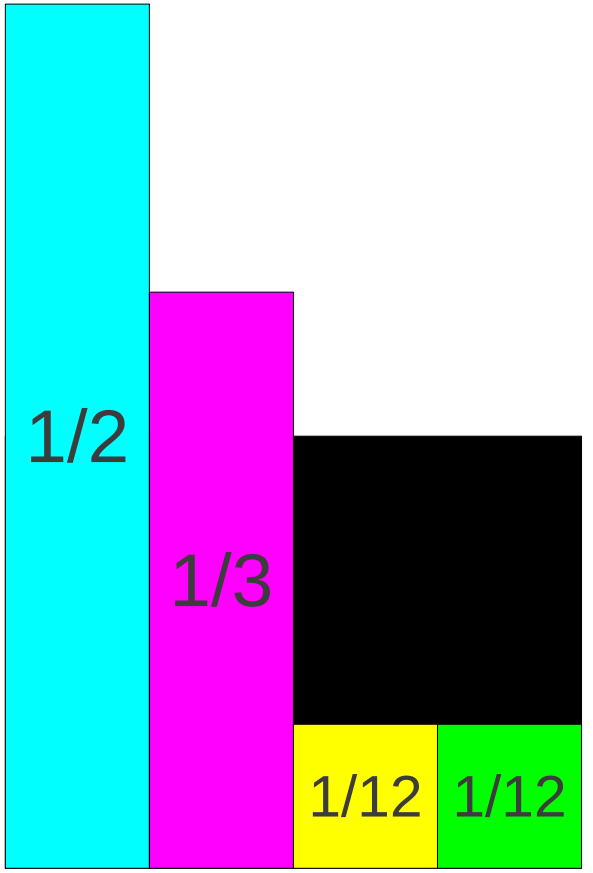
We have gotten rid of the blank spaces. Every dart tossed will hit something!

But now, each column isn't a biased coin.



Getting Rid of Spaces

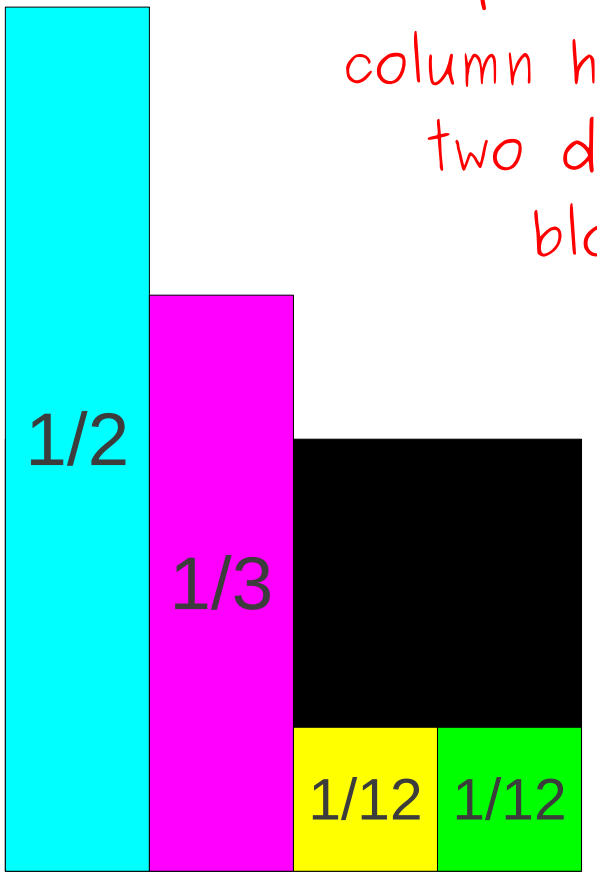
- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$



Getting Rid of Spaces

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

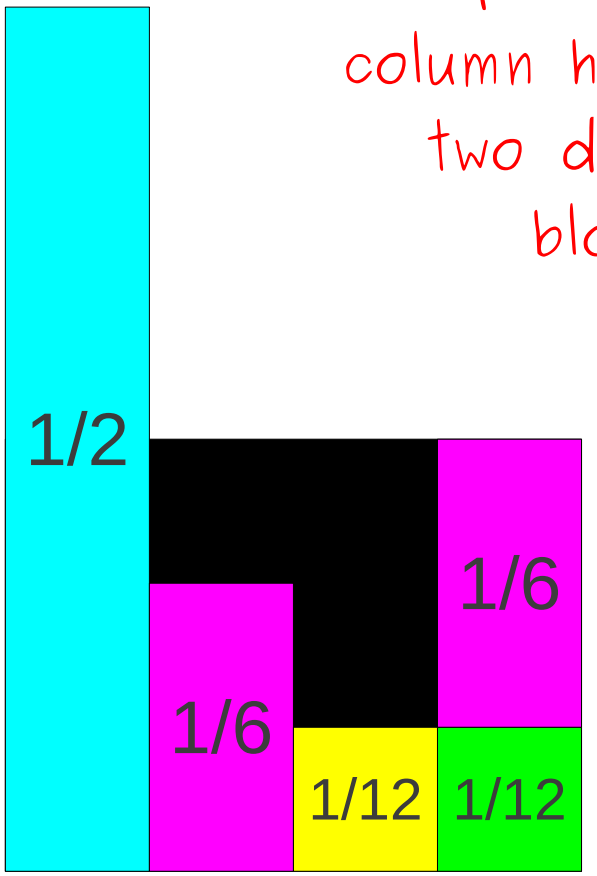
Goal: Reshape this setup so that each column has at most two different blocks.



Getting Rid of Spaces

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

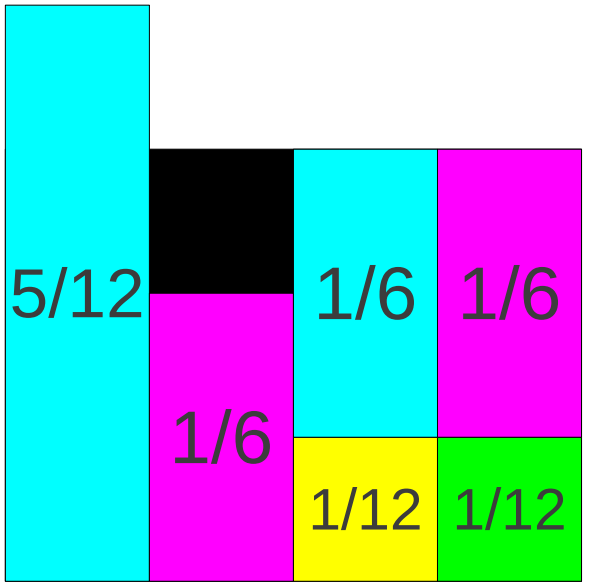
Goal: Reshape this setup so that each column has at most two different blocks.



Getting Rid of Spaces

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

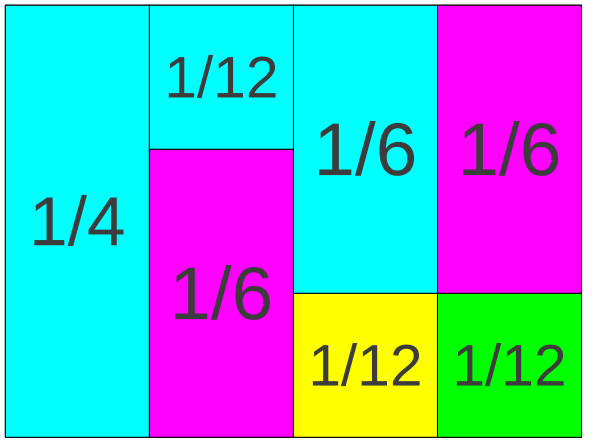
Goal: Reshape this setup so that each column has at most two different blocks.

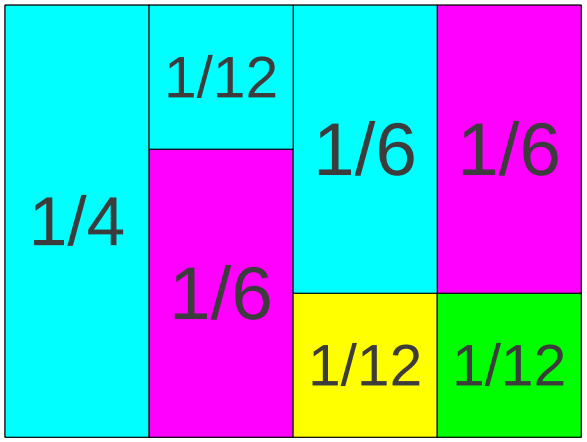


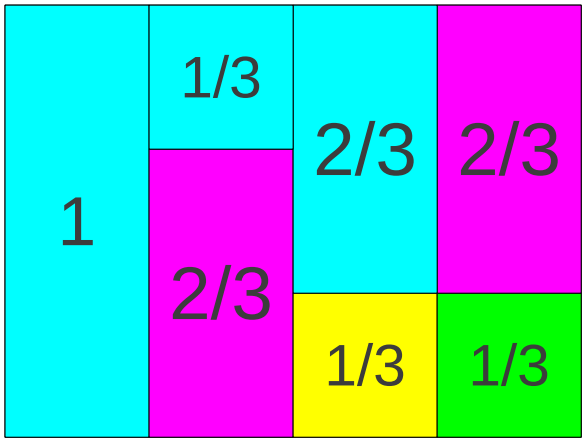
Getting Rid of Spaces

- A four-sided die:
  - $p_1 = 1/2$
  - $p_2 = 1/3$
  - $p_3 = 1/12$
  - $p_4 = 1/12$

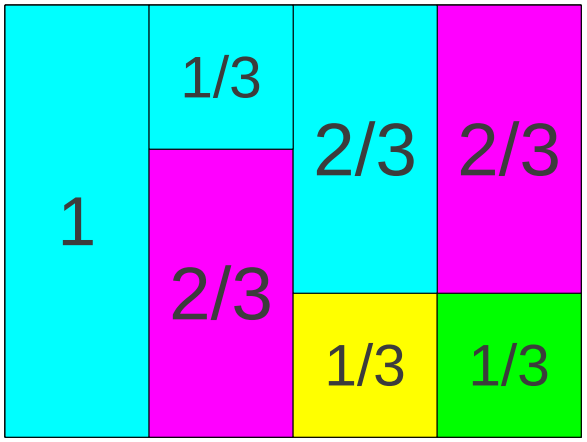
Goal: Reshape this setup so that each column has at most two different blocks.

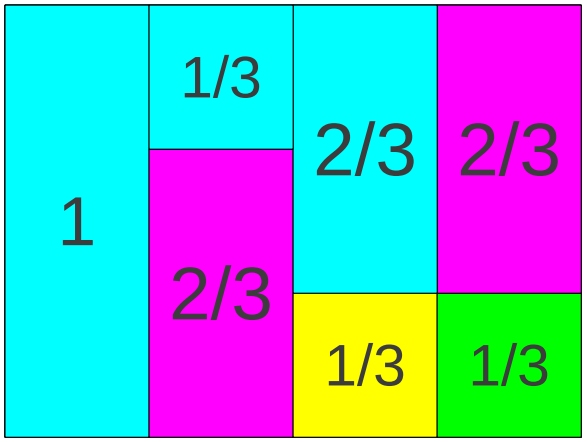



1/4
1/12
1/6
1/6
1/6
1/12
1/12


1
1/3
2/3
2/3
2/3
1/3
1/3

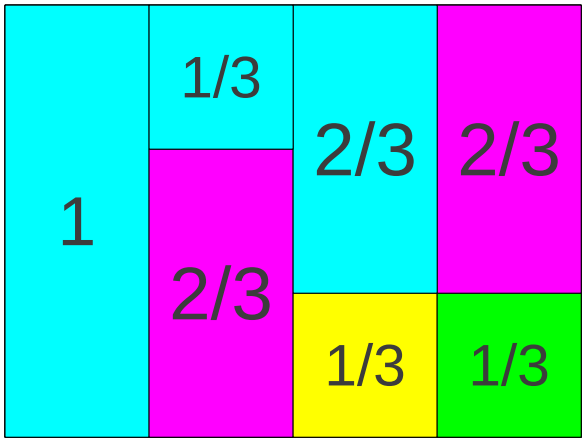
We can choose the dart's x coordinate by rolling a fair die.





We can choose the dart's x coordinate by rolling a fair die.

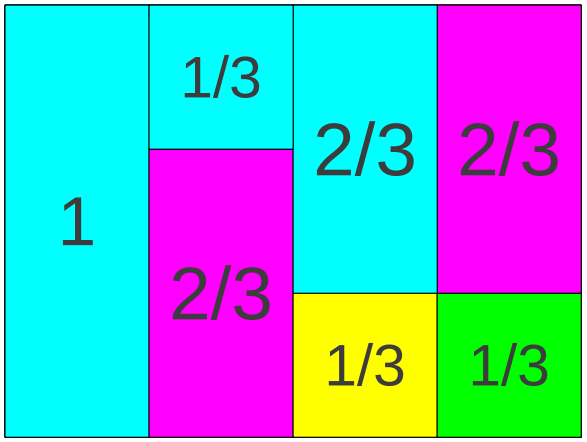
We can choose the dart's y coordinate by flipping a biased coin.



We can choose the dart's x coordinate by rolling a fair die.

We can choose the dart's y coordinate by flipping a biased coin.

If the coin yields heads, output the lower half as the answer.



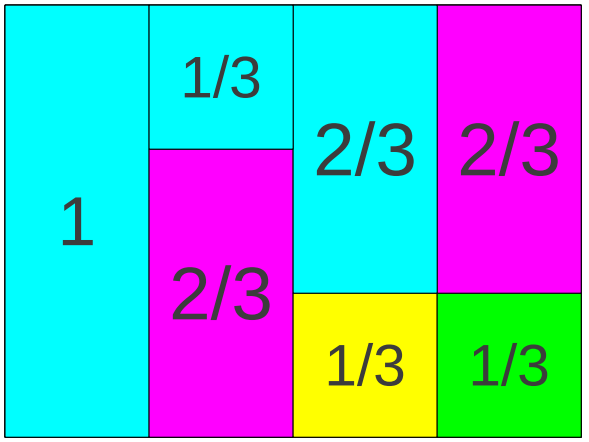
We can choose the dart's x coordinate by rolling a fair die.

We can choose the dart's y coordinate by flipping a biased coin.

If the coin yields heads, output the lower half as the answer.

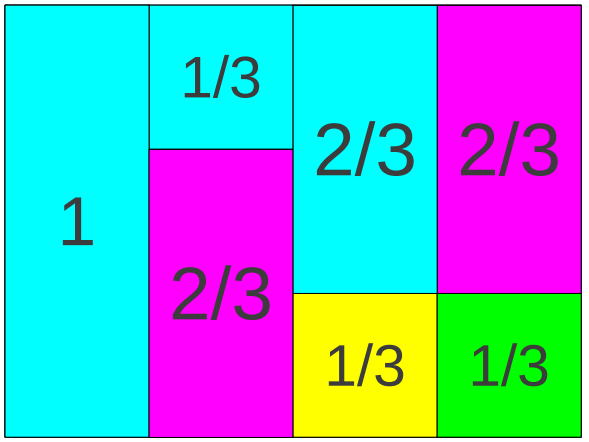
If the coin yields tails, output the upper half as the answer.

The Alias Method



- Distribute probabilities such that
  - Each column has height 1 / n
  - Each column has at most two blocks in it.
- To generate a roll of the die:
  - Roll a fair die to choose the column.
  - Flip a biased coin to choose the upper or lower block.
- **Generation time is now O(1)**.

The Alias Method



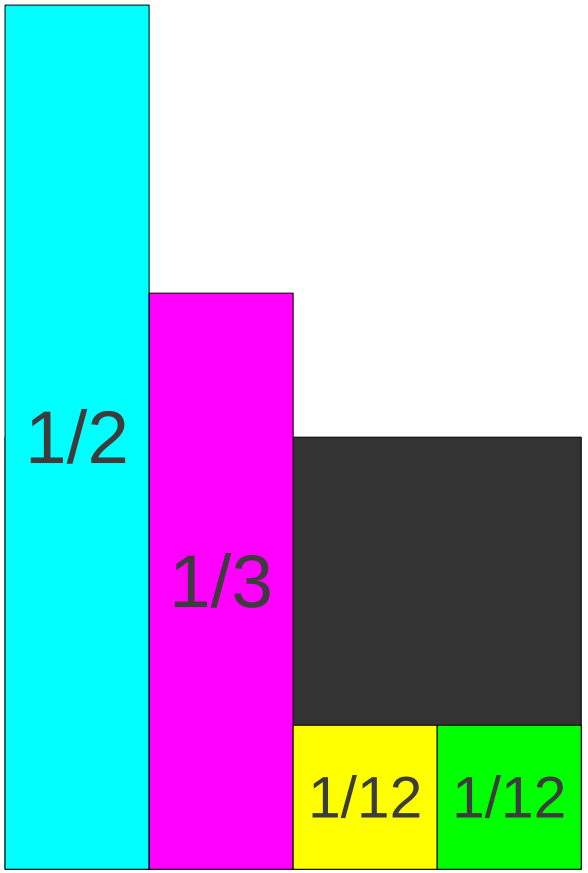
- For each column, have to store
  - The probability that of the coin for that column.
  - What true and false evaluate to.
- One column per side of the die.
- **Total space: Θ(n)**

Building the Dartboard

Building the Dartboard

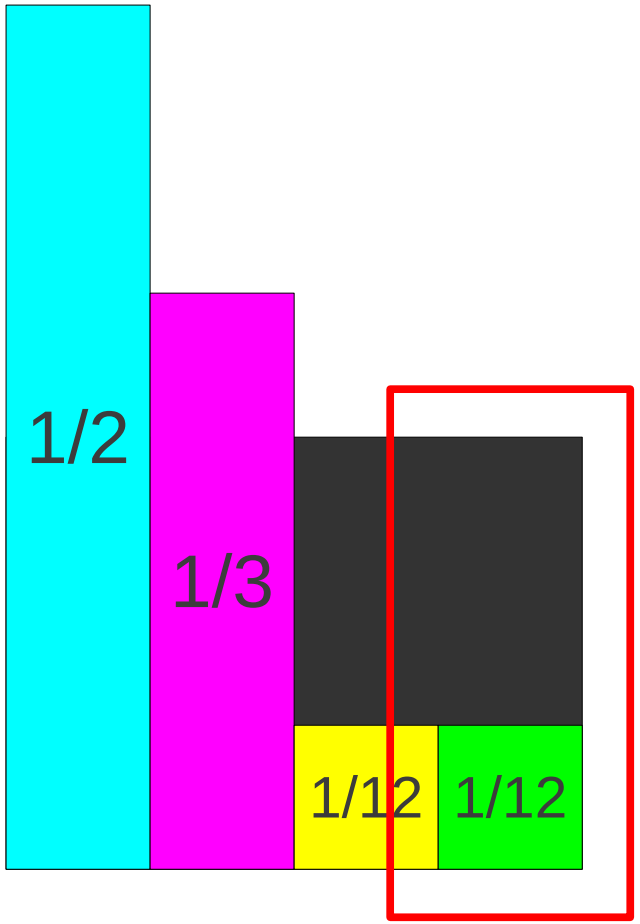
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



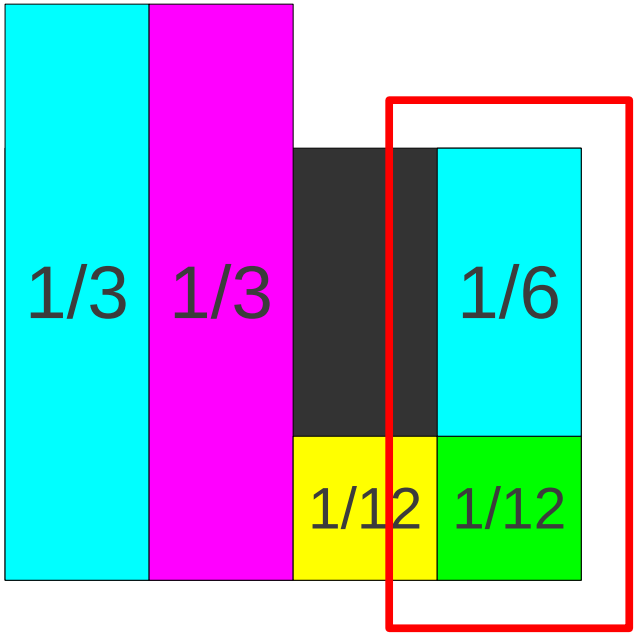
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



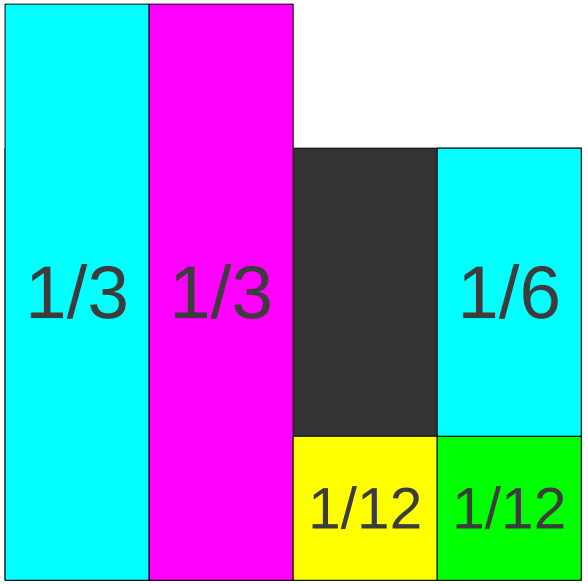
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



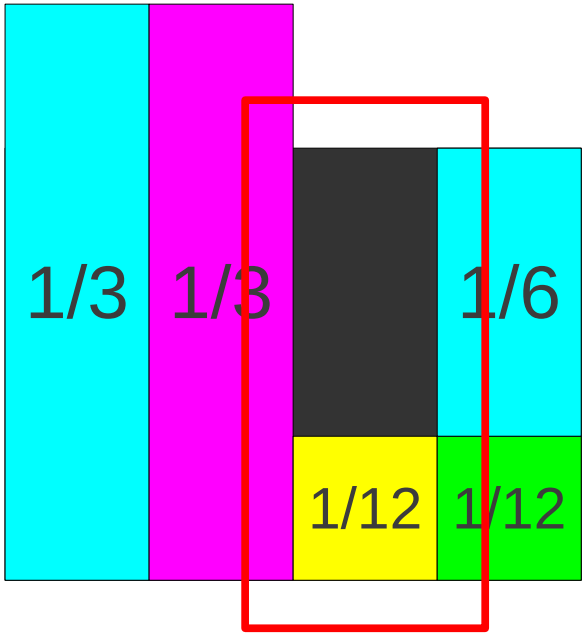
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



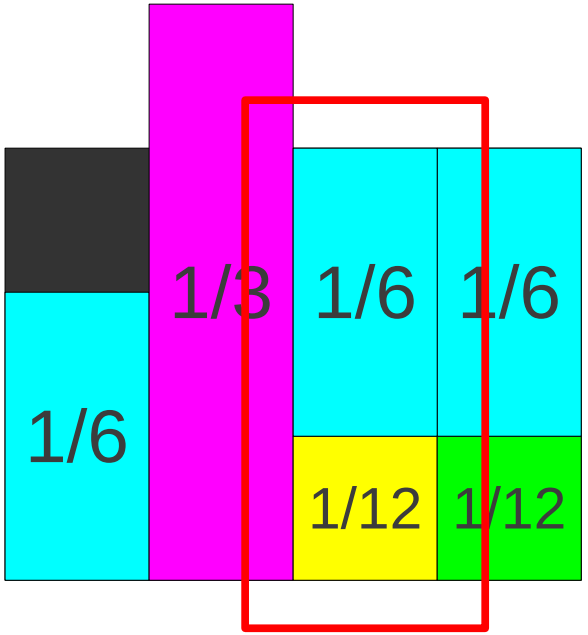
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



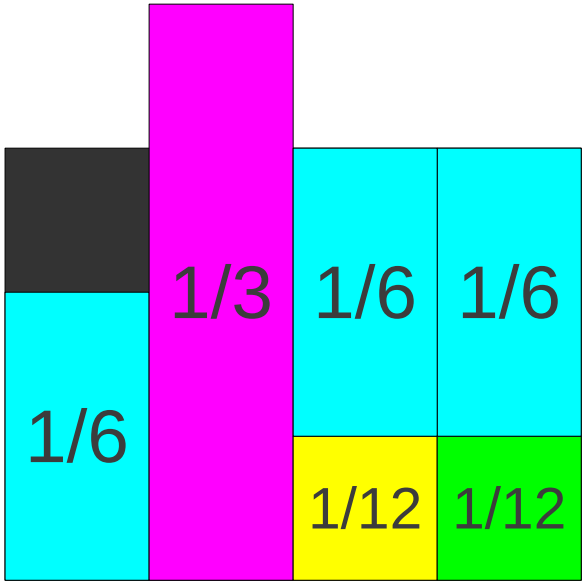
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



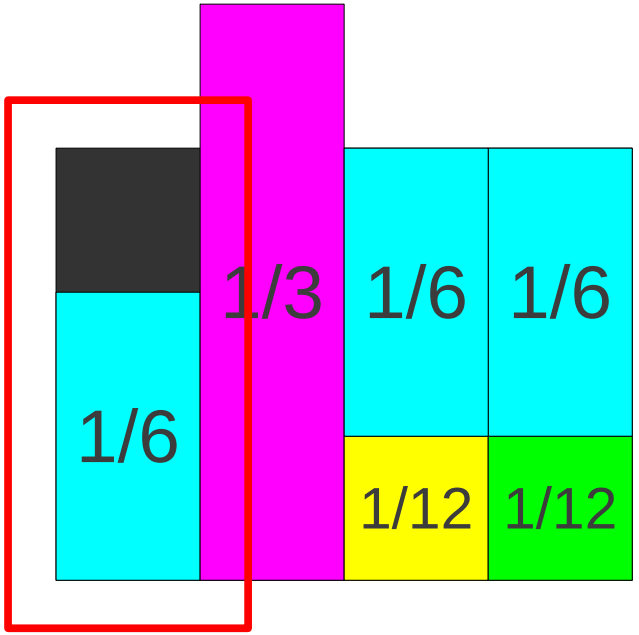
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



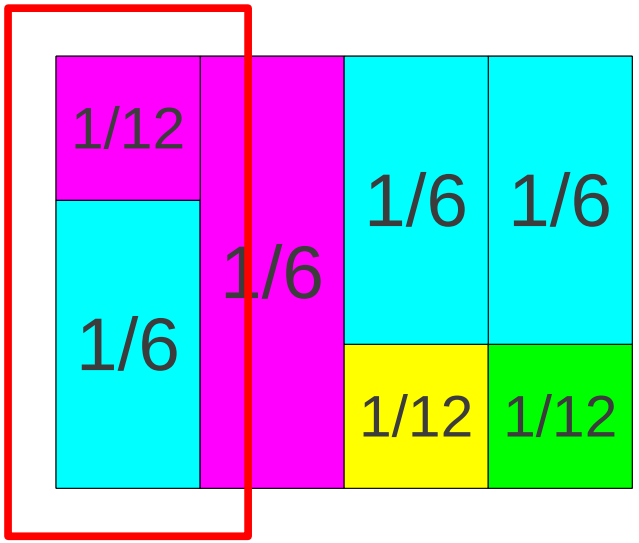
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



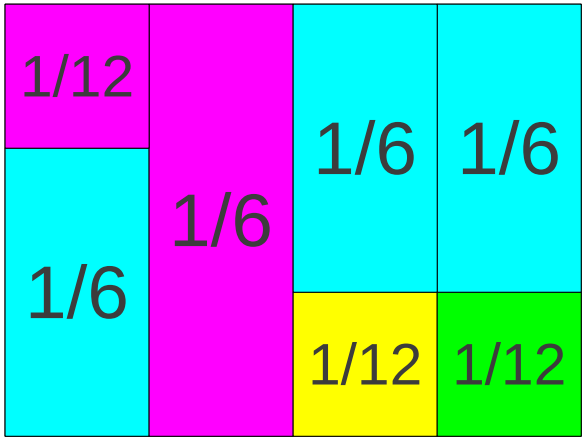
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



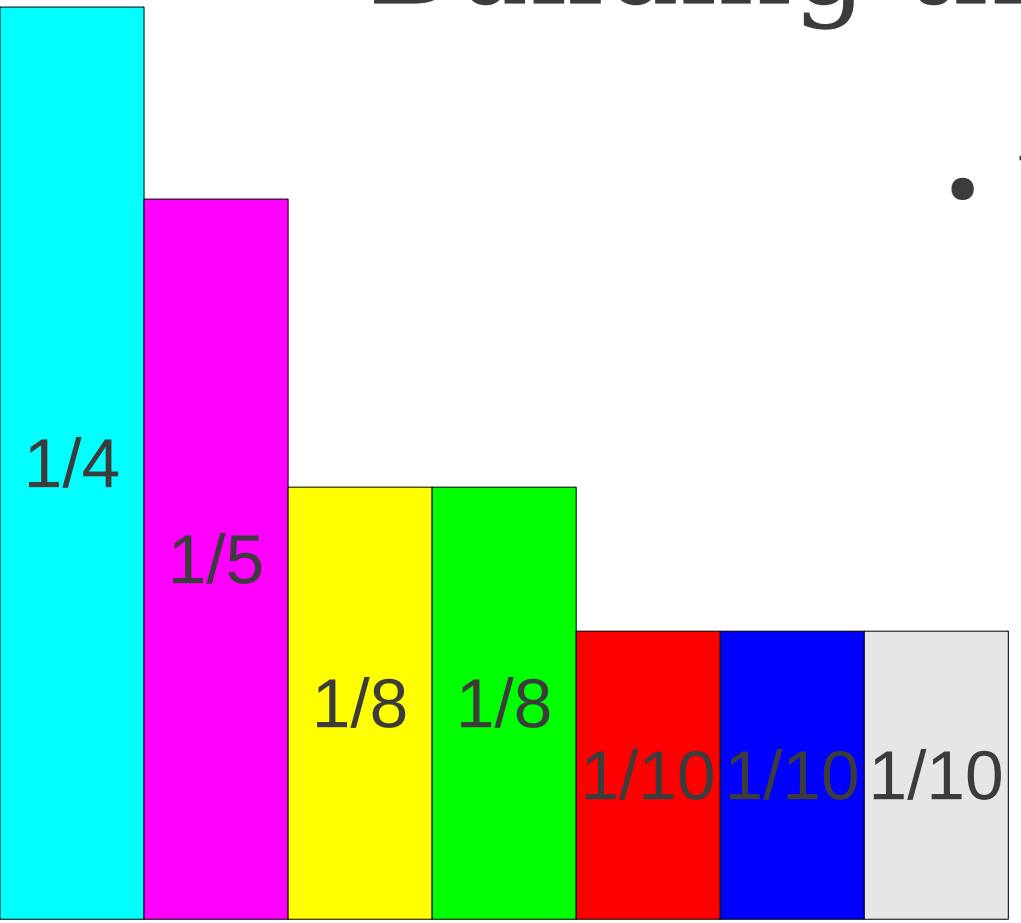
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



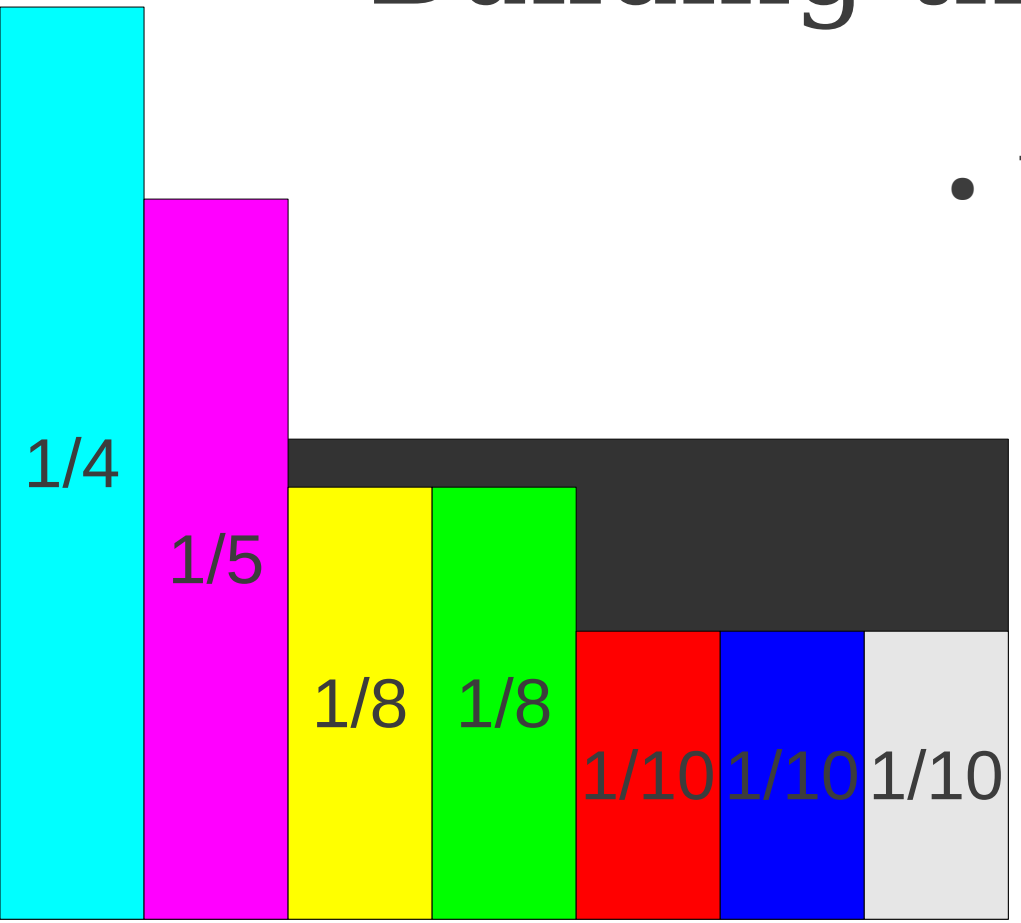
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



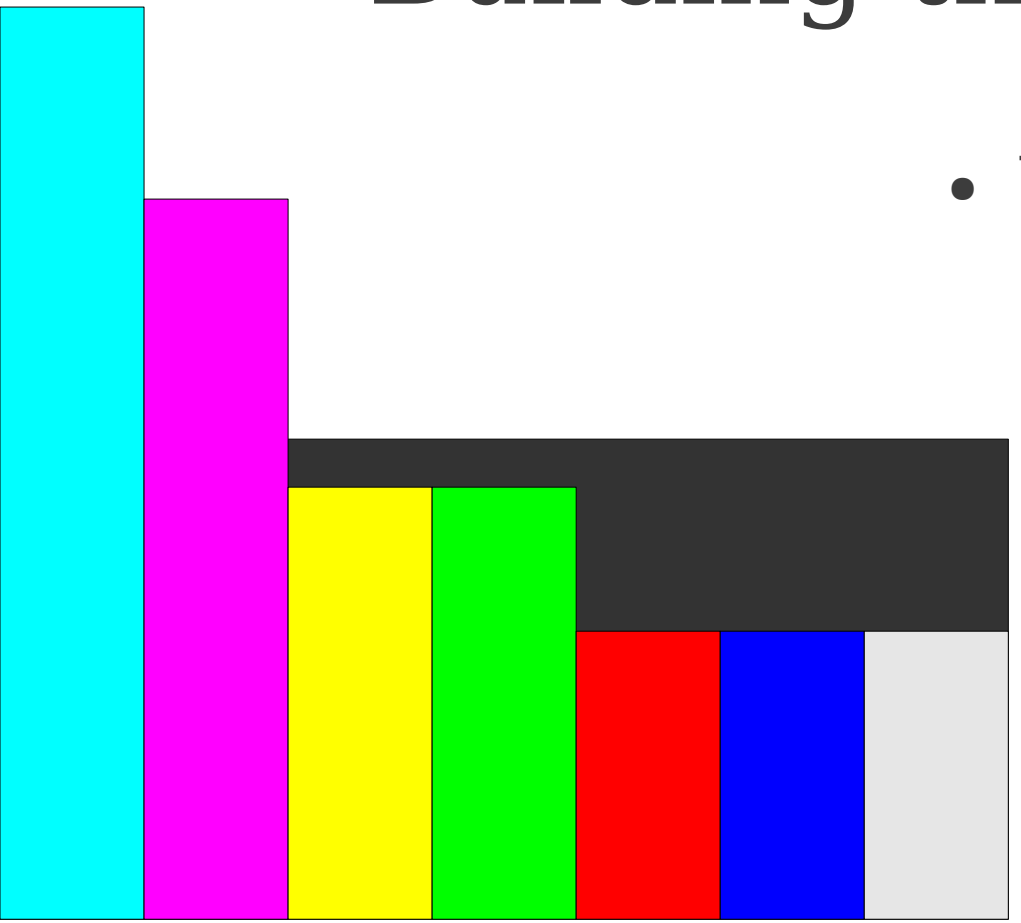
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



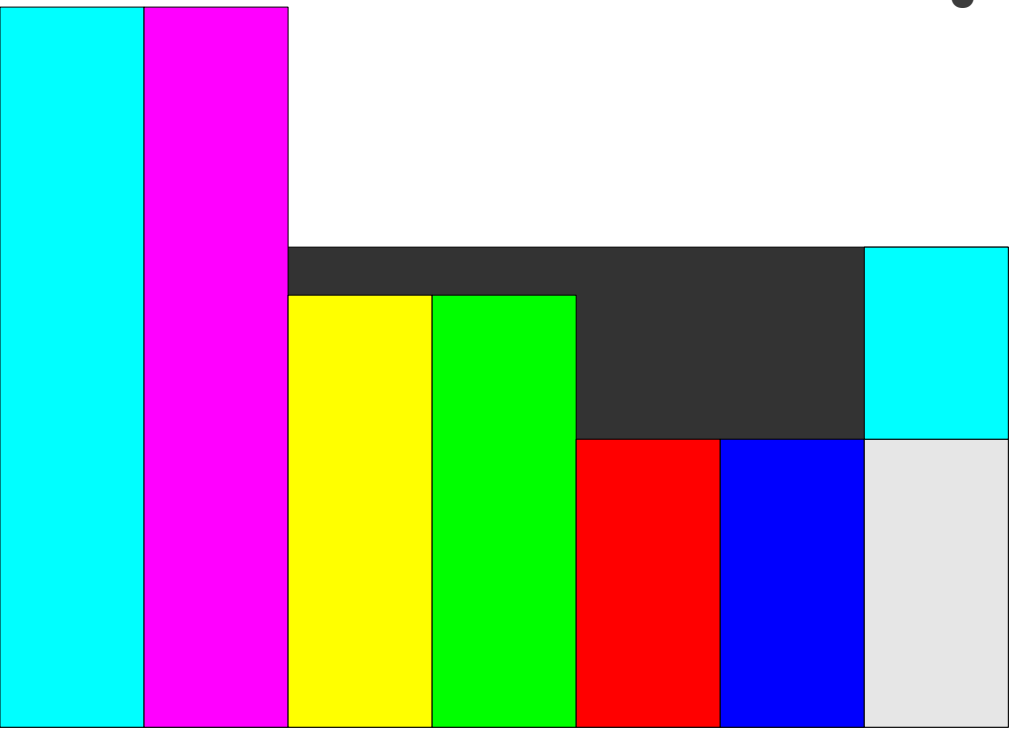
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



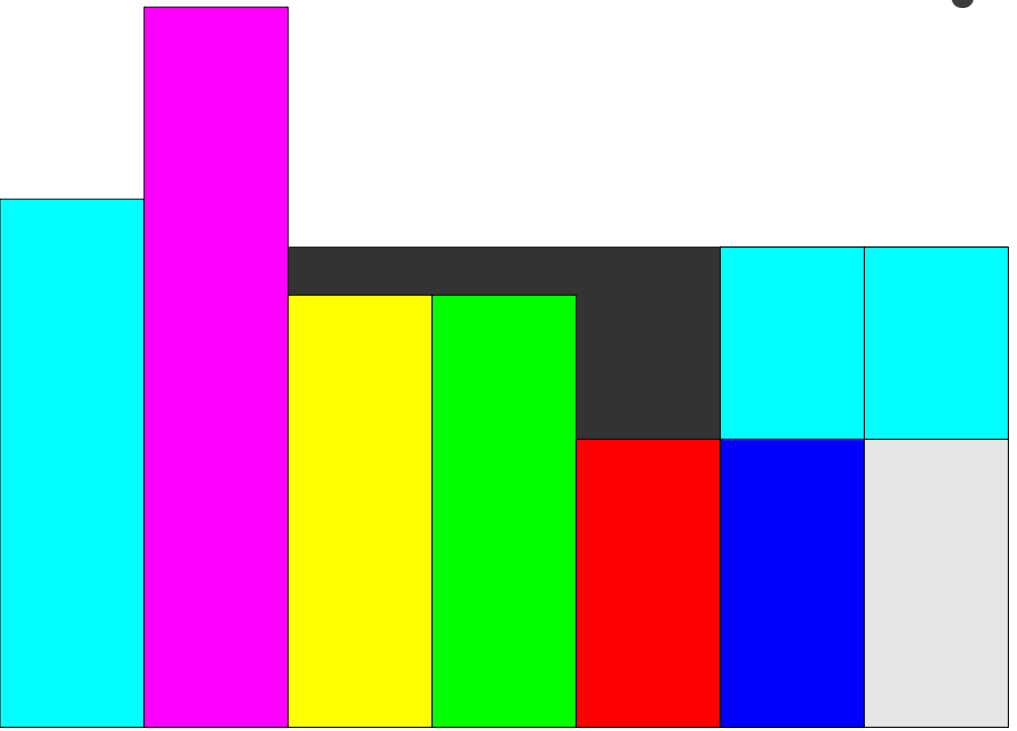
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



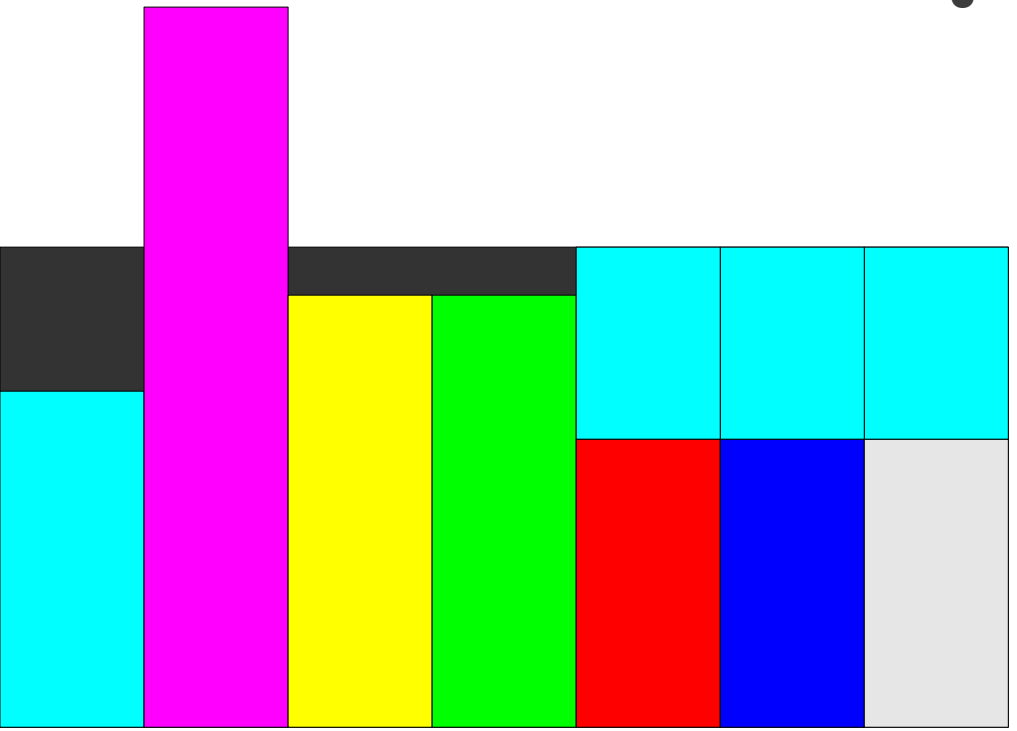
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



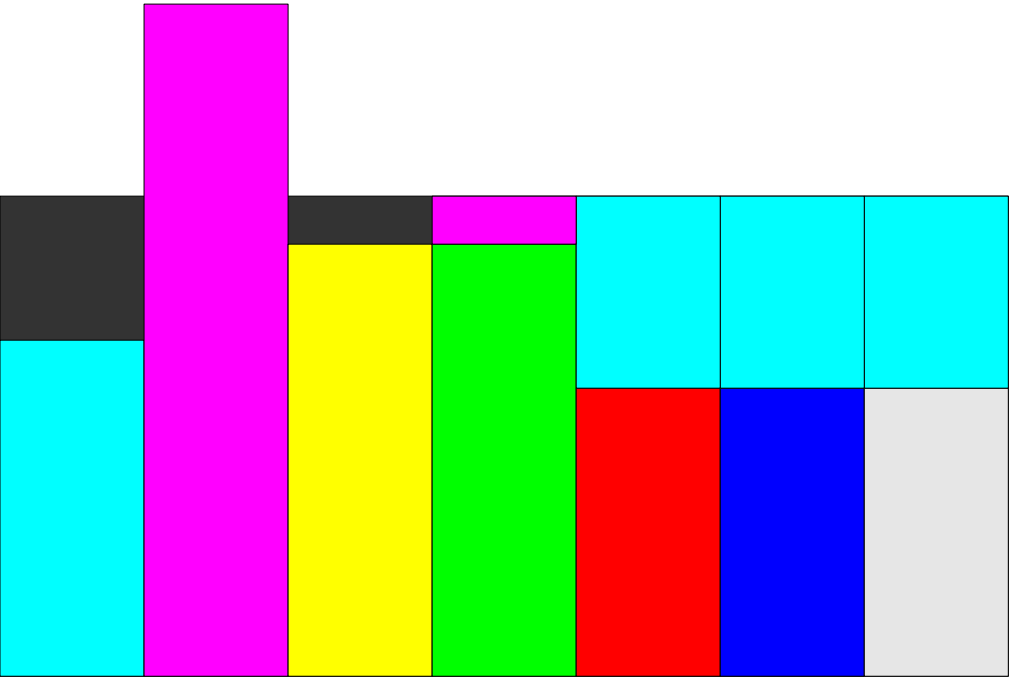
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



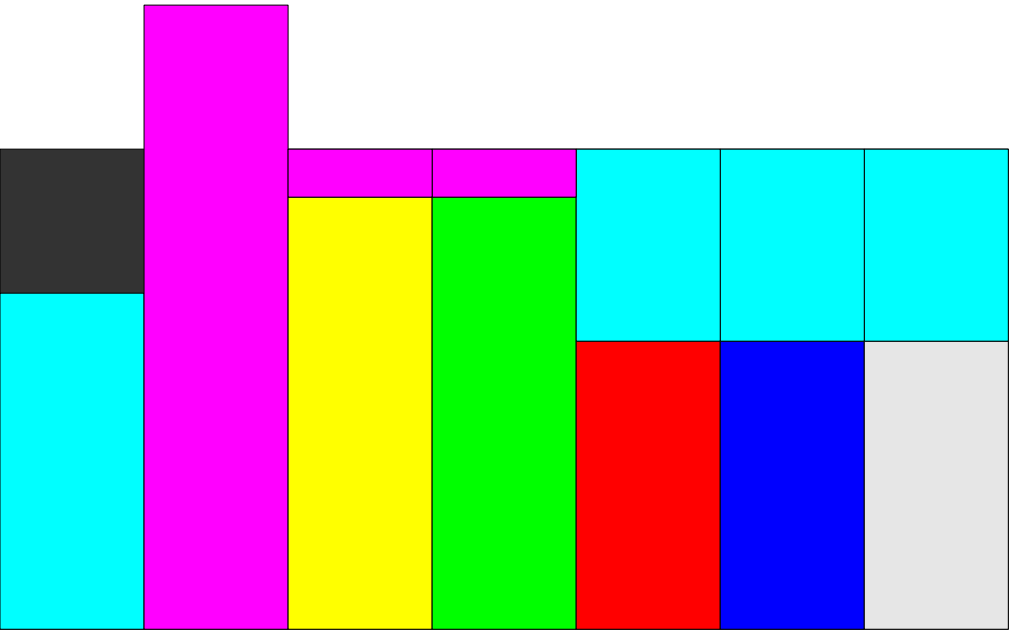
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



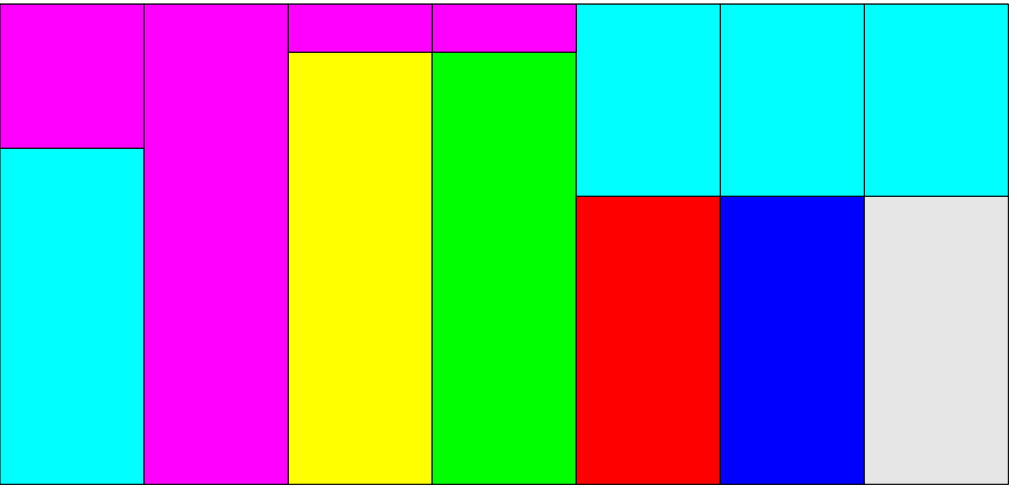
- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Building the Dartboard



- Until no holes remain:
  - Find a column that has a hole.
  - Find a column that has some excess.
  - Move enough from the column with an excess into the column with a gap to fill it in.

Correctness Proof Sketch

- If not all columns are balanced:
  - Something must be too small.
  - Since the total probability just covers the dartboard, something must be too big as well.
  - Always possible to move probability from one column to another.
- Process eventually terminates:
  - Each iteration takes a column that wasn't balanced and balances it.
  - Eventually all columns become balanced.

Summary

- We can efficiently roll a fair die.
  - Gives an inefficient algorithm for loaded dice.
- We can efficiently flip a biased coin.
  - Generalizes to an efficient algorithm for loaded dice.
- We can reduce rolling a loaded die to rolling a fair die and flipping a biased coin.
  - Generalizes to an **extremely** efficient algorithm for loaded dice.

For More on These Algorithms

http://www.keithschwarz.com/darts-dice-coins/

Questions?