Knuth-Yao and Rolling Dice

How to roll an n-sided die

Hera Brown

The Oxford Compsoc

January 31, 2026

The problem

- I'm playing a game of D&D, but I've forgotten my dice at home! All I have is a single coin.

Figure 1: It's all I have to my name.

- Unfortunately, I need to roll a d6 on my next move. Am I stuck?
- No! Using the power of computer science, I don't have to borrow a friend's die. I can get by just using my coin.

An easy solution

- One solution is to flip my coin, and treat each of the flips as a bit of a three-bit number.
- If I get a heads on the i^{th} flip, then I set the i^{th} bit of my string to 1. Otherwise, if I get a tails on the i^{th} flip, then I set the i^{th} bit of my string to 0.
- At the end I have a binary number between 0 and 7. If I get a number 1–6, then that's the outcome of my die roll. Otherwise, I try the whole process again.

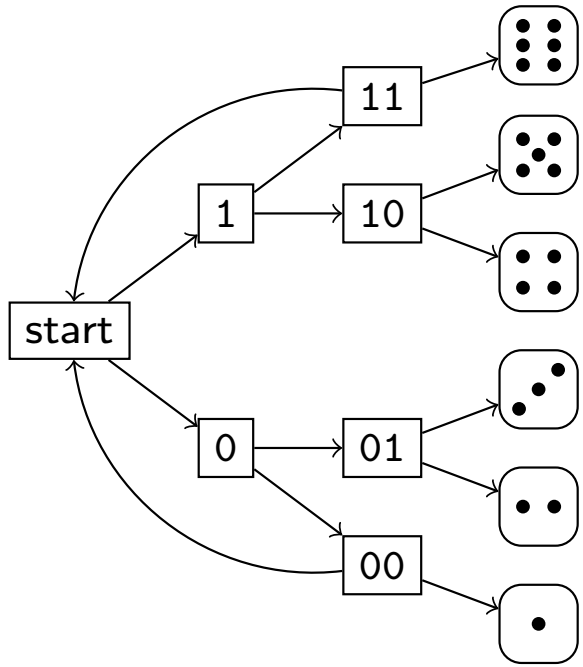

Figure 2: A diagram illustrating our algorithm.

A good solution

- We can do better. My turn's coming up soon, I don't have much time.
- If we look at the binary expansion of $1/6$, we'll see it's the following.

$$(1/6)_2 = 0.0010101010\ldots.$$

- Looks awfully regular. Looks like we see an outcome with probability $1/6$ just when we see it after three coin flips, or five coin flips, or seven coin flips, and so on. We turn this into a diagram:

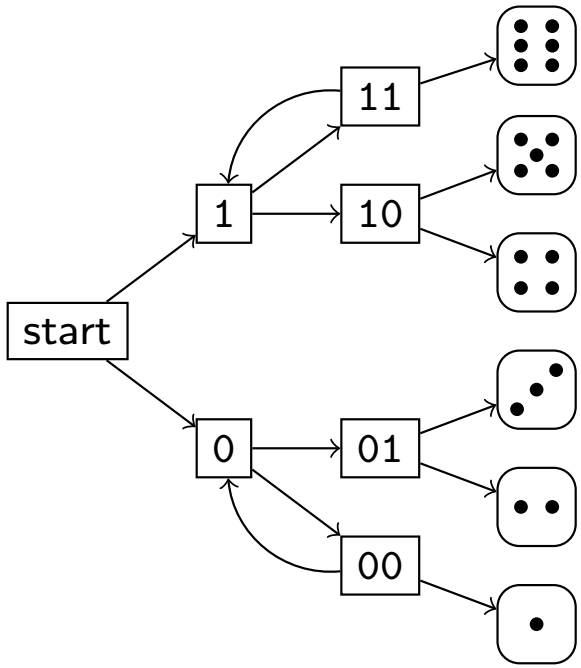

Figure 3: A better algorithm.

- And Knuth and Yao proved this algorithm is optimal.

Fiveses

- Oh no! I've played my turn, and rolled the d6 (I got a three if you were curious). But next turn I'm going to have to roll a d5! How am I going to get out of this one?
- Thankfully five has a binary expansion as well. It's the following:

$$(1/5)_2 = 0.00110011001100\ldots.$$

- So I want to see some outcome only after 3 coin flips, or 4, or 6, or 7, or so on. The probability of that happening is $1/5$, the probability of that *not* happening is $4/5$. So we build the following machine.

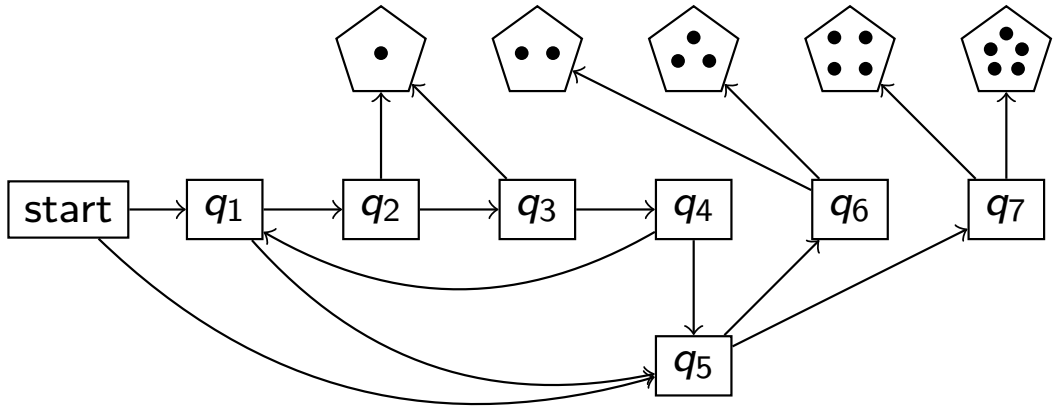

Figure 4: Our d5-machine.

Sevenses

- Great. I've rolled my d5 and got a two. Next turn I'm up for a d7 roll, but I'm confident I can work out how to emulate it with my coin.
- We look at the binary expansion of $1/7$:

$$(1/7)_2 = 0.001001001001001\ldots$$

- And we turn it into a die-rolling algorithm.

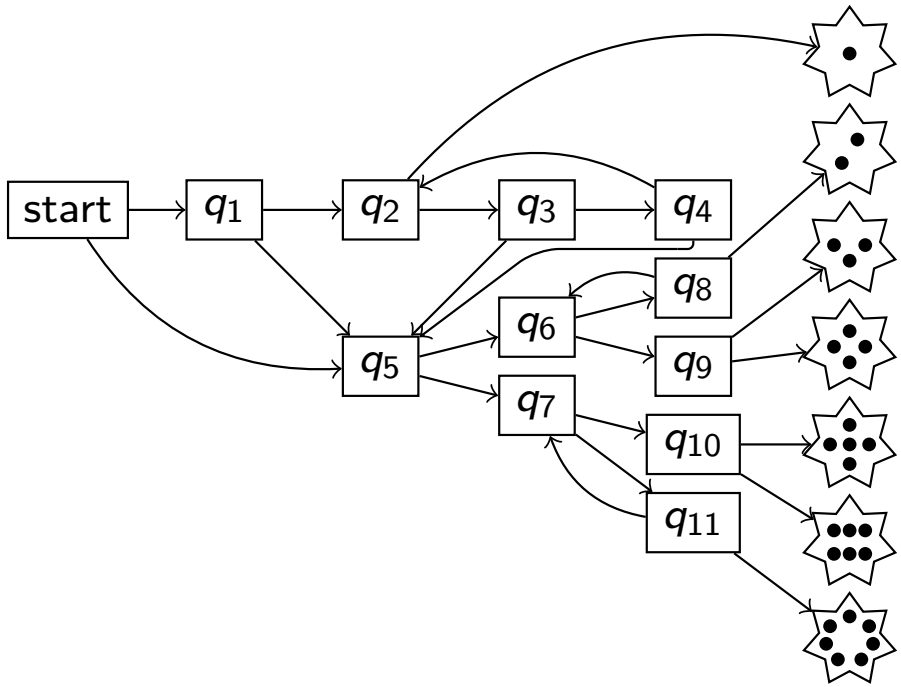

Figure 5: What I'm going to have to think about in a turn's time.

And we generalise

- Of course there's nothing really special about the numbers $5, 6$, and 7. We just used the fact that their binary expansions were regular.
- There's a reason I'm using the word "regular"; we can link this to automata theory. For those in the know, here's a definition.

Definition

An infinite binary string s is *automatic* if the language $\{s\}$ is ω-regular.

- I'm pretty sure that all rational fractions are automatic, so we can simulate *any* n-sided die in this way. Hooray.

So, the main takeaways:

- We can simulate dice using coin flips (and I can play D&D).
- We can do this really quickly using number theory & automata theory.
- You should take probabilistic model checking in 4^{th} year.

Thanks for listening!

Any questions?