Felix Dilke

Word Games in Scala

May 31, 2019 • London,UK

Info at github.com/fdilke/scala_exps

A simple word game

Players take it in turns to name a US state, subject to these rules:

You have to name a state that hasn’t been named so far, and that has at least 1 letter in common with the previously named state.

Example: If I start with UTAH, you can continue with MINNESOTA but not WYOMING.

Play continues until someone is left with no moves, whereupon they lose! And the other player wins.

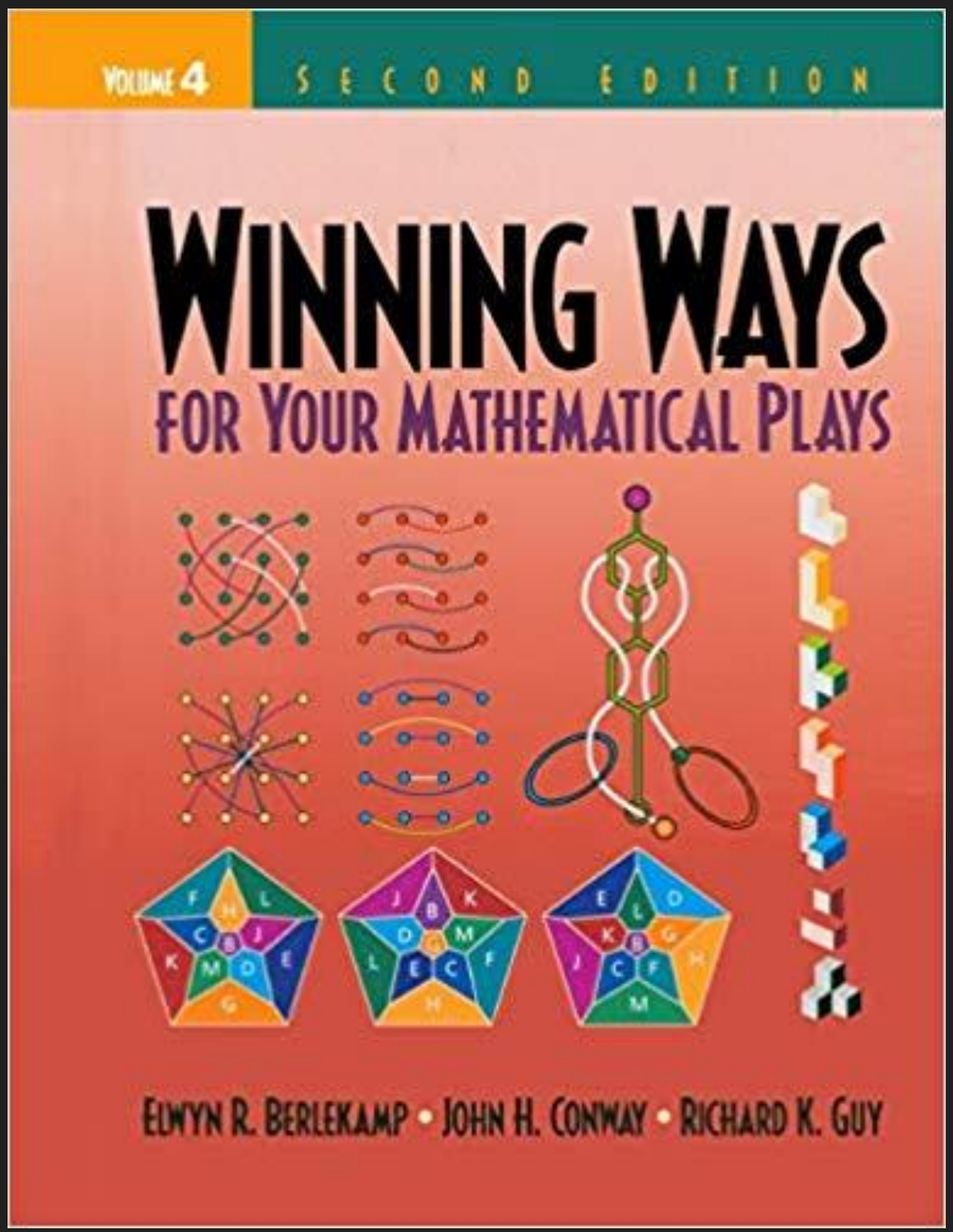
For some reason, this game has been rattling around in my head for a while so I decided to write an app to play it.

The “states” game is clearly:

- A two player game
- of perfect information
- which you can play on any graph

We can consider two nodes (e.g. states) “adjacent” if they have a letter in common.

Let’s wheel in the theory of combinatorial games



This book is great. It’s a quadrilogy about two-player games of perfect information.

Among other things, it tells how you can have half a move in a game.

The authors go on to define a whole number system just for measuring the values of various games.

We’re going to use only a tiny part of the theory in *Winning Ways*:

This game is “impartial”, which means the moves are the same for each player.

All that matters is whether the Previous or Next player is going to win.

So… we can speak of P-positions and N-positions.

If it’s your move and you are looking at a P-position, you’ve already lost! The trick is to leave your opponent stuck with one.

It’s obvious then, that:

- One player has a winning strategy
- We can work out which one, and also spell out the strategy explicitly, by just doing a depth-first search.
- Using the terminology of *Winning Ways*, we can classify a position as P or N depending on whether the PREVIOUS or NEXT player has won, respectively.

Here’s an excerpt of the gloriously recursive code that does this:

```
def winningMoves: Set[Int] =
 allowedMoves.filter { newWord =>
   move(newWord).isP
 }

def isP: Boolean =
 winningMoves.isEmpty

def isN: Boolean =
 !isP
```

Expectations were that:

The game would be N (rather than P), i.e. the first player would have a winning strategy.

What actually happened of course:

It takes too long to calculate

So now:

The project becomes an opportunity to learn about optimization of graph algorithms. How to speed things up?

Some helpful observations

- Calculating whether two words have any letters in common (without tripping up over things like the space in “New Hampshire”) is absurdly awkward / slow, so you should probably cache it

```
def letters(string: String) =
 string.chars().iterator().asScala.toSet

private def unequalAndWithACommonLetter(str1: String,
str2: String): Boolean =
 (str1 != str2) &&
   letters(str1).intersect(
     letters(str2)
   ).exists {
     x => !x.toChar.isWhitespace
   }

def adjacent(str1: String, str2: String): Boolean =
 unequalAndWithACommonLetter(
   str1.toLowerCase(),
   str2.toLowerCase()
 )
```

Some helpful observations (2)

- Almost all US state names contain an “A”
- E.g. Florida, Alabama, Texas, Washington, etc etc
- This means most pairs of states are adjacent
- If you’re in a game where all the remaining possible states are adjacent to each other (a “complete graph”), you can compute the winner just by counting parity
- I.e. whether there are oddly many or evenly many states left

This points to a promising possible optimization

BUT: it should probably be combined with...

Some helpful observations (3)

- At any point in the game, you can further narrow down the possible moves by eliminating states that are no longer reachable by any path from the last named one
- This can cut down the size of the search space quite dramatically

So: a three point plan then!

- Index the nodes (states) and cache the adjacency matrix
- Eliminate unreachable nodes
- Detect complete graphs and compute parity

I haven’t quite done all this yet, but no doubt, it will help speed things up.

What is mathematics? It is only a systematic effort of solving puzzles posed by nature.

Shakuntala Devi

THANK YOU