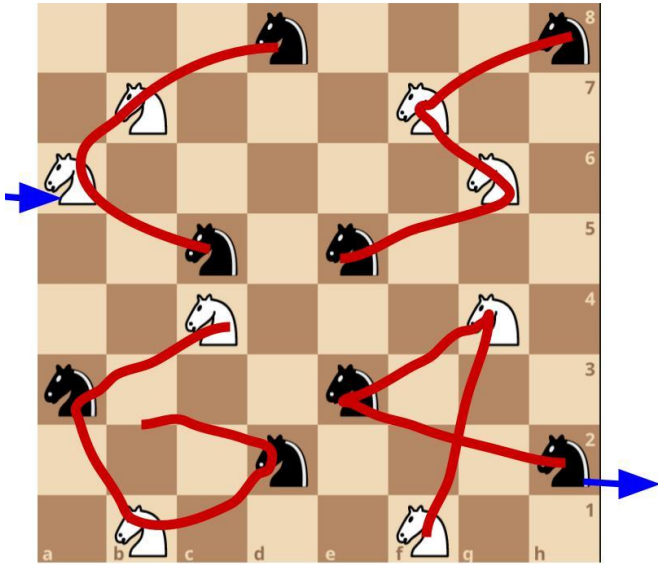
CS64: Computation for Puzzles and Games



Autumn 2022

Ian Tullis

Lecture 1: Word Games / Puzzles

- Super quick class overview
- How to win (and lose) at Scrabble
- If time: Tries and Ghost-busting

A fun 1-unit break from Stanford intensity

- Each week: lecture on Wednesday, optional puzzle/problem session on Friday

- S/NC grading. To get an S: either attend 6 of 9 Wednesday lectures, or do a small informal project on something that interests you. Or both if you want!
  - Attendance will be self-reported, later in the quarter – keep track of which you came to, and don't stress about it!

- Ed forum for questions / discussions / sharing fun stuff

- Will attempt recordings and post on Canvas

- https://web.stanford.edu/class/cs64 has more complete details

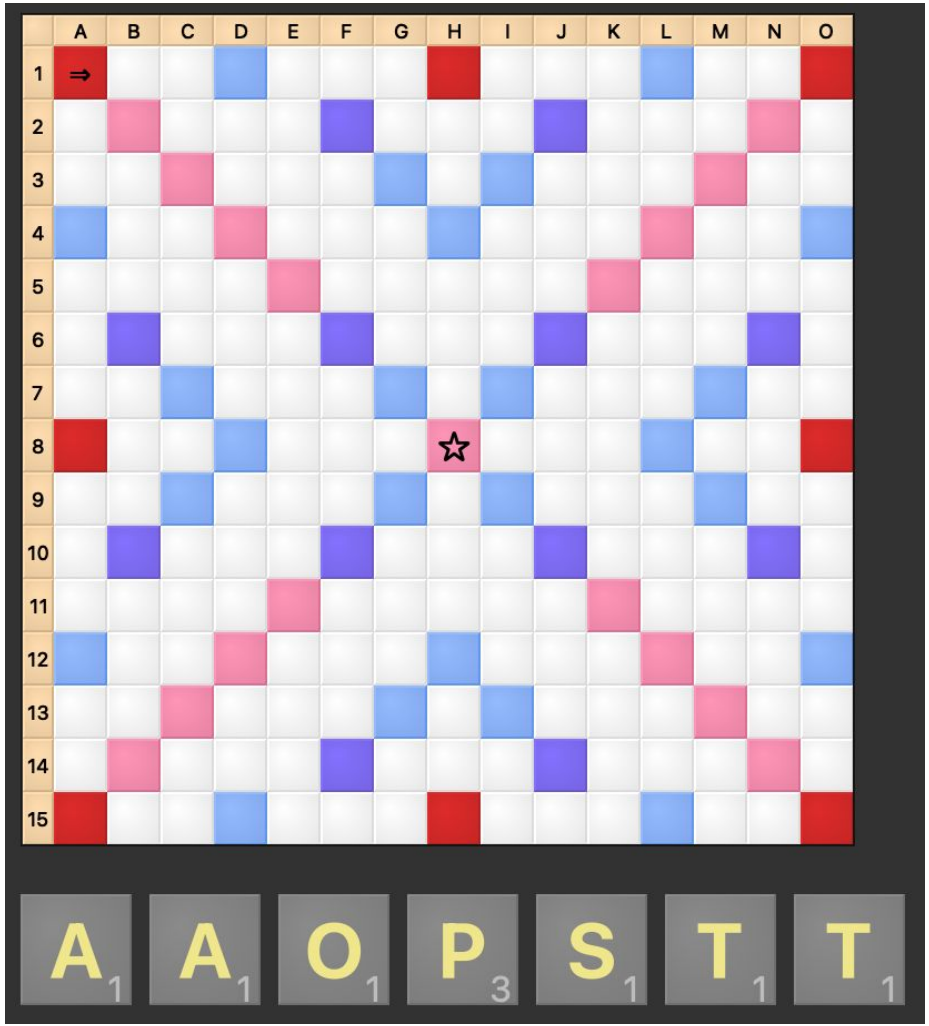
Course goals

- Explore how we can apply CS theory and AI to games and puzzles to make them even more fun
- Introduce some useful algorithms / ideas that are often not covered in our standard CS classes
- Enjoy solving puzzles / playing games
- Get ourselves (even more) excited about CS theory and AI

Scrabble

Alfred Mosher Butts, inventor of Scrabble. Two of his three names are legal words!

one of the only still-playable classic American board games


A B C D E F G H I J K L M N O
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
A1 A1 O1 P3 S1 T1 T1


A B C D E F G H I J K L M N O
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
A1 T1 O1 P3 ⇒
A1 A1 O1 P3 S1 T1 T1

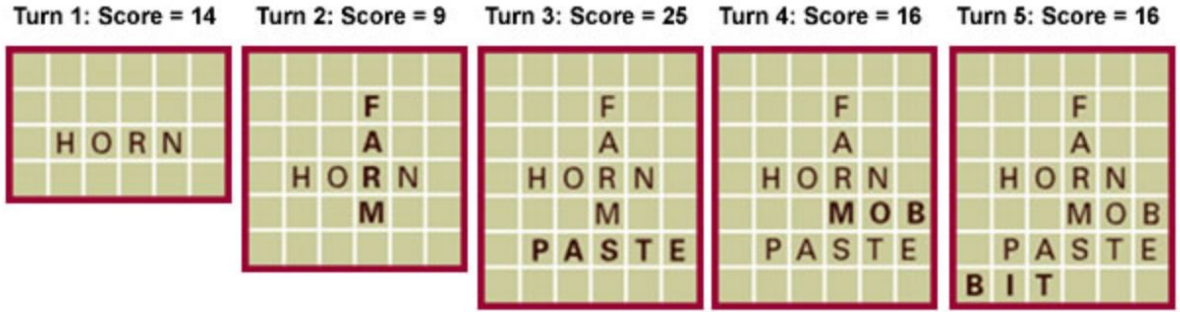

A B C D E F G H I J K L M N O
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
A1 T1 O1 P3
C3 E1 I1 R1 R1 S1 T1


A B C D E F G H I J K L M N O
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
T1 R1 I1 C3 E1 R1 A1 T1 O1 P3 S1 ⇒
C3 E1 I1 R1 R1 S1 T1

More typical "living room" play

SCORING EXAMPLES

In the following, the words added on five successive turns are shown in bold type. The scores shown are the correct scores if the letter R is placed on the center square. In Turn 1, count HORN: in Turn 2, FARM; in Turn 3, PASTE and FARMS; in Turn 4, MOB, NOT and BE; in Turn 5, BIT, PI and AT.

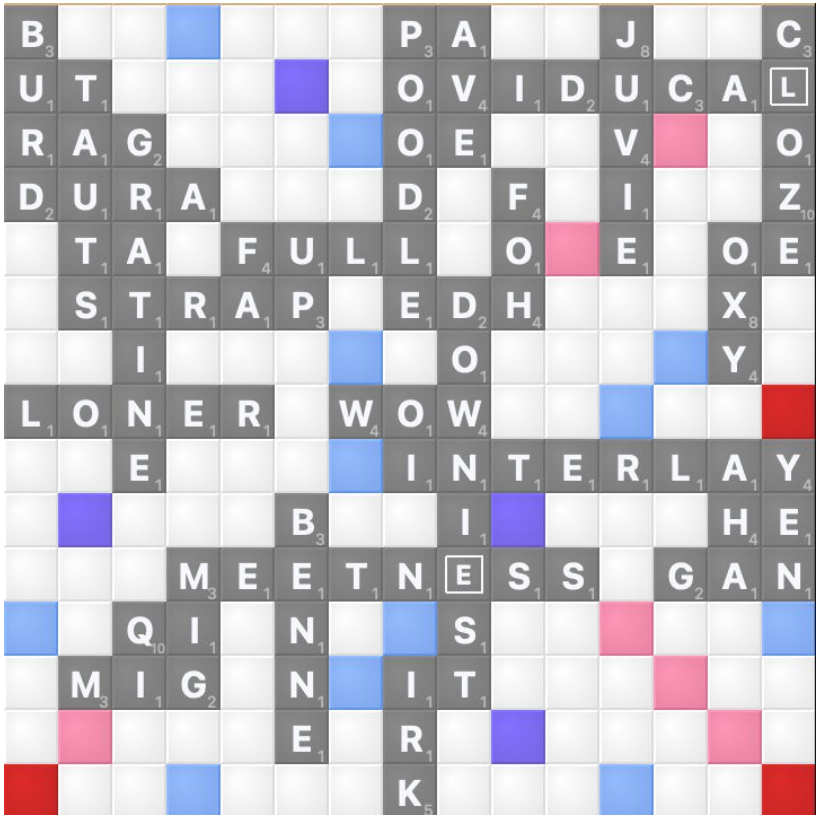


Bingos are arguably the most important part of the game.

#8

BINGO! If you play seven tiles on a turn, it's a Bingo. You score a premium of 50 points after totaling your score for the turn.

What high-level play actually looks like



To git gud at tournament-level Scrabble, you have to memorize a ton of obscure words. Knowing their meanings is only marginally useful.

What a winning Scrabble AI needs to do on each turn:

- The easy part: Identify all possible legal moves
- The hard part: Pick the best one

Easy part: Identifying valid board plays

- The played tiles have to all be in the same row or column.
- Pick an ordered (sub)set of your 7 tiles...
  - 7! + (7 choose 6)*6! + ... = 13699
- ...and a row/column and a starting point...
  - (15*2) * 15 = 450
- and then put your chosen tiles down, in your chosen order, skipping over already-filled cells (and rejecting plays that go off the board).
- 13699 * 450 = a mere 6 million or so, easy for a computer!

Easy part: Checking a potential play

- Checking which new words have been formed is inefficient naively, but can be improved a bit...
  - e.g., only check rows/columns that actually had tiles added to them
  - there is probably a neat data structure for this

- Checking whether the newly formed words are all legal is difficult for humans but easy for a computer (it knows all the words!)
  - Humans can bluff other humans...

Hard part: How good is a move?

- Is it a good idea to always pick a move with the highest score?

Hard part: How good is a move?

- Is it a good idea to always pick a move with the highest score?
    - Even this is enough to create a formidable opponent, but it's far from optimal.
    - Making a highest-scoring move can leave crappy tiles behind in your rack (e.g. a Q that just sits there turn after turn) and hamper your future success.

A better way to evaluate moves

- What factors are important?
  - Score matters! This is how you win games!
  - The tiles left behind in your rack
  - ...

A better way to evaluate moves

- What factors are important?
  - Score matters! This is how you win games!
  - The tiles left behind in your rack
  - The board position that you leave (are there high-scoring opportunities available for the opponent?)
  - Side note: The tiles in the *opponent*'s rack can be inferred, up to a point. Can you imagine how?

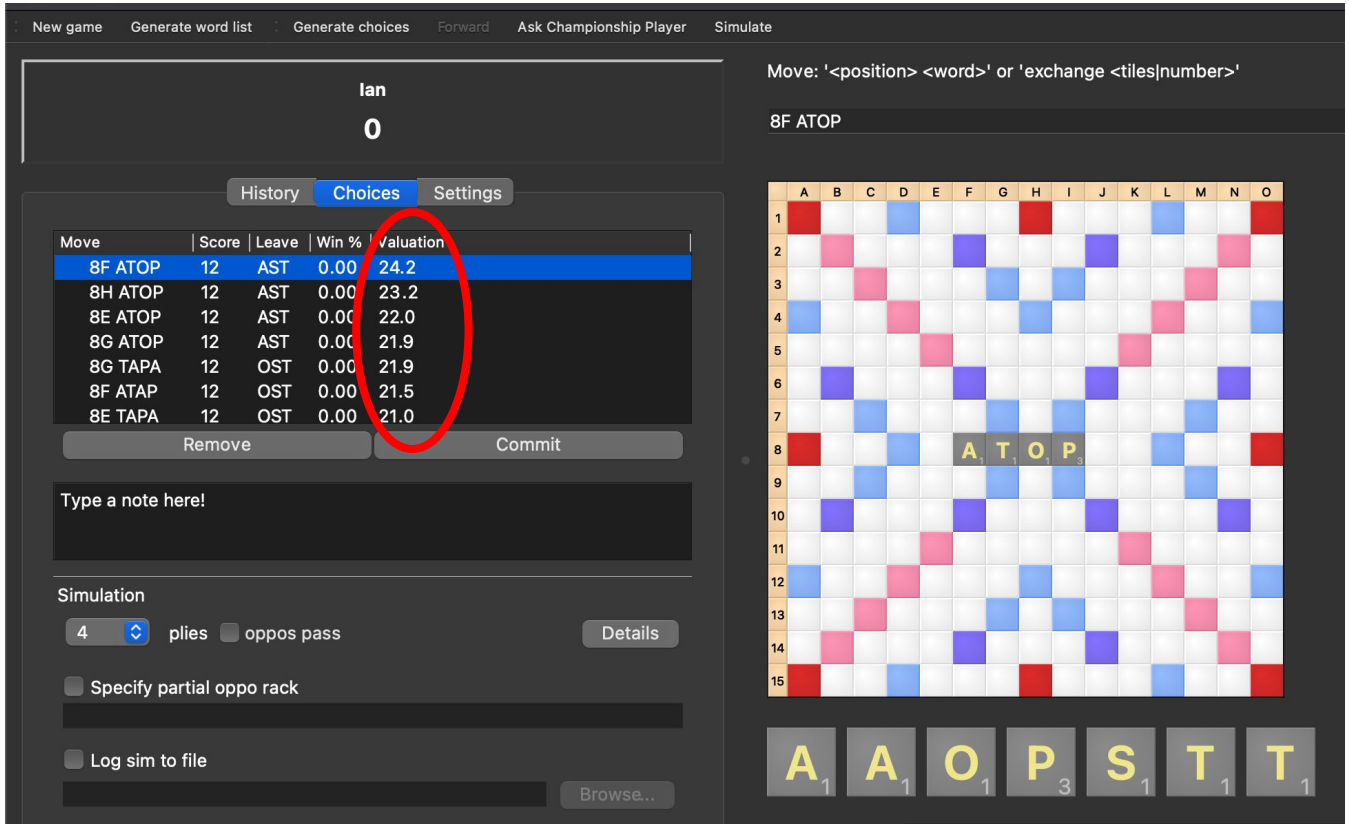
- How to quantify all this? How much do these factors matter relative to one another?

One idea: some kind of linear model

- Value = *a* * score + *b* * tiles the move leaves in your rack + *c* * board position + *d* * opponent's tiles...
  - but how do you quantify the strength of the tiles left? or (especially) the board position?

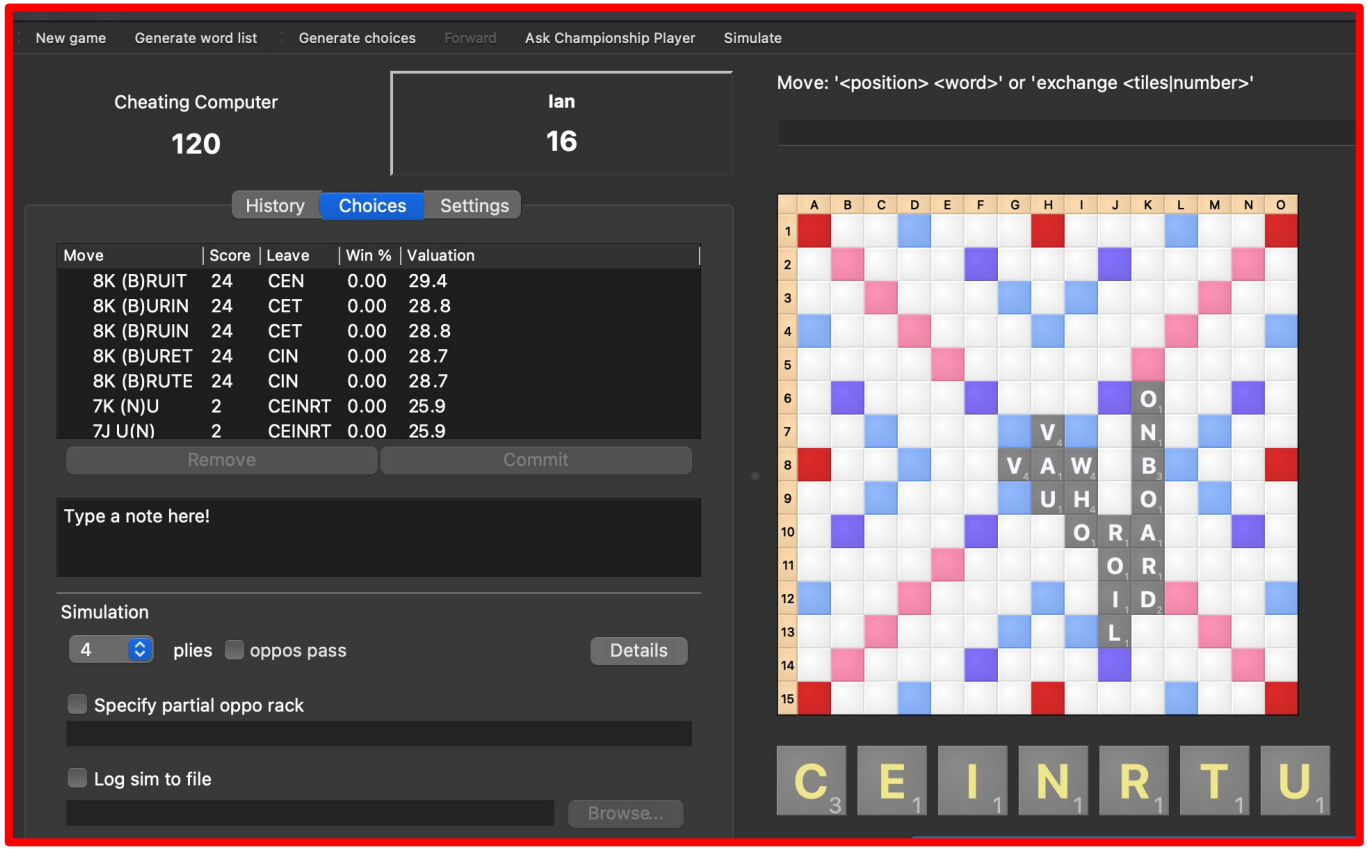
- This is hard! (I tried to do a CS238 project like this.) Let's see how one of the most prominent Scrabble AIs does it...

The Quackle AI finds the most promising moves



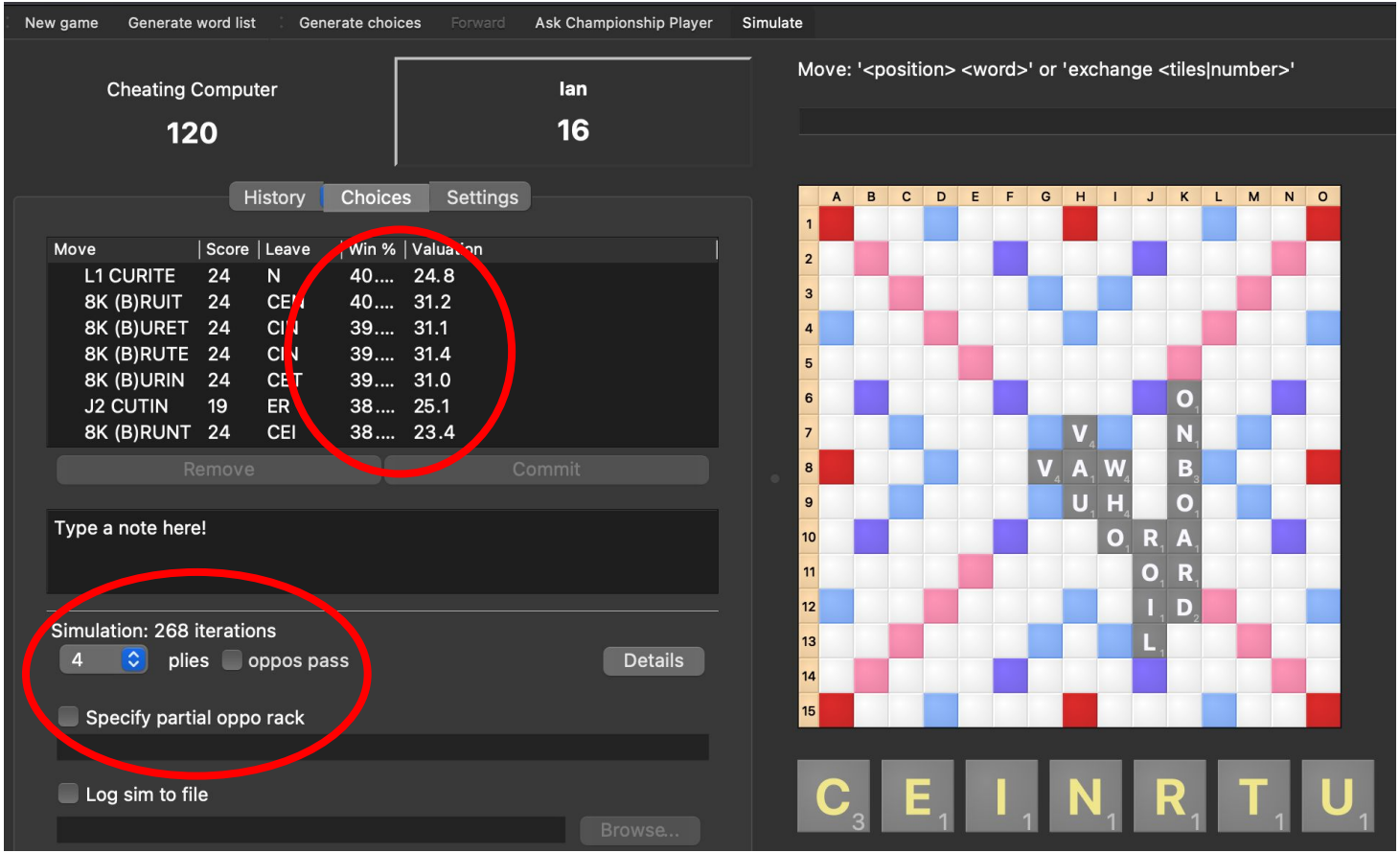
Notice that Quackle knows that this positioning of ATOP is better than our old one (which put vowels next to double letter squares)

Static evaluation



Playing NU for 2 points is still considered pretty strong because it leaves behind CEINRT, which has a high probability of allowing a bingo next turn

Dynamic evaluation (simulation)

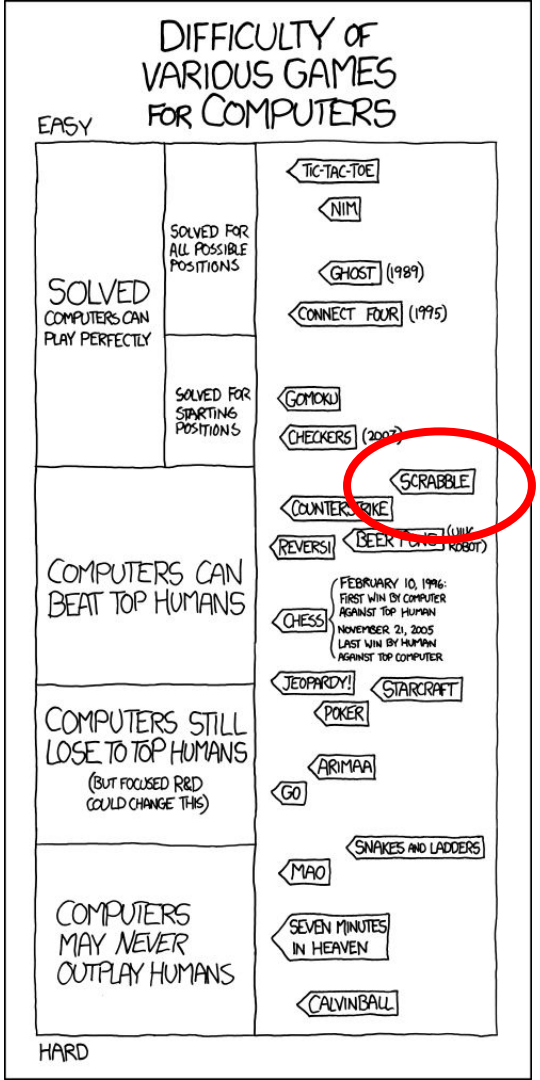


How good can this get?

- Well, how much time / memory / computing power do you have?

- Exhaustive simulation (checking every possible thing that could happen) is just not possible except in the "endgame" (when there are very few tiles left)
  - jargon: the state space is just too large

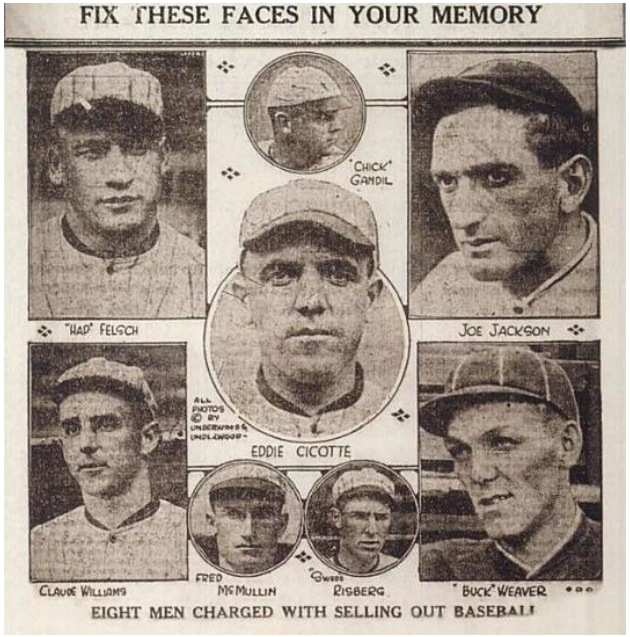
- Some prep can be done "offline"! Parameters (like the strength of tiles left behind in the rack) can be estimated by having the AI play millions of games against itself, and then they can be hardcoded in...



How good is Scrabble AI?

- This XKCD comic is from 2012.
- A 2022 article (see our course site) argues that Scrabble should be much lower down here and the AI's strength is overstated

What if we want to *lose* at Scrabble?



The 1919 White Sox, perhaps the most infamous (alleged) intentional losers in sports history

Hear me out. This gets surprisingly interesting.

Playing to lose

- If you want to lose, and the other player wants to win, congrats! There is no conflict, your dreams probably both come true, this is boring.

- What if *both* players are playing to lose (not to tie)?

    - For many games, this may end up being a kind of uninteresting mirror image of trying to win, with similar strategies.

    - In Scrabble, though, it becomes something altogether different and ridiculous...

Wait, why would we want to do this?

- Remember, CS64 is a fun class! Not everything we discuss will be practical. But...

- Can anyone think of a situation (in any game/sport, not necessarily Scrabble) in which both players/teams might actually play to lose?

Wait, why would we want to do this?

- Remember, CS64 is a fun class! Not everything we discuss will be practical. But...

- Situations like this *can* arise organically, usually as a result of bad design that creates perverse incentives.
  - For example, in some tournament structures, there may be a match between two teams who are already guaranteed to advance, but the winning team ends up facing a stronger opponent in future rounds.

Why is losing in Scrabble hard?

- In Scrabble, on each turn, you can:
  - play one or more tiles
  - exchange as many of your tiles as you want
  - pass

V. Ending the Game

V.A. Final Play

The game ends when one player has successfully played all of his/her tiles, and the bag is empty.

V.B. Six-Zero Rule

The game may also end by either player neutralizing the game timer after a sixth successive zero-scoring play from passes, exchanges, challenges, or illegal plays.

Aside: badly written rules

#9

The game ends when all letters have been drawn and one player uses his or her last letter; or when all possible plays have been made.

GOT QUESTIONS? **READ FAQ**

The position from which no play is possible no matter what tiles are held, which is reached with the fewest plays and tiles (found by Kyle Corbin of North Carolina) is:

```
  (J)
J U S
  S O X
   (X) U
```

Fun side note: the game can in theory end very early because of no playable moves.

Ruling out some silly stuff...

- In a tournament, you can lose the game by going first, never making a move, and running out of time. Or flipping the table. We won't consider this kind of thing.

- Let's not deal with the challenge rules:
  - In some tournaments, an incorrect challenge awards 5 to 10 points to the other player. In our setup, there would be no reason not to do this every time the opponent played a word.
  - We'll assume all plays are valid.

An exchange-based strategy

V.G. Adjusting the Score (Unplayed Tiles, Overtime)

If you go out, increase your score by double the value of your opponent's unplayed tiles. If the game ends with neither player going out, each player's score is reduced by the total value of his/her unplayed tiles.

0 Points - Blank tile.
1 Point - A, E, I, L, N, O, R, S, T and U.
2 Points - D and G.
3 Points - B, C, M and P.
4 Points - F, H, V, W and Y.
5 Points - K.
8 Points - J and X.
10 Points - Q and Z.

So we want to accumulate high-valued tiles in our rack by exchanging, with the goal of having a higher total than the opponent at the time that the game ends from the six-zero rule.

Even this is complex

- How do we know which tiles we should exchange?
    - If we have a Q or Z (10 points), we should keep it. Conversely, blanks are worth 0 and should always be exchanged.
    - What if we have a D (2 points)? Should we save it, or exchange and gamble on getting an even better tile?
        - Depends on the distribution of tiles...
        - And our decision might change if it's the first round of exchanges vs. the third...

But wait, it gets worse

Anyone see why just thinking about exchanges isn't enough?

Suppose I'm player 2. I'm about to make my third exchange, which would end the game...

But wait, it gets worse

Suppose I'm player 2. I'm about to make my third exchange, which would end the game...

- but I might not want to! What if I know I have tiles that are weaker than average?
- I want to extend the game to try to get more chances to improve my rack.
- So I need to play something that scores points. But scoring points is bad, so I play a 2-letter word like IS. This resets the six-zero counter.

But wait, it gets worse

- If a player is about to end the game with a sixth zero-scoring turn, they should only do so if they think they are "ahead".

- This complicates the strategy a lot! For example, suppose I have a rack of low-valued tiles after my first exchange, including a blank. If I suspect I'm going to have to play a weak word to extend the game, should I actually *save* the 0-point blank to reduce the points I score on that play?

And even worse

- In fact, either player has the power to reset the six-zero count at any time, by playing a word!
- Since at least one player will usually think they could be behind, this means someone will always keep grudgingly playing a low-scoring word to extend the game...
- Knowing this, is it better to optimize one's rack for low-scoring plays, rather than for the penalty on remaining tiles?
- Open questions! (I may try to do a paper on this)

Some takeaways

- Strategy and game theory can get surprisingly complicated!

- Playing to lose can be as complicated as playing to win!

- When designing an AI – whether it's "winning" or "losing" – how can we definitively claim optimality? How do you know you haven't overlooked some even better strategy?
  - What if there's not even a single optimal solution (e.g., paper-scissors-rock)?

Ghost

a good game for long car rides?

...in the age before smartphones, at least. But at least the driver can play!

The rules of Ghost

- Players take turns naming letters.
- If a player's added letter causes the ordered string of all letters so far to become a word, that player loses.
- After a player adds a letter, the opponent can challenge them to produce a valid word that could still be formed. If the player can do that, they win. Otherwise, the opponent wins.
- The game often descends into arguing about whether a word is legitimate...

An example

- Player 1 says **B**.
- Player 2 can't say E, for example (it would make BE). They decide to say **R**.
- Player 1 says **U**, perhaps hoping to trap player 2 into eventually making BRUTAL.
- Player 2 says **S**, hoping to trap player 1 into making BRUSH.
- Player 1 has a trick up their sleeve! They say **C**, hoping to eventually trap player 2 into eventually making BRUSCHETTA.
- Player 2 challenges Player 1, and then the game descends into an argument about whether BRUSCHETTA is a word.
- Players 1 and 2 go out for Italian food so that Player 2 can discover the joys of bruschetta.

Strategy

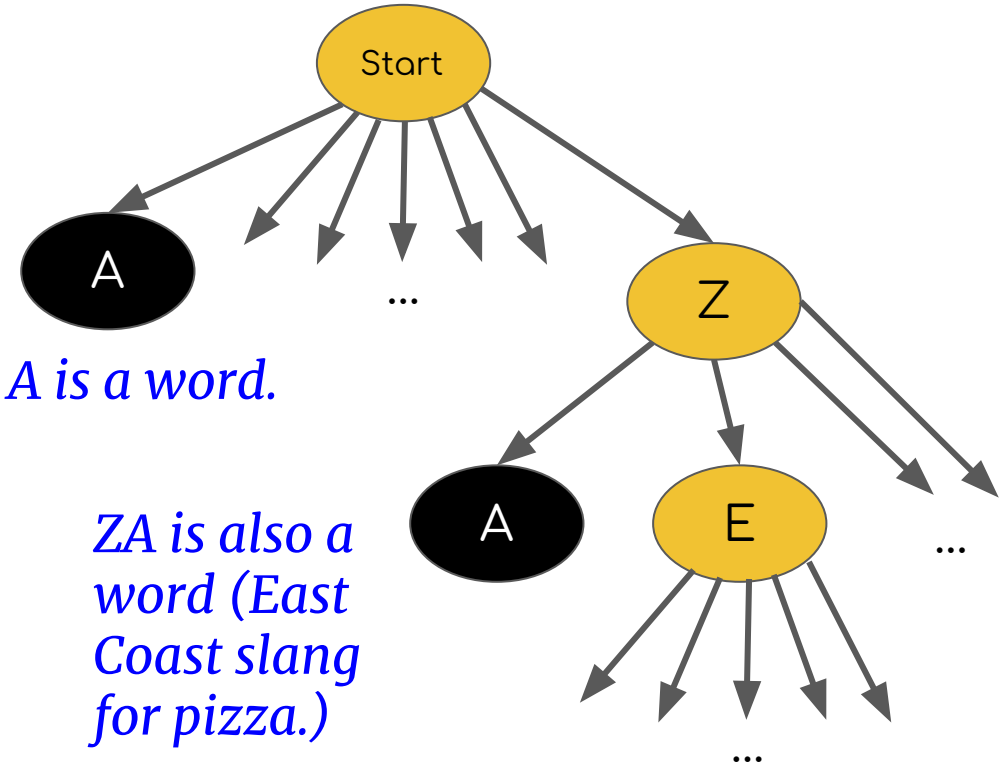
- This is a lot simpler than Scrabble!
- Once again, we will assume there is no bluffing.
- Call the list of letters that have been named so far a *state*.
- Claim: Each state is either "winning" or "losing" for the current player, defined in a recursive way:
  - If all choices would either be illegal or complete a word, the state is losing. (base case)
  - If at least one choice would hand the other player a losing state, the state is winning.
  - Otherwise (i.e. if all choices would complete a word or hand the other player a winning state), the state is losing.

Back to our example

- Some losing states include BRUSCHETT, BRUSCHE, BRUSC.
- Some winning states include BRUSCHET, BRUSCH, BRUS.
- Notice that *every* state is either winning or losing, so this game can be completely solved (as long as the players agree on a wordlist in advance).
- That is, if both players play optimally, the game is either always a win for the first player or always a loss for the first player (though we don't know which).
  - Is there a point to playing a solved game?

How to decide whether a state is winning?

- Notice that the game can be described by a tree:



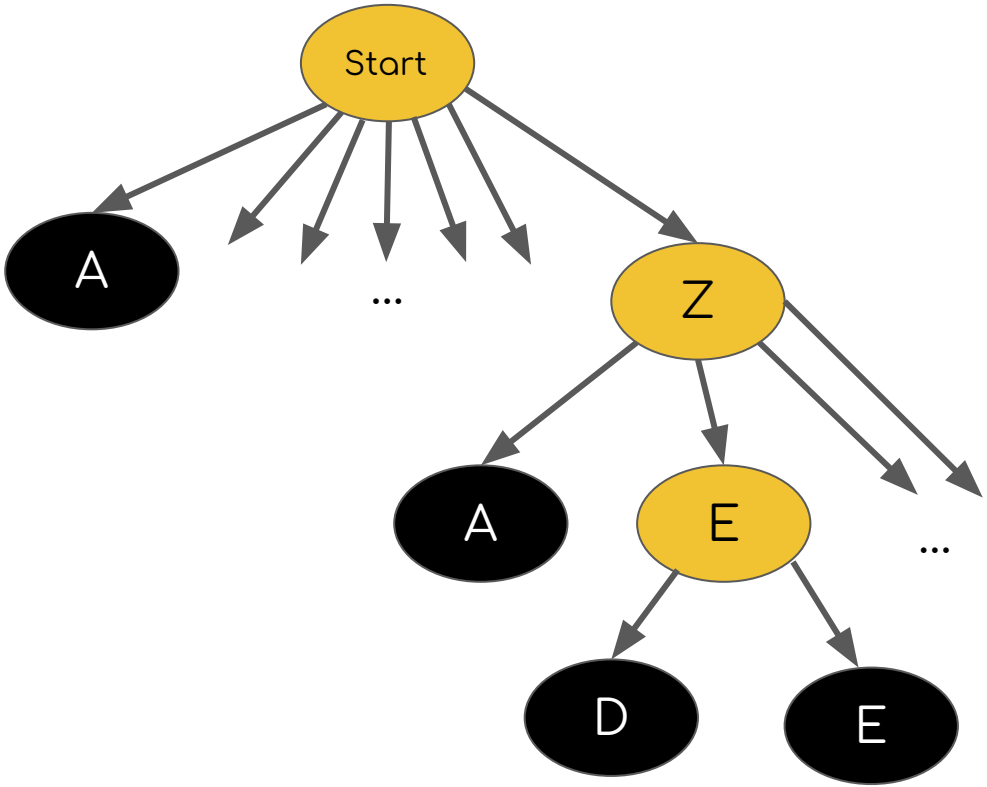
A is a word.

ZA is also a word (East Coast slang for pizza.)

black = is a word
red = definitely losing
green = definitely winning
yellow = we're not sure yet

How to decide whether a state is winning?

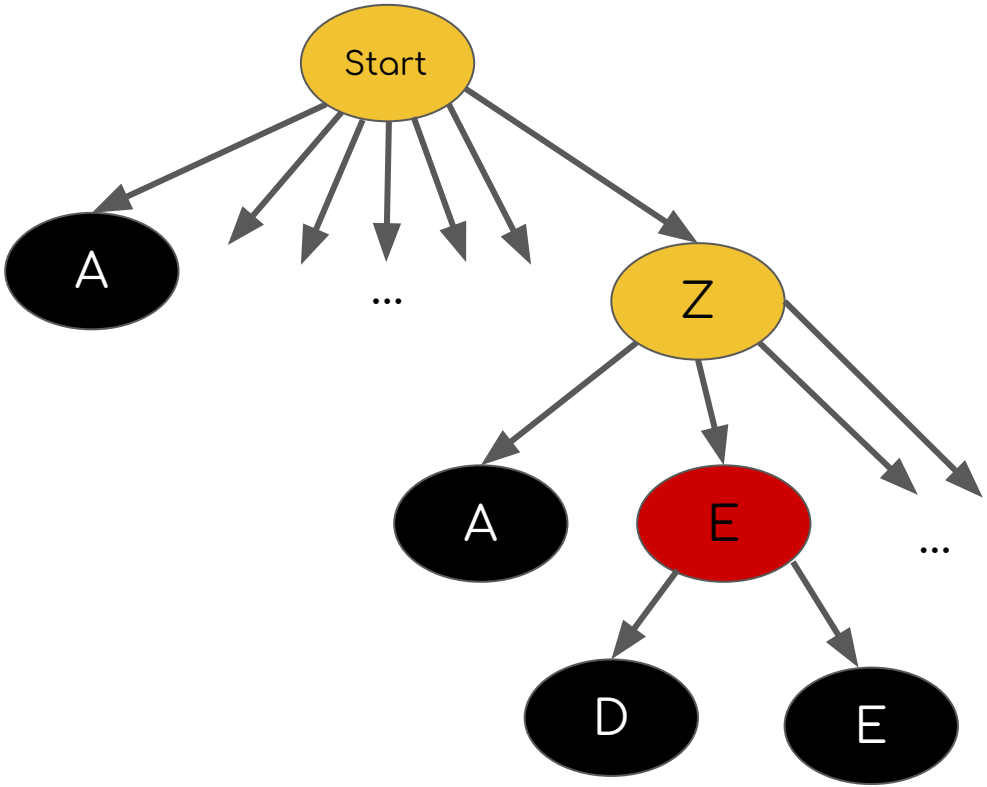
- Notice that the game can be described by a tree:



*Pretend, just for simplicity, that the **only** words extending ZE are ZED and ZEE.*

How to decide whether a state is winning?

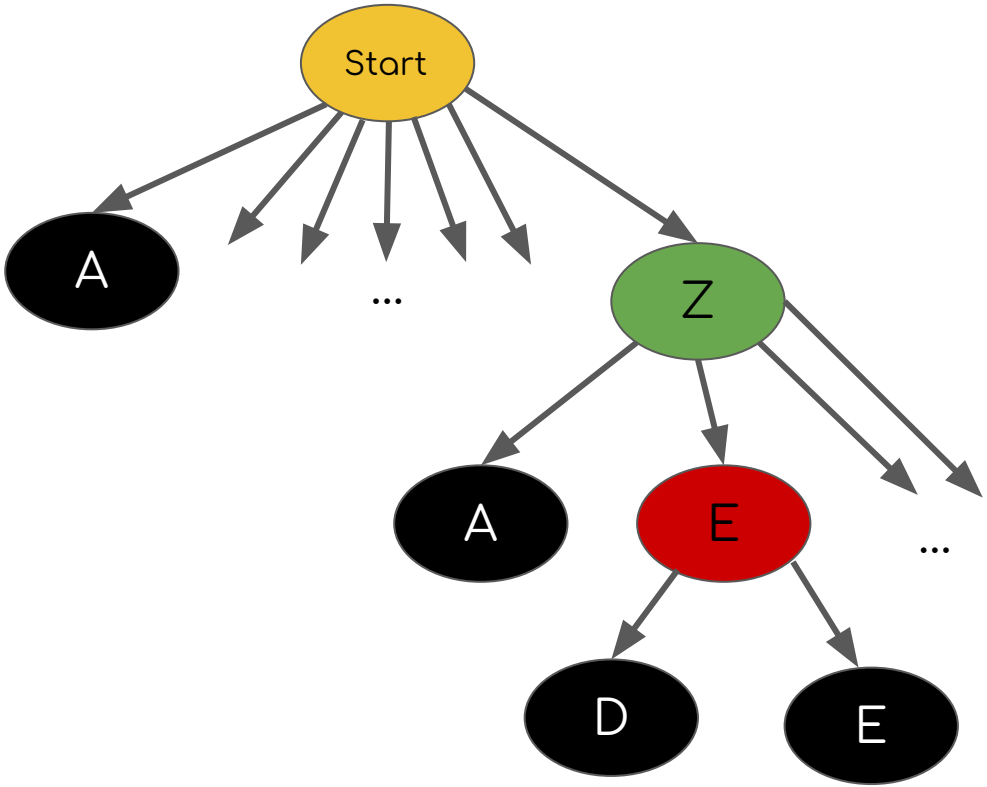
- Notice that the game can be described by a tree:



Now we know E is a losing state. There is no valid move from there.

How to decide whether a state is winning?

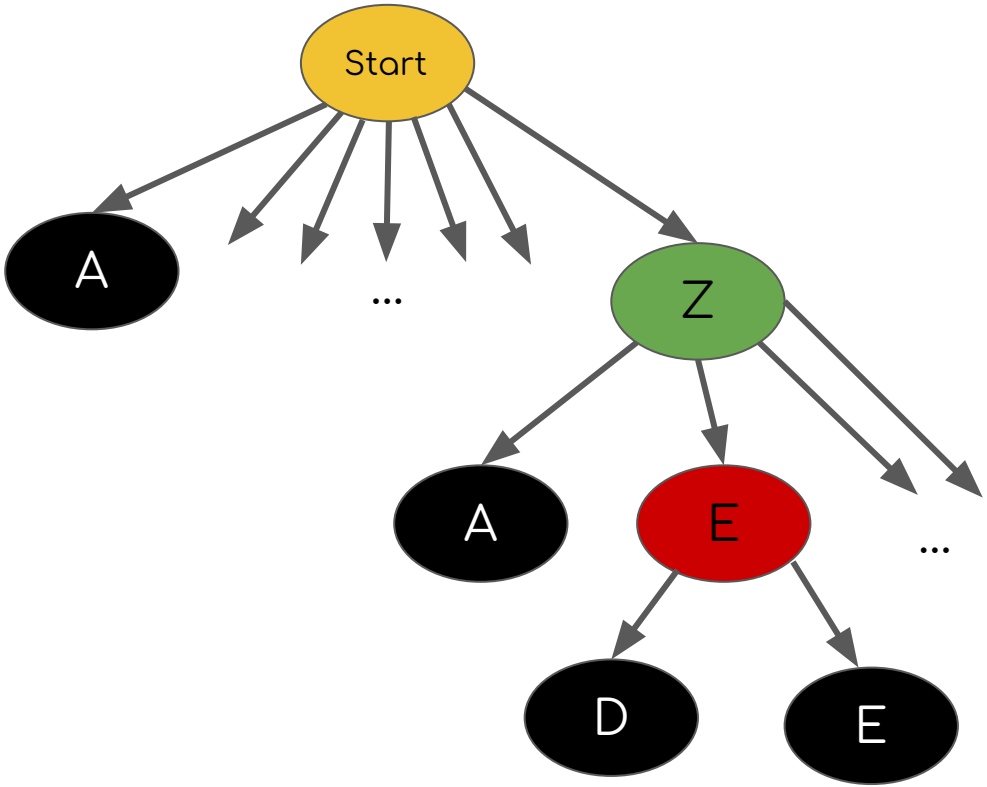
- Notice that the game can be described by a tree:



We also now know that Z is a winning state, since that player can play E from there to leave the opponent with a losing state.

How to decide whether a state is winning?

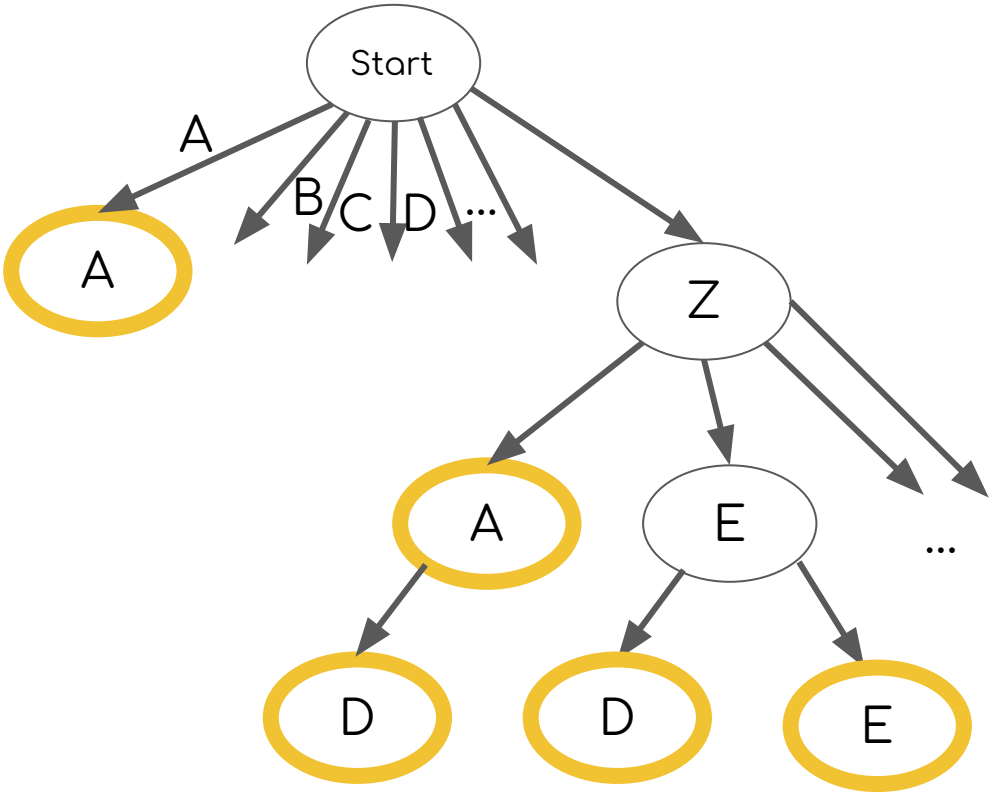
- Notice that the game can be described by a tree:



We're still not sure about the start state. It's still possible that one of the other initial moves (like B) is a losing state, so the start state could be a winning state.

But if A, ZA, ZED, ZEE were the only words, the start state would be a losing state.

Representation as a "trie"



Each node is either NULL or has 27 fields:

1 for each of the possible extensions A, ..., Z from here

1 for whether this node itself ends a word

For example, suppose we also had ZAD in this wordlist in addition to A, ZA, ZED, ZEE.

Why tries?

- It's easy to build an entire wordlist into a trie. Think about how you would start with an empty trie and add words... (don't worry about how you would delete words)

- Once you have a trie, it's easy to determine whether each node is winning or losing in Ghost. Traverse the tree and recursively decide for each node by examining its descendants (if any).
  - Because you instantly lose in Ghost if you make a valid word, don't explore descendants of nodes with the "this is a word" flag set.

- This is much more efficient than repeatedly iterating through the word list to see whether adding each letter to the current word produces another valid word.
  - Hash tables would also work, but would take up much more space (by storing each word completely, whereas the idea of the trie is to exploit common prefixes)