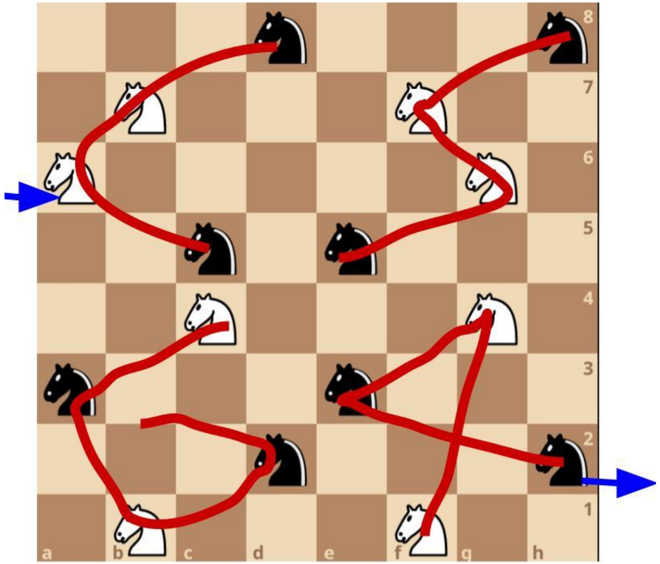
CS64: Computation for Puzzles and Games



Autumn 2022

Lecture 9: Video Games and Speedruns

A small plug

It's good!

And has some puzzly stuff

Announcements

- Reminder: this is our last Wednesday lecture! (because Week 10 is a grind for everyone as is)
  - I'm sorry we never got to Grundy numbers. Go look them up, it's nice to be aware of them...

Announcements

- Reminder: this is our last Wednesday lecture! (because Week 10 is a grind for everyone as is)
  - I'm sorry we never got to Grundy numbers. Go look them up, it's nice to be aware of them...

- Please fill out the attendance / puzzle hunt time Google form! (See the pinned Ed post)
  - I will follow up if you don't, but still, please do

Announcements

- Reminder: this is our last Wednesday lecture! (because Week 10 is a grind for everyone as is)
  - I'm sorry we never got to Grundy numbers. Go look them up, it's nice to be aware of them...

- Please fill out the attendance / puzzle hunt time Google form! (See the pinned Ed post)
  - I will follow up if you don't, but still, please do

- The puzzle hunt looks like it will take 2-3 hours. It has a nice payoff at the end, so I encourage doing the whole thing! There will also be prizes for the fastest team(s).
  - Final announcement of date/time tonight

Tool-assisted speedruns

- Whereas different video games have different notions of "score" (if any), time is a universal currency. How fast can we beat this video game that we like?
- ...and can we do it even faster than that? Can we shave off another second? Can we get the time below some round-number threshold like the 2 minute mark?
- What if we get computers to help us?

https://www.youtube.com/watch?v=brSsTm_yz0U

https://www.youtube.com/watch?v=qf-tu2ojOb8

https://www.youtube.com/watch?v=_Y-S-Btd2zU

Interlude: Why do we care?

- Especially in the frenetic modern era, humans insist on doing everything fast
- Consider the similarity to the problem of getting from point A to point B as fast as possible during commute traffic...
  - and the "state" could be complicated, e.g., how much do you spend on bridge tolls? how much gas do you use?
- Researchers (e.g., DeepMind) use video games as test beds for AI because they are complex but not too complex

How can computers help us here?

- Execution: Perform acts of frame-perfect dexterity not (consistently) achievable by our puny, fallible human bodies

How can computers help us here?

- Execution: Perform acts of frame-perfect dexterity not (consistently) achievable by our puny, fallible human bodies

- Planning: Find glitches and optimal routes

How can computers help us here?

- Execution: Perform acts of frame-perfect dexterity not (consistently) achievable by our puny, fallible human bodies

- Planning: Find glitches and optimal routes

- Why isn't optimal routefinding easy? Just do the thing that gets you the farthest, the fastest, right?

https://www.youtube.com/watch?v=BhIgB-VZRKQ

What happened there?

- **"Damage boosting"**: our hero took damage from a bat to get knocked back onto a platform, avoiding a long trip downstairs

- The game state is more complicated than it may seem:

  - We only have so much health. We may be able to refill it using items, but doing that takes time!

  - A special subweapon (the watch) was needed to stop time to get the bat to arrive at the right time. Getting a subweapon takes time!

  - Subweapons consume hearts, so it matters how many hearts we have. Getting hearts takes time!

Dynamic programming interlude

Mario's extremely basic adventure
(probably like 50 bucks on Switch)

In this game, Mario has two kinds of move

Option 1: Go forward one step

Option 2: Jump

coins are good
you want as many as possible
because Mario's life is empty






enemies do not move

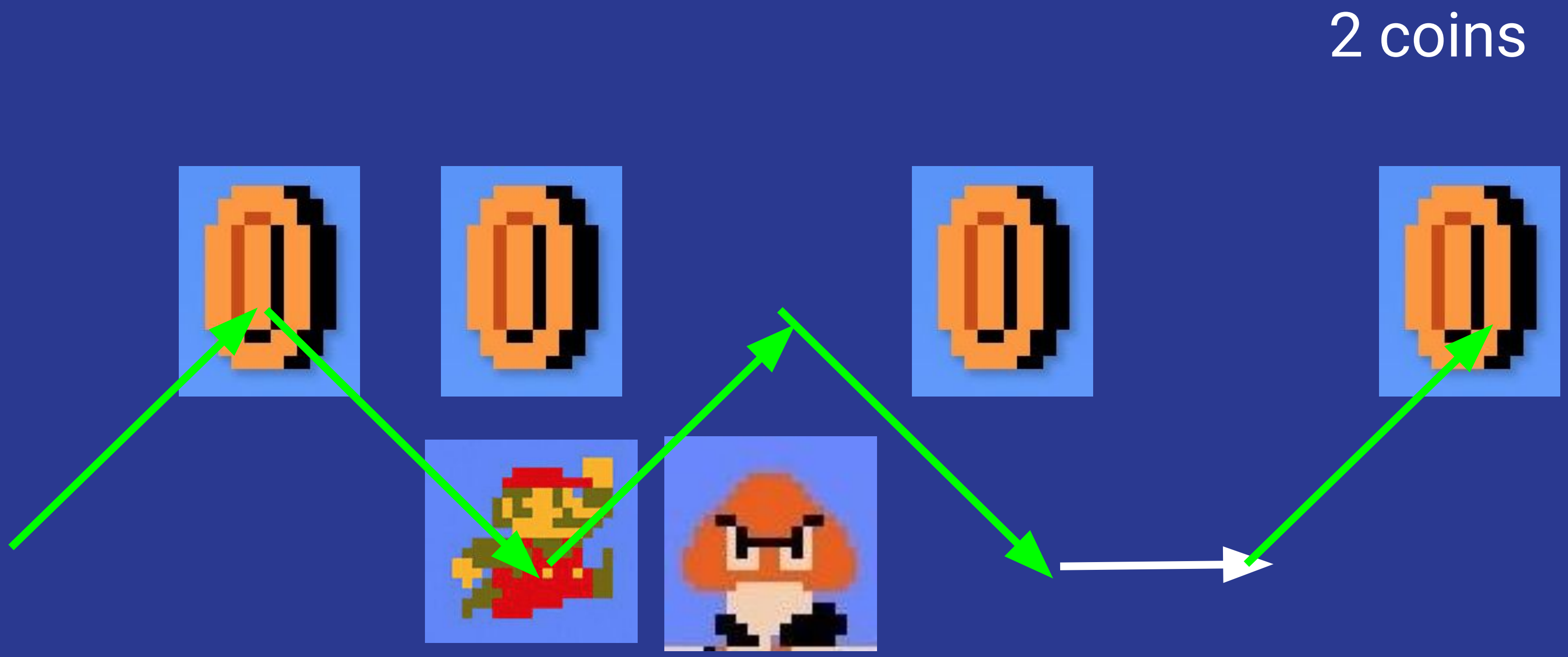
(they've been doing this for 35+ years, the excitement isn't there anymore)

not OK to walk into enemies

how did you get hit, it was just standing there

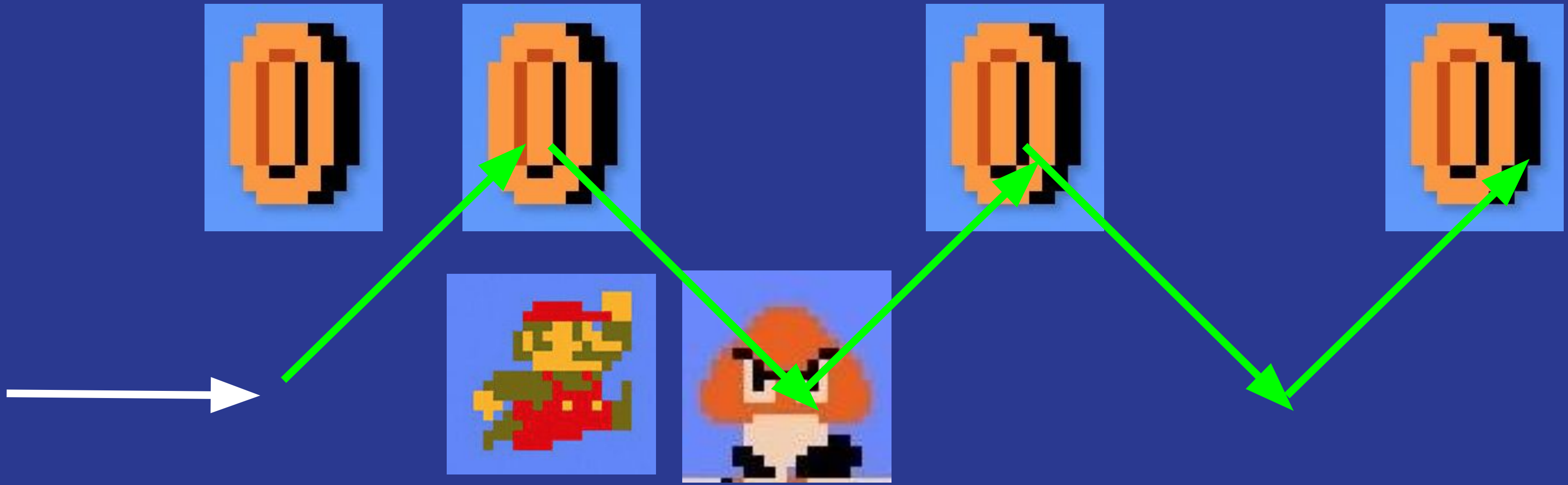
OK to land on enemies
because Mario is an asshole

Greedy strategies aren't always optimal



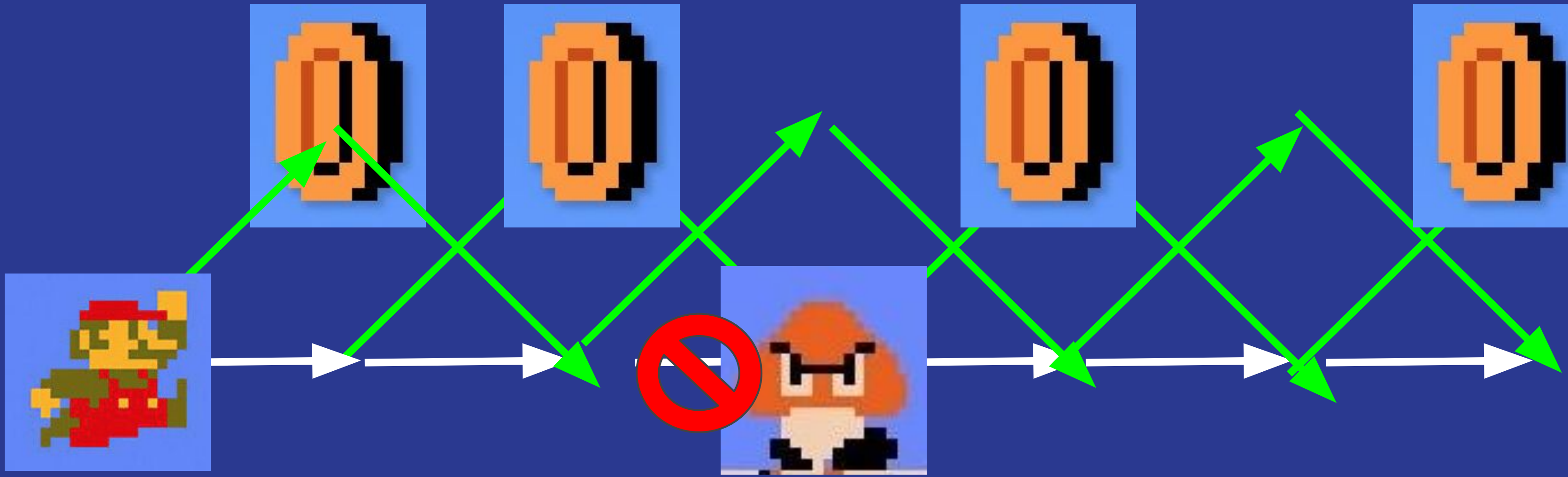
What we should have done

3 coins



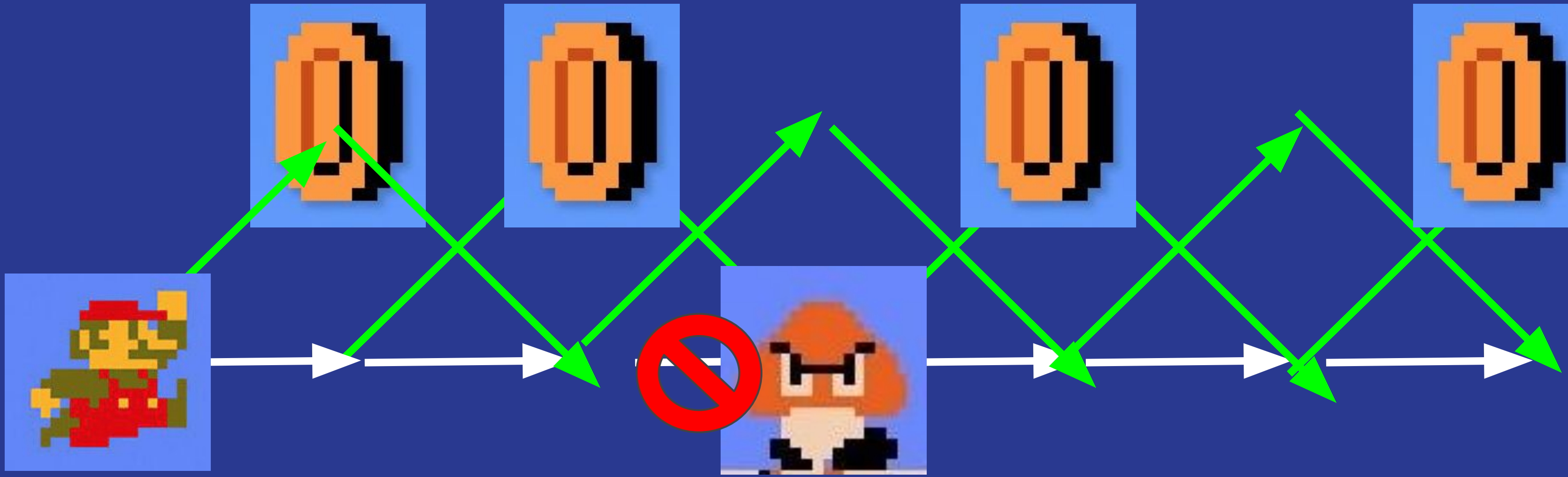
Why not just try every path?

Exponential number...



Why not just try every path?

Exponential number... <u>so any solution that explicitly considers them all is exponential</u>



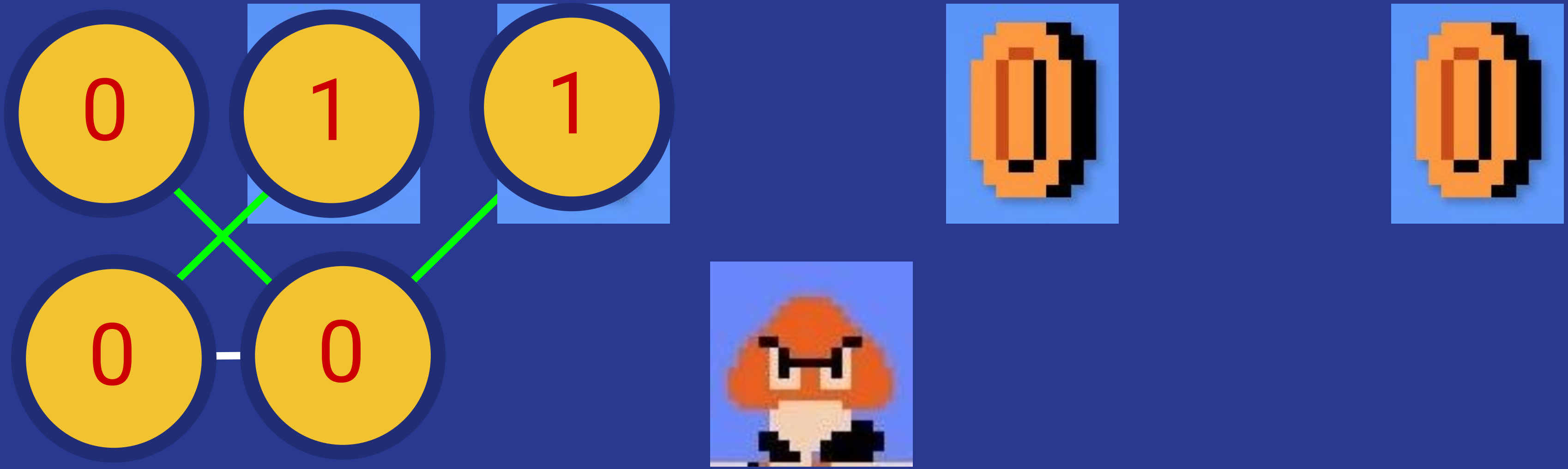
Solving via DP

what? we can't get here... but you'll see why we need it

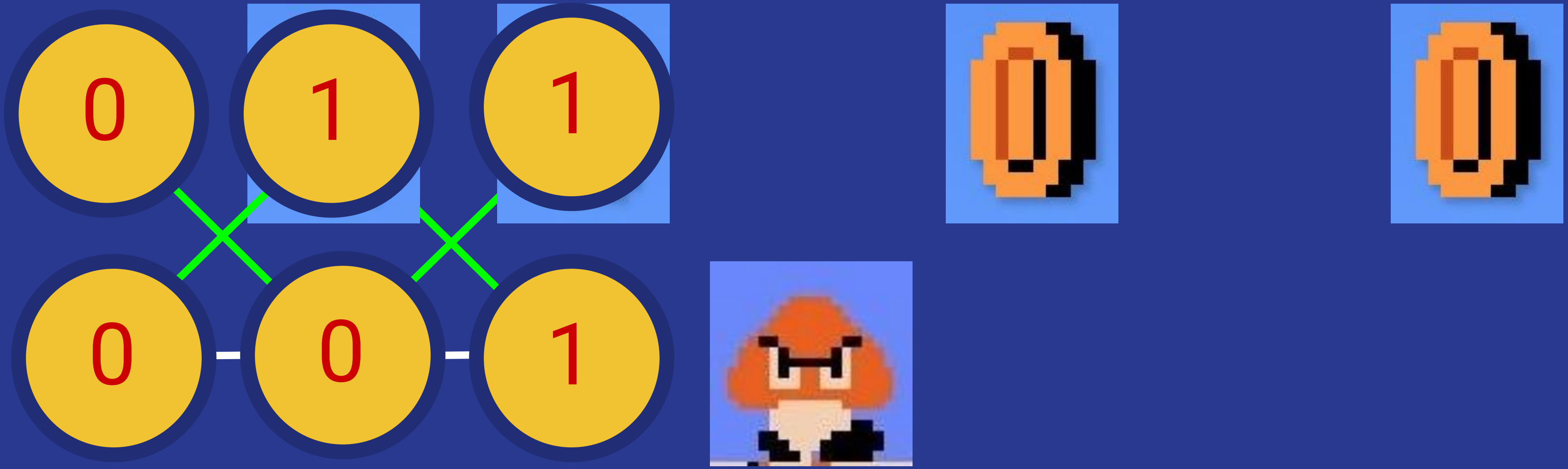
top row cell:
value from downleft,
plus 1 if coin

bottom row cell:
max of:
1. value from upleft
2. value from left if
no enemy here

top row cell:
value from downleft,
plus 1 if coin

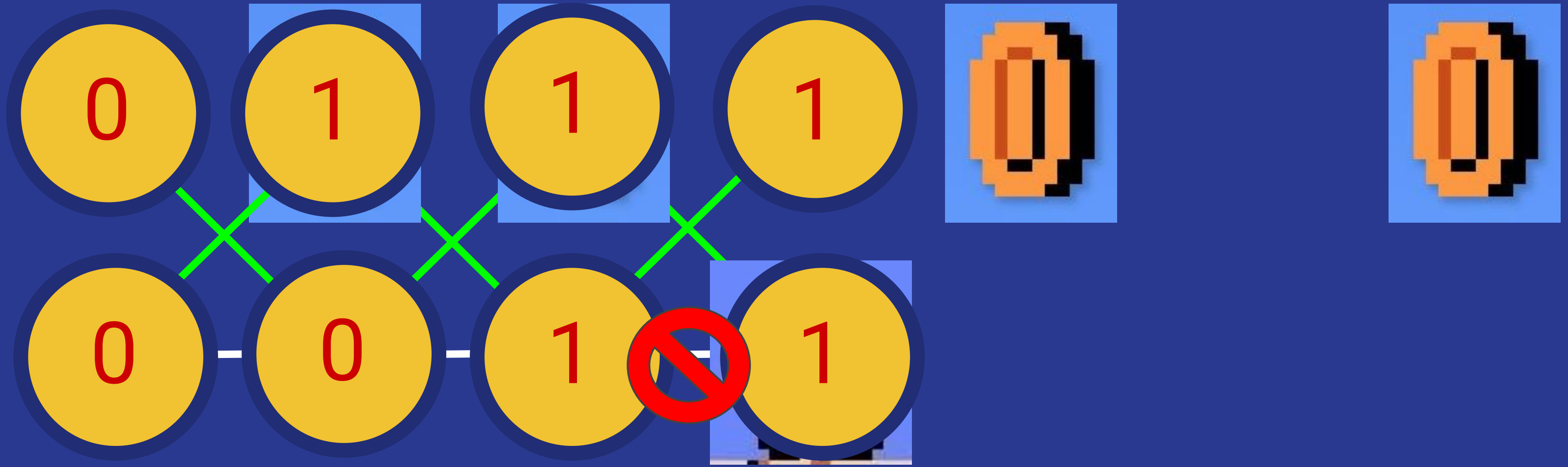


bottom row cell:
max of:
1. value from upleft
2. value from left if no enemy here

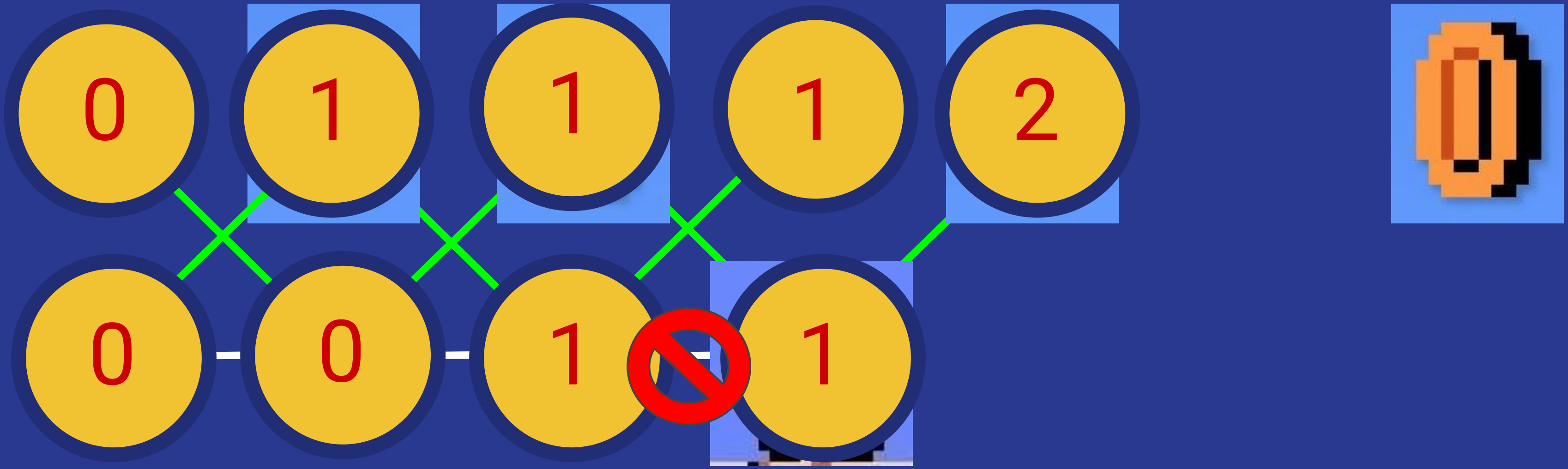


top row cell:
value from downleft,
plus 1 if coin

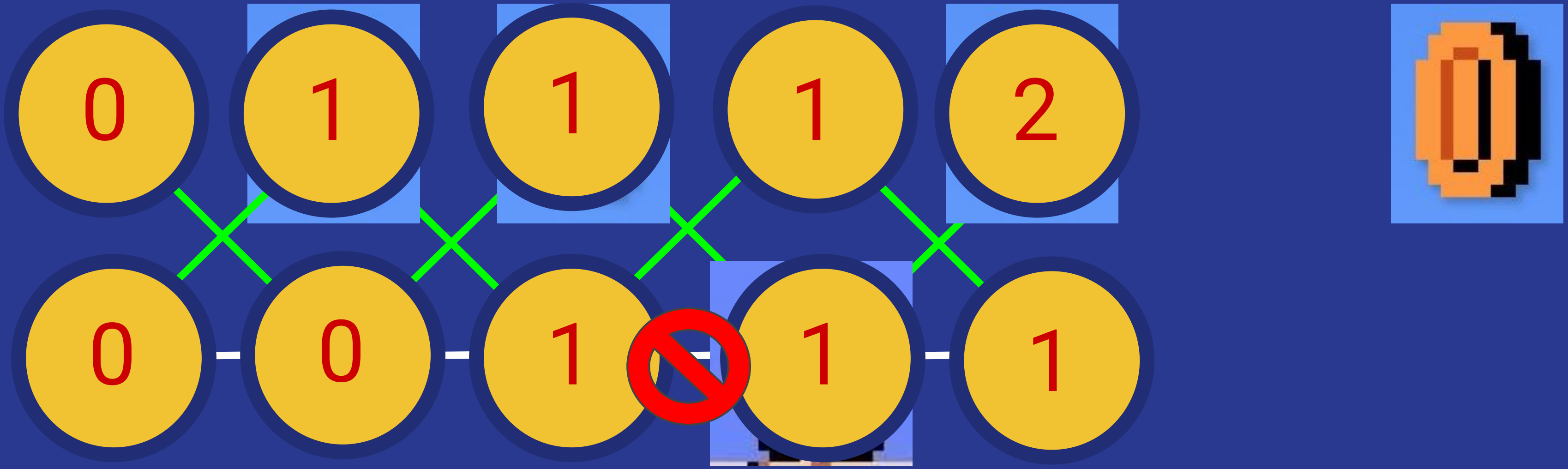
bottom row cell:
max of:
1. value from upleft
2. value from left if no enemy here



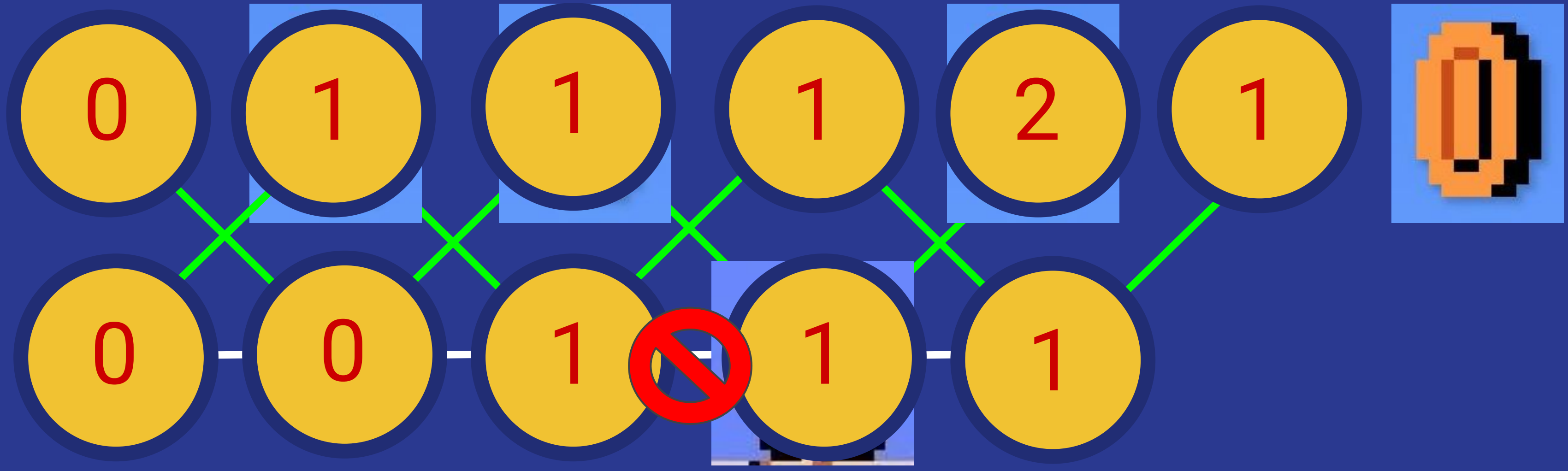
top row cell:
value from downleft,
plus 1 if coin



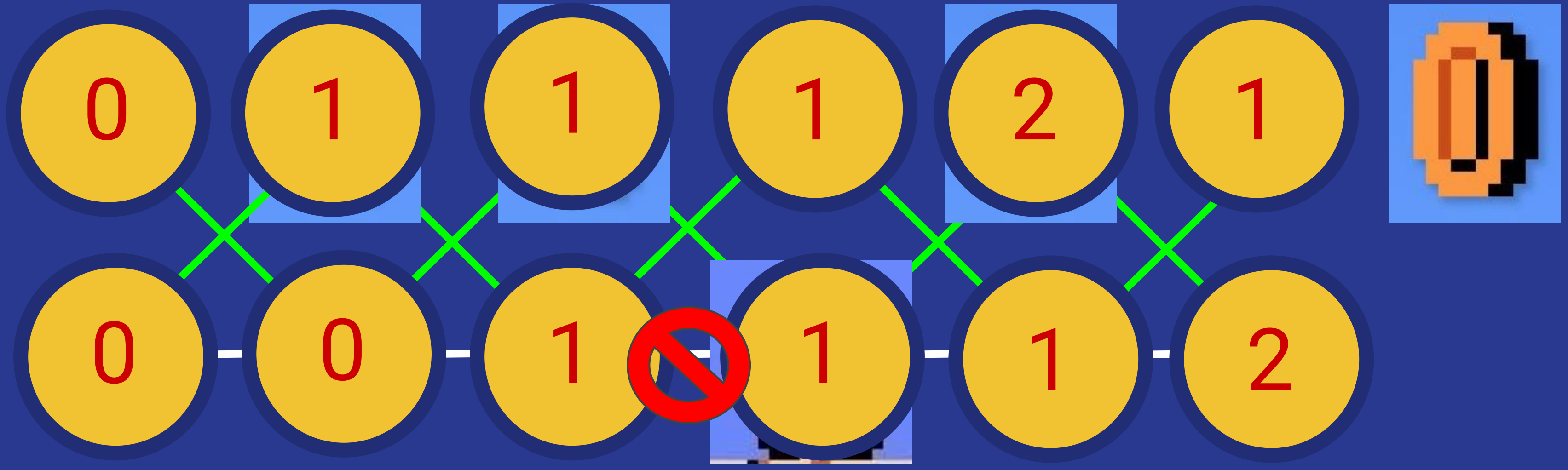
bottom row cell:
max of:
1. value from upleft
2. value from left if no enemy here



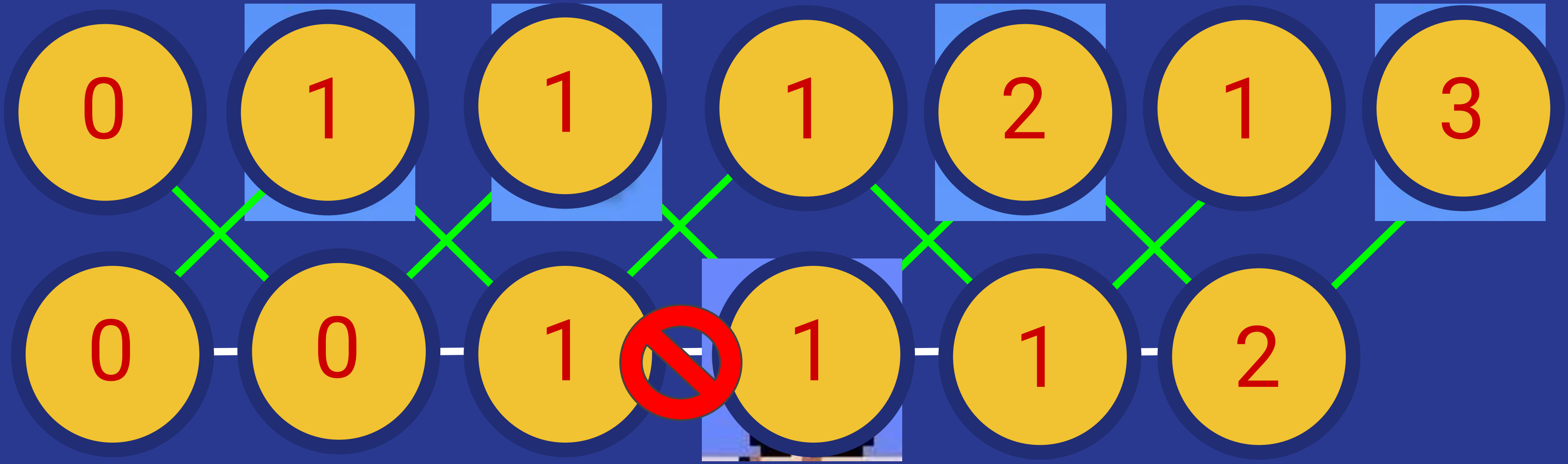
top row cell:
value from downleft,
plus 1 if coin



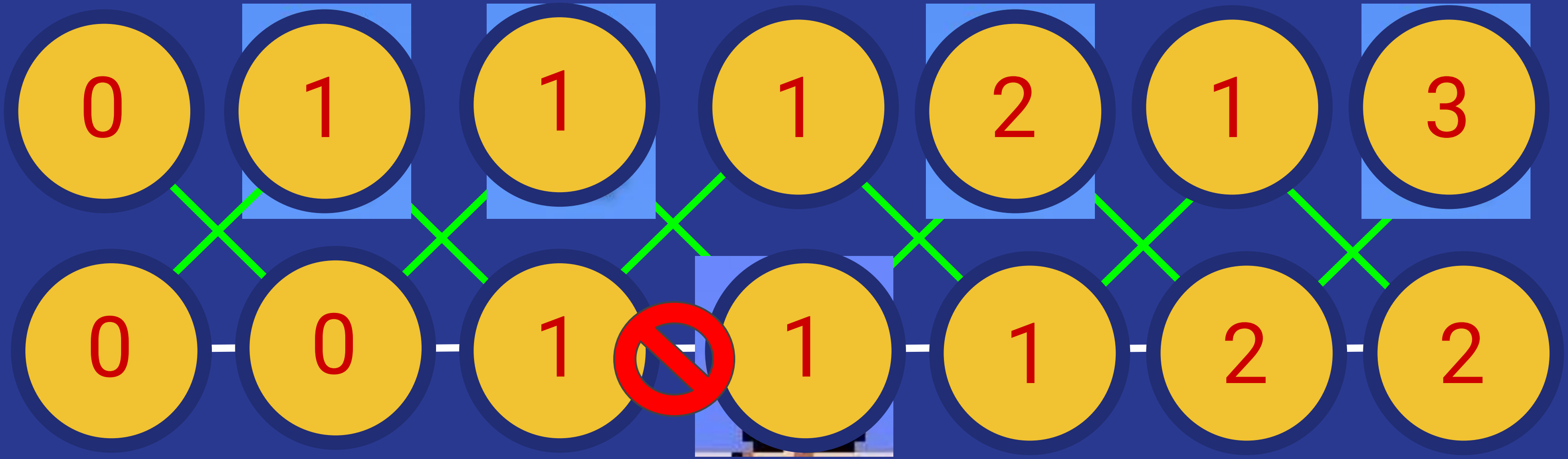
bottom row cell:
max of:
1. value from upleft
2. value from left if no enemy here



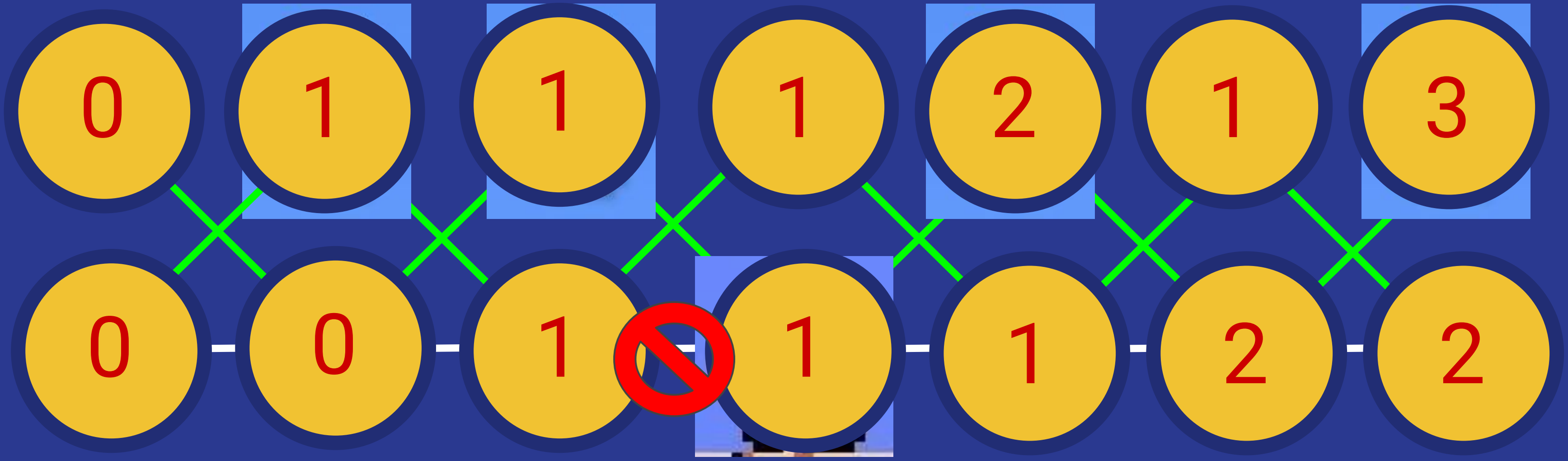
top row cell:
value from downleft,
plus 1 if coin



bottom row cell:
max of:
1. value from upleft
2. value from left if no enemy here



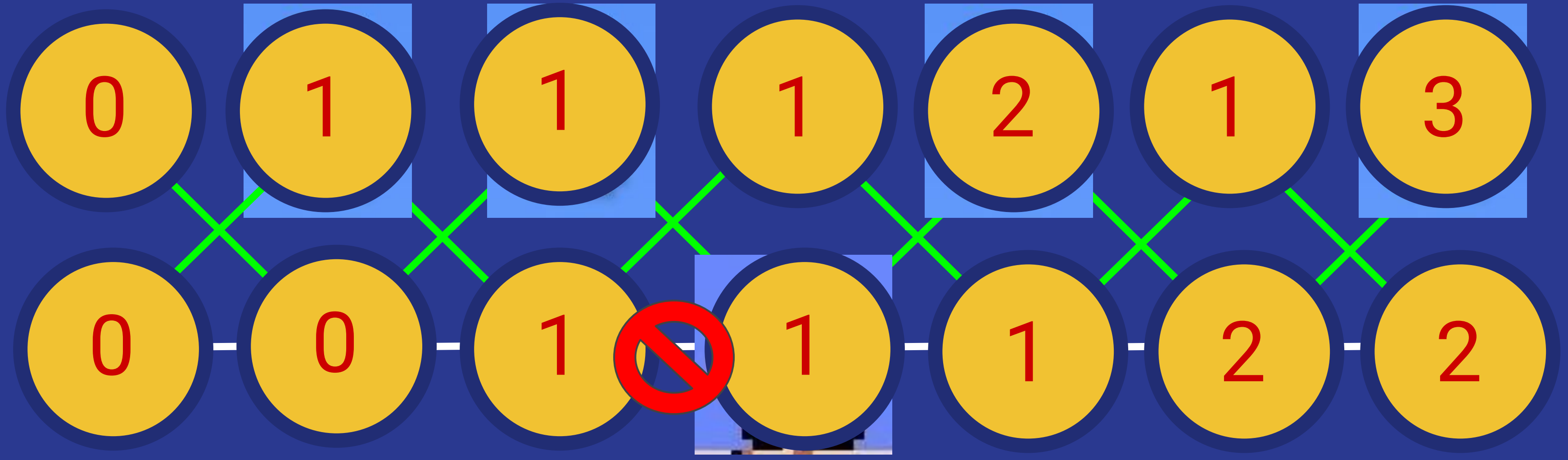
Wait a minute...
Isn't this just the "exponential" slide again?



Wait a minute...

Isn't this just the "exponential" slide again?

No! We took <u>linear time</u>.



once we get this far, the strategy from then on doesn't depend on how we got there

Code!

```
def solve(length, coins, enemies):
    dp = [[0 for _ in range(2)] for __ in range(length)]
    for i in range(1, length):
        # in second index, 1 = top row, 0 = bottom row
        dp[i][1] = dp[i-1][0] + (1 if coins[i] else 0)
        dp[i][0] = max(-1 if enemies[i] else dp[i-1][0], dp[i-1][1])
    print(dp)
    print(max(dp[length-1][0], dp[length-1][1]))

solve(7, [False, True, True, False, True, False, True],
      [False, False, False, True, False, False, False])

# (base) Ians-MacBook-Air:Desktop iantullis$ python mario.py
### [[0, 0], [0, 1], [1, 1], [1, 1], [1, 2], [2, 1], [2, 3]]
# 3
```

More space-efficient code!

```python
def solve(length, coins, enemies):
    prv = [0, 0]
    nxt = [0, 0]
    for i in range(1, length):
        # in second index, 1 = top row, 0 = bottom row
        nxt[1] = prv[0] + (1 if coins[i] else 0)
        nxt[0] = max(-1 if enemies[i] else prv[0], prv[1])
        prv = nxt
    print(max(nxt[0], nxt[1]))

solve(7, [False, True, True, False, True, False, True],
      [False, False, False, True, False, False, False])

# (base) Ians-MacBook-Air:Desktop iantullis$ python mario.py
# 3
```

Even more space-efficient code (thx Manas!)

```
def solve(length, coins, enemies):
    curr = 0          # the column we are in
    nxt = None        # the column to the right of that
    nxtnxt = None     # the column to the right of THAT
    for i in range(length-1):
        if curr is not None:
            # walk
            if not enemies[i+1]:
                nxt = curr if nxt is None else max(nxt, curr)
            # or jump
            newscore = curr + (1 if coins[i+1] else 0)
            nxtnxt = newscore if nxtnxt is None else max(nxtnxt, newscore)
        # shift to next column
        curr = nxt
        nxt = nxtnxt
        nxtnxt = 0
    print(max(curr, nxt))

solve(7, [False, True, True, False, True, False, True],
      [False, False, False, True, False, False, False]) # answer 3

solve(4, [True, True, True, True],
      [False, False, False, False]) # answer 2

solve(3, [False, True, False],
      [False, False, False]) # answer 1

solve(3, [False, False, True],
      [False, True, False]) # answer 0
```

This eliminates the need for a 2D array – and now only uses 3 values – but is a little harder to understand.

Back to speedrunning

- Can do the same sort of thing with time instead of number of coins.
  - "What's the earliest time we can possibly reach this point in the level?"

- What if we have other stuff like life total, number of hearts, which subweapon...
  - "What's the earliest time we can possibly reach this point in the level, with this much life, this many hearts, this subweapon..." etc. – explosion in complexity, but possible
  - Pretty much the same thing but with a multi-dimensional array of values

amazingly, there have been CS theory papers on the computational complexity of solving the damage-boosting problem...

▶ **Theorem 2.** *DAMAGE BOOSTING is FPT in $k + r$, where k is the number of possible damage values and r the number of chicken events. Moreover, an optimal solution can be found in time $\mathcal{O}(2^r(2k(r+1)+r)^{2.5(2k(r+1)+r)} poly(n))$.*

Proof. Let C be the set of chicken events of S, and suppose $r = |C|$. We simply "guess" which of the 2^r subsets of C to take. That is, for each subset $C' \subseteq C$, we find the maximum time gain achievable under the condition that the chicken events taken are exactly C', hence the 2^r factor in the complexity. For the rest of the proof, assume $C = \{c_0, c_1, \ldots, c_r, c_{r+1}\}$ is a set of chicken events such that $c_i <_S c_{i+1}$ for $0 \leq i \leq r$, each of which must be taken. For notational convenience, we have added chicken $c_0 = c_{r+1} = (0, 0)$, where c_0 (respectively c_{r+1}) is a chicken event that occurs before (resp. after) every event of S.

Corollary 4. *There exist games $\mathscr{G}_1$, $\mathscr{G}_2$, $\mathscr{G}_3$, featuring doors and pressure plates, in which the avatar has to reach an exit location, such that:*

(a) *In $\mathscr{G}_1$, pressure plates can only open doors, crossovers are allowed, and $\mathscr{G}_1$ is **P**-complete.*

(b) *In $\mathscr{G}_2$, no two pressure plates control the same door, and $\mathscr{G}_2$ is **NP**-complete.*

(c) *In $\mathscr{G}_3$, each door may be controlled by two pressure plates, and $\mathscr{G}_3$ is **PSPACE**-complete.*

...and on the hardness of games based on their design elements.

Mechanics	Portals	Long Fall	Complexity
None	No	Yes	P (§3)
Emancipation Grills, No Terminal Velocity	Yes	Yes	Weakly NP-hard (§4)
Turrets	No	Yes	NP-hard (§5)
Timed Door Buttons and Doors	No	No	NP-hard (§6)
HEP Launcher and Catcher	Yes	No	NP-hard (§7)
Cubes, Weighted Buttons, Doors	No	No	PSPACE-comp. (§8)
Lasers, Relays, Moving Platforms	Yes	No	PSPACE-comp. (§8)
Gravity Beams, Cubes, Weighted Buttons, Doors	No	No	PSPACE-comp. (§8)

Table 1: Summary of New Portal Complexity Results

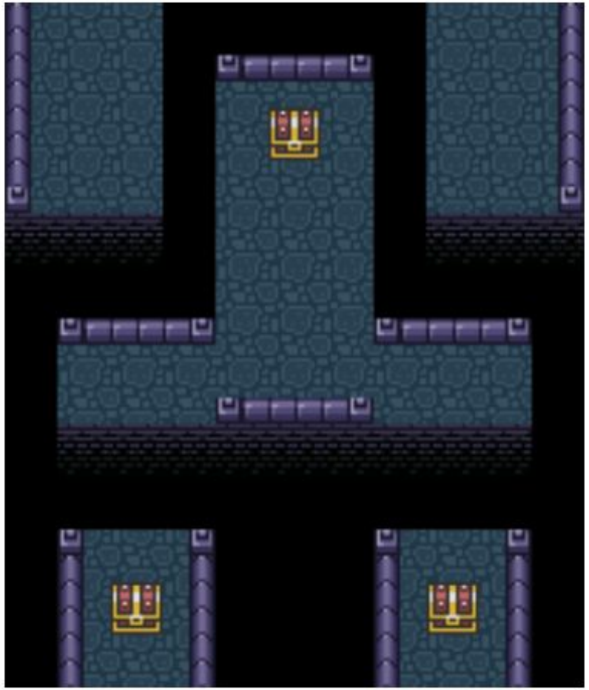


Figure 25: Variable gadget for Zelda

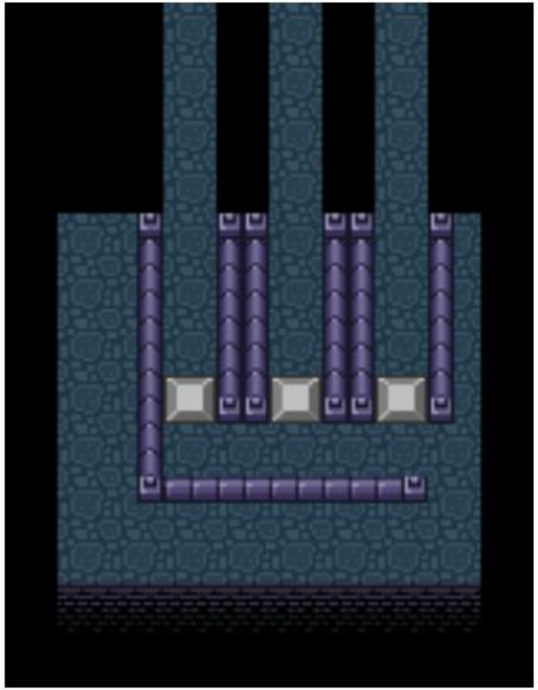


Figure 26: Clause gadget for Zelda

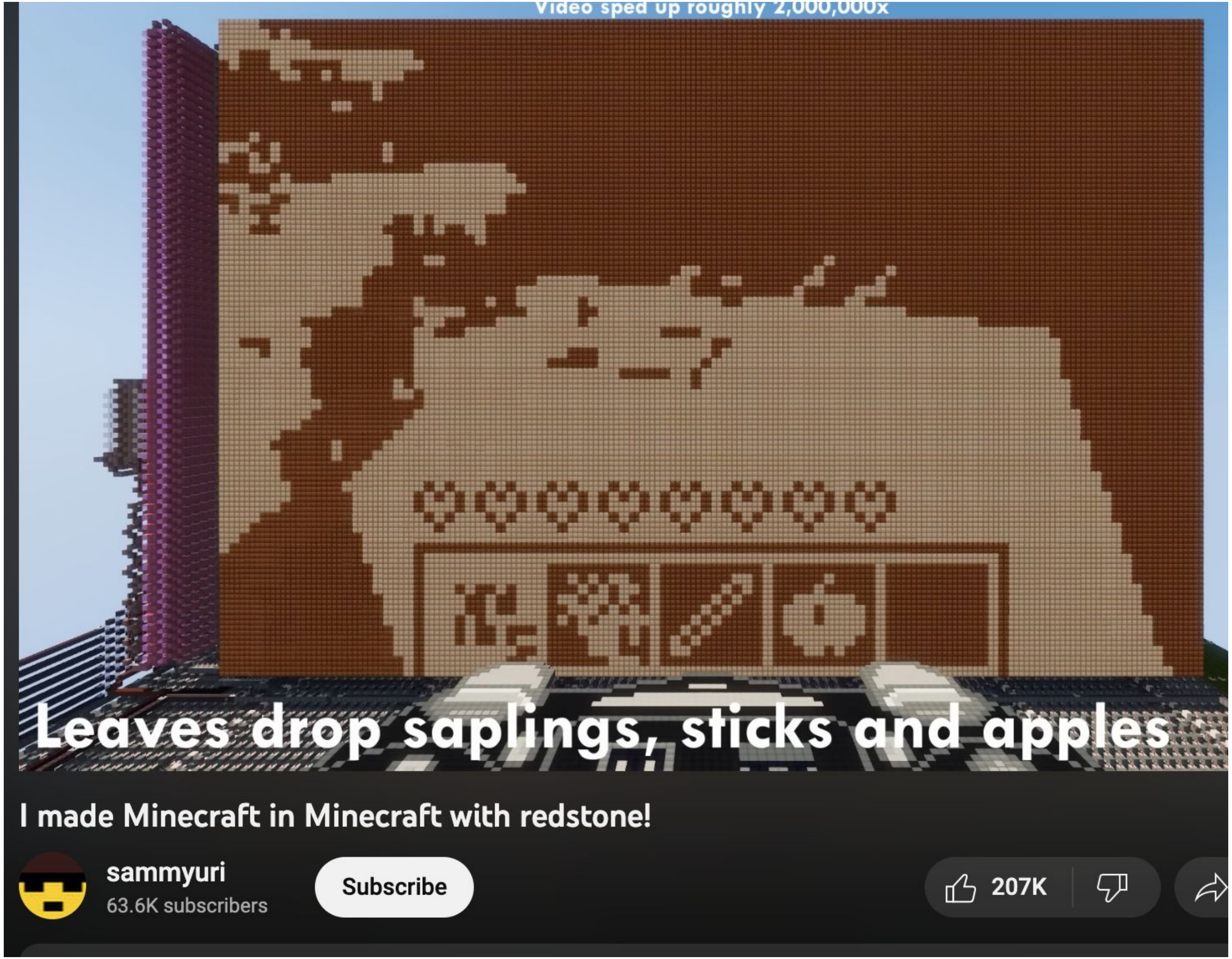
Manipulating randomness

- In old games (and many real-life situations!), a pseudorandom number generator is used to determine random events (e.g., how many bats appear).
- Sometimes you can figure out how the pseudorandom number generator works and reverse-engineer it.

https://www.youtube.com/watch?v=_Y-S-Btd2zU

Performing computation within games

https://www.youtube.com/watch?v=aSlstPplW-E


Video sped up roughly 2,000,000x
Leaves drop saplings, sticks and apples
I made Minecraft in Minecraft with redstone!
sammyuri
63.6K subscribers
Subscribe
207K

Training AIs to play games

- Nice: write a AI that is tailored to be good at one particular game after observing humans (AlphaGo)

Training AIs to play games

- Nice: write a AI that is tailored to be good at one particular game after observing humans (AlphaGo)

- Nicer still: write a AI that is tailored to be good at one particular game even without observing human play (AlphaGo Zero)

Training AIs to play games

- Nice: write a AI that is tailored to be good at one particular game after observing humans (AlphaGo)
- Nicer still: write a AI that is tailored to be good at one particular game even without observing human play (AlphaGo Zero)
- Impressive: write an AI that is good at playing games in general, given the rules (Alpha Zero)

Training AIs to play games

- Nice: write a AI that is tailored to be good at one particular game after observing humans (AlphaGo)

- Nicer still: write a AI that is tailored to be good at one particular game even without observing human play (AlphaGo Zero)

- Impressive: write an AI that is good at playing games in general, given the rules (Alpha Zero)

- Even more impressive: write an AI that is good at playing games in general, *even when it has to infer the rules* (MuZero)

Modern reinforcement learning

giant disclaimer: I have not taken CS234 and am not an RL expert

- In contemporary games, it is not possible for an AI agent to consider and evaluate all possible moves at each state (there could be quajillions of them)

Modern reinforcement learning

giant disclaimer: I have not taken CS234 and am not an RL expert

- In contemporary games, it is not possible for an AI agent to consider and evaluate all possible moves at each state (there could be quajillions of them)

- The tl;dr is that these AIs learn the "landscape" of what moves are good using deep learning
  - which is basically a bunch of linear algebra with nonlinear functions mixed in to allow for more complexity

Modern reinforcement learning

giant disclaimer: I have not taken CS234 and am not an RL expert

- In contemporary games, it is not possible for an AI agent to consider and evaluate all possible moves at each state (there could be quajillions of them)
- The tl;dr is that these AIs learn the "landscape" of what moves are good using deep learning
  - which is basically a bunch of linear algebra with nonlinear functions mixed in to allow for more complexity
- With a sense of this "landscape" in mind, the AIs do variants of dynamic programming
  - and also tune the model parameters in clever ways

Article | Published: 30 October 2019

Grandmaster level in StarCraft II using multi-agent reinforcement learning

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, ... David Silver

+ Show authors

Nature 5[illegible]5, 350–354 (2019) | Cite this article

Abstract

Many real-world applications require artificial agents to compete and coordinate with other agents in complex environments. As a stepping stone to this goal, the domain of StarCraft has emerged as an important challenge for artificial intelligence research, owing to its iconic and enduring status among the most difficult professional esports and its relevance to the real world in terms of its raw complexity and multi-agent challenges. Over the course of a decade and numerous competitions[1,2,3], the strongest agents have simplified important aspects of the game, utilized superhuman capabilities, or employed hand-crafted sub-systems[4]. Despite

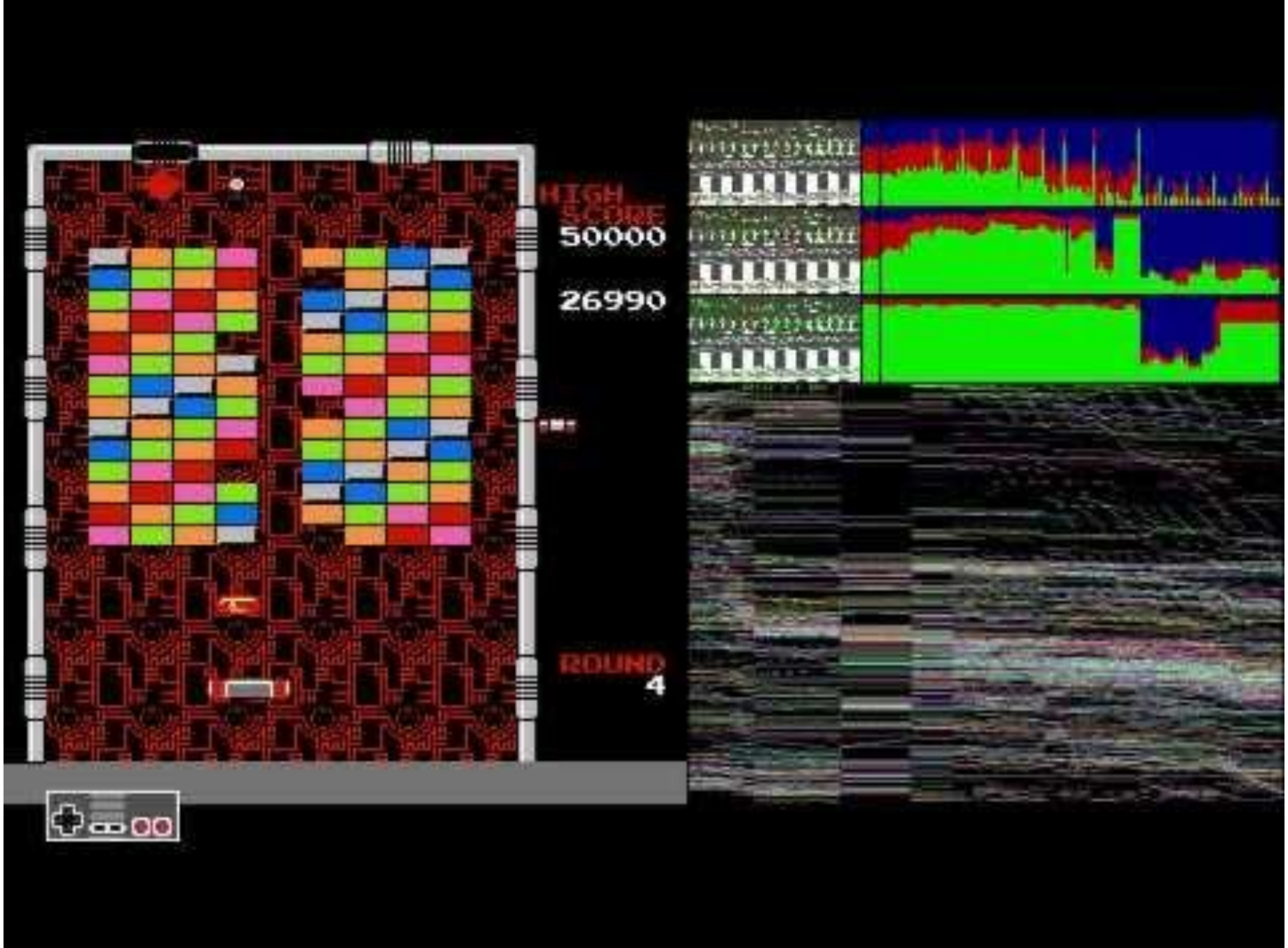
- intentionally gave the agent a limited camera view of the game and limited its movement speed
- got a SC pro to consult
- beat 99.8% of human players on Battle.net

A more digestible example

- **LearnFun/PlayFun** by Tom7 (suckerpinch on YouTube, watch all his stuff!!!!) for a "fun" conference in the early 2010s.
  - key idea: it is usually good when values in the game's memory (score, position in the level...) go up
  - The AI watches a human play for a little bit and then builds its own objective function ("score") based on how values in memory change
  - There are some subtleties (how to identify values in memory that are stored as, e.g., *two* 8-bit numbers)
- Learns to play (general) NES games... with varying degrees of success.

https://ww
Q-WgQcn

https://www.youtube.com/watch?v=qf-tu2ojOb8



https://www.youtube.com/watch?v=Q-WgQcnessA

Takeaways

- What DeepMind and other cutting-edge researchers want is *general* AI / algorithmic solving, and games and puzzles are a useful stepping stone

Takeaways

- What DeepMind and other cutting-edge researchers want is *general* AI / algorithmic solving, and games and puzzles are a useful stepping stone
- Reinforcement learning, driven by neural networks / deep learning, seems to be the the best way we have to tame the combinatorial explosion of how many possible things a game agent could potentially try
    - still a very open problem how to implement this in a general way

Takeaways

- What DeepMind and other cutting-edge researchers want is *general* AI / algorithmic solving, and games and puzzles are a useful stepping stone
- Reinforcement learning, driven by neural networks / deep learning, seems to be the the best way we have to tame the combinatorial explosion of how many possible things a game agent could potentially try
    - still a very open problem how to implement this in a general way
- Let's not forget that games and puzzles are fun and often mathematically beautiful. We each have only so much time on this earth! Joy should be part of our personal objective functions!


MARIO
058700
×20
WORLD
1-4
TIME
200
THANK YOU MARIO!
BUT OUR DEPTH/RIGOR IS IN
ANOTHER CLASS !

- **Theory**
    - CS **154** (Computational Complexity), **254**, **254B**
    - CS **151** (Logic Programming), CS **157** (Logic)
    - CS **161** (Algorithms), CS **168** (Modern Algorithmic Toolbox)
    - CS **164**??? someday? (expanded version of this class)
    - CS **250** (Error-correcting codes)
    - CS **269I** (Incentives in CS) – game theory
    - Econ and MS&E have a bunch of classes on game theory
- **Math**
    - Math **61DM**, **62DM**, **63DM** – discrete math
    - Math **107** (Graph Theory), **108** (Combinatorics)
    - Math **109/120** (Abstract Algebra), **104/113** (Linear Algebra)
    - Math **193** (Polya Problem Solving Seminar)
    - Math **231** (Math/Stats of Gambling) or anything with Persi Diaconis

- **AI**
    - CS **221** (Intro to AI) – has a really fun Pac-Man project
    - CS **227B** (General Game Playing)
    - CS **229** (Machine Learning) – warning, eats your life, don't take it first
    - CS **230** (Deep Learning), CS **234** (Reinforcement Learning), CS **224R** (Deep Reinforcement Learning), CS **332** (Advanced RL)
    - CS **238** (Decision Making Under Uncertainty) - take this and/or 221 first?
    - lots of others, I'm sure (e.g., vision, robotics...)
- **Design** - I know nothing about these but design is important
    - CS **146** (Intro to Game Design)
    - CS **247G** (Design for Play)
    - CS **377G** (Designing Serious Games)

Please do the feedback form when it comes out

Although my time at Stanford is ending soon, even after I'm down in San Diego I may try to continue remotely teaching this class or a more rigorous variant, CS164. So your thoughts would be very helpful!

Thank you for taking this class!