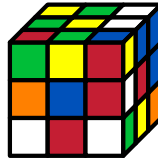


## Lesson 7. Big DPs and the Curse of Dimensionality

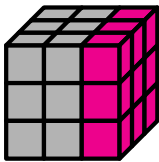
### 1 Solving a Rubik's cube

- In a classic Rubik's cube, each of the 6 faces is covered by 9 stickers
- Each sticker can be one of 6 colors: white, red, blue, orange, green and yellow

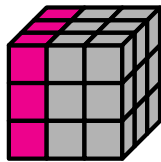


- Each face of the cube can be turned independently

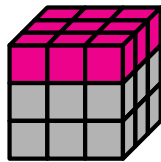
- Notation:



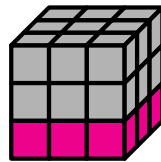
R



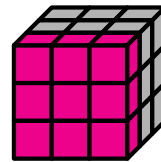
L



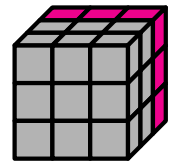
U



D



F



B

- The letter means turn the face clockwise  $90^\circ$ 
  - ◊ For example, **R** means turn the right face clockwise  $90^\circ$
- The letter primed means turn the face counter-clockwise  $90^\circ$ 
  - ◊ For example, **R'** means turn the right face counter-clockwise  $90^\circ$
- The problem: given an initial configuration of the cube, find a *shortest* sequence of turns so that each face has only one color
  - You may assume that you are allowed at most  $T$  turns
  - It turns out that any configuration can be solved in 26 turns or less: <http://cube20.org/qtm/>
- How can we formulate this problem as a dynamic program?

- Stages:

- States in stage  $t$  (nodes):

- Decisions, transitions, and rewards/costs at stage  $t$  (edges):

- Source node:

- Sink node:

- Shortest/longest path?

- Minimum number of turns required to solve the cube:

- Actual sequence of turns that give the minimum number of turns to solve the cube:

## 2 Tetris

- You've all played Tetris before, right? Just in case...
- Tetris is a video game in which pieces fall down a 2D playing field, like this:



- Each piece is made up of four equally-sized bricks, and the playing field is 10 bricks wide and 20 bricks high
- As the pieces fall, the player can rotate them 90° in either direction, or move them left and right
- When a row is constructed without any holes, the player receives a point and the corresponding row is cleared
- The game is over once the height of bricks exceeds 20
- The problem: given a predetermined sequence of  $T$  pieces<sup>1</sup>, determine how to place each piece in order to maximize the number of points accumulated over the course of the game
- How can we formulate this problem as a dynamic program?

---

<sup>1</sup>Normally, the sequence of falling pieces is random and infinitely long. We'll consider this easier version here.

- Stages:

- States in stage  $t$  (nodes):

- Decisions, transitions, and rewards/costs at stage  $t$  (edges):

- Source node:

- Sink node:

- Shortest/longest path?

- Maximum number of points:

- Actual placement of pieces that give the maximum number of points:

### 3 Big DPs and the curse of dimensionality

- How big are these DPs we just formulated?

- Tetris:

- Number of states per stage:

- Number of stages  $T$

⇒ Number of nodes:

- Rubik's cube:

- Number of states per stage:

- Number of stages  $T$

⇒ Number of nodes:

- The number of states is huge for both these DPs!

⇒ The DPs we formulated (as-is) are not solvable using today's computing power

- This is known as **the curse of dimensionality** in dynamic programming

- **Approximate dynamic programming** is an active area of research that tries to address the curse of dimensionality in various ways

- For example, for Tetris: <https://papers.nips.cc/paper/5190-approximate-dynamic-programming-finally-performs-well-in-the-game-of-tetris.pdf>