# CS 2112 / ENGRD 2112 Object-Oriented Design and Data Structures — Honors

Fall 2024

Cornell University

## Course staff

Instructor: Andrew Myers

TAs and Consultants:

Jonathan Gabor

Michael Xing

Sean Zhang

Noah Schiff

Jake Silver

James Zhang

## Goal

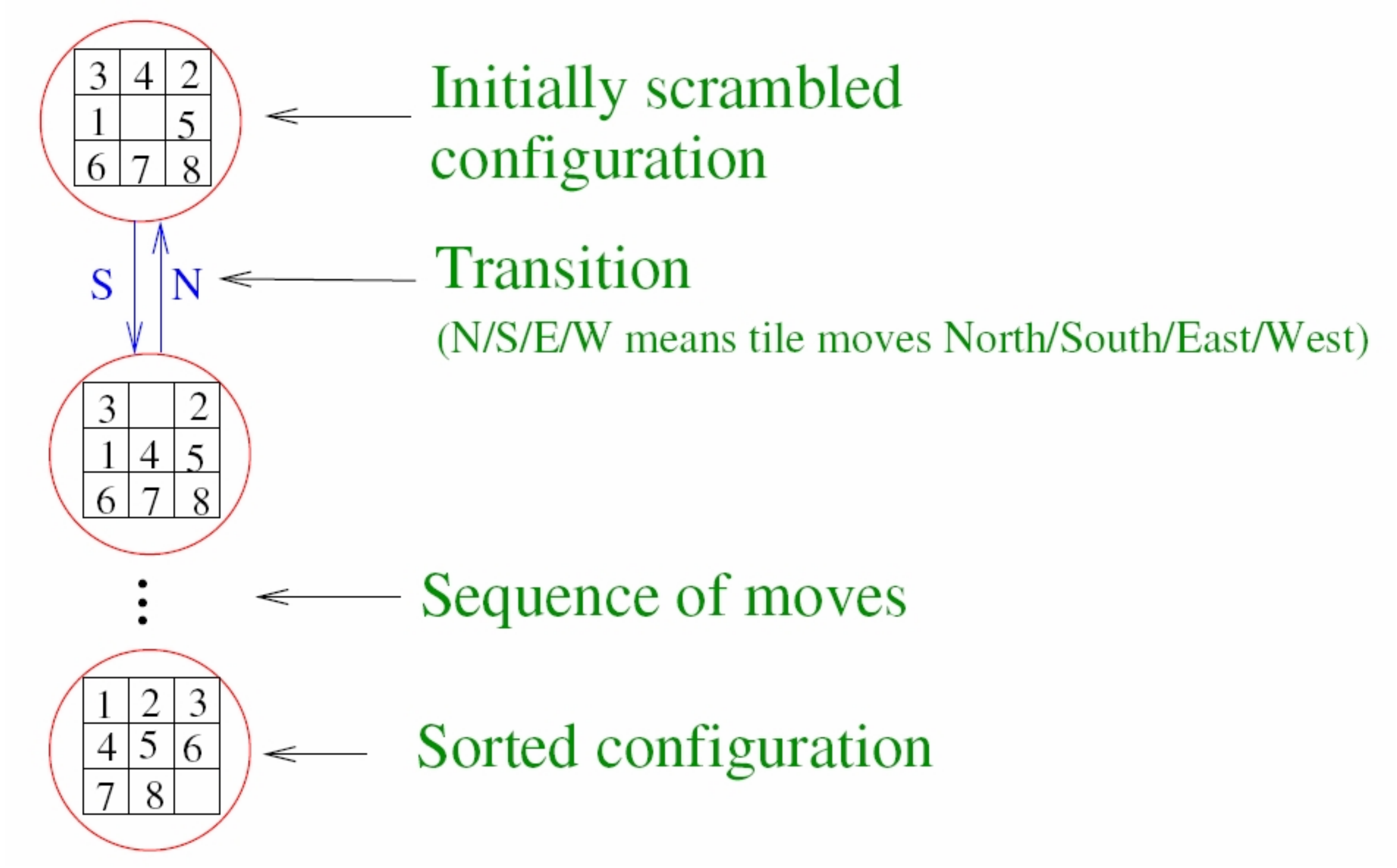
- Tools and confidence to take creative ideas and turn them into working code.

# Content

Introduction to computer science and software engineering

- **Programming language features**
    - –data abstraction, subtyping, generic programming
    - –concurrency and threads
    - –Not a course *about* Java, but you will become comfortable with Java
- **Object-oriented design — organizing large programs**
    - –specifications
    - –design patterns
    - –frameworks and event-driven programming
- **Data structures and algorithms**
    - –recursive algorithms and data structures
    - –reasoning about algorithm correctness and efficiency
        - –induction, asymptotic complexity
    - –arrays, lists, stacks, queues, trees, graphs, hash tables, and associated algorithms

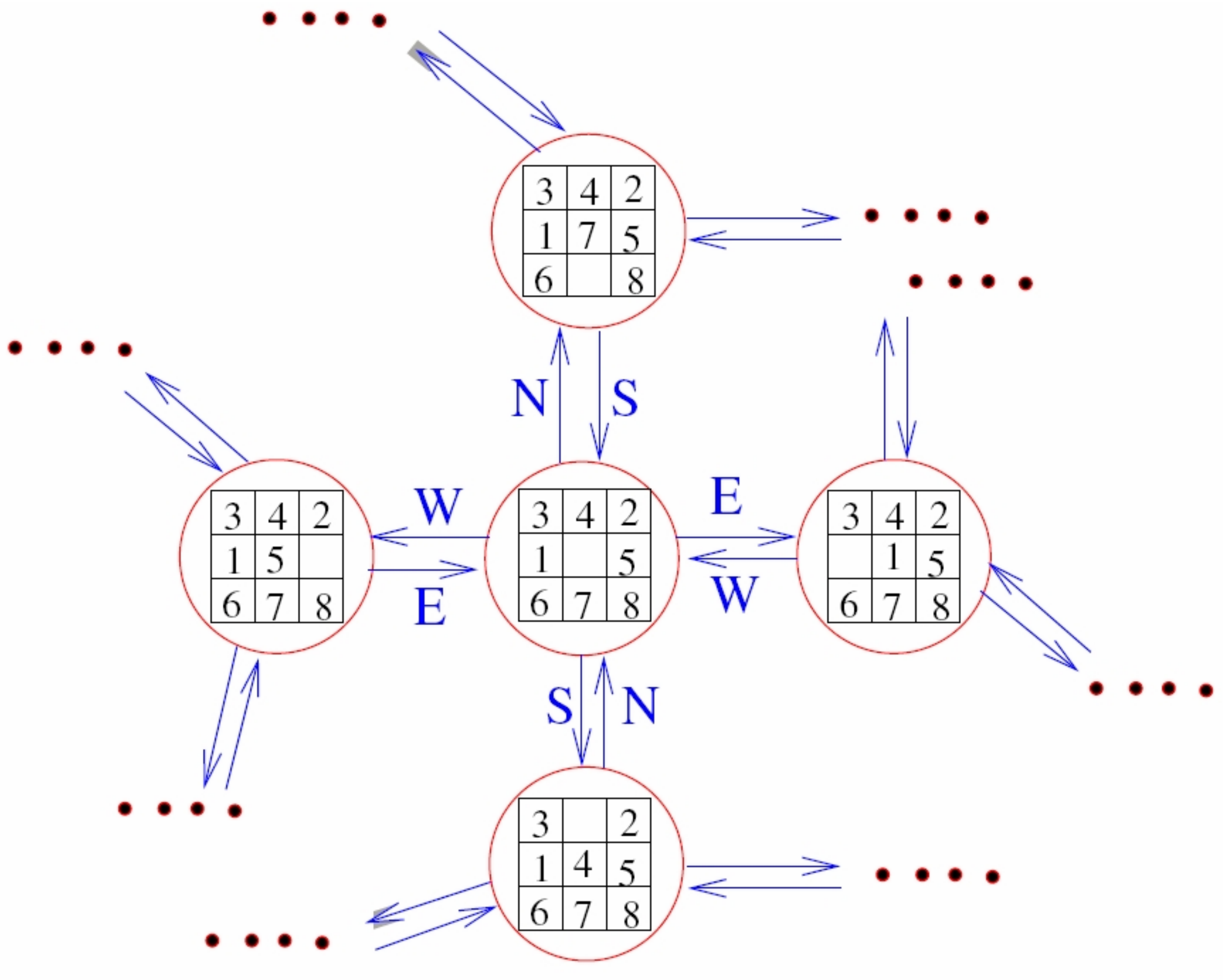
# Sam Loyd's 8 Puzzle



Goal: Given an initial configuration of tiles, find a sequence of moves that will lead to the sorted configuration.

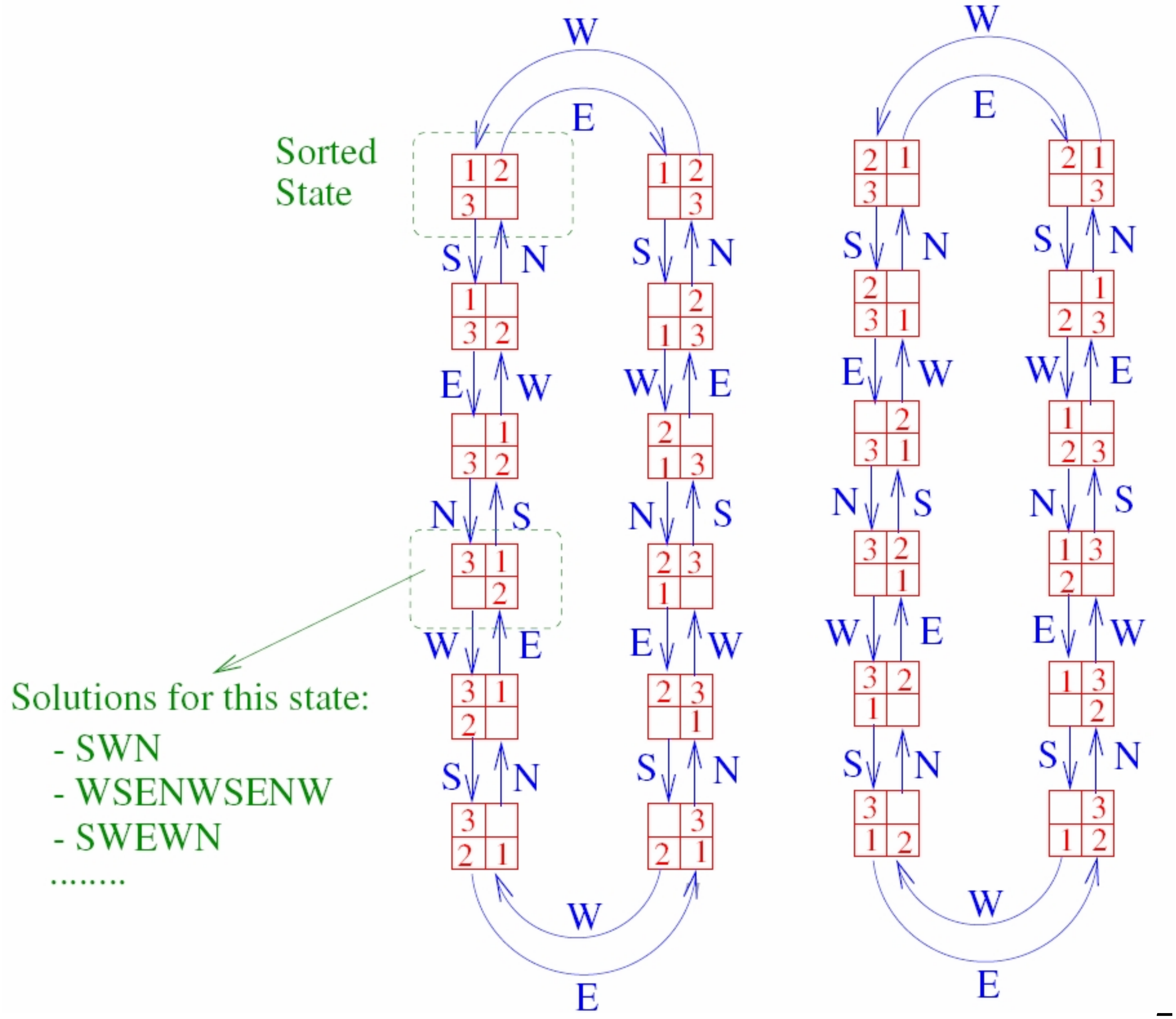
A particular configuration is called a ***state*** of the puzzle.

# State Transition Diagram of 8-Puzzle



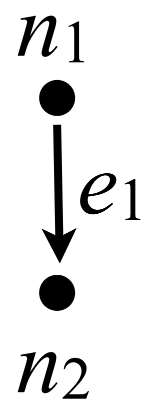
***State Transition Diagram***: picture of adjacent states.
A state Y is ***adjacent*** to state X if Y can be reached from X in one move.

# State Transition Diagram for a 2x2 Puzzle



## Graphs

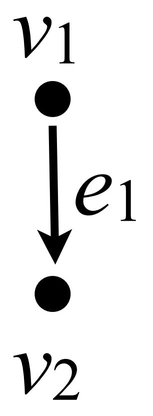
- State transition diagram in previous slide is an an example of a graph: a mathematical abstraction
  - –nodes (or vertices) : the puzzle states
  - –edges (or arcs) : the transitions, possibly labeled



- Graphs are all around us : airline routes, roadmaps, org charts, pipelines, family trees, …

# Graph algorithms

- Large toolbox of efficient algorithms for graphs help us solve problems:
  - searching for best nodes/shortest paths
  - finding maximum flow through graph
  - minimum spanning trees
  - ...
- And known hardness results (e.g., finding Hamiltonian cycles) tell you what you can't solve.



## Software design choices

- What operations should puzzle objects have?
- How do we represent states? The initial state? Transitions?
- How do we present information to the user and support interaction?
- How do we break the coding up into parts that can be coded independently?
- How to structure code so it can be maintained, upgraded?

# Why you need CS 2112

- Data structures and algorithms to solve problems efficiently and effectively
- Design techniques to produce code that works quickly and keeps working
- Computer science:
  - –algorithms, data structures, programming languages, design principles, knowledge of what is possible and feasible
- Good programmers have more fun!
  - –10x more productive
  - –better able to adapt, grow, see opportunities, change the world

## CS 2112 or ENGRD 2112?

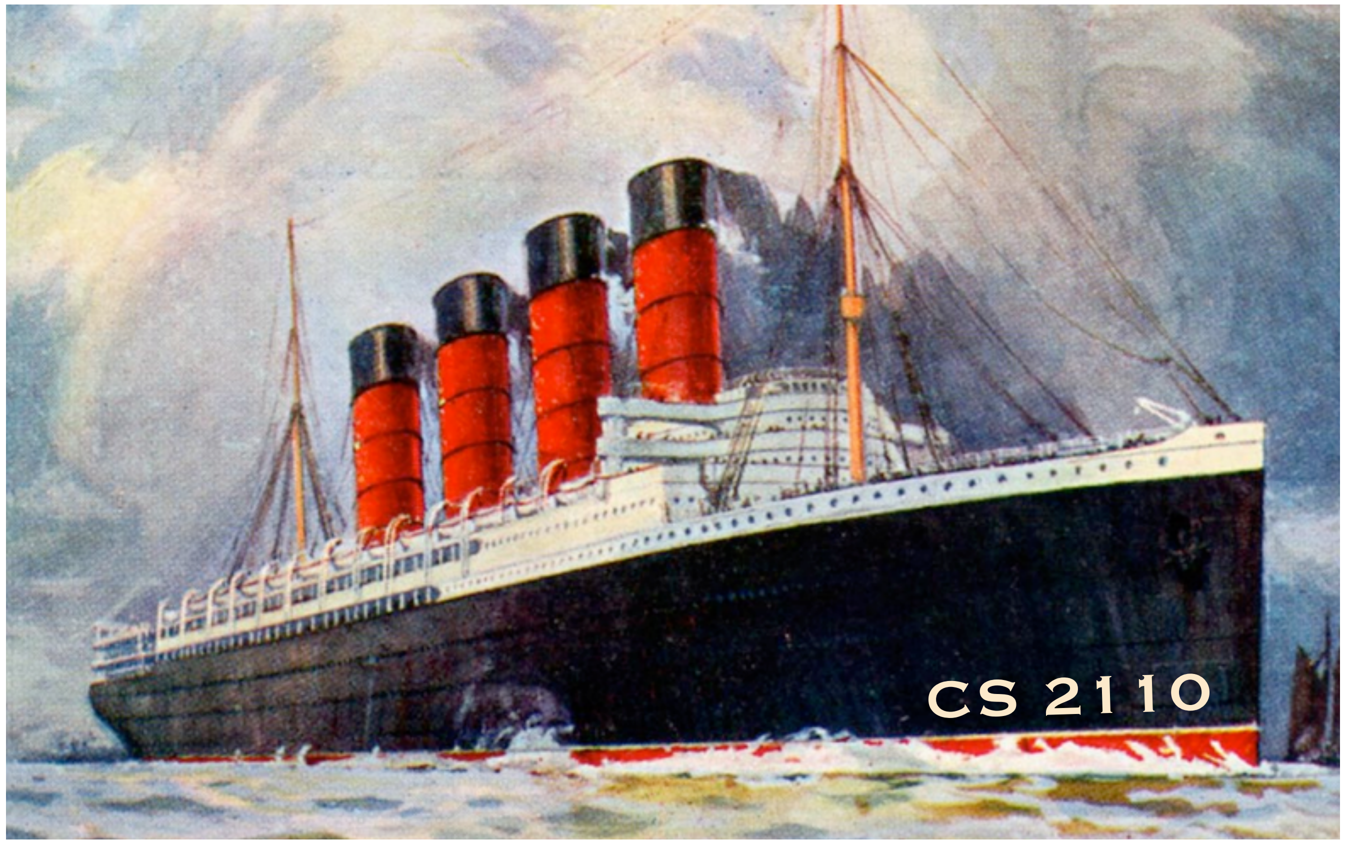
Doesn’t matter

# CS/ENGRD 2110 or CS/ENGRD 2112?

- 2112 is an honors version of 2110
  - aimed at CS majors
  - much smaller (<100 vs. ~600)
  - more material
    - more algorithms and their analysis (theory)
    - more about design and design patterns (practice)
  - more difficult and more interesting assignments, with more programming and building code from scratch.
  - cool final project spanning 3 assignments and a final tournament.
  - more credits (4 vs 3)

## 2110 vs 2112

Warning: you will be challenged here



It's fine to switch in either direction during the first three weeks.

## Web site

- Your best source for information:

  http://courses.cs.cornell.edu/cs2112

  – Canvas links here
  – Lecture notes: you are expected to read
    - may not include *everything* covered in lecture
    - may include extra material *not* covered in lecture
    - often updated after the lecture based on questions
  – Assignments
    - may be updated (w/ datestamp) after initial release
  – Pointers to resources

## Communicating with staff

- Best: Ed Discussions
  - Answering other questions (well) counts as participation
  - Watch out for violations of academic integrity!
- Course announcements ☛ Ed (email if urgent)
- See website for office hours, Zoom link
  - Front line for answering questions – consulting hours start next week

## CMS(X)

- `https://cmsx.cs.cornell.edu/`
- Assignments ☛ CMS
- Grades, solutions ☛ CMS
- Regrade requests ☛ CMS soon after receiving grade (remind us if necessary...)
- You should be registered if you are here

## Meetings

- **Lectures**: TR 10:10–11am, Baker Lab 200
- **Discussions** (attend **one** per week, as assigned)
  - T 12:25–1:15, Gates Hall 114
  - W 1:25–2:15, Bard Hall 140
- **Labs** (attend **one** per week)
  - M 7:30–8:20, Gates 114
  - W 7:30–8:20, Gates 114
- attendance is expected on days you are assigned to

## Assignments

- 6 assignments
  - –mostly programming but some written problems
  - –45% of total score
- First assignment done alone
- Second and third alone or with one partner
- Final project (last 3 assignments) with one or two partners
- Late submissions incur penalty per day: 5%, 15%, 25%, …

## Exams

- Two in-person exams
- Worth 52% of total score (20%, 32%)

## Labs

- Programming exercises, solve problems, learn about tools that are helpful for assignments

- First lab this week - IntelliJ demo and working with Java I/O

## Course Reading

Object-Oriented Design and Data Structures
https://andrewcmyers.github.io/oodds/toc.html

Chapters linked from the course website

## Other Reading

Data Structures and Abstractions with Java, 4th ed., Frank M. Carrano and Timothy Henry, Pearson Education, 2014

- Available at Campus Store
- On reserve in library
- Recommended, not required
- Not heavily used
- Earlier editions are mostly ok

## Software

- Java 21 (current LTS version of Java)
- IntelliJ IDEA is the supported IDE
  - You may use Eclipse, etc.

## CS 2110 Java HyperText

- http://www.cs.cornell.edu/courses/JavaAndDS/
- David Gries's online text for OO programming in Java and data structures
- excellent resource esp. for students with little Java experience
- searchable glossary of terms
- information on Java, Eclipse, data structures, code style, program correctness, recursion, …
- short tutorials, pdf files, videos

## Academic integrity

- You must never misrepresent someone else's work as your own or let others misrepresent your work as theirs
  - Copying code or answers is never okay
  - Aiding others' AI violations is also a violation
    ⇒ Letting others copy you is also a violation
  - You must be able to explain your answers fully
  - Discussions with others are perfectly fine if they could have happened in a lightless room
- We use effective tools for detecting plagiarism
- Report any discussions about assignments and any use of external code
- Our goal: spend time on course content

## Social integrity

Everyone is to be treated with **respect**, regardless of background, experience, religion, ethnicity, citizenship, gender identity, or sexual orientation.

If you are made to feel unwelcome or disrespected, please contact me.

If you become aware of anyone else being made to feel unwelcome or disrespected, it is good to speak up! Also encourage them to contact course staff.

## GAI policy

- Do not use for generating code
- May use as a second pass to fix grammar
- May use to generate test cases

## Next steps

- Bookmark and keep an eye on the 2112 website
- Download the first programming assignment, released soon
- Make sure you have IntelliJ downloaded and working — see course staff for help
- Attend lab M/W for help getting started
- Have fun!