CS 2110 / ENGRD 2110 Object-Oriented Programming and Data Structures

Fall 2022

Cornell University

Instructors:

Prof. Myers

Dr. Muhlberger

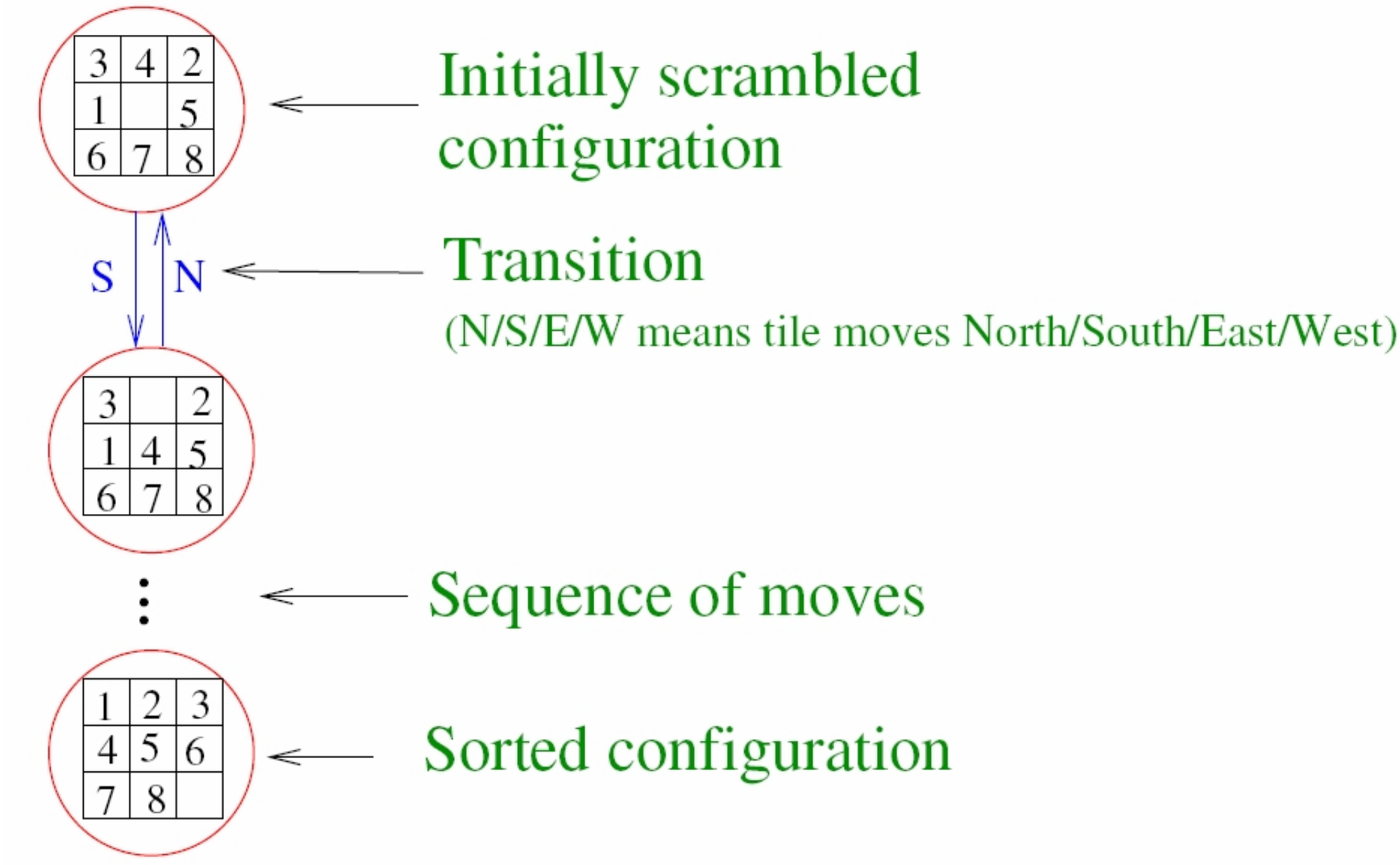
This class is 100% full — please fill in every seat!

What it's about

Introduction to computer science and software engineering

- **Programming language features and programming models**
  - –data abstraction, subtyping, generic programming
  - –concurrency and event-driven programming
  - –Not a course *about* Java, but you will become comfortable with Java
- **Data structures and algorithms**
  - –recursive algorithms and data structures:
    - –arrays, lists, stacks, queues, trees, graphs, hash tables, and associated algorithms
  - –reasoning about algorithm correctness and efficiency
    - –specifications and invariants
    - –induction, asymptotic complexity

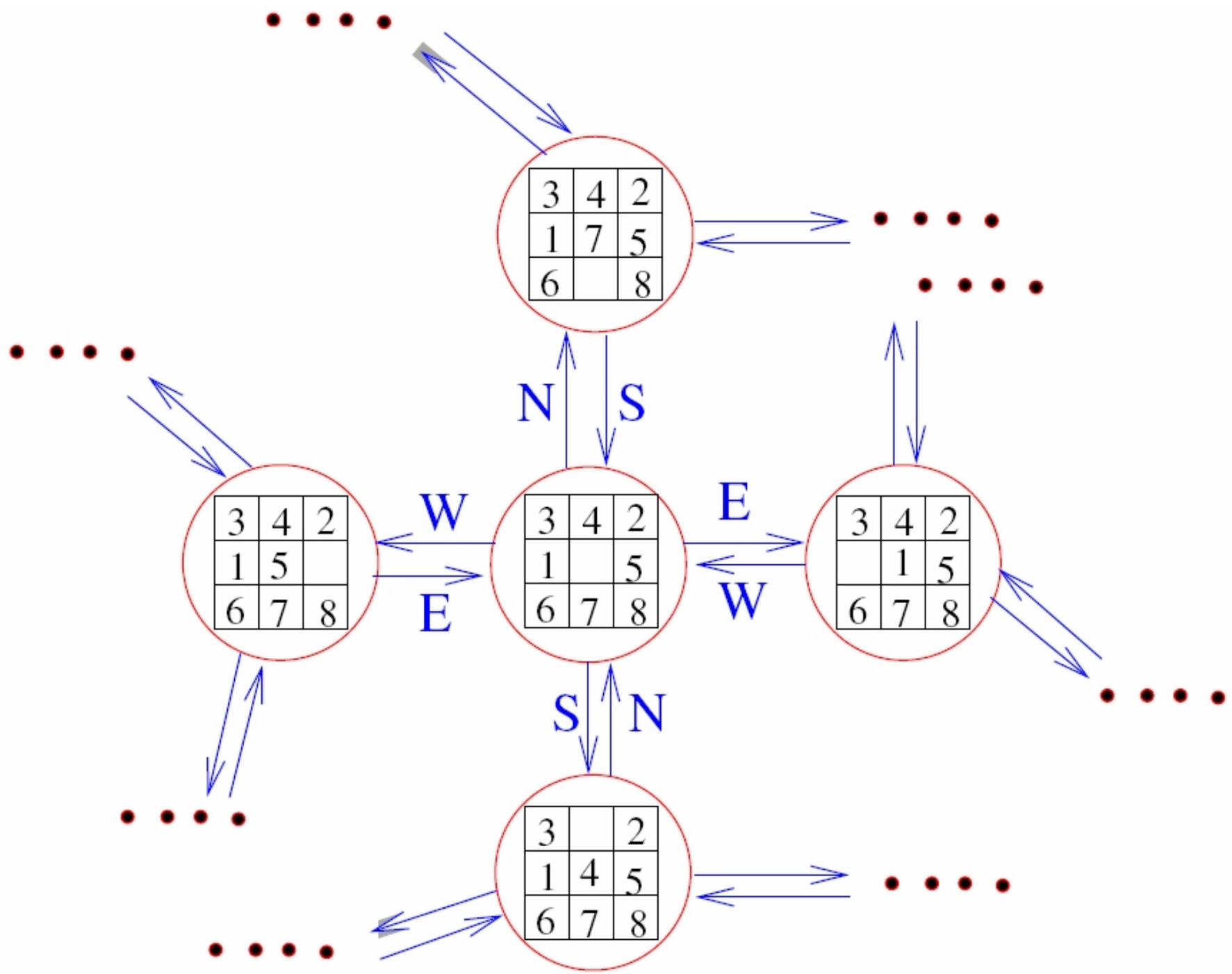
Sam Loyd's 8 Puzzle



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.

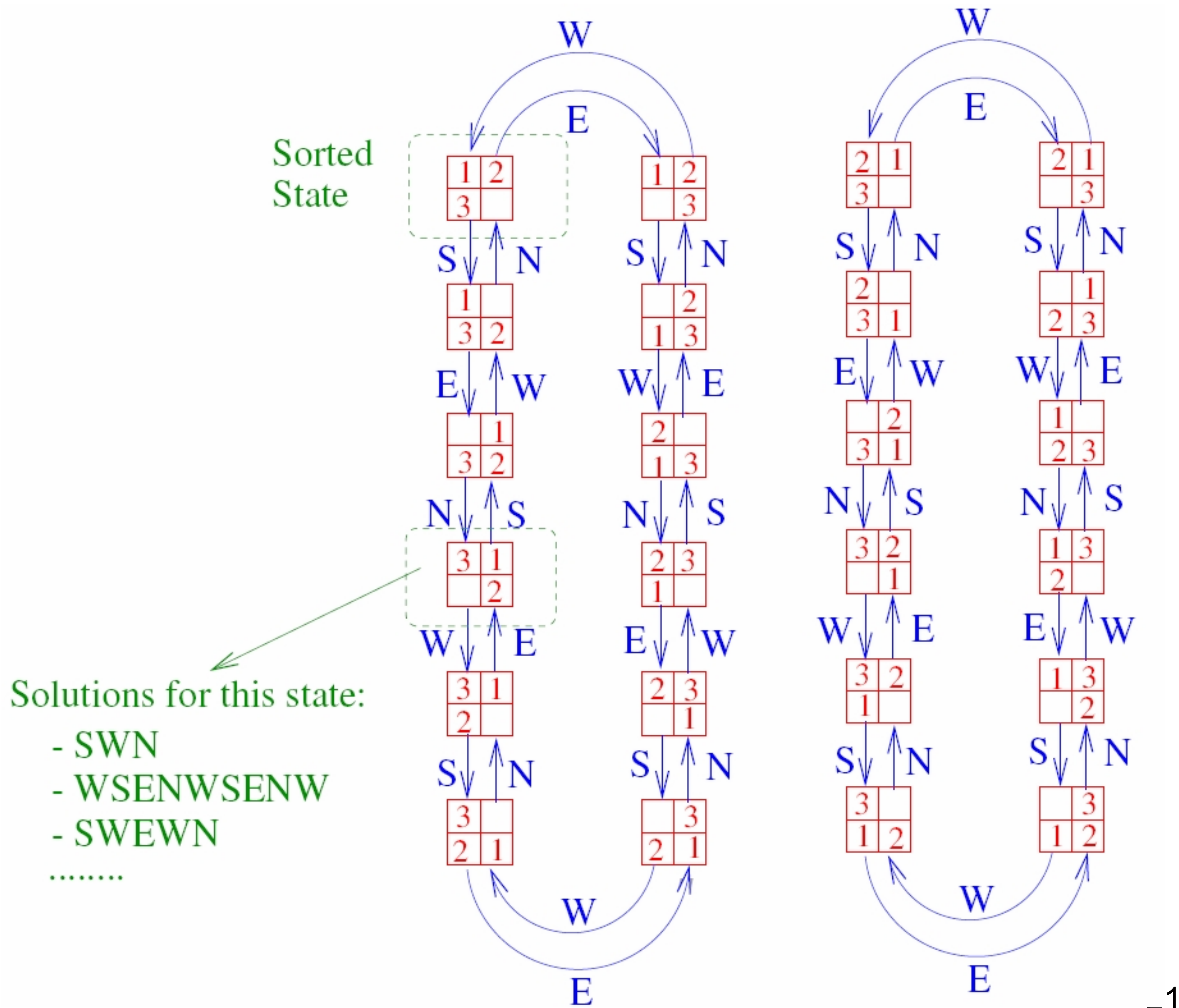
State Transition Diagram of 8-Puzzle



State Transition Diagram: picture of adjacent states.

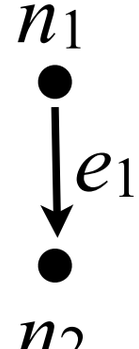
A state Y is ***adjacent*** to state X if Y can be reached from X in one move.

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) : puzzle states
  - –edges (or arcs) : 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
  - –maximizing flows
  - –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 implemented by a team?
- How to structure code so it can be maintained, upgraded?

Why you need CS 2110

- Programming methods to quickly produce code that works and keeps working
- Data structures and algorithms to solve problems efficiently and effectively