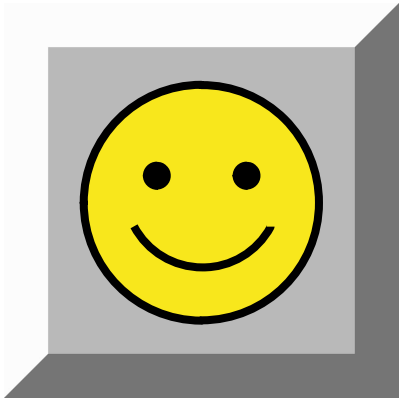
Minesweeper CSP

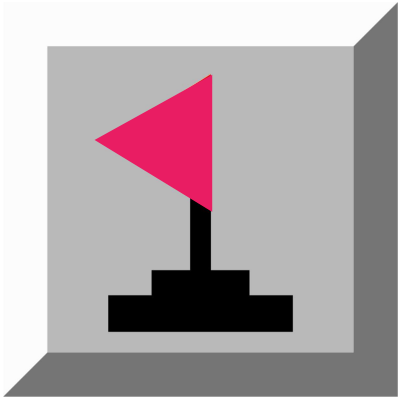
Nahom Amanuel, Irene Han, Andrew Hyssop Cha, Mia Ndousse-Fetter,
Karthik Sellakumaran Latha

Objectives

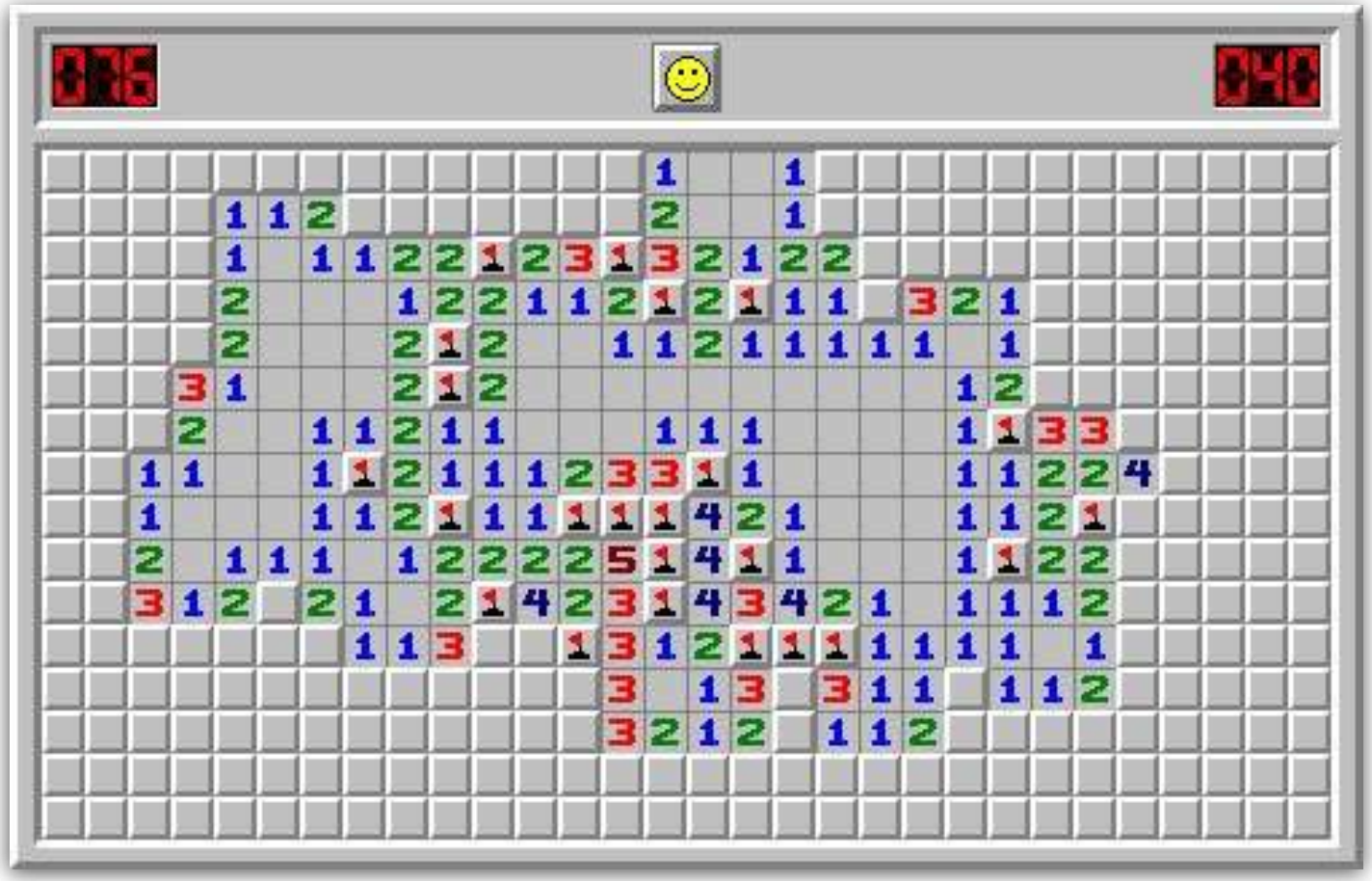


- Design CSP solver algorithm
- Given minesweeper board, use CSP to solve
  - Should output location of:
    - Mines to flag
    - Safe tiles to uncover
- Design minesweeper simulation to test CSP solver
- Add UI, allowing player to access CSP solver through minesweeper simulation

Minesweeper Logistics



- A tile on the board can be:
  - Hidden: untouched by player
  - Empty: no surrounding mines
  - Integer from 1-8: # of surrounding mines
  - Mine: ends game when uncovered
- Can uncover any hidden tile until end of game
- Can mark any hidden tile with a flag
- The goal is to uncover all tiles other than the mines

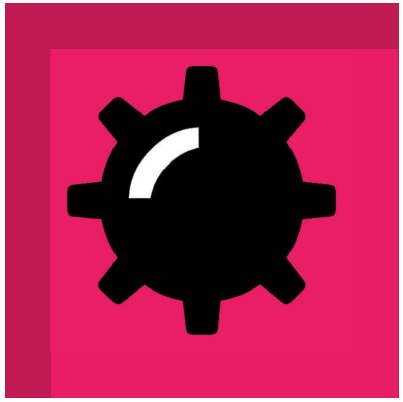


Constraint Satisfaction in Minesweeper

How we formulate as a CSP:

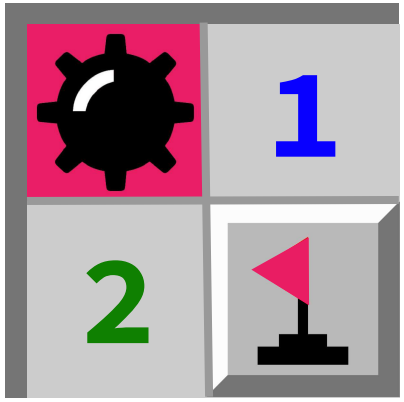
- Create constraint graph
  - Constraints are known tiles
  - Variables are hidden tiles surrounding constraints
  - Find list of variables for each constraint
  - Link constraint to associated variables
- Solve all trivial constraints and simplify
- Run backtracking search

Backtracking for Minesweeper CSP



- Recursive function arguments:
  - Solutions
  - List of variable values
  - Current variables
  - Current constraints
- Stops when all variables have been solved
- Marks tiles that could have mines/no mines
- Adds to list of variable values

Minesweeper User Interface



Basic info on UI:

- Visuals generated by wx python
- Board is shown after player makes move
- Solution board is shown when CSP solver is run
- Solver available as “cheat button”, does not affect game
- Player can “cheat” and uncover tiles until end of game

Visual Examples of UI in Action