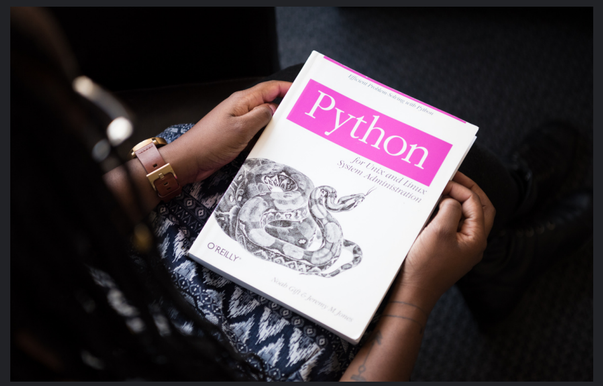
COMP 3981 AI

By Manjot Randhawa, Brytton Tsai,
Jimmy Wong, Nikolay Rozanov

Agenda

1. Introduction
2. Design
3. Brute Force Algorithm
4. CSP Algorithm
5. Conclusion

TOOLS



PYTHON

- Used for the front-end logic
- Used for the algorithm logic

Makefile
The original build tool

MAKEFILE + BASH

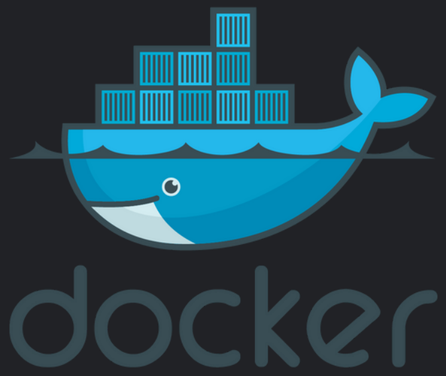
- Automating Builds
- Development Consistency

aws

AWS

- Offers the option to run the application on the cloud
- Provides a reliable and secure hosting environment

TOOLS



DOCKER

- To containerize our app before AWS deployment

PYSIDE6 + QT

- Used for the cross-platform GUI of the app
- Handles the event loop
- QML integrated development environment which simplifies GUI application development

TERRAFORM

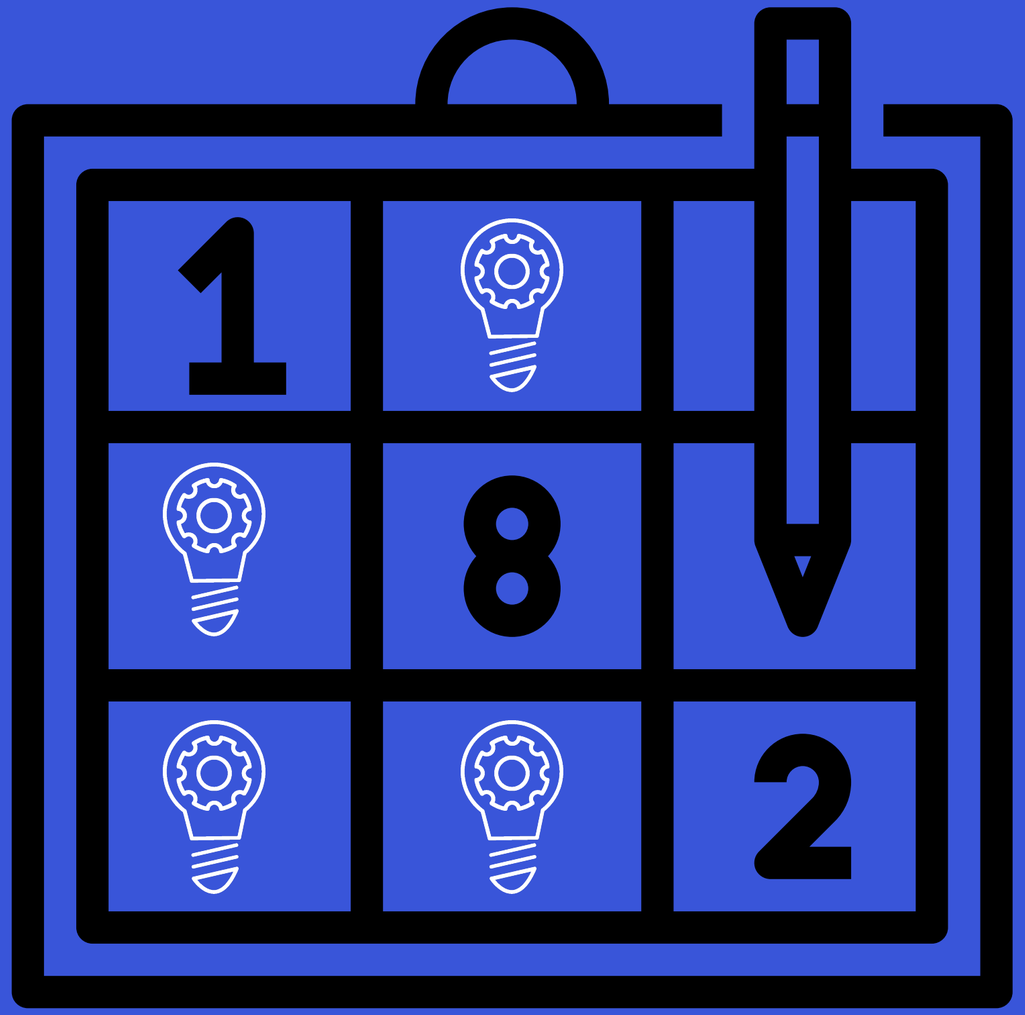
- Infrastructure as code tool that lets you build cloud resources safely and efficiently

INFRASTRUCTURE OVERVIEW

Too big to fit on the slides

SUDOKU GENERATION

- Script-based random generation
- File-based random generation
- Pre-made puzzle loading



Brute Force Algorithm

- Finding Empty Cells
- Finding Possible Values
- Random Selection
- Recursion
- Timeout / Retry

Finding Empty Cells

- Iterating through a two dimensional array
- Check if values that are equal to 0
- If cell has value equal to 0 means cell is empty

```
[[3, 7, 5, 8, 2, 4, 6, 1, 0],
 [8, 0, 6, 1, 7, 9, 2, 5, 3],
 [9, 2, 1, 5, 6, 3, 8, 4, 7],
 [4, 5, 9, 2, 1, 8, 3, 7, 6],
 [6, 3, 2, 9, 5, 7, 1, 8, 4],
 [1, 8, 7, 3, 4, 6, 5, 9, 2],
 [2, 6, 4, 7, 8, 1, 9, 3, 5],
 [5, 9, 8, 4, 3, 2, 7, 6, 1],
 [7, 1, 3, 6, 9, 5, 4, 2, 8]]
```

Finding Possible Values

- Given column and row check possible values
- Check all values in the same row
- Check all values in the same column
- Check all values in the same grid
- Determine list of possible values

```
[[3, 7, 5, 8, 2, 4, 6, 1, 0],
 [8, 0, 6, 0, 7, 9, 2, 5, 0],
 [9, 2, 0, 5, 6, 3, 8, 4, 7],
 [4, 5, 9, 2, 1, 8, 3, 7, 6],
 [6, 3, 2, 9, 5, 7, 1, 8, 4],
 [1, 8, 7, 3, 4, 6, 5, 9, 2],
 [2, 6, 4, 7, 8, 1, 9, 3, 5],
 [5, 9, 8, 4, 3, 2, 7, 6, 1],
 [7, 0, 3, 6, 9, 5, 4, 2, 8]]
```

Random Selection

- Randomly select a value from the possible set of values
- Shuffle the list of possible values
- Then select the first element
- Example: [1, 3, 4]

[[0, 7, 5, 8, 2, 4, 6, 1, 0],
[8, 0, 6, 0, 7, 9, 2, 5, 0],
[9, 2, 0, 5, 6, 3, 8, 4, 7],
[4, 5, 9, 2, 1, 8, 3, 7, 6],
[6, 0, 2, 9, 5, 7, 1, 8, 4],
[1, 8, 7, 3, 4, 6, 5, 9, 2],
[2, 6, 4, 7, 8, 1, 9, 3, 5],
[5, 9, 8, 4, 3, 2, 7, 6, 1],
[7, 0, 3, 6, 9, 5, 4, 2, 8]]

Recursion

- Recursively call the last 3 steps until no empty cells are found

[[3, 7, 5, 8, 2, 4, 6, 1, 9],
[8, 4, 6, 1, 7, 9, 2, 5, 3],
[9, 2, 1, 5, 6, 3, 8, 4, 7],
[4, 5, 9, 2, 1, 8, 3, 7, 6],
[6, 3, 2, 9, 5, 7, 1, 8, 4],
[1, 8, 7, 3, 4, 6, 5, 9, 2],
[2, 6, 4, 7, 8, 1, 9, 3, 5],
[5, 9, 8, 4, 3, 2, 7, 6, 1],
[7, 1, 3, 6, 9, 5, 4, 2, 8]]

Timeout / Retry

- Issues with backtracking and recursive calls
- Setting a timeout for each solve attempt
- Setting a retry attempt of 15 times

Board Size	Time (Seconds
9x9	5
12x12	15
16x16	16
25x25	30

Statistics

	Average time (seconds)	Standard deviation	Success
9x9	0.06772525	0.19890018	100%
12x12	3.5306933	6.14235701	100%
16x16	19.2738021	20.8148636	100%
25x25	144.814612	67.8336156	60%

Brute Force Design Choices

- Choosing Cells With Least Possible Value
- Shuffling Array To Increase Randomness
- Added Timeout / Retry

CSP Algorithm

- Backtracking Search
- Multiprocessing
- Iteration & Timeout

CSP Backtracking Preparation

Identify Empty Cells

Find All Arcs

Find Empty Cells Domains

CSP Backtracking Search

Select Variable	MRV and Degree heuristic
Order Values for Assignment	Least-constraining-value heuristic
Inference	MAC

CSP Optimization

Code Optimization

```python
class CSPSolver(SudokuSolver):
    def __init__(self):
        self._board = []
        self._size = 0
        self._empty_cells = set()
        self._domains = {}
        self._empty_cells_in_rows = {}
        self._empty_cells_in_cols = {}
        self._empty_cells_in_grids = {}
        self._related_cells = {}
        self._assignment = {}
        self._start_time = None
        self._terminate_time = None
```

```python
def _revise(self, cell_i, cell_j):
    """
    part of ac-3, revise domains base on constraints
    """
    i_domains = self._domains[cell_i]
    j_domains = self._domains[cell_j]
    domain_changed = False
    if len(j_domains) == 1 and len(i_domains) == 1:
        i_domain = next(iter(i_domains))
        if i_domain in j_domains:
            i_domains.clear()
            domain_changed = True
    elif not j_domains:
        j_assignment = self._assignment[cell_j]
        if j_assignment in i_domains:
            i_domains.remove(j_assignment)
            domain_changed = True
    unique_candidate_updated = self._update_unique_candidate(cell_i)
    return domain_changed or unique_candidate_updated
```

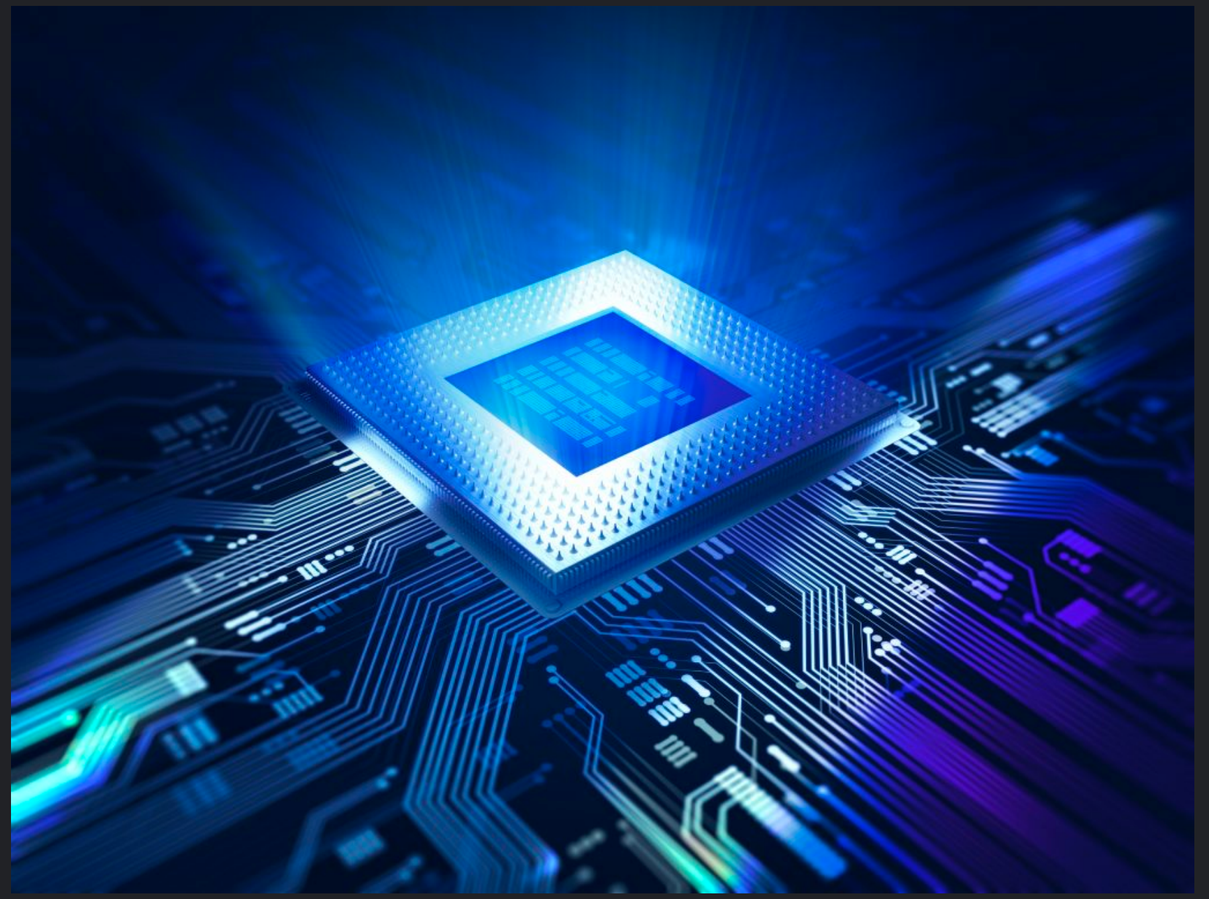
Sudoku Technique

Unique Candidate

Multiprocessing

Multiprocessing Module

- Queue()
- Event()



CSP Statistics

Cut-off time: 6 minutes

	Average time (seconds)	Standard deviation	Success
9x9	0.0349	0.0720	100%
12x12	0.0349	0.0117	100%
16x16	0.5071	1.2454	100%
25x25	13.7136	25.4175	92.9%

CSP Statistics (AWS)

Cut-off time: 5 minutes

- 16x16: 22.83% improvement
- 25x25: 67% improvement

	Average time (seconds)	Standard deviation	Success
16x16	0.4029522	0.18766359	100%
25x25	4.398898	3.885618	90%

THANK YOU FOR LISTENING!