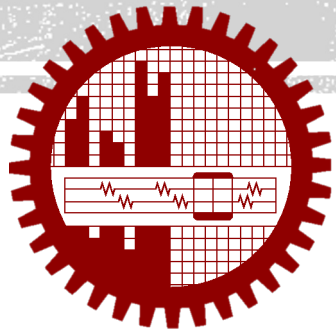
CSE-318
OFFLINE-4 ON KE

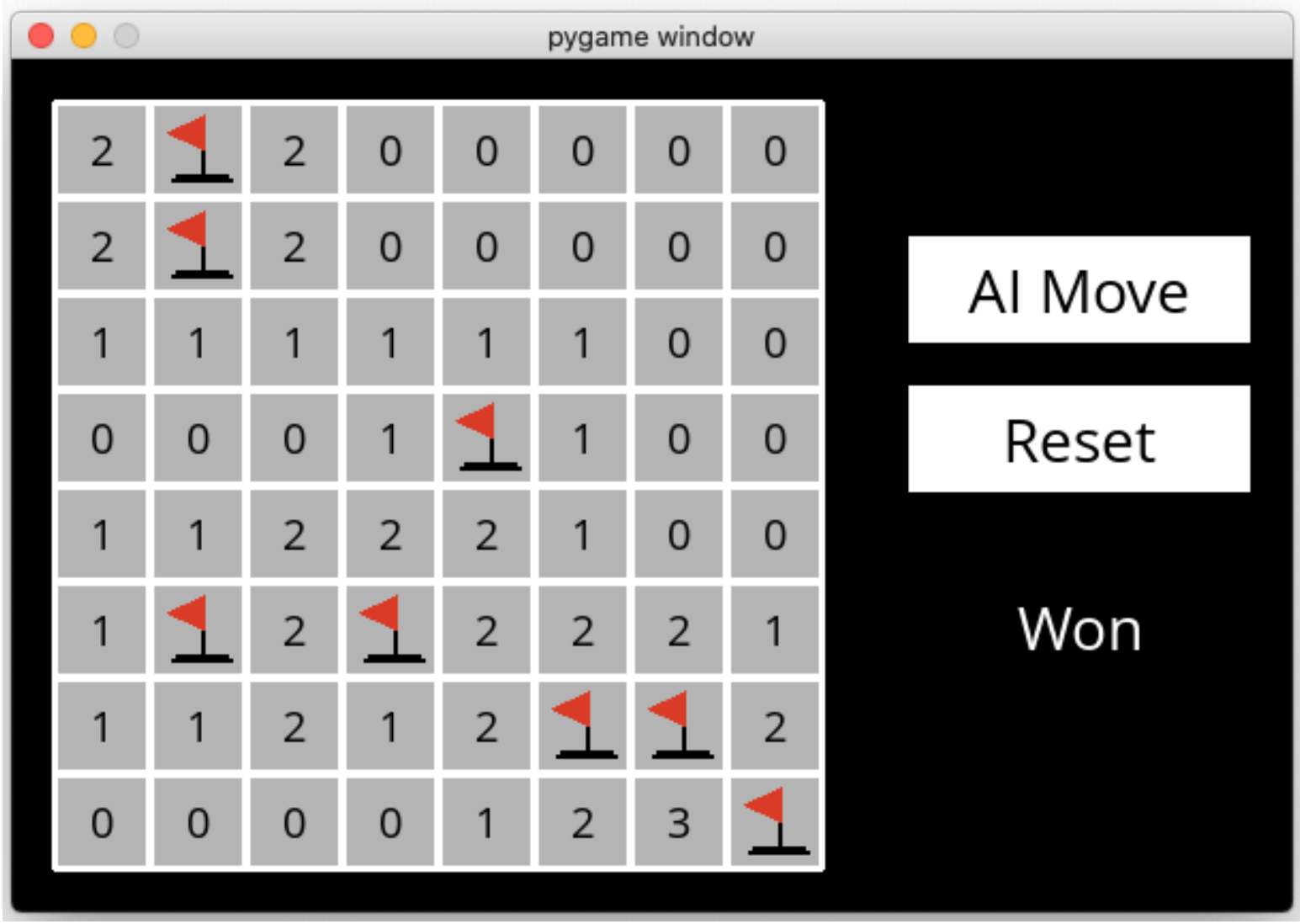
Dr. M A Nayeem

Assistant Professor

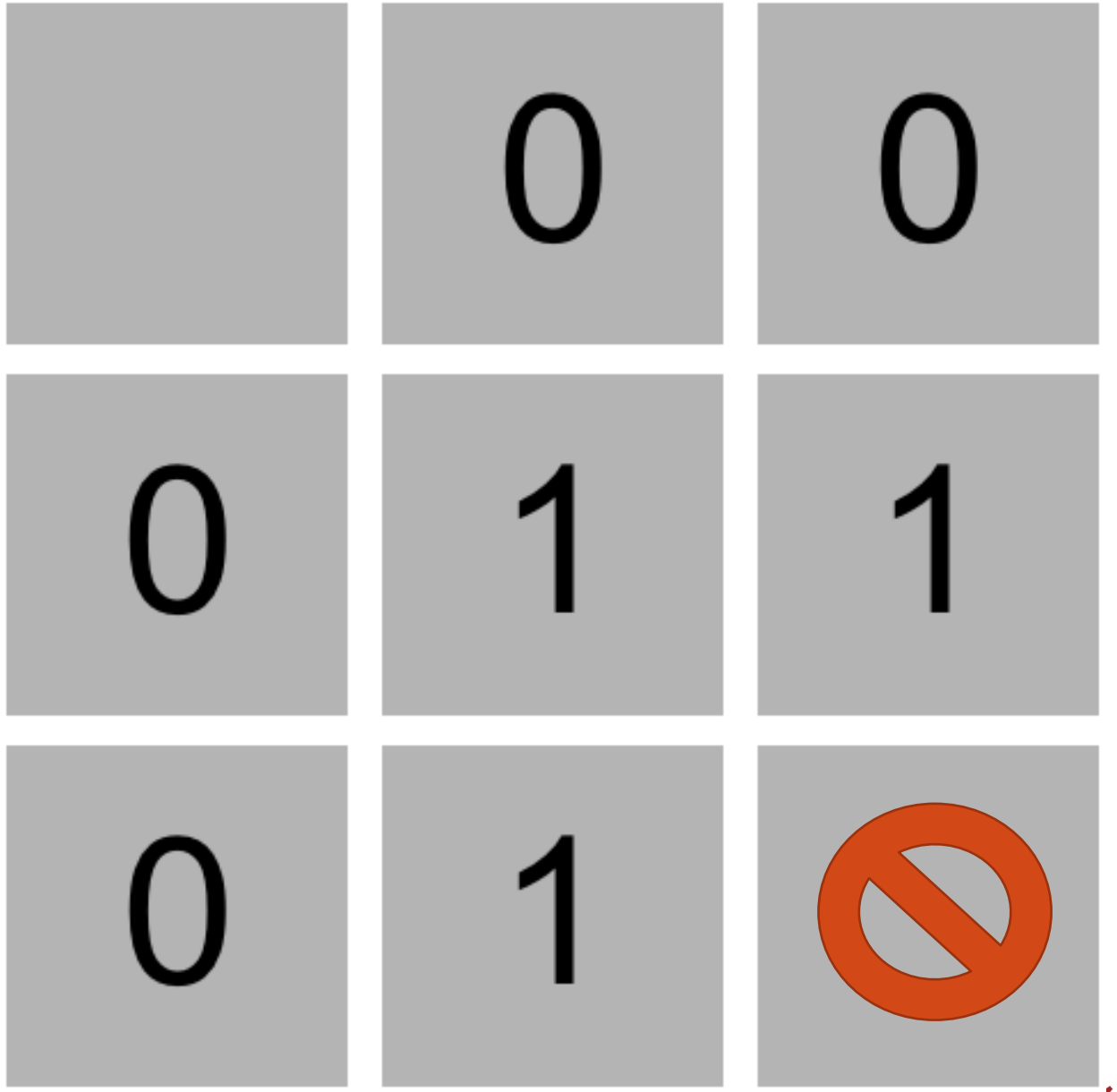
CSE, BUET



MINESWEEPER



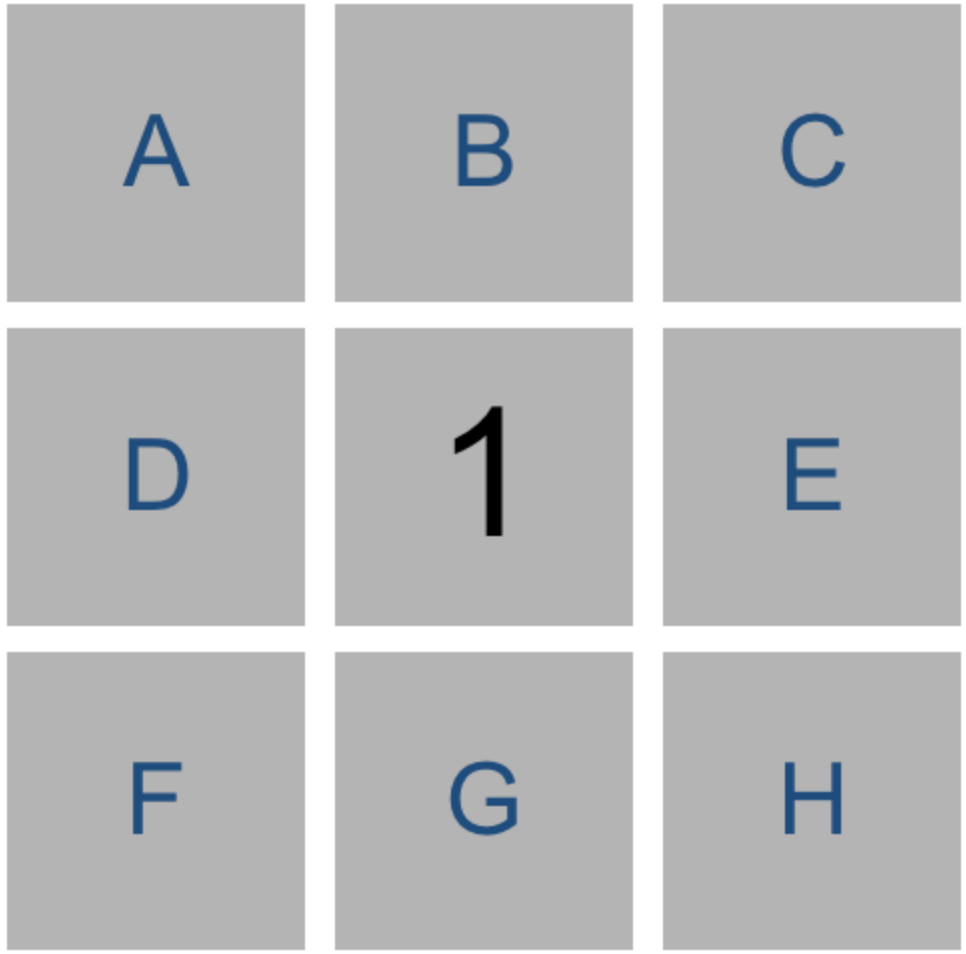
INFERENCE



REPRESENTATION

Or(A, B, C, D, E, F, G, H)

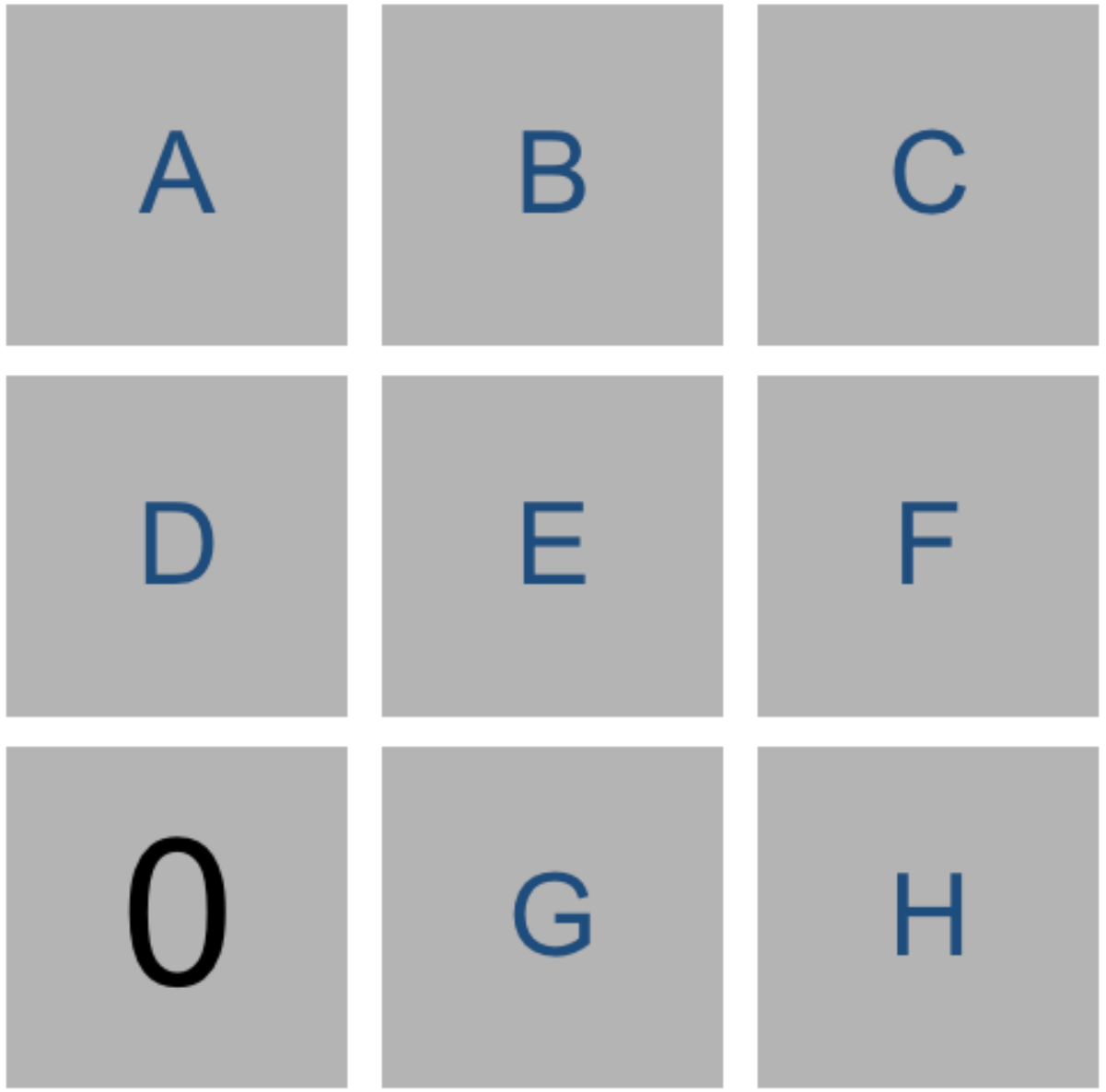
```
Or(
    And(A, Not(B), Not(C), Not(D), Not(E), Not(F), Not(G), Not(H)),
    And(Not(A), B, Not(C), Not(D), Not(E), Not(F), Not(G), Not(H)),
    And(Not(A), Not(B), C, Not(D), Not(E), Not(F), Not(G), Not(H)),
    And(Not(A), Not(B), Not(C), D, Not(E), Not(F), Not(G), Not(H)),
    And(Not(A), Not(B), Not(C), Not(D), E, Not(F), Not(G), Not(H)),
    And(Not(A), Not(B), Not(C), Not(D), Not(E), F, Not(G), Not(H)),
    And(Not(A), Not(B), Not(C), Not(D), Not(E), Not(F), G, Not(H)),
```



{A, B, C, D, E, F, G, H} = 1

INFERENCE

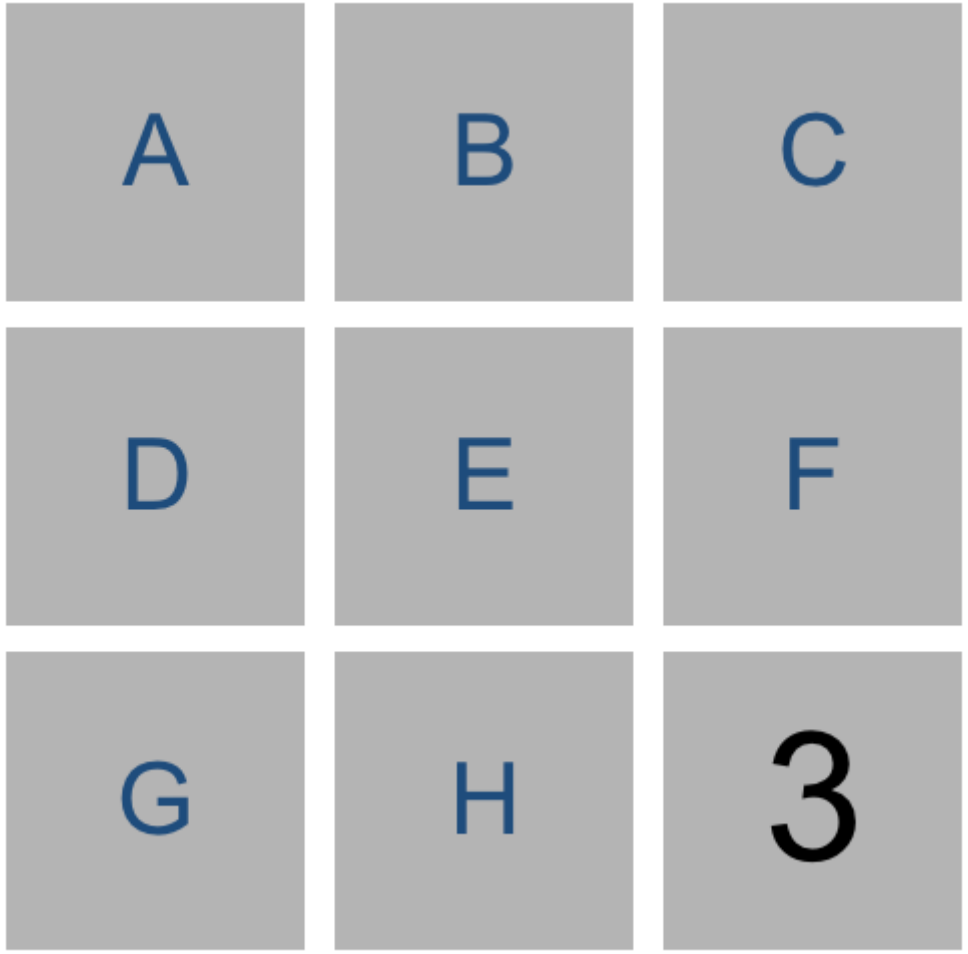
- {D, E, G} = 0
- Each cell is safe



INFERENCE

- {E, F, H} = 3

- Each cell is a mine



UPDATE SENTENCE

- {A, B, C} = 2

- New knowledge
  - C is safe

- Update {A, B, C} = 2
  - {A, B} = 2

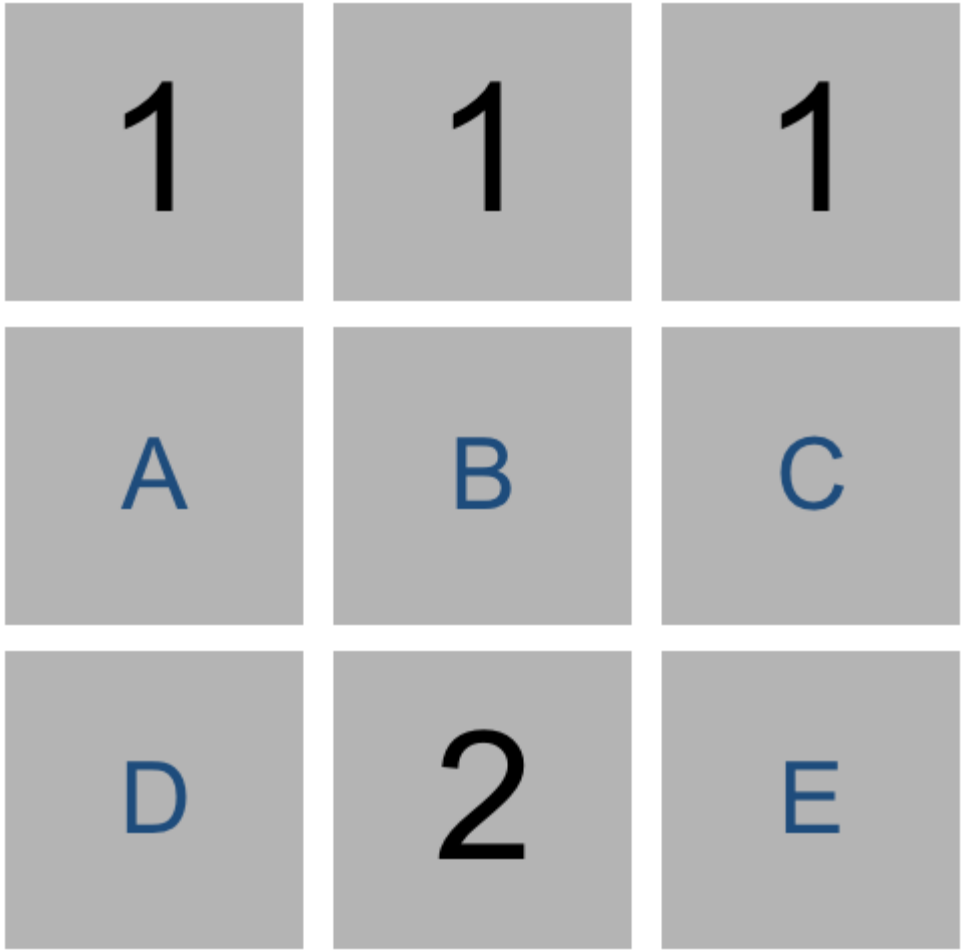
UPDATE SENTENCE

- {A, B, C} = 2

- New knowledge
  - C is mine

- Update {A, B, C} = 2
  - {A, B} = 1

- {A, B, C} = 1
- {A , B, C, D, E} = 2
- {D, E} = 1
- Set1 = count1, Set2 = count2
  - Set1 is a subset of Set2
  - Set2 – Set1 = count2 – count1



SENTENCE CLASS

- Implement
  - `known_mines()`
  - `known_safes()`
  - `mark_mine(cell)`
  - `mark_safe(cell)`

Sentence Class

- The `known_mines` function should return a set of all of the cells in `self.cells` that are known to be mines.
- The `known_safes` function should return a set of all the cells in `self.cells` that are known to be safe.
- The `mark_mine` function should first check to see if `cell` is one of the cells included in the sentence.
  - If `cell` is in the sentence, the function should update the sentence so that `cell` is no longer in the sentence, but still represents a logically correct sentence given that `cell` is known to be a mine.
  - If `cell` is not in the sentence, then no action is necessary.
- The `mark_safe` function should first check to see if `cell` is one of the cells included in the sentence.
  - If `cell` is in the sentence, the function should update the sentence so that `cell` is no longer in the sentence, but still represents a logically correct sentence given that `cell` is known to be safe.
  - If `cell` is not in the sentence, then no action is necessary.

MINESWEEPERAI CLASS

- Implement
  - `add_knowledge(cell, count)`
    - A lot of works
  - `make_safe_move()`
  - `make_random_move()`

- `add_knowledge` should accept a `cell` (represented as a tuple `(i, j)`) and its corresponding `count`, and update `self.mines`, `self.safes`, `self.moves_made`, and `self.knowledge` with any new information that the AI can infer, given that `cell` is known to be a safe cell with `count` mines neighboring it.
  - The function should mark the `cell` as one of the moves made in the game.
  - The function should mark the `cell` as a safe cell, updating any sentences that contain the `cell` as well.
  - The function should add a new sentence to the AI's knowledge base, based on the value of `cell` and `count`, to indicate that `count` of the `cell`'s neighbors are mines. Be sure to only include cells whose state is still undetermined in the sentence.
  - If, based on any of the sentences in `self.knowledge`, new cells can be marked as safe or as mines, then the function should do so.
  - If, based on any of the sentences in `self.knowledge`, new sentences can be inferred (using the subset method described in the Background), then those sentences should be added to the knowledge base as well.
  - Note that any time that you make any change to your AI's knowledge, it may be possible to draw new inferences that weren't possible before. Be sure that those new inferences are added to the knowledge base if it is possible to do so.

- `make_safe_move` should return a move `(i, j)` that is known to be safe.
  - The move returned must be known to be safe, and not a move already made.
  - If no safe move can be guaranteed, the function should return `None`.
  - The function should not modify `self.moves_made`, `self.mines`, `self.safes`, or `self.knowledge`.
- `make_random_move` should return a random move `(i, j)`.
  - This function will be called if a safe move is not possible: if the AI doesn't know where to move, it will choose to move randomly instead.
  - The move must not be a move that has already been made.
  - The move must not be a move that is known to be a mine.
  - If no such moves are possible, the function should return `None`.

RULES

- Discard diagonal cells as neighbors

- `make_random_move()`
  - Works completely randomly
  - Makes no attempt to increase the probability of safety

THANKS
KEEP SMILING :)