Subgoals

CMPUT 355: Games, Puzzles, and Algorithms

Lecture Outline

1. Logistics & Recap
2. Subgoals in the sliding tile puzzle
3. Connected components: A subgoal-based proof

Logistics

- **Practice quiz questions #2:**
  - Will be posted today (possibly early tomorrow)
  - Answers will be posted following Tuesday (Feb 3)
- **Quiz 1 marking:**
  - Still underway, should be done in the next couple of days
- **Quiz 2:** Next Friday, **Feb 6**
  - In-class, full 50 minutes
  - **No need to email** if you have to miss it;
    up to 3 missed quizzes replaced by final exam **automatically**
  - Coverage: up to the end of **today's lecture**
  - Questions will be very similar to practice questions

Recap: Heuristic Search

- **Heuristic:** an estimate of the remaining "distance" (cost) from a node to the target
  - Not always accurate!
- **Admissible heuristic:** guaranteed to be a lower bound on remaining cost
  - i.e., a lower bound on how "bad" a given node is
- **A*** explores nodes in order of their **estimated total distance:**
  - (distance from the source to n) + (**estimated** distance from n to target)
  - First distance is real, second distance is computed by the heuristic
- **Sliding tile specific heuristics:**
  - Misplaced tiles
  - Taxicab distance
- These heuristics are good enough for A* to find solution to a **solvable** 4×4 position in under 1s
  - But **unsolvable** instances are still intractable

Domain-Specific Information

- Heuristics are a way to exploit domain-specific information that is cheap to compute
    - Function of the position rather than the graph structure
- **Today:** What if we solved for subgoals instead? ("baby steps")
- Approach:
    1. Divide the problem into "easier" subproblems
    2. Solved in order means that the overall problem is solved
    3. Breadth-first search to solve each subproblem

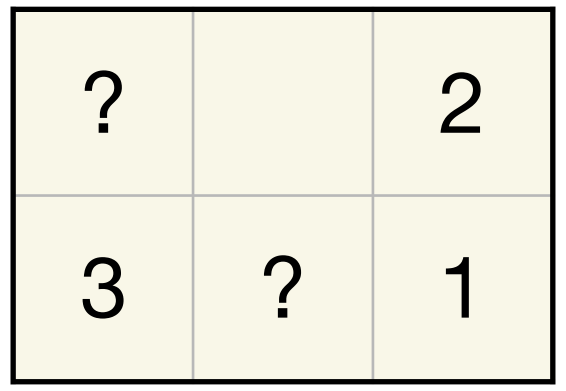
Questions:

1. What makes this **domain-specific**?
2. Will this be **faster**? (**why?**)
3. Is this guaranteed to find the **shortest** solution?
4. What could go **wrong**?

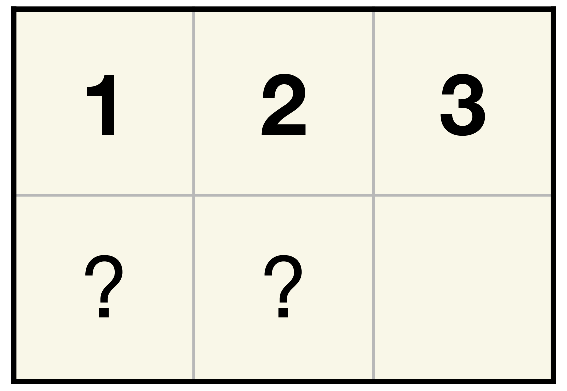
Example: Subgoals for 2×3 puzzles

- First method, solve two subproblems:
  A. Get top row arranged correctly
  B. Finish puzzle *without touching top row*

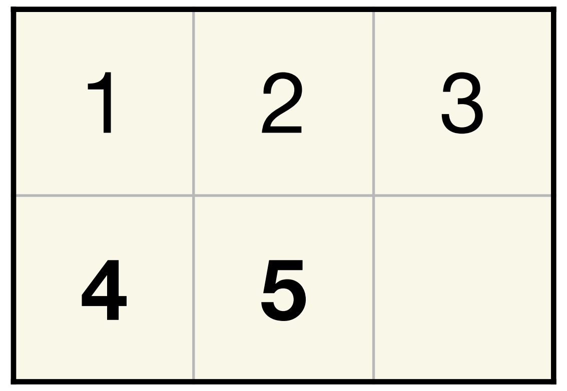
- Second method, solve two subproblems:
  A. Get leftmost column arranged correctly
  B. Finish puzzle *without touching left column*



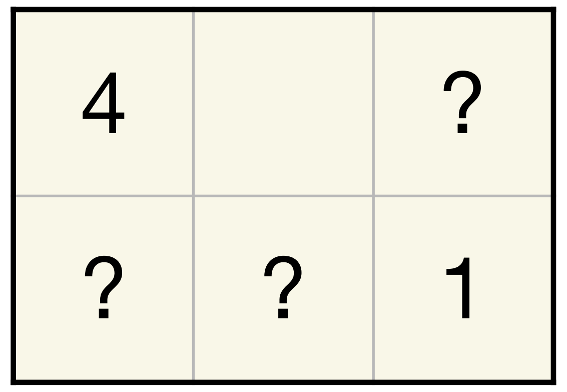
Initial



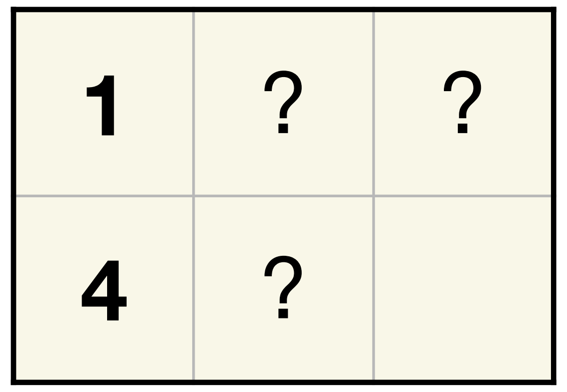
After A



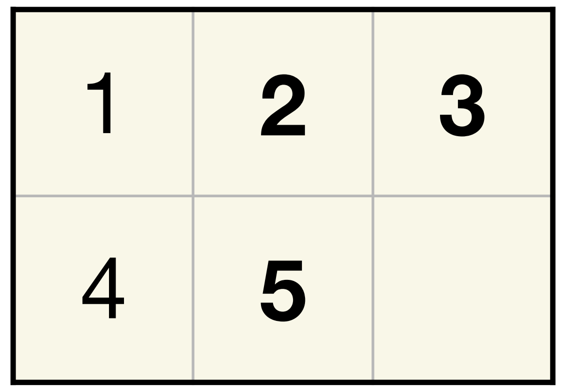
After B



Initial



After A



After B

Example: Subgoals for 3×3

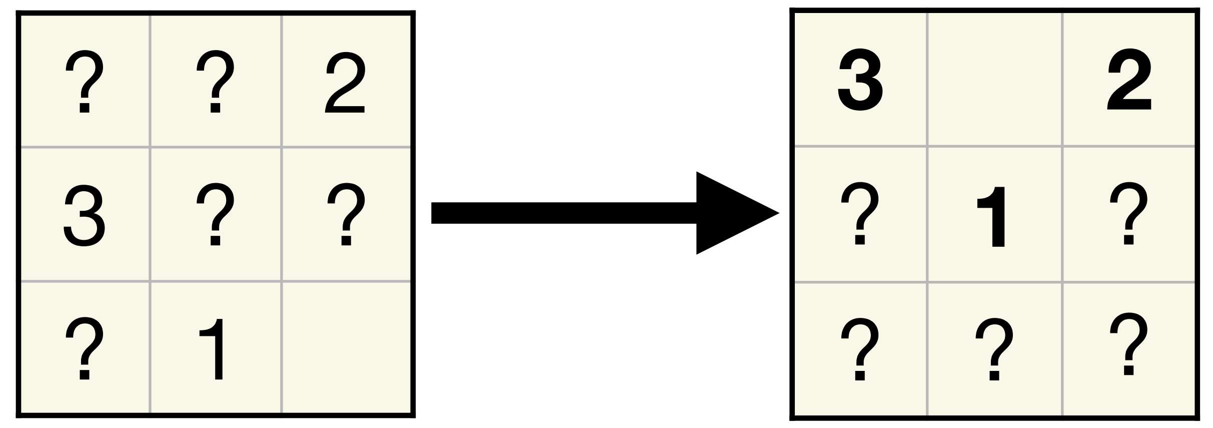
A. Move the blank and $1,2,3$ to the top two rows

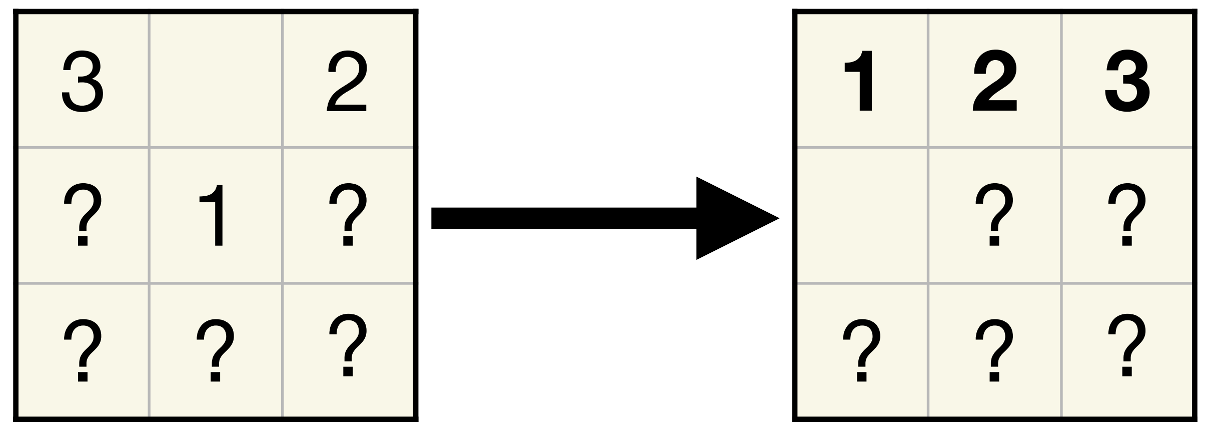
B. Make the top row correct *using only the top two rows*

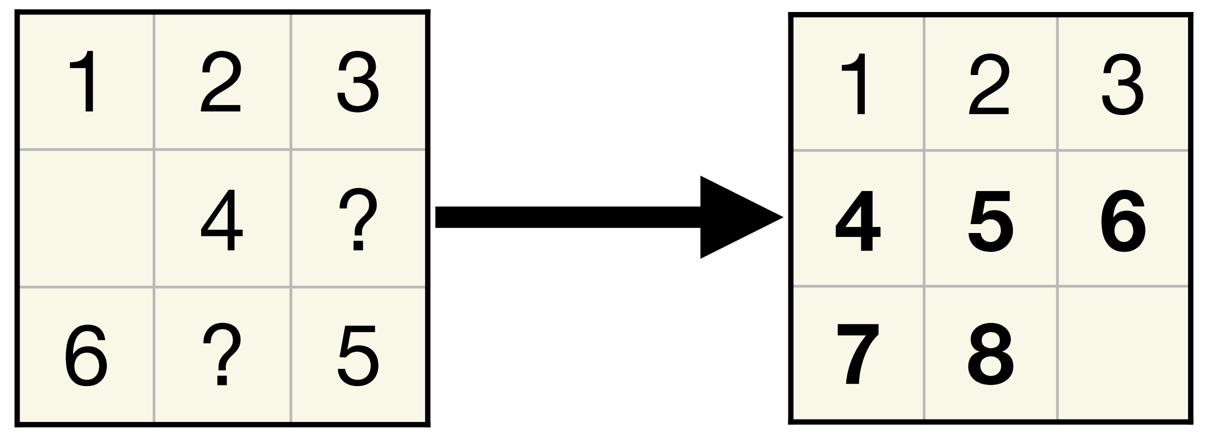
C. Make the bottom two rows correct *without touching the top row*

- e.g., by using a 2×3 method from the previous slide!

- **Question:** Is this method guaranteed to work? Why or why not?







Example Subgoals for 4×4

X. Place top, then left column, then solve 3×3:

[[1,2,3,4],[5,9,13],[6,7,8,10,11,12,14,15]]

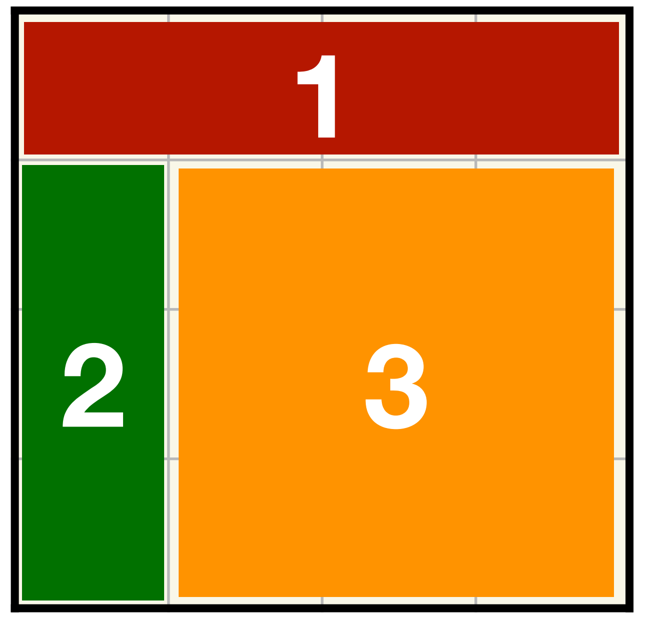
Y. Place one or two stones at a time:

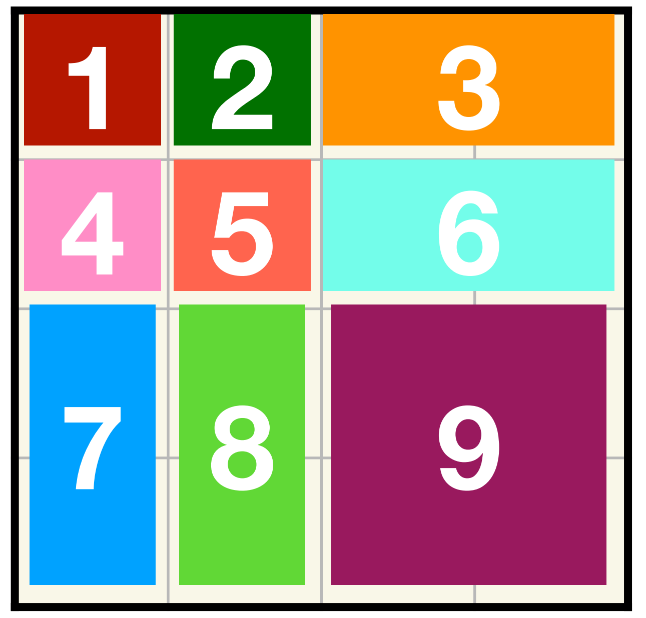
[[1],[2],[3,4],[5],[6],[7,8],[9,13],[10,14],[11,12,15]]

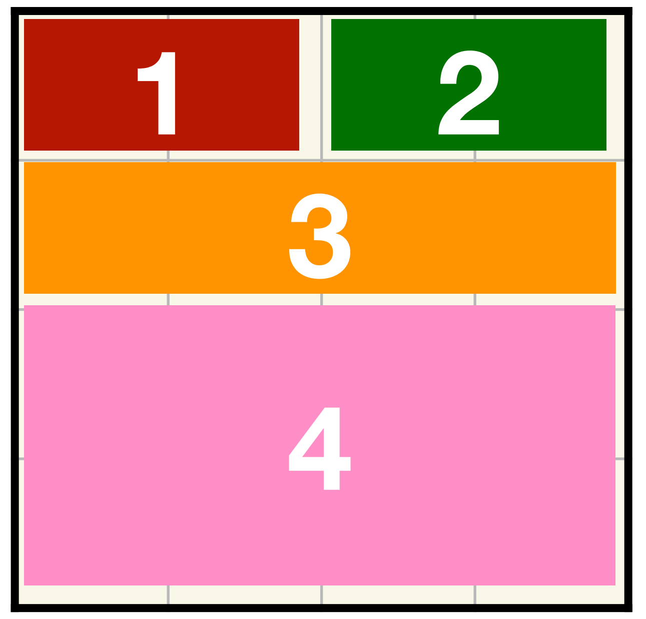
Z. Place first row in two steps, then second row, then solve bottom two rows:

[[1,2],[3,4],[5,6,7,8],[9,10,11,12,13,14,15]]

- **Question:** Which subgoal schedule will find solution **fastest**? (**why?**)
- **Question:** Which subgoal structure will find **shortest** solution? (**why?**)
- **Demo:** stile/15puzzle.py





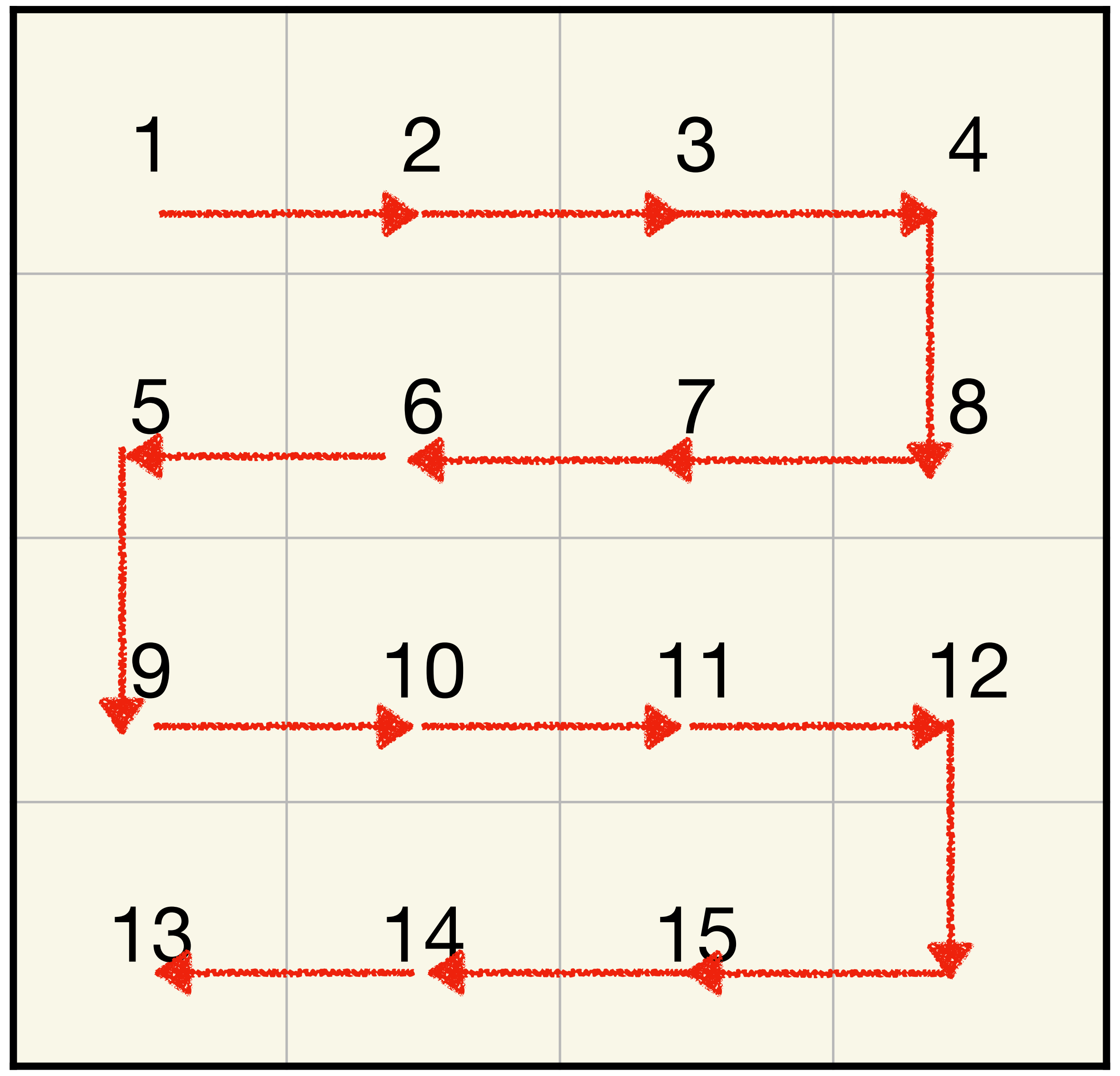


Sliding Tiles: Is Even Parity Sufficient?

- **Recall:**
  1. All **solvable** positions are **even-parity**
     Either inversions or inversions+row_height are even, depending on width
  2. **Solvable** positions are a **connected component**
- **But:** Are all **even-parity** positions **solvable**?
  - i.e., are the even-parity positions a connected component?
- We can use **subgoals** in a proof that all **even-parity positions** can be solved

1. Arrangements

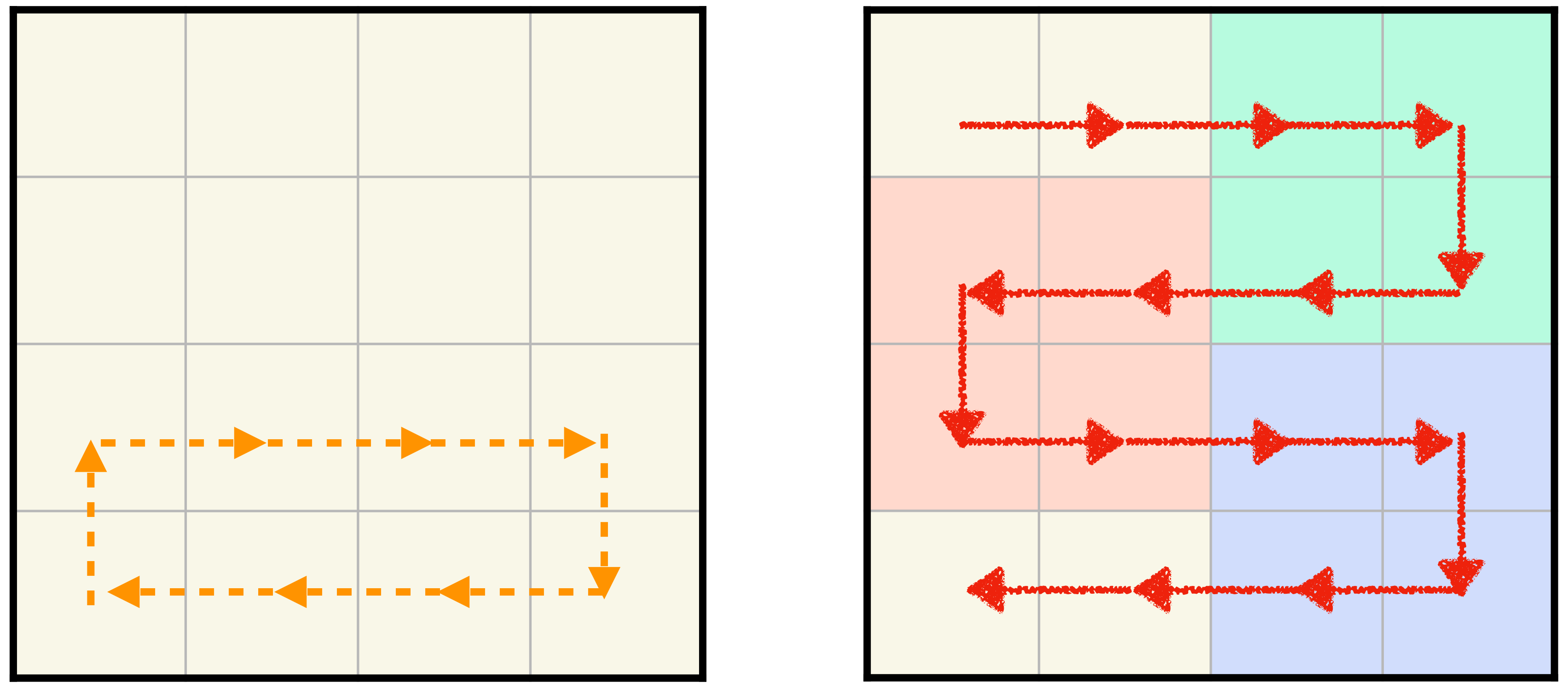
- Read off the numbers on the tiles, skipping the blank
- Instead of going left-to-right on each row, go left-to-right, then right-to-left, etc.
- This order is called an **arrangement**
- So the arrangement for the solved position is 1, 2, 3, 4, 8, 7, 6, 5, 9, 10, 11, 12, 15, 14, 13
- **Claim:** you can move the blank to any position without changing the arrangement (**how?**)

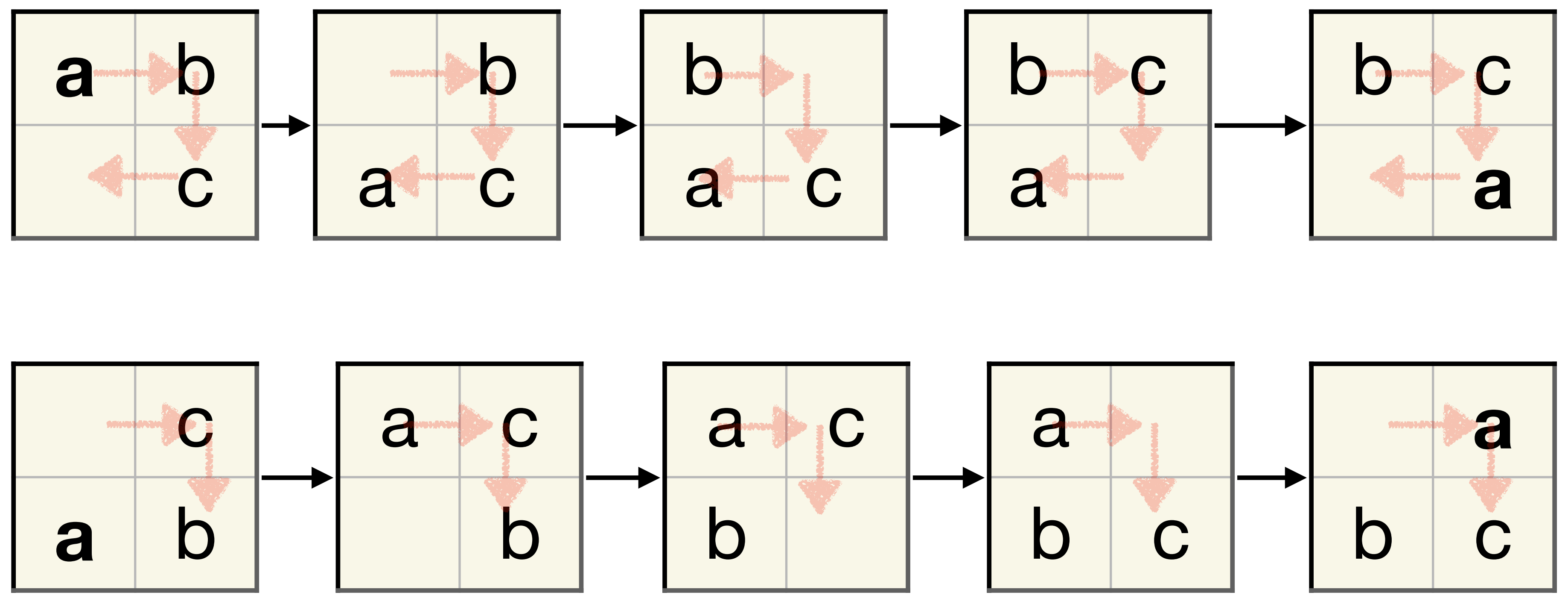


2. "Passing Over" Tiles

Claim: Any tile can be moved backward or forward by two positions in any arrangement, *without* changing the rest of the arrangement

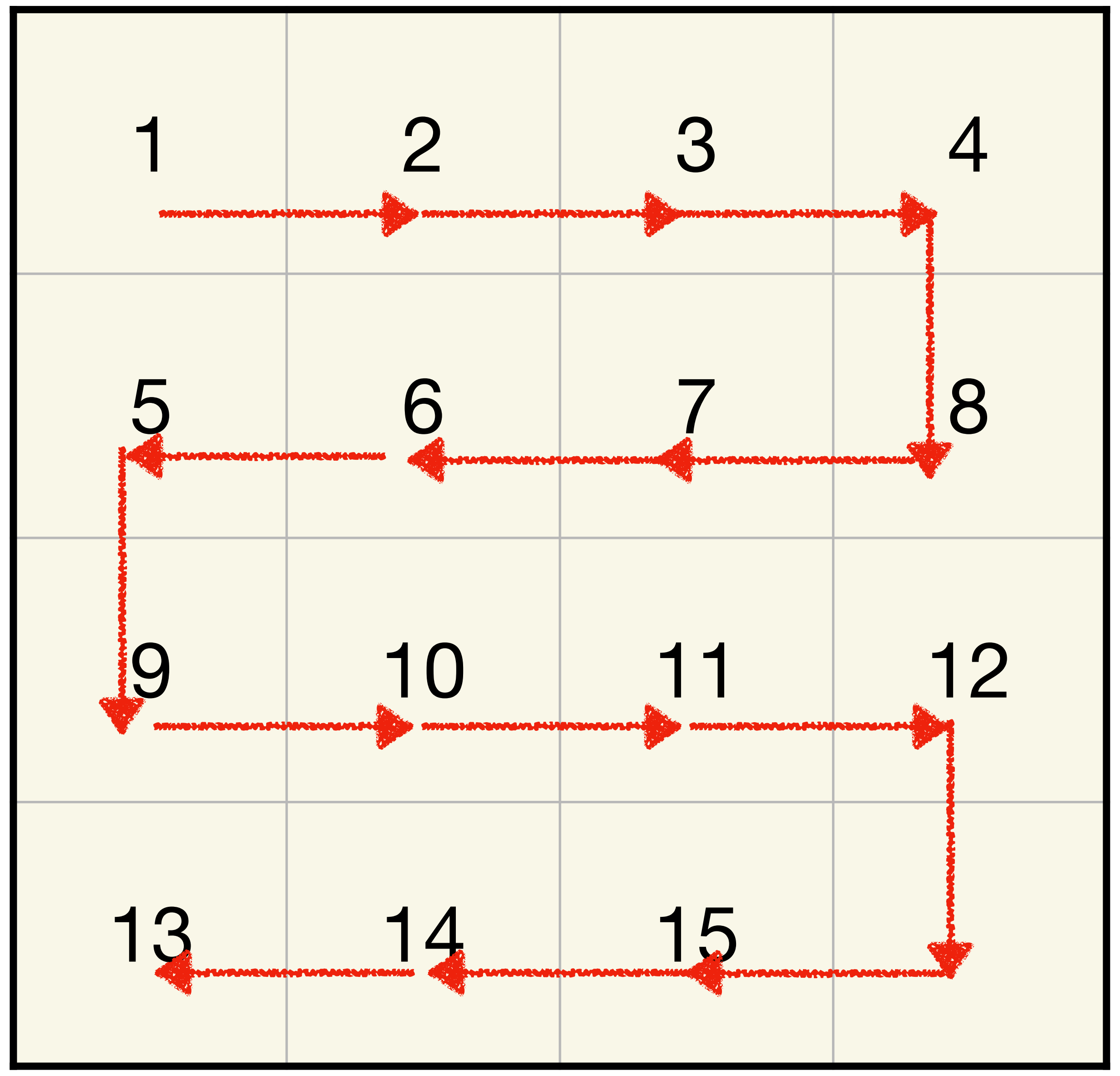
1. Move the blank along the arrangement to the tile's row
2. If necessary, rotate the two rows containing the tile and its predecessors (successors) until the tile and its predecessors (successors) are in an "elbow": four squares that are sequential in the arrangement
3. Rotate the elbow to skip the tile forward or backward by two positions
4. Undo the rotation in step 2





Subgoal: Single-tile placement

- If there are n tiles, then the first $n-2$ tiles in the arrangement can be correctly placed
  - (Placing each tile in order is a **subgoal**!)
- First, place the first tile by skipping it back **2 at a time**
- It will either reach its home spot, or 1 spot to the right
  - If it's in the home spot, we're done
  - If it's one spot to the right, then skip the tile in the home spot forward by two
- Iteratively do the same for the second, third, etc.
- After the $(n-2)$-nd tile (i.e., 13) has been placed, two cases:
  - 14 and 15 are already in correct order: we have succeeded
  - 14 and 15 are out of order: position is odd-parity (**why?**)
- Therefore, any **even-parity position** can be brought to the solved position, so the even-parity positions are a **connected component**
- **Question:** What almost-identical argument shows that **odd-parity** positions are also a **connected component**?



Summary

- Some search problems can be solved more efficiently by solving intermediate **subgoals** (instead of the whole problem at once)
    - **Shrinks the search space**
    - If guaranteed that all but last subgoals can be accomplished in any position, then proving unsolvable can even be feasible
- **Bad news:** May not find the shortest solution
- More subgoals make solving **faster**, but may lead to **worse** solution
- Fewer subgoals give a **better** solution, but may take **longer**
    - Extreme case: A single subgoal, which is the original goal
- Can use a **subgoal-structured proof** to show that **all even positions are solvable** in a $k \times k$ sliding tile puzzle