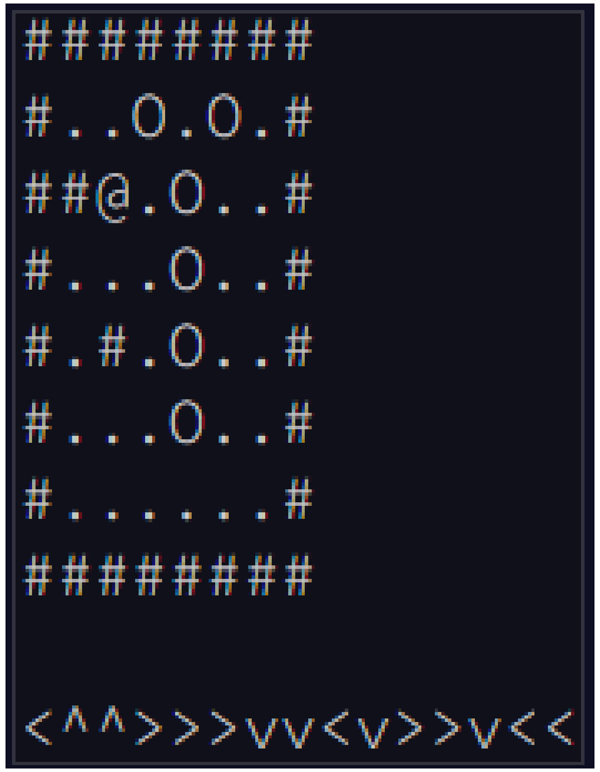
AoC 2024, Day 15: Warehouse Woes

Part 1

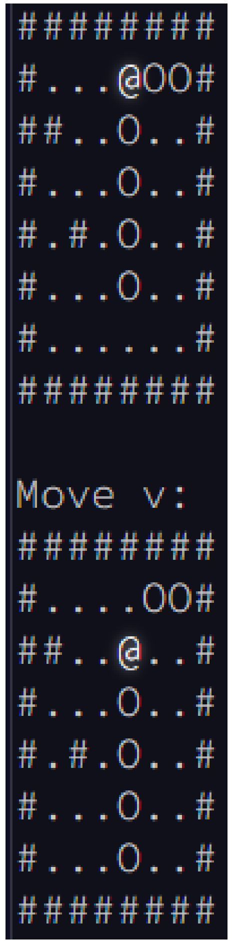


```
########
#..O.O.#
##@.O..#
#...O..#
#.#.O..#
#...O..#
#......#
########

<^^>>>vv<v>>v<<
```

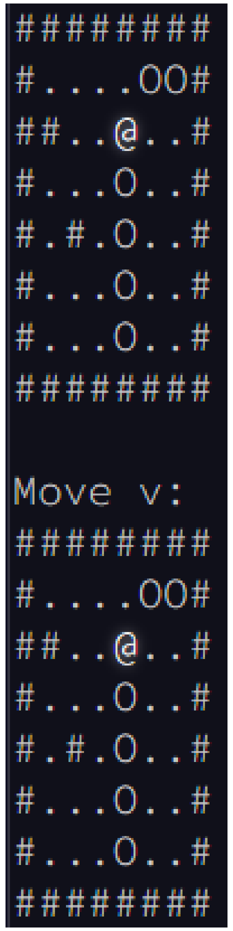
Input

- 2D grid of
  - Walls (#)
  - Boxes (O)
  - Open space (.)
  - A robot (@)
  - A sequence of ^, v, >, < characters representing robot movements one space to the north, south, east, and west, respectively

Goal and Approach



```
########
#...@OO#
##..O..#
#...O..#
#.#.O..#
#...O..#
#......#
########

Move v:
########
#....OO#
##..@..#
#...O..#
#.#.O..#
#...O..#
#...O..#
########
```



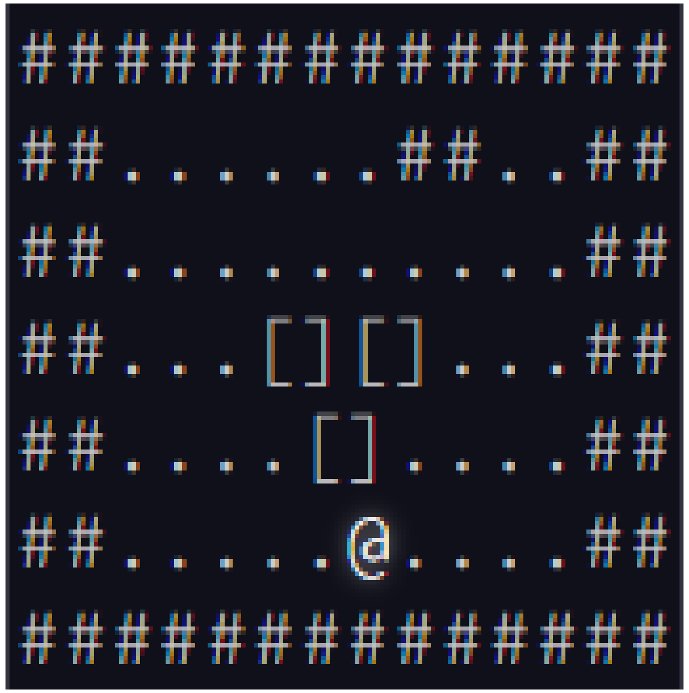
```
########
#....OO#
##..@..#
#...O..#
#.#.O..#
#...O..#
#...O..#
########

Move v:
########
#....OO#
##..@..#
#...O..#
#.#.O..#
#...O..#
#...O..#
########
```

```
initialize robot location
for each movement symbol:
    collect coords and symbols until:
        if empty space found:
            shift symbols to new coords
            update robot location
        else if wall found:
            cannot push; no change
```

- Answer calculated based on final locations of boxes
- Used a stack for the collection to shift symbols in reverse order
- Backfill robot space with a ‘.’ after shift!

AoC 2024, Day 15: Warehouse Woes

Part 2

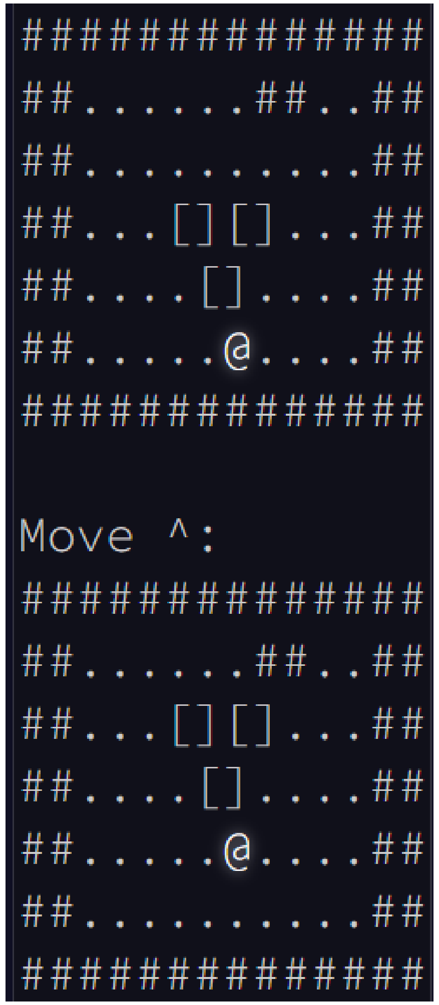


```
##############
##......##..##
##..........##
##...[][]...##
##....[]....##
##.....@....##
##############
```

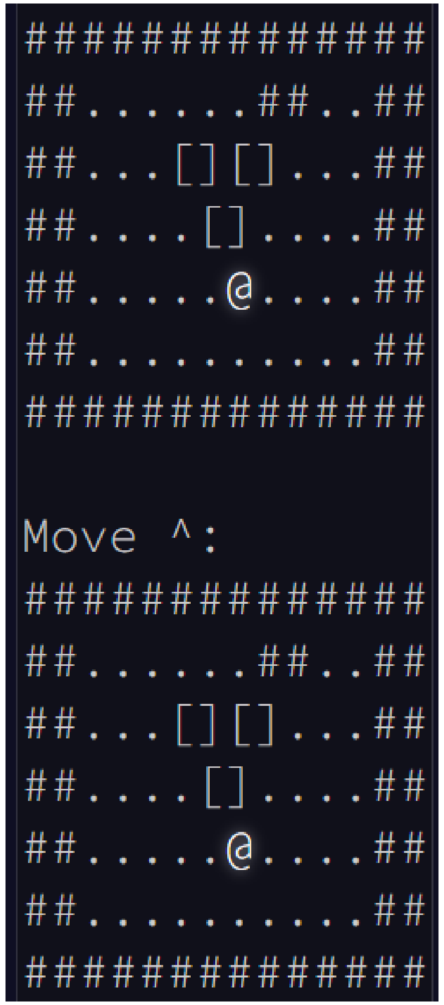
Input

- 2D grid of
  - Walls (#)
  - Double-wide boxes! ([ ])
    - (but still single-height)
  - Open space (.)
  - A robot (@)
  - A sequence of ^, v, >, < characters representing robot movements one space to the north, south, east, and west, respectively

Goal and Approach



```
##############
##......##..##
##..........##
##...[][]...##
##....[]....##
##.....@....##
##############

Move ^:
##############
##......##..##
##...[][]...##
##....[]....##
##.....@....##
##..........##
##############
```



```
##############
##......##..##
##...[][]...##
##....[]....##
##.....@....##
##..........##
##############

Move ^:
##############
##......##..##
##...[][]...##
##....[]....##
##.....@....##
##..........##
##############
```

- Note: we can use the same solution from Part 1 for east/west movement
- Two steps for north or south:
    1. to_check: *all* involved robot/box *sides* for no-wall (can quit as soon as *anything* is blocked)
    2. If not blocked, add them to collection to move (to_move)
- Stored in to_move as (r,c,symb)
- Why? Can use natural sort (or reversed natural sort) to shift rows farthest from robot first
- Again, answer calculated based on final locations of boxes