T254 - Robo's Recursive Realm

Kelvin Chow {Lrt1088}
2025-05-10

Table of Contents

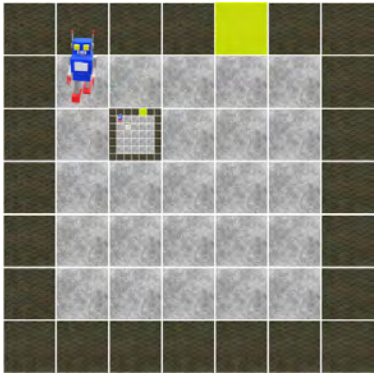
Background

Problem Idea by `mtyeung1`
Preparation by `Lrt1088` and `WongChun1234` (Thanks!)

(Partially inspired by Patrick's Parabox, IOI 2023 Robot, IOI 2012 Pebbling Odometer, and the MIPS training)

- If you attended the MIPS lecture, you would notice that many types of instructions are not "new"!
- Prior experience could aid you to get used to the environment much quicker.

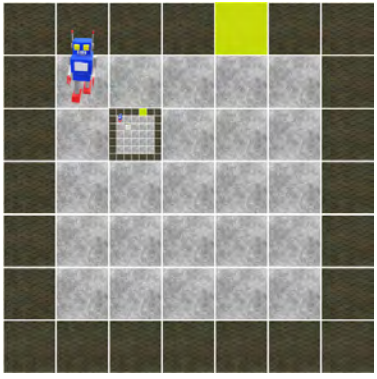
Trainer's Best: 57 for $R = 0$, 93 for $R = 1$.



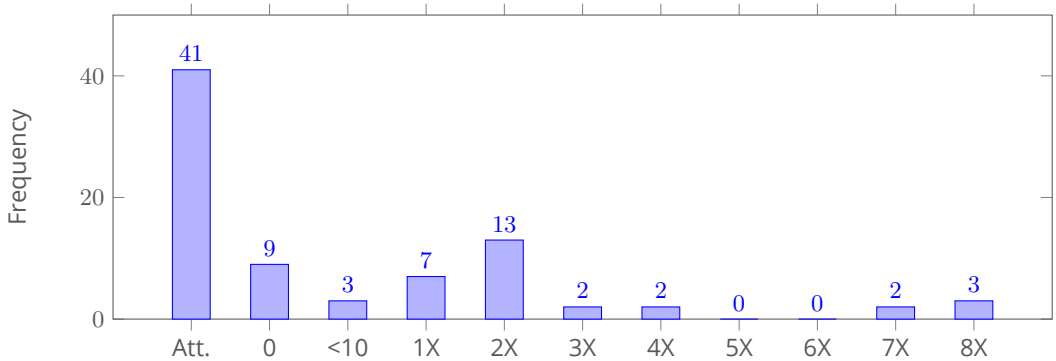
Problem Restatement

Write a Robo program to push the box to the exit.

- Available Commands: `F`, `L`, `R`, `BB` (branch on box), `BW` (branch on wall), `J` (jump).
- *Recurbox*: The box will trigger a teleportation when pushed from one direction.
- If it's impossible to do so, terminate with the special instruction `DIE`.
- Use **as few instructions as possible**. (Partial scoring on the number of instructions)



Statistics



First solved by no one! And no one got any marks on Subtask 6!

Subtasks

Subtask	Points	R	Box on row 2?	Exit on top?	Constraints
1	8	0	✓	✓	$N = 5$
2	15		✓	✓	
3	37		✗	✗	
4	14	1	✓	✓	Always possible
5	10		✓	✓	
6	16		✗	✗	

For this solution session, instead of going through the solutions subtask by subtask, we will focus on the key ideas which could be useful for problem solving in general.

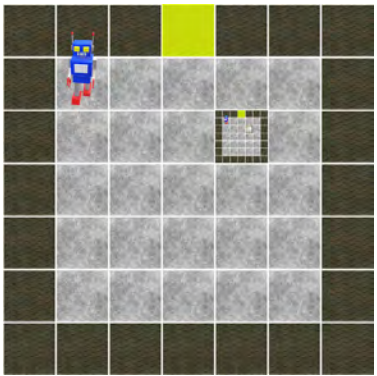
Table of Contents

The Box Pushing Mechanics – Motivation

- Robo is dealing with two major **uncertainties** – unknown location of the box and unknown location of the exit.
- Under different subtask constraints, it will be useful for us to make sense of what operations are allowed, and how we could reduce these uncertainties to make sure that Robo pushes the box to the exit.
- Some questions to consider:
  - Is the box guaranteed to be on row 2 initially?
  - Is the exit guaranteed to be on row 1?
  - Is the box a recurbox?

The Box Pushing Mechanics – Subtask 2

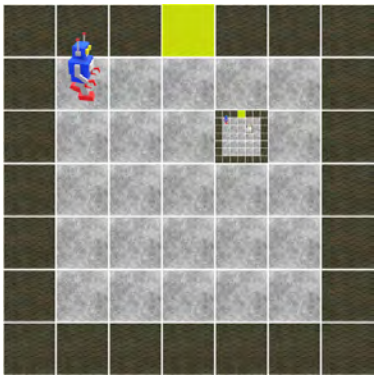
R	Box on row 2?	Exit on top?
0	✓	✓



- If we already know the location of the box and the exit, pushing the box to the exit is simple.
- What if we only know that the box is at row 2?
  Idea: Always push the box to a fixed location first.
- Go to the rightmost of row 2, and "push all the way left". Commands: `L` $\underbrace{\texttt{FF}\ldots\texttt{F}}_{N-1 \text{ times}}$ `RFR` $\underbrace{\texttt{FF}\ldots\texttt{F}}_{N-2 \text{ times}}$.
  $\Rightarrow$ Box is at $(2, 1)$ eventually.

The Box Pushing Mechanics – Subtask 2

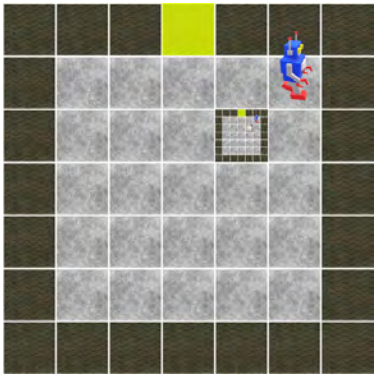
R	Box on row 2?	Exit on top?
0	✓	✓



- If we already know the location of the box and the exit, pushing the box to the exit is simple.
- What if we only know that the box is at row 2?
  Idea: Always push the box to a fixed location first.
- Go to the rightmost of row 2, and "push all the way left". Commands: `L` $\underbrace{\texttt{FF}\ldots\texttt{F}}_{N-1 \text{ times}}$ `RFR` $\underbrace{\texttt{FF}\ldots\texttt{F}}_{N-2 \text{ times}}$.
  $\Rightarrow$ Box is at $(2, 1)$ eventually.

The Box Pushing Mechanics – Subtask 2

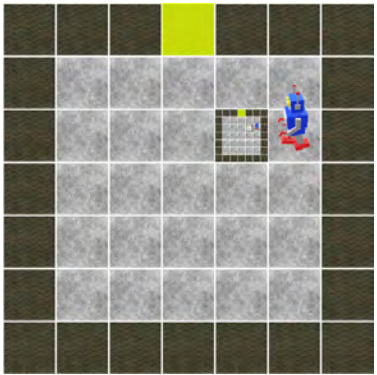
R	Box on row 2?	Exit on top?
0	✓	✓



- If we already know the location of the box and the exit, pushing the box to the exit is simple.
- What if we only know that the box is at row 2?
  Idea: Always push the box to a fixed location first.
- Go to the rightmost of row 2, and "push all the way left". Commands: `L` $\underbrace{\texttt{FF}\ldots\texttt{F}}_{N-1 \text{ times}}$ `RFR` $\underbrace{\texttt{FF}\ldots\texttt{F}}_{N-2 \text{ times}}$.
  $\Rightarrow$ Box is at $(2, 1)$ eventually.

The Box Pushing Mechanics – Subtask 2

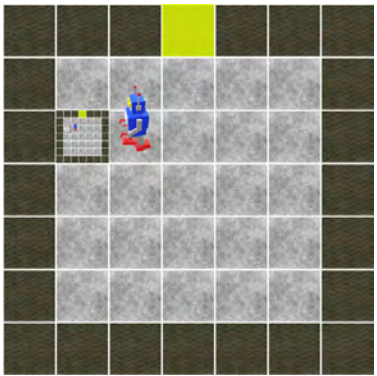
R	Box on row 2?	Exit on top?
0	✓	✓



- If we already know the location of the box and the exit, pushing the box to the exit is simple.
- What if we only know that the box is at row 2?
  Idea: Always push the box to a fixed location first.
- Go to the rightmost of row 2, and "push all the way left". Commands: `L` $\underbrace{\texttt{FF}\ldots\texttt{F}}_{N-1 \text{ times}}$ `RFR` $\underbrace{\texttt{FF}\ldots\texttt{F}}_{N-2 \text{ times}}$.
  $\Rightarrow$ Box is at $(2, 1)$ eventually.

The Box Pushing Mechanics – Subtask 2

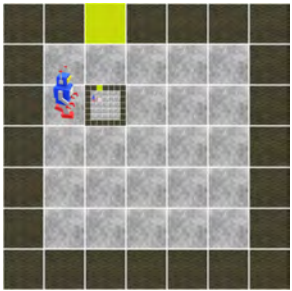
R	Box on row 2?	Exit on top?
0	✓	✓



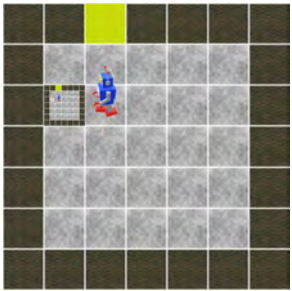
- If we already know the location of the box and the exit, pushing the box to the exit is simple.
- What if we only know that the box is at row 2?
  Idea: Always push the box to a fixed location first.
- Go to the rightmost of row 2, and "push all the way left". Commands: L $\underbrace{\texttt{FF...F}}_{N-1 \text{ times}}$ RFR $\underbrace{\texttt{FF...F}}_{N-2 \text{ times}}$.
  $\Rightarrow$ Box is at $(2, 1)$ eventually.
- There is a problem...

The Box Pushing Mechanics – Subtask 2

Critical Observation: Some operations are **irrevertible**, i.e., once we made some incorrect pushes, it could be impossible to revert to the initial state.



$(2, 2)$: can be pushed in all directions.

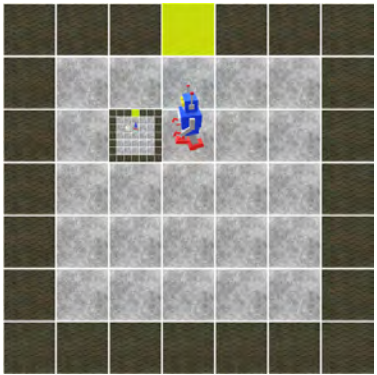


$(2, 1)$: cannot push to the right anymore.

The Box Pushing Mechanics – Subtask 2

R	Box on row 2?	Exit on top?
0	✓	✓

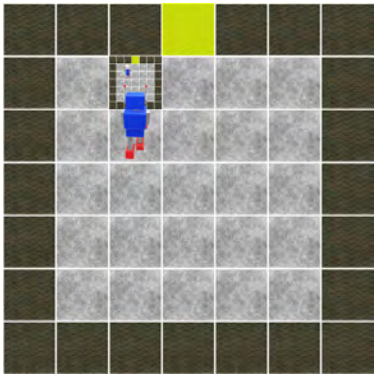
- **Push the box to** $(2, 2)$ **instead**.
  Commands: `L` $\underbrace{\texttt{FF...F}}_{N-1 \text{ times}}$ `RFR` $\underbrace{\texttt{FF...F}}_{N-3 \text{ times}}$.



The Box Pushing Mechanics – Subtask 2

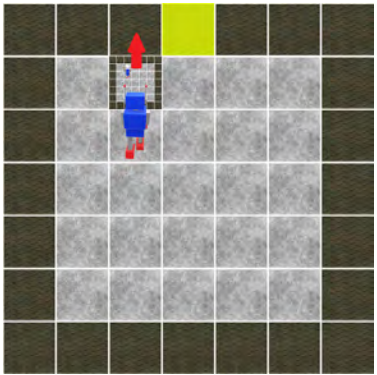
R	Box on row 2?	Exit on top?
0	✓	✓

- **Push the box to** $(2, 2)$ **instead**.
  Commands: `L` $\underbrace{\texttt{FF...F}}_{N-1 \text{ times}}$ `RFR` $\underbrace{\texttt{FF...F}}_{N-3 \text{ times}}$.
- Can we push the box to row 1?
  Yes, since the exit is always at row 1.



The Box Pushing Mechanics – Subtask 2

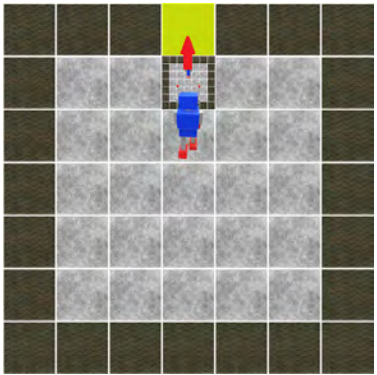
R	Box on row 2?	Exit on top?
0	✓	✓



- **Push the box to** $(2, 2)$ **instead**.
  Commands: `L` $\underbrace{\texttt{FF...F}}_{N-1 \text{ times}}$ `RFR` $\underbrace{\texttt{FF...F}}_{N-3 \text{ times}}$.
- Can we push the box to row 1?
  Yes, since the exit is always at row 1.
- How can we find the exit?
  Consider **brute force** (maybe)?
  Commands: $\underbrace{\texttt{FLFRFRFRFLFL...}}_{N-2 \text{ times}}$

The Box Pushing Mechanics – Subtask 2

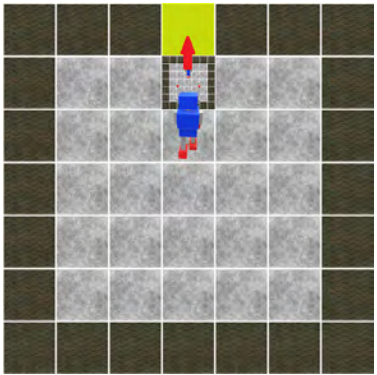
R	Box on row 2?	Exit on top?
0	✓	✓



- **Push the box to** $(2, 2)$ **instead**.
  Commands: `L` $\underbrace{\texttt{FF...F}}_{N-1 \text{ times}}$ `RFR` $\underbrace{\texttt{FF...F}}_{N-3 \text{ times}}$.
- Can we push the box to row 1?
  Yes, since the exit is always at row 1.
- How can we find the exit?
  Consider **brute force** (maybe)?
  Commands: $\underbrace{\texttt{FLFRFRFRFLFL...}}_{N-2 \text{ times}}$

The Box Pushing Mechanics – Subtask 2

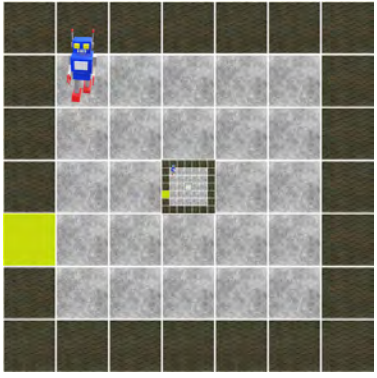
R	Box on row 2?	Exit on top?
0	✓	✓



- **Push the box to** $(2, 2)$ **instead**.
  Commands: `L` $\underbrace{\texttt{FF...F}}_{N-1 \text{ times}}$ `RFR` $\underbrace{\texttt{FF...F}}_{N-3 \text{ times}}$.
- Can we push the box to row 1?
  Yes, since the exit is always at row 1.
- How can we find the exit?
  Consider **brute force** (maybe)?
  Commands: $\underbrace{\texttt{FLFRFRFRFLFL...}}_{N-2 \text{ times}}$
- This gets you $\approx$ 17 points with `FLR` only!

The Box Pushing Mechanics – Normal Box

R	Box on row 2?	Exit on top?
0	✗	✗

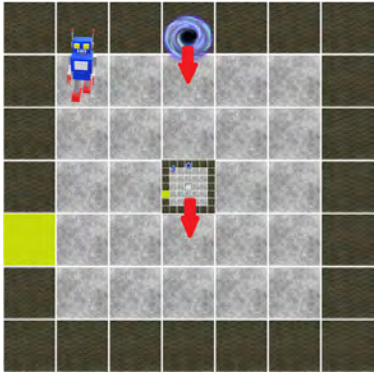


- Pushing to an edge is an **irrevertible action**.
- We should push the box **only** to the edge where the exit is located at.
- This means Robo must find the exit first before handling the box!
- However, the box can still be pushed towards the four directions, which means that Robo can push the box to $(2, 2)$ safely.

The Box Pushing Mechanics – Recurbox

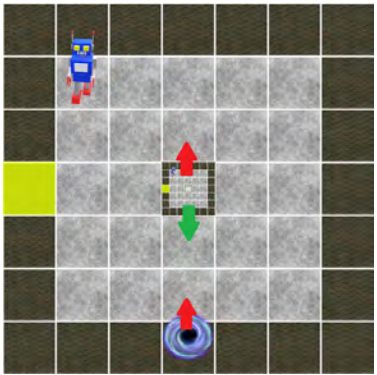
R	Box on row 2?	Exit on top?
1	✗	✗

- What happens if we use *recurbox* instead?
- Note that there is one direction that triggers a teleportation. Since the box's location remains unchanged, it is simply *impossible* to push the box towards that direction.
- This gives us a simple metric to determine impossibility.



The Box Pushing Mechanics – Recurbox

R	Box on row 2?	Exit on top?
1	✗	✗



- What if it is possible to solve? Can we adapt any solution from Subtask 3?
- Not really, it is not safe to push the box to $(2, 2)$ anymore.
- In the example, once we pushed the box down, it is impossible to push the box back up $\Rightarrow$ another example of **irrevertible action**!
- This means that we have to be extra careful – aside from determining which **edge** the exit belongs to, its **location** is now important too.

Table of Contents

The Instruction Set

Here are the instructions available:

Type	Mnemonic	Description
`F`	Forward	Move forward.
`L`	Turn Left	Turn left.
`R`	Turn Right	Turn right.
`BB`	Branch on Box	If Robo is blocked by the box, jump to `label`.
`BW`	Branch on Wall	If Robo is blocked by the wall, jump to `label`.
`J`	Jump	Jump to `label`.

Does that remind you of something? **MIPS** (or your favourite assembly language)!

We taught this in training!

In MIPS, loops are implemented using **branch** or **jump** instructions. Control flow is only controlled by a **Program Counter** (PC).

Type	Mnemonic	Operation
`beq`	Branch Equal	if (R[rs] == R[rt]) PC=PC+4+BranchAddr
`j`	Jump	PC=JumpAddr

```
while ($t0 < 10) {
    $t0 = $t0 + 1;
}
```

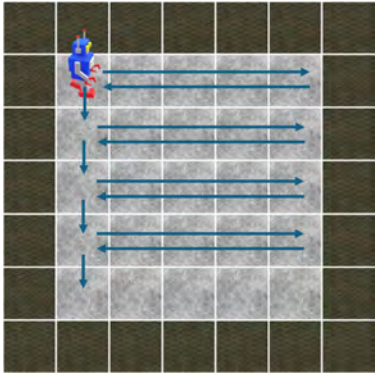
```
loop:  slti $t1, $t0, 10
       beq  $t1, $zero, end
       addi $t0, $t0, 1
       j    loop
end:   [more...]
```

This is same as the control instructions in IOI 2012 Pebbling Odometer.

Writing Loops

What does this entail? We can use **create our own loops** using branch/jump instructions.

Example 1: Write a loop to traverse through all cells in the grid!



Writing Loops

Example 1: Write a loop to traverse through all cells in the grid!

1. While we are not facing a wall, move forward.

```
loop-1:   BW end-1
          F
          J loop-1
```

2. Turn around.

```
end-1:    R
          R
```

3. While we are not facing a wall, move forward.

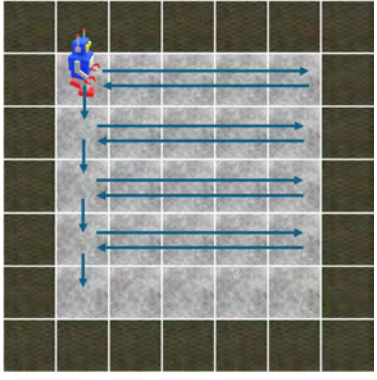
```
loop-2:   BW end-2
          F
          J loop-2
```

4. Move downwards.

```
end-2:    L
          F
```

5. Repeat this until we reach the bottom. **(Nested Loop)**

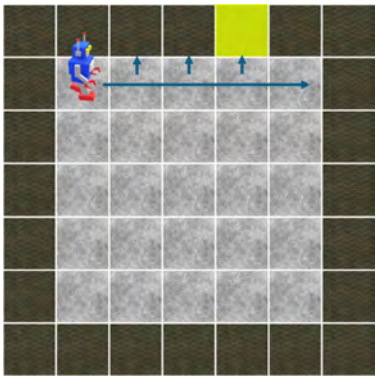
```
          BW done
          L
          J loop-1
done:     [more...]
```



Writing Loops

Example 2: Write a loop to find the exit!

- Suppose the exit is at the top row. Robo can iterate through the cells on the top row one by one, and turn to the top to check whether the wall exists.



Writing Loops

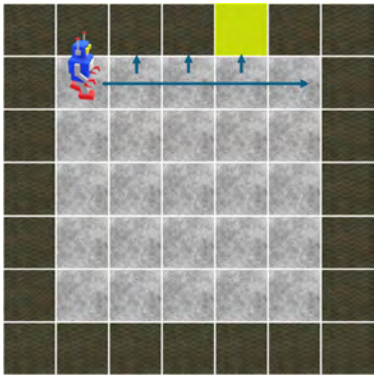
Example 2: Write a loop to find the exit!

1. Move forward and check whether there is a wall on the top (to Robo's left).

```
loop-1:   BW end-1
          F
          L
          BW ok-1
          J exit-found-1
ok-1:     R
          J loop-1
```

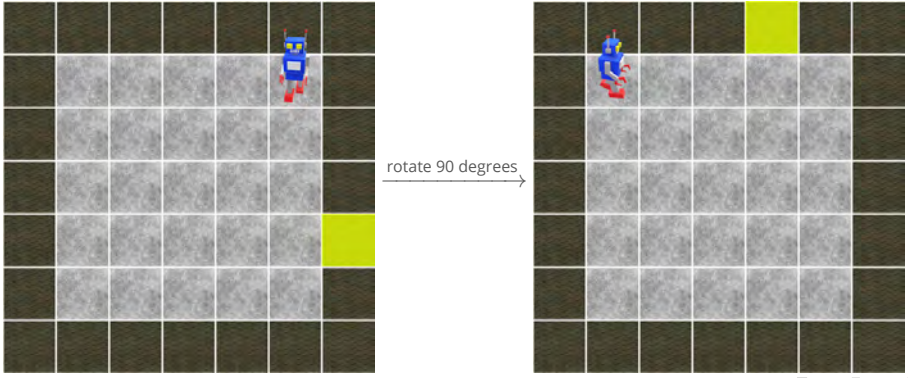
2. If we have reached the end, turn right and work on the next edge.

```
end-1:    R
loop-2:   [more...]
```



Writing Loops with the *Grid Rotation Technique*

- Do we really need 4 loops for the 4 edges? Not really!
- After exploring an edge, we (implicitly) **rotate the grid by 90 degrees** ⇒ Fresh state!



Writing Loops with the *Grid Rotation Technique*

Example 2: Write a loop to find the exit!

1. Handle a single edge.

```
loop-1:   BW end-1
          F
          L
          BW not
          J exit-found
not:      R
          J loop-1
```

2. Turn right and work on the next edge.

```
end-1:    R
          J loop-1
```

This loop always terminates as long as there is an exit.

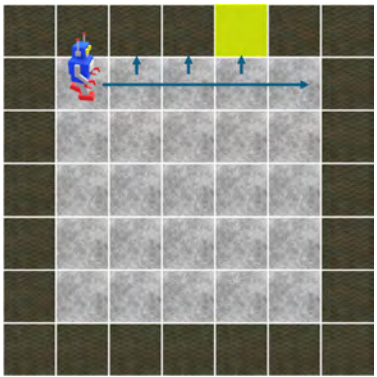


Table of Contents

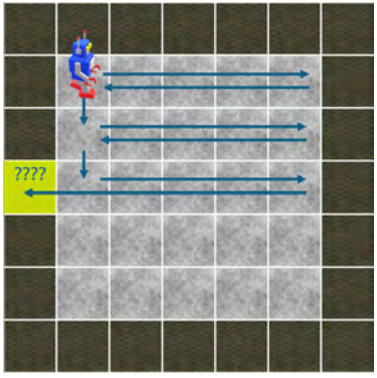
Piecing Everything Together

The solution should combine the following elements:

- Find the location of the box.
- Find the location of the exit.
- Push the box to the exit.

Should we find the box or the exit first?

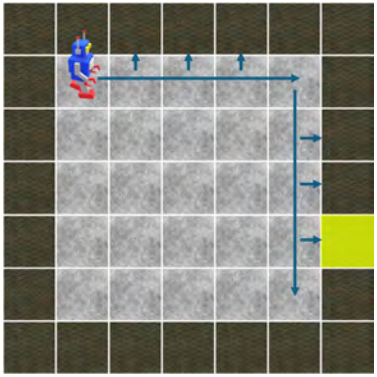
- Better to find exit first.
- When traversing the grid, the exit will create corner cases. But these can be eradicated if we are sure the exit is on the top!

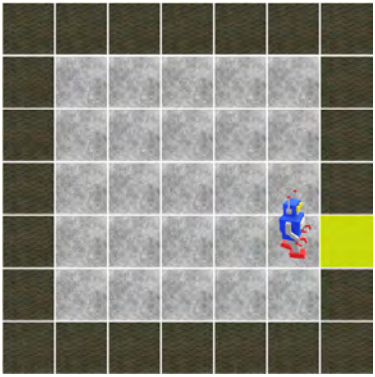


Piecing Everything Together

Step 1: Find the location of the exit using the grid rotation loop.

- We should be facing the exit when the loop breaks.

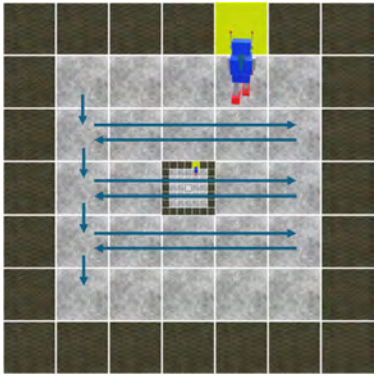


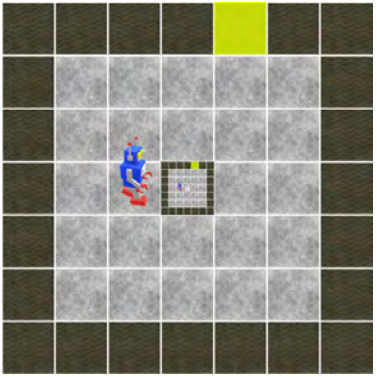


Piecing Everything Together

Step 2: Now we can assume the exit is at the top! Find the box by traversing all cells in the grid.

- Add a terminating condition (box detection) to the 2D loop.

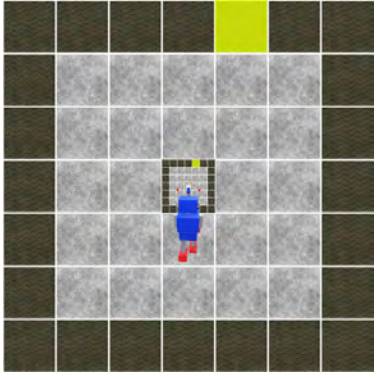


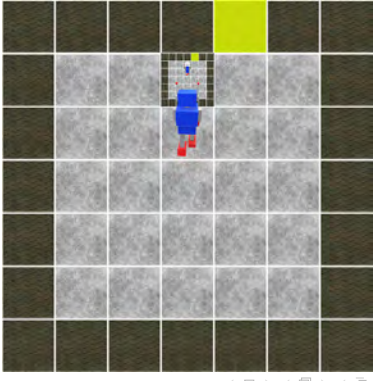


Pushing to the Exit

Step 3: Push the box to the exit. Hardest segment!

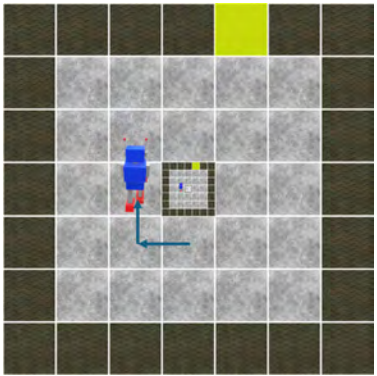
- Let's consider a simpler task: Push the box to the top. Is that easy?

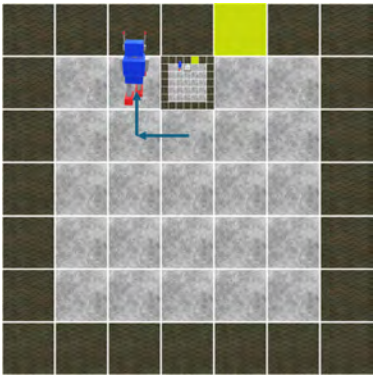




Pushing to the Exit

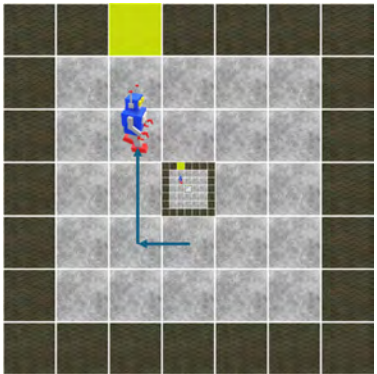
- For every step, we should check whether there is a wall ahead of the box (use the sides!).

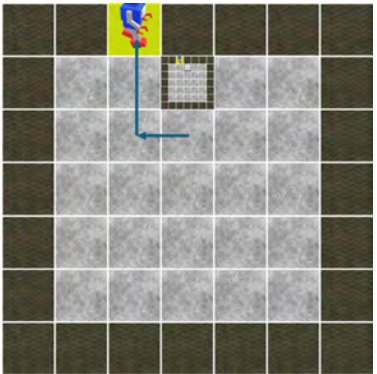




Pushing to the Exit

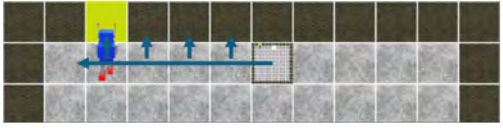
- What if the column on the left has the exit?
  - We get stuck indefinitely!
  - We will have to move one more step, and check if there is a wall on the right.



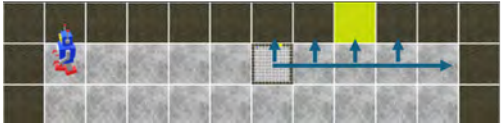


Pushing to the Exit

- Next, for the *recurbox*, we have to check whether the exit is to the left or to the right of the current location. (To prevent unnecessary teleportations!)
- If there is an exit to the left, try "brute force" pushing the box on all cells to the left. Otherwise, "brute force" to the right instead.



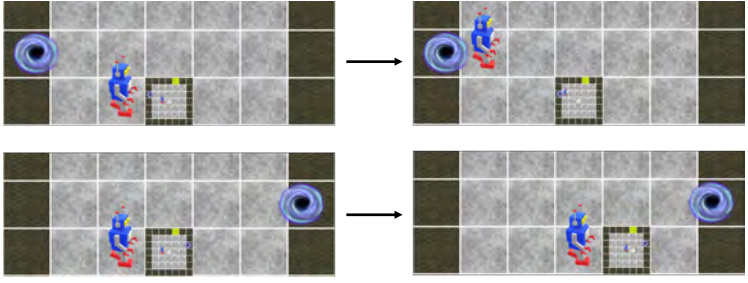
Exit to the left



Otherwise

Pushing to the Exit

- Finally... we still have to check whether a teleportation occurred due to the *recurbox*.
  - We only made necessary pushes. Hence, we can directly `DIE` at teleportation.
- Can we check if the box is still in front of Robo?



Pushing to the Exit

- We might still run into an infinite loop...

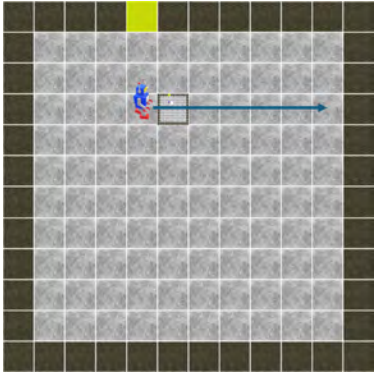
- Solution: Turn around to check whether we are behind a wall!
  (Reason: The black hole is always lying on an edge.)

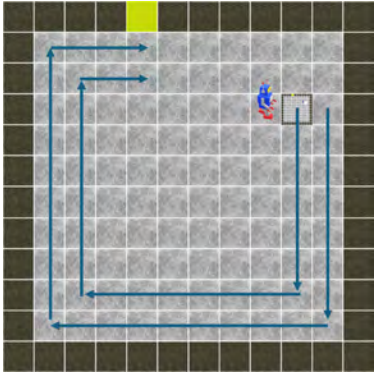
Pushing to the Exit

- While this solution works for the normal box, it uses many unnecessary instructions:
  - We push the box to the top, special handling the case where we crashed onto the exit.
  - We check whether the exit is to the left or right, and handle both differently.
- Observe that we can push the normal box elsewhere!
  - It's OK to do something stupid. We're optimizing for the number of **instructions**, not Robo's exeuction time.

Pushing to the Exit

Idea: First, push the box to the **right** instead (column $N-1$, avoiding irrevertible action). Then, bring the box & check for the exit as a **round trip** (using the grid rotation trick!).





Full Solution

Watch our full solution in action!
https://www.youtube.com/watch?v=TjMK2sP2efE

Takeaways

1. Don't be scared by unfamiliar task formats. There is still *some* basis for the task which you might have seen before, and the scoring is usually generous (don't get disadvantaged!).
2. Come up with as many solutions as you can! Although it might not give you any points instantly, the ideas might be useful in the future.
3. Upsolve contest tasks, similar things might appear anytime in the future!
4. Yes, takeaways are recurring (these 3 bullet points are same as last year's).