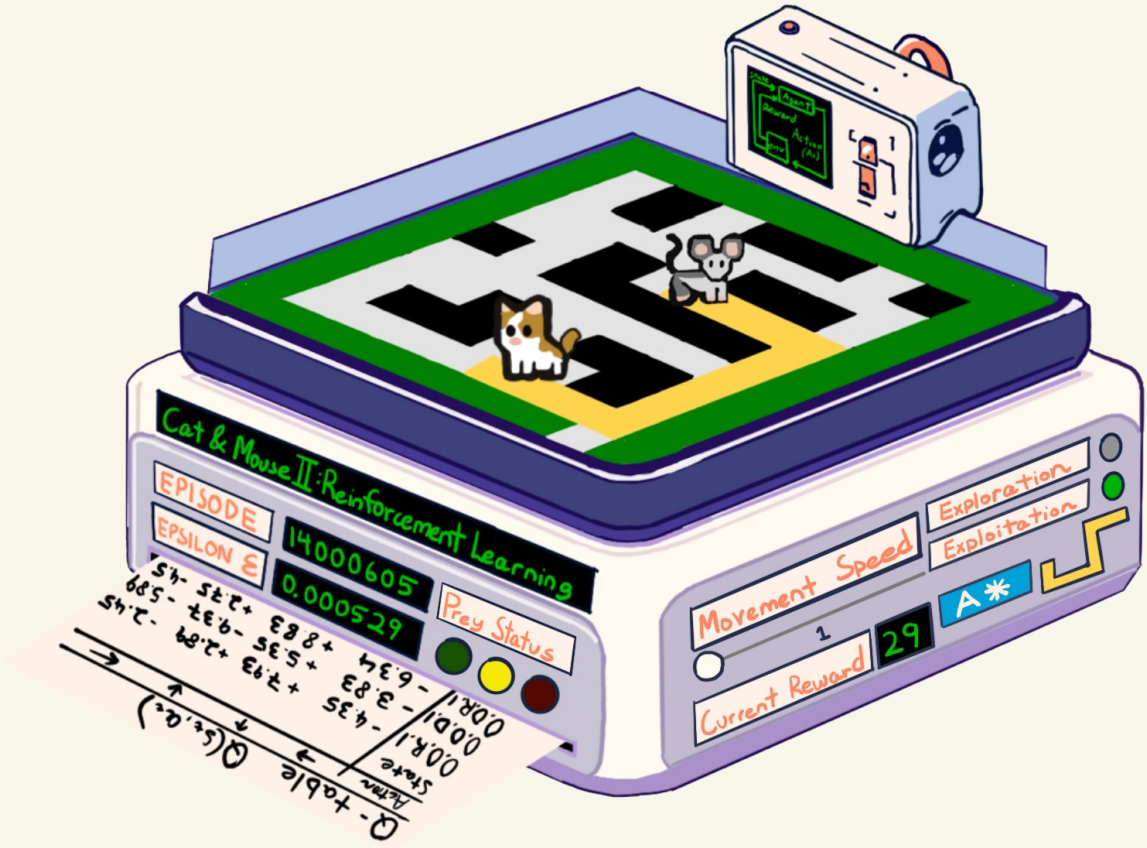
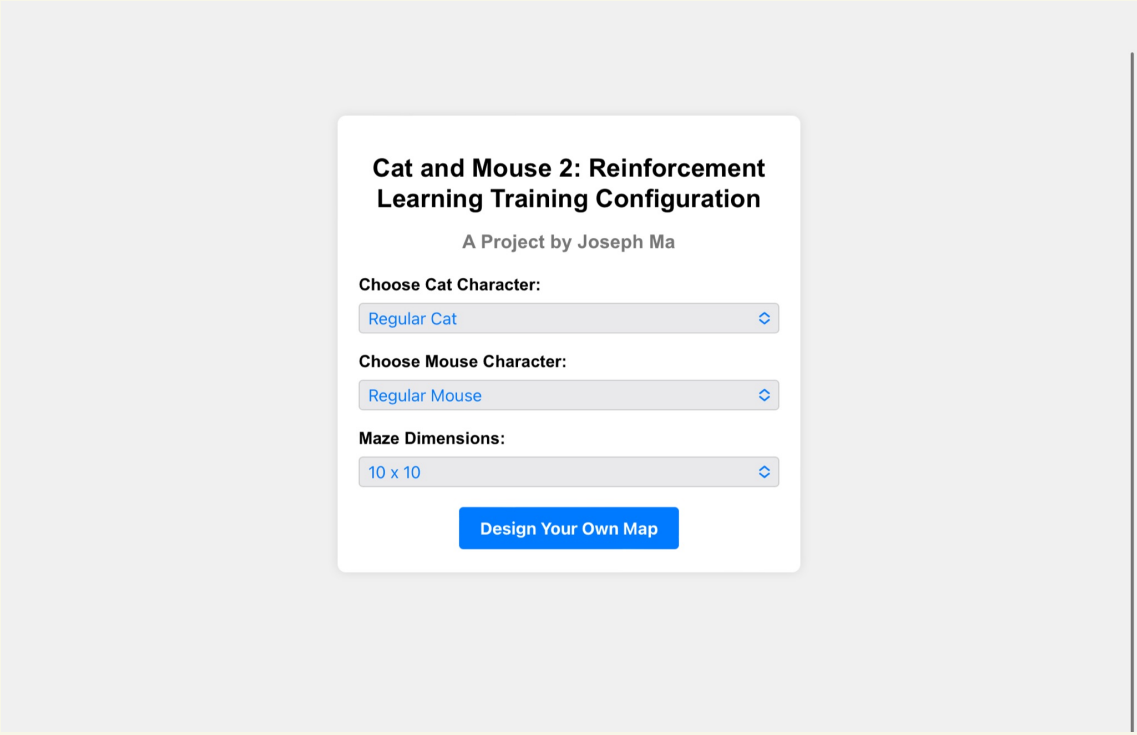
CAT AND MOUSE 2: Reinforcement Learning
A Project by Joseph Ma

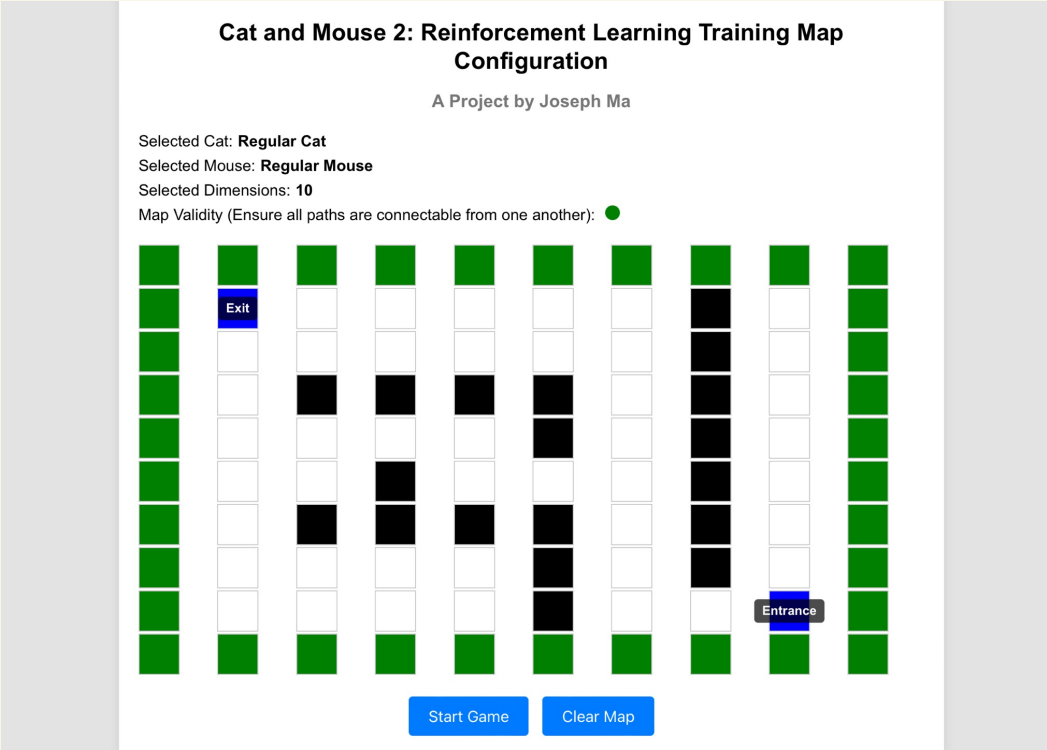
Full guide page is currently in the works- this document serves as a preliminary guide, providing an overview of the projects core concepts, logic and implementation.



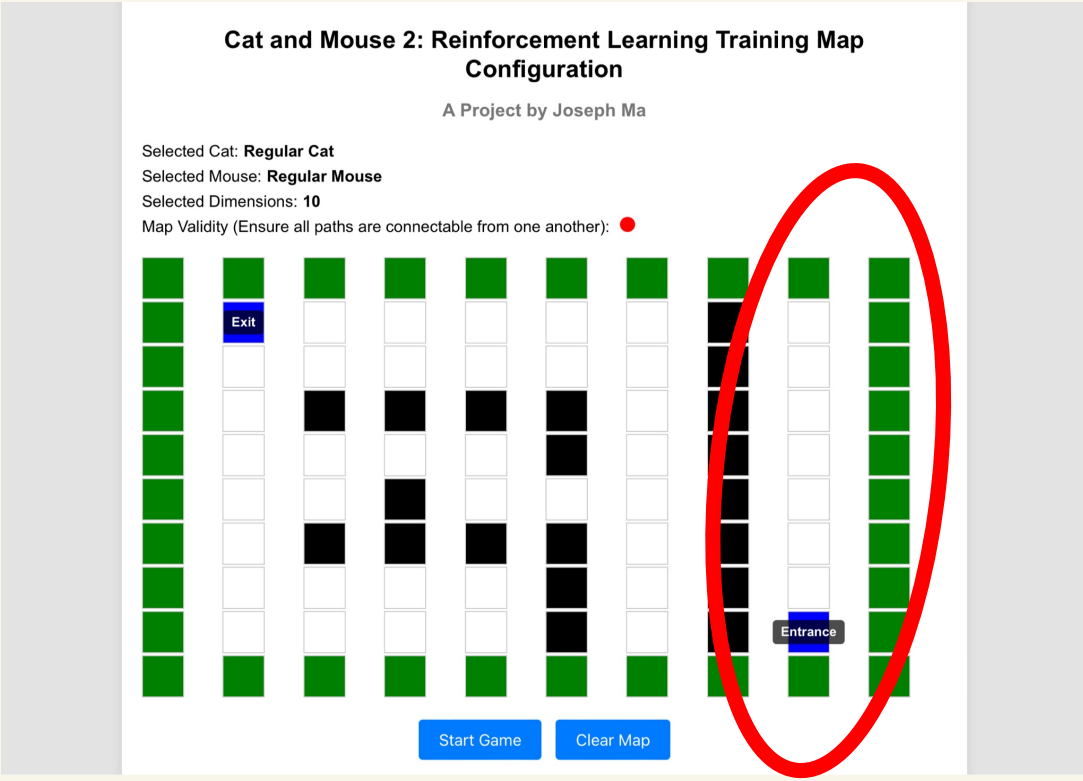
The first page allows you to select a character for training and select the map size. Note that larger map size means longer training time.



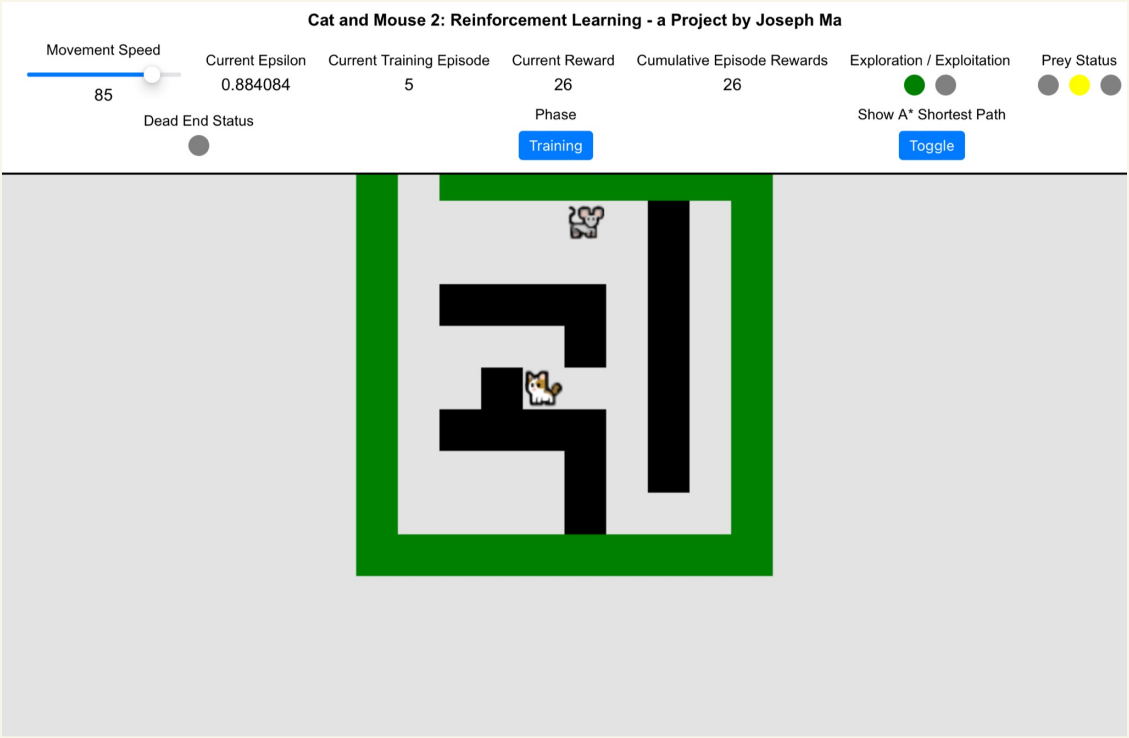
The next page allows you to design your own map for the training environment. Place walls down as you would like (in black).



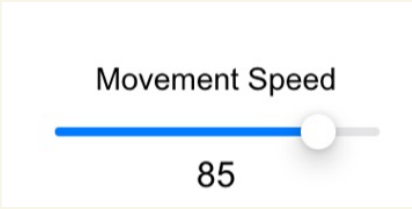
Note that if a portion of the map is disconnected from the others this will invalidate it. Ensure all paths are connectable from one another. There should also be at least one path from entrance to exit.



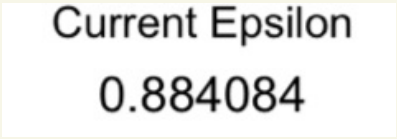
Let’s begin training



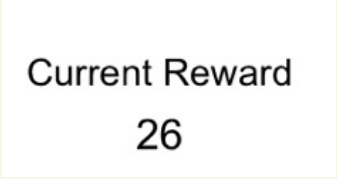
Adjust the speed at which the episodes play out



We use the greedy epsilon strategy, this is the chance the mouse chooses the best action to take. And 1 - epsilon the chance the mouse chooses a random action.



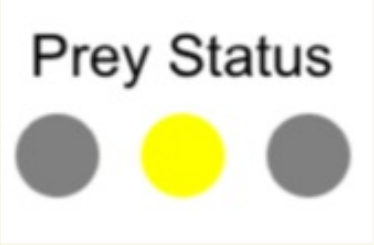
The current reward is a reflection of how “good” the mouse’s action is. The more positive the better, and the more negative the worse.



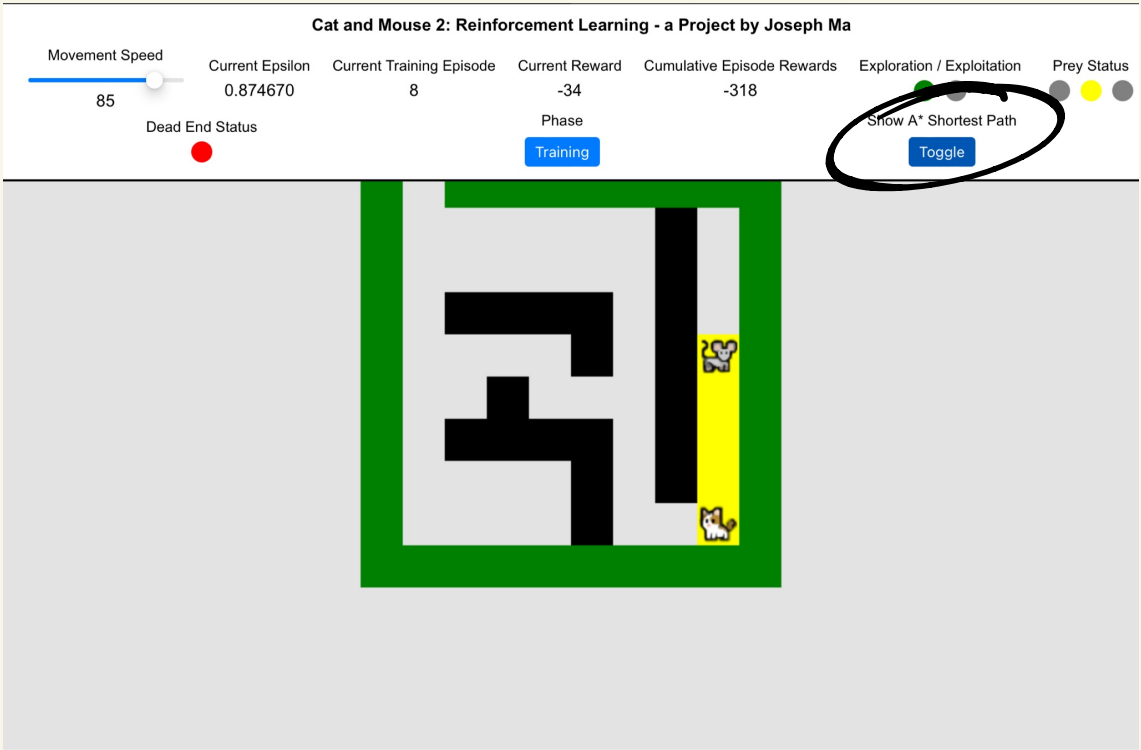
The total rewards accumulated through the current episode.

Indicates whether the most recent action by the mouse was exploration or exploitation

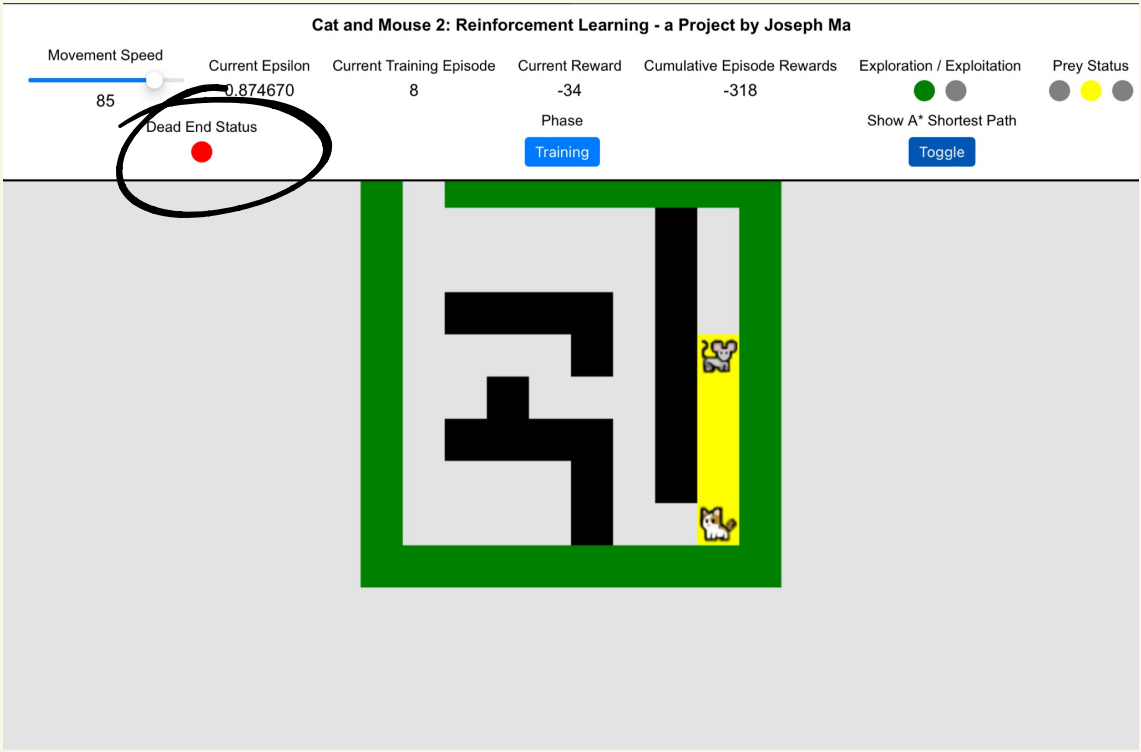
Prey status: Green will light up at the most left if mouse escapes, Yellow will light up in the middle of mouse is still alive, Red will light up at the most right if mouse gets caught by the cat



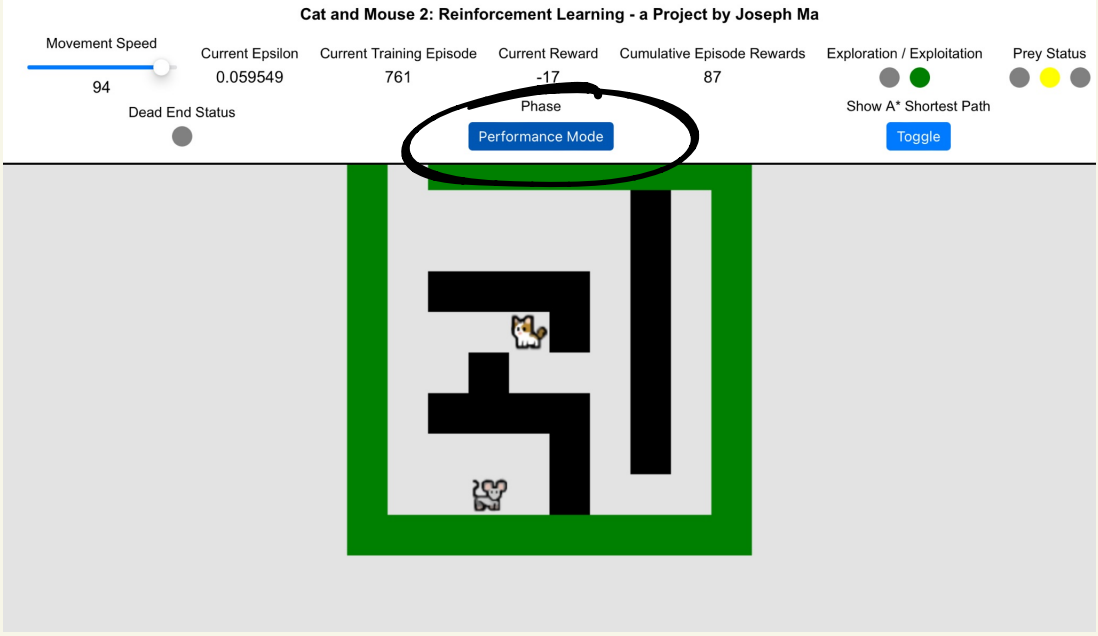
Click toggle to show the A* shortest path calculated by the cat to reach the mouse. The cat always takes the shortest path to the mouse.



If the mouse finds itself cornered by the cat, the dead end status will light up. A double DFS algorithm was written to check for this.



Once epsilon has decayed, toggle the training button. This will cause the mouse to take the best action in each move. In other words it will exploit every action. In this mode, the Qtable and epsilon will remain frozen until we return to regular training mode. This mode is to visualize how intelligent the mouse has gotten.



Reward function

NOTE: MAX_DISTANCE is defined as an estimate of the longest possible distance from one point of the grid to the other.

Condition (IF-ELSE-IF)	Reward	Explanation
Dead End: False Maintained Distance from Cat: True Got Closer to Exit: True	MAX_DISTANCE	The mouse maintains distance from the cat and gets closer to the exit.
Dead End: False Maintained Distance from Cat: False (The cat got closer by 1 unit) Got Closer to Exit: False Exit Distance to Mouse < Exit Distance to Cat	-(MAX_DISTANCE - DISTANCE_TO_CAT)	Given that the cat got closer by 1 unit, that implies the mouse has not made a meaningful move. Since the cat uses A*, a decrease of 1 unit between the cat and mouse means the mouse did not move while the cat did. The fact that the mouse is closer to the exit than the cat means it can move towards the exit without fear of getting caught.
Dead End: False Maintained Distance from Cat: True Got Closer to Exit: False (The direction the cat was previously coming from != The previous direction from mouse -> exit) OR (The new distance to exit < distance from cat to exit)	-(MAX_DISTANCE - DISTANCE_TO_CAT)	The mouse was close enough to the exit (that is it could have reached it before the cat), but it decided not to move towards it. Basically, the mouse is wasting time.
Dead End: False Maintained Distance from Cat: True Got Closer to Exit: False	MAX_DISTANCE / 2	The mouse maintains distance from the cat but gets further from the exit. In this case the cat was likely closer to the exit than the mouse was so it makes sense the mouse moves further away.
Dead End: False Got Closer to Exit: True Exit Distance to Mouse < Exit Distance to Cat	MAX_DISTANCE	The mouse is closer to the exit than the cat is. We can disregard the cat and move closer to the exit without worry.

Dead End: False Above conditions not met	-(MAX_DISTANCE - DISTANCE_TO_CAT)	The cat got closer to us. That is terrifying. 😕 😱
Dead End: True Maintained Distance from Cat: True	-(MAX_DISTANCE - DISTANCE_TO_CAT) - 0.5 * MAX_DISTANCE	The mouse is cornered in a dead end and moves inwards towards the dead end leading it closer to its inevitable death.
Dead End: True Maintained Distance from Cat: False (The cat got closer by 1 unit)	-(MAX_DISTANCE - DISTANCE_TO_CAT) - 0.5 * MAX_DISTANCE	Any time the cat got closer by unit this means the mouse is idle. A* guarantees the Cat will either stay the same distance or get closer. Mouse is waiting to get eaten.
Dead End: True Maintained Distance from Cat: False (The cat got closer by 2 units)	(MAX_DISTANCE - DISTANCE_TO_CAT) + 0.5 * MAX_DISTANCE	We might think that the cat getting closer by the maximum amount (2 units) is a bad thing. But when the mouse is in a dead end, this is the best thing it can do. Move towards the cat until we are out of the dead end!
We switched from Dead End: True -> Dead End: False	1.5 * MAX_DISTANCE	Well done! We successfully got out of the dead end.
We switched from Dead End: False -> Dead End: True	-1.5 * MAX_DISTANCE	This is not good. The mouse just wandered into a dead end.
Dead End: True We got Caught	0	If the mouse did all it could to escape (defined by the above conditions) sometimes getting caught is inevitable
Dead End: True We got Caught	-1.5 * MAX_DISTANCE	You dead 😕
Dead End: False We got Caught	1.5 * MAX_DISTANCE	Well done mouse escaped 🙂

The Markov Decision Process

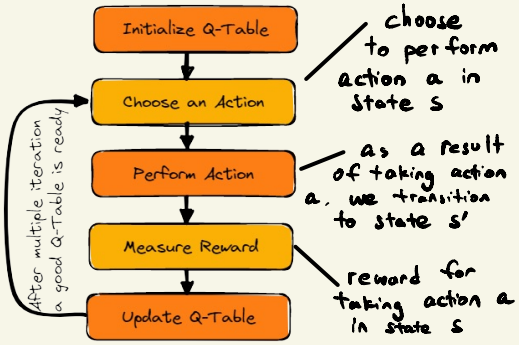
$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

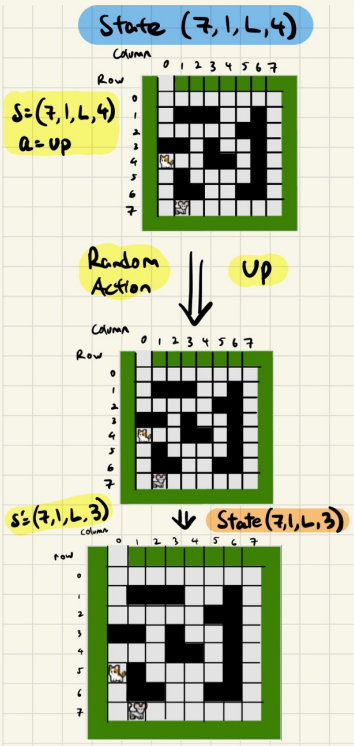
Old Q Value · Reward · Discount Rate (0 ~ 1) · New Q Value · Learning Rate (0 ~ 1) · Maximum Q value of transition destination state · TD error

possible actions

possible states	up	down	left	right
S0	Q(S0, up)	Q(S0, down)	Q(S0, left)	Q(S0, right)
S1	Q(S1, up)	Q(S1, down)	Q(S1, left)	Q(S1, right)
S2	Q(S2, up)	Q(S2, down)	Q(S2, left)	Q(S2, right)
⋮	⋮	⋮	⋮	⋮
Sn	Q(Sn, up)	Q(Sn, down)	Q(Sn, left)	Q(Sn, right)

Assuming n states





Qtable — Actions

States	up	down	left	right
0,0,R,1	+5	+3	-10	-13
0,0,R,2	+4.5	+3.3	-11	-15
0,0,R,3	+6.3	+2.7	-14	-14.5
⋮	⋮	⋮	⋮	⋮
7,1,L,1	-2.7	-1.8	-28.3	+12
7,1,L,2	-4.5	-3.9	-15.4	+11.5
7,1,L,3	-2.7	-4.4	-27.5	+9.3
7,1,L,4	-1.3	-4.5	-33.3	+3.7
⋮	⋮	⋮	⋮	⋮
7,2,L,1	-12.8	-14.3	-47.3	+17.4
7,2,L,2	-13.4	-13.5	-34.3	+15.7
7,2,L,3	-14.5	-13.8	-29.8	+16.3
7,2,L,4	-13.6	-14.5	-33.5	+16.5
7,2,L,5	-13.8	-14.3	-35.7	+18.4

$\gamma = 0.95$ $S=(7,1,L,4)$ $a=up$

$\gamma = 0.95$

$S'=(7,1,L,3)$

State	up	down	left	right
7,1,L,3	-2.7	-4.4	-27.5	+9.3

$\max_{a'} Q(s',a') = +9.3$

State	up	down	left	right
7,1,L,4	-1.3	-4.5	-33.3	+3.7

$Q(s,a) = -1.3$

Any other case	-(Maximum distance - distance from cat to mouse) = -(15 - 2) = -13

$r = -(15-3) = -12$

$Q(s,a) = (1-0.95)(-1.3) + 0.95(-12 + 0.95(9.3)) = -2.28$

State	up	down	left	right
7,1,L,4	-2.28	-4.5	-33.3	+3.7

$\gamma = 0.95$ $S=(7,1,L,4)$ $a=right$

$\gamma = 0.95$

$S'=(7,2,L,4)$

State	up	down	left	right
7,2,L,4	-13.6	-14.5	-33.5	+16.5

$\max_{a'} Q(s',a') = +16.5$

State	up	down	left	right
7,1,L,4	-1.3	-4.5	-33.3	+3.7

$Q(s,a) = +3.7$

Distance from cat to mouse stays the same, mouse gets closer to exit	Maximum Distance = 15

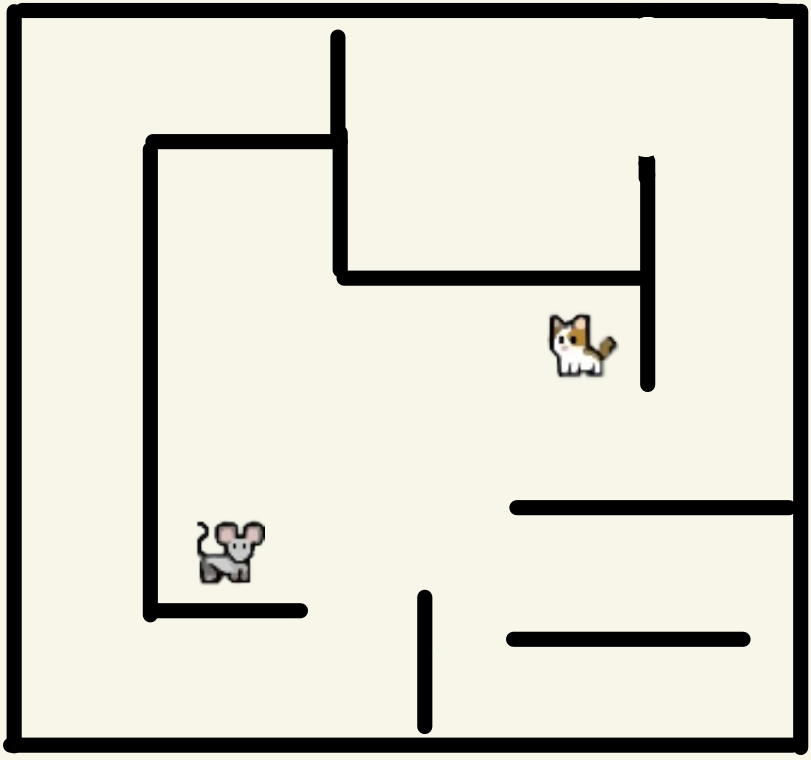
$r = 15$

$Q(s,a) = (1-0.95)(+3.7) + 0.95(15 + 0.95(16.5)) = 29.33$

State	up	down	left	right
7,1,L,4	-1.3	-4.5	-33.3	+29.33

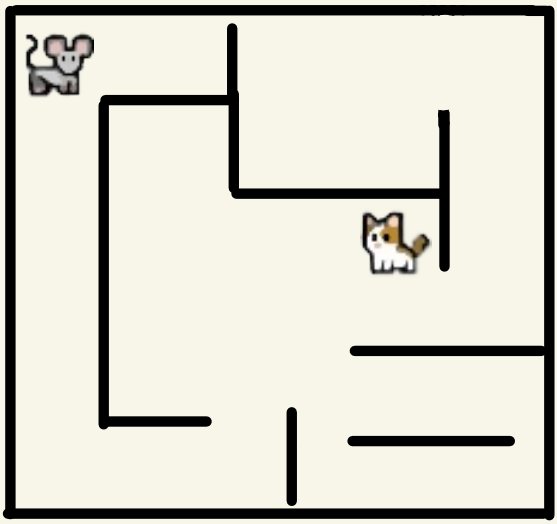
The Double DFS End Algorithm

What constitutes as a dead end?- that is any spot where we are truly cornered and will have limited mobility (just one direction). The concept seems subjective as it may depend on both the location of the cat/mouse and orientation of the maze. Here we go in depth into the double DFS dead end algorithm I came up with.

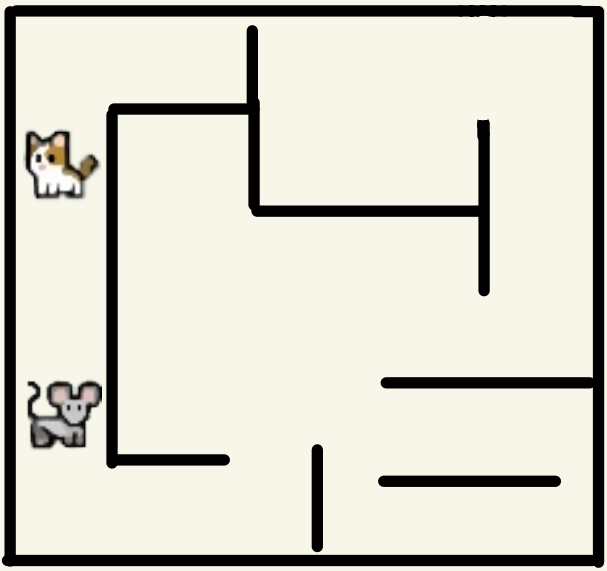


Whether we are in a dead end may also depend on our relative position to the cat

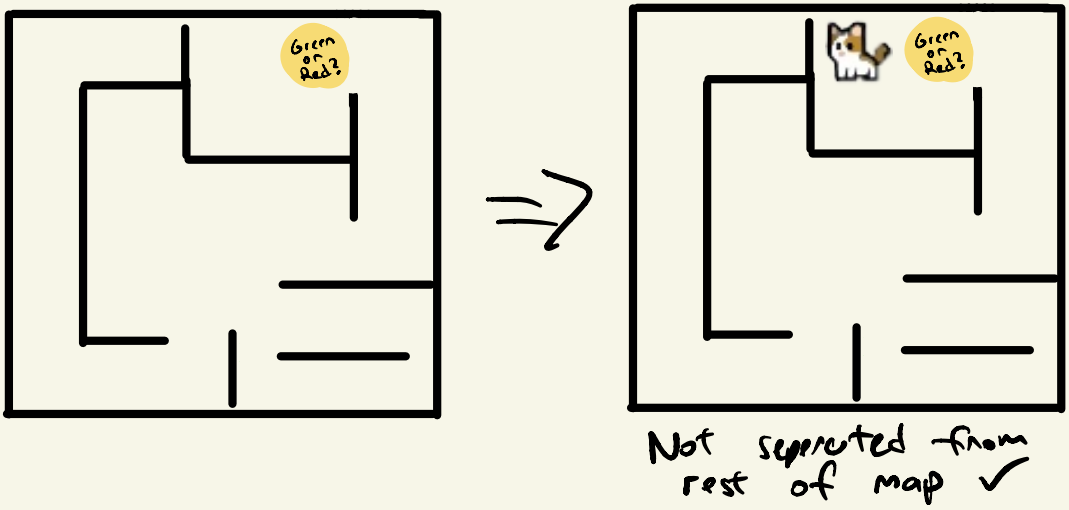
Dead end: true

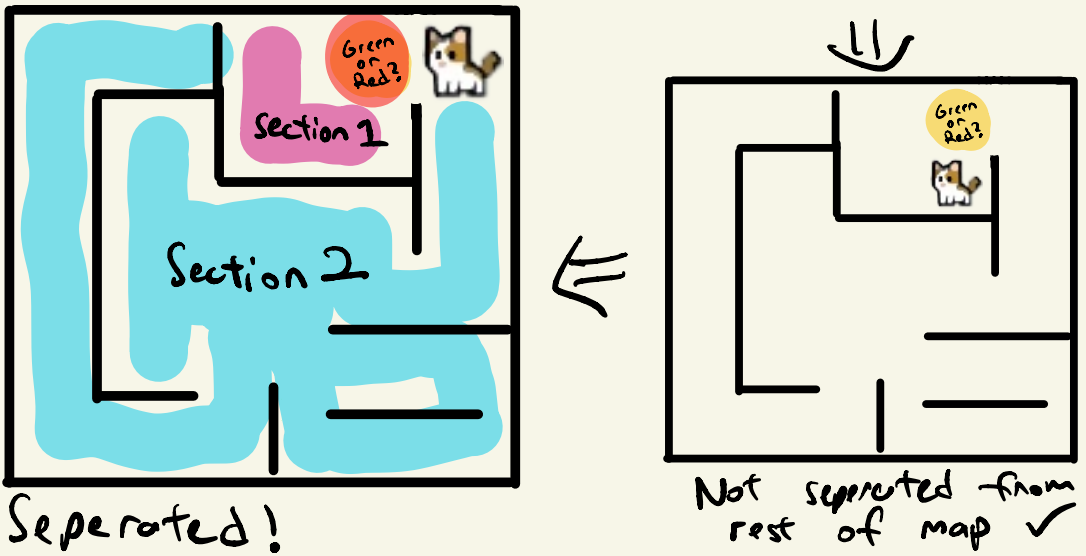


Dead end: false

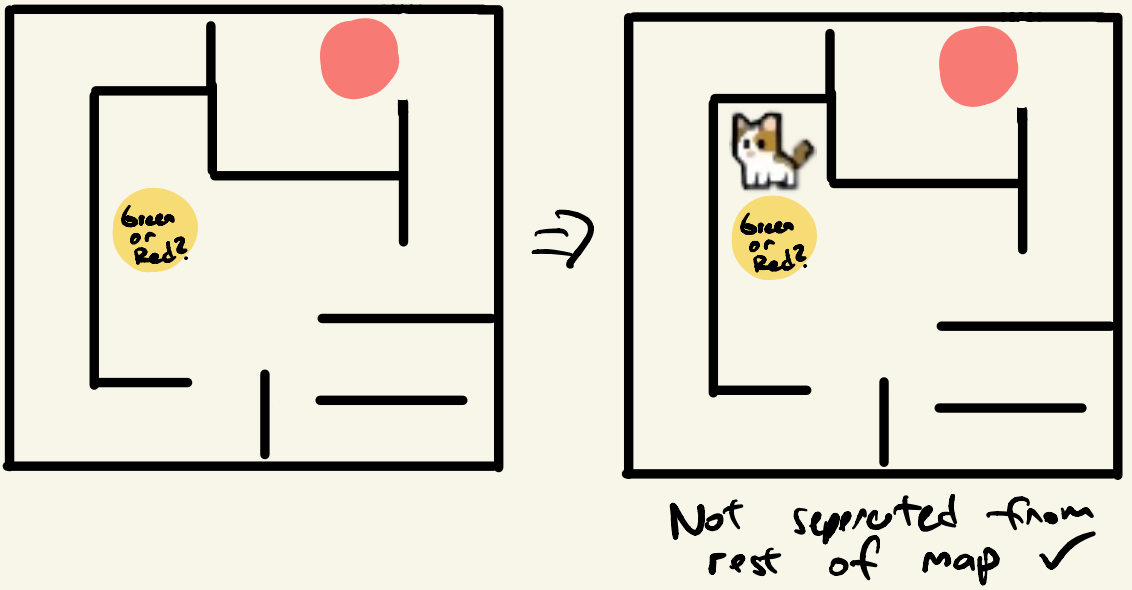


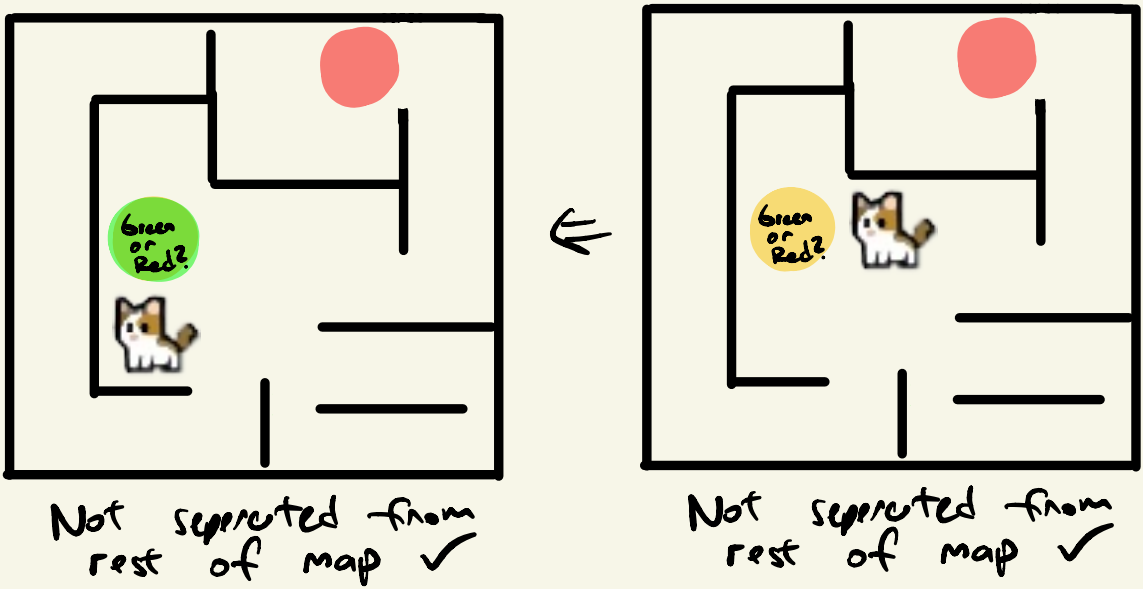
First step of the dead end algorithm: we first highlight the dots where we look in all 4 directions, placing an imaginary cat right next to that point. If the imaginary cat separates us from the rest of the map in any 4 of those points, we mark it red. Otherwise we mark it green.



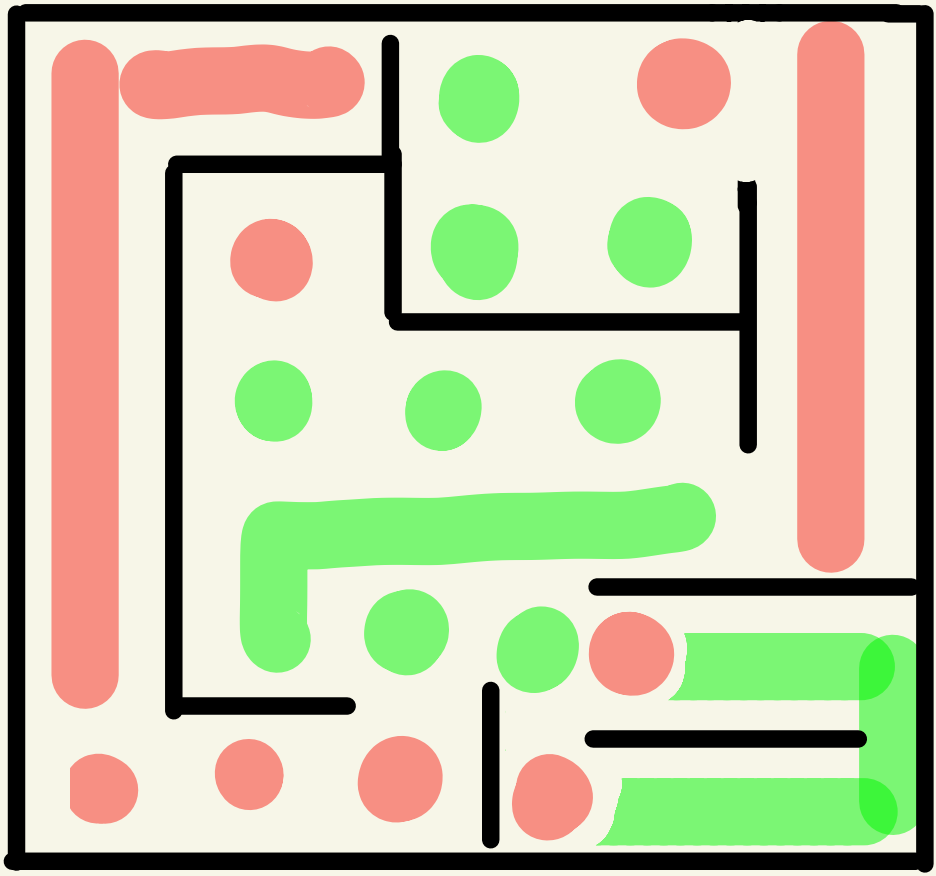


Let's do one more example!



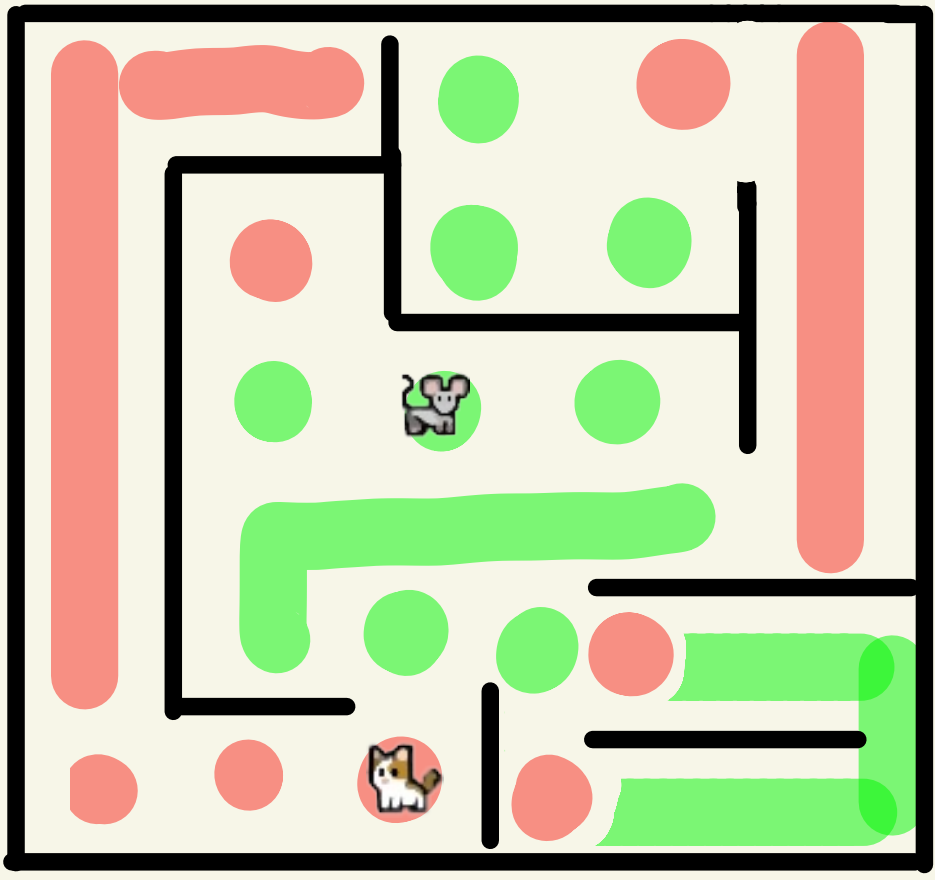


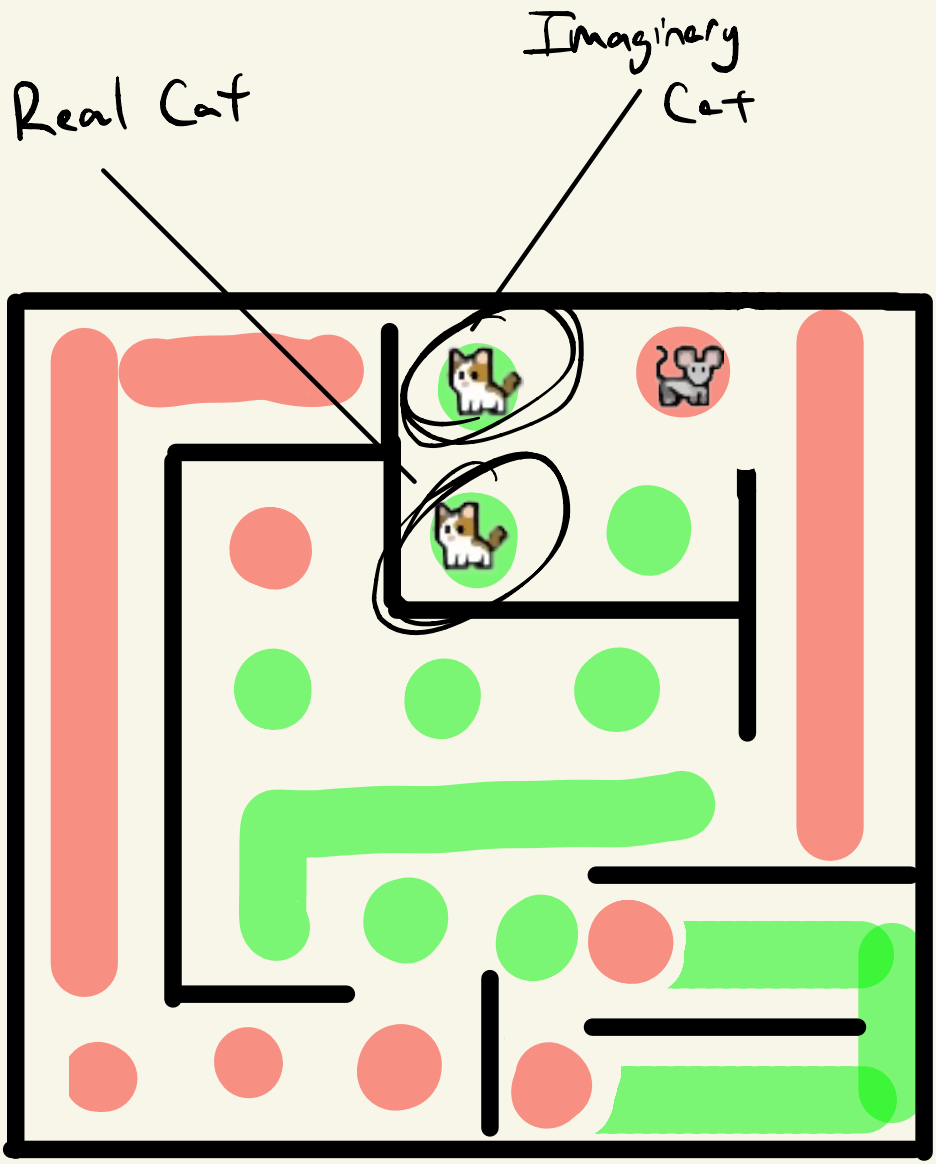
Result:



Second step of the dead end algorithm: for each evaluation of whether the mouse is in a dead end, if the mouse is on a green tile, we are NOT in a dead end. If the mouse is on a red tile we place an imaginary cat right next to the mouse (that is an imaginary cat from the direction the cat is coming from). If the imaginary cat does not separate us from the rest of the map, we are NOT in a dead end. If the imaginary cat separates us from the rest of the map, check whether we are connected to any green tiles. If we are connected to green tiles, we are NOT in a dead end. Otherwise if we are only connected to red tiles we are in a dead end.

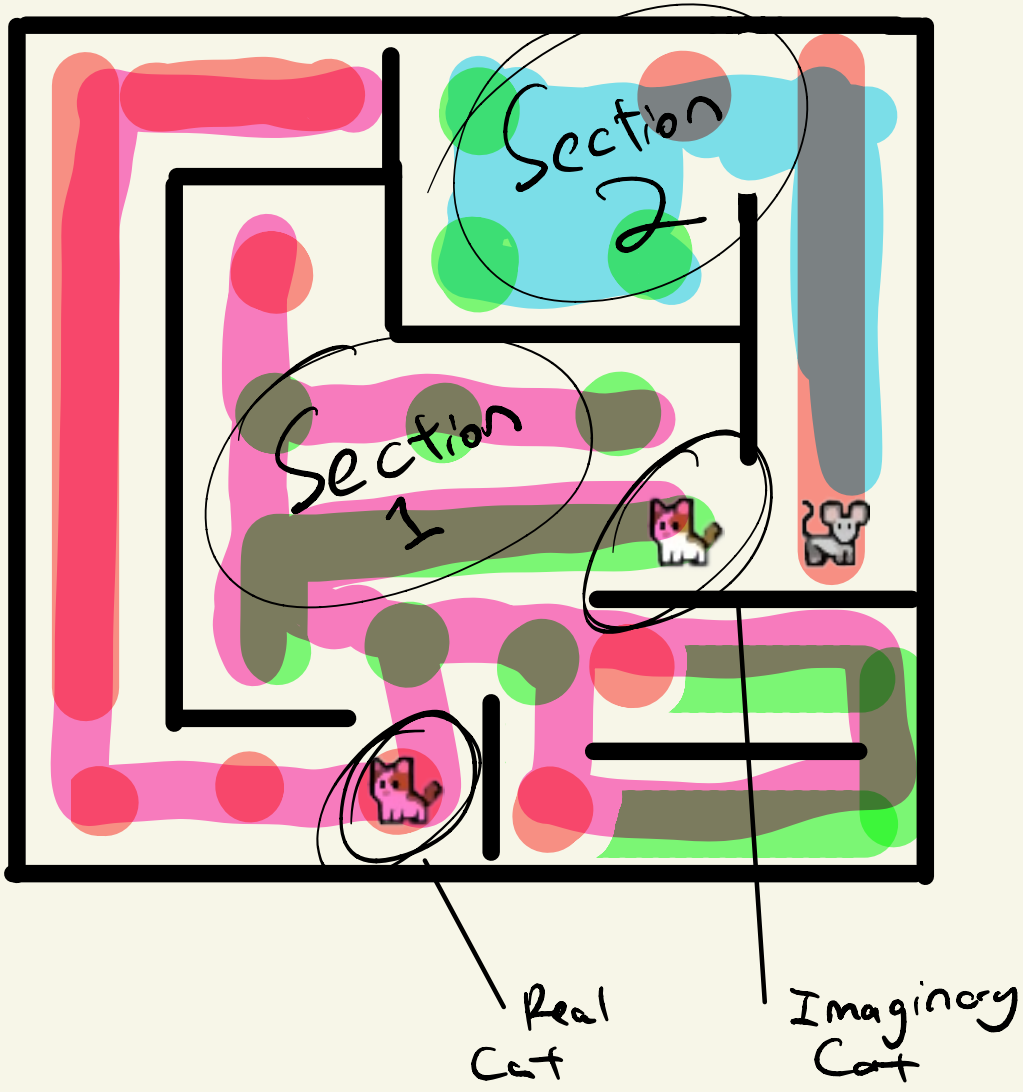
We are on a green tile. DEAD END: TRUE





Since the mouse is not separated from the rest of the map due to the imaginary cat- DEAD END: FALSE

The imaginary cat separates us from the rest of the map but we are connected to green tiles-
DEAD END: FALSE



The imaginary cat separates us from the rest of the map AND we are NOT connected to green tiles- DEAD END: TRUE