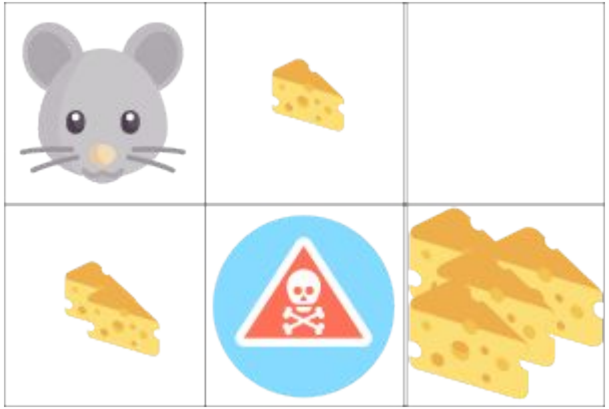
A Parallel Implementation of Q-learning

Yumeng Pan, Qianqian Guo

Problem Definition

- Brute force
  - Very time consuming for large maze
- Randomly choose actions
  - Stuck into infinite loop

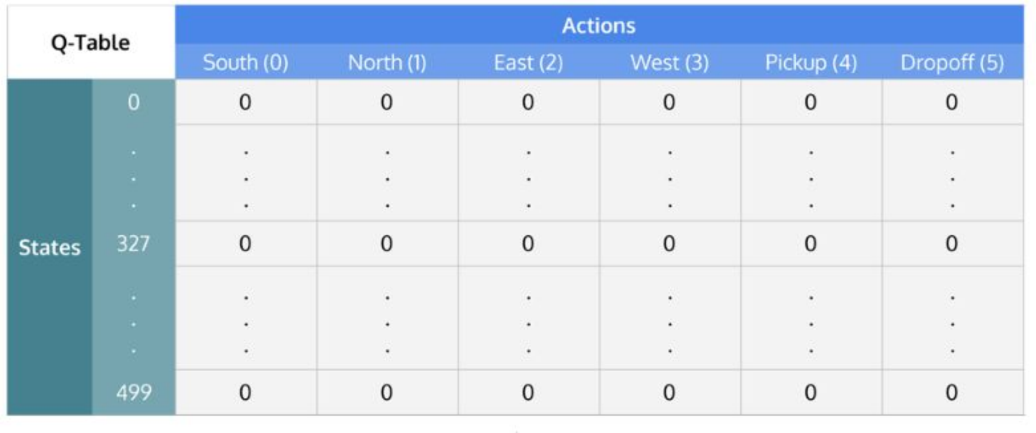


Introduction

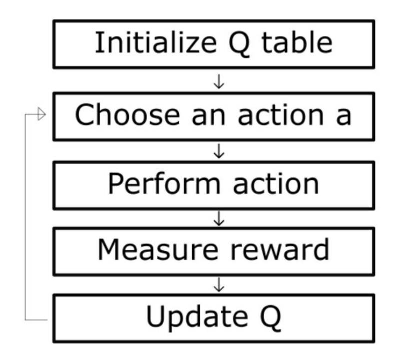
- ## What is Q-learning?

Q-Learning is a value-based Reinforcement Learning algorithm that deals with the problem of learning to control autonomous agents. The learning process works based on interactions by trial and error with a dynamic environment which provides reward signals for each action the agent executes.

- ## How does it work?



Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	327	0	0	0	0	0	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	499	0	0	0	0	0	0



Initialize Q table

Choose an action a

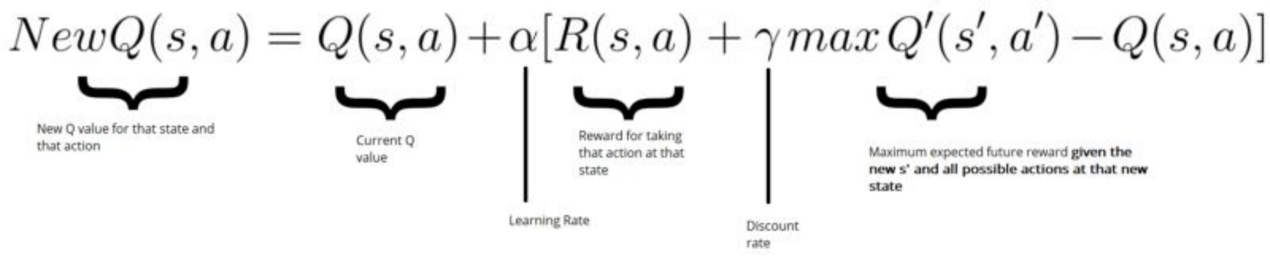
Perform action

Measure reward

Update Q

Introduction (continued)

- How to update Q-Table?



$$NewQ(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma\, max\, Q'(s',a') - Q(s,a)]$$

New Q value for that state and that action

Current Q value

Reward for taking that action at that state

Maximum expected future reward **given the new s' and all possible actions at that new state**

Learning Rate

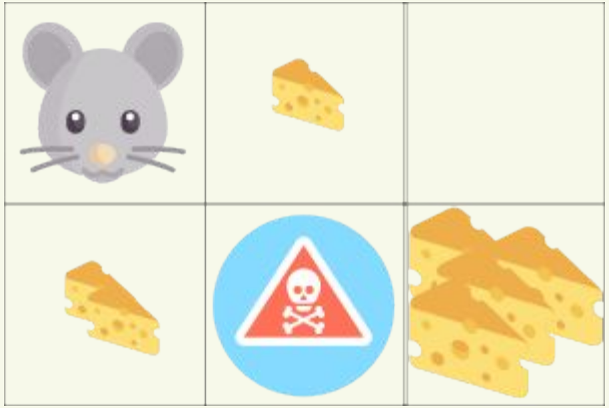
Discount rate

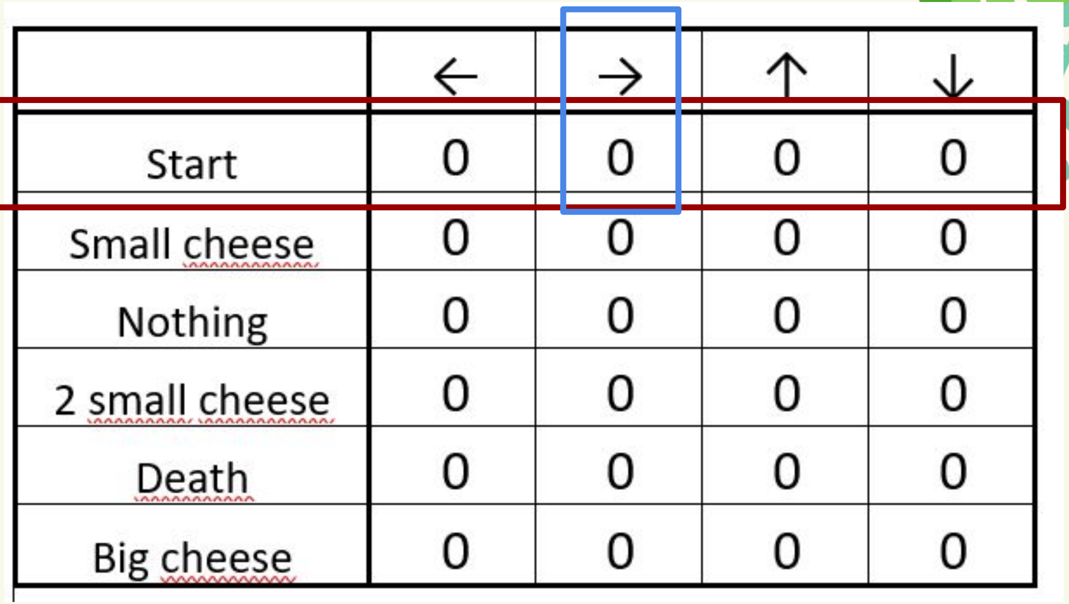
s: Sate.   a: Action.   r: Reward.

alpha: Learning rate parameter.

gamma: Decay rate (future reward discount) parameter.

Example





	←	→	↑	↓
Start	0	0	0	0
Small cheese	0	0	0	0
Nothing	0	0	0	0
2 small cheese	0	0	0	0
Death	0	0	0	0
Big cheese	0	0	0	0

6 States, 4 Actions

The Parallel Implementation

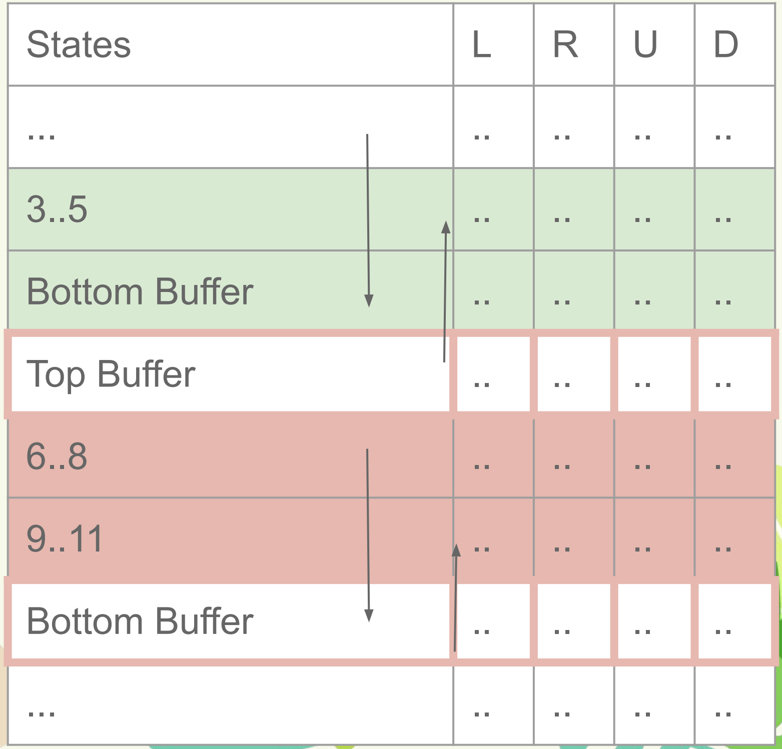
We used OpenMPI to parallel the Q-Table.

Maze

8x3

0	1	2
3	4	5
6	7	8
9	10	11
12	13	14
15	16	17
18	19	20
21	22	23

Q-Table



States	L	R	U	D
...	..	..	..	..
3..5	..	..	..	..
Bottom Buffer	..	..	..	..
Top Buffer	..	..	..	..
6..8	..	..	..	..
9..11	..	..	..	..
Bottom Buffer	..	..	..	..
...	..	..	..	..

The Parallel Implementation (continued)

For each episode…

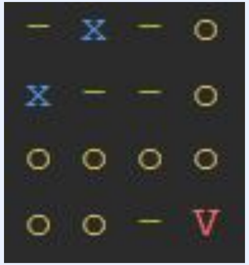
1. Reach max step
2. Reach end point
3. Fail
4. Reach boundaries (but…)

…	…	…	…
	👀		💀
	💀		
		😃	
👀			
…	…	…	…

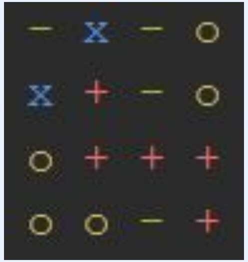
What's the point of parallel if communication is needed for each episode?

Update buffers once in a while! (e.g. every 100 episodes)

Results

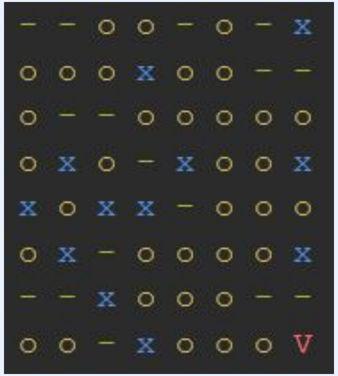


```
-  x  -  o
x  -  -  o
o  o  o  o
o  o  -  V
```

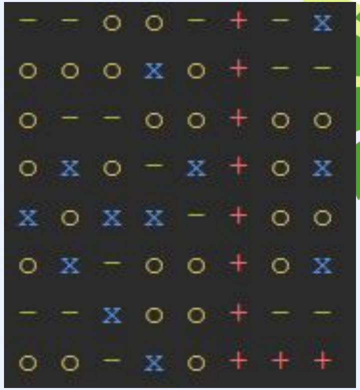


```
-  x  -  o
x  +  -  o
o  +  +  +
o  o  -  +
```

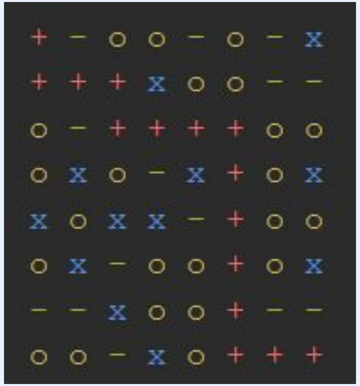
4x4 Maze



```
-  -  o  o  -  o  -  x
o  o  o  x  o  o  -  -
o  -  -  o  o  o  o  o
o  x  o  -  x  o  o  x
x  o  x  x  -  o  o  o
o  x  -  o  o  o  o  x
-  -  x  o  o  o  -  -
o  o  -  x  o  o  o  V
```



```
-  -  o  o  -  +  -  x
o  o  o  x  o  +  -  -
o  -  -  o  o  +  o  o
o  x  o  -  x  +  o  x
x  o  x  x  -  +  o  o
o  x  -  o  o  +  o  x
-  -  x  o  o  +  -  -
o  o  -  x  o  +  +  +
```

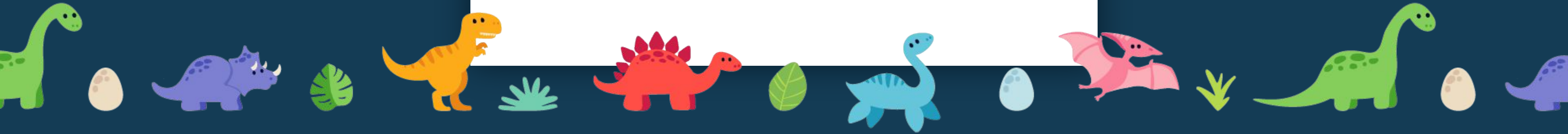


```
+  -  o  o  -  o  -  x
+  +  +  x  o  o  -  -
o  -  +  +  +  +  o  o
o  x  o  -  x  +  o  x
x  o  x  x  -  +  o  o
o  x  -  o  o  +  o  x
-  -  x  o  o  +  -  -
o  o  -  x  o  +  +  +
```

8x8 Maze

-	no reward	o	small reward
x	trap	V	end of maze

DEMO



Conclusion

Serial vs Parallel Q-learning:

- Parallelization does not seem to improve the calculation time😭
- For small number of episodes and large maze, parallel q-learning is more global 😉