CS143: Artificial Intelligence

V table

```
Updating value function V(s) for iteration=5
===============================================
 0.00  0.66  0.73  0.81
 0.66  0.00  0.81  0.90
 0.73  0.00  0.00  1.00
 0.81  0.90  1.00  0.00
```

Q table

```
LEFT   DOWN   RIGHT  UP
[ 0.00,  0.00,  0.00,  0.00, ]
[ 0.00,  0.00,  0.66,  0.00, ]
[ 0.00,  0.73,  0.73,  0.66, ]
[ 0.66,  0.81,  0.73,  0.73, ]
[ 0.00,  0.66,  0.00,  0.00, ]
[ 0.00,  0.00,  0.00,  0.00, ]
[ 0.00,  0.00,  0.81,  0.66, ]
[ 0.73,  0.90,  0.81,  0.73, ]
[ 0.66,  0.73,  0.00,  0.00, ]
[ 0.00,  0.00,  0.00,  0.00, ]
[ 0.00,  0.00,  0.00,  0.00, ]
[ 0.00,  1.00,  0.90,  0.81, ]
[ 0.73,  0.73,  0.81,  0.66, ]
[ 0.73,  0.81,  0.90,  0.00, ]
[ 0.81,  0.90,  1.00,  0.00, ]
[ 0.00,  0.00,  0.00,  0.00, ]
```

Value Iteration Algorithm
on Grid environments

Drake
UNIVERSITY

Recap: What is the MDP for Frozen Lake?

- The transition function encodes the movements in the Lake
  - In Frozen Lake, model or dynamics is known
  - $P(s'|a,s)$ is read as agent transitions from state s to s' by taking action a

- How many states are there?
  - How can you show the probability transitions $P(s'|a,s)$ from each state to the next?

- Let's simplify and consider a simpler problem where there only a couple of states
  - Let's consider a simple dice game with 2 states only

Recap: MDP for a Simple Dice

- You choose stay or quit
  - If quit, you get $10 and we end the game
  - If stay, you get $4 and then I roll a 6-sided dice
    - If the dice results in 1 or 2, we end the game
    - Otherwise, if the dice results in 3, 4, 5, or 6, we continue to the next round

WHAT IS THE BEST STRATEGY FOR THIS GAME?

stay

Expected rewards:

12.0

quit

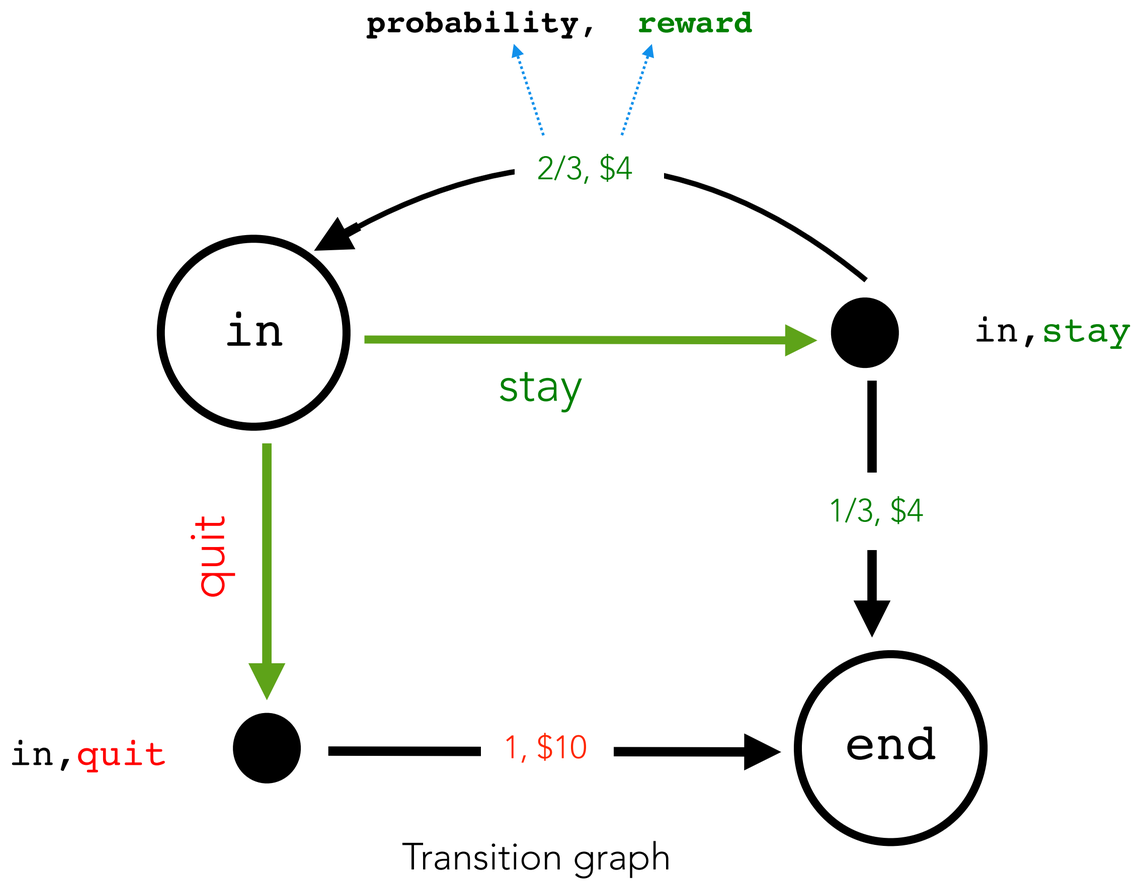
Expected rewards:

10.0

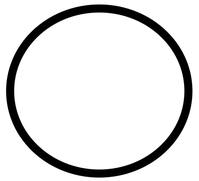
If you quit, then you'll get a reward of $10. Therefore, the stay strategy is preferred, even though sometimes you'll get less than $10.

Recap: MDP for a Simple Dice Game

- Let us formalize the dice game as a Markov decision process (MDP)



Transition graph



state node

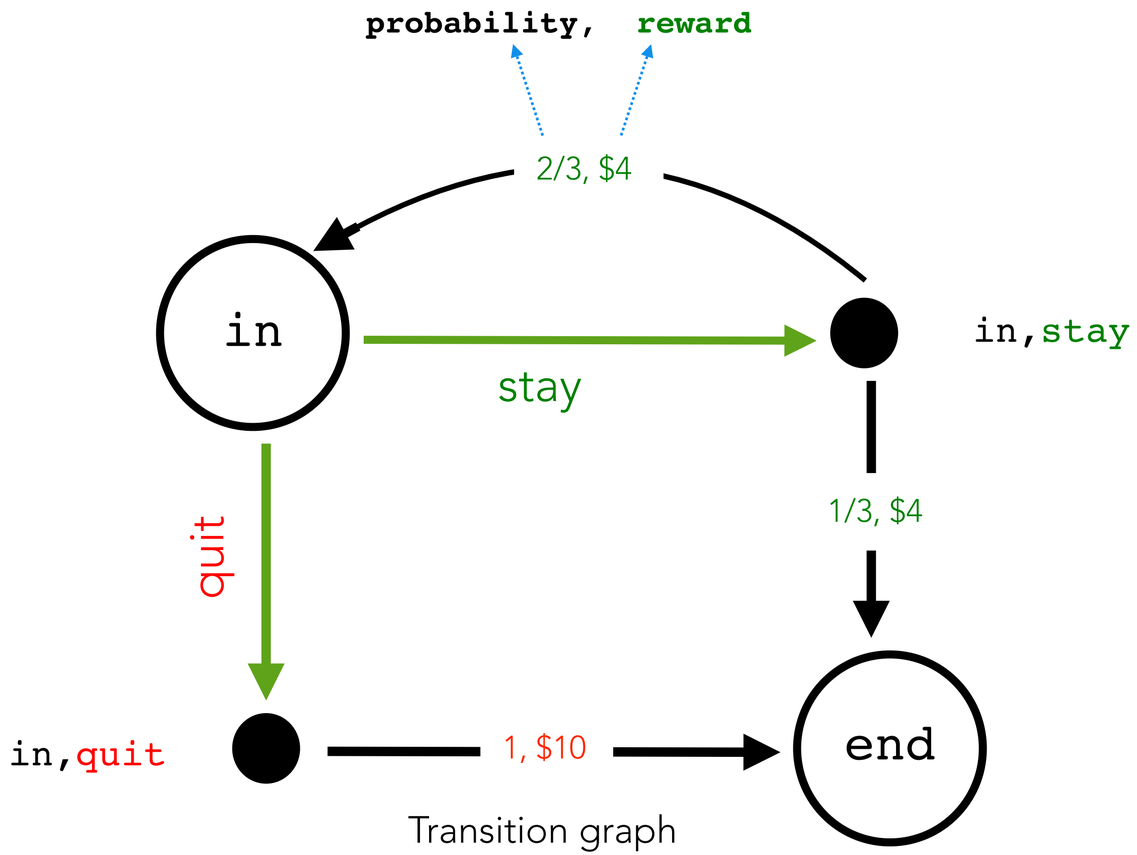
- Edges coming out of a state node are the possible actions from that state node leading to a action node

action node

- Edges coming out of a action node are the possible random outcomes of that action, which end up back in state nodes
- Sum of probabilities coming out of a action node is 1.0

Recap: Optimal action-value or q-value function

- **q-value function** $Q_{opt}(s, a)$: estimate of the expected return obtained by taking action ***a*** in state ***s***, and thereafter following the optimal policy π_{opt} optimally thereafter



Transition graph

Recap: Relationship: value and q-value functions

- Looks like from the previous to visualizations, the relationship between the **value function** and the **Q-value** function can be expressed recursively. For example, from the previous tree graph, it can be seen that the value function can be computed by taking the maximum over multiple Q-values corresponding to different actions. Mathematically, this can be written as follows:

$$V_{opt}(s) = \max \begin{Bmatrix} Q_{opt}(s, quit) \\ Q_{opt}(s, stay) \end{Bmatrix}$$

THERE IS ONLY ONE NON-TERMINAL STATE: in

$V_{opt}(in) = ?$

TERMINAL STATE: end

$V_{opt}(end) = 0$

$$V_{opt}(in) = \max \begin{Bmatrix} P(in \mid s, quit)[R(s, quit, in) + \gamma V_{opt}(in)] + P(end \mid s, quit)[R(s, quit, end) + \gamma V_{opt}(end)] \\ P(in \mid s, stay)[R(s, stay, in) + \gamma V_{opt}(in)] + P(end \mid s, stay)[R(s, stay, end) + \gamma V_{opt}(end)] \end{Bmatrix}$$

$$= \max \begin{Bmatrix} P(end \mid s, quit)[R(s, quit, end) + \gamma V_{opt}(end)] \\ P(in \mid s, stay)[R(s, stay, in) + \gamma V_{opt}(in)] + P(end \mid s, stay)[R(s, stay, end) + \gamma V_{opt}(end)] \end{Bmatrix}$$

$$\frac{2}{3}(4 + \gamma V_{opt}(in)) + \frac{1}{3}(4 + \gamma V_{opt}(end))$$

p(s'|s,a) r(s,a,s') discount p(s'|s,a) r(s,a,s') discount

$Q_{opt}(in, stay)$

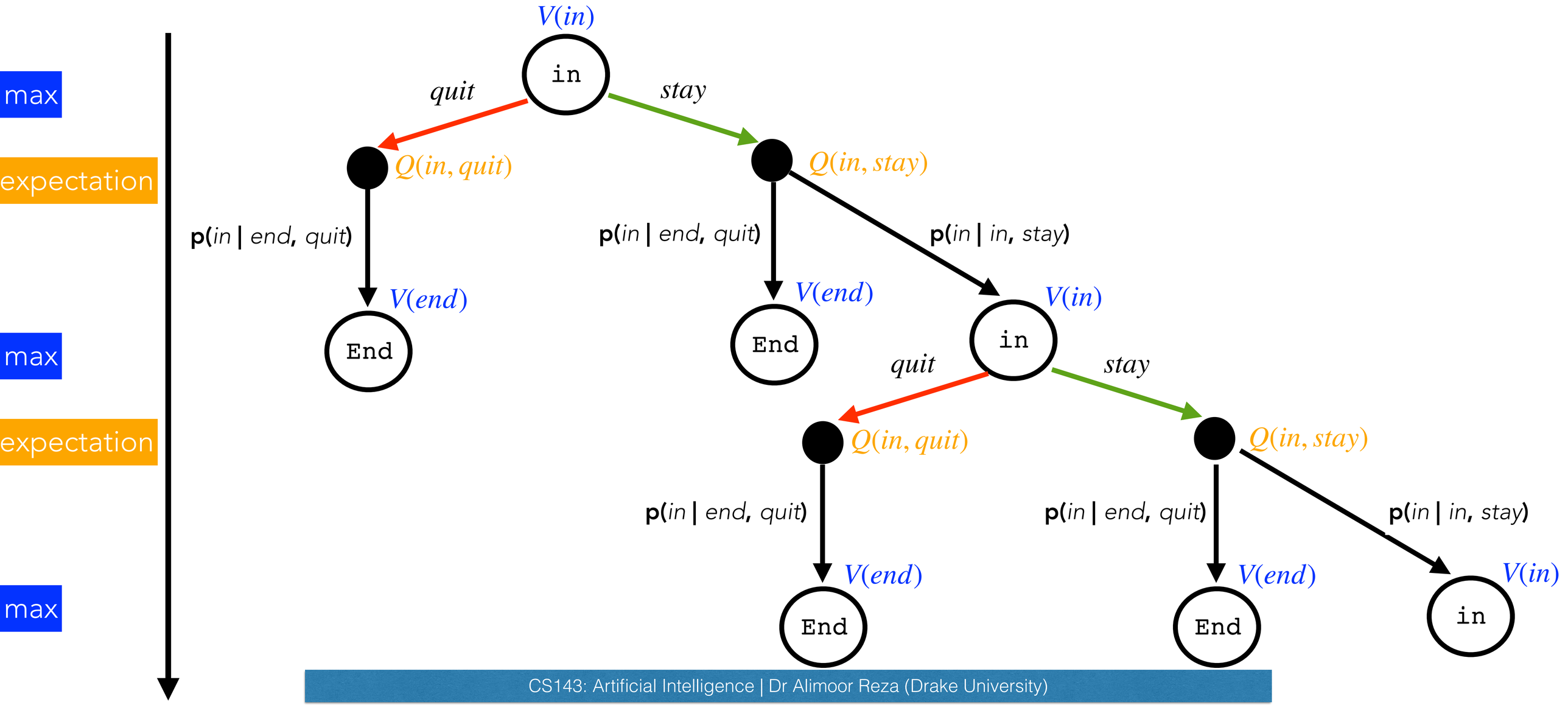
Question: Then how do you solve an MDP?

Solution to an MDP is called a **‘policy’**

Answer: Value Iteration Algorithm

An iterative algorithm to find the **optimal policy** (best solution) for an MDP

Build an Expectiminimax-like search tree



Mathematically: How to calculate V(s) values?

- How do we calculate $V_{opt}(in)$(optimal V-values for each state) from the recursive Bellman equation:

$$V_{opt}(in) = \max_{a \in actions(s)} \sum_{s'} p(s' \mid in, a)[R(in, a, s') + \gamma V_{opt}(s')]$$

A RECURRENCE RELATION

- Turn recursive Bellman equations into update equations:

$$V^0_{opt}(in) = 0$$

$$V^t_{opt}(in) = \max_{a \in actions(s)} \sum_{s'} p(s' \mid in, a)[R(in, a, s') + \gamma V^{t-1}_{opt}(s')]$$

Mathematically: How to calculate V(s) values?

- More precisely the update equations are as follows:

NON-TERMINAL STATE

$$V^t_{opt}(in) = \max \left\{ \begin{array}{c} Q^t_{opt}(in, quit) \\ Q^t_{opt}(in, stay) \end{array} \right\}$$

$$= \max \left\{ \begin{array}{c} \frac{1}{1}(10 + \gamma V^{t-1}_{opt}(end)) \\ \frac{2}{3}(4 + \gamma V^{t-1}_{opt}(in)) + \frac{1}{3}(4 + \gamma V^{t-1}_{opt}(end)) \end{array} \right\}$$

TERMINAL STATE

$$V_{opt}(end) = 0$$

THE RECURRENCE RELATION IS NOW EXPRESSED AS ITERATIVE UPDATES THAT CAN BE COMPUTED INCREMENTALLY

Algorithmically: Value iteration algorithm

INITIALIZE VALUES WITH SOME ROUGH ESTIMATES

REFINE IT OVER AND OVER AGAIN UNTIL THEIR VALUES DO NOT CHANGE

Algorithm 2: Value Iteration Algorithm

Initialize state-value $v^0_{opt}(s)$ for all the states s

//Repeat the iterative procedure for some iterations, let's say $T = 10000$

for $t = 1$ **to** T **do**

  for *each state* s **do**

    $v^t_{opt}(s) \leftarrow \max_{a \in actions(s)} \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma v^{t-1}_{opt}(s')]$

This is because the "value iteration algorithm" is now finding the best policy out of all possible candidate policies

Coding activity: Value iteration algorithm on our MDP

TERMINAL STATE

$$V_{opt}(end) = 0$$

NON-TERMINAL STATE

$$V^t_{opt}(in) = \max \left\{ \begin{array}{c} Q^t_{opt}(in, quit) \\ Q^t_{opt}(in, stay) \end{array} \right\}$$

$$= \max \left\{ \begin{array}{c} \frac{1}{1}(10 + \gamma V^{t-1}_{opt}(end)) \\ \frac{2}{3}(4 + \gamma V^{t-1}_{opt}(in)) + \frac{1}{3}(4 + \gamma V^{t-1}_{opt}(end)) \end{array} \right\}$$

Algorithm 2: Value Iteration Algorithm

Initialize state-value $v^0_{opt}(s)$ for all the states s

//Repeat the iterative procedure for some iterations, let's say $T = 10000$

for $t = 1$ **to** T **do**

  for *each state* s **do**

    $v^t_{opt}(s) \leftarrow \max_{a \in actions(s)} \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma v^{t-1}_{opt}(s')]$

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
$V_{opt}(end)$	0	0	0	0	0	0	0	0	0	0
$\pi_{opt}(end)$	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
$V_{opt}(in)$	0									
$\pi_{opt}(in)$										

Question: how to solve an MDP for grid environment like Frozen Lake?

Solution to an MDP is called a **‘policy’**

Answer: Value Iteration Algorithm

An iterative algorithm to find the **optimal policy** (best solution) for an MDP

MDP for Frozen Lake

- The states are description of the world
  - A simpler state representation could be agent's current position: linear index or (row_index, column_index)

- The actions are the finite number of possible actions (in discrete case)
  - In Frozen Lake, agent can take four actions:
    - move left 
    - move down 
    - move right 
    - move up 

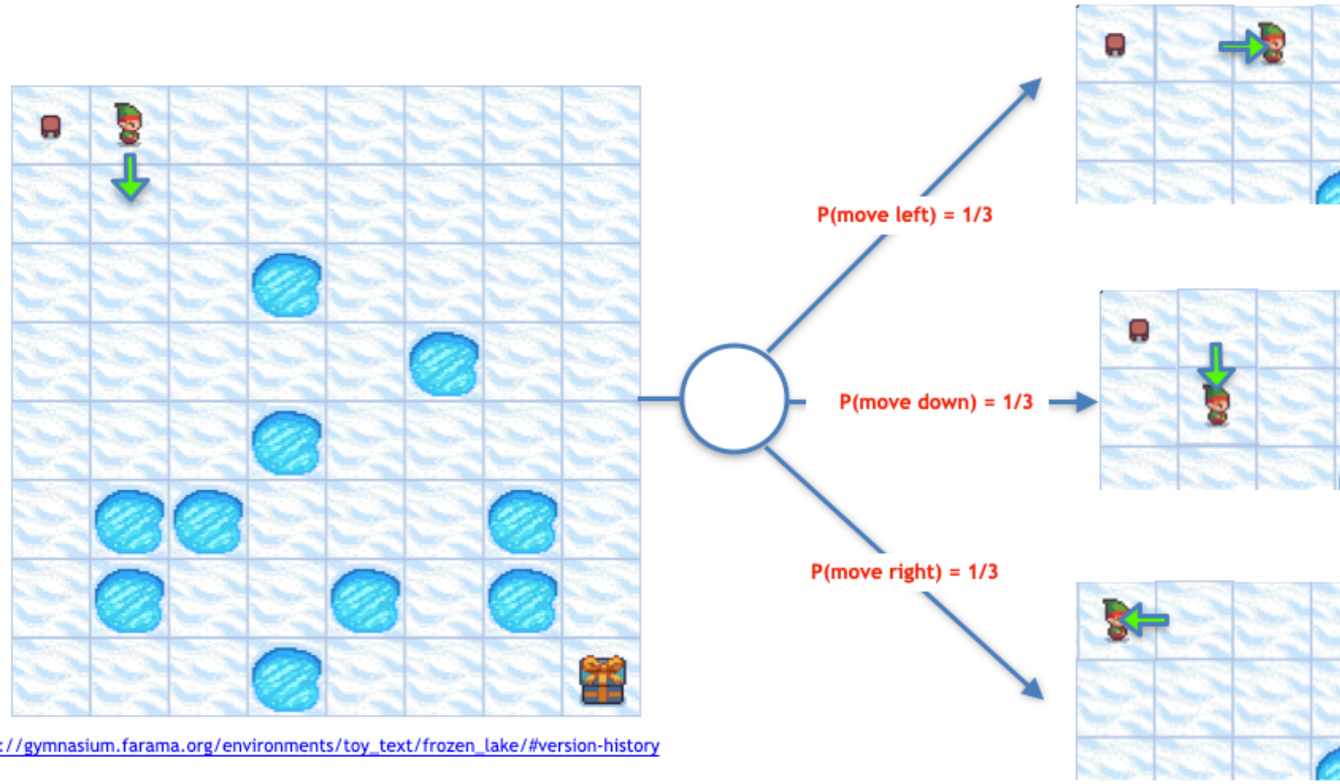
- The reward function encodes the reward agent receives
  - In Frozen Lake, reward function is known
  - $R(s, a, s')$ is the amount of reward the agent receives when it transitions from state s to s' by taking action a
    - reach goal (+1)
    - reach frozen tile (0)
    - reach hole (0)

MDP for Frozen Lake

- The transition function encodes the movements in the Lake
  - $P(s' \mid a, s)$ specify the probability of ending up in state s' if action a is taken (by agent) from state s
  - It should satisfy the probability rule:

$$\sum_{s' \in states} p(s' \mid s, a) = 1$$

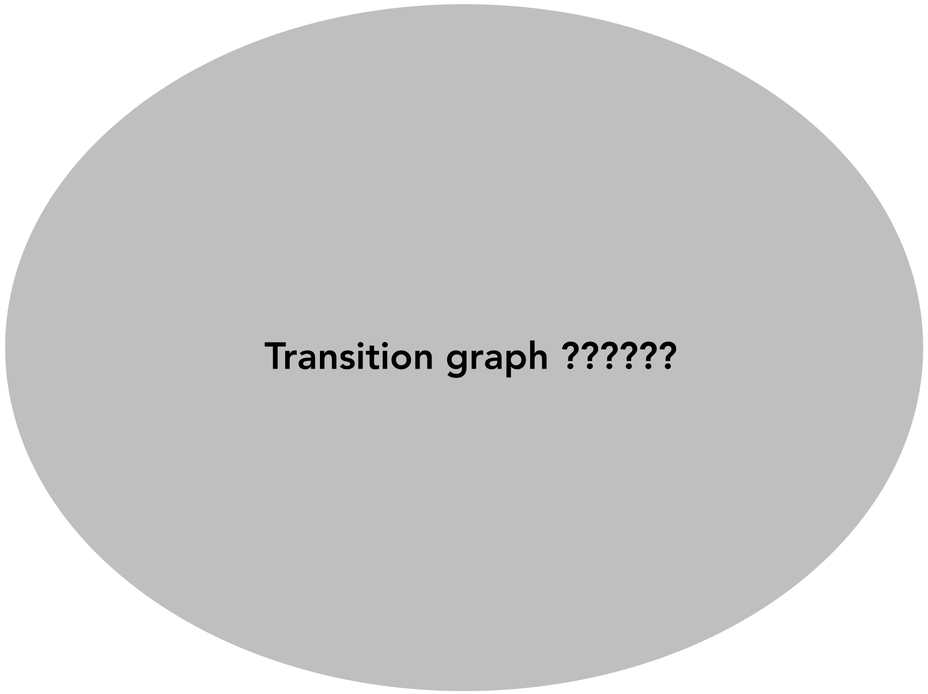
- In Frozen Lake, model or dynamics is known
  - Agent moves in intended direction with probability 1/3
  - It also moves in either perpendicular direction with equal probability (1/3 and 1/3)

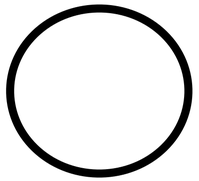


https://gymnasium.farama.org/environments/toy_text/frozen_lake/#version-history

MDP for Frozen Lake

- Let us formalize the dice game as a Markov decision process (MDP)





state node

- Edges coming out of a state node are the possible actions from that state node leading to a action node

action node

- Edges coming out of a action node are the possible random outcomes of that action, which end up back in state nodes
- Sum of probabilities coming out of a action node is 1.0

Optimal value function

- The optimal (or best) value $V_{opt}(s)$ is the maximum value attained among all possible policies for a given MDP

- The optimal value from state s can be formulated as

$$V_{opt}(s) = \max_{a \in actions(s)} Q_{opt}(s, a)$$

$$= \max_{a \in actions(s)} \sum_{s'} p(s' \mid s, a)[r(s, a, s') + \gamma V_{opt}(s')]$$

Mathematically: How to calculate V(s) values?

- How do we calculate $V_{opt}(s)$ (optimal V-values for each state) from the recursive Bellman equation:

$$V_{opt}(s) = \max_{a \in actions(s)} \sum_{s'} p(s' \mid s, a)[r(s, a, s') + \gamma V_{opt}(s')]$$

- Turn recursive Bellman equations into update equations:

$$V^0_{opt}(s) = 0$$

$$V^t_{opt}(s) = \max_{a \in actions(s)} \sum_{s'} p(s' \mid s, a)[r(s, a, s') + \gamma V^{t-1}_{opt}(s')]$$

Value in time step ***t*** can be computed via retrieving values from its previous iteration ***(t-1)***

Algorithmically: Value iteration algorithm

Initialize values with some rough estimates

Refine it over and over again until their values do not change

Algorithm 2: Value Iteration Algorithm

Initialize state-value $v^0_{opt}(s)$ for all the states s

//Repeat the iterative procedure for some iterations, let's say $T = 10000$

for $t = 1$ **to** T **do**

  for *each state* s **do**

    $v^t_{opt}(s) \leftarrow \max_{a \in actions(s)} \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma v^{t-1}_{opt}(s')]$

This is because the "value iteration algorithm" is now finding the best policy out of all possible candidate policies

Value iteration algorithm

- Determining the sizes of V and Q tables for your current instance of Frozen Lake

  How many entries will there be in the value-function table **V** ?

  - Answer: **16** **because there are 16 states**
  - You can initialize all of the entries with zeros as follows:

Index (1d):	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Index (2d):	(0,0)	(0,1)	(0,2)	(0,3)	(1,0)	(1,1)	(1,2)	(1,3)	(2,0)	(2,1)	(2,2)	(2,3)	(3,0)	(3,1)	(3,2)	(3,3)
V:	`[ 0.00,`	`0.00,`	`0.00,`	`0.00,`	`0.00,`	`0.00,`	`0.00,`	`0.00,`	`0.00,`	`0.00,`	`0.00,`	`0.00,`	`0.00,`	`0.00,`	`0.00,`	`0.00, ]`

Value iteration algorithm

- Determining the sizes of V and Q tables for your current instance of Frozen Lake

  How many entries will there be in the value-function table **Q** ?

  - Answer: **16x4** `because there are 16 states and for each state we need to compute 4 action values`
  - You can initialize all of the entries with zeros as follows:

Index (1d)	Index (2d)	Q LEFT	Q DOWN	Q RIGHT	Q UP
0	(0,0)	0.00	0.00	0.00	0.00
1	(0,1)	0.00	0.00	0.00	0.00
2	(0,2)	0.00	0.00	0.00	0.00
3	(0,3)	0.00	0.00	0.00	0.00
4	(1,0)	0.00	0.00	0.00	0.00
5	(1,1)	0.00	0.00	0.00	0.00
6	(1,2)	0.00	0.00	0.00	0.00
7	(1,3)	0.00	0.00	0.00	0.00
8	(2,0)	0.00	0.00	0.00	0.00
9	(2,1)	0.00	0.00	0.00	0.00
10	(2,2)	0.00	0.00	0.00	0.00
11	(2,3)	0.00	0.00	0.00	0.00
12	(3,0)	0.00	0.00	0.00	0.00
13	(3,1)	0.00	0.00	0.00	0.00
14	(3,2)	0.00	0.00	0.00	0.00
15	(3,3)	0.00	0.00	0.00	0.00

Value iteration algorithm (Iter#0)

Initialize values with all zeros estimates

Refine it over and over again until their values do not change

Index (1d):	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Index (2d):	(0,0)	(0,1)	(0,2)	(0,3)	(1,0)	(1,1)	(1,2)	(1,3)	(2,0)	(2,1)	(2,2)	(2,3)	(3,0)	(3,1)	(3,2)	(3,3)

V: `[ 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00 ]`

1D V table can be visualized in 2D table as follows

```
0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00
```

$$V^0_{opt}(s) = 0$$

Value iteration algorithm (Iter#0)

Initialize values with all zeros estimates

Refine it over and over again until their values do not change

V table

```
Updating value function V(s) for iteration=0
===============================================
 0.00 0.00 0.00 0.00
 0.00 0.00 0.00 0.00
 0.00 0.00 0.00 0.00
 0.00 0.00 0.00 0.00
```

$V^0_{opt}(s)$

Q table

LEFT	DOWN	RIGHT	UP	STATE INDEX (1d)
0.00,	0.00,	0.00,	0.00,	0
0.00,	0.00,	0.00,	0.00,	1
0.00,	0.00,	0.00,	0.00,	2
0.00,	0.00,	0.00,	0.00,	3
0.00,	0.00,	0.00,	0.00,	4
0.00,	0.00,	0.00,	0.00,	5
0.00,	0.00,	0.00,	0.00,	6
0.00,	0.00,	0.00,	0.00,	7
0.00,	0.00,	0.00,	0.00,	8
0.00,	0.00,	0.00,	0.00,	9
0.00,	0.00,	0.00,	0.00,	10
0.00,	0.00,	0.00,	0.00,	11
0.00,	0.00,	0.00,	0.00,	12
0.00,	0.00,	0.00,	0.00,	13
0.00,	0.00,	0.00,	0.00,	14
0.00,	0.00,	0.00,	0.00,	15

Turn recursive Bellman equations into update equations:

$$V^0_{opt}(s) = 0$$

Value iteration algorithm (Iter#1)

Initialize values with all zeros estimates

Refine it over and over again until their values do not change

V table

```
Updating value function V(s) for iteration=1
=====================================
 0.00 0.00 0.00 0.00
 0.00 0.00 0.00 0.00
 0.00 0.00 0.00 1.00
 0.00 0.00 1.00 0.00
```

$V^1_{opt}(11)$

$V^1_{opt}(14)$

$$V^1_{opt}(11) = \max \begin{cases} \frac{1}{1}(0.0 + \gamma V^0_{opt}(10)) & \text{LEFT} \\ \frac{1}{1}(1.0 + \gamma V^0_{opt}(15)) & \text{DOWN} \\ \frac{1}{1}(0.0 + \gamma V^0_{opt}(11)) & \text{RIGHT} \\ \frac{1}{1}(0.0 + \gamma V^0_{opt}(7)) & \text{UP} \end{cases}$$

$$V^1_{opt}(14) = \max \begin{cases} \frac{1}{1}(0 + \gamma V^0_{opt}(13)) & \text{LEFT} \\ \frac{1}{1}(0 + \gamma V^0_{opt}(14)) & \text{DOWN} \\ \frac{1}{1}(1.0 + \gamma V^0_{opt}(15)) & \text{RIGHT} \\ \frac{1}{1}(0 + \gamma V^0_{opt}(10)) & \text{UP} \end{cases}$$

Q table

STATE INDEX (1d)	LEFT	DOWN	RIGHT	UP
0	0.00	0.00	0.00	0.00
1	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00
8	0.00	0.00	0.00	0.00
9	0.00	0.00	0.00	0.00
10	0.00	0.00	0.00	0.00
11	0.00	1.00	0.00	0.00
12	0.00	0.00	0.00	0.00
13	0.00	0.00	0.00	0.00
14	0.00	0.00	1.00	0.00
15	0.00	0.00	0.00	0.00

During iteration#1: Let's check update equations for two states state=11 and state=14

Value iteration algorithm (Iter#1)

Initialize values with all zeros estimates

Refine it over and over again until their values do not change

V table

```
Updating value function V(s) for iteration=1
==============================================
 0.00 0.00 0.00 0.00
 0.00 0.00 0.00 0.00
 0.00 0.00 0.00 1.00
 0.00 0.00 1.00 0.00
```

$$V^{1}_{opt}(s)$$

During iteration #1, similar update equations are applied to the remaining 14 states.

Q table

STATE INDEX (1d)	LEFT	DOWN	RIGHT	UP
0	0.00	0.00	0.00	0.00
1	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00
8	0.00	0.00	0.00	0.00
9	0.00	0.00	0.00	0.00
10	0.00	0.00	0.00	0.00
11	0.00	1.00	0.00	0.00
12	0.00	0.00	0.00	0.00
13	0.00	0.00	0.00	0.00
14	0.00	0.00	1.00	0.00
15	0.00	0.00	0.00	0.00

Value iteration algorithm (Iter#2)

Initialize values with all zeros estimates

Refine it over and over again until their values do not change

V table

```
Updating value function V(s) for iteration=2
===============================================
 0.00 0.00 0.00 0.00
 0.00 0.00 0.00 0.90
 0.00 0.00 0.00 1.00
 0.00 0.90 1.00 0.00
```

$$V^2_{opt}(s)$$

Q table

STATE INDEX (1d)	LEFT	DOWN	RIGHT	UP
0	0.00	0.00	0.00	0.00
1	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00
7	0.00	0.90	0.00	0.00
8	0.00	0.00	0.00	0.00
9	0.00	0.00	0.00	0.00
10	0.00	0.00	0.00	0.00
11	0.00	1.00	0.90	0.00
12	0.00	0.00	0.00	0.00
13	0.00	0.00	0.90	0.00
14	0.00	0.90	1.00	0.00
15	0.00	0.00	0.00	0.00

Turn recursive Bellman equations into update equations:

$$V^2_{opt}(s) = \max_{a \in actions(s)} \sum_{s'} p(s' \mid s, a)[r(s, a, s') + \gamma V^1_{opt}(s')]$$

Value iteration algorithm (Iter#3)

Initialize values with all zeros estimates

Refine it over and over again until their values do not change

V table

```
Updating value function V(s) for iteration=3
===============================================
 0.00 0.00 0.00 0.81
 0.00 0.00 0.81 0.90
 0.00 0.00 0.00 1.00
 0.81 0.90 1.00 0.00
```

$$V^3_{opt}(s)$$

Q table

STATE INDEX (1d)	LEFT	DOWN	RIGHT	UP
0	0.00	0.00	0.00	0.00
1	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00
3	0.00	0.81	0.00	0.00
4	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00
6	0.00	0.00	0.81	0.00
7	0.00	0.90	0.81	0.00
8	0.00	0.00	0.00	0.00
9	0.00	0.00	0.00	0.00
10	0.00	0.00	0.00	0.00
11	0.00	1.00	0.90	0.81
12	0.00	0.00	0.81	0.00
13	0.00	0.81	0.90	0.00
14	0.81	0.90	1.00	0.00
15	0.00	0.00	0.00	0.00

Turn recursive Bellman equations into update equations:

$$V^3_{opt}(s) = \max_{a \in actions(s)} \sum_{s'} p(s' \mid s, a)[r(s, a, s') + \gamma V^2_{opt}(s')]$$

Value iteration algorithm (Iter#4)

Initialize values with all zeros estimates

Refine it over and over again until their values do not change

V table

```
Updating value function V(s) for iteration=4
===============================================
 0.00 0.00 0.73 0.81
 0.00 0.00 0.81 0.90
 0.73 0.00 0.00 1.00
 0.81 0.90 1.00 0.00
```

$$V^4_{opt}(s)$$

Q table

STATE INDEX (1d)	LEFT	DOWN	RIGHT	UP
0	0.00	0.00	0.00	0.00
1	0.00	0.00	0.00	0.00
2	0.00	0.73	0.73	0.00
3	0.00	0.81	0.73	0.73
4	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00
6	0.00	0.00	0.81	0.00
7	0.73	0.90	0.81	0.73
8	0.00	0.73	0.00	0.00
9	0.00	0.00	0.00	0.00
10	0.00	0.00	0.00	0.00
11	0.00	1.00	0.90	0.81
12	0.73	0.73	0.81	0.00
13	0.73	0.81	0.90	0.00
14	0.81	0.90	1.00	0.00
15	0.00	0.00	0.00	0.00

Turn recursive Bellman equations into update equations:

$$V^4_{opt}(s) = \max_{a \in actions(s)} \sum_{s'} p(s' \mid s, a)[r(s, a, s') + \gamma V^3_{opt}(s')]$$

Value iteration algorithm (Iter#5)

Initialize values with all zeros estimates

Refine it over and over again until their values do not change

V table

```
Updating value function V(s) for iteration=5
===============================================
 0.00  0.66  0.73  0.81
 0.66  0.00  0.81  0.90
 0.73  0.00  0.00  1.00
 0.81  0.90  1.00  0.00
```

$$V^5_{opt}(s)$$

Q table

STATE INDEX (1d)	LEFT	DOWN	RIGHT	UP
0	0.00	0.00	0.00	0.00
1	0.00	0.00	0.66	0.00
2	0.00	0.73	0.73	0.66
3	0.66	0.81	0.73	0.73
4	0.00	0.66	0.00	0.00
5	0.00	0.00	0.00	0.00
6	0.00	0.00	0.81	0.66
7	0.73	0.90	0.81	0.73
8	0.66	0.73	0.00	0.00
9	0.00	0.00	0.00	0.00
10	0.00	0.00	0.00	0.00
11	0.00	1.00	0.90	0.81
12	0.73	0.73	0.81	0.66
13	0.73	0.81	0.90	0.00
14	0.81	0.90	1.00	0.00
15	0.00	0.00	0.00	0.00

Turn recursive Bellman equations into update equations:

$$V^5_{opt}(s) = \max_{a \in actions(s)} \sum_{s'} p(s' \mid s, a)[r(s, a, s') + \gamma V^4_{opt}(s')]$$

Value iteration algorithm (Iter#6)

Initialize values with all zeros estimates

Refine it over and over again until their values do not change

V table

```
Updating value function V(s) for iteration=6
===============================================
0.59 0.66 0.73 0.81
0.66 0.00 0.81 0.90
0.73 0.00 0.00 1.00
0.81 0.90 1.00 0.00
```

$$V^5_{opt}(s)$$

Q table

LEFT	DOWN	RIGHT	UP	STATE INDEX (1d)
0.00	0.59	0.59	0.00	0
0.00	0.00	0.66	0.59	1
0.59	0.73	0.73	0.66	2
0.66	0.81	0.73	0.73	3
0.59	0.66	0.00	0.00	4
0.00	0.00	0.00	0.00	5
0.00	0.00	0.81	0.66	6
0.73	0.90	0.81	0.73	7
0.66	0.73	0.00	0.59	8
0.00	0.00	0.00	0.00	9
0.00	0.00	0.00	0.00	10
0.00	1.00	0.90	0.81	11
0.73	0.73	0.81	0.66	12
0.73	0.81	0.90	0.00	13
0.81	0.90	1.00	0.00	14
0.00	0.00	0.00	0.00	15

Turn recursive Bellman equations into update equations:

$$V^6_{opt}(s) = \max_{a \in actions(s)} \sum_{s'} p(s' \mid s, a)[r(s, a, s') + \gamma V^5_{opt}(s')]$$