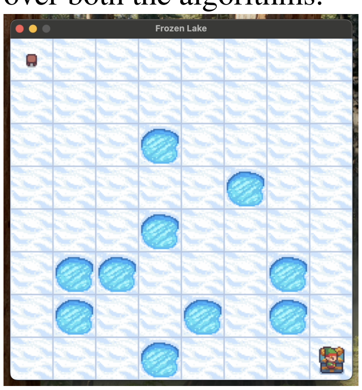
II a. Policy Iteration

```
sv.xxt@Sahdevs-MacBook-Pro ~/localDocuments/ds699/assignment2 python3 code_base.py -method policy_iteration -seeds 1
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en
/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using this package or pin to Setuptoo
ls<81.
  from pkg_resources import resource_stream, resource_exists
---- Policy Iteration----

There are 15 iterations in policy iteration.
policy: [['D' 'D' 'D' 'D' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'R' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'L' 'D' 'R' 'D' 'D']
 ['R' 'R' 'R' 'R' 'D' 'L' 'D' 'D']
 ['R' 'R' 'U' 'L' 'D' 'D' 'R' 'D']
 ['D' 'L' 'L' 'R' 'R' 'D' 'L' 'D']
 ['D' 'L' 'R' 'U' 'L' 'D' 'L' 'D']
 ['R' 'R' 'U' 'L' 'R' 'R' 'R' 'L']]
Total running time is: 0.8847732543945312  average number of iterations is: 15.0
Episode reward: 1.0
```

Value Iteration

```
sv.xxt@Sahdevs-MacBook-Pro ~/localDocuments/ds699/assignment2 python3 code_base.py -method value_iteration -seeds 1
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en
/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using this package or pin to Setuptoo
ls<81.
  from pkg_resources import resource_stream, resource_exists
---- Value Iteration----

There are 15 iterations in value iteration.
policy: [['D' 'D' 'D' 'D' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'R' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'L' 'D' 'R' 'D' 'D']
 ['R' 'R' 'R' 'R' 'D' 'L' 'D' 'D']
 ['R' 'R' 'U' 'L' 'D' 'D' 'R' 'D']
 ['D' 'L' 'L' 'R' 'R' 'D' 'L' 'D']
 ['D' 'L' 'R' 'U' 'L' 'D' 'L' 'D']
 ['R' 'R' 'U' 'L' 'R' 'R' 'R' 'L']]
Total running time is: 0.8831090927124023  average number of iterations is: 15.0
Episode reward: 1.0
```

As the episode reward is 1.0 in both the cases, our agent has learned the optimal policy and value functions over both the algorithms.



ii b. For **python3 code_base.py -method policy_iteration -seeds 50**

```
Total running time is: 13.196136951446533  average number of iterations is: 14.42
```

For **python3 code_base.py -method value_iteration -seeds 50**

```
Total running time is: 13.187311172485352  average number of iterations is: 15.0
```

II c. $O(|S| * |A| * K)$ is the worst case time complexity in a fully connected (stochastic) environment for the Policy Evaluation function. Since it is deterministic in the Frozen Lake stimulation for this assignment, the complexity is $O(|S| * K)$, where K is the number of iterations required for convergence.

II d. Given the assumptions in **II c**, the time complexity (t.c) of Policy Improvement is $O(|S| * |A|)$.

II e. T.C of one iteration of Policy Iteration is $O(|S| * K) + O(|S| * |A|) = O(|S| * (K + |A|))$, where K is the number of evaluations in a complete policy evaluation.

II f. T.C of one iteration of Value Iteration is $O(|S| * |A|) + O(|S| * |A|) = O(|S| * |A|)$.

III a. Output for **python3 code_base.py -method value_iteration -seeds 1 -epsilon 0.5**

```
sv.xxt@Sahdevs-MacBook-Pro  ~/localDocuments/ds699/assignment2  python3 code_base.py -method value_iteration -seeds 1 -epsilon 0.5
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setup
tools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using
 this package or pin to Setuptools<81.
  from pkg_resources import resource_stream, resource_exists
---- Value Iteration----

There are 8 iterations in value iteration.
policy: [['L' 'L' 'L' 'L' 'L' 'D' 'D' 'D']
 ['L' 'L' 'L' 'L' 'D' 'D' 'D' 'D']
 ['L' 'L' 'L' 'L' 'D' 'R' 'D' 'D']
 ['L' 'L' 'R' 'R' 'D' 'L' 'D' 'D']
 ['L' 'L' 'L' 'L' 'D' 'D' 'R' 'D']
 ['L' 'L' 'L' 'R' 'R' 'D' 'L' 'D']
 ['L' 'L' 'R' 'U' 'L' 'D' 'L' 'D']
 ['L' 'L' 'U' 'L' 'R' 'R' 'R' 'L']]
Total running time is: 0.9231522083282471  average number of iterations is: 8.0
The agent didn't reach a terminal state in 100 steps.
```

III b. There are 8 iterations. No, it isn't an optimal policy since the agent doesn't reach the terminal state.

III c. The agent requires 15 iterations to converge at the optimal policy in **II a**, which is more than that of **III a**.

III d. As we apply the bellman optimality equation, the value function is driven towards the optimal point as we back up using the best action. More iterations would further encourage this movement towards the fixed point. With initial iterations, large steps are taken as the initialized random condition vary significantly from the true optimal values and later, finer adjustments are made. These adjustments are governed by epsilon. As epsilon reduces, the precision of convergence increases, requiring more iterations to achieve the situation where value function across states observe minimal difference from the next iteration. If its large, we may miss the point because only a few iterations are required to meet the stopping criteria and therefore, the number of back ups through the bellman optimality equations will reduce, resulting in a non-optimal policy.

IV a **python3 code_base.py -method value_iteration -gamma 0**

```
sv.xxt@Sahdevs-MacBook-Pro  ~/localDocuments/ds699/assignment2  python3 code_base.py -method value_iteration -gamma 0
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setup
tools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using
 this package or pin to Setuptools<81.
  from pkg_resources import resource_stream, resource_exists
---- Value Iteration----

There are 2 iterations in value iteration.
policy: [['L' 'L' 'L' 'L' 'L' 'L' 'L' 'L']
 ['L' 'L' 'L' 'L' 'L' 'L' 'L' 'L']
 ['L' 'L' 'L' 'L' 'L' 'L' 'L' 'L']
 ['L' 'L' 'L' 'L' 'L' 'L' 'L' 'L']
 ['L' 'L' 'L' 'L' 'L' 'L' 'L' 'L']
 ['L' 'L' 'L' 'L' 'L' 'L' 'L' 'L']
 ['L' 'L' 'L' 'L' 'L' 'L' 'L' 'D']
 ['L' 'L' 'L' 'L' 'L' 'L' 'R' 'L']]
Total running time is: 1.0000038146972656  average number of iterations is: 2.0
The agent didn't reach a terminal state in 100 steps.
```

IV b As shown in **IV a**, $\gamma = 0$ doesn't generate an optimal policy. The agent values the immediate reward and this is simply 0 in the given environment. There's no incentive to drive the agent to learn the optimal policy.

$\gamma = 1$ doesn't generate an optimal policy although the algorithm converges after 15 iterations. The converged value function for all states is as follows :

```
Value Function in value_iter  [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1
. 0. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0. 1. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 0.]
```

It clearly shows no incentive to move to prefer one action over the other to move to its neighboring states, starting from 0. With no penalization for delay, the reward at 63 has not propagated meaningfully through the states. Image of the results is on the next page.

```
sv.xxt@Sahdevs-MacBook-Pro  ~/localDocuments/ds699/assignment2  python3 code_base.py -method value_iteration -gamma 1
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setup
tools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using
 this package or pin to Setuptools<81.
  from pkg_resources import resource_stream, resource_exists
---- Value Iteration----

There are 15 iterations in value iteration.
policy: [['L' 'L' 'L' 'L' 'L' 'L' 'L' 'L']
 ['L' 'L' 'L' 'L' 'L' 'L' 'L' 'L']
 ['L' 'L' 'L' 'L' 'D' 'L' 'L' 'L']
 ['L' 'L' 'L' 'L' 'L' 'L' 'D' 'L']
 ['L' 'L' 'L' 'L' 'D' 'L' 'L' 'L']
 ['L' 'L' 'L' 'D' 'L' 'L' 'L' 'D']
 ['L' 'L' 'D' 'L' 'L' 'D' 'L' 'D']
 ['L' 'L' 'L' 'L' 'D' 'L' 'L' 'L']]
Total running time is: 0.9138970375061035  average number of iterations is: 15.0
The agent didn't reach a terminal state in 100 steps.
```

$\gamma = 2$ leads to overflow in the bellman optimal equations, never converging and thus, not generating the optimal policy.

```
sv.xxt@Sahdevs-MacBook-Pro  ~/localDocuments/ds699/assignment2  python3 code_base.py -method value_iteration -gamma 2
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setup
tools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using
 this package or pin to Setuptools<81.
  from pkg_resources import resource_stream, resource_exists
---- Value Iteration----

/Users/sv.xxt/localDocuments/ds699/assignment2/code_base.py:201: RuntimeWarning: overflow encountered in scalar multiply
  q_vals[action] += prob * (reward + gamma * prevValFunc[nextState])
/Users/sv.xxt/localDocuments/ds699/assignment2/code_base.py:206: RuntimeWarning: invalid value encountered in subtract
  if np.linalg.norm(value_function-prevValFunc, np.inf) <= epsilon:
```

IV c With $\gamma = 0.5$, we don't get an optimal policy.

```
sv.xxt@Sahdevs-MacBook-Pro  ~/localDocuments/ds699/assignment2  python3 code_base.py -method value_iteration -gamma 0.5
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setup
tools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using
 this package or pin to Setuptools<81.
  from pkg_resources import resource_stream, resource_exists
---- Value Iteration----

There are 11 iterations in value iteration.
policy: [['L' 'L' 'D' 'D' 'D' 'D' 'D' 'D']
 ['L' 'D' 'D' 'R' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'L' 'D' 'R' 'D' 'D']
 ['R' 'R' 'R' 'R' 'D' 'L' 'D' 'D']
 ['R' 'R' 'U' 'L' 'D' 'D' 'R' 'D']
 ['L' 'L' 'L' 'R' 'R' 'D' 'L' 'D']
 ['D' 'L' 'R' 'U' 'L' 'D' 'L' 'D']
 ['R' 'R' 'U' 'L' 'R' 'R' 'R' 'L']]
Total running time is: 0.9140889644622803  average number of iterations is: 11.0
The agent didn't reach a terminal state in 100 steps.
```

Although, γ and ϵ are not directly related, the former determines the importance of the future value functions and the latter controls the number of iterations required for convergence. As we have higher γs, we require more iterations to converge under ϵ since multiple future states are taken into account and their values need to propagated through intermediatory states which takes time. For the given environment and agent, as you decrease γ, you must reduce ϵ by multiple factors of 10 to obtain an optimal policy (an empirical observation).

IV d Yes, for $\epsilon = 0.0001$.

```
✘ sv.xxt@Sahdevs-MacBook-Pro  ~/localDocuments/ds699/assignment2  python3 code_base.py -method value_iteration -gamma 0.5 -epsilon
0.0001
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setup
tools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using
 this package or pin to Setuptools<81.
  from pkg_resources import resource_stream, resource_exists
---- Value Iteration----

Value Function in value_iter  [1.221e-04 2.441e-04 4.883e-04 9.766e-04 1.953e-03 3.906e-03 7.812e-03 1.562e-02 2.441e-04 4.883e-04 9.7
66e-04 1.953e-03 3.906e-03 7.812e-03 1.562e-02 3.125e-02 4.883e-04 9.766e-04 1.953e-03 0.000e+00 7.812e-03 1.562e-02 3.125e-02 6.250e-
02 9.766e-04 1.953e-03 3.906e-03 7.812e-03 1.562e-02 0.000e+00 6.250e-02 1.250e-01 4.883e-04 9.766e-04 1.953e-03 0.000e+00 3.125e-02 6
.250e-02 1.250e-01 2.500e-01 2.441e-04 0.000e+00 0.000e+00 3.125e-02 6.250e-02 1.250e-01 0.000e+00 5.000e-01 4.883e-04 0.000e+00 7.812
e-03 1.562e-02 0.000e+00 2.500e-01 0.000e+00 1.000e+00 9.766e-04 1.953e-03 3.906e-03 0.000e+00 2.500e-01 5.000e-01 1.000e+00 0.000e+00
]
There are 15 iterations in value iteration.
policy: [['D' 'D' 'D' 'D' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'R' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'L' 'D' 'R' 'D' 'D']
 ['R' 'R' 'R' 'R' 'D' 'L' 'D' 'D']
 ['R' 'R' 'U' 'L' 'D' 'D' 'R' 'D']
 ['D' 'L' 'L' 'R' 'R' 'D' 'L' 'D']
 ['D' 'L' 'R' 'U' 'L' 'D' 'L' 'D']
 ['R' 'R' 'U' 'L' 'R' 'R' 'R' 'L']]
Total running time is: 0.9193150997161865  average number of iterations is: 15.0
Episode reward: 1.0
```

V a **python3 code_base.py -init_action=0** Left

```
✘ sv.xxt@Sahdevs-MacBook-Pro  ~/localDocuments/ds699/assignment2  python3 code_base.py -init_action=0
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setup
tools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using
 this package or pin to Setuptools<81.
  from pkg_resources import resource_stream, resource_exists
---- Policy Iteration----

There are 15 iterations in policy iteration.
policy: [['D' 'D' 'D' 'D' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'R' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'L' 'D' 'R' 'D' 'D']
 ['R' 'R' 'R' 'R' 'D' 'L' 'D' 'D']
 ['R' 'R' 'U' 'L' 'D' 'D' 'R' 'D']
 ['D' 'L' 'L' 'R' 'R' 'D' 'L' 'D']
 ['D' 'L' 'R' 'U' 'L' 'D' 'L' 'D']
 ['R' 'R' 'U' 'L' 'R' 'R' 'R' 'L']]
Total running time is: 1.0243360996246338  average number of iterations is: 15.0
Episode reward: 1.0
```

V b Yes, it is optimal and requires 15 iterations.

V c **python3 code_base.py -init_action=1** Down

```
sv.xxt@Sahdevs-MacBook-Pro  ~/localDocuments/ds699/assignment2  python3 code_base.py -init_action=1
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setup
tools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using
 this package or pin to Setuptools<81.
  from pkg_resources import resource_stream, resource_exists
---- Policy Iteration----

There are 13 iterations in policy iteration.
policy: [['D' 'D' 'D' 'D' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'R' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'L' 'D' 'R' 'D' 'D']
 ['R' 'R' 'R' 'R' 'D' 'L' 'D' 'D']
 ['R' 'R' 'U' 'L' 'D' 'D' 'R' 'D']
 ['D' 'L' 'L' 'R' 'R' 'D' 'L' 'D']
 ['D' 'L' 'R' 'U' 'L' 'D' 'L' 'D']
 ['R' 'R' 'U' 'L' 'R' 'R' 'R' 'L']]
Total running time is: 0.917262077331543  average number of iterations is: 13.0
Episode reward: 1.0
```

python3 code_base.py -init_action=2 Right

```
sv.xxt@Sahdevs-MacBook-Pro  ~/localDocuments/ds699/assignment2  python3 code_base.py -init_action=2
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setup
tools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using
 this package or pin to Setuptools<81.
  from pkg_resources import resource_stream, resource_exists
---- Policy Iteration----

There are 13 iterations in policy iteration.
policy: [['D' 'D' 'D' 'D' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'R' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'L' 'D' 'R' 'D' 'D']
 ['R' 'R' 'R' 'R' 'D' 'L' 'D' 'D']
 ['R' 'R' 'U' 'L' 'D' 'D' 'R' 'D']
 ['D' 'L' 'L' 'R' 'R' 'D' 'L' 'D']
 ['D' 'L' 'R' 'U' 'L' 'D' 'L' 'D']
 ['R' 'R' 'U' 'L' 'R' 'R' 'R' 'L']]
Total running time is: 0.9225232601165771  average number of iterations is: 13.0
Episode reward: 1.0
```

python3 code_base.py -init_action=3 Up

```
sv.xxt@Sahdevs-MacBook-Pro  ~/localDocuments/ds699/assignment2  python3 code_base.py -init_action=3
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setup
tools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using
 this package or pin to Setuptools<81.
  from pkg_resources import resource_stream, resource_exists
---- Policy Iteration----

There are 15 iterations in policy iteration.
policy: [['D' 'D' 'D' 'D' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'R' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'L' 'D' 'R' 'D' 'D']
 ['R' 'R' 'R' 'R' 'D' 'L' 'D' 'D']
 ['R' 'R' 'U' 'L' 'D' 'D' 'R' 'D']
 ['D' 'L' 'L' 'R' 'R' 'D' 'L' 'D']
 ['D' 'L' 'R' 'U' 'L' 'D' 'L' 'D']
 ['R' 'R' 'U' 'L' 'R' 'R' 'R' 'L']]
Total running time is: 0.9151170253753662  average number of iterations is: 15.0
Episode reward: 1.0
```

V d Initializing all states to Right or Down reduces the number of iterations. These are reasonable starting approximations as the goal state is at the bottom right corner. Overall, it will require fewer policy changes.

V e There are 119 evaluation steps for init_action=2

```
 sv.xxt@Sahdevs-MacBook-Pro  ~/localDocuments/ds699/assignment2  python3 code_base.py -init_action=2
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setup
tools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using
 this package or pin to Setuptools<81.
  from pkg_resources import resource_stream, resource_exists
---- Policy Iteration----

Evaluation steps are  4
Evaluation steps are  4
Evaluation steps are  5
Evaluation steps are  7
Evaluation steps are  7
Evaluation steps are  8
Evaluation steps are  9
Evaluation steps are  10
Evaluation steps are  11
Evaluation steps are  12
Evaluation steps are  13
Evaluation steps are  14
Evaluation steps are  15
There are 13 iterations with 119 evaluation steps in policy iteration.
policy: [['D' 'D' 'D' 'D' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'R' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'L' 'D' 'R' 'D' 'D']
 ['R' 'R' 'R' 'R' 'D' 'L' 'D' 'D']
 ['R' 'R' 'U' 'L' 'D' 'D' 'R' 'D']
 ['D' 'L' 'L' 'R' 'R' 'D' 'L' 'D']
 ['D' 'L' 'R' 'U' 'L' 'D' 'L' 'D']
 ['R' 'R' 'U' 'L' 'R' 'R' 'R' 'L']]
Total running time is: 0.907548189163208  average number of iterations is: 13.0
Episode reward: 1.0
```

V f We lose re-usable information of the previous iteration by re-initializing the value function to zero at each iteration. This introduces redundancy as we recalculate the value function over the same steps, increasing the number of evaluation steps needed as we progress. Moreover, information can be lost in this process as well.

V g To resolve the situation in **V f**, we can add a new parameters called **use_prevIter**, a boolean, to policy_iteration and **init_value**, a variable, to policy_evaluation. When **use_prevIter** is set to **True**, we pass the value_function used in policy_iteration, which is initialized with zeros and updated in its loop, to policy_evaluation's **init_value** parameter. In policy_evaluation, **init_value** is used to initialize value_function. This time, there are only 30 evaluation steps.

```
 sv.xxt@Sahdevs-MacBook-Pro  ~/localDocuments/ds699/assignment2  python3 code_base.py -init_action=2
/opt/homebrew/lib/python3.13/site-packages/pygame/pkgdata.py:25: UserWarning: pkg_resources is deprecated as an API. See https://setup
tools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using
 this package or pin to Setuptools<81.
  from pkg_resources import resource_stream, resource_exists
---- Policy Iteration----

Evaluation steps are  4
Evaluation steps are  2
Evaluation steps are  2
Evaluation steps are  3
Evaluation steps are  3
Evaluation steps are  2
Evaluation steps are  2
Evaluation steps are  2
Evaluation steps are  2
Evaluation steps are  2
Evaluation steps are  2
Evaluation steps are  2
Evaluation steps are  2
There are 13 iterations with 30 evaluation steps in policy iteration.
policy: [['D' 'D' 'D' 'D' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'R' 'D' 'D' 'D' 'D']
 ['D' 'D' 'D' 'L' 'D' 'R' 'D' 'D']
 ['R' 'R' 'R' 'R' 'D' 'L' 'D' 'D']
 ['R' 'R' 'U' 'L' 'D' 'D' 'R' 'D']
 ['D' 'L' 'L' 'R' 'R' 'D' 'L' 'D']
 ['D' 'L' 'R' 'U' 'L' 'D' 'L' 'D']
 ['R' 'R' 'U' 'L' 'R' 'R' 'R' 'L']]
Total running time is: 0.9746339321136475  average number of iterations is: 13.0
Episode reward: 1.0
```

I code_base.py

```python
### MDP Value Iteration and Policy Iteration
import random

from get_args import get_args
```

```python
import numpy as np
import gymnasium as gym
import time

np.set_printoptions(linewidth=np.inf)

np.set_printoptions(precision=3)


def interpret_policy(policy, nrow, ncol):
    '''

    interpret a 2-D policy from number to action using: 0: L 1: D 2: R
       3: U
    :param policy: generated policy by a method
    :param nrow: number of rows
    :param ncol: number of columns
    :return: re_policy: policy of each state with action's first letter
    '''
    policy = policy.reshape(nrow, ncol)
    re_policy = np.zeros((nrow, ncol), dtype=str)
    for i in range(len(policy)):
        for j in range(len(policy[i])):
            if policy[i][j] == 0:
                re_policy[i][j] = 'L'
            elif policy[i][j] == 1:
                re_policy[i][j] = 'D'
            elif policy[i][j] == 2:
                re_policy[i][j] = 'R'
            elif policy[i][j] == 3:
                re_policy[i][j] = 'U'
    return re_policy


def policy_evaluation(P, nS, policy, init_value, gamma=0.9,
   epsilon=1e-3):
    """

    Evaluate the value function from a given policy.
    :param P: transition probability
    :param nS: number of states
    :param policy: the policy to be evaluated
    :param gamma: gamma parameter used in policy evaluation
    :param epsilon: epsilon parameter used in policy evaluation
    :return: value_function: value function from policy evaluation
             evalution_steps: the number of steps need for policy
                evaluation
```

```
"""

############################
# Your Code #
# Modify the following line for initialization optimization in
   question 5.(a)
# Hint: Please add a new parameter for the policy_iteration function
   and use this parameter to control the initialization.

# Initialize value function as all zeros - done under policy
   iteration
value_function = init_value.copy()
############################

# evaluation_steps: the number of steps needed for policy evaluation
   in each iteration
evaluation_steps = 0

############################
# Your Code #
# Please use np.linalg.norm(x, np.inf) to calculate the infinity
   norm. #
# Please use while loop to finish this part. #
# Remember to update the evaluation_steps. #

while True:
# Synchronous Backup Implementation Steps:
# 1. Save a copy of old values: Copy the current value function to
   'value_function_prev' before each iteration
# 2. Iterate over all states: Compute new values uniformly based on
   'value_function_prev' to avoid immediate updates affecting other
   states in the current iteration
# 3. Convergence criterion: Calculate the infinity norm (max
   absolute difference) between old and new value functions.
   Terminate if below 'epsilon'
    value_function_prev = value_function.copy()

    for state in range(nS):
        action = policy[state]
        nextVal = 0.0
        for prob, nextState, reward, _ in P[state][action]:
            nextVal += prob * (reward + gamma  *
               value_function_prev[nextState]) # following the
               bellman equation in textbook
        value_function[state] = nextVal
    evaluation_steps += 1
```

```
        if np.linalg.norm(value_function - value_function_prev, np.inf)
           <= epsilon:
            break

    ############################

    return value_function, evaluation_steps


def policy_improvement(P, nS, nA, value_function, gamma=0.9):
    """
    Use the value function to improve the policy.
    :param P: transition probability
    :param nS: number of states
    :param nA: number of actions
    :param value_function: value function from policy iteration
    :param gamma: gamma parameter used in policy improvement
    :return: new_policy: An array of integers. Each integer is the
       optimal action to take in that state according to
                the environment dynamics and the given value function.
    """

    new_policy = np.zeros(nS, dtype="int")

    ############################
    # Your Code #
    # Please use np.argmax to select the best actions after getting the
       q value of each action. #
    for state in range(nS):
        q_values = np.zeros(nA)
        for action in range(nA):
            for prob, nextState, reward, _ in P[state][action]:
                q_values[action] += prob * (reward + gamma *
                   value_function[nextState])
        new_policy[state] = np.argmax(q_values)
    ############################
    return new_policy


def policy_iteration(P, nS, nA, use_prevIter = True, init_action=-1,
   gamma=0.9, epsilon=1e-3):
    """
    Runs policy iteration. Please call the policy_evaluation() and
       policy_improvement() methods to implement this method.
    :param P: transition probability
    :param nS: number of states
    :param nA: number of actions
```

```
:param init_action: initial action for all the states, -1 for random
   action
:param gamma: gamma parameter used in policy_evaluation() and
   policy_improvement()
:param epsilon: epsilon parameter used in policy_evaluation()
:return: value_function: np.ndarray[nS]
             policy: np.ndarray[nS]
             iteration: int, the number of iterations needed for
                policy iteration
"""

value_function = np.zeros(nS)

############################
# Your Code #
# for the question of policy iteration initialization optimization #
# Initialize policy #
init_policy = np.random.randint(0, nA, nS) if init_action == -1 else
   np.ones(nS, dtype=int) * init_action
############################

# Number of iterations. The iteration does not include the steps of
   policy evaluation.
iteration = 0

# previous policy: the policy of last iteration.
policy_prev = init_policy

############################
# Your Code #
# Please call the policy_evaluation() and policy_improvement() to
   update the policy. #
# Remember to update the iteration and policy_prev. #
# Please use while loop to finish this part. #
# The time complexity of the code within the while loop represents
   the running time required "in one iteration" as mentioned in
   II.(c)#
net_evalSteps = 0
while True:
    init_value = value_function if use_prevIter else np.zeros(nS)
    value_function, evaluation_steps = policy_evaluation(P, nS,
       policy_prev, init_value, gamma=gamma, epsilon=epsilon)
    print('Evaluation steps are ',evaluation_steps)
    net_evalSteps += evaluation_steps
    policy = policy_improvement(P, nS, nA, value_function, gamma)
    iteration += 1
    if (np.linalg.norm(policy-policy_prev,1) == 0):
```

```
            break
        policy_prev = policy.copy()
    ############################

    print(f"There are {iteration} iterations with {net_evalSteps}
       evaluation steps in policy iteration.")
    return value_function, policy, iteration


def value_iteration(P, nS, nA, init_value=0.0, gamma=0.9, epsilon=1e-3):
    """
    Learn value function and policy by using value iteration method for
       a given gamma and environment.
    :param P: transition probability
    :param nS: number of states
    :param nA: number of actions
    :param init_value: initial value for value iteration
    :param gamma: gamma parameter used in value_iteration()
    :param epsilon: epsilon parameter used in value_iteration()
    :return: value_function: np.ndarray[nS]
                 policy: np.ndarray[nS]
                 iteration: int, the number of iterations needed for
                    value iteration
    """



    # Initialize value #
    value_function = np.ones(nS) * init_value

    # policy: the policy output from the generated value function after
       value iteration.
    policy = np.zeros(nS, dtype=int)
    iteration = 0

    ############################
    # Your Code #
    # Please use np.argmax to select the best action after getting the q
       value of each action. #
    # Please use np.linalg.norm(x, np.inf) to calculate the infinity
       norm. #
    # Please use while loop to finish this part. #
    # The time complexity of the code within the while loop represents
       the running time required "in one iteration" as mentioned in
       II.(d)#
    while True:
        prevValFunc = value_function.copy()
        for state in range(nS):
```

```
            q_vals = np.zeros(nA)
            for action in range(nA):
                for prob, nextState, reward, _ in P[state][action]:
                    q_vals[action] += prob * (reward + gamma *
                        prevValFunc[nextState])
            #print("Q_vals ",q_vals)
            value_function[state] = np.max(q_vals)
            #print('Value Function in value_iter ', value_function)
        iteration += 1
        if np.linalg.norm(value_function-prevValFunc, np.inf) <= epsilon:
            break
    print('Value Function in value_iter ', value_function)
    for state in range(nS):
        q_values = np.zeros(nA)
        for action in range(nA):
            for prob, nextState, reward, _ in P[state][action]:
                q_values[action] += prob * (reward + gamma *
                   value_function[nextState]) # using the value function
                   obtained from covergence of Bellman Optimal Equation
        policy[state] = np.argmax(q_values)


    ############################

    # uncomment the following line if you need to print the value
       function
    #print('value_function:', value_function)

    print(f"There are {iteration} iterations in value iteration.")
    return value_function, policy, iteration


def render_single(env, policy, max_steps=100):
    '''

    This function does not need to be modified
    Renders policy once on environment. Watch your agent play!

    :param env: gym.core.Environment. Environment to play on. Must have
       nS, nA, and P as attributes.
    :param policy: np.array of shape [env.nS]. The action to take at a
       given state
    :param max_steps: the maximum number of iterations
    :return: None
    '''

    episode_reward = 0
```

```
    state, _ = env.reset()
    for t in range(max_steps):
        env.render()
        time.sleep(0.25)
        action = policy[state]
        state, reward, done, _, _ = env.step(action)
        episode_reward += reward
        if done:
            break
    env.render()
    if not done:
        print(f"The agent didn't reach a terminal state in {max_steps}
           steps.")
    else:
        print(f"Episode reward: {episode_reward}")


# Edit below to run policy and value iteration on different environments
   and
# visualize the resulting policies of actions.

if __name__ == "__main__":
    # get arguments from get_args.py
    args = get_args()

    # Initialize the gym environment and render
    env = gym.make('FrozenLake-v1', desc=None, map_name="8x8",
       render_mode=args.render_mode, is_slippery=False)
    # Please check this link for the definition of state and actions of
       the FrozenLake game:
    # https://www.gymlibrary.dev/environments/toy\_text/frozen\_lake/

    env = env.unwrapped

    # Number of state is 8 * 8 = 64
    env.nS = env.nrow * env.ncol
    # Number of action is 4
    env.nA = 4

    # Uncomment the following line to check and understand the format of
       the transition probability of FrozenLake.
    #print('transition probability:', env.P[0][0])

    # Running time start point
    start = time.time()
    # Initialize the average iteration
    avg_iteration = 0
```

```
# Run the algorithm for "args.seeds" times. Each time with a
  different random seed.
for i in range(args.seeds):

    # Reset the environment
    env.reset()
    # Set the random seed
    np.random.seed(i)
    random.seed(i)

    if args.method == 'policy_iteration':
        # Run policy iteration
        print("---- Policy Iteration----\n")
        value, policy, iteration = policy_iteration(
            env.P, env.nS, env.nA, init_action=args.init_action,
               gamma=args.gamma, epsilon=args.epsilon)

    elif args.method == 'value_iteration':
        # Run value iteration
        print("---- Value Iteration----\n")
        value, policy, iteration = value_iteration(
            env.P, env.nS, env.nA, init_value=args.init_value,
               gamma=args.gamma, epsilon=args.epsilon)
    else:
        raise ValueError('Unknown method')
    # Cumulate the number of iterations
    avg_iteration += iteration

    # Print the policy, interpreted to the actions' first letter
       (check the interpret_policy function).
    print('policy:', interpret_policy(policy.reshape(env.nrow,
       env.ncol), env.nrow, env.ncol))

print('Total running time is:', time.time() - start,
      ' average number of iterations is:', avg_iteration /
         args.seeds)

# Render the policy, the rendering do not require screenshots.
render_single(env, policy, 100)
```