In [3]:
```
import warnings ; warnings.filterwarnings('ignore')

import gymnasium as gym # Changed gym to gymnasium
import gym_walk
import numpy as np

import random
import warnings

warnings.filterwarnings('ignore', category=DeprecationWarning)
np.set_printoptions(suppress=True)
random.seed(123); np.random.seed(123)
```

```
Gym has been unmaintained since 2022 and does not support NumPy 2.0 amongst oth
er critical functionality.
Please upgrade to Gymnasium, the maintained drop-in replacement of Gym, or cont
act the authors of your software and request that they upgrade.
See the migration guide at https://gymnasium.farama.org/introduction/migratio
n_guide/ for additional information.
```

In [2]:
```
pip install git+https://github.com/mimoralea/gym-walk#egg=gym-walk
```

```
Collecting gym-walk
  Cloning https://github.com/mimoralea/gym-walk to /tmp/pip-install-7jdahbnw/gy
m-walk_284bcdd3e74946d6b7975f4d61c996c4
  Running command git clone --filter=blob:none --quiet https://github.com/mimor
alea/gym-walk /tmp/pip-install-7jdahbnw/gym-walk_284bcdd3e74946d6b7975f4d61c996
c4
  Resolved https://github.com/mimoralea/gym-walk to commit b915b94cf2ad16f8833a
1ad92ea94e88159279f5
  Preparing metadata (setup.py) ... done
Requirement already satisfied: gym in /usr/local/lib/python3.12/dist-packages
(from gym-walk) (0.25.2)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.12/dist-
packages (from gym->gym-walk) (2.0.2)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.12/
dist-packages (from gym->gym-walk) (3.1.1)
Requirement already satisfied: gym-notices>=0.0.4 in /usr/local/lib/python3.12/
dist-packages (from gym->gym-walk) (0.1.0)
Building wheels for collected packages: gym-walk
  Building wheel for gym-walk (setup.py) ... done
  Created wheel for gym-walk: filename=gym_walk-0.0.2-py3-none-any.whl size=537
7 sha256=96f6628bf12910217dafabad9e32e0d0362919e61e960bb798ed6f9d6f11a194
  Stored in directory: /tmp/pip-ephem-wheel-cache-pkhpwz2x/wheels/bf/23/e5/a94b
e4a90dd18f7ce958c21f192276cb01ef0daaf2bc66583b
Successfully built gym-walk
Installing collected packages: gym-walk
Successfully installed gym-walk-0.0.2
```

In [4]:
```
def print_policy(pi, P, action_symbols=('<', 'v', '>', '^'), n_cols=4, title='
    print(title)
    arrs = {k:v for k,v in enumerate(action_symbols)}
    for s in range(len(P)):
        a = pi[s]
```

```
        print("| ", end="")
        if np.all([done for action in P[s].values() for _, _, _, done in actio
            print("".rjust(9), end=" ")
        else:
            print(str(s).zfill(2), arrs[a].rjust(6), end=" ")
        if (s + 1) % n_cols == 0: print("|")
```

In [5]:
```
def print_state_value_function(V, P, n_cols=4, prec=3, title='State-value func
    print(title)
    for s in range(len(P)):
        v = V[s]
        print("| ", end="")
        if np.all([done for action in P[s].values() for _, _, _, done in actio
            print("".rjust(9), end=" ")
        else:
            print(str(s).zfill(2), '{}'.format(np.round(v, prec)).rjust(6), en
        if (s + 1) % n_cols == 0: print("|")
```

In [7]:
```
env = gym.make('FrozenLake-v1')
P = env.unwrapped.P
init_state = env.reset()
#goal_state = 6
#LEFT, RIGHT = range(2)
```

In [8]:
```
P
```

Out[8]:
```
{0: {0: [(0.33333333333333337, 0, 0, False),
         (0.3333333333333333, 0, 0, False),
         (0.33333333333333337, 4, 0, False)],
        1: [(0.33333333333333337, 0, 0, False),
         (0.3333333333333333, 4, 0, False),
         (0.33333333333333337, 1, 0, False)],
        2: [(0.33333333333333337, 4, 0, False),
         (0.3333333333333333, 1, 0, False),
         (0.33333333333333337, 0, 0, False)],
        3: [(0.33333333333333337, 1, 0, False),
         (0.3333333333333333, 0, 0, False),
         (0.33333333333333337, 0, 0, False)]},
    1: {0: [(0.33333333333333337, 1, 0, False),
         (0.3333333333333333, 0, 0, False),
         (0.33333333333333337, 5, 0, True)],
        1: [(0.33333333333333337, 0, 0, False),
         (0.3333333333333333, 5, 0, True),
         (0.33333333333333337, 2, 0, False)],
        2: [(0.33333333333333337, 5, 0, True),
         (0.3333333333333333, 2, 0, False),
         (0.33333333333333337, 1, 0, False)],
        3: [(0.33333333333333337, 2, 0, False),
         (0.3333333333333333, 1, 0, False),
         (0.33333333333333337, 0, 0, False)]},
    2: {0: [(0.33333333333333337, 2, 0, False),
         (0.3333333333333333, 1, 0, False),
         (0.33333333333333337, 6, 0, False)],
        1: [(0.33333333333333337, 1, 0, False),
         (0.3333333333333333, 6, 0, False),
         (0.33333333333333337, 3, 0, False)],
        2: [(0.33333333333333337, 6, 0, False),
         (0.3333333333333333, 3, 0, False),
         (0.33333333333333337, 2, 0, False)],
        3: [(0.33333333333333337, 3, 0, False),
         (0.3333333333333333, 2, 0, False),
         (0.33333333333333337, 1, 0, False)]},
    3: {0: [(0.33333333333333337, 3, 0, False),
         (0.3333333333333333, 2, 0, False),
         (0.33333333333333337, 7, 0, True)],
        1: [(0.33333333333333337, 2, 0, False),
         (0.3333333333333333, 7, 0, True),
         (0.33333333333333337, 3, 0, False)],
        2: [(0.33333333333333337, 7, 0, True),
         (0.3333333333333333, 3, 0, False),
         (0.33333333333333337, 3, 0, False)],
        3: [(0.33333333333333337, 3, 0, False),
         (0.3333333333333333, 3, 0, False),
         (0.33333333333333337, 2, 0, False)]},
    4: {0: [(0.33333333333333337, 0, 0, False),
         (0.3333333333333333, 4, 0, False),
         (0.33333333333333337, 8, 0, False)],
        1: [(0.33333333333333337, 4, 0, False),
         (0.3333333333333333, 8, 0, False),
         (0.33333333333333337, 5, 0, True)],
```

```
 2: [(0.33333333333333337, 8, 0, False),
  (0.3333333333333333, 5, 0, True),
  (0.33333333333333337, 0, 0, False)],
 3: [(0.33333333333333337, 5, 0, True),
  (0.3333333333333333, 0, 0, False),
  (0.33333333333333337, 4, 0, False)]},
5: {0: [(1.0, 5, 0, True)],
 1: [(1.0, 5, 0, True)],
 2: [(1.0, 5, 0, True)],
 3: [(1.0, 5, 0, True)]},
6: {0: [(0.33333333333333337, 2, 0, False),
  (0.3333333333333333, 5, 0, True),
  (0.33333333333333337, 10, 0, False)],
 1: [(0.33333333333333337, 5, 0, True),
  (0.3333333333333333, 10, 0, False),
  (0.33333333333333337, 7, 0, True)],
 2: [(0.33333333333333337, 10, 0, False),
  (0.3333333333333333, 7, 0, True),
  (0.33333333333333337, 2, 0, False)],
 3: [(0.33333333333333337, 7, 0, True),
  (0.3333333333333333, 2, 0, False),
  (0.33333333333333337, 5, 0, True)]},
7: {0: [(1.0, 7, 0, True)],
 1: [(1.0, 7, 0, True)],
 2: [(1.0, 7, 0, True)],
 3: [(1.0, 7, 0, True)]},
8: {0: [(0.33333333333333337, 4, 0, False),
  (0.3333333333333333, 8, 0, False),
  (0.33333333333333337, 12, 0, True)],
 1: [(0.33333333333333337, 8, 0, False),
  (0.3333333333333333, 12, 0, True),
  (0.33333333333333337, 9, 0, False)],
 2: [(0.33333333333333337, 12, 0, True),
  (0.3333333333333333, 9, 0, False),
  (0.33333333333333337, 4, 0, False)],
 3: [(0.33333333333333337, 9, 0, False),
  (0.3333333333333333, 4, 0, False),
  (0.33333333333333337, 8, 0, False)]},
9: {0: [(0.33333333333333337, 5, 0, True),
  (0.3333333333333333, 8, 0, False),
  (0.33333333333333337, 13, 0, False)],
 1: [(0.33333333333333337, 8, 0, False),
  (0.3333333333333333, 13, 0, False),
  (0.33333333333333337, 10, 0, False)],
 2: [(0.33333333333333337, 13, 0, False),
  (0.3333333333333333, 10, 0, False),
  (0.33333333333333337, 5, 0, True)],
 3: [(0.33333333333333337, 10, 0, False),
  (0.3333333333333333, 5, 0, True),
  (0.33333333333333337, 8, 0, False)]},
10: {0: [(0.33333333333333337, 6, 0, False),
  (0.3333333333333333, 9, 0, False),
  (0.33333333333333337, 14, 0, False)],
 1: [(0.33333333333333337, 9, 0, False),
```

```
  (0.3333333333333333, 14, 0, False),
  (0.33333333333333337, 11, 0, True)],
 2: [(0.33333333333333337, 14, 0, False),
  (0.3333333333333333, 11, 0, True),
  (0.33333333333333337, 6, 0, False)],
 3: [(0.33333333333333337, 11, 0, True),
  (0.3333333333333333, 6, 0, False),
  (0.33333333333333337, 9, 0, False)]},
11: {0: [(1.0, 11, 0, True)],
 1: [(1.0, 11, 0, True)],
 2: [(1.0, 11, 0, True)],
 3: [(1.0, 11, 0, True)]},
12: {0: [(1.0, 12, 0, True)],
 1: [(1.0, 12, 0, True)],
 2: [(1.0, 12, 0, True)],
 3: [(1.0, 12, 0, True)]},
13: {0: [(0.33333333333333337, 9, 0, False),
  (0.3333333333333333, 12, 0, True),
  (0.33333333333333337, 13, 0, False)],
 1: [(0.33333333333333337, 12, 0, True),
  (0.3333333333333333, 13, 0, False),
  (0.33333333333333337, 14, 0, False)],
 2: [(0.33333333333333337, 13, 0, False),
  (0.3333333333333333, 14, 0, False),
  (0.33333333333333337, 9, 0, False)],
 3: [(0.33333333333333337, 14, 0, False),
  (0.3333333333333333, 9, 0, False),
  (0.33333333333333337, 12, 0, True)]},
14: {0: [(0.33333333333333337, 10, 0, False),
  (0.3333333333333333, 13, 0, False),
  (0.33333333333333337, 14, 0, False)],
 1: [(0.33333333333333337, 13, 0, False),
  (0.3333333333333333, 14, 0, False),
  (0.33333333333333337, 15, 1, True)],
 2: [(0.33333333333333337, 14, 0, False),
  (0.3333333333333333, 15, 1, True),
  (0.33333333333333337, 10, 0, False)],
 3: [(0.33333333333333337, 15, 1, True),
  (0.3333333333333333, 10, 0, False),
  (0.33333333333333337, 13, 0, False)]},
15: {0: [(1.0, 15, 0, True)],
 1: [(1.0, 15, 0, True)],
 2: [(1.0, 15, 0, True)],
 3: [(1.0, 15, 0, True)]}}
```

In [9]:

```python
def decay_schedule(
    init_value, min_value, decay_ratio,
    max_steps, log_start = -2, log_base=10):

  #Write your code here
  decay_steps = int(max_steps * decay_ratio)
  rem_steps = max_steps - decay_steps
  values = np.logspace(
```

```
      log_start, 0, decay_steps, base=log_base, endpoint=True
  )[::-1]
  values = (values - values.min()) / (values.max() - values.min())
  values = (init_value - min_value) * values + min_value
  values = np.pad(values, (0, rem_steps), 'edge')

  return values
```

In [15]:

```
from itertools import count
def generate_trajectory(
    select_action, Q, epsilon,
    env, max_steps=200):
  done, trajectory = False, []

  #Write your code here
  state_tuple = env.reset()
  state = state_tuple[0] # Extract the observation

  for t in count():
    action = select_action(state, Q, epsilon)
    next_state_tuple, reward, done, _, info = env.step(action) # Get the full
    next_state = next_state_tuple # In FrozenLake-v1, the observation is the s
    trajectory.append((state, action, reward, next_state, done))
    state = next_state
    if done or t >= max_steps - 1:
      break

  return np.array(trajectory, object)
```

In [19]:

```
from tqdm import tqdm

def mc_control(env,
               gamma=1.0,
               init_alpha=0.5,
               min_alpha=0.01,
               alpha_decay_ratio=0.5,
               init_epsilon=1.0,
               min_epsilon=0.1,
               epsilon_decay_ratio=0.9,
               n_episodes=3000,
               max_steps=200,
               first_visit=True):

    nS, nA = env.observation_space.n, env.action_space.n
    goal_state = nS - 1 # Assuming the last state is the goal state

    discounts = np.logspace(
        0, max_steps,
        num=max_steps + 1, base=gamma,
        endpoint=True)

    alphas = decay_schedule(
        init_alpha, min_alpha,
```

```
        alpha_decay_ratio,
        n_episodes)

    epsilons = decay_schedule(
        init_epsilon, min_epsilon,
        epsilon_decay_ratio,
        n_episodes)

    pi_track = []
    Q = np.zeros((nS, nA), dtype=np.float64)
    Q_track = np.zeros((n_episodes, nS, nA), dtype=np.float64)
    reached_goal_list = [] # Initialize a list to store reached_goal for each

    select_action = lambda state, Q, epsilon:np.argmax(Q[state]) if np.random.

    for e in tqdm(range(n_episodes), leave=False):
        trajectory = generate_trajectory(select_action, Q, epsilons[e], env, m
        visited = np.zeros((nS, nA), dtype=bool)
        episode_reached_goal = False # Flag for the current episode

        for t, (state, action, reward, next_state, done) in enumerate(trajectc
            if visited[state][action] and first_visit:
                continue
            visited[state][action] = True

            G = 0
            for i, (_, _, r, _, _) in enumerate(trajectory[t:]):
                G += (gamma**i) * r

            Q[state][action] = Q[state][action] + alphas[e] * (G - Q[state][ac

            if next_state == goal_state and done:
                episode_reached_goal = True

        reached_goal_list.append(episode_reached_goal) # Append the flag for t
        Q_track[e] = Q.copy()
        pi_track.append(np.argmax(Q, axis=1))


    v = np.max(Q, axis=1)
    pi = np.argmax(Q, axis=1)
    return Q, v, pi, Q_track, pi_track, reached_goal_list # Return the list
```

In [22]:
```
# optimal_Q, optimal_V, optimal_pi, Q_track, pi_track = mc_control (env)
# print_state_value_function(optimal_Q, P, n_cols=4, prec=2, title='Action-val
# print_state_value_function(optimal_V, P, n_cols=4, prec=2, title='State-valu
# print_policy(optimal_pi, P)
```

In [25]:
```
optimal_Q, optimal_V, optimal_pi, Q_track, pi_track, reached_goal_list = mc_cc

# Calculate the percentage of episodes where the goal was reached
percentage_reached_goal = (sum(reached_goal_list) / len(reached_goal_list)) *
```

```
# Calculate the average discounted return
# Need to re-run the trajectories to calculate returns since they are not stor
total_discounted_returns = []
# Get the gamma value used in mc_control - assuming default 1.0 if not explici
mc_gamma = 1.0 # Default value, if mc_control was called with a different gamm
for e in tqdm(range(len(pi_track)), leave=False):
    trajectory = generate_trajectory(lambda state, Q, epsilon: pi_track[e][sta
    discounted_return = 0
    for i, (_, _, r, _, _) in enumerate(trajectory):
        discounted_return += (mc_gamma**i) * r  # Use the correct gamma variab
    total_discounted_returns.append(discounted_return)

average_discounted_return = np.mean(total_discounted_returns)


print(f"Percentage of episodes where the goal was reached: {percentage_reached
print(f"Average discounted return: {average_discounted_return:.4f}")

print_state_value_function(optimal_Q, P, n_cols=4, prec=2, title='Action-value
print_state_value_function(optimal_V, P, n_cols=4, prec=2, title='State-value
print_policy(optimal_pi, P)
print('JEEVANESH')
print('REG: 212222243002')
```

```
Percentage of episodes where the goal was reached: 7.63%
Average discounted return: 0.2690
Action-value function:
| 00 [0.24 0.23 0.19 0.18] | 01 [0.07 0.05 0.03 0.21] | 02 [0.07 0.12 0.09 0.0
9] | 03 [0.04 0.02 0.   0.  ] |
| 04 [0.26 0.18 0.15 0.1 ] |           | 06 [0.17 0.11 0.07 0.  ] |           |
| 08 [0.17 0.14 0.21 0.27] | 09 [0.14 0.34 0.13 0.12] | 10 [0.41 0.15 0.18 0.1
] |           |
|           | 13 [0.11 0.19 0.1  0.35] | 14 [0.13 0.5  0.67 0.46] |           |
State-value function:
| 00   0.24 | 01   0.21 | 02   0.12 | 03   0.04 |
| 04   0.26 |           | 06   0.17 |           |
| 08   0.27 | 09   0.34 | 10   0.41 |           |
|           | 13   0.35 | 14   0.67 |           |
Policy:
| 00      < | 01      ^ | 02      v | 03      < |
| 04      < |           | 06      < |           |
| 08      ^ | 09      v | 10      < |           |
|           | 13      ^ | 14      > |           |
JEEVANESH
REG: 212222243002
```

In [ ]: