Frozen Lake

This environment is part of the Toy Text environments which contains general information about the environment.

Action Space	`Discrete(4)`
Observation Space	`Discrete(16)`
import	`gymnasium.make("FrozenLake-v1")`

Frozen lake involves crossing a frozen lake from start to goal without falling into any holes by walking over the frozen lake. The player may not always move in the intended direction due to the slippery nature of the frozen lake.

Description

The game starts with the player at location [0,0] of the frozen lake grid world with the goal located at far extent of the world e.g. [3,3] for the 4x4 environment.

Holes in the ice are distributed in set locations when using a pre-determined map or in random locations when a random map is generated.

The player makes moves until they reach the goal or fall in a hole.

The lake is slippery (unless disabled) so the player may move perpendicular to the intended direction sometimes (see `is_slippery`).

Randomly generated worlds will always have a path to the goal.

Elf and stool from https://franuka.itch.io/rpg-snow-tileset. All other assets by Mel Tillery http://www.cyaneus.com/.

The action shape is `(1,)` in the range `{0, 3}` indicating which direction to move the player.

- 0: Move left
- 1: Move down
- 2: Move right
- 3: Move up

Observation Space

The observation is a value representing the player's current position as current_row * ncols + current_col (where both the row and col start at 0).

For example, the goal position in the 4x4 map can be calculated as follows: 3 * 4 + 3 = 15. The number of possible observations is dependent on the size of the map.

The observation is returned as an `int()`.

Starting State

The episode starts with the player in state `[0]` (location [0, 0]).

Rewards

Reward schedule:

- Reach goal: +1
- Reach hole: 0
- Reach frozen: 0

Episode End

The episode ends if the following happens:

- Termination:
  1. The player moves into a hole.
  2. The player reaches the goal at `max(nrow) * max(ncol) - 1` (location `[max(nrow)-1, max(ncol)-1]`).
- Truncation (when using the time_limit wrapper):
  1. The length of the episode is 100 for 4x4 environment, 200 for FrozenLake8x8-v1 environment.

`step()` and `reset()` return a dict with the following keys:

- p - transition probability for the state.

See `is_slippery` for transition probability information.

Arguments

```
import gymnasium as gym
gym.make('FrozenLake-v1', desc=None, map_name="4x4", is_slippery=True)
```

`desc=None`: Used to specify maps non-preloaded maps.

Specify a custom map.

```
    desc=["SFFF", "FHFH", "FFFH", "HFFG"].
```

The tile letters denote:

- "S" for Start tile
- "G" for Goal tile
- "F" for frozen tile
- "H" for a tile with a hole

A random generated map can be specified by calling the function `generate_random_map`.

```
from gymnasium.envs.toy_text.frozen_lake import generate_random_map

gym.make('FrozenLake-v1', desc=generate_random_map(size=8))
```

`map_name="4x4"`: ID to use any of the preloaded maps.

```
        "FHFH",
        "FFFH",
        "HFFG"
        ]

    "8x8": [
        "SFFFFFFF",
        "FFFFFFFF",
        "FFFHFFFF",
        "FFFFFHFF",
        "FFFHFFFF",
        "FHHFFFHF",
        "FHFFHFHF",
        "FFFHFFFG",
    ]
```

If `desc=None` then `map_name` will be used. If both `desc` and `map_name` are `None` a random 8x8 map with 80% of locations frozen will be generated.

`is_slippery=True`: If true the player will move in intended direction with probability of 1/3 else will move in either perpendicular direction with equal probability of 1/3 in both directions.

For example, if action is left and is_slippery is True, then:

- P(move left)=1/3
- P(move up)=1/3
- P(move down)=1/3

Version History

- v1: Bug fixes to rewards
- v0: Initial version release