UWAGA: Wczytaj do Colab plik **frozen_lake.py** (intrukcja w pliku **COLAB_instrukcja.pdf**)

FrozenLake 1

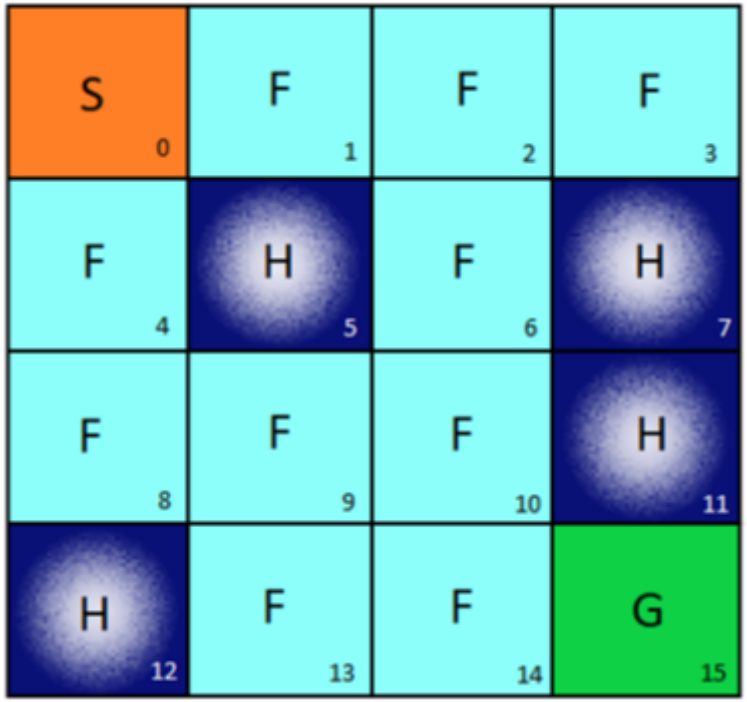
Wprowadzenie

Agent porusza się w świecie złożonym z **16 pól (stanów)**. Stany są ponumerowane od 0 do 15.

Niektóre pola siatki są dostępne do chodzenia (**F**-frozen), a inne są przeręblami (**H**-hole).

Możliwe są 4 akcje: **0 - LEFT, 1 - DOWN, 2 - RIGHT, 3 - UP**

Agent jest nagradzany (**R=1**) za dotarcie do pola **G**. W pozostałych przypadkach **R=0.**



Załadowanie biblioteki (wcześniej konieczne załadowanie pliku **frozen_lake.py** do Colaba - instrukcja w pliku **PDF**).

```
import gym
```

```
from frozen_lake import FrozenLakeEnv
```

Wczytanie środowiska:

```
env=gym.make("FrozenLake-v0")
```

```
env = FrozenLakeEnv()
```

```
env = gym.make("FrozenLake-v0", is_slippery=False)
```

Sprawdzamy ilość możliwych stanów (16) i akcji (4)

```
print(env.nS)
print(env.nA)
```

```
16
4
```

Dynamika

Dynamika opisana jest za pomoca: `env.P[s][a]`

gdzie: **s** to **stan** (0,1,2,...,15), **a** to **akcja** (0,1,2,3).

Rozważmy przykład: w stanie 0 agent wykonuje akcję 1 (porusza się w dół):

```
env.P[0][1]
```

```
[(0.3333333333333333, 0, 0.0, False),
 (0.3333333333333333, 4, 0.0, False),
 (0.3333333333333333, 1, 0.0, False)]
```

Powyższą czwórkę interperetujemy jako: **(prawdopodobieństwo, nowy stan, nagroda, czy koniec?)**.

Czyli w powyższym przykładzie: po wykonaniu w stanie 0 akcji 1 prawdopodobieństwo przejścia do stanu 4 wynosi 1, nagroda 0, agent nie wpadł do przerębli ani nie dotarł do pola G.

Polecenie 1 (do uzupełnienia)

Sprawdź dynamikę dla dla następujących przypadków:

W **stanie 1** agent **przechodzi w dół**:

```
env.P[0][1]
```

```
[(0.3333333333333333, 0, 0.0, False),
 (0.3333333333333333, 4, 0.0, False),
 (0.3333333333333333, 1, 0.0, False)]
```

W **stanie 10** agent **przechodzi w lewo**:

```
env.P[9][0]
```

```
[(0.3333333333333333, 5, 0.0, True),
```

```
 (0.3333333333333333, 8, 0.0, False),
 (0.3333333333333333, 13, 0.0, False)]
```

W **stanie 14** agent **przechodzi w prawo**:

```
env.P[13][2]
```

```
[(0.3333333333333333, 13, 0.0, False),
 (0.3333333333333333, 14, 0.0, False),
 (0.3333333333333333, 9, 0.0, False)]
```

Poruszanie i wizualizacja

W świecie FrozenLake możemy się poruszać wykonując 4 akcje (omówione powyżej). Podgląd położenia uzyskujemy za pomocą `env.render()` (wcześniej resetujemy położenie agenta).

```
env.reset()
env.render()
```

```
SFFF
FHFH
FFFH
HFFG
```

Wykonajmy dwa ruchy w prawo i jeden w dół:

```
env.reset()
env.step(2)
env.step(2)
env.step(1)
```

```
env.render()
```

```
  (Down)
SFFF
FHFH
FFFH
HFFG
```

Metoda `step` zwraca krotkę (**nowy stan**, **nagroda**, **czy koniec ruchu**,_). Koniec następuje wtedy gdy agent wpadł do przerębli lub dotarł do pola 15 - GOAL). Sprawdźmy to.

Z pola początkowego 0 agent rusza się w prawo (akcja - 2) na pole 1 i zdobywa nagrodę 0:

```
env.reset()
env.step(2)
```

```
(1, 0.0, False, {'prob': 0.3333333333333333})
```

Agent kontynuuje ruch: rusza się w prawo (akcja - 2) na pole 2 i zdobywa nagrodę 0:

```
env.step(2)
```

```
(2, 0.0, False, {'prob': 0.3333333333333333})
```

Polecenie 2 (do uzupełnienia)

Przeprowadź agenta dowolną droga z pola 0 do pola 15 (GOAL). Sprawdź czy nagroda po wejściu na to pole wynosi 1.

```
env.reset()
```

```
env.step(1)
env.step(1)
env.step(2)
env.step(2)
env.step(1)
env.step(2)
env.render()
```

```
  (Right)
SFFF
FHFH
FFFH
HFFG
```

Ruch agenta w pętli

Ruch agenta można zapętlić. Na razie akcja w każdym stanie generowana jest losowo (wykorzystujemy metodę: `env.action_space.sample()`). Agent wykona 10 akcji.

UWAGA: kiedy agent jest na polu oznaczonym **H** (stany 5,7,11,12) **dowolna akcja pozostawia go na tym polu** (agent nie może uciec z przerębli).

```
env.reset()
for i in range(10):
    action = env.action_space.sample()
    obs, rew, fin, _ = env.step(action)
    print("Action=",action,"State =",obs,"Reward =",rew,"End =",fin)
```

```
 Action= 0 State = 0 Reward = 0.0 End = False
 Action= 3 State = 0 Reward = 0.0 End = False
 Action= 3 State = 0 Reward = 0.0 End = False
 Action= 2 State = 1 Reward = 0.0 End = False
 Action= 0 State = 0 Reward = 0.0 End = False
 Action= 1 State = 4 Reward = 0.0 End = False
 Action= 2 State = 5 Reward = 0.0 End = True
```

```
Action= 0 State = 5 Reward = 0 End = True
Action= 1 State = 5 Reward = 0 End = True
Action= 2 State = 5 Reward = 0 End = True
```

Polecenie 3 (do uzupełnienia)

Sprawdź **czy możliwe jest dotarcie agenta do pola G** w przypadku gdy **akcje są generowane losowo**. Przeprowadź dużą liczbę testów (zbuduj odpowiednią pętlę). Zawsze gdy agent wpadnie do przerębli przerwij pętlę.

Poniżej wpisz kod:

```
MAX_ITERATIONS = 100
for i in range(MAX_ITERATIONS):
    random_action = env.action_space.sample()
    new_state, reward, done, info = env.step(
       random_action)
    env.render()
    if done:
        break

       (Left)
     SFFF
     FHFH
     FFFH
     HFFG
```

TWOJE PODSUMOWANIE TESTÓW: da się tak zrobić

1. Element listy
2. Element listy

1. Element listy
2. Element listy