

UWAGA: Wczytaj do Colab plik **frozen_lake_slippery.py** lub **frozen_lake.py** (instrukcja w pliku **COLA**

▼ FrozenLake 3

```
from frozen_lake import FrozenLakeEnv
#from frozen_lake_slippery import FrozenLakeEnv
import numpy as np
```

```
env = FrozenLakeEnv()
```

Chcemy napisać funkcję, która korzystając z określonych wartości zwrotów $V(s)$ (dla wszystkich s konkretnego stanu s i dla wszystkich akcji a możliwych do wykonania w stanie s).

Założmy, że mamy dane $V(s)$ takie jak na rysunku poniżej:

$V(s)$

0.16807	0.2401	0.343	0.2401
0.2401	0.	0.49	0.
0.343	0.49	0.7	0.
0.	0.7	1.	0.



Wartości zwrotów $V(s)$ dla każdego stanu zapiszemy w tablicy:

```
V = np.array([0.16807,0.2401,0.343,0.2401,0.2401,0.,0.49,0.,0.343,0.49,0.7,0.,0.,0.7,1.,0.
print(V)
```

```
↳ [0.16807 0.2401 0.343 0.2401 0.2401 0. 0.49 0. 0.343
    0.49 0.7 0. 0. 0.7 1. 0. ]
```

Funkcję zdefiniujemy korzystając z formuły:

$$q_{\pi}(s, a) = \sum_{s', r} p(s', r | s, a) [r + \gamma v_{\pi}(s')]$$

▼ Polecenie 1 (do uzupełnienia)

Funkcja dla danego s i znanego V ma zwracać wartości zwrotów **dla czterech akcji** możliwych do wyglądać tak (**UZUPEŁNIJ DEFINICJĘ FUNKCJI**):

```
def Q_from_V(env, V, s, gamma=0.99):
    Q = np.zeros(env.nA)

    for action in range(env.nA):
        for next_state in range(len(env.P[s][action])):
            prob, next_state, reward, done = env.P[s][action][next_state]
            Q[action] += prob * (reward + gamma * V[next_state])
            #DO UZUPEŁNIENIA

    return Q
```

OBJAŚNIENIE: Argumenty funkcji (oprócz V i s) to zmienna **env** związana ze środowiskiem **Frozen** powyższym wzorze. Q zdefiniowane w pierwszej linijce definicji to **4 elementowa tablica złożona z** wykonać w środowisku określonym przez **env**). W pętli **for** mają być wyliczone **wartości zwrotów c** tę mają być zapisane w tablicy Q . Funkcja zwróci tę tablicę.

▼ Polecenie 2 (do uzupełnienia)

Przetestuj działanie funkcji **Q_from_V** dla domyślnej wartości **gamma=0,99**

Wartości zwrotów dla 4 akcji w stanie **s=0**:

```
print(Q_from_V(env, V, 0))
#zwrot dla akcji 0
#zwrot dla akcji 1
#zwrot dla akcji 2
#zwrot dla akcji 3

↳ [0.1663893 0.237699 0.237699 0.1663893]
```

Wartości zwrotów dla 4 akcji w stanie **s=8**:

```
print(Q_from_V(env,V,8))
#zwrot dla akcji 0
#zwrot dla akcji 1
#zwrot dla akcji 2
#zwrot dla akcji 3
```

↳ [0.33957 0. 0.4851 0.237699]

Wartości zwrotów dla 4 akcji w stanie **s=15**:

```
print(Q_from_V(env,V,15))
#zwrot dla akcji 0
#zwrot dla akcji 1
#zwrot dla akcji 2
#zwrot dla akcji 3
```

↳ [0. 0. 0. 0.]

Przetestuj działanie funkcji **Q_from_V** dla mniejszej wartości **gamma=0.1**

Wartości zwrotów dla 4 akcji w stanie **s=0**:

```
print(Q_from_V(env,V,0,gamma=0.1))
#zwrot dla akcji 0
#zwrot dla akcji 1
#zwrot dla akcji 2
#zwrot dla akcji 3
```

↳ [0.016807 0.02401 0.02401 0.016807]

Wartości zwrotów dla 4 akcji w stanie **s=8**:

```
print(Q_from_V(env,V,8,0.1))
#zwrot dla akcji 0
#zwrot dla akcji 1
#zwrot dla akcji 2
#zwrot dla akcji 3
```

↳ [0.0343 0. 0.049 0.02401]

Wartości zwrotów dla 4 akcji w stanie **s=15**:

```
print(Q_from_V(env,V,15,0.1))
#zwrot dla akcji 0
#zwrot dla akcji 1
#zwrot dla akcji 2
#zwrot dla akcji 3
```

```
#zwrot dla akcji 3
```

```
[0. 0. 0. 0.]
```

▼ Polecenie 3 (do uzupełnienia)

Jaki wpływ na wyniki miała zmiana wartości parametru **gamma** i dlaczego taki?



WPISZ ODPOWIEDŹ:

Zmiana wartości parametru gamma miała duży wpływ na wyniki, ponieważ jest ona mnożona przez wartość zwrotu dla danego stanu