Ćwiczenie nr 6
Wprowadzenie do sztucznej inteligencji

Autor
Maciej Prostak

1 Wprowadzenie

Tematem zadania są metody uczenia ze wzmocnieniem. Zaimplementowany zostanie algorytm Q-Learning, służący do wyznaczenia polityki decyzyjnej. Zadanie jakie należy rozwiązać to wyszukiwanie najkrótszej trasy w labiryncie. Agent, który zamierzamy nauczyć będzie mógł wykonywać cztery ruchy na mapie o układzie w stylu Manhattan.

1.1 Technologia

Rozwiązanie zostało zaimplementowane w języku Python (3.9.0) z wykorzystaniem biblioteki pygame (2.1.2) oraz numpy (1.12.4). Dodatkowo dla poprawy czytelności kodu zostały dołączone biblioteki black (21.9b0), flake8 (4.0.1) oraz isort (5.9.3).

1.2 Akcje

Agent posiada cztery ruchy (akcje):

- ruch w górę,
- ruch w dół,
- ruch w lewo,
- ruch w prawo.

Jeżeli agent najedzie na końcowy punkt, gra zostanie zakończona.

1.3 Stany

Domyślnie na mapie o wielkości 8x8 będziemy posiadać 64 różne stany w których może pojawić się agent. Miejsca w którym są ściany (stażyści) również został uwzględniony jako stan. Nasza plansza jest tablicą dwuwymiarową, a w tablicy polityki decyzynej będzie ona spłaszona do wielkości 64. Odpowiednie punkty (x, y) będą zamieniane na indeks stanu za pomocą wzoru x*8 + y.

1.4 Nagroda

W naszym przypadku za każdy krok w czasie zawsze nagroda będzie równa -1. Nie jest ona równa 0 jako, że chcemy odnaleźć najkrótszą trasę. Agent po niepoprawnym wjechaniu w ścianę (stażystę) otrzyma nagrodę o wartości -10. Przypadek ten został w moim programie tak zaimplementowany, że gra nie zostaje zakończona, tylko agent pozostaje w tym samym miejscu. Innym możliwym przypadkiem brzegowym jest podjęcia akcji wyjścia poza planszy na krawędziach labiryntu. W tych przypadkach zachodzi ten sam wynik co w przypadkach ścian. Agent nie ruszy się, a otrzyma również nagrodę równą -10.

1.5 Polityka

Polityka decyzyjna, którą wyznaczymy po odpowiednim trenowaniu agenta będzie wyznacza z Q tablicy (o rozmiarach 64x4). Dla konkretnego stanu będziemy wyciągać indeks akcji o największej wartości: ArgMax(Q[indeks_stanu]).

2 Wyniki

2.1 Parametr learning rate

Zmiana parametru learning rate mówi nam jak bardzo nadpisywać stare wartości w tablicy Q. W podanych przykładach można zauważyć, że w przypadku lr=1 dość szybko otrzymujemy poprawną politykę. W przypadku dużej wartości (lr=2) otrzymujemy dość chaotyczne wyniki. Tylko w przykładzie lr=2 polityka została błędnie wyznaczona.

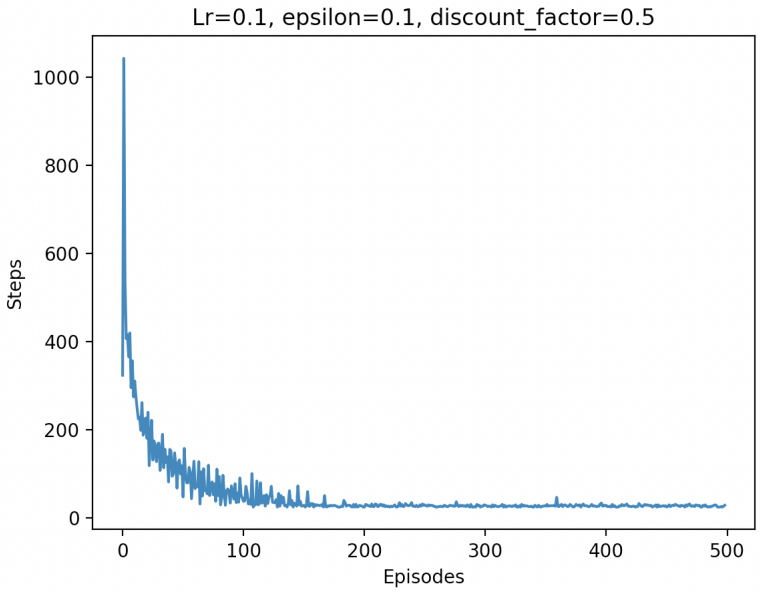

Fig. 1 Wykres ilości kroków do ilości epizodów.

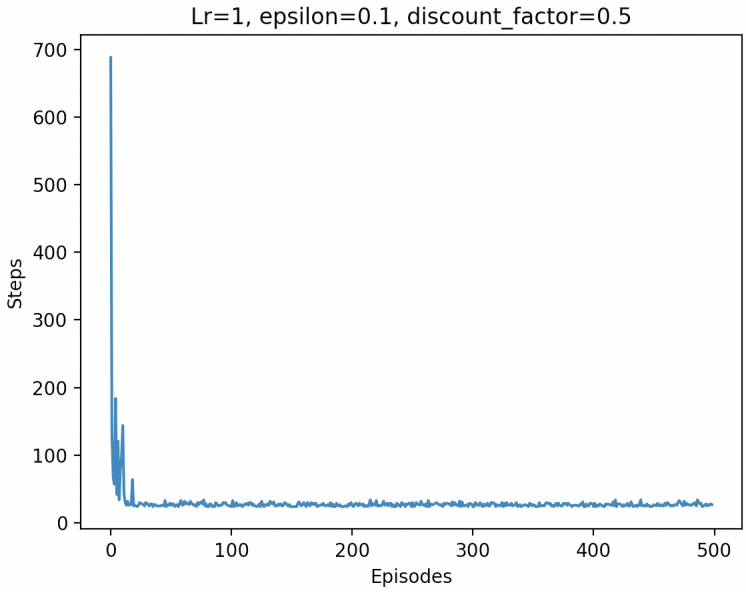

Fig. 2 Wykres ilości kroków do ilości epizodów.

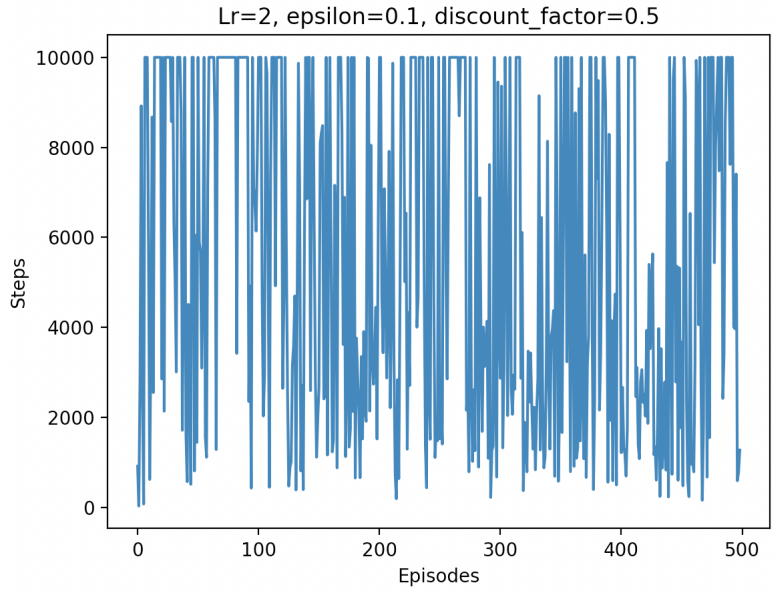

Fig. 3 Wykres ilości kroków do ilości epizodów.

2.2 Parametr epsilon

Parametr epsilon oznacza jak często agent podejmuje losową decyzję. W podanych przypadkach widzimy, że epsilon dopiero przy dość wysokiej wartości wpływa na ilości kroków w trakcie trenowania. Nie wpłynęło to jednak negatywnie na wyznaczoną politykę decyzyjną.

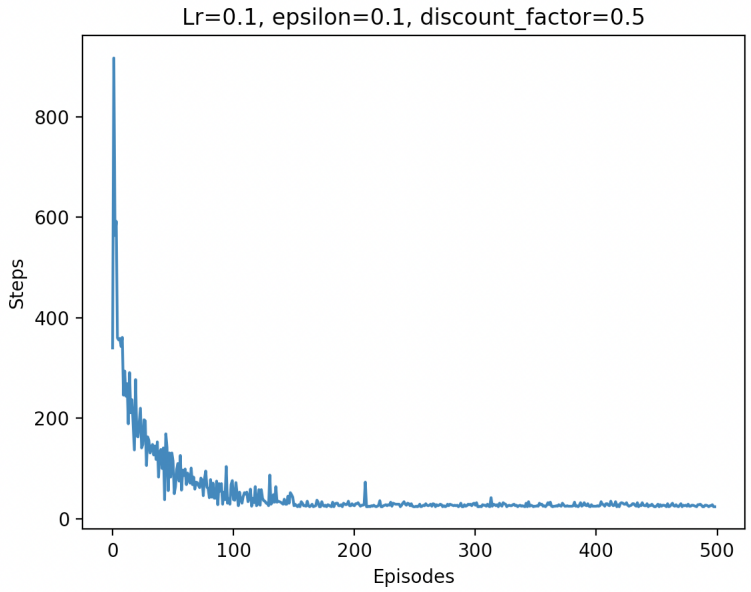

Fig. 4 Wykres ilości kroków do ilości epizodów.

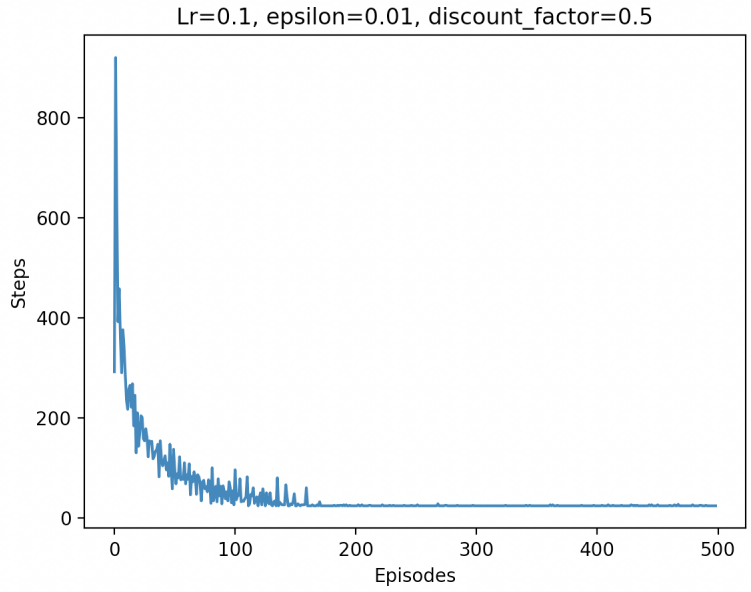

Fig. 5 Wykres ilości kroków do ilości epizodów.

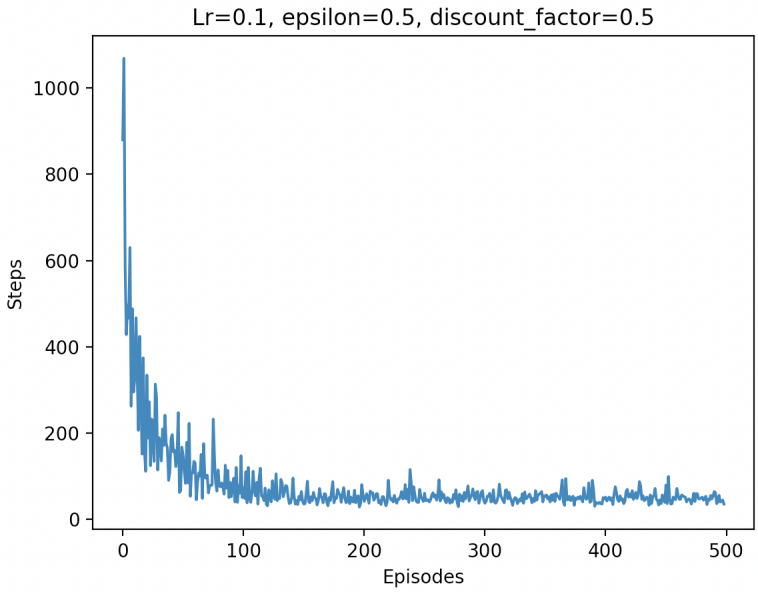

Fig. 6 Wykres ilości kroków do ilości epizodów.

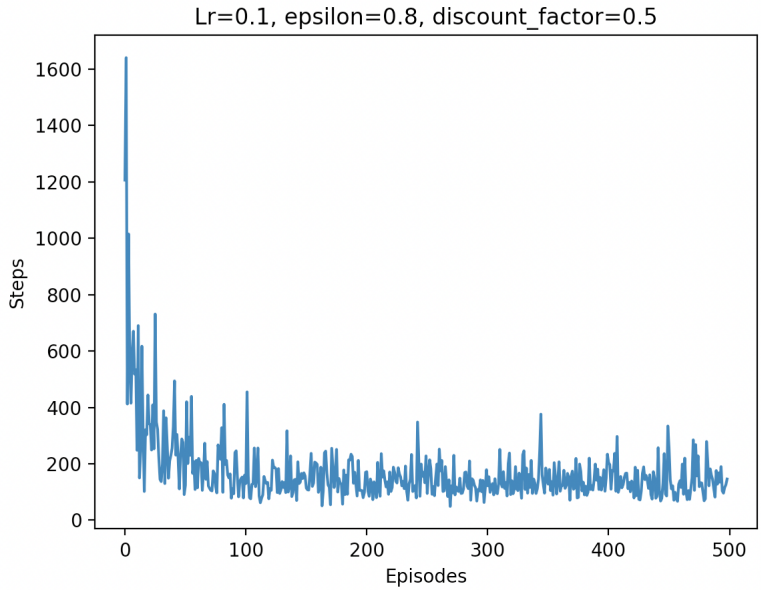

Fig. 7 Wykres ilości kroków do ilości epizodów.

2.3 Parametr discount factor

W przypadkach podania wartości 0.1 i 0.2 dla discount factor, w pewnym momencie zacząłem otrzymywać błędne wyniki.

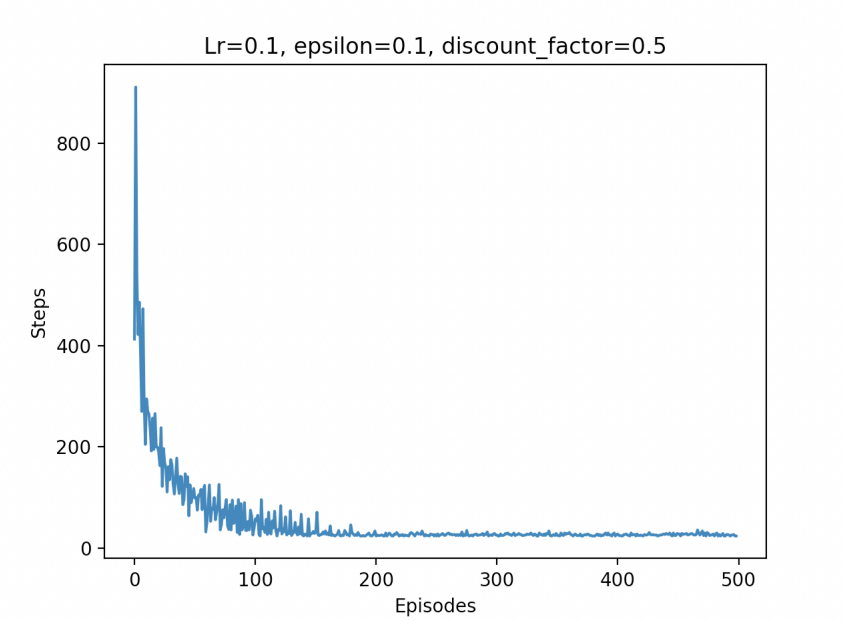

Fig. 8 Wykres ilości kroków do ilości epizodów.

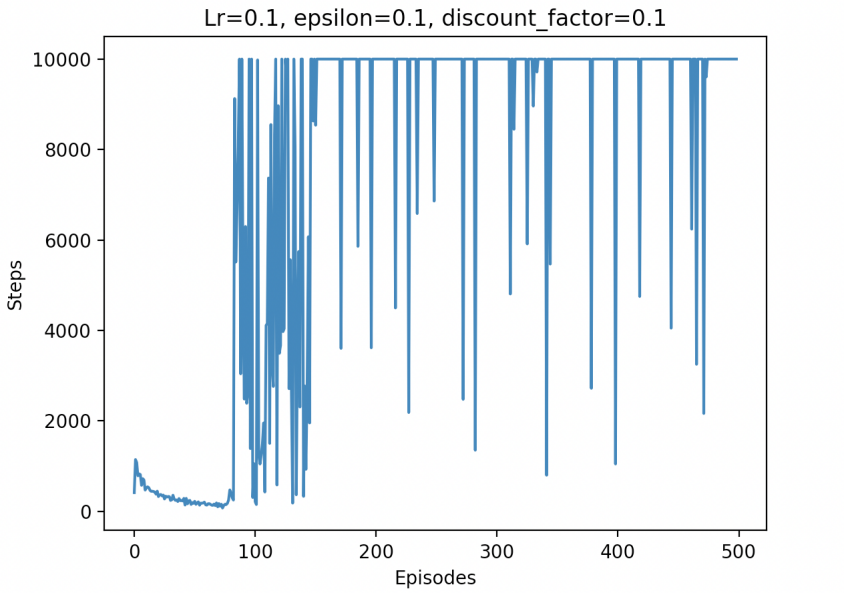

Fig. 9 Wykres ilości kroków do ilości epizodów.

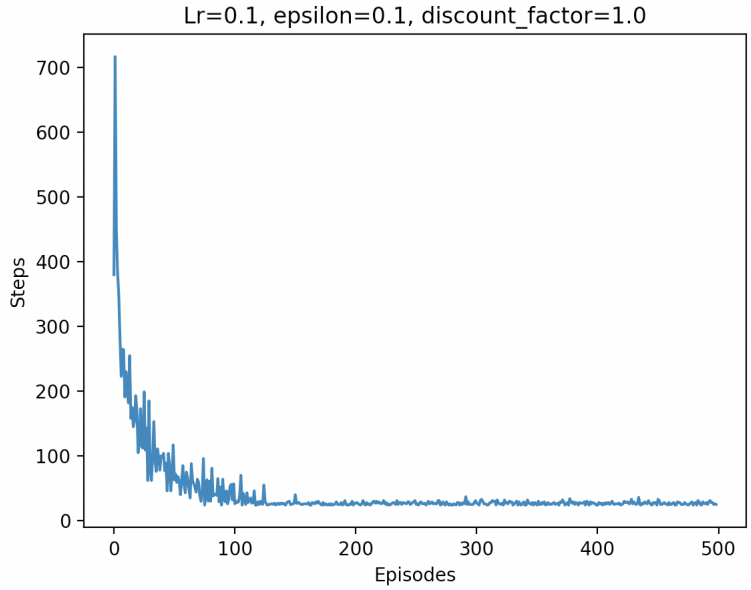

Fig. 10 Wykres ilości kroków do ilości epizodów.

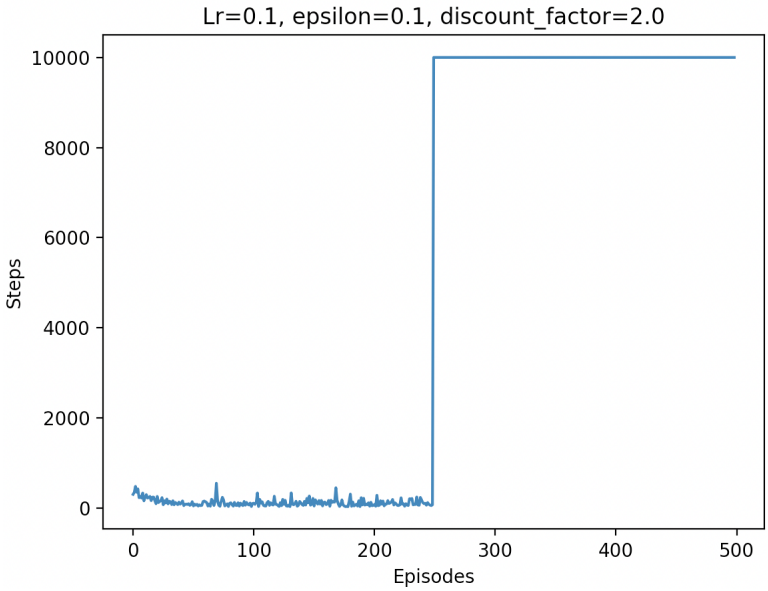

Fig. 11 Wykres ilości kroków do ilości epizodów.

2.4 Losowy agent

W zadaniu należało również zaimplementować agenta podejmującego losowe akcje. Na wykresie podanym poniżej widzimy, że agent musiał średnio wykonać parę tysięcy kroków w celu dojścia do celu. W porównaniu z nauczonym agentem, jest to bardzo duża różnica. Należy również podkreślić, że agent losowy nie kończy gry w momencie "najechania" na ścianę, tylko otrzymuje negatywną nagrodę.

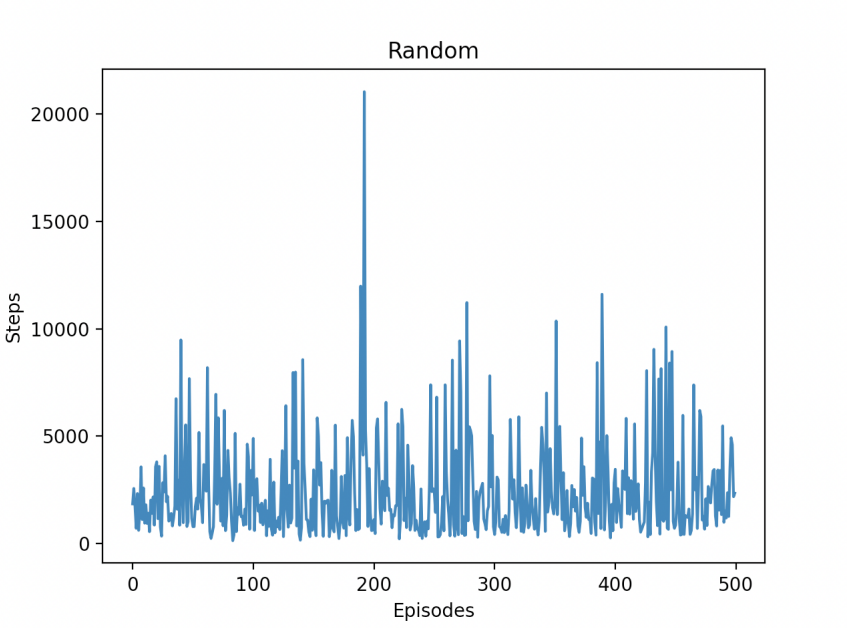

Fig. 12 Wykres ilości kroków do ilości epizodów w przypadku losowego agenta.

2.5 Poprawnie wyznaczona polityka decyzyjna

Można zaobserwować wizualizację labiryntu z nałożoną polityką decyzyjną w postaci strzałek (Fig. 13). Kwadrat o kolorze czerwonym został oznaczony agent. Kwadraty z obwódką (bez wypełnienia) przedstawiają odpowiednio początek i koniec labiryntu.

2.6 Niepoprawnie polityka decyzyjna

Przy znacznym zmniejszeniu learning rate i niedostatecznej ilości epizodów można zauważyć niepoprawną wyznaczoną politykę decyzyjną (Fig. 14). Większość trasy została poprawnie wyznaczona, ale początek trasy okazała się zapętlona.

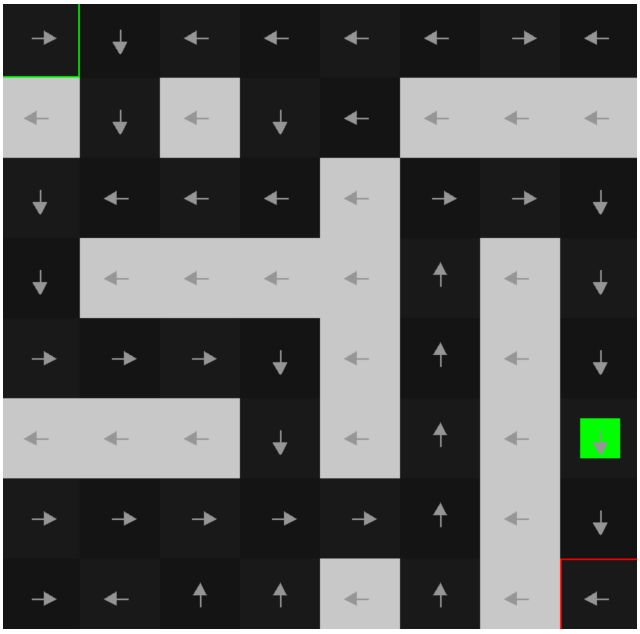

Fig. 13 Wizualizacja labiryntu z poprawną wyznaczoną polityką decyzyjną.

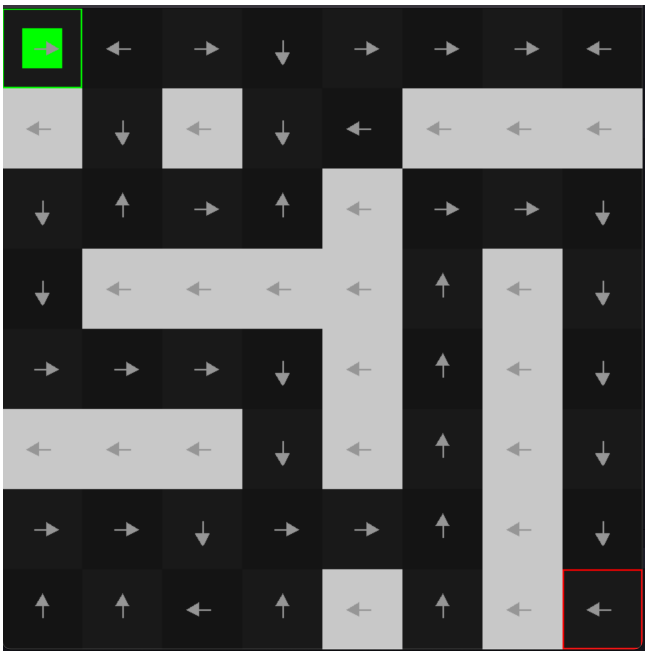

Fig. 14 Wizualizacja labiryntu z niepoprawną wyznaczoną polityką decyzyjną.

3 Wnioski

Algorytm Q-Learning okazał się dość prosty w implementacji. Cięższe jednak okazały się dobranie parametrów, ale jednak po poprawnym dobraniu wartości dla naszego zadania wyniki okazały się dość powtarzalne dla różnych stworzonych labiryntów. Zauważyłem również, że zdarzały się przypadki, w której wyniki na wykresach wydawały się obiecujące, ale w przypadku puszczenia agenta polityka była błędnie wyznaczona i agent zapętlał się na początku labiryntu.