Alle runs zijn met 10000 episodes uitgevoerd.

* 1. Monte-Carlo Policy Evaluation

|  |  |
| --- | --- |
| Discount 1, Optimal Policy |  |
| Discount 0.9, Optimal Policy |  |
| Discount 1, Random Policy |  |
| Discount 0.9 Random Policy |  |

* 1. Temporal Difference Learning

|  |  |
| --- | --- |
| Discount 1, Optimal Policy |  |
| Discount 0.9, Optimal Policy |  |
| Discount 1, Random Policy |  |
| Discount 0.9 Random Policy |  |

2.1 On-policy first-visit Monte-Carlo Control

|  |  |
| --- | --- |
| Discount 1, Optimal Policy |  |
| Discount 0.9, Optimal Policy |  |
| Discount 1, Random Policy | Error omdat het niet met sprongen van acties kan omgaan omdat het first episode gebruikt, wanneer ik first episode uitzet krijg ik een error omdat stil staan geen geldige actie is. |
| Discount 0.9 Random Policy | Zie reden hier boven. |

2.2 SARSA

|  |  |
| --- | --- |
| Discount 1, Optimal Policy |  |
| Discount 0.9, Optimal Policy |  |
| Discount 1, Random Policy |  |
| Discount 0.9 Random Policy |  |

2.3 SARSAMAX a.k.a Q-Learning

|  |  |
| --- | --- |
| Discount 1, Optimal Policy |  |
| Discount 0.9, Optimal Policy |  |
| Discount 1, Random Policy |  |
| Discount 0.9 Random Policy |  |

Overal ben ik tevreden met deze opdracht, de code is relatief clean en de resultaten zijn bijna allemaal zoals gewenst. Ik vond het wel verassend hoe q learning de enige is die met de random policy tot de optimal value function uitkomt en de rest niet, dit is waarschijnlijk een fout in mijn implementatie. Ik heb meerdere dingen geprobeerd zoals naar voorbeelden kijken en verschillende randomness te gebruiken maar voor alsnog heb ik deze resultaten er uit gekregen.

Ik vond deze opdracht een stuk pittiger dan de vorige maar ik heb het prima naar mij gevoel afgesloten. De resultaten zijn zoals verwacht, wanneer je een discount van 0.9 gebruikt zal er minder op toekomstige waardes vertrouwd worden maar als we de optimal policy gebruiken hoort de vertrouwen juist 100% te zijn.