Felix Bühler - 2973140
Jan Leusmann - 2893121
Jamie Ullerich - 3141241

Exercise 2

Task 1 - Formulating Problems

a) The states would be the configuration of the chess pieces on the board. These are discrete and finite states, but not every sate can be reached sometimes, depending on the current configuration of pieces. The actions are the specific actions which all the pieces in the current state can perform. These are discrete and finite, as well. For rewards, it would be good to punish when a piece is removed and reward actions which lead to check or other movements which may lead to winning. It is important, that there is a great reward for actual winning to avoid that the agent finds a way to for example only remove pieces and instead optimises achieving the real goal of winning.

b) As states, we would use the position of the arm. These are therefore discrete and finite. This way, the actions would be the movement of the arm, which are finite as well. A positive reward should be given if the robot can pick objects and performs fast, this means taking a long time or dropping objects would be punished with a negative reward.

c) There could be two discrete states, stable and unstable. These indicate whether the drone must be stabilised or not. As actions, the movement of the drone, e.g. left, right, up, down, forward, back could be used. These are finite and can be continuous since every position is possible. A positive reward can be given if the drone flies balanced and a negative one if it needs a lot of time to balance out.

d) Another example could be the development of analogue films. Here are the states data of sensors, because when developing films, certain chemicals have to be kept at an exact temperature for a certain time. As actions, temperature or other factors can be changed. These would be finite on a continuous scale. A reward is positive if the film is successfully developed or negative if the film is damaged by the wrong settings.

Task 2 - Value Functions

1.

2. Definitions given in lecture:
   $v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \quad \forall s \in S$
   $q_\pi(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$
   $\rightarrow$ insert $q_\pi(s, a)$ into $v_\pi(s)$:
   $v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a)$

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

Task 3 - Bruteforce the Policy Space

a)

$|policies| = |A|^{|S|}$
in our example: $4^9 = 262144$

b)

```
Value function for policy_left (always going left):
[0.          0.          0.53691275 0.          0.          1.47651007
0.          0.          5.         ]
Value function for policy_right (always going right):
```

Felix Bühler - 2973140
Jan Leusmann - 2893121
Jamie Ullerich - 3141241

```
[0.41401777 0.77456266 1.31147541 0.36398621 0.8185719  2.29508197
0.13235862 0.         5.        ]
```

As expected the 'going_right'-method is a far better choice. The 'going_left'-method can never reach the goal except from the left collumn. The 'going_right'-method can reach the goal from all states, but accidently fall in the hole.

c)

```
Optimal value function:
[0.49756712 0.83213812 1.31147541 0.53617147 0.97690441 2.29508197
0.3063837  0.         5.        ]
number optimal policies:
2
optimal policies:
[[1 2 2 3 3 2 0 0 0]
[2 2 2 3 3 2 0 0 0]]
```

The only choice is in the first state going down or right. Every other state has a single policy.

d)

the calculation will take much longer. The assumtion to calculate every possible policy is not practical if working with bigger map sizes.