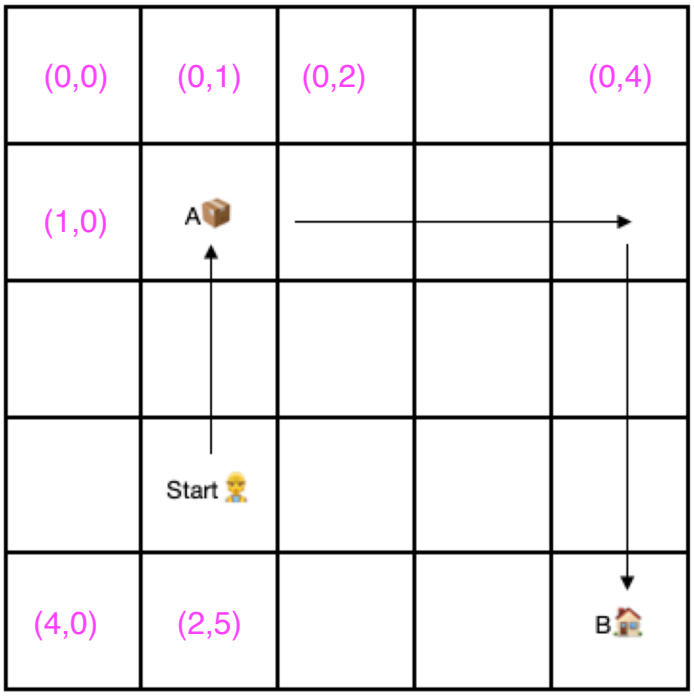
Lab - Week 5

A First Grid World

The purpose of this exercise is to ensure that you have a working software basis to perform RL experiments in grid worlds. What you implement today will also from the basis for your assignment. Today we will just implement a "dumb" grid world and we will add learning in the next week.

1. Implementing a simple Grid World



- grid (world / env)
- agent = [(x1,y1), (x2,y2), (x3,y3)…]
- loaded = [T, F, T, F….]
- goal (A,B)

State comprises:
agent position, loaded (whether it has pickup at A), (XA, YA)

action is not part of the state; action is carried based on the policy , and the state changed based on the action

You will write code for a single agent in a square grid world of size *n* to learn a simple transport task. The agent's task is to pick up the item at location *A* and deliver it to a fixed target location *B*. A is not known in advance, ie. it varies each time the agent needs to solve the task. B is the bottom right corner of the grid at coordinates (*n*,*n*). The coordinates of *A* are part of the state information that the agent receives.

Your agent has four actions that it can execute: move north/move south/move west/move east (as in the examples given in the seminars). It starts at a random location. When it reaches location A it *automatically* picks up the item, when it reaches location B it *automatically* discharges the item. At this point, it has completed its task.

The task the agent has to learn is to pick up the load at A and deliver it to B taking as few steps as

possible regardless of its (random) starting position.

Tasks

1. Determine what the state space of this agent needs to look like so that it can meaningfully learn to determine its actions.
2. Implement a grid world, in which the agent can move and execute its task. To make the task manageable, we use a 5x5 grid world. However, your code should be set up to work for any size (so use parameters for the size and for the target location). We simply choose a smaller grid here to limit memory and time requirements for the training. Note that you will have to integrate this with a visualization in the final task. It is highly advisable that you consider this for your code design right from the beginning.

   Since we are not yet learning the correct behaviour, you should implement a fixed policy. This should either be a “good” policy, implementing the right behaviour or just a random policy. Ideally, you implement this policy as a function that takes the state of the agent as its (only) input and returns an action.
3. Determine a reward structure that will allow the agent to learn this task efficiently using reinforcement learning.