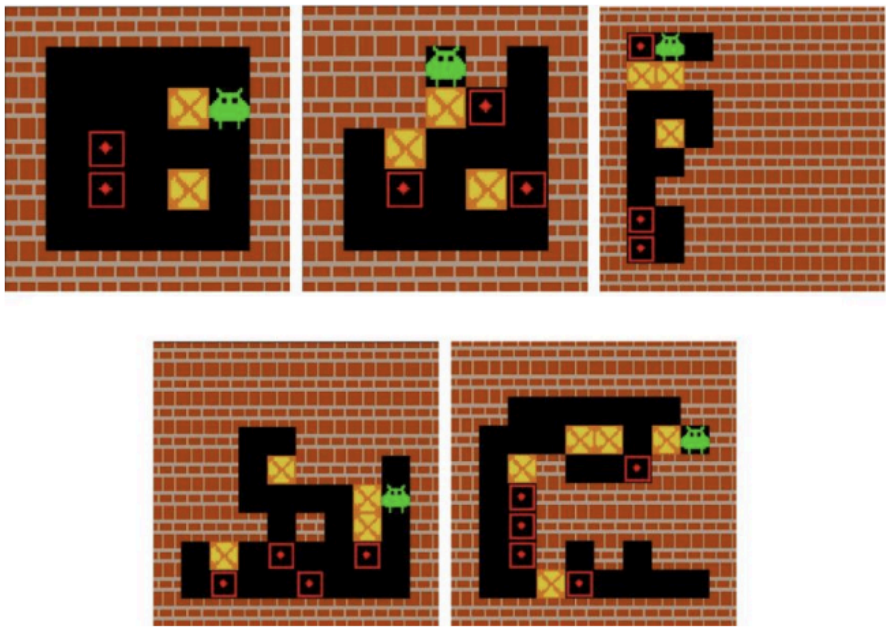
Intro to Reinforcement Learning
Final Project - 2024

The main goal of this final project is to summarize the main topics that we have discussed in the course using some practice and theory, and especially the second part of the course (Deep RL).

Project Definition:
In this project, you will solve a variation of the SOKOBAN environment.
The Environment: https://github.com/mpSchrader/gym-sokoban



Sokoban (meaning 'warehouse keeper') is a Japanese puzzle video game genre in which the player pushes crates or boxes around in a warehouse, trying to get them to storage locations.

The game is a **transportation puzzle**, where the player has to **push all boxes in the room on the storage locations/targets.**
The possibility of **making irreversible mistakes makes these puzzles so challenging**, especially for Reinforcement Learning algorithms, which mostly lack the ability to think ahead.

The room generation is random and therefore, will allow the training of Deep Neural Networks without overfitting on a set of predefined rooms.
The game is played on a board of squares, where each square is a floor or a wall. Some floor squares contain boxes, and some floor squares are marked as storage locations.

The player is confined to the board and may move horizontally or vertically onto empty squares (never through walls or boxes).

The player can move a box by walking up to it and pushing it to the square beyond. Boxes cannot be pulled, and they cannot be pushed to squares with walls or other boxes. The number of boxes equals the number of storage locations.
The puzzle is solved when all boxes are placed at storage locations.

The details of the environment, actions space, reward space, etc you can find in the attached Github repository.

You will find a demonstration of how to use/render the game (random actions) in the template notebook:
https://colab.research.google.com/drive/1QG-mvqB24enuAi182SIByqOfg6xyB86f?usp=sharing

Explore the environment documentation carefully and make sure you understand what is the environment, the observation space, the action space, and how are the rewards defined. Also, make sure you understand what's the difference between the different versions. After entering the notebook click on the "File" tab and then on "Save a copy into drive". You can then work on the project within your personal drive environment.

The main goal for all the environments/exercises is to "solve" as fast as you can (fewer episodes) - **this is a competition**.

You will find a running example of each of the above environments in the template notebook.

Guidelines:

- Start early, training often hits a barrier and improvements are hard, especially as each training might take a while.
- Take advantage of Google Colab tools and resources.
- **The main goal is to "solve" as fast as you can - this is a competition part.**
- You are allowed to apply a function over the reward (Reward Shaping) to make learning faster\easier. Explain in the repost if you did and what it actually does.
- You are allowed to make preprocessing and any enhancements you wish, as long as you explain what you did and why you think they help (sometimes it is unclear it's ok, most of the time there is a reason though).
- Supply a graph describing the average rewards on 100 episodes (after finishing the training phase). Note: stop an episode after 500 iterations (if it was not finished). Even if you did not solve any of the environments, you will still get points, share this.
- You are allowed to use existing code libraries, but, if you use any algorithm not learnt in class, you will have to explain its main ideas, advantages and disadvantages in the report. Also, report results (even if not highly optimized) of at least one algorithm that was learned in class for comparison.
- Report the main algorithm you chose, and what were the main difficulties and pros of using it.

- Note, this project simulates a (simple) real-world scenario where you will likely not rewrite the whole code yourselves, however, you also need to understand what you are doing and how this specific problem is different from others, in ways you can utilize for better performance.
- You can of course use existing DL platforms such as Keras, Tensorflow, and Pytorch. You can also use existing RL platforms such as gym stable baselines.
- Experiments - In your final report, compare different approaches, different parameters, different initialization values for the environment, different discount factors, epsilon-greedy, etc. You can use a method that encourages exploration. Show graphs (In the notebook and report) comparing the different parameters and different methods. In addition, Show visualizations, for example, Q, V of “interesting” states.
  As this project is open-ended, expect a working solution to get you most of the way, some improvements to get you a better score, and full scores would be awarded for few students with excellent visualization, robust analysis, and or proof of deep knowledge and experimentation.
- For the most successful experiment, in each exercise, Show (In the notebook and report) the number of steps it took the agent to solve the environment, and a convergence graph (Rewards according to the number of episodes). In addition, show two video clips (In the notebook). The first video clip shows the agent in the middle of the training process and the second video clip shows the agent after the learning is completed and converged, (for example, if it took you 40 episodes to converge, show how the agent tries to solve the environment after 20 episodes in the second video, show how the agent tries to solve the environment after 40 episodes). Even if you did not solve an environment - supply graphs describing the average rewards on 100 episodes (after finishing the training phase).
- All training outputs will be displayed in the notebook.
- Submit a final report in pdf format when your ID numbers are included in the file name.
- You need to supply your code as Google Colab notebooks. Such that the course team can run it.
- Write a clean code! Separate code cells and make extensive use of text cells for documentation. A sloppy notebook will result in a lower grade.
- Do not change anything in the notebook after the final submission date. A notebook that has been ran/changed after the submission date will be automatically disqualified.
- Environment code location: https://github.com/mpSchrader/gym-sokoban
- Report the results on the following 3 exercises, with Grid size: 112 X 112 (RGB):
  - **EX1** - **FIX SCENARIO** - Easy: One box, One Target, i.e. Every episode you get the same configuration, no need to regenerate scenarios.
  - **EX2** - **“Push & Pull”** - Medium: RANDOM GENERATED SCENARIO - One box, One Target.
  - **EX3** (OPTIONAL) - **“Push & Pull”** – Medium +: RANDOM GENERATED SCENARIO - **Two** boxes, **Two** Targets.
- Template for project submission: https://www.overleaf.com/read/hnfnzjsmstsg

Due Date:
Final Project submission is due **July 31th at 23:59 through Moodle only**.

Submission: Detailed report including graphs, source code links (to Google Colab), relevant images, and a README file containing relevant explanations for running the notebook. Your report should be in the style of a conference paper, including an introduction, motivation, related work, etc.
Each part of the project must have experiments, if you failed in a particular experiment, explain why you think you failed and what you have done to improve from there. You are expected to use very elaborate explanations and use visual means to show your results (plots, graphs, etc.). Your grade would be greatly affected by the report you write so write the report thoroughly!

Everything in the report should be in your own words - all quotes must be clearly attributed.
We check for any cheating inside and outside of the class, deep and shallow - please don't put us in difficult situations.