Reinforcement Learning
Mid Semester Project - 2023

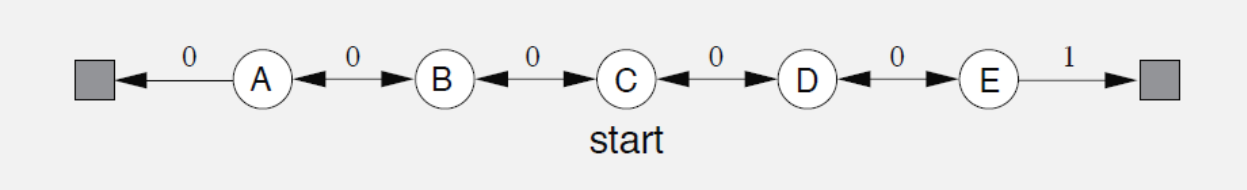
The main goal of this mid semester project is to summarize the main topics that we have discussed in the course using some practice and theory.

The project has two parts. The first part will contain theoretical questions and the second part will contain a practical aspect represented by a programming task.

Theoretical Questions

Your solutions must be written digitally (i.e written on a computer) handwritten solutions will not be accepted!
The submission file for this part will be called **theoretical.pdf**.

Question 1:



We discussed the Random walk example in class.
In this Markov Reward Process (MRP), all episodes start at the center state, C (state:3), then proceed either left or right by one state on each step, with equal probability.

Episodes terminate either on the extreme left (s0) or the extreme right (s6).
When an episode terminates on the right, a reward of +1 occurs, all other rewards are zero

The initial values are as follows:

$$v_0^{(0)} = 0 \quad v_1^{(0)} = \frac{1}{6} \quad v_2^{(0)} = \frac{2}{6} \quad v_3^{(0)} = \frac{3}{6} \quad v_4^{(0)} = \frac{4}{6} \quad v_5^{(0)} = \frac{4}{6} \quad v_6^{(0)} = 0$$

Using Dynamic Programming, run (theoretically) Policy Evaluation and show what will be the states values at next iteration. **Show your computations in details.**

Question 2:

Given the following grid-world problem. The goal is to reach the upper left or lower right corners.

0	-14	-20	-22
-14	-18	-20	-20
-20	-20	S2	-14
-22	S1	-14	0

Actions: UP, DOWN, RIGHT, LEFT

Rewards: -1 for every movement (even movements that tries to move into a wall and stays in place)

Transition Model: Deterministic

Discount factor: $\gamma = 1$

Random Policy: 0.25 probability in each direction

In this environment, we were partially given the "Values" for certain states (according to the current policy). Calculate (manually) the Values (according to the current policy) for states: S1, S2, **Show your computations in details.**

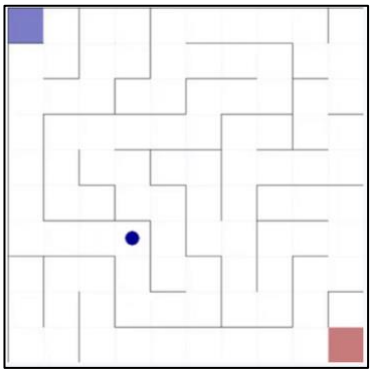
Programming Task

In this part of the assignment, you will find three programming exercises where each exercise refers to a different version of the environment.

Environment Definition

In this part you will solve several variations of the 2D maze environment.

2D maze is an environment where an agent (blue dot) finds its way from the top left corner (blue square) to the goal at the bottom right corner (red square). The objective is to find the shortest path from the start to the goal.



Action space:

The agent may only choose to go up, down, right, or left ("N", "S", "E", "W"). If the way is blocked, it will remain at the same location.

The environment you will work in is a stochastic environment, when the agent takes a certain action there is a probability of 0.9 that the environment will allow him to reach the state he intended to reach. For example, if our agent wants to turn right, there is a probability of 0.9 that he will actually turn right but there is also a probability of 0.1 that he will move up, down or left (with equal probability for each of the three directions) (**the "original" environment in the attached notebook is deterministic, you must add in your implementation as for the stochastic part described above**).

Observation space:

The observation space is the (x, y) coordinate of the agent. The top left cell is (0, 0).

Reward:

As default: a reward of 1 is given when the agent reaches the goal.

For every step in the maze, the agent receives a reward of $\frac{0.1}{number\ of\ cells}$.

This is a default choice of the environment. during your experiments, you are expected to try different rewards.

End condition:
The maze reset when the agent reaches the goal.

Maze Versions:
The maze has three versions:

- 5 cells x 5 cells: maze2d_5x5.npy
- 15 cells x 15 cells: maze2d_15x15.npy
- 25 cells x 25 cells: maze2d_25x25.npy

You will find demonstration of how to use/render the game (random actions) in the template notebook:

https://colab.research.google.com/drive/1JeQr8SAVD8ULUYA_3qwz3Yrn9zVdeBve?usp=sharing

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.

Explore the environment carefully and make sure you understand the observation-space, the action-space, and how the rewards are defined. Also, make sure you understand what's the difference between the maze versions.

Click here to see an example of environment customization and a stochastic environment implementation

Exercise 1:

Solve the 5x5 maze environment using dynamic programming.
Use the Policy iteration algorithm

Exercise 2:

Solve the 15x15 maze environment using three different algorithms:

- Monte Carlo
- Q-Learning
- SARSA

The goal is to run different Polices and of course solve in **as few steps as possible**.
Show and compare different variations auc as different initializations, different learning rates, different exploration methods and more.

Exercise 3:

Solve the 25x25 maze environment.
This task is a little more difficult, so here you will use a tool that will help you create the "agent encouragement" and set two Reward cells in strategic locations in the environment. The first reward will be a "positive reward" and the second reward will be a "negative reward" (punishment). The location of the cells and the value of the Rewards is at your discretion.
Note: **The reward cell should be permanent**. Upon selection the cells should not change/update and you will not be able to replace them every Episode/Step. For the benefit of solving the environment, you can choose any algorithm from what you have learned during the lectures (Deep Reinforcement Learning algorithms are forbidden).
The goal is to solve the environment in **as few steps as possible**.

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

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

Guidelines:

1. You are **not allowed to use an existing RL libraries**. Write the algorithm yourself. Remember: We want to see your understanding of RL and not your ability to use libraries and modules.

2. You are **not allowed to use a deep reinforcement** solutions or RL algorithms that you haven't learnt in the course.

3. **Experiments (Must)** - 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.

4. After working and analyzing with the learnt exploration-exploitation methods, you can consider using advanced methods from an external source.

5. 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 steps/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 completed and converged, (for example, if it took you 40 steps/episodes to converge, show how the agent tries to solve the environment after 20 steps/episodes in the second video,

show how the agent tries to solve the environment after 40 steps/episodes).
Even if you did not solve an environment - supply graphs describing the average rewards on 100 steps/episodes (after finishing the training phase).
Note: In the second exercise there are three algorithms, What is written above applies to each one of them.
Note: **Do not run more than 500 steps/episode iterations (even if it was not finished)**.

6. All training outputs will be displayed in the notebook.

7. Submit a final report in pdf format when your ID numbers are included in the file name.

8. You need to supply your code as Google Colab notebooks. Such that the course team can run it.

9. **Write a clean code**! Separate code cells and make extensive use of text cells. A sloppy notebook will result in a lower grade.

10. Do not change anything in notebook after the final submission date. A notebook that has been ran/changed after the submission date will be **automatically disqualified**.

Due Date:
Final Project submission is due **January 5th at 23:59**.

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 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!

The report will be written using an online collaborative LaTeX editor called overleaf (link below).

An example: https://www.overleaf.com/read/jkstfjsxrfnd

Everything in the report should be in your own words - all quotes must be clearly attributed.

The report will be submitted to the submission box. The file name of the report will contained your IDs as following: **report_ID1_ID2.pdf**

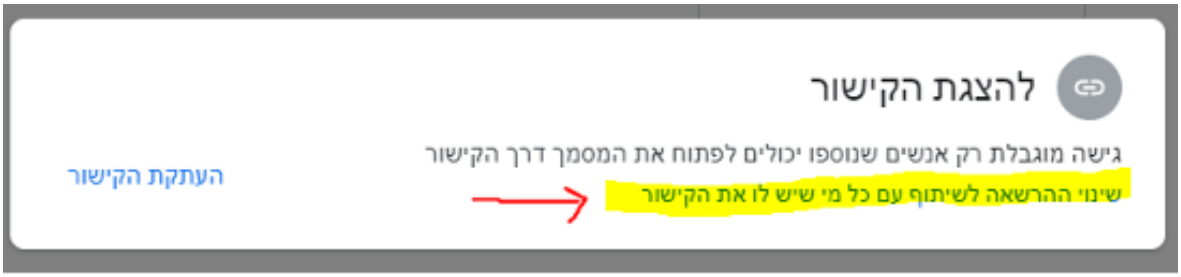
Be very clear about what code you've used from other sources, if any. Clear citations are essential. Failure to credit ideas and code from external sources is cheating and will result in immediate disqualification.
Make sure you evaluate both the good and bad points of your approach.
Even if you didn't accomplish your goal, evaluate what you did.

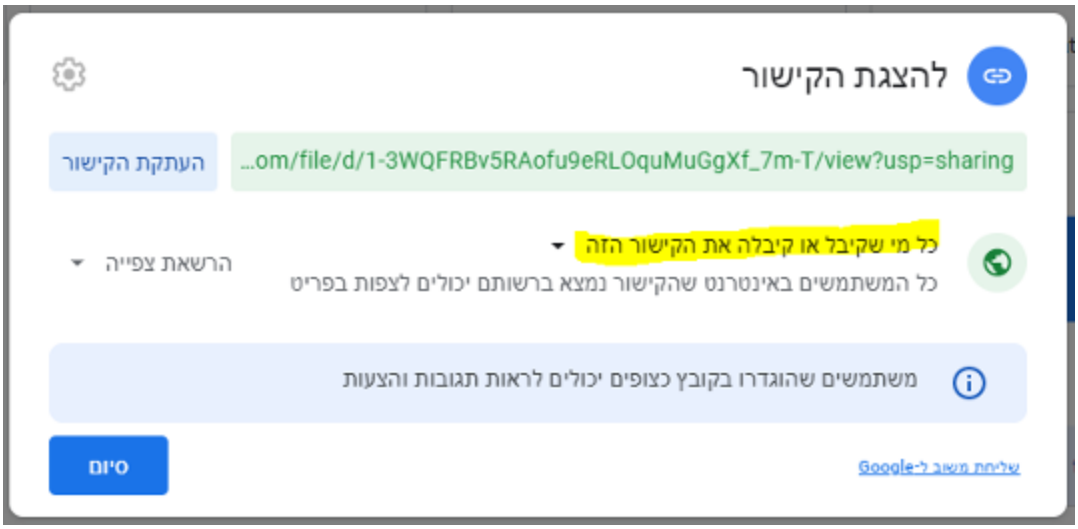
Do not forget to include the project title, your name and ID in this file.
Max number of pages: 10 (but you don’t have to use them all !!).

The notebook will be presented in an orderly and clean manner, it will contain separate code cells and text cells that explain the actions performed.
** **Very important** ** - when submitting, the notebooks will contain **all the outputs** relevant to the training results.

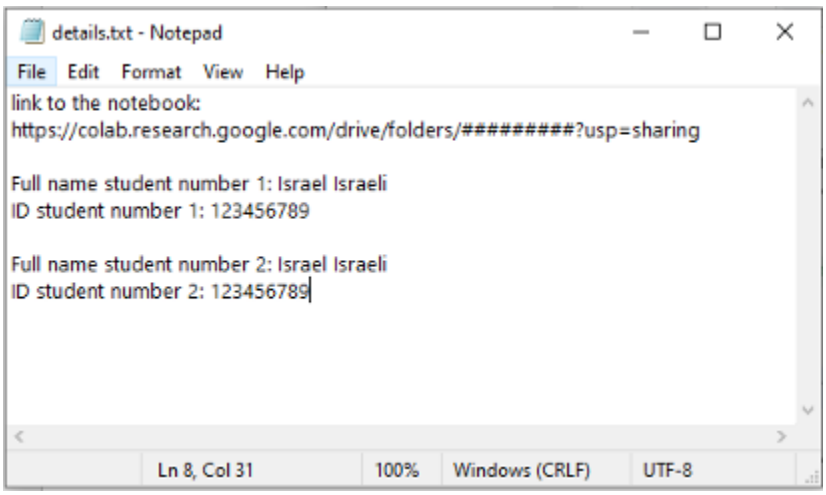
To the submission box you will also submit an **explainer.md** file that contains instructions and explanations on how to operate your notebook and other relevant details that need to be known to those who want to use your notebook.

You will share the notebook from your "Google Drive" account, it can be shared with anyone who holds the link as follows:





Submission will be done in pairs when only one of the partners submits the assignment to the submission box.
Enter your details in the text file named **submit.txt**, the address of your notebook, the names and ID of the two partners, as follows:



To sum up, in the submission box, submit the following files:

1. submit.txt
2. theoretical.pdf
3. report_ID1_ID2.pdf
4. explainer.md

Team size
The project will be performed in groups of 2 students.

Academic Integrity

Team/Student may not copy code from other teams/students. Copying answers or code from other students for a project is a violation of the university's honor code and will be treated as such. All suspicious activity will be reported to the head of the department and the university authorities.

Giving code to another student is also considered a violation. Students are responsible for protecting their own work from copying.

If you build some of your code on existing work and utilize existing code (your own or code found on the web), you must give proper attribution to all existing work that you used and make it clear what you changed and contributed. Any unattributed or uncited work that you use will be considered a breach of academic honesty and dealt with according to the course policy in the syllabus.

In any matter related to the project, even administrative, you can contact Aviv German at:
aviv.german@post.runi.ac.il