Website: IvLabs
Follow Us: Facebook, Instagram, YouTube

TechnoSeason 2023

TechnoSeason is the annual recruitment drive for prospective Summer Interns of IvLabs, VNIT, Nagpur.

General Instructions:

- This task consists of two topics: Python and OpenCV.
- The points you'll get will be judged based on your implementation, the logic used, etc.
- While you may take help from the internet for the errors you are facing or understand any algorithm, directly copying solutions is strictly prohibited.
- Few tasks have some bonus points. This will be awarded if someone finds a unique way to solve the problem or does something out of the box.
- A partially completed task is also acceptable. Points will be awarded according to the extent of completion.
- After completion of tasks, create a ZIP folder with all the files and upload them on this google form TechnoSeason Submission.
- The last date of submission is 25 March before 11:59 pm. The final decision lies with the event coordinators only and is not subject to change.

Python Task

General Instructions:

- This part consists of 2 questions along with one Bonus part (included in 2nd question).
- You can submit either .ipynb files (Jupyter Notebooks, Google Collab file not the link) or python .py files (in a ZIP folder). You are free to use multiple files per project.
- The questions are leveled from easy to moderate and have respective points.
- Do not hardcode the inputs, make it user-driven.
- Add comments to your code so that anyone can understand code.

1. Cipher

Difficulty Level: Moderate

Points: 30

Problem Statement:

In a war between Elves and Orks, the Elves used a Cipher machine to encipher their messages but the Orks managed to break the code multiple times, giving them access to secret information about the Elves' battle plans. The Orks were careful not to get caught and took extreme precautions to hide their knowledge as with every win for Orks, Elves got more and more suspicious that their code was broken. Now, as a war analyst, your job is to help the Orks win as many battles as possible while keeping the suspicion of the Elves (S_e), below a certain threshold value (S_t=0.6), since more suspicion of the Elves that their code has been broken, they will reset their communication. The Orks can choose to win or lose any battle they want, and some battles are more important than others, as denoted by their importance factors (F_i) measured on a scale of 0 to 10.

S_e is calculated as an average of the results of all previous battles using the following method:

- If the Orks win we add their probability of losing to the average, and if they lose we subtract the probability of their winning from the average.
- If the S_e value is negative at any instant then consider it as zero and then continue the process.

Consider the following scenario as an example -

Battle No.	Probability of Orks Win	F_i	Decision	S_e
1	0.6	6	Win	(0.4)/1 = 0.4
2	0.2	8	Win	(0.4+0.8)/2 = 0.6

Battle No.	Probability of Orks Win	F_i	Decision	S_e
1	0.6	6	Win	(0.4)/1 = 0.4
2	0.2	8	Lose	(0.4-0.2)/2 = 0.1

Battle No.	**Probability of Orks Win**	$\mathbf{F_i}$	**Decision**	$\mathbf{S_e}$
1	0.6	6	Lose	$(0-0.6)/1 = -0.6 \leq 0$
2	0.2	8	Win	$(0+0.8)/2 = 0.4$

In this scenario, Orks should prefer to lose the battle having smaller F_i (1st battle) to maximize their importance factor (F_i). Winning both battles will make $S_e = 0.6$ and Elves will get suspicious of the Cipher code being broken and will reset their communication.

- The number of battles will be given as input by the user. The values of the importance factor (F_i) and the probability of Orks winning will also be given by the user for each battle.
- You have to find the best strategy (deciding which battle to lose and which to win) such that you get the maximum sum of importance factors at the end of the war while keeping S_e below the threshold value.
- You have to print the result for each battle (Win/Lose) and the sum of importance factors at the end.

2. Maze Runner

Difficulty Level: Moderate

Points: 30

Bonus: 5

Problem Statement:

You have to implement a single-player Maze solver Game in NxM board using python. The player starts from the top left corner (1,1) and has to reach the bottom right corner(N, M).

Grid size is taken as input from the player. Some squares of the grid are blocked whereas some contain rewards. It can move to adjacent squares given it is not blocked. The player cannot move diagonally.

Generate your own maze using the Random function.

For example:
" # " denotes an obstacle.
" * " denotes an empty square.
" @ " denotes a reward.

Your task is to generate the board and then reach the destination in the least possible moves. For every action, print the direction you moved in. (Right, Left, Up, Down)

Bonus- You have to collect all the possible rewards before reaching the destination.