Q2
a. Is entering via the light blue boundary a common strategy used by Team2 on the T (terrorist) side?

To solve this problem, I have implemented a function that checks whether a given point is inside a boundary or not. The function takes the inner boundary and the coordinates of the exit edge as inputs. It then creates a new outer boundary by extending the sides of the inner boundary and including an additional space of 30 units at the exit edge.

The main logic involves tracking the path of each player throughout the round and checking the location of each player against the inner and outer boundaries. We also keep track of the previous location's status (inside or outside both boundaries) to determine the player's movement.

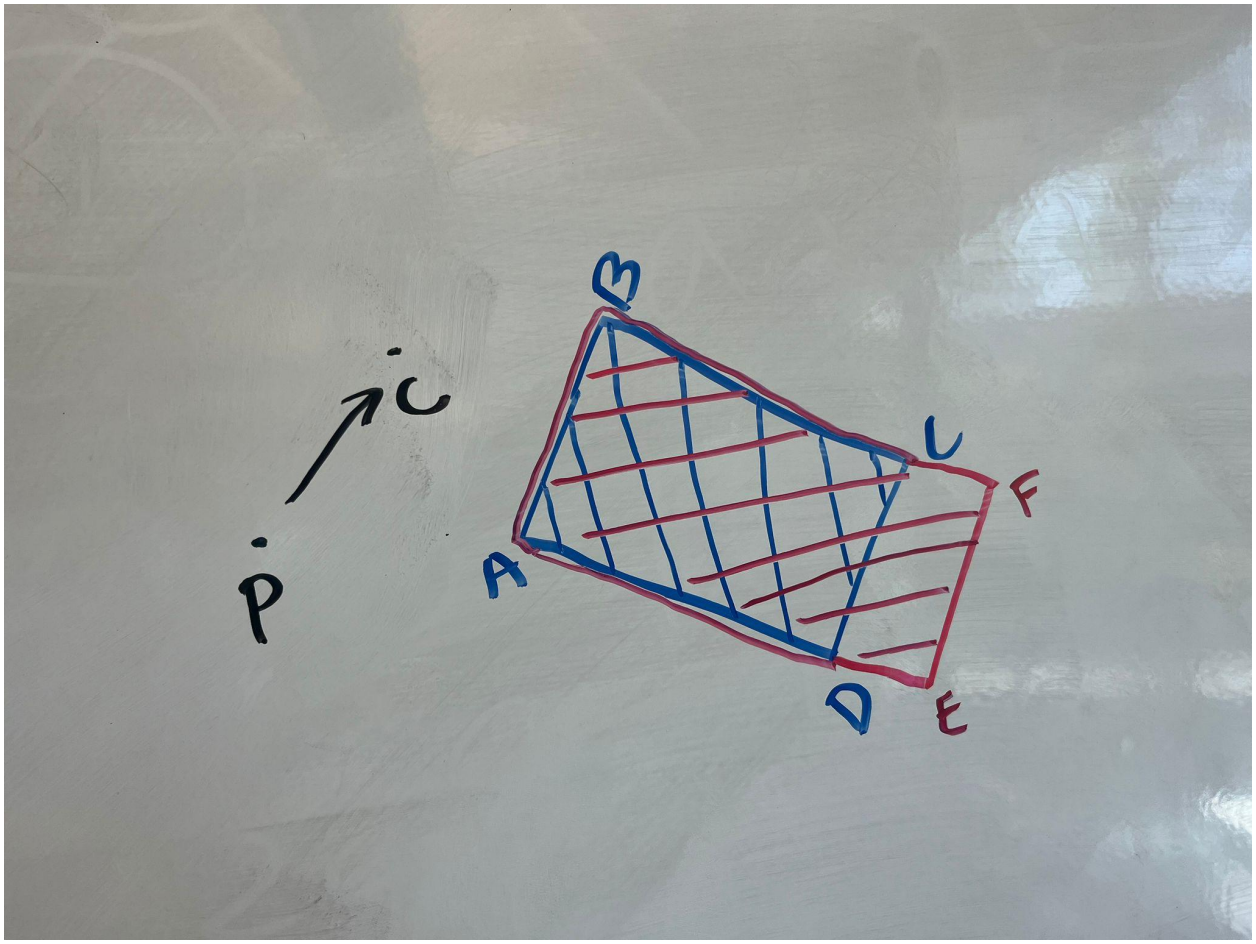
There are five possible scenarios:

For all Scenarios below:

Inner Boundary - A B C D
Exit Edge - C D
Outer Boundary - A B C F E D
P - Previous Point (X1, Y1)
C - Current Point (X2, Y2)

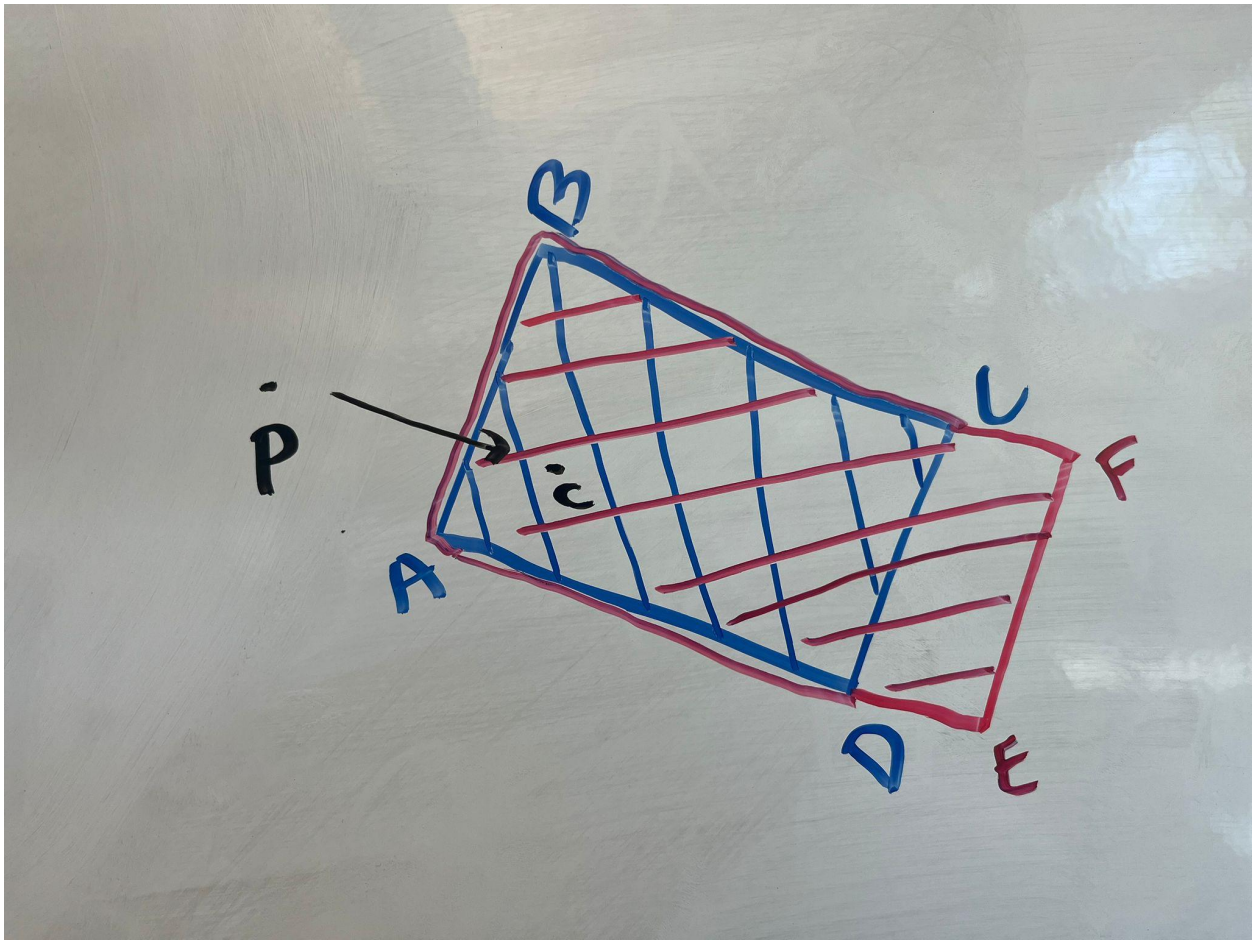
Scenario 1:

Previous location: (X1, Y1)
    Inside Inner Boundary: False
    Inside Outer Boundary: False
Current location: (X2, Y2)
    Inside Inner Boundary: False
    Inside Outer Boundary: False
This scenario is not relevant to our analysis, so we can skip checking it.



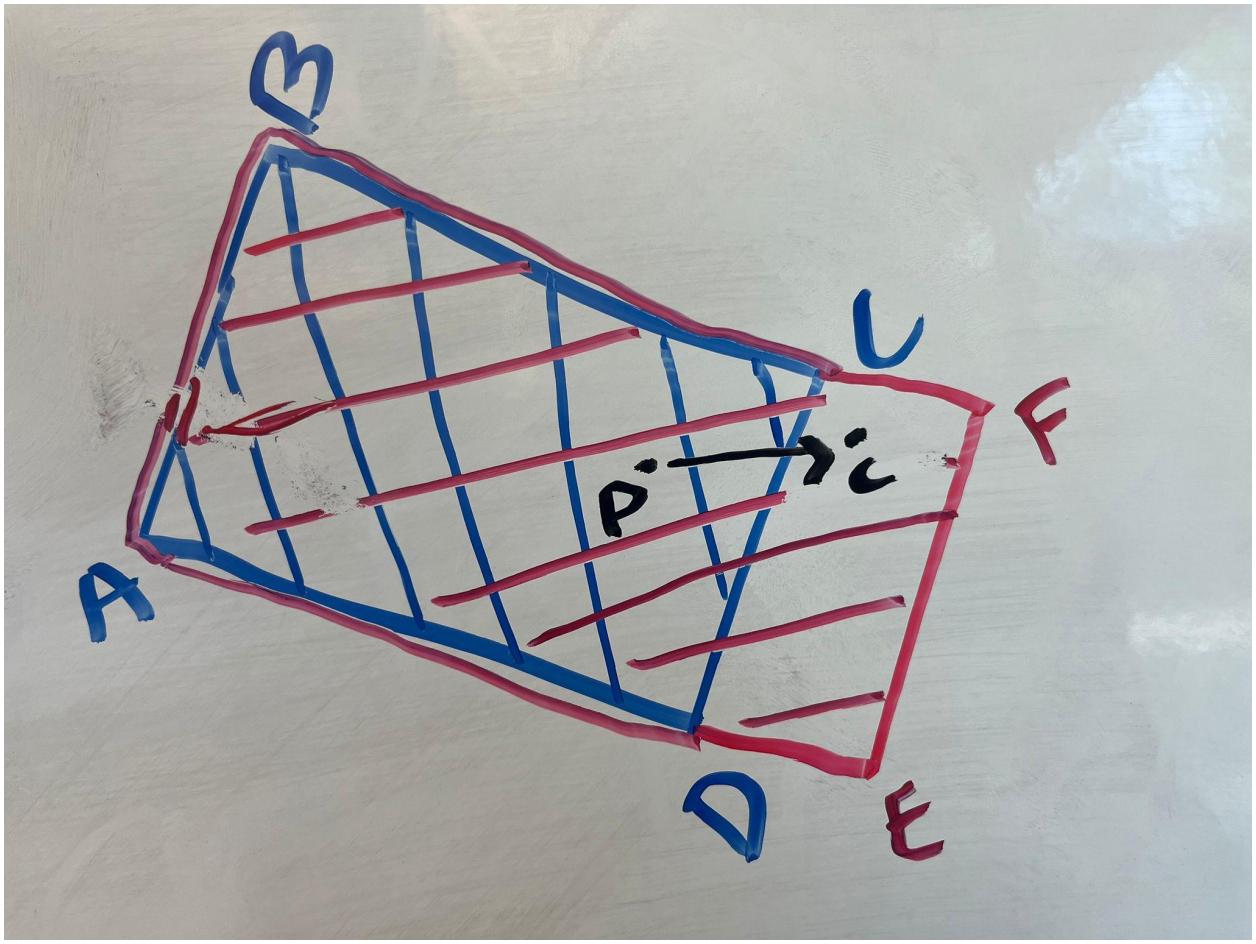
Scenario 2:

Previous location: (X1, Y1)
    Inside Inner Boundary: False
    Inside Outer Boundary: False
Current location: (X2, Y2)
    Inside Inner Boundary: True
    Inside Outer Boundary: True
This scenario indicates that the player is entering the inner boundary from the desired direction.



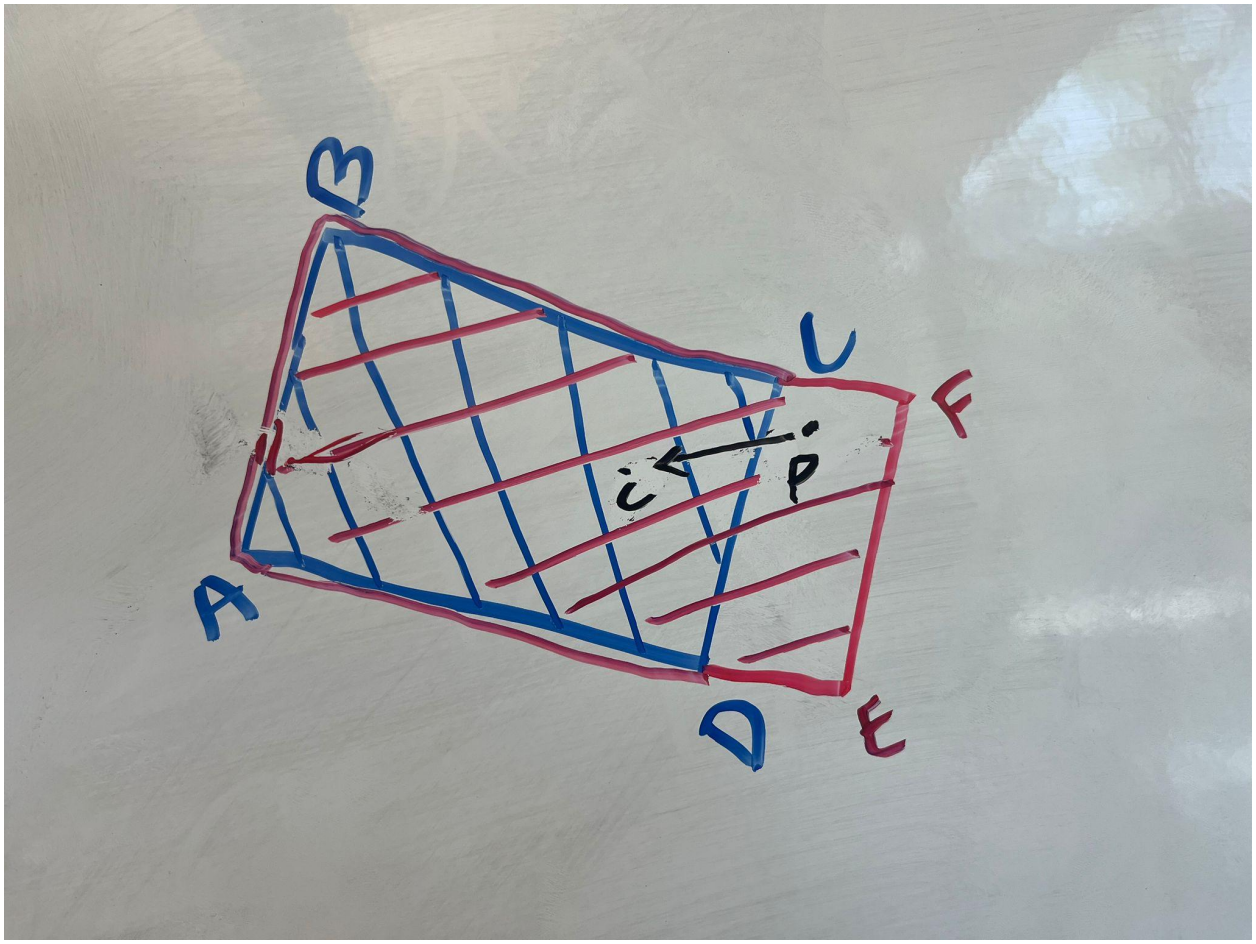
Scenario 3:

Previous location: (X1, Y1)
    Inside Inner Boundary: True
    Inside Outer Boundary: True
Current location: (X2, Y2)
    Inside Inner Boundary: False
    Inside Outer Boundary: True
This scenario signifies that the player is exiting the boundary through the desired exit edge.



Scenario 4:

Previous location: (X1, Y1)
    Inside Inner Boundary: False
    Inside Outer Boundary: True
Current location: (X2, Y2)
    Inside Inner Boundary: True
    Inside Outer Boundary: True
This scenario suggests that the player is using the tunnel to exit the bomb site instead of entering.

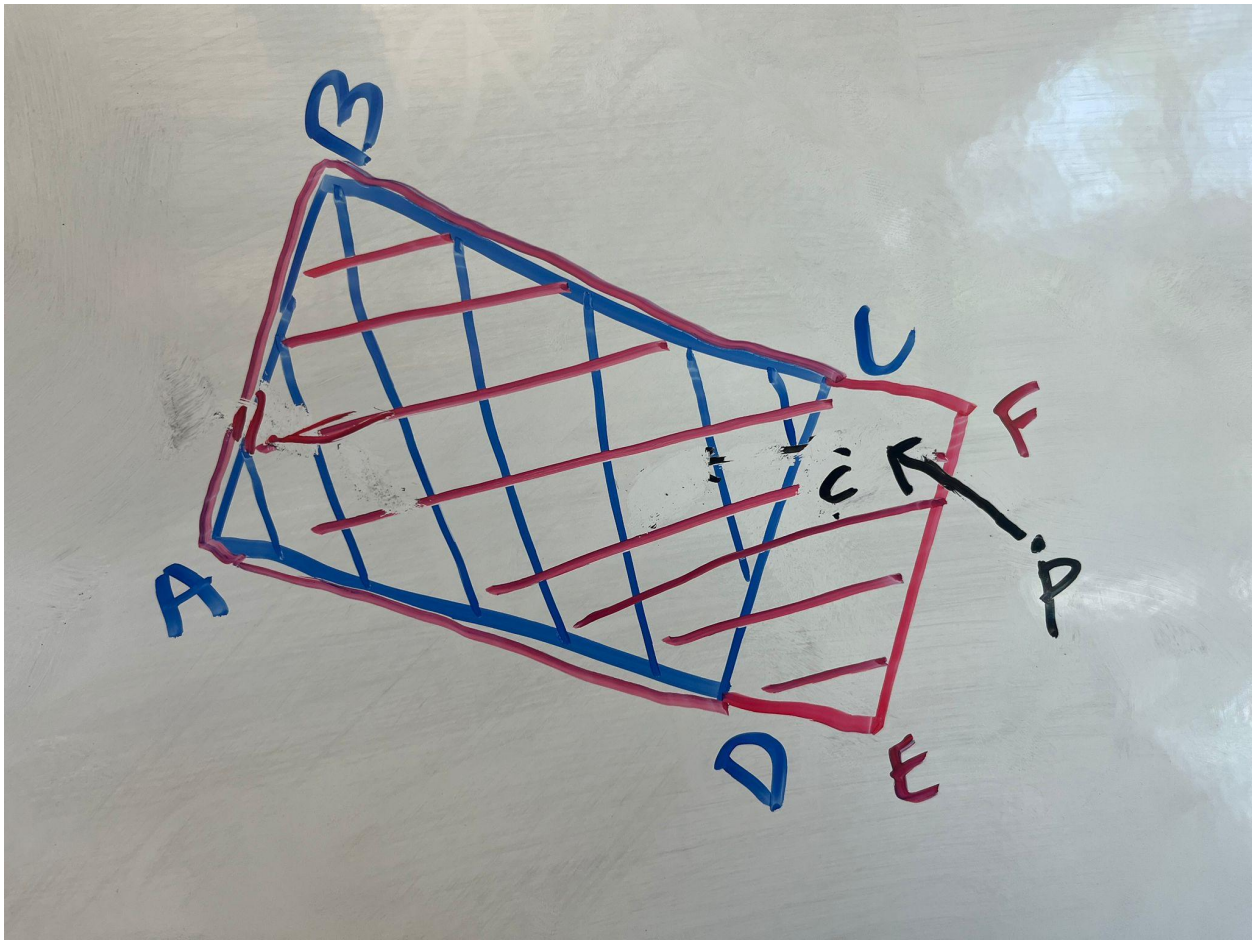


Scenario 5:

Previous location: (X1, Y1)
- Inside Inner Boundary: False
- Inside Outer Boundary: False

Current location: (X2, Y2)
- Inside Inner Boundary: False
- Inside Outer Boundary: True

This scenario also indicates that the player is using the tunnel to exit the bomb site instead of entering.



These scenarios help us determine whether the players have used the desired strategy or not.

Thus For every timestamp of the player, I am checking for Scenario 3 to find out if they used this strategy or not.

b. What is the average timer that Team2 on T (terrorist) side enters "BombsiteB" with at least 2 rifles or SMGs?

1. **Grouping the Dataset:** I began by grouping the dataset by rounds, which allowed me to organize the data effectively for further analysis.
2. **Sorting and Filtering**: Next, I sorted the dataset in increasing order based on the 'tick' value and filtered it to include only instances where Team2 was playing on the T side.
3. **Focusing on "BombsiteB" Entries**: To narrow down the analysis, I applied a filter to the dataset based on the "area_name" column, selecting only the rows corresponding to entries into "BombsiteB."
4. **Identifying First Entries**: Within this filtered dataset, I identified the first occurrence of each player's movement. This step enabled me to determine the timing of their initial entry into "BombsiteB" for each round.

5. **Extracting Weapon Classes**: To determine the weapon classes possessed by the players at the time of entry, I made use of the "extract_weapon_classes" function. This function allowed me to count the number of rifles and SMGs held by the players upon entering "BombsiteB."
6. **Timing Calculation**: For each entry where Team2 had at least two rifles or SMGs upon entering "BombsiteB," I recorded the clock timer values.
7. **Average Calculation**: Finally, I calculated the average timer by taking the mean of all the recorded clock timer values.

c. Now that we've gathered data on Team2 T side, let's examine their CT (counter-terrorist) Side. Using the same data set, tell our coaching staff where you suspect them to be waiting inside "BombsiteB"

1. **Grouping the Dataset:** Initially, I grouped the dataset by rounds to facilitate further analysis and organization of the data.
2. **Sorting and Filtering**: To focus specifically on Team2 playing on the CT side, I sorted the dataset in ascending order based on the timestamp and filtered it to include only instances where Team2 was playing on the CT side.
3. **Narrowing Down to "BombsiteB" Entries**: I further narrowed down the analysis by filtering the dataset based on the "area_name" column, selecting only the rows corresponding to entries into "BombsiteB."
4. **Analyzing the Data Player-Wise**: Within this filtered dataset, I grouped the data by players, allowing me to analyze the movements and waiting times of each individual player.
5. **Grouping Data by Coordinates:** I then grouped the dataset by coordinates on the x-y plane. This step involved examining each x-y coordinate and determining the maximum waiting time spent by the player at that specific point.
6. **Creating a New DataFrame:** Using the collected data, I created a new DataFrame with columns representing the x and y coordinates, along with the corresponding waiting time.
7. **Heat Map Creation**: Finally, I utilized the newly created DataFrame to generate a heat map. The heat map visually represents the positions within "BombsiteB" where Team2 players spent the highest amount of waiting time while being alive.

3. (No Coding) Most of the time, our stakeholders (in this case, the CS:GO coaching staff) aren't tech-savvy enough to run code themselves. Propose a solution to your product manager that:
a. could allow our coaching staff to request or acquire the output themselves
b. takes less than 1 weeks worth of work to implement

Some Solutions for this -

- We can create a web-app that will allow the coaching staff to upload the data for analysis and provide the output.
- Similar to Q2a, we can create an interactive map on the frontend where the staff can select the boundary and the exit edge and then the software will process the data and show the result on the frontend.
- We can also similarly on this map select the boundary and we will be able to find hiding spots inside on a single button click by generating a heatmap for that area.

Some Additional details -

- The web app would be hosted on a cloud server, so that the coaching staff could access it from anywhere
- We should maintain a database as a cache so that we can display results faster in case same inputs are used again.
- The web app could be developed in less than a week using a standard web development framework, such as Django or Flask.

Highlights of my solution -

- I have made use of local cache to reduce the computation. I could have used a database for cache but it would make the setup more complex to judge this assignment.
- I have created API end points in Flask for easy access to all function calls and their results. More details of each endpoint are given on the home page of the server.

To run the project,

- Install the dependencies first by doing pip install requirements.txt
- Just run the python file python3 main.py.
- Go to url http://127.0.0.1:8000/ where you can see a short description of all the end points and how to access them