MDP Formulation

Objective

The objective of the problem is to maximise the profit earned over the long-term.

Decision Epochs

The decisions are made at an hourly interval; thus, the decision epochs are discrete.

Assumptions

1. The taxis are electric cars. It can run for 30 days non-stop, i.e., 24*30 hrs. Then it needs to recharge itself. If the cab-driver is completing his trip at that time, he'll finish that trip and then stop for recharging. So, the terminal state is independent of the number of rides covered in a month, it is achieved as soon as the cab-driver crosses 24*30 hours.
2. There are only 5 locations in the city where the cab can operate.
3. All decisions are made at hourly intervals. We won't consider minutes and seconds for this project. So for example, the cab driver gets requests at 1:00 pm, then at 2:00 pm, then at 3:00 pm and so on. So, he can decide to pick among the requests only at these times. A request cannot come at (say) 2.30 pm.
4. The time taken to travel from one place to another is considered in integer hours (only) and is **dependent on the traffic**. Also, the traffic is dependent on the hour-of-the-day and the day-of-the-week.

State

The state space is defined by the driver's current location along with the time components (hour-of-the-day and the day-of-the-week). A state is defined by three variables:
$s = X_i T_j D_k \; where \; i = 0 \ldots m-1; j = 0 \ldots . t-1; k = 0 \ldots . . d-1$
Where X_i represents a driver's current location, T_j represents time component (more specifically hour of the day), D_k represents the day of the week

- Number of locations: m = 5
- Number of hours: t = 24
- Number of days: d = 7

Note: the model is not aware of the requests coming to it so the state must be amend to s = $X_i T_j D_k$Ri,j,k $where \; i = 0 \ldots m-1; j = 0 \ldots . t-1; k = 0 \ldots . . d-1$; Ri,j,k = {a ∀.a ∈ action space}.

A **terminal state** is achieved when the cab completes his 30 days, i.e., an episode is 30 days long or taking a wrong actions.

Actions

Every hour, ride requests come from customers in the form of **(pick-up, drop)** location. Based on the current 'state', he needs to take an action that could maximise his monthly revenue. If some passenger is already on-board, then the driver won't get any requests. Therefore, an action is represented by the tuple (pick-up, drop) location. In a general scenario, the number of requests the cab-driver can get at any state is not the same. We can model the number of requests as follows:
The number of requests (possible actions) at a state is dependent on the location. Say, at location A, you get 2 requests on average and at location B, you get 12 requests on average. That means, when at A, the cab-driver is likely to get 2 customer requests from anywhere to

anywhere of the form (p, q). For each location, you can sample the number of requests from a Poisson distribution using the mean λ defined for each location as below:

Location	**λ (**of Poisson Distribution)
Location A	2
Location B	12
Location C	4
Location D	7
Location E	8

The upper limit on these customers' requests (p, q) is 15.
Apart from these requests, the driver always has the option to go 'offline' (accept no ride). The no-ride action just moves the time component by 1 hour. So, you need to append (0,0) action to the customer requests.
There'll never be requests of the sort where pickup and drop locations are the same. So, the action space A will be: $(m-1) * m + 1$ for *m* locations. Each action will be a tuple of size 2. You can define action space as below:

- pick up and drop locations (p, q) where *p* and *q* both take a value between 1 and m;
- $(0, 0)$ tuple that represents 'no-ride' action.

For example, if the set of all possible locations is of size 3: {A, B, C}. So, at state (A, 6:00 pm, Wednesday), his possible actions would be of the form: (i, j) where *i* and *j* can be any location from {A, B, C}, but i ≠ j. The following table shows all possible actions the driver can take.

(A, B)	(C, A)
(A, C)	(C, B)
(B, C)	(B, A)
(0, 0)	

The average number of requests received when the driver is in location A is 2. So, the cab driver will get any two random requests from the above table, say (A, B) and (C, A). Also, he'll always have the option of 'no-ride'. So, his possible actions at that state would be: $(A, B), (C, A), (0, 0)$
His action should be to pick the best request among these three.

State Transition

Given the current state $s = X_i T_j D_k$, the next state s' will be as following:
s' = $XqTt'Dd'$ a = (p, q)
$XpTt'Dd'$ a = (0,0)
Note: If the model takes a wrong actions there's no transitions.
Where t', d' represents the time and day respectively after taking an action.
You can calculate the total time taken to reach from one point to other from the Time Matrix (calculated basis the historical data) provided to you in the zip file. You don't need to learn a distribution of the time taken; all possible values of (pick up, drop, t, d) are available in the 'TM.npy' file, you simply need to look them up.

Time Matrix is a 4-D matrix. The 4 dimensions are as below:
• Start location
• End location

• Time-of-the-day
• Day-of-the-week

Python indices for these dimensions are as:
$Time - matrix[start - loc][end - loc][hour - of - the - day]\ [day - of - the - week]$

This matrix has been calculated considering the distance between two locations and traffic conditions, which generally depends on the hour-of-the-day and the-day-of-the-week. To make the problem manageable, we divided the 24-hour frame into 4 segments:
from 12:00 am to 6:00 am, 6:00 am to 12:00 pm, 12:00 pm to 6:00 pm and 6:00 pm to 12:00 am.

You are given this time-matrix in the zip file shared. (Please run it once to understand its dimensions).

Reward

Your objective is to maximize the profit of a driver. Let C_f be the amount of battery consumed per hour and R_k be the revenue he obtains from the customer for every hour of the ride. The values of these parameters are defined in the skeleton code provided to you. Also, we have assumed that both the cost and the revenue are purely functions of time, i.e. for every hour of driving, the cost (of battery and other costs) and the revenue (from the customer) is the same - irrespective of the traffic conditions, speed of the car etc.
So, the reward function will be (revenue earned from pickup point p to drop point q) - (Cost of battery used in moving from pickup point p to drop point q) - (Cost of battery used in moving from current point i to pick-up point p). Mathematically,

$$R(s = X_i T_j D_k) = \begin{cases} R_k * (Time(p,q)) - C_f * (Time(p,q) + Time(i,p)) & a = (p,q) \\ -C_f & a = (0,0) \\ -10000 \text{ if the action takes the wrong action note: not added to the total reward metric.} & \end{cases}$$