Lab 13: Modeling for Decision Making

By: *Prof. N. Hemachandra & R. Deval* Date 01-Nov-2023

Objective: In this lab, students will try to propose a solution to an open-ended problem based on their respective understanding of the given scenario.

Instructions: Below are some instructions. Please go through them carefully:

- Given below is a scenario where students need to propose a solution-based
- Use the traditional approach to name your files for submission:
  - **ROLLNUMBER_IE507_Lab13.ipynb;**
  - **ROLLNUMBER_IE507_Lab13_Report.pdf**

Before describing the scenario, we need to detour into graphs and networks. In this lab, you must apply your decision-making skills and various tools from OR and logical reasoning.
Note: This lab is a part of modeling an open-ended problem based on your understanding. The given scenario information is sufficient to model the problem and can be incomplete enough from the given information point-of-view. Make appropriate assumptions in case need.

Graphs and Networks: A Brief detour

In context of algorithms, *graphs* **G** (given below) consists of a *set of vertices* **V** together with *a set* **E** *of edges*. These graph have many types of applications which one can observe in real life like:

- supply chain where the nodes can be the agents of the supply chain and edges can be the connections between the players.
- social media like Facebook, Instagram or WhatsApp, can be another example where the users are the nodes, and the interconnections between them are the edges
- world wide web (www) is a network of the web-pages where a web-page acts as a node, and the links on that web-page (to other web-pages) are the edges to the other nodes.
- similarly, other examples can be the road network in a city or a rail network of a nation can be typical examples of the graphs.

Example: Friendship Graph

Consider the graph in Fig. 1 where the vertices are people, and there is an edge between two people if and only if they are friends.

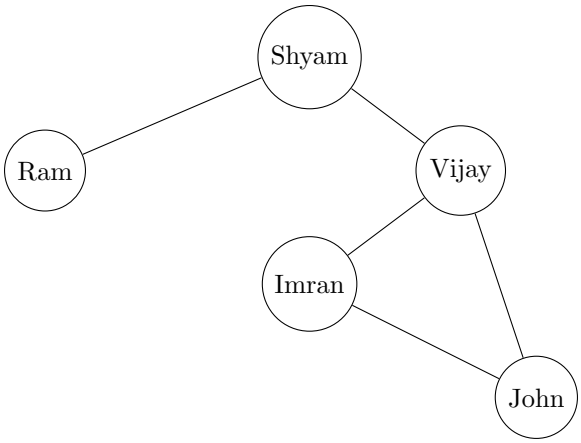


Figure 1: Friendship Graph

Let us formally define a graph in mathematical terms,

"*A graph* $\mathbf{G}$ *is a triple consisting of a vertex set* $\mathbf{V(G)}$*, and an edge set* $\mathbf{E(G)}$*, and a relation that associates with each edge two vertices (not necessarily distinct) called its endpoints*"

Graphs are essential because any binary relation is a graph, so a graph can represent essentially any relationship.

Example

Let us consider the graph $\mathbf{G_1} := (\mathbf{V_1}, \mathbf{E_1})$ and $\mathbf{G_2} := (\mathbf{V_2}, \mathbf{E_2})$ which are given below: From the above Fig.2,

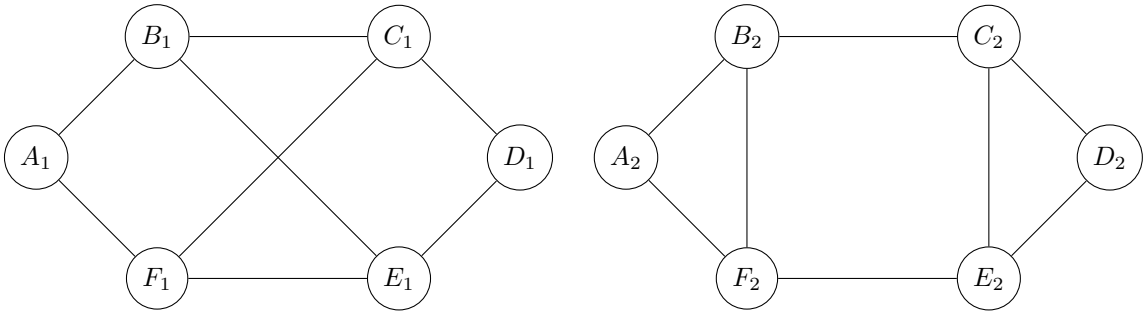


Figure 2: Left $\mathbf{G_1}$ & Right $\mathbf{G_2}$, two different graphs of the same numbers of nodes and edges

it can be seen that a cyclic path is formulated in $\mathbf{G_2(V_2, E_2)}$ where the vertices in set $\mathbf{S}_2^1 = \{A_2, B_2, F_2\}$ and $\mathbf{S}_2^2 = \{C_2, D_2, E_2\}$ formulate a triangle i.e., each node in $\mathbf{S}_2^1$ and $\mathbf{S}_2^2$ are connected to each node in the set itself. Thus, it is called a three-order-cyclic (refer Fig.3 below).

Notes:

1. A graph or *undirected graph* $\mathbf{G} := (\mathbf{V}, \mathbf{E})$ is an ordered pair of $\mathbf{V}$ vertices/nodes/points and $\mathbf{E}$ edges/arcs/links, where we define the set of elements as the vertices and the set of ordered pairs joining $[\mathbf{i}, \mathbf{j}]$ are called edges.
2. A *directed graph* or *digraph* $\mathbf{G}$ is an ordered pair $\mathbf{G} := (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is a set whose elements are called vertices or nodes, and $\mathbf{A}$ is a set of ordered pairs of vertices of the form $(\mathbf{i}, \mathbf{j})$, called arcs/edges.

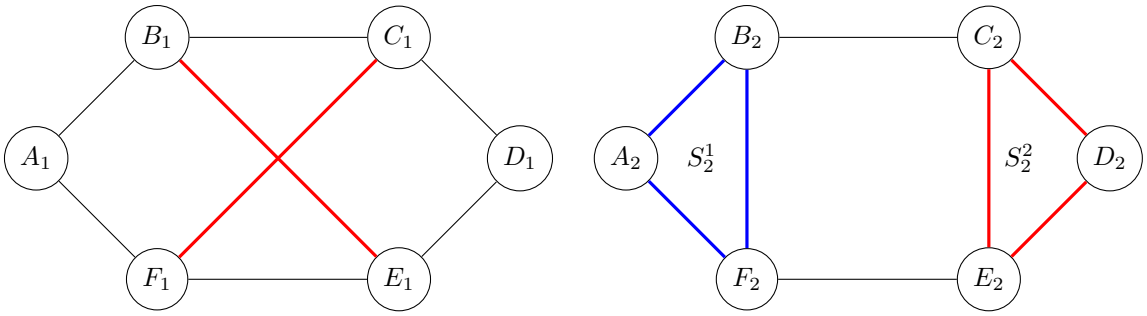
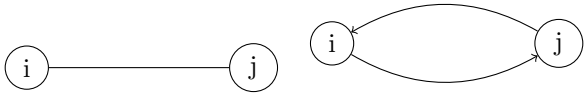


Figure 3: Graph $\mathbf{G_2}$ on the right hand is an cyclic graph of order 3 and on left $\mathbf{G_1}$ is a graph which is not a cyclic graph

In an edge $(\mathbf{i}, \mathbf{j})$ node $\mathbf{i}$ is called the tail of the edge and node $\mathbf{j}$ the head of the edge.

We can represent an undirected graph (left) and directed graph (right) by simply as follows



3. A *path* (in the case of the directed path) is an ordered list of vertices $(v_1, \dots v_k)$, so that $(v_i, v_{i+1}) \in \mathbf{E}$, $((v_i, v_{i+1}) \in \mathbf{A}) \forall i = 1, \dots\dots, k$. The length of a path is $|(v_1, \dots\dots v_k)| = k$. In simpler words, a path is a sequence of edges $e_1, e_2, \dots e_k$, such that $e_i \in \mathbf{E} \equiv \{u, v\}$. In the case of the directed graph, the sequence of $\{u, v\}$ matters as it denotes the direction but in the case of the undirected graph, the order does not matter for the path.
4. A *shortest path* of a graph is a path in $\mathbf{G} := (\mathbf{V}, \mathbf{E})$ to move from $v_i \in \mathbf{V}$ to $v_j \in \mathbf{V}$ using the minimum number of edges.
5. A *diameter of a graph* is defined as the longest shortest path and thus can be calculated as follows

$$\text{diameter of } \mathbf{G} = \max_{u,v \in \mathbf{V}} \{\min_{path} \quad \text{shortest path for } \{u, v\}\}$$

6. An (undirected) graph is said to be *connected* if, for every pair of nodes, there is an (undirected) path starting at one node and ending at the other node.
7. A *complete graph* is a graph whose each vertex is connected to each of the other vertices. The following is an example of a complete graph.

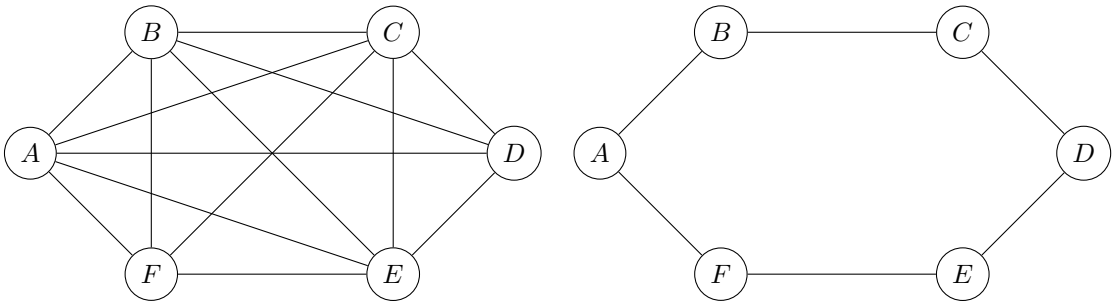
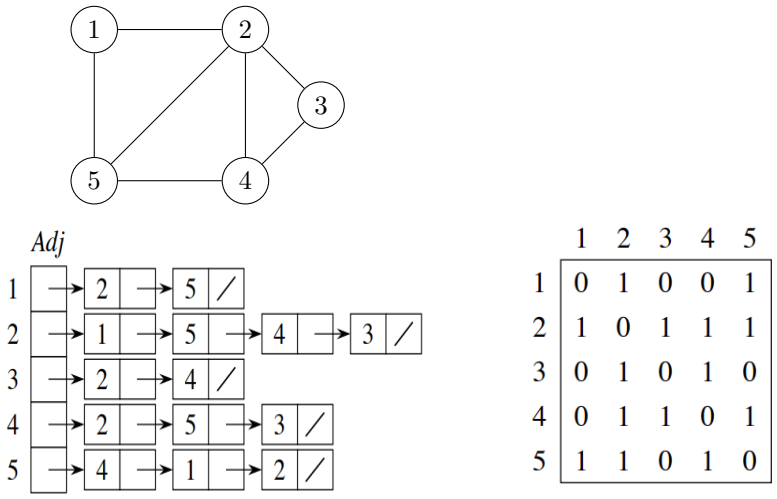


Figure 4: Graph on the left is a dense graph (and complete) & on the right is a sparse graph

So, we define a graph to dense graph (above left) where $|\mathbf{E}| \approx |\mathbf{V}|^2$ whereas a graph will be called a sparse graph (above right) if $|\mathbf{E}| \approx |\mathbf{V}|$.

8. Consider the following graph $\mathbf{G} := (\mathbf{V}, \mathbf{E})$



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Figure 5: On top, we have a graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$ and bottom-left we have an *adjacency list*, bottom-right we have *adjacency matrix* to store the graph

So, a graph can be stored as an adjacency list or a matrix. The adjacency matrix requires storage of $\mathcal{O}(|\mathbf{V}|^2)$, whereas; the adjacency list requires a $\mathcal{O}(|\mathbf{V}| + |\mathbf{E}|)$. So based on our need we prefer one over the other. In the adjacency matrix is just like a simple matrix $[\mathbf{A}]_{ij}$

9. In case of directed graph we have a adjacency matrix for the following graph as

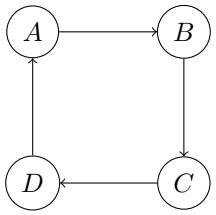


Figure 6: A the directed graph

	A	**B**	**C**	**D**
A	0	1	0	0
B	0	0	1	0
C	0	0	0	1
D	1	0	0	0

10. The *degree of a vertex* is the number of edges incident to the vertex. In a directed graph the *in-degree* of a node is the number of incoming arcs that have that node as a head. The *out-degree* of a node is the number of outgoing arcs from a node, that have that node as a tail.

11. A *cycle (directed cycle)* is an ordered list of vertices $v_0, \ldots, v_k$ so that $(v_i, v_{i+1}) \in E$, $((v_i, v_{i+1}) \in A) \forall i = 1, 2, \ldots, n$ and $v_0 = v_k$. The length of cycle is $|(v_0, ..., v_k)| = k$.

Question 1: Introduction to Graphs

1. Create an adjacency matrix for the given graph.

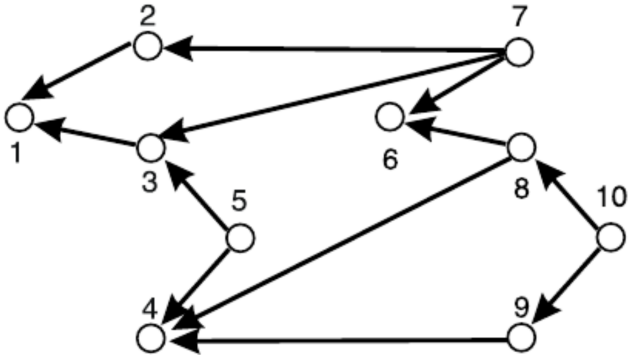


Figure 7: A Directed Graph

2. Create a graph from the adjacency matrix

   (a)

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

   Using the above matrix, create a hand-made graph (we can name nodes based on your convenience).

   (b) Given below is an adjacency matrix for a complex network; create a graph and try to interpret the graph.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0	1	1	0	0	0	0	0	0	1	1	0
2	1	0	0	1	0	1	0	0	0	0	0	1
3	1	0	0	0	1	0	0	0	0	1	0	1
4	0	1	0	0	0	1	1	1	0	0	0	0
5	0	0	1	0	0	1	0	0	0	0	1	1
6	0	1	0	1	1	0	0	0	0	1	0	0
7	0	0	0	1	0	0	0	1	1	0	1	0
8	0	0	0	1	0	0	1	0	0	1	1	1
9	0	0	0	0	0	0	1	0	0	0	1	0
10	1	0	1	0	0	1	0	1	0	0	0	0
11	1	0	0	0	1	0	1	1	1	0	0	0
12	0	1	1	0	1	0	0	1	0	0	0	0

Question 2: Efficient Tour Planning- A Heuristic Approach

Consider that **S1** and **S2** are two cities miles apart as hometown and workplace (are known as source nodes, **S**). A list of cities of attractions for your itinerary is given as a set **C** (representing the cities you plan to visit along with your parents). Further, these cities of attractions are distributed over a geographical region, and direct connectivity may/may not be possible from one tourist attraction to the other. Both **S1** and **S2** belong to metropolitan regions, and a direct route is possible from **S** to **C**. But you are constrained by choice to select a meeting point from a set **M**, which intersects on $\mathbf{S1} - \mathbf{S2}$ route. All cities in **C** are well spread and may be connected from **M** via. some distance and trains respectively. Below is a graph in Fig. 8 for the network connecting various cities in **M** and **C**. Note that a dashed-blue vertex is between the edges of elements of **C** with a meeting point in **M**. A black vertex represents the route link between a meeting point in set **M** and a source city from **S**. Below is a sample layout for a graph with two source nodes, one meeting points, and one city of attractions. Further, we provide you with some datasets to describe the characteristics of the below network.

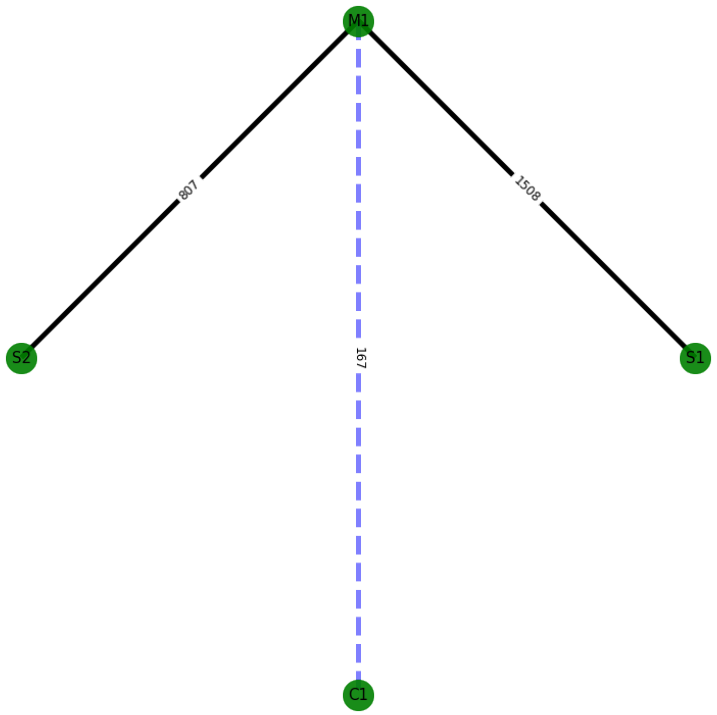


Figure 8: A layout for network with two source cities $(\mathbf{S1}, \mathbf{S2})$, one meeting point **M1** and one city of attraction **C1**

Data Set $(\forall \mathbf{i}, \mathbf{j} \in \mathbf{S} \cup \mathbf{C} \cup \mathbf{M})$

- $\mathbf{S} = \{\mathbf{S1}, \mathbf{S2}\}$ represents source nodes
- $\mathbf{C} = \{\mathbf{C1}, \mathbf{C2}, \cdots \mathbf{Cp}\}$ represents nodes for cities of attraction where $p = 1$
- $\mathbf{M} = \{\mathbf{M1}, \mathbf{M2}, \cdots \mathbf{Mq}\}$ represents nodes for possible meeting points where $q = 1$

- $\mathbf{N} = [\mathbf{n}]_{\mathbf{ij}}$ denotes the *adjacency matrix* representing the possible route between node i to node j.

	S1	S2	M1	C1
S1	0	0	1	0
S2	0	0	1	0
M1	1	1	0	1
C1	0	0	1	0

- $\mathbf{D} = [\mathbf{d}]_{\mathbf{ij}}$ denotes the *distance matrix* representing the distance (in *kms*) between node i to node j.

	S1	S2	M1	C1
S1	0	0	1508	0
S2	0	0	807	0
M1	1508	807	0	167
C1	0	0	167	0

- $\mathbf{F} = [\mathbf{f}]_{\mathbf{ij}}$ denotes the *frequency matrix* representing the frequency of trains between node i to node j.

	S1	S2	M1	C1
S1	0	0	8	0
S2	0	0	5	0
M1	8	5	0	6
C1	0	0	6	0

- $\mathbf{TT} = [\mathbf{tt_k}]_{\mathbf{ij}}$ denotes the *travel time matrix* representing the travel time for k^{th} train between node i to node j where $\mathbf{k} \in \mathbf{F_{ij}}$. Note that travel time is symmetric that is travel time for node j to node i is same as node i to node j.
  S1 to M1: [18:35, 21:27, 19:46, 26:25, 28:36, 20:28, 32:16, 22:10]
  S2 to M1: [16:25, 19:38, 20:56, 09:13, 12:08]
  M1 to C1: [03:42, 02:56, 04:15, 03:18, 05:08, 03:50]

- $\mathbf{DEP} = [\mathbf{dep_k}]_{\mathbf{ij}}$ denotes the *departure time matrix* representing the departure time for k^{th} train between node i to node j where $\mathbf{k} \in \mathbf{F_{ij}}$.
  S1 to M1: [16:30, 14:40, 03:55, 01:25, 10:26, 12:25, 06:35, 20:15]
  M1 to S1: [18:25, 06:20, 02:25, 05:15, 09:20, 15:15, 20:18, 22:20]

  S2 to M1: [04:20, 08:35, 12:15, 18:12, 22:18]
  M1 to S2: [12:25, 15:16, 18:21, 07:25, 21:20]

  M1 to C1: [05:50, 10:25, 12:30, 18:15, 21:05, 16:20]
  C1 to M1: [04:00, 07:15, 10:55, 18:35, 21:25, 23:50]

- $\mathbf{ARR} = [\mathbf{arr_k}]_{\mathbf{ij}}$ denotes the *arrival time matrix* representing the arrival time for k^{th} train between node i to node j where $\mathbf{k} \in \mathbf{F_{ij}}$.

NOTE: Model the scenario below using your programming skills so that your proposed solution works for large networks. For simplicity, assume that there is zero stoppage time at each node. Also, all the train timetable is in a 24-hour format.

Scenario A: Creating DataFrames

1. You are given above the travel and departure times for each type of train on each network link. Identify the Arrival Time for the above trains. Also, create a dataframe using the above data for Arrival and Departure with the *row as From-station* and the *column as To-station*.
2. Execute a code to create an Arrival-Departure Board for all the nodes based on the above-given data set.
   Note [1]**:** While creating dataframe for Arrival-Departure Board each train must be named for identification. For more details, refer to the footnote given below.

Scenario B: Scheduling Intersection

Note [2]**:** While providing an output of a feasible itinerary, you need to use an approach similar to the elementary set theory; for more details, refer to the footnote below.

1. A ***layover*** in our context significantly differs from the ***waiting time*** at the meeting point. Layover is when both parties come separately and wait for the next leg of the journey. At the same time, *waiting time* here refers to either of the parties reaching early at the meeting point and waiting for the other party. Create a plot for feasible trains arriving at the meeting point **M1** arriving from $\{\mathbf{S1}, \mathbf{S2}\}$ against the possible number of feasible departures for **C1** with a minimum layover time $\{1, 2, 5\}$ hours. Describe your obtained plot based on your understanding. Also, report your feasible travel plan using the prescribed notation.
2. Assume that your tour planning is now constrained by waiting time at meeting point **M1** say, waiting time not more $\{1.5, 2.5\}$ hours. You can continue to assume a minimum layover time of $\{1, 2, 5\}$ hours. Plot your feasible set of trains from the source nodes $\{\mathbf{S1}, \mathbf{S2}\}$ against departing trains at **M1** to **C1** with required layovers and waiting time. Describe your obtained plot based on your understanding. Also, report your feasible travel plan using the prescribed notation.

Hint: Try to use a heatmap as a plot!!!

Scenario C: Parting Ways

Assuming both passengers board the same train from **C1** to return to their destinations for $\{\mathbf{S1}, \mathbf{S2}\}$, but at **M1** they part their respective ways. A layover for both passengers can be expected on return as well, followed by a waiting time for passenger departing later from **M1**. In this scenario, a layover can be computed as the time between the arrival of a train at **M1** from **C1** and the time of the first passenger departure. Following layover time, a waiting time for other passenger starts, considering a maximum waiting time of $\{1, 2, 4, 10, 15\}$ hours for the other passenger at **M1** can be expected. Based on the given waiting time for the second passenger, plot the combination for feasible Out-bound trains from **M1** to $\{\mathbf{S1}, \mathbf{S2}\}$ against Inbound trains from **C1** to **M1**. Also, plot a heatmap(s) with one axis as an arrival from **C1** to **M1** and the other axis as a feasible departure from **M1** to $\{\mathbf{S1}, \mathbf{S2}\}$.

---

[1] $< DepartureTime(\text{at Origin Node}) > \_ < OriginNode > < DestinationNode > \_ < ArrivalTime(\text{at Dest. Node}) >$

For more details refer to Scenario D

[2] $\{\{\text{Train No. for S1 to M1 train, Train No. for S2 to M1}, \{\text{Train No. for all feasible train from M1 to C1}\}\}, \cdots \{\cdots\}\}$

In case if no feasible mode is available for a certain combination of train $\mathbf{S1}, \mathbf{S2}$ return feasible solution as $\{\phi\}$

Scenario D: Introducing Direct Trains from Source to Cities to City of Attractions

Railways has introduced two direct trains from **S1** to **C1** and **S2** to **C1** and return trains as well, with details given below:

- **S1** $\longrightarrow$ **M1** $\longrightarrow$ **C1**(05:30_S1C1_05:15): Departing **S1** @ 05:30, reaching **M1**@01 : 30 (+1 day) and finally arriving **C1**@05 : 15 (+1 day).
- **S1** $\longrightarrow$ **M1** $\longrightarrow$ **C1**(20:15_S1C1_22:15): Departing **S1** @ 20:15, reaching **M1**@18 : 15 (+1 day) and finally arriving **C1**@22 : 15 (+1 day).
- **S2** $\longrightarrow$ **M1** $\longrightarrow$ **C1**(15:45_S2C1_08:45): Departing **S2** @ 15:45, reaching **M1**@05 : 45 (+1 day) and finally arriving **C1**@08 : 45 (+1 day).
- **S2** $\longrightarrow$ **M1** $\longrightarrow$ **C1**(07:30_S2C1_00:45): Departing **S1** @ 07:30, reaching **M1**@22 : 45 and finally arriving **C1**@00 : 45 (+1 day).
- **C1** $\longrightarrow$ **M1** $\longrightarrow$ **S1**(08:15_C1S1_08:00): Departing **C1** @ 08:15, reaching **M1**@12 : 00 and finally arriving **S1**@08 : 00 (+1 day).
- **C1** $\longrightarrow$ **M1** $\longrightarrow$ **S1**(02:00_C1S1_04:00): Departing **C1** @ 02:00, reaching **M1**@06 : 00 and finally arriving **S1**@04 : 00 (+1 day).
- **C1** $\longrightarrow$ **M1** $\longrightarrow$ **S2**(15:00_C1S2_09:45): Departing **C1** @ 15:00, reaching **M1**@18 : 00 and finally arriving **S2**@09 : 45 (+1 day).
- **C1** $\longrightarrow$ **M1** $\longrightarrow$ **S2**(09 : 00_$C1S2$_02 : 15): Departing **C1** @ 09:00, reaching **M1**@11 : 45 and finally arriving **S2**@02 : 15 (+1 day).

1. Modify your existing Arrival-Departure boards at $\{\mathbf{S1}, \mathbf{S2}, \mathbf{M1}, \mathbf{C1}\}$ with the above newly introduced trains.
2. Perform Scenario B and Scenario C using the new Arrival-Departure Board, assuming both travelers want to avoid direct trains and have a layover at **M1**. Comment on the observation made compared to Scenario B observations. Explain.

Scenario E: Train Hopping

Assuming either of the one passenger boards a direct train from origin point $\{\mathbf{S1}, \mathbf{S2}\}$ and other passenger plans to hops into the train at **M1** . Depending upon whether it is a UP or DOWN journey, other passenger will board or De-board at the meeting point. An interesting takeaway from such an itinerary will be about zero-layover time. But it can have some significant waiting time for passenger who prefer to hop into another train. Considering the maximum waiting time at **M1** for hopping passenger is $\{\mathbf{1}, \mathbf{2}, \mathbf{5}\}$ hours. What is the feasible itinerary when:

1. passenger from **S1** plans to hop into direct train from **S2** $\longrightarrow$ **C1**.
2. passenger from **S2** plans to hop out from direct train **C1** $\longrightarrow$ **S1**.
3. passenger from **S1** only prefers to travel in a direct train.
4. Try to visualize the above feasible itinerary using an appropriate plot.