ITP Mini Project

Problem Statement

For a seamless eCommerce shopping experience, it is essential to deliver the product promptly to the customer. And that's where a professional courier service plays a vital role. "Fastrack" courier company stores the relevant data of its clients and parcels in the form of a dictionary. Create a dictionary for storing shipment information in key-value pairs. Shipment id is used as a key and a list of other attributes like sender, receiver, start date, Delivery Date,sender_location, Receiver_location, Delivery status, Shipping cost is associated with the shipment id. Use the data shown in the table below.

Shipment id	Sender	Receiver	Start date	Delivery Date	sender_location	Receiver_location	Delivery status	Shipping cost
101	1	3	14-03-2020	25-03-2020	Area1	Area 6	Delivered	198
102	4	1	18-06-2020	09-07-2020	Area2	Area4	Delivered	275
103	2	3	01-12-2020	Null	Area5	Area1	In-Transit	200
104	1	5	23-06-2020	25-06-2020	Area1	Area4	Delivered	314
105	3	4	29-08-2020	10-09-2020	Area5	Area3	Delivered	275
106	5	2	28-06-2020	Null	Area3	Area1	In-Transit	270

Use below table to refer to clients data. Please note that a client can be a sender or receiver.

Client_id	Client Name
1	Phillip
2	Omega
3	Ramya
4	Romesh
5	John

Q1. Create a Dictionary of lists to store the information of shipments given in the table

Solution Hint:
Dictionary can be created using id as key and a list of other data as value. The first row can be created as follows
ship = {101:[1, 3, '14-03-2020','25-03-2020','Area1','Area6','Delivered',198],key2:[values….]}

Q2. Create a Dictionary to store the information of clients given in the table.

Solution Hint:
Create the dictionary with key as Client_id and value as Client Name.
dict_client = {client_id1:client_name1,client_id2:client_name2.....}

Q3. Write a code to replace clients' IDs with their respective names in the shipment dictionary using a loop and dictionary comprehension.

Solution Hint:
The original row1 is like below

Shipment id	Sender	Receiver	Start date	Delivery Date	sender_location	Receiver_location	Delivery status	Shipping cost
101	1	3	14-03-2020	25-03-2020	Area1	Area 6	Delivered	198

This should be changed as

Shipment id	Sender	Receiver	Start date	Delivery Date	sender_location	Receiver_location	Delivery status	Shipping cost

101	Phillip	Ramya	14-03-2020	25-03-2020	Area1	Area 6	Delivered	198

Use loop/dictionary comprehension to replace for all sender/receiver IDs by Names.

Q4. Print all shipment details that are sent by Phillip

Solution Hint:
Use the modified dictionary of Question 3
Use loops to check Phillip in the sender data
The expected output is
['Phillip', 'Ramya', '14-03-2020', '25-03-2020', 'Area1', 'Area6', 'Delivered'],

['Phillip',, 'John', '23-06-2020', '25-06-2020', 'Area1', 'Area4', 'Delivered']

Q5. Print all shipment details that are received by Ramya

Solution Hint:
Use the modified dictionary of Question 3
Use loops to check Ramya in the receiver data
The expected output is
['Phillip', 'Ramya', '14-03-2020', '25-03-2020', 'Area1', 'Area6', 'Delivered'],
['Omega III', 'Ramya', '01-12-2020', 'Null', 'Area5', 'Area1', 'In-Transit']

Q6. Print all shipments which are in 'In-Transit' status

Solution Hint:
Use the modified dictionary of Question 3
Use loops to check 'In Transit' in the status
The expected output is
['Omega III', 'Ramya', '01-12-2020', 'Null', 'Area5', 'Area1', 'In-Transit'],
['John', 'Omega III', '28-09-2020', 'Null', 'Area3', 'Area1', 'In-Transit']

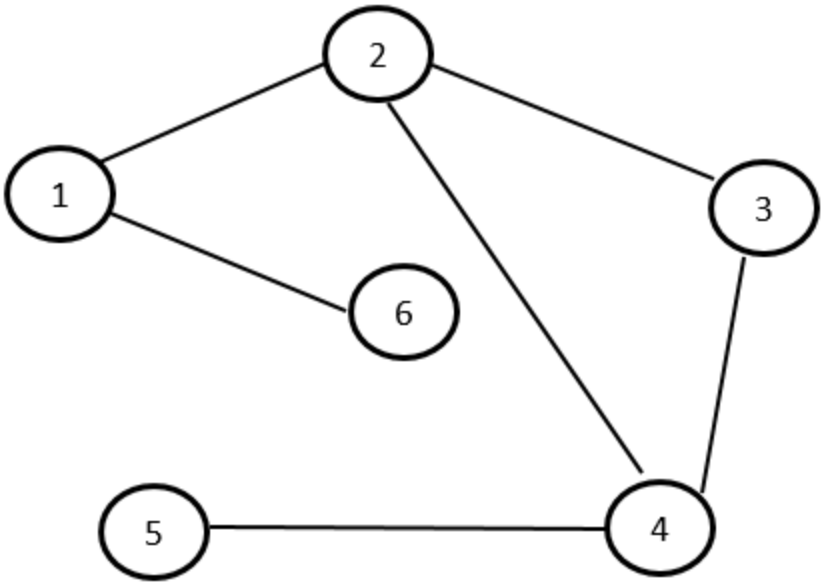
Q7. Print all shipments which are delivered within 7 days of courier Start date

Solution Hints:
Use Datetime library.
Convert the date-related data i.e. 'start date' and 'delivery date' to date data type.
Use timedelta for calculating date difference; using loops compare the date difference with 7 days.
Expected output is
['Phillip', 'John', '23-06-2020', '25-06-2020', 'Area1', 'Area4', 'Delivered']

Q8. Print all shipments which are delivered after 15 days of courier start date or have not yet been delivered.

Solution Hints:
Use Datetime library
Convert the date-related data i.e. 'start date' and 'delivery date' to date data type
Use timedelta for calculating date difference
Expected output is
[4, 1, '18-06-2020', '09-07-2020', 'Area2', 'Area4', 'Delivered', 275]
[2, 3, '01-12-2020', 'Null', 'Area5', 'Area1', 'In-Transit', 200]
[5, 2, '28-09-2020', 'Null', 'Area3', 'Area1', 'In-Transit', 270]

Q9. The graph is used to represent networks of pickup and delivery areas. Consider below the graph diagram for given area locations in the table.
Sender location and receiver location are represented by nodes with area numbers.
The connection between two nodes shows a route exists between those areas. E.g there exists a path from area 1 to area 2 but there is no direct route between area1 and area5. please note that these routes are bidirectional.
To reach area5 from area1, the delivery person can take any route like 1-2-4-5 or 1-2-3-4-5



	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	1	0	0
3	0	1	0	1	0	0
4	0	1	1	0	1	0
5	0	0	0	1	0	0
6	1	0	0	0	0	0

Solution Hint:

Graph Theory

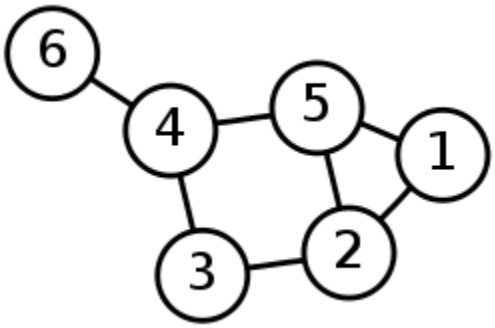
Graphs are mathematical concepts that have found many uses in computer science. Graphs come in many different flavors, many of which have found uses in computer programs. Some flavors are

- Simple graphs
- Undirected or directed graphs
- Cyclic or acyclic graphs
- labeled graphs
- Weighted graphs
- Infinite graphs
- ... and much more too numerous to mention.

Most graphs are defined as a slight alteration of the following rules.

- A graph is made up of two sets called Vertices and Edges.
- The Vertices are drawn from some underlying type, and the set may be finite or infinite.
- Each element of the Edge set is a pair consisting of two elements from the Vertices set.
- Graphs are often depicted visually, by drawing the elements of the Vertices set as boxes or circles, and drawing the elements of the edge set as lines or arcs between the boxes or circles. There is an arc between v1 and v2 if (v1,v2) is an element of the Edge set.

Adjacency. If (u,v) is in the edge set we say u is adjacent to v (which we sometimes write as u ~ v).



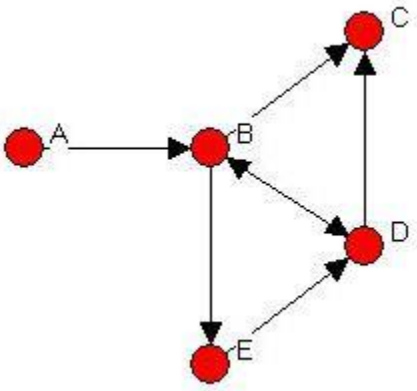
Has the following parts.

- The underlying set for the Vertices set is the integers.
- The Vertices set = {1,2,3,4,5,6}
- The Edge set = {(6,4),(4,5),(4,3),(3,2),(5,2),(2,1),(5,1)}

Kinds of Graphs

Various flavors of graphs have the following specializations and particulars about how they are usually drawn.

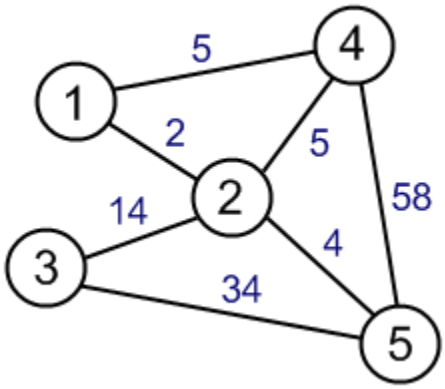
- Undirected Graphs.
  In an undirected graph, the order of the vertices in the pairs in the Edge set doesn't matter. Thus, if we view the sample graph above we could have written the Edge set as {(4,6),(4,5),(3,4),(3,2),(2,5)),(1,2)),(1,5)}. Undirected graphs usually are drawn with straight lines between the vertices.
  The adjacency relation is symmetric in an undirected graph, so if u ~ v then it is also the case that v ~ u.
- Directed Graphs.
  In a directed graph the order of the vertices in the pairs in the edge set matters. Thus u is adjacent to v only if the pair (u,v) is in the Edge set. For directed graphs, we usually use arrows for the arcs between vertices. An arrow from u to v is drawn only if (u,v) is in the Edge set. The directed graph below



Has the following parts.

- The underlying set for the Vertices is set in capital letters.
- The Vertices set = {A,B,C,D,E}
- The Edge set = {(A,B),(B,C),(D,C),(B,D),(D,B),(E,D),(B,E)}

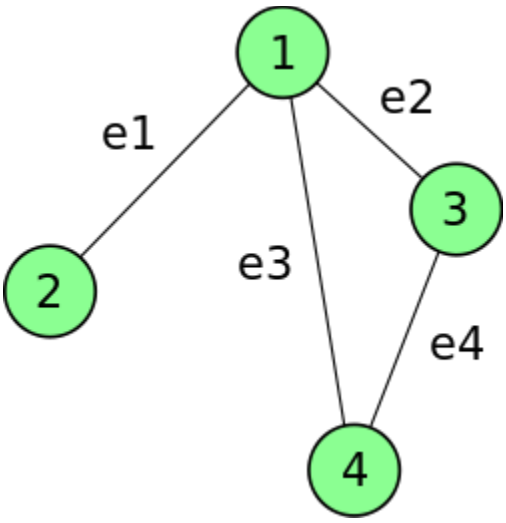
- Note that both (B,D) and (D,B) are in the Edge set, so the arc between B and D is an arrow in both directions.
-
- Weighted Graphs.
  A weighted graph is an edge labeled graph where the labels can be operated on by the usual arithmetic operators, including comparisons like using less than and greater than. In Haskell, we'd say the edge labels are in the Num class. Usually, they are integers or floats. The idea is that some edges may be more (or less) expensive, and this cost is represented by the edge labels or weight. In the graph below, which is an undirected graph, the weights are drawn adjacent to the edges and appear in dark purple.



Here we have the following parts.

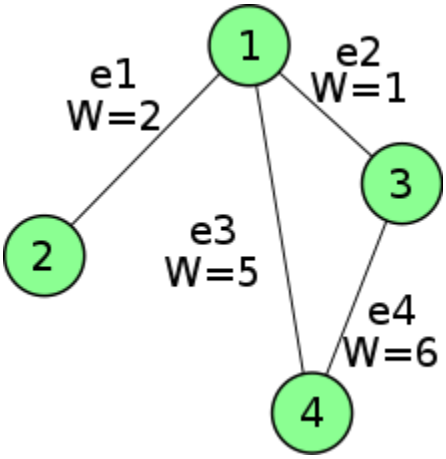
- The underlying set for the Vertices set is Integer.
- The underlying set for the weights is Integer.
- The Vertices set = {1,2,3,4,5}
- The Edge set = {(1,4,5) ,(4,5,58) ,(3,5,34) ,(2,4,5) ,(2,5,4) ,(3,2,14) ,(1,2,2)}

Adjacency Matrix



$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Adjacency Matrix - Weighted Graph



$$\begin{bmatrix} 2 & 1 & 5 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 5 & 6 \end{bmatrix}.$$

The given problem can be solved using following approach:

1. The adjacency matrix can be stored as a nested list.
2. Write a recursive function for finding all routes

3. Keep a track of places that are already visited to confirm that no place is visited repeatedly or in a cyclic manner.
4. Use the appropriate Graph traversal method to find the paths. (Breadth-First Search/Depth-First Search)