Zülal Karın **19/03/2022**
12622989076 **Section 02**

CMPE 343 HW-1 REPORT

Problem Statement and Code Design

Task 1:

In the first question of the assignment, I need to create an undirected graph with the input the user entered, and determine whether the city in the second line of the input has a cycle in the graph. Next, if the vertex has a cycle, I need to print the cities in the cycle, in other words the vertexes, in alphabetical order. To do this, I implemented a Graph class that creates an undirected graph with user data, a Cycle class that checks if there is a Cycle with Depth First Search and holds the vertices of the cycle, and a driver class that takes inputs from the user and outputs them. I have added a structure chart below to better explain the structure chart I created.

Task 2:

In the second question, just like the first question, I had to create an Undirected graph with the data in the first line of the input. What is required of me is to find paths of the length given in the fourth line of the input, between the city names, that is, the vertexes, in the second and third lines of the input. I created these paths in a class called Paths and tried to sort them alphabetically. I created the following structure chart to see the things I've done, separatley.

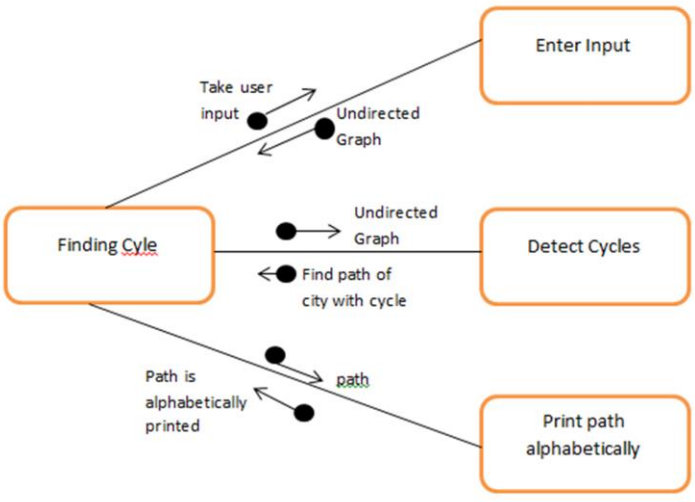


Figure 1: Structure chart of Q1

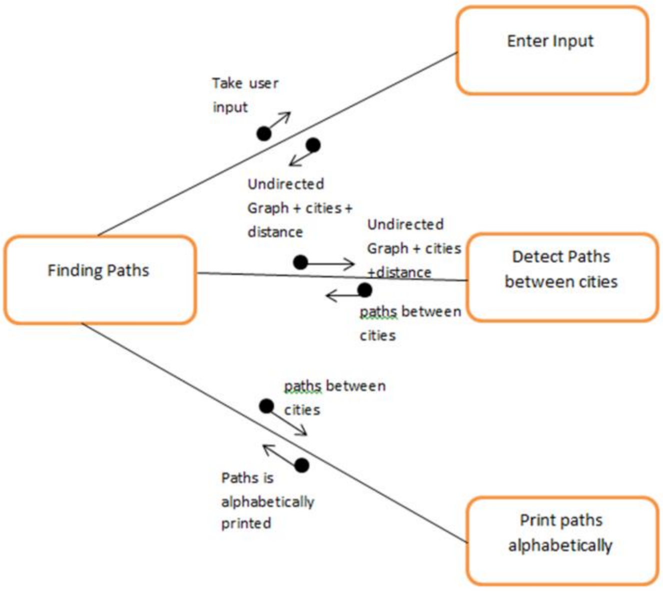

Enter Input
Take user input
Undirected Graph + cities + distance
Finding Paths
Undirected Graph + cities +distance
Detect Paths between cities
paths between cities
paths between cities
Paths is alphabetically printed
Print paths alphabetically

Figure 2: Structure chart of Q2

Implementation and Functionality

Task 1:

For this task, I created Graph, Cycle and Driver classes. I will describe these classes and the methods within them using the Structure Diagram in figure1.

1)Enter Input: I received and separated String inputs separated by commas and dashes by the user in the Driver class. I converted string inputs to integers. I then created an undirected Graph by calling the Graph class.

2)Detect Cycles: To find out if the given vertex has a cycle, I created the Cycle class and implemented the dfs method with three parameters. In this method, I put all the vertex in the cycle into an arrayList. So I created a path.

3)Print Path Alphabetically: In the arrayList I created in the Cycle class, I created a list containing the vertex ids of the Cycle. In the Driver class, I printed the elements in this list in alphabetical order by using collections class and print the output.

//pseudocode of dfs

Task 2:

For this task, I created Graph, Paths and Driver classes. I will describe these classes and the methods within them using the Structure Diagram in figure2.

1)Enter Input: Just like in part1, I took the vertex and edge information separated by hyphens and commas in the first line of the input from the user in the Driver class. I then cast them an arrayList called vertexArray to use their index number. So I would have converted the vertexes into integers. I got the cities in the second and third rows of the input, and the distance between the cities in the fourth row.

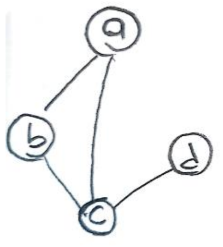
2)Detect Paths Between Cities: By creating a Path object in the Driver class, I found all the paths between the two cities given in the input. I did this using Stack in Paths class. I created a recursive method called enumerate. In this method, I put the other cities between the two cities in the stack and created a path.

3)Print Paths Alphabetically: I said that I created a Paths object in the Driver class. This object holds all the paths between the two cities we entered. What we need is to find paths with a certain distance and print them. To do this, I tried to print the vertexes in the stack to the screen in alphabetic order by using collections class.

Testing

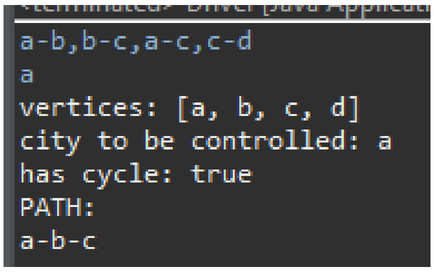
Task 1:

I tested my algorithm with the graph and input values seen in Task1 below. I got the input correctly and printed the vertexes in the cycled in alphabetical order (Figure 3). If there is no cyle, I printed "No cycle" in the screen as you can see in figüre 4.



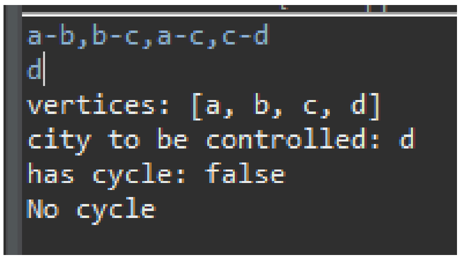
1st input;
a-b, a-c, b-c, c-d
a

2nd input:
a-b, a-c, b-c, c-d
d



```
a-b,b-c,a-c,c-d
a
vertices: [a, b, c, d]
city to be controlled: a
has cycle: true
PATH:
a-b-c
```

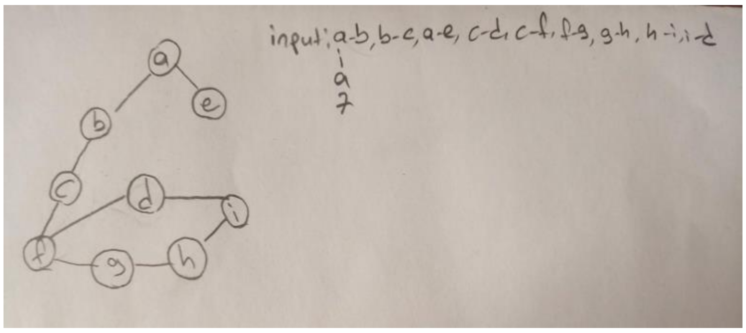
Figure 3: Output for 1st input

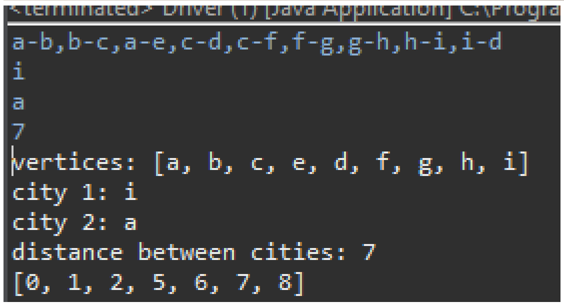


```
a-b,b-c,a-c,c-d
d
vertices: [a, b, c, d]
city to be controlled: d
has cycle: false
No cycle
```

Figure 4: Output for 2st input

Task 2: I tested my algorithm with the graph and input values seen in Task2 below. As you can see in figure-5, I took the inputs, separated them, used the path finding algorithm and printed the path of the given length on the screen. However, the only problem is that the path I printed was printed according to the ID of the vertex.





```
<terminated> Driver (1) [Java Application] C:\Progra
a-b,b-c,a-e,c-d,c-f,f-g,g-h,h-i,i-d
i
a
7
vertices: [a, b, c, e, d, f, g, h, i]
city 1: i
city 2: a
distance between cities: 7
[0, 1, 2, 5, 6, 7, 8]
```

Figure 5: Output for 1st input

Final Assessments

-Getting a String input made it difficult for me, as I'm used to always taking the input as an integer in graphs.

-In Task 1, I could easily find out if a vertex has a cycle, but it was difficult for me to keep the vertex it contains.

-In Task 2, it was okay for me to find all the paths that the vertex has, but I had a hard time printing vertexes with a certain length.

-In this assignment, I had the opportunity to implement the graph and used most graph features.