Home Challenge For Software Development Intern

Step by Step Approch

Code Editor: Visual Studio Code.

Need To install:

Download and install NodeJs: https://nodejs.org/dist/v14.15.4/node-v14.15.4-x64.msi

Install from cmd: npm install csv-parser

How I approach to solve the Problem:

1.I read the csv file. It is not easy to read a CSV file in JavaScript like other language Python, R. For Doing this I need a package which I already mention it.

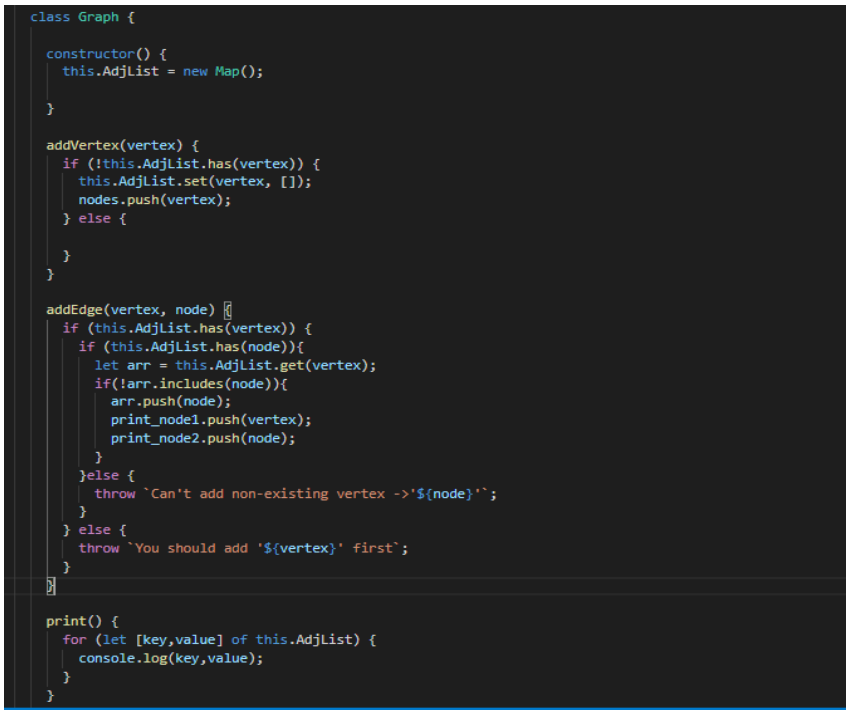
```
//create a file structure
const fs = require('fs')
//create a CSV parser
const csv = require('csv-parser')
```

2. I create list and save the data into list from CSV file to make my work easy.

```
//create array or list to save the column data from csv file
var name=[];
var volunteerName=[];
var volunteerId=[];
var date=[];
var shift=[];
var shiftReason=[];
```

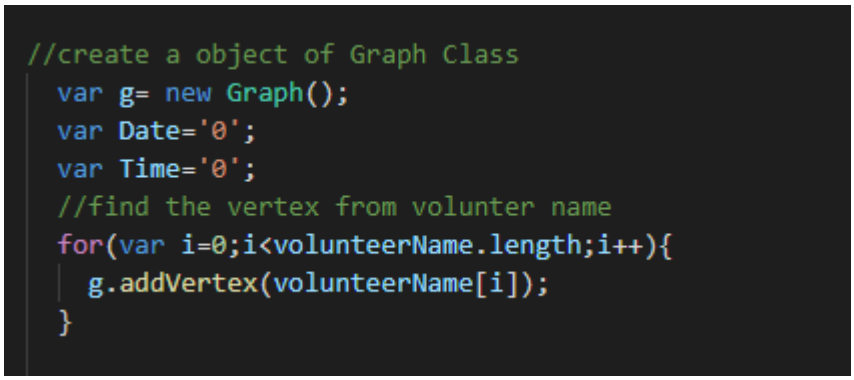
```
fs.createReadStream('volunteer_attendance_data.csv')
  .pipe(csv())
  .on('data', function (row){

      date.push(row.date)
      shift.push(row.shift)
      volunteerId.push(row.volunteerId)
      volunteerName.push(row.volunteerName)
      shiftReason.push(row.shiftReason)
    })
```

3. Create a Graph class to create a graph structure form the csv values. In graph I use adjacent list method to create the graph structure.In Graph class has constructor which has a AdjList map to save the Vertex and their nodes based on Edge. There are also three method In graph class addVertex,addEdge and print. addVertx add the vertex and addEdge find the Edge and save the nodes along to the Vertex. Print method just print the graph



```
class Graph {

  constructor() {
    this.AdjList = new Map();

  }

  addVertex(vertex) {
    if (!this.AdjList.has(vertex)) {
      this.AdjList.set(vertex, []);
      nodes.push(vertex);
    } else {

    }
  }

  addEdge(vertex, node) {
    if (this.AdjList.has(vertex)) {
      if (this.AdjList.has(node)){
        let arr = this.AdjList.get(vertex);
        if(!arr.includes(node)){
          arr.push(node);
          print_node1.push(vertex);
          print_node2.push(node);
        }
      }else {
        throw `Can't add non-existing vertex ->'${node}'`;
      }
    } else {
      throw `You should add '${vertex}' first`;
    }
  }

  print() {
    for (let [key,value] of this.AdjList) {
      console.log(key,value);
    }
  }
```

4. Crete a graph object and find the vertexes according to the name of volunteerName. I use a for loop to which length is equal to VolunteerName list then I call the addvertex method of graph class which create the adjacent vertex.



```
//create a object of Graph Class
  var g= new Graph();
  var Date='0';
  var Time='0';
  //find the vertex from volunter name
  for(var i=0;i<volunteerName.length;i++){
    g.addVertex(volunteerName[i]);
  }
```

5. This step is the most important step. In this step I create a logic to find out the edge between Vertexes. I take two variable Date and Time and Initialize with value ‘0’. There are nested for loop first loop based on nodes which I get from addVertex method like nodes=[Bobita,Shabana,Rajjak,Kabori,Illias] with length 5 and second for loop based on length of VolunteerName. In first if condition it assign the date and time along with first node. Like it traverse and find the date and time of Bobita and in else if condition it check that is date and time is same then it call the addEdge method of Graph and send the nodes value and the volunteerName which has same date and shift with Node[i] and save it as adjacent value of the nodes.

```
//find the edge for verteces

for(var k=0;k<nodes.length;k++){
  for(var j=0;j<volunteerName.length;j++){
    if(nodes[k]==volunteerName[j]&& Date!=date[j] && Time!=shift[j]){
      Date=date[j];
      Time=shift[j];
    }

    else if(nodes[k]!=volunteerName[j]&& Date==date[j] && Time==shift[j]){
      g.addEdge(nodes[k],volunteerName[j]);
    }
  }
  Date='0'
  Time='0'
}
```

7.Then I write a code to save the Vetex and it nodes as CSV file.

```
function writeToCSVFile(n1,n2) {
  const filename = 'output.csv';

  fs.writeFile(filename, extractAsCSV(n1,n2), err => {
    if (err) {
      console.log('Error writing to csv file', err);
    } else {
      console.log(`saved as ${filename}`);
    }
  });
}

function extractAsCSV(n1,n2) {
  const header = ['Node1,Node2'];
  first_node=[];
  secode_node=[]
  for(var n=0;n<n1.length;n++){
    first_node.push(n1[n])
    secode_node.push(n2[n])
  }
  var rows=[first_node,secode_node]
  return header.concat(rows).join('\n');
}
```

Challenged I have Faced:

- I am not fluent in JavaScript so it takes too much time to read and write the csv file.
- I read Graph theory in academic books or slide but I never try It in code. With an unknown language it was too hard to me.
- I have faced a problem to create the logic for connection Edges.

Limitation Of my Program:

- Code unable to find the perfect Edge for lower row voluenteerName.
- Code can't create the CSV file with nice way.
- Code can't find the weight of the graph.