Sarvesh Upendra Rembhotkar

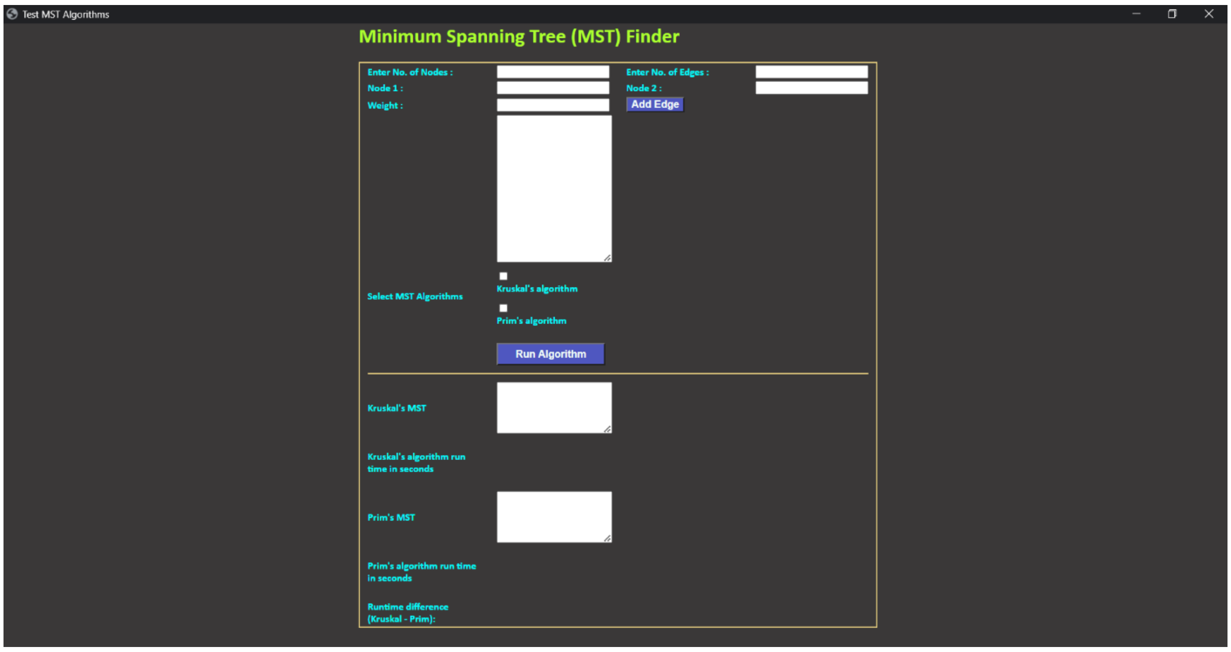
UTA ID: 1001966297

CSE 5311 - 004 Programming Project

INTRODUCTION

I have selected Project 4 (Minimum Spanning Tree) as my project for this course (i.e., CSE 5311). This project is implemented using Python and Python eel package which is used for creating HTML GUI. Using eel, functions can be exposed in python which can be directly accessed from JavaScript.

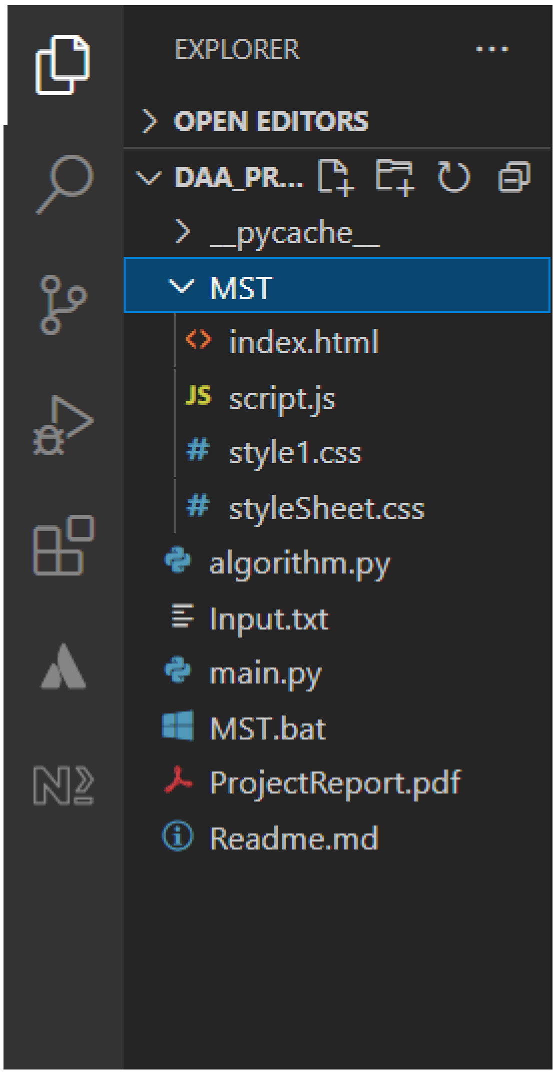
The UI of this project is as follows:



(Figure 1)

FILE STRUCTURE

The folder structure implemented in this project looks like as represented in Figure 2.

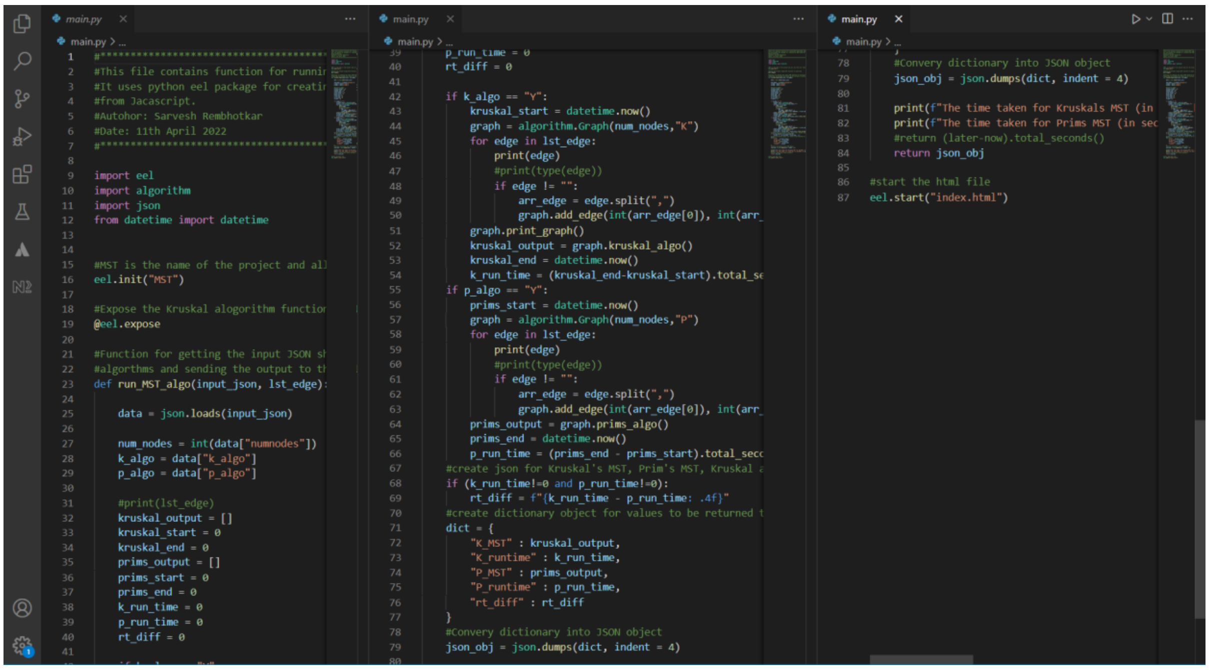


(Figure 2)

The main directory is DAA_Project_sxr6297, and it contains the ‘algoirthm.py’ and ‘main.py’ python files. The Reademe.md file contains the instruction for installing and running this project. The Graph class and functions for calculating MST using Kruskal’s and Prim’s algorithm are implemented in algorithm.py file. The main.py file contains the function for running the selected algorithms. The function written in main.py is exposed to JS using @eel.expose command which makes it accessible to the script.js file which is used for implementing the front-end part of this project. There is a subdirectory called MST in the parent directory DAA which contains the ‘script.js’ file to handle the frontend operations initiated from the GUI. The files ‘index.html’, ‘style1.css’ and ‘styleSheet.css’ are used for structuring and styling the GUI. Any one style sheet can be used for styling the GUI. It also contains a batch file ‘MST.bat’ to run the program directly by double clicking it. The input.txt file contains the input data that can be used for testing this program.

PYTHON BACKEND FILES

main.py



```python
#**************************************
#This file contains function for runnir
#It uses python eel package for creatir
#from Jacascript.
#Autohor: Sarvesh Rembhotkar
#Date: 11th April 2022
#**************************************

import eel
import algorithm
import json
from datetime import datetime


#MST is the name of the project and all
eel.init("MST")

#Expose the Kruskal alogorithm functior
@eel.expose

#Function for getting the input JSON sł
#algorthms and sending the output to tł
def run_MST_algo(input_json, lst_edge):

    data = json.loads(input_json)

    num_nodes = int(data["numnodes"])
    k_algo = data["k_algo"]
    p_algo = data["p_algo"]

    #print(lst_edge)
    kruskal_output = []
    kruskal_start = 0
    kruskal_end = 0
    prims_output = []
    prims_start = 0
    prims_end = 0
    k_run_time = 0
    p_run_time = 0
    rt_diff = 0

    if k_algo == "Y":
        kruskal_start = datetime.now()
        graph = algorithm.Graph(num_nodes,"K")
        for edge in lst_edge:
            print(edge)
            #print(type(edge))
            if edge != "":
                arr_edge = edge.split(",")
                graph.add_edge(int(arr_edge[0]), int(arr_
        graph.print_graph()
        kruskal_output = graph.kruskal_algo()
        kruskal_end = datetime.now()
        k_run_time = (kruskal_end-kruskal_start).total_se
    if p_algo == "Y":
        prims_start = datetime.now()
        graph = algorithm.Graph(num_nodes,"P")
        for edge in lst_edge:
            print(edge)
            #print(type(edge))
            if edge != "":
                arr_edge = edge.split(",")
                graph.add_edge(int(arr_edge[0]), int(arr_
        prims_output = graph.prims_algo()
        prims_end = datetime.now()
        p_run_time = (prims_end - prims_start).total_secc
    #create json for Kruskal's MST, Prim's MST, Kruskal a
    if (k_run_time!=0 and p_run_time!=0):
        rt_diff = f"{k_run_time - p_run_time: .4f}"
    #create dictionary object for values to be returned t
    dict = {
        "K_MST" : kruskal_output,
        "K_runtime" : k_run_time,
        "P_MST" : prims_output,
        "P_runtime" : p_run_time,
        "rt_diff" : rt_diff
    }
    #Convery dictionary into JSON object
    json_obj = json.dumps(dict, indent = 4)

    print(f"The time taken for Kruskals MST (in
    print(f"The time taken for Prims MST (in sec
    #return (later-now).total_seconds()
    return json_obj

#start the html file
eel.start("index.html")
```

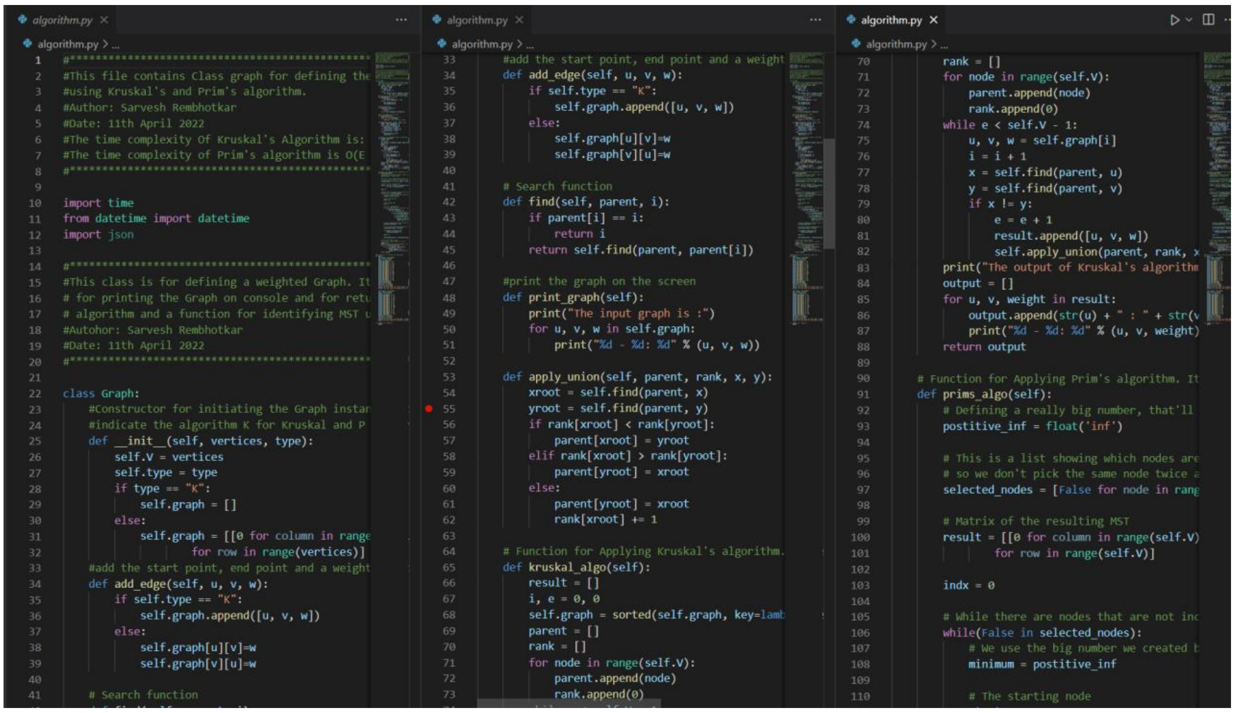
(Figure 3)

The main.py is completely responsible for running the program from the start. It imports the algorithm package written for implementing the MST algorithms and the python eel package (eel can be installed by running the command “pip install eel” from command line), This file initialises the web folder MST using eel.init(“MST”) command and exposes the function run_MST_algo() using @eel.expose command. Eel allows annotating methods and functions in Python so that we can call them from JavaScript and vice versa. This library is designed to take the disturbance out of writing simple and short GUI applications.

When main.py is run, eel launches the index.html page as it can host the local web server. The functions written in script.js file is called from the front-end HTML file. The .css files are used for styling the HTML page. The look and feel of the index.html page can be changed by modifying any of the two CSS files and referencing it in the HTML page.

The function ‘run_MST_algo’ is defined for getting the input JSON object and graph edge list shared by calling JavaScript function when user clicks on “Run Algorithm” button on the UI. It reads the parameters from JSON object such as no. of nodes, no. of edges, algorithms to run etc. and creates the graph object from the information it receives from the front end. It then calls the specific algorithm function, calculates execution time for each of the algorithm, the time difference between the execution time if both the algorithms are selected and sends the MST, execution time and time difference to the calling function in the form of JSON object.

algorithm.py


algorithm.py

```python
#**************************************************
#This file contains Class graph for defining the
#using Kruskal's and Prim's algorithm.
#Author: Sarvesh Rembhotkar
#Date: 11th April 2022
#The time complexity Of Kruskal's Algorithm is:
#The time complexity of Prim's algorithm is O(E
#**************************************************

import time
from datetime import datetime
import json

#**************************************************
#This class is for defining a weighted Graph. It
# for printing the Graph on console and for retu
# algorithm and a function for identifying MST u
#Autohor: Sarvesh Rembhotkar
#Date: 11th April 2022
#**************************************************

class Graph:
    #Constructor for initiating the Graph instar
    #indicate the algorithm K for Kruskal and P
    def __init__(self, vertices, type):
        self.V = vertices
        self.type = type
        if type == "K":
            self.graph = []
        else:
            self.graph = [[0 for column in range
                    for row in range(vertices)]
    #add the start point, end point and a weight
    def add_edge(self, u, v, w):
        if self.type == "K":
            self.graph.append([u, v, w])
        else:
            self.graph[u][v]=w
            self.graph[v][u]=w

    # Search function
```

algorithm.py

```python
    #add the start point, end point and a weight
    def add_edge(self, u, v, w):
        if self.type == "K":
            self.graph.append([u, v, w])
        else:
            self.graph[u][v]=w
            self.graph[v][u]=w

    # Search function
    def find(self, parent, i):
        if parent[i] == i:
            return i
        return self.find(parent, parent[i])

    #print the graph on the screen
    def print_graph(self):
        print("The input graph is :")
        for u, v, w in self.graph:
            print("%d - %d: %d" % (u, v, w))

    def apply_union(self, parent, rank, x, y):
        xroot = self.find(parent, x)
        yroot = self.find(parent, y)
        if rank[xroot] < rank[yroot]:
            parent[xroot] = yroot
        elif rank[xroot] > rank[yroot]:
            parent[yroot] = xroot
        else:
            parent[yroot] = xroot
            rank[xroot] += 1

    # Function for Applying Kruskal's algorithm.
    def kruskal_algo(self):
        result = []
        i, e = 0, 0
        self.graph = sorted(self.graph, key=lamb
        parent = []
        rank = []
        for node in range(self.V):
            parent.append(node)
            rank.append(0)
```

algorithm.py

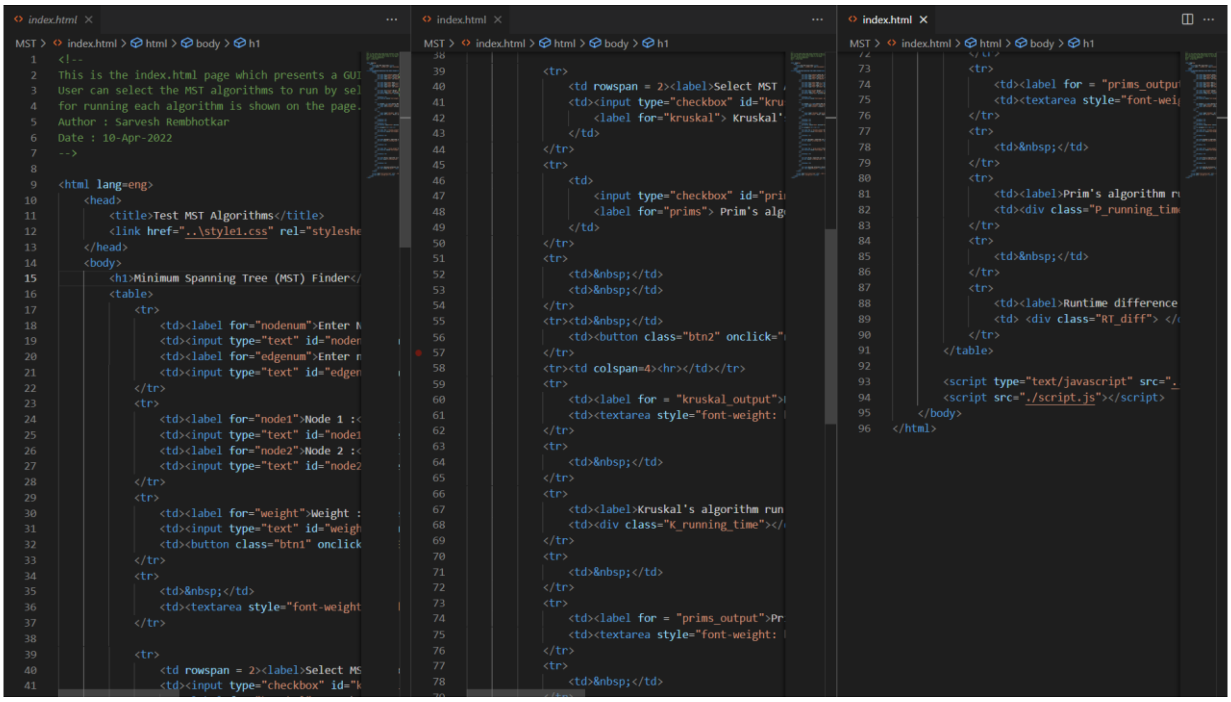
```python
        rank = []
        for node in range(self.V):
            parent.append(node)
            rank.append(0)
        while e < self.V - 1:
            u, v, w = self.graph[i]
            i = i + 1
            x = self.find(parent, u)
            y = self.find(parent, v)
            if x != y:
                e = e + 1
                result.append([u, v, w])
                self.apply_union(parent, rank, x
        print("The output of Kruskal's algorithm
        output = []
        for u, v, weight in result:
            output.append(str(u) + " : " + str(v
            print("%d - %d: %d" % (u, v, weight)
        return output

    # Function for Applying Prim's algorithm. It
    def prims_algo(self):
        # Defining a really big number, that'll
        postitive_inf = float('inf')

        # This is a list showing which nodes are
        # so we don't pick the same node twice a
        selected_nodes = [False for node in rang

        # Matrix of the resulting MST
        result = [[0 for column in range(self.V)
                for row in range(self.V)]

        indx = 0

        # While there are nodes that are not inc
        while(False in selected_nodes):
            # We use the big number we created t
            minimum = postitive_inf

            # The starting node
```

(Figure 4)

The above figure shows the algorithm.py file which includes ‘class Graph’ for defining the spanning tree data structure and functions ‘kruskal_algo’ and ‘prims_algo’ for determining the minimum spanning tree (MST) using Kruskal’s and Prim’s algorithm respectively. Function ‘add_edge’ is defined for adding the two nodes and the edge weight between the two nodes. ‘print_graph’ function is defined to print the user-defined spanning tree on the GUI.

The function main() in algorithm.py is written for testing the algorithms only and it contains hard coded graph data. This function is not called anywhere in the actual project.

FRONTEND FILES



```
<!--
This is the index.html page which presents a GUI
User can select the MST algorithms to run by sel
for running each algorithm is shown on the page.
Author : Sarvesh Rembhotkar
Date : 10-Apr-2022
-->

<html lang=eng>
    <head>
        <title>Test MST Algorithms</title>
        <link href="..\style1.css" rel="styleshe
    </head>
    <body>
        <h1>Minimum Spanning Tree (MST) Finder</
        <table>
            <tr>
                <td><label for="nodenum">Enter N
                <td><input type="text" id="noden
                <td><label for="edgenum">Enter n
                <td><input type="text" id="edgen
            </tr>
            <tr>
                <td><label for="node1">Node 1 :<
                <td><input type="text" id="node1
                <td><label for="node2">Node 2 :<
                <td><input type="text" id="node2
            </tr>
            <tr>
                <td><label for="weight">Weight :
                <td><input type="text" id="weigh
                <td><button class="btn1" onclick
            </tr>
            <tr>
                <td>&nbsp;</td>
                <td><textarea style="font-weight
            </tr>

            <tr>
                <td rowspan = 2><label>Select MS
                <td><input type="checkbox" id="k
```

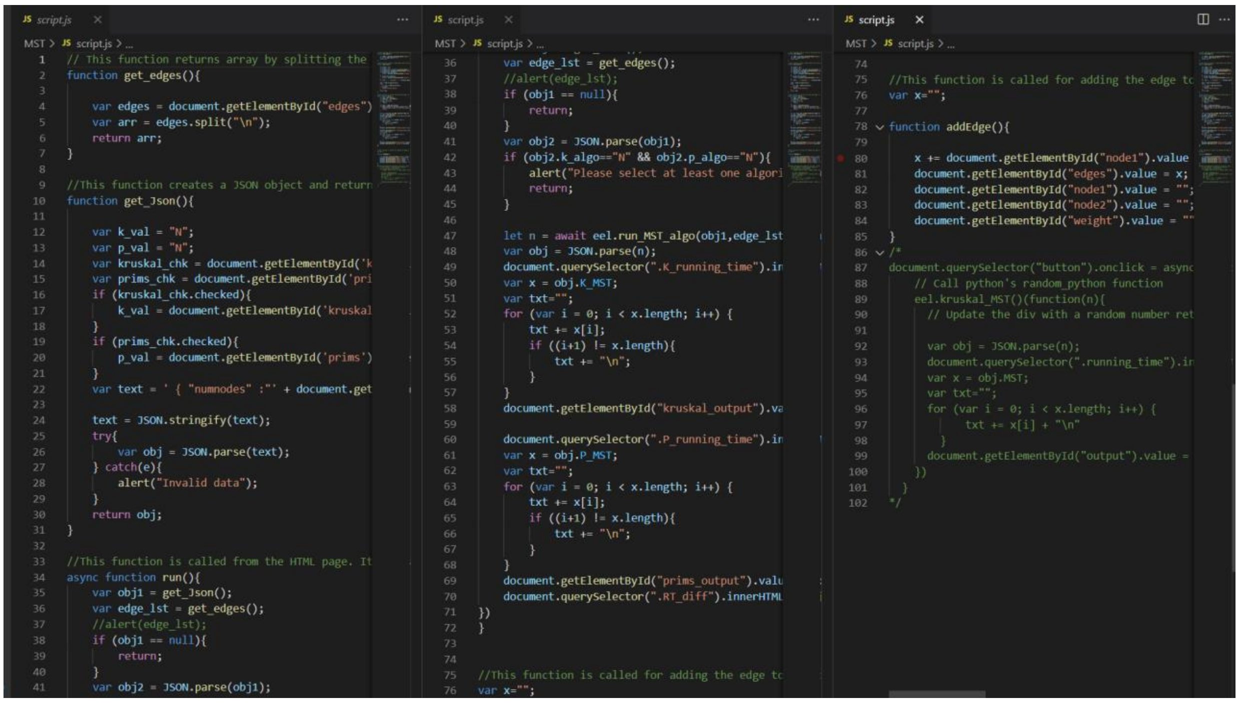
```
            <tr>
                <td rowspan = 2><label>Select MST
                <td><input type="checkbox" id="kru
                    <label for="kruskal"> Kruskal'
                </td>
            </tr>
            <tr>
                <td>
                    <input type="checkbox" id="pri
                    <label for="prims"> Prim's alg
                </td>
            </tr>
            <tr>
                <td>&nbsp;</td>
                <td>&nbsp;</td>
            </tr>
            <tr><td>&nbsp;</td>
                <td><button class="btn2" onclick="
            </tr>
            <tr><td colspan=4><hr></td></tr>
            <tr>
                <td><label for = "kruskal_output">
                <td><textarea style="font-weight:
            </tr>
            <tr>
                <td>&nbsp;</td>
            </tr>
            <tr>
                <td><label>Kruskal's algorithm run
                <td><div class="K_running_time"></
            </tr>
            <tr>
                <td>&nbsp;</td>
            </tr>
            <tr>
                <td><label for = "prims_output">Pr
                <td><textarea style="font-weight:
            </tr>
            <tr>
                <td>&nbsp;</td>
```

```
            <tr>
                <td><label for = "prims_outpu
                <td><textarea style="font-wei
            </tr>
            <tr>
                <td>&nbsp;</td>
            </tr>
            <tr>
                <td><label>Prim's algorithm ru
                <td><div class="P_running_tim
            </tr>
            <tr>
                <td>&nbsp;</td>
            </tr>
            <tr>
                <td><label>Runtime difference
                <td> <div class="RT_diff"> </
            </tr>
        </table>

        <script type="text/javascript" src=".
        <script src="./script.js"></script>
    </body>
</html>
```

(Figure 5)

The file 'index.html' represents a GUI for entering the number of nodes and no. of edges. Edges to the graph (both directed as well as undirected) and the weight for respective edge can be added by entering the data in "Node 1", "Node 2" and "Weight" text boxes. When the adjacent button is clicked, the edge is added to the text area shown on the UI. User must enter input data for all the edges before selecting and running the algorithms. User can select Prim's or Kruskal's or both the algorithms by selecting the respective check box on the UI. When the "Run Algorithm" button is clicked, the MST for either or both the algorithms are shown on the screen. Also, the time taken by each of the algorithms along with the differences in their run time if both the algorithms are selected is also displayed on the UI.

Style1.css or stylesheet.css files define styles for the elements used in the HTML page.



```
// This function returns array by splitting the
function get_edges(){

    var edges = document.getElementById("edges")
    var arr = edges.split("\n");
    return arr;
}

//This function creates a JSON object and return
function get_Json(){

    var k_val = "N";
    var p_val = "N";
    var kruskal_chk = document.getElementById('k
    var prims_chk = document.getElementById('pri
    if (kruskal_chk.checked){
        k_val = document.getElementById('kruskal
    }
    if (prims_chk.checked){
        p_val = document.getElementById('prims')
    }
    var text = ' { "numnodes" :"' + document.get

    text = JSON.stringify(text);
    try{
        var obj = JSON.parse(text);
    } catch(e){
        alert("Invalid data");
    }
    return obj;
}

//This function is called from the HTML page. It
async function run(){
    var obj1 = get_Json();
    var edge_lst = get_edges();
    //alert(edge_lst);
    if (obj1 == null){
        return;
    }
    var obj2 = JSON.parse(obj1);
```

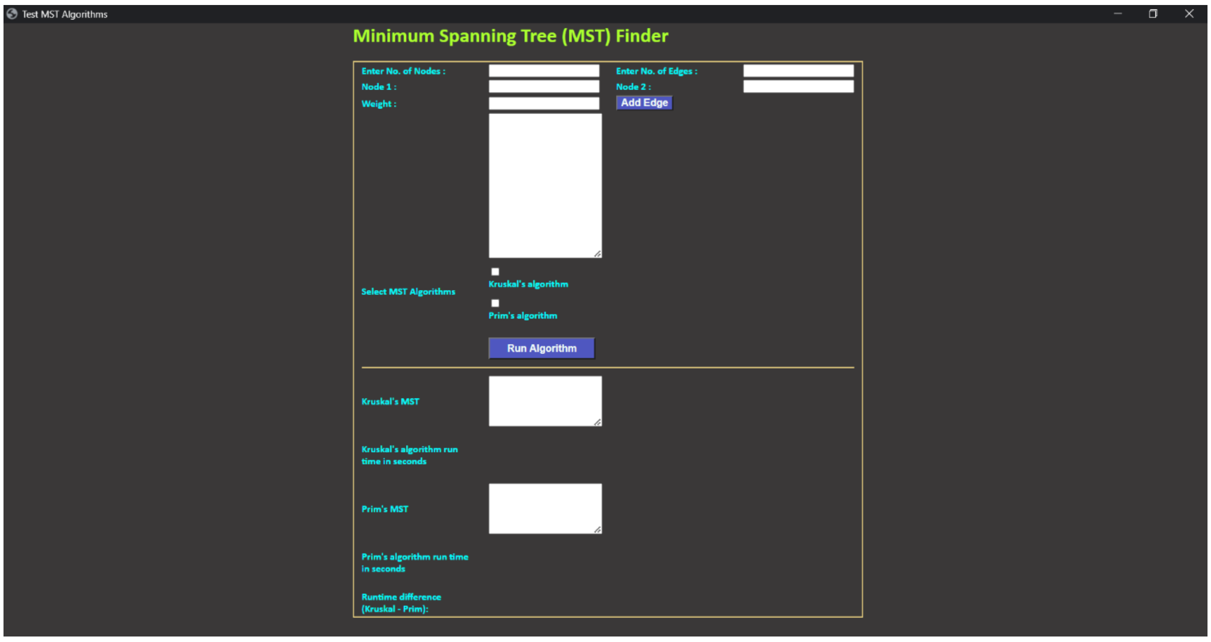
```
    var edge_lst = get_edges();
    //alert(edge_lst);
    if (obj1 == null){
        return;
    }
    var obj2 = JSON.parse(obj1);
    if (obj2.k_algo=="N" && obj2.p_algo=="N"){
        alert("Please select at least one algori
        return;
    }

    let n = await eel.run_MST_algo(obj1,edge_lst
    var obj = JSON.parse(n);
    document.querySelector(".K_running_time").in
    var x = obj.K_MST;
    var txt="";
    for (var i = 0; i < x.length; i++) {
        txt += x[i];
        if ((i+1) != x.length){
            txt += "\n";
        }
    }
    document.getElementById("kruskal_output").va

    document.querySelector(".P_running_time").in
    var x = obj.P_MST;
    var txt="";
    for (var i = 0; i < x.length; i++) {
        txt += x[i];
        if ((i+1) != x.length){
            txt += "\n";
        }
    }
    document.getElementById("prims_output").valu
    document.querySelector(".RT_diff").innerHTML
})
}


//This function is called for adding the edge tc
var x="";
```

```

//This function is called for adding the edge tc
var x="";

function addEdge(){

    x += document.getElementById("node1").value
    document.getElementById("edges").value = x;
    document.getElementById("node1").value = "";
    document.getElementById("node2").value = "";
    document.getElementById("weight").value = ""
}
/*
document.querySelector("button").onclick = async
    // Call python's random_python function
    eel.kruskal_MST()(function(n){
      // Update the div with a random number ret

      var obj = JSON.parse(n);
      document.querySelector(".running_time").in
      var x = obj.MST;
      var txt="";
      for (var i = 0; i < x.length; i++) {
          txt += x[i] + "\n"
      }
      document.getElementById("output").value =
    })
  }
*/
```

(Figure 6)

The ‘script.js’ file contains multiple functions which are used in performing front end operations. The first function ‘get_edges’ returns array by splitting the given string. The edges in the text area are separated by “\n” character and get edges returns the array of edges. The second function ‘get_Json’ creates a JSON object and returns it to the calling function. The third function ‘run’ is an asynchronous function that is called from index.html file when “Run Algorithm” button is clicked. It calls the python function exposed by eel and displays the output returned by Python function. And finally, ‘addEdge’ function is defined for adding the edge to text area input field that is displayed on the GUI.

WORKING OF THE GUI



(Figure 7)

The above image represents the GUI which is user-interactive and very easy to use. Multiple textboxes are used for entering the number of nodes, number of edges and the weights between each edge. It represents two checkboxes for selecting either one or both the MST algorithms to determine the MST.

The experimental results shown below represents the minimum spanning tree (MST) computed by the two algorithms for the user defined graph along with their individual running time in seconds and the difference between their running times.

LIMITATIONS

The program only accepts nodes or vertices as integers and the vertices must start from 0.

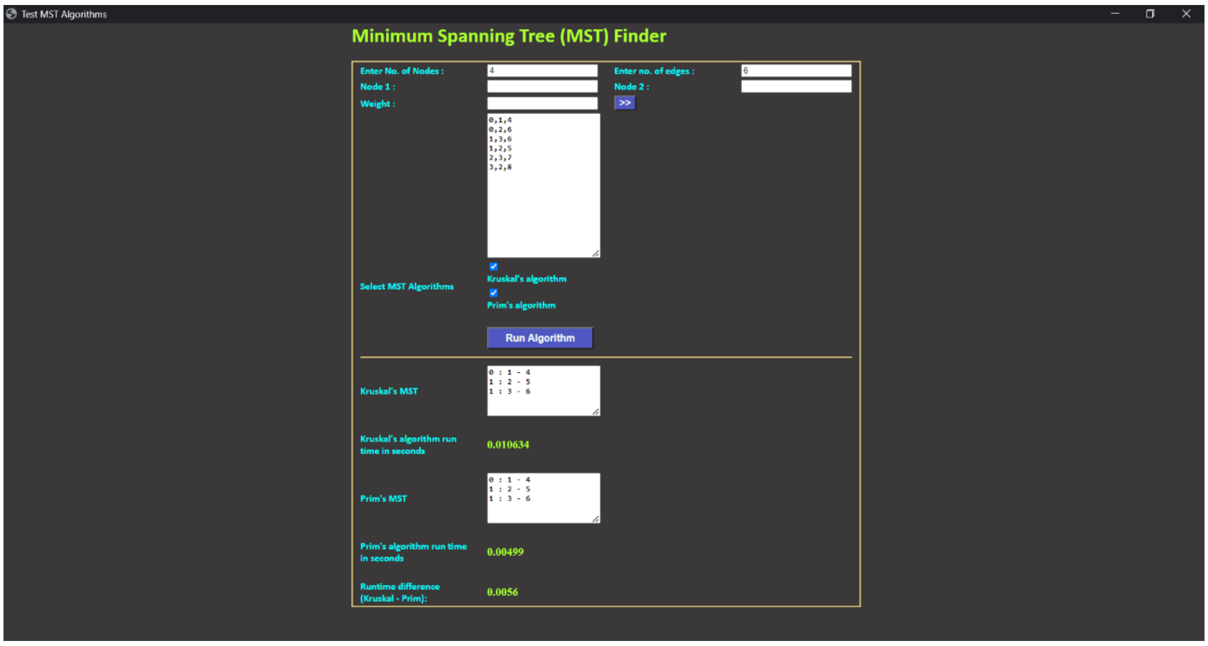
EXPERIMENTAL RESULTS

MST Algorithms	Spanning Tree Example 1	Spanning Tree Example 2	Spanning Tree Example 3	Spanning Tree Example 4
No. of Nodes and Edges	Nodes – 4 Edges - 6	Nodes – 5 Edges - 7	Nodes – 6 Edges - 8	Nodes – 9 Edges - 14

Weighted edges of the graph	0 : 1 -> 4 0 : 2 -> 6 1 : 3 -> 6 1 : 2 -> 5 2 : 3 -> 7 3 : 2 -> 8	0 : 1 -> 15 0 : 2 -> 20 1 : 2 -> 13 1 : 3 -> 5 2 : 4 -> 6 2 : 3 -> 10 3 : 4 -> 8	0 : 1 -> 4 0 : 2 -> 4 1 : 2 -> 2 2 : 3 -> 3 2 : 4 -> 2 2 : 5 -> 4 3 : 5 -> 3 4 : 5 -> 3	0 : 1 -> 4 0 : 2 -> 7 1 : 2 -> 11 1 : 3 -> 9 1 : 5 -> 20 2 : 5 -> 1 3 : 6 -> 6 3 : 4 -> 2 4 : 6 -> 10 4 : 8 -> 15 4 : 7 -> 5 4 : 5- > 1 5 : 7 -> 3 6 : 8 -> 5 7 : 8 -> 12
Kruskal’s Running Time (in sec)	0.010634	0.00797	0.008974	0.00800
Prim’s Running time (in sec)	0.00499	0.00299	0.002996	0.00501
Difference in running time in sec. (Kruskal’s – Prim’s)	0.0056	0.0050	0.0060	0.0030

As from the above table, we can see that for different spanning tree examples defined by the user we get different run time. The run time of Kruskal’s algorithm increases as the no. of nodes and no. of edges increases. But in case of Prim’s algorithm the change in run time with different size of graph is very minimal. Also, the run time of Kruskal’s algorithm is more than the Prim’s algorithm and it can be seen from the run time difference recorded for each of the graph inputs.

OUTPUT



Enter No. of Nodes : 4
Enter no. of edges : 6
Node 1 :
Node 2 :
Weight :

```
0,1,4
0,2,6
1,3,6
1,2,5
2,3,7
3,2,8
```

Select MST Algorithms
[✓] Kruskal's algorithm
[✓] Prim's algorithm

Kruskal's MST

```
0 : 1 - 4
1 : 2 - 5
1 : 3 - 6
```

Kruskal's algorithm run time in seconds 0.010634

Prim's MST

```
0 : 1 - 4
1 : 2 - 5
1 : 3 - 6
```

Prim's algorithm run time in seconds 0.00499

Runtime difference (Kruskal - Prim): 0.0056

(Figure 8)

The above figure represents the output for user defined spanning tree example 1 which has 4 nodes and 6 edges. As from the figure, after entering the number of nodes, number of edges and the weights of each edge, the two algorithms are selected and after clicking the “Run Algorithm” button both Prim’s and Kruskal’s are executed to find the minimum spanning tree (MST). It displays the MST below along with execution time of both the algorithms and the differences in their running times.

CONCLUSION

Both Kruskal’s and Prim’s algorithm finds the Minimum Spanning Tree and follow the Greedy approach of problem-solving, but there are few major differences between them.

Kruskal’s Algorithm	Prim’s Algorithm
It starts to build the Minimum Spanning Tree from the vertex carrying minimum weight in the graph.	It starts to build the Minimum Spanning Tree from any vertex in the graph.
It traverses one node only once.	It traverses one node more than one time to get the minimum distance.
Kruskal’s algorithm’s time complexity is O (E log V), V being the number of vertices.	Prim’s algorithm has a time complexity of $O(V^2)$, V being the number of vertices and can be improved up to O

	(E log V) using Fibonacci heaps.
Kruskal’s algorithm can generate forest (disconnected components) at any instant as well as it can work on disconnected components	Prim’s algorithm gives connected component as well as it works only on connected graph.
Kruskal’s algorithm runs faster in sparse graphs.	Prim’s algorithm runs faster in dense graphs.

The experimental results also shows that Prim’s algorithm runs faster in case of directional weighted graphs.

REFERENCES

https://www.techiedelight.com/graph1-implementation-python/

https://pythonwife.com/kruskal-and-prims-algorithm-in-python/

https://www.geeksforgeeks.org/difference-between-prims-and-kruskals-algorithm-for-mst/

https://stackabuse.com/graphs-in-python-minimum-spanning-trees-kruskals-algorithm/

https://www.geeksforgeeks.org/create-html-user-interface-using-eel-in-python/