# Explaining Algorithms
## Task 1

```
for i in tqdm(range(len(data['data']))): # Iterating through all the data of json file
    time = int(data['data'][i]['time']) / 1000 # Converting miliseconds to seconds
    date = datetime.utcfromtimestamp(time).strftime('%Y-%m-%d') # Converting date from seconds

    if not os.path.exists(date): # Make directory by the name of the date to export csv for each days
        path = os.path.join("./", date)
        os.mkdir(path)

    for j in range(len(data['data'][i]["Responses"])): # Itererting to all responses (Ultimate goal is
to find line_id)

        try: # There are few missing values. Ignoring them
            for k in range(len(data['data'][i]["Responses"][j])):
                lines = data['data'][i]["Responses"][j]['lines'][k]  # Iterating through all the lines
                dataframe = pd.DataFrame(lines) # Convert line response dict to dataframe
                csv_join = lines["lineId"] + ".csv" # # To save the csv file name by line_id
                file_name = os.path.join(date, str(csv_join))
                dataframe.to_csv(file_name, index=False) # Saving the dataframe

        except:
            pass
```

1. Initially, it Iterates through all the **data** of the JSON file.
2. Then convert time to **date** from the data loop.
3. Make a **folder** by the name of the date (to segregate line id)

As we already got the date, now then we need to segregate the line_id. To segregate line id:

4. After that, it iterates through all the **responses** inside the data loop.
5. Then again, it iterates through all the **lines** inside the response loop

Finally, we can access the line_id. So, we'll save the **line data** as a dataframe by the name of the line_id.

As we calculate the date inside the first loop, line_id will be saved as CSV inside the corresponding date.

# Task 2

**Identifying missing vehicle ids:**

In this problem, we have to find out any missing vehicles that exist in stop_times.txt but don't exist in the JSON files.

After carefully finding the pattern of these two files. I've noticed that, in the JSON file, the **"pointId"** is the last stop. And in stop_times.txt files, there is **stop_id**. So, we can match between them and find which vehicles are missing.

To solve this problem:

1. I tried to find out all the pointId (append them in pointIds list) in two JSON files.
2. Load stop_times.txt as dataframe.
3. Find out which stop_id is not found inside pointIds list. Apparently, they are the missing vehicles.