Utilization of algorithms, Dynamic programming, Optimization

Firstly, we are importing pandas and numpy module.Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool. Numpy refers to numeric python.

In [1]:

```
import pandas as pd
import numpy as np
```

Reading the dataset

We are reading the csv file and storing it as dataframe

In [2]:

```
df=pd.read_csv("Dataset - Sheet3.csv")
```

Data pre-processing

head() method - Returns the first 5 rows of the dataframe.

In [3]:

```
df.head()
```

Out[3]:

	Date	Day	Time	Empty level in cm(Total size=100cm)
0	03/01/21	Monday	7:00 AM	80
1	03/01/21	Monday	9:00 AM	60
2	03/01/21	Monday	11:00 AM	60
3	03/01/21	Monday	1:00 PM	50
4	03/01/21	Monday	3:00 PM	90

tail() method - Returns the last 5 rows of the dataframe.

In [4]:

```
df.tail()
```

Out[4]:

	Date	Day	Time	Empty level in cm(Total size=100cm)
239	03/28/21	Sunday	3:00 PM	60
240	03/28/21	Sunday	4:00 PM	40
241	03/28/21	Sunday	5:00 PM	35
242	03/28/21	Sunday	6:00 PM	100
243	03/28/21	Sunday	7:00 PM	80

Columns attribute return the column labels of the given Dataframe.

In [5]:

```
df.columns
```

Out[5]:

```
Index(['Date', 'Day', 'Time', 'Empty level in cm(Total size=100cm)'], dtype='object')
```

dtypes attributes returns a Series with the data type of each column.

In [6]:

```
df.dtypes
```

Out[6]:

```
Date                                 object
Day                                  object
Time                                 object
Empty level in cm(Total size=100cm)    int64
dtype: object
```

describe() method is used to view some basic statistical details like percentile, mean, std etc. of a data frame

In [7]:

```
df.describe()
```

Out[7]:

	Empty level in cm(Total size=100cm)

count 244.000000

mean 67.860656

std 24.060731

min 15.000000

25% 50.000000

50% 70.000000

75% 90.000000

max 100.000000

info() function is used to print a concise summary of a DataFrame. This method prints information about a DataFrame including the index dtype and column dtypes, non-null values and memory usage.

In [8]:

```
df.info()
```

```
RangeIndex: 244 entries, 0 to 243
Data columns (total 4 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Date                            244 non-null    object
 1   Day                             244 non-null    object
 2   Time                            244 non-null    object
 3   Empty level in cm(Total size=100cm)  244 non-null    int64
dtypes: int64(1), object(3)
memory usage: 7.8+ KB
```

Using the below snippet code we are converting the datatypes of the columns(object-->int).

In [9]:

```
from sklearn.preprocessing import LabelEncoder
category= ['Date','Day','Time']
encoder= LabelEncoder()
```

```
for i in category:
    df[i] = encoder.fit_transform(df[i])
df.dtypes
```

Out[9]:

```
Date                                int32
Day                                 int32
Time                                int32
Empty level in cm(Total size=100cm)    int64
dtype: object
```

corr() is used to find the pairwise correlation of all columns in the dataframe. Any nan values are automatically excluded. For any non-numeric data type columns in the dataframe it is ignored. Note: The correlation of a variable with itself is 1.

In [10]:

```
df.corr()
```

Out[10]:

	Date	Day	Time	Empty level in cm(Total size=100cm)
Date	1.000000	-0.056907	-0.020373	-0.032041
Day	-0.056907	1.000000	0.018148	-0.104016
Time	-0.020373	0.018148	1.000000	0.200872
Empty level in cm(Total size=100cm)	-0.032041	-0.104016	0.200872	1.000000

From the above result we can conclude that all variables are independent of each other. So the idea of linear regression also fails

*values attribute returns the numpy representation of the given DataFrame. 'Date','Day','Time' are taken into X and these independent variables

In [11]:

```
X=df[['Date','Day','Time']].values
X[0:5]    #we are converting dataframe into numpy arrays.
```

Out[11]:

```
array([[ 0,  1,  9],
       [ 0,  1, 12],
       [ 0,  1,  1],
       [ 0,  1,  3],
       [ 0,  1,  5]])
```

*values attribute returns the numpy representation of the given DataFrame. 'Empty level in cm(Total size=100cm)' ar