# PART 1: SUMMARY REPORT

**Xingchen (Estella) Ye**
Department of Computer Science
Columbia University
New York, NY 10027, USA
xy2527@columbia.edu

## 1 PROBLEM FORMULATION

The goal of this problem is to calculate the number of hours that are considered as a given peak type, for a given ISO and period.

First, by the problem prompt, the time intervals for different peak types are defined as follows:

- **onpeak:** HE7 (6:00 AM) to HE22 (10:00 PM) on non-holiday weekdays.
- **offpeak:** HE1-HE6 (0:00 AM to 6:00 AM) and HE23-HE24 (11:00 PM to 12:00 PM) on non-holiday weekdays.
- **flat:** Every hour of every day (HE1-HE24).
- **2x16H:** HE7 (6:00 AM) to HE22 (10:00 PM) on weekends and holidays.
- **7x8:** HE1-HE6 (0:00 AM to 6:00 AM) and HE23-HE24 (11:00 PM to 12:00 PM) every day.

For this problem, it is important to handle Daylight Saving Time (DST) changes. At the start of DST, the clock is set forward one hour, resulting the disappearance of the interval 2:00 AM to 3:00 AM; at the end of DST, the clock is set backward for one hour, resulting a repetitive 1:00 AM to 2:00 AM.

If the given period contains the start or the end of DST, DST would impact on the hour count of offpeak, flat, and 7x8.

## 2 METHODOLOGY

1. **Check the Period Type:**
   - Identify the type of the period parameter. The period can be daily, monthly, quarterly, or annually, represented by the different formats:
     - Daily: "YYYY-MM-DD"
     - Monthly: "YYYYMon"
     - Quarterly: "YYYYQn"
     - Annually: "YYYYA"
   - Determine the year and the start and end dates based on the format of the period: For example, a daily period (e.g., "2019-05-01"), the start date is "2019-05-01 00:00:00" and the end date is "2019-05-02 00:00:00".
   - Generate an array of timestamps for each hour between the start and end dates (exclusive). For example, if the period is "2019May", the array of timestamps will be: "2019-05-01 00:00:00", "2019-05-01 01:00:00", ..., "2019-05-31 23:00:00".
2. **Handle Daylight Saving Time (DST):**
   - Use an auxiliary function `is_dst` to determine if there are DST adjustments between the start and end dates.
   - Consider DST for all ISOs except MISO. If the given time period includes the start or end date of DST, there is an adjustment of either -1 or +1 hour to the off-peak, flat, and 7x8 peak types.

- Adjust the total count of hours by 0, -1, or +1 hour as needed.

3. **Count Over All Hours:**
   - Generate a range of hours between the start and end dates.
   - For each hourly timestamp in this range, check if it is a valid hour for the given peak type using the `is_valid_hour` function, and then verify if the hour falls within the peak type's interval.
   - Count the valid hours to determine the total number of hours for the specified peak type and period.

## 3 Code Implementation

1. **Auxiliary Functions:**
   - `is_dst(start_time, end_time, timezone='America/New_York')`
     - Determine if there is a DST change between the start and end dates in the specified timezone.
     - Return -1 if the period is shorter by one hour due to DST, +1 if longer, and 0 if no change.
   - `is_valid_hour(peak_name, curr_time, holidays_arr, valid_times)`
     - Check if a given hour is valid for the specified peak type.
     - Consider the day of the week, holidays, and valid time intervals to determine if the hour should be counted.
     - If the peak type is 'onpeak' or 'offpeak', the hour is valid only on non-holiday weekdays (Monday to Friday).
     - If the peak type is '2x16H', the hour is valid only on weekends (Saturday and Sunday) or holidays.
     - Then, check if the current hour falls within the defined valid time intervals for the peak type.
2. **Main Function:**
   - `get_hours(iso, peak_type, period)`
     - Define the peak hours for different peak types.
     - Parse the period to determine the start and end dates.
     - Adjust for DST if applicable, considering only 'offpeak', 'flat', and '7x8' peak types.
     - Iterate over all hours within the date range, checking if each hour is valid based on the peak type.
     - Return a dictionary with the ISO, peak type, start date, end date, and the total number of valid hours.

## 4 Results

To demonstrate the functionality of the `get_hours` function, we can look at several examples with different ISOs, peak types, and periods.

### 4.1 Example 1: ERCOT Onpeak Hours for May 2019

Running the function:

```
get_hours("ERCOT", "onpeak", "2019May")
```

Returns:

```
{'iso': 'ERCOT',
'peak_type': 'ONPEAK',
```

```
'startdate': '2019-05-01',
'enddate': '2019-05-31',
'num_hours': 352}
```

The result is consistent with the sample run provided in the prompt.

### 4.2 Example 2: Various Peak Types for ERCOT in May 2019

Running the function:

```
example_onpeak = get_hours("ERCOT", "onpeak", "2019May")
example_offpeak = get_hours("ERCOT", "offpeak", "2019May")
example_flat = get_hours("ERCOT", "flat", "2019May")
example_216 = get_hours("ERCOT", "2x16H", "2019May")
example_78 = get_hours("ERCOT", "7x8", "2019May")
```

The returned dictionaries only differ in `num_hours`, 352 for onpeak, 176 for offpeak, 744 for flat, 144 for 2x16h, 248 for 7x8.

For the flat peak type, the total number of hours is calculated as 744 hours, which corresponds to 31 days multiplied by 24 hours per day. For the onpeak and offpeak types, there are 22 non-holiday weekdays in May 2019. The onpeak hours are calculated as 352 hours, which is 22 days multiplied by 16 hours per day (from 6:00 AM to 10:00 PM), while the offpeak hours are calculated as 176 hours, which is 22 days multiplied by the remaining weekday hours. For the 2x16H peak type, the total is 144 hours, which is 8 weekend days in May 2019 multiplied by 16 hours per day. Lastly, for the 7x8 peak type, the total is 248 hours, which is 31 days multiplied by 8 hours per day.

### 4.3 Example 3: DST Adjustment for March 2023

```
example_1 = get_hours("MISO", "flat", "2023Mar")
example_2 = get_hours("ERCOT", "flat", "2023Mar")
```

The returned dictionaries differ in `num_hours` by one hour, with ERCOT having one hour less than MISO, which is consistent with the DST adjustment to ERCOT.