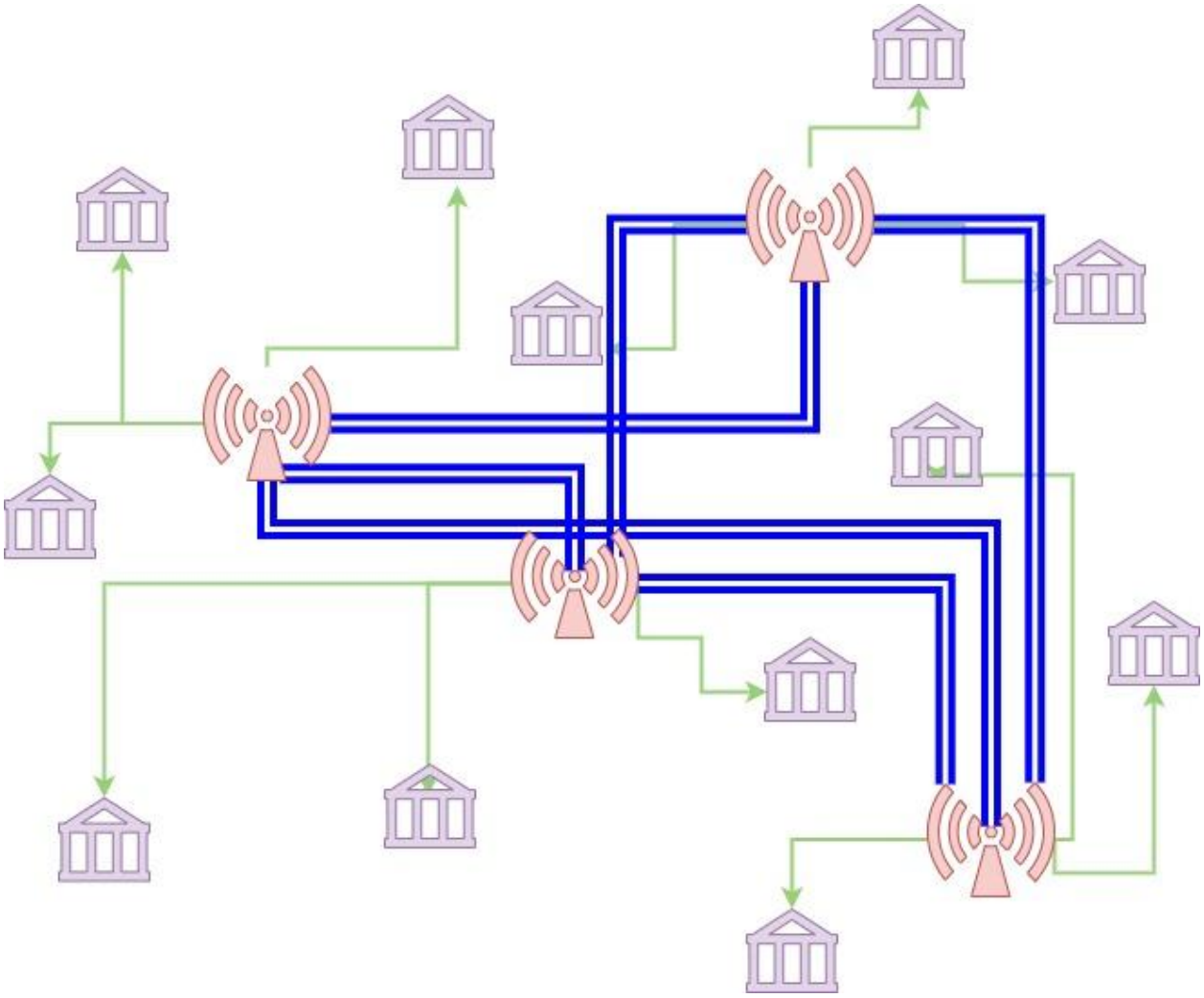
Power Grid System

Introduction

For this project implementation we have taken a scenario of a power grid system and electricity distribution to the houses attached with these power grids. Every power grid has power supply nodes/propagators that are providing power to different areas and localities more precisely to different houses.

Let's assume that there is a list of propagators from P_1 to P_n. Each of these propagators are further connected with each other and different houses to provide the power. As we know that every power grid has its own capacity and providing power more than its capacity is not possible. Similarly each of the propagators comes with their own limitation and should not exhaust the assigned power limit. Each of the propagators may be connected with the H_1 to H_k to provide the power supply.

Let’s look at the block diagram of the proposed setup below. Each of these houses are connected with only one of the propagators in the power grid. Let’s look at some of the more insights on the diagram.



- The propagator and house are connected using a directed edge with some weight. This weight represents the supply provided to the house based on the requirement.
- The propagators are connected with each other as well but the weight will be by default 1. Please understand that each of the propagators are connected for obvious technical reasons. This connection will also be directed.
- Any new edge or house will be assigned to a propagator only if the demanded power request is available.
- All the propagators are locally stable and overall demand for a single propagator is always lesser than its capacity.

Housekeeping points

- This is a minimal example and may not follow some standard practices
- The focus is on the main flow, with minimal error handling. Errors in validation logic should be handled appropriately though.
- Some of the files will be hard coded and the path provided is relative.

Program Organization

This section will contain the information related to the different files available in the project.

1. C02_P02_Power_System.py:
    a. **House:** This class holds information related to house objects. This includes house No, required power and assigned propagator number. You can also assign different propagators based on the request.
    b. **Propagator:** This class holds the information related to propagator and the houses connected to a propagator. The goal of this class is to ensure the operation related to propagators such as creation, connecting to the houses, checking if a house is already connected with the propagator and removing the house from the list of propagators.
        i. **connected_house_info:** This method returns the list of all the houses connected to the propagator.
        ii. **is_house_present:** Checks if the given house_no is connected to the propagator or not.
        iii. **add_house:** Adding the house in list of houses connected with the propagator
        iv. **remove_house:** removing the house from the list of houses connected with the propagator.

    c. **PowerGrid:** This class manages a dictionary that holds the information of each propagator and has methods to add, remove houses from the propagators. As we know that each propagator has a certain list of houses that it is connected with, so all the add or remove house operation calls are also included in the PowerGrid class.
        i. **add_propagator:** This method is supposed to add the asked propagator into the adjacency list format dictionary. This implementation is similar to adjacency list implementation provided in the Live session material of Graph.
        ii. **remove_propagator:** This method will simply remove the propagator from the adjacency list and all the houses associated with it.
        iii. **add_house:** This method will add the house in a list of particular propagators.
        iv. **remove_house:** This method will remove the given house from a list of particular propagators.

d. **create_power_grid:** This method will simply load the data from app.csv into the assigned object of the classes and data structures.

2. ***app.csv:*** This file contains data that will help you in creating the propagators, houses and the requirements provided by houses. Please do not make any changes into this file. This file does not require any changes.
3. ***output.txt:*** This is a sample output file. Once your code is in working state and all the methods are implemented properly, your output should appear like this.

Problem Statement

The program structure is already set and there are specific methods that you are expected to implement. Please also read the comments in the code, especially in the methods to be implemented. You can modify C02-P02-Power-Grid-System.py to add further demonstration calls.

1. ***is_house_present:*** This method checks if the given house is currently connected with the given propagator. If house is present then return the index else return -1
2. **add_house:** Adding the house to the assigned propagator. Before adding we need to check if the connection is already there. If it is there then print connections already exist else add the house with the propagator.
3. **remove_house:** Removing the house from the assigned propagator. Before removing we need to check if the connection is already there or not. If it is there then perform deletion and return True else return False.
4. **add_propagator:** Adding the propagator in the power grid. Check if the propagator is already present in the dictionary, if it is not present then add the propagator. Return True after successful operation else return False
5. **remove_propagator:** Removing the propagator from the power grid. Check if the propagator is already present in the dictionary, if it is not present then removal is not possible else delete the propagator along with the houses in it.

Evaluation Rubric

Total Project Points: **20**

- Basic compilation without errors (**10%**) : **2 Points**
- Correctness:
  - Problem statement - 1 (**30%**) : **6 Points**
  - Problem statement - 2 (**30%**) : **6 Points**

- Problem statement - 3 (**10%**) : **2 Points**
- Problem statement - 4 (**10%**) : **2 Points**
- Problem statement - 5 (**10%**) : **2 Points**

Program Instructions

1. Download the zipped folder named **C02-P02-Power-Grid-System.zip**, and unzip it on your local machine. Go into the directory named **C02-P02-Power-Grid-System**

2. Make sure that you have Python 3.6, or higher, installed. At your command prompt, run:
   ```
   $ python --version
   Python 3.7.3
   ```

   If not installed, install the latest available version of Python 3.

3. To run the code in the source code folder, run the following command:
   ```
   $ python3 C02-P02-Power-Grid-System.py (On many Linux/Mac platforms)
                    OR

   $ python C02-P02-Power-Grid-System.py (On Windows/Mac platforms)
   ```

   In any case, one of these two commands should work.

4. Alternatively, you could install a popular Python IDE, such as PyCharm or Visual Studio Code, and select a command to build the project from there.

5. You will be making very frequent changes into the C02-P02-Power-Grid-System.py python files. The idea is to complete both the scripts so that it satisfies all the requirements. Once you have completed the changes in the code and it is executed without any error. Zip the folder as **C02-P02-Power-Grid-System.zip** & now it is ready for submission.