Team Name: Lachesis Assembly Graph Tool
Team Members: Delaney Dow, Marlee Feltham, James Maher, Drew Gross

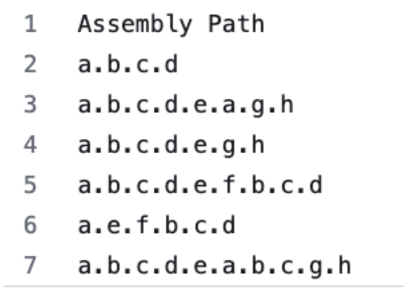
I. Short Project Description

Lachesis is a visual, automated Assembly Graph Tool. Other assembly tools that currently exist, such as BioBricks, rely on users to manually assemble end goal parts, which can often be laborious and time-consuming. As the need for complex assembly of parts increases in the world of synthetic biology, Lachesis provides users with the ability to automate the assembly process, and visualize the assembly output, also with the capability to customize certain parameters. Users are able to enter an end goal part either in text file form or by manual entry, and can then also select whether they would like to find an assembly method that maximized or minimizes cost. They also have the ability to customize a cost function, where both wet lab cost and computational time are taken into consideration. The Lachesis tool then takes all of the user specifications into account and outputs a visual of the worst and/or best-case assembly paths, as well as the cost associated with those paths.

II. Major Software Components

User Interface:

1. End Goal Part Entry
   a. User can enter the goal entry part manually, or by uploading a csv file.
   b. The part must be in “a.b.c.d.” format. An example dataset is provided below, and also available for accessing in the Github Repository.



```
Assembly Path
a.b.c.d
a.b.c.d.e.a.g.h
a.b.c.d.e.g.h
a.b.c.d.e.f.b.c.d
a.e.f.b.c.d
a.b.c.d.e.a.b.c.g.h
```

Figure 1: Example .csv file input

2. Cost Function Specifications
   a. The user can enter the cost value per stages in the assembly graph, which correlates to assembly time, but it can either correspond to an hourly amount or a monetary value of cost per hour.
   b. The user can enter the cost value per steps in the assembly graph, which corresponds to the theoretical wet lab work cost.
3. Assembly Graph Display
   a. The user interface displays assembly graphs for the optimized assembly tree process to the display after the user enters all of their specifications. This is the minimum cost graph.

    b. The user interface displays assembly graphs for the de-optimized assembly tree process to the display after the user enters all of their specifications. This is the maximum cost graph.
    c. The interface also displays the monetary cost associated with the optimized and de-optimized assembly paths.

User Databases:

1. Users may upload a file from any existing database, such as databases that the synthetic biology application, Clotho, stores.
2. Input files must be in .csv files format, but can use any type of database, such as iGem data or databases from one's local computer.

Assembly Graph Generation:

1. The input part from the user interface is sent to the assembly graph generation function.
2. The function decomposes the part recursively into left and right subtrees, then finds the "best" and "worst" graph between each subtree.
    a. *Best & Worst Subgraph Generation*:
        i. A recursive call from the assembly graph generation function that compares the number of steps and stages the current graph contains to the stored "best" or "worst" function, and updates accordingly after that comparison.
        ii. Any ties will arbitrarily choose between the current and stored values.
3. The final output graph(s) is stored into a Hash Map data structure and outputted to the main driver function to handle multiple parts being input at once.

Cost Function Calculation:

1. A cost function is calculated for the maximum and minimum cost based on the steps and stages of the worst cost and best cost graph.
2. The actual monetary values associated with the computational time and wet lab cost are inputted by the users.
3. Results are stored as a monetary value and sent back to the user.

III. Special Instructions for Compiling the Code (ReadMe Duplicate)

1. Install python 3.10 and pip

2. Install Python Imaging Library

```
python3 -m pip install --upgrade pip python3 -m pip install --upgrade Pillow
```

3. Clone Github Repository on your device

```
git clone https://github.com/marleef/EC552-Project.git
```

Minimum of project.cpp and project_gui.py are required to be in the same directory to run the program.

4. Run Python Program using instructions below.

For ease of use, ensure that the file of the part(s) you want to use is in the same directory as the file.

```
python project_gui.py
```