# Convex Hulls

This project implements two methods for generating convex hulls. Each method has its own entry point and can be run as follows:

**Brute Force:**

```
python bruteForce.py [inputFile]
```

**Graham Scan:**

```
python grahamScan.py [inputFile]
```

where [inputFile] is a text file with input for the program. The input file format is discussed in the next section.

There are two additional auxiliary entry points, discussed in the Auxiliary Methods section of this document that are unnecessary to run the main functionality.

## Input File Format

The only required input for the program is the name of the input file. The input file should be a text file. The first line of the file should be an integer specifying how many points are supplied for the input (n). Each of the n following line should have two integers, separated by a space, indicating the x and y coordinates.

## Optional Input

It is important to note that these are ***OPTIONAL*** flags to make the writeup for this assignment easier.

- When the flag `--display=True` is included, both programs will generate a graph using MatPlotLib that displays the points and the generated convex hull.
- When the flag `--runtime=True` is included, both programs will print out the time it takes to run their method to the console before terminating the program.

## Output

The file will write to a new text file with the name of the input file, plus `_out_[method].txt`, in the same location as the input file.

So, for example, if the input is in a file called `tenPoints.txt`, and you run the Graham Scan method on that input, the output will be written to `tenPoints_out_graham_scan.txt`.

## Colinear Points

To stay consistent, both the brute force and Graham Scan methods include colinear points when calculating convex hulls.

# Required Libraries

The following Python libraries are required to run the program. They are also listed in the file `requirements.txt`. Many Python IDEs can detect this file and automatically install the required libraries.

- argparse
- matplotlib
- numpy

# Auxiliary Methods

## Point Generator

The file `pointGenerator.py` generates a specified number of random points with the input file format specified above. It can be run as follows:

```
python pointGenerator.py [output file path] [numPts]
```

Running `python pointGenerator.py random100.txt 100` will produce a file `random100.txt` with 100 random points.

## Plot Runtimes

The file `plotRuntimes.py` is just a simple, hardcoded file that I used to generate the runtime graph in my runtime analysis document.

# References

In addition to lecture notes and the textbook, the following references were helpful in creating my implementation:


- https://www.youtube.com/watch?v=B2AJoQSZf4M&t=110s (https://www.youtube.com/watch?v=B2AJoQSZf4M&t=110s)
- https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html (https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html)
- https://pavcreations.com/clockwise-and-counterclockwise-sorting-of-coordinates/ (https://pavcreations.com/clockwise-and-counterclockwise-sorting-of-coordinates/)