2023 Career Fair Coding Challenge

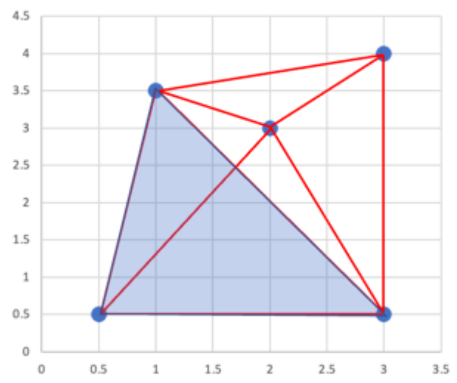
Problem Revealed: Wednesday, 8 March 2023
Submissions due: **Monday, 13 March 2023 at midnight**

Table of Contents

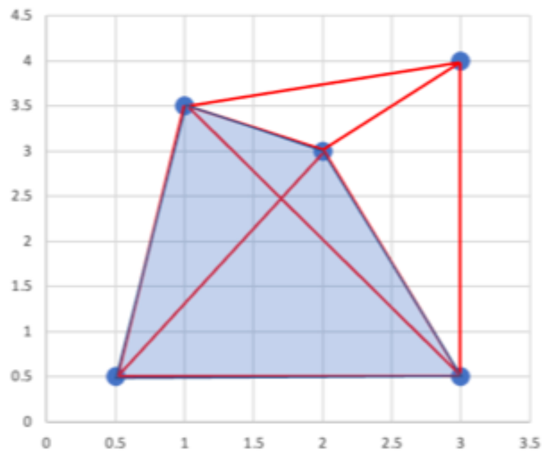
Problem I

You are given a list of points that represent the 2D coordinates of the terrain where a person is located during an earthquake. Each point represents a location that is unsafe during an earthquake, such as next to a multi-storey building or a power grid. The task is to find a safe location where a person can wait out the earthquake. A safe location can be identified as the middle of a large triangle of empty space, where the vertices of the triangle are three points among those given. You are tasked with finding the largest triangle of empty space given the list of unsafe points.

For example: Given the five points (0.5, 0.5), (1, 3.5), (2,3) (3, 0.5) and (3, 4), we can form several triangles of open space, as shown with the red lines in the figure below. The largest such triangle is shaded blue.



Building on this, your next task will be to combine triangles to form the largest convex polygon of empty space, as shown by the shaded area in the figure below.



Further details of each task are given below.

Task 1

Using Java or Python3, define a program FindSafeTriangle that reads in a list of points from a CSV file which is passed as a command line argument. The file will contain a list of points represented as pairs of floating-point numbers (x,y). Your program will write the 3 vertices of the triangle with the largest area to a CSV file named safetriangle.csv. The points should be arranged in increasing order of x-coordinates then y-coordinates.

Your program will be run in the format below

For python3

```
python3 FindSafeTriangle.py points.csv
```

For Java

```
javac FindSafeTriangle.java
```

```
java FindSafeTriangle points.csv
```

The input points.csv file would have the following structure (the points listed here are simply examples):

```
X, Y

0.5, 0.5

1, 3.5

2,3

3, 0.5

3, 4
```

The resulting safetriangle.csv file would have the following structure (the points listed here are simply examples):

```
X, Y

0.5, 0.5

1, 3.5

3, 0.5
```

Task 2

Your next task is to define a program FindSafePolygon that combines adjacent smaller triangles to form a polygon such that the combined shape formed is convex. Convex, in this sense, means that any line drawn between points on the boundary of the polygon falls within the polygon. If there are no such larger polygons, your program will return the triangle with the largest area just as the FindSafeTriangle program. Like the previous program, FindSafePolygon will also read in a list of points from a CSV file which will be passed as a command line argument. The file will contain a list of points represented as pairs of floating-point numbers (x,y). Your program will write the vertices that form the largest such polygon to a CSV file named safepolygon.csv. The points should be arranged in increasing order of x-coordinates then y-coordinates.

Your program will be run in the format below

For python3

```
python3 FindSafePolygon.py points.csv
```

For Java

```
javac FindSafePolygon.java

java FindSafePolygon points.csv
```

The input points.csv file would have the following structure (the points listed here are simply examples):

```
X, Y

0.5, 0.5

1, 3.5

2,3

3, 0.5

3, 4
```

The resulting safepolygon.csv file would have the following structure the points listed here are simply examples):

```
X, Y
0.5, 0.5
1, 3.5
2, 3
3, 0.5
```

Constraints

1. All points have unique coordinates.
2. The list of points has at least 3 entries and at most 10^4 entries.
3. All coordinates are between 0 and 10^6 inclusive.
4. No other points beyond those listed in the input csv file should be considered.

Note that sample input files have been provided for testing purposes, but please feel free to create your own tests. Your code will be evaluated on a different set of input points from those provided.

Problem II

An earthquake has hit a city, and a person is trapped amidst the debris. You have access to the coordinates of several points in the area via a points.csv file which will be passed as a command line argument, and you need to find the smallest triangle that can be formed by three of these points such that the person is inside the triangle, and the area of the triangle does not contain any other points. You need to find the vertices of this triangle and write them to a safesurrounding.csv file. The first point in the input points.csv file represents the location of the person.

Consequently, you need to implement a FindSafeSurrounding program. If there exists an empty triangle containing the person, write the vertices of the smallest empty triangle to the safesurrounding CSV file. The points should be arranged in increasing order of x-coordinates then y-coordinates. If multiple triangles have the same largest area, return any one of them. If no such triangle exists, return the two closest points to the person. Again, if there is a tie you may return any of them.

Constraints:

1. All points have unique coordinates.
2. The list of points has at least 3 entries and at most 10^3 entries.
3. All coordinates are between 0 and 10^6 inclusive.
4. No other points beyond those listed in the input points.csv file should be considered.

Note that sample input files have been provided for testing purposes, but please feel free to create your own tests. Your code will be evaluated on a different set of input points from those provided.

Examples

Example 1

Assuming your points.csv has the following structure:

```
X, Y
2.8, 1.6
0.5, 0.5
1, 3.5
3, 4
2, 3
3, 0.5
```

Your program will be run in the format below

For python3

```
python3 FindSafeSurrounding.py points.csv
```

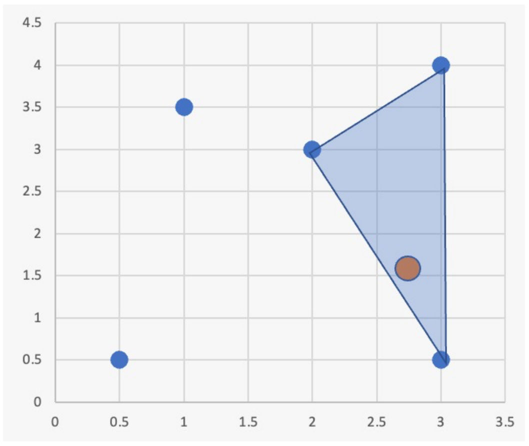
For Java

```
javac FindSafeSurrounding.java
java FindSafeSurrounding points.csv
```

Your program should generate a safesurrounding.csv file with the following format:

```
X, Y
2, 3
3, 0.5
3, 4
```

Below is a visualisation of the points. The orange point represents the location of the person.



Example 2

Assuming your points.csv has the following structure:

```
X, Y
2.5, 2.5
1, 1
2, 2
3, 3
4, 4
5, 5
```

Your program will be run in the format below

For python3

```
python3 FindSafeSurrounding.py points.csv
```

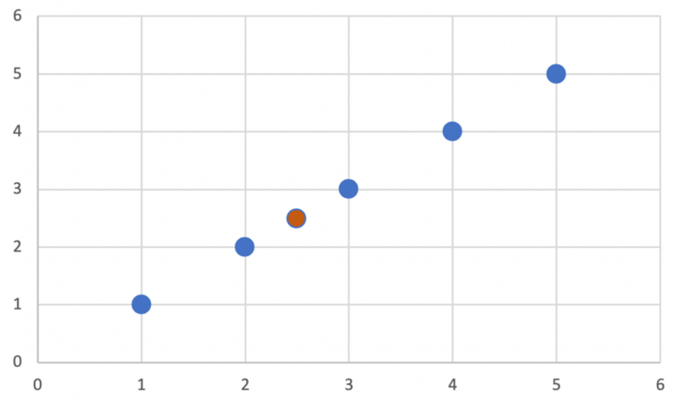
For Java

```
javac FindSafeSurrounding.java
java FindSafeSurrounding points.csv
```

Your program should generate a safesurrounding.csv file with the following format:

```
X, Y
2, 2
3, 3
```

Below is a visualisation of the points. The orange point represents the location of the person.



Explanation:

Here, no triangles could be formed since all the points were in a straight line. The 2 closest points to the person are (2,2) and (3,3).

What to submit

You should submit:

1. Your code and data files
2. A readme.txt file indicating which problems and tasks your code supports and any additional information needed to run your code.
3. A 1-page document briefly describing your high-level approach, and listing any references (see rules below).

Rules of the Challenge

1. Participation in the competition is by Ashesi students, either as **individuals** or as a **team** made up of two students from the same year group. No collaboration is allowed except between two members of a team. Individuals who are not Ashesi students may not participate. AI or automated agents may not participate or give assistance to participants.
2. In addition to overall winners, there will be winners for each year group, so members of all year groups are encouraged to participate and not feel intimidated by the idea that they may be "competing" against more experienced programmers.
3. Solutions must be coded in Python or Java. Standard built-in classes and libraries can be used (e.g. file reading libraries, the math library in Python and classes in the java.util package in Java). In general, no external/third-party libraries can be used in either language. If planning to use any non-standard packages, it is best to enquire first about whether the package will be allowed.
4. All submitted code must be your own. Code may not be copied from any source.
5. You are allowed to do research if needed. However, you must cite all resources you consult.

Evaluation Criteria

We reserve the right to disqualify submissions that do not follow the specified code structure or do not comply with the submission instructions.

Solutions will be evaluated according to the following criteria. The order below will be used to rank the various submissions.

1. Completion & correctness (the number of questions answered correctly).
2. Efficiency (the time it takes to execute your solution approach). Accommodation will be made for the inherent difference in speed between Java and Python. However, no accommodation will be given for unnecessary I/O, so your program should output only the information that is necessary.
3. Code formatting and structure: code should be modular, well-structured, well-formatted, and well-commented.
4. Clarity of 1-page documentation of approach