# CSCI 270 Fall 2019 Programming Assignment 2

Instructor: Joseph Bebel

**Due: Monday, November 18th, 2019, 4:59 pm**

## Minimum Area Convex Hexagons

In topics such as statistical clustering and pattern recognition, a very common problem is finding a small set of data points that form a certain mathematical pattern or structure. In geometric situations, this is often represented by *convex* shapes.

A convex hexagon is a 6-sided shape (6 points connected by straight lines) such that all interior angles are less than 180 degrees. Write an algorithm that on input $n$ points $P$ in the 2-D plane, finds the minimum area of a convex hexagon with all 6 vertices in $P$ in time $O(n^4)$.

- You should use dynamic programming
- This problem only requires two geometric subroutines: computing if a point is on the left or right side of a line, and computing the area of a triangle. You may use the sample formulas below to compute both
- The coordinates of every point $p_i = (x_i, y_i)$ are integers
- You may assume that all $n$ points are distinct
- You may assume that for every pair of points $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ that there are no other points in the input that are colinear with $p_i, p_j$

Given two points $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$, a point $p = (x, y)$ is on the left side of the line $\overline{p_i p_j}$ if and only if:

$$(x - x_i)(y_j - y_i) - (y - y_i)(x_j - x_i) > 0$$

Since we know that $p$ does not lie on the line $\overline{p_i p_j}$ (by the assumption that no other points are colinear with $p_i$ and $p_j$) the point $p$ is either on the left of $\overline{p_i p_j}$ or on the right of $\overline{p_i p_j}$.

The other formula that is useful is to compute the area of the triangle formed by points $p_i = (x_i, y_i)$, $p_j = (x_j, y_j)$, $p_k = (x_k, y_k)$:

$$\frac{|x_i y_j + x_j y_k + x_k y_i - x_j y_i - x_k y_j - x_i y_k|}{2}$$

## 0.1 I/O Format

### 0.1.1 Input

Your program should read from **standard input**. The first line contains an integer $n$, the number of points on the plane. Each of the next $n$ lines contains two integers $x_i, y_i$ separated by space.

### 0.1.2 Output

Output a single number to **standard output** - the minimum area of a convex hexagon with vertices from the given points. The output is considered correct if within an absolute error of $10^{-4}$.

(Actually, the mathematically precise answer can only have .5 or .0 as fractional part. You can use floating-point numbers, and if necessary, you can round your final answer to 1 decimal place. Alternatively you may compute everything in integers, storing twice the actual area at every step, and divide by 2 at the very end.)

### 0.1.3 Sample I/O

Please see `resource/asnlib/publicdata/...` on *Vocareum*.

## 0.2 Data Range

- For all test cases, $-10^4 \leq x_i, y_i \leq 10^4$
- For 50% of full credit, pass all test cases with $n \leq 15$. (Expected complexity: $O(n^6)$)
- For 100% of full credit, pass all test cases with $n \leq 50$. (Expected complexity: $O(n^4)$)
- For an optional chocolate prize, pass all test cases, $n \leq 100$. (Expected complexity: $O(n^3)$). Alternative prize available if chocolate is not desired.

## 0.3 Implementation Details

You should submit a **single** source file containing the implementation of your solution via *Vocareum*, in one of the following programming languages:

**C** Filename should be `assignment2.c`. Compile flags: `-lm -std=c11`.

**C++** Filename should be `assignment2.cpp`. Compile flags: `-std=c++11`.

**Java** Filename should be `assignment2.java` and the package-private class containing your `main` method should be named `Solution`. You should not declare any package.

**Python** Filename should be `assignment2.py`. The environment is `pypy, Python 3.6.1`.

The performance of your program is evaluated based on your actual run time. The time limit is **1 second** for C/C++/Java and **2 seconds** for Python.

## 0.4 Grading

We will grade based on the performance and correctness of your algorithm on a set of sample and hidden test cases.

You will receive points of a test case if and only if your solution terminates within the time limit and your output is correct. If your algorithm is not correct (fails on a test case) or too slow, then you will get partial credit depending on what test cases your algorithm does pass.