**Hausdorff v.1.0**

**Matvei Slavenko**

Charles University in Prague

Faculty of Mathematics and Physics

Obecná matematika, the second semester

NMIN102 Programming II

Winter semester, school year 2016/2017

# General Information

This programme calculates the Hausdorff distance between two convex polygons; the vectors, on which the distance is reached. It also determines whether the current mutual position of the polygons is optimal or not.

The programme has a simple command-line user interface. The user can set the input and output streams at his/her convenience: the programme could both use files and standard input-output system. The list of commands and their detailed descriptions are available in the internal help module of the programme.

# Data Structures Used

The programme uses broadly uses the Linked List data structure. In the programme, all polygons are represented as a sequence of their vertices, stored in linked lists.

The data structure’s implementation and the procedures implemented allow turning each linked list to a ring buffer or a priority queue.

# General Ideas of Algorithm

These are the main logical steps of the algorithm:

## Preparation of the external resources

The programme prepares the external resources: assigns the files on the disk to the proper variables and opens them.

The procedure prepareSources(inPath1, inPath2, outPath: String) is responsible for this.

## Parsing of the polygons

The programme one by one reads tokens from the input streams and constructs the internal representations of the polygons. The procedure readPolygon(target: pLinkedList; var source: Text) is responsible for it.

## Calculation of the Hausdorff distance

The programme calculates the vectors, on which the Hausdorff distance is reached. Knowing these vectors, it also calculates the Hausdorff distance.

The procedure hausdorfDistanceVectors(target, pol1, pol2: pLinkedList) and the function vectorLength(a: Vector) are responsible for it.

## Sorting of the Hausdorff distance vectors

On this stage, the programme sorts the vectors calculated on the previous stage. The vectors are sorted by the angle between the OX-axis and the vector in the positive direction. The order is non-descending. This stage allows to create a convex polygon out of these vectors.

Procedure sortByAngle(target, source: pLinkedList) is responsible for actions of this step.

## Optimality test

Using the sorted list of the vectors got on the step 4, the programme tests the optimality of the current mutual position of the polygons. It is optimal if and only if the zero vector is contained in the internal area of the polygon, created by the vectors from the step 3. (See, for example, [1])

The procedure isOptimal(distVecs: pLinkedList) is responsible for it.

## Closing of the external resources

All external resources are closed and freed on this staged. The procedure closeSources() ensures these actions.

See the documentation included to the source code for detailed information about each step, actual implementation and other details. Source code is distributed freely and could be downloaded, for example, from the [GitHub repository](https://github.com/slavenkof/HausdorffPascal) (https://github.com/slavenkof/HausdorffPascal).

# Input Data Format

The single number on the first line specifies the number of points in the polygon. Each of the following lines contains two numbers with an x- and y-coordinates of the point separated by one space. Coordinates are real numbers.

Example:

4

1,1 1

-1 1

-1 -1

1 -1

# Examples of Programme’s Work

Input1: in10.txt

4

2 1

-2 1

-2 -1

2 -1

Input2: in11.txt

4

1 2

-1 2

-1 -2

1 -2

Output: out1.txt

## -1.00000; 0.00000

## 1.00000; 0.00000

## -0.00000; 1.00000

## -0.00000; -1.00000

## Length: 1.00000

Position is optimal: TRUE

Input1: in10.txt

4

2 1

-2 1

-2 -1

2 -1

Input2: in20.txt

1

0 0

Output: out2.txt

-2.00000; -1.00000

2.00000; -1.00000

2.00000; 1.00000

-2.00000; 1.00000

Length: 2.23607

Position is optimal: TRUE

Input1: in10.txt

4

1 1

-1 1

-1 -1

1 -1

Input2: in20.txt

1

0 0

Output: out3.txt

-1.00000; -0.00000

1.00000; -0.00000

Length: 1.00000

Position is optimal: TRUE

Input1: in40.txt

3

0 200

86.6025403 50

173.2050807 200

Input2: in41.txt

3

213.3974597 550.0

386.6025403 550.0

300.0 700.0

Output: out4.txt

126.79492; 500.00000

126.79492; 500.00000

Length: 515.82647

Position is optimal: FALSE

Input1: in40.txt

3

0 200

86.6025403 50

173.2050807 200

Input2: in43.txt

3

1.2256532319732116E-5 99.99999291765553

173.20509285653213 99.99999291765553

86.60255255653232 249.99999291765545

Output: out5.txt

43.30128; -25.00001

0.00000; 49.99999

-43.30126; -25.00000

-43.30126; -25.00000

43.30128; -25.00001

-0.00000; 49.99999

Length: 50.00001

Position is optimal: TRUE

# Ideas for Programme’s Improvement

1. To implement the BASH-friendly working mode;
2. To deal with memory leaks;
3. To implement the analytical and/or numeric algorithms of optimization (see [1]);
4. To implement the analytical way of optimization for nonconvex polygons;
5. To implement the graphical output;
6. To implement a more user-friendly interface.

# Literature

# Lakhtin A.S., Ushakov V.N. Minimization of the Hausdorff distance between convex polyhedrons.// Journal of Mathematic Science, New York, Vol.126, №. 6, 2005. 1553-1560 p.