Convex Hull

Generated by Doxygen 1.8.18

Chapter 1

Main Page

1.1 Algorithm

This algorithm deals with finding convex hull over the points given in the input.
How it works
A simple overview of the algorithm:
Step 1: Recursively compute the upper hull
Step 2: Recursively compute the lower hull
Step 3: Combine both the hulls and return the output

Steps to Compile and Run :
1) *cd* into the src directory
2) Run *g++ main.cpp* which generates an executable called *a.out* in the same directory
3) Run the executable using *./a.out* (on linux)
3.1) The executable takes a dataset from command line argument. For example, to use an existing dataset, run *./a.out ../datasets/edge.txt*
3.2) If no command-line argument is given, it takes input from the shell directly (stdin)

Performance of the algorithm is documented in the report

1.1.1 Input

Input is given as follows:

- Input can be given both from file (via command-line args) or stdin
- First line must contain the no of Points to be taken as input by the program.
- Each of next line must contain 2 integers, space seperated denoting the (x, y) coordinates of each point.
- Each coordinate must be of integer type in the range -10^6 to 10^6.
- Number of coordinates must be less than 1 Billion.

1.1.2 Output

- Each line of output gives the coordinates present on the convex hull in clockwise order

---

1.1.3 Author

The algorithm is implemented and documented by **Rikil Gajarla (2017A7PS0202H)**.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Point Class Reference

Structure to efficiently store and compare points.
`#include <Point.h>`

Public Member Functions

- Point ()

  Default constructor for Point class.
- Point (long X, long Y)

  Constructor for Point class.
- bool **operator==** (const Point &p)

Public Attributes

- long x
- long y

4.1.1 Detailed Description

Structure to efficiently store and compare points.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Point()

```
Point::Point (
            long X,
            long Y )  [inline]
```

Constructor for Point class.

Parameters

X	x coordinate of point
Y	y coordinate of point

4.1.3 Member Data Documentation

4.1.3.1 x

`long Point::x`
x coordinate of the point

4.1.3.2 y

`long Point::y`
y coordinate of the point
The documentation for this class was generated from the following file:

- Point.h

4.2 PointList Class Reference

Structure to store a list of Point class objects(points)
`#include <PointList.h>`

Public Member Functions

- PointList ()

  Default constructor for Point class.
- PointList (vector< Point > points)

  vector based constructor for Point class
- void setSentinels (bool value, bool upperHull)

  add or remove sentinel nodes
- auto emplace_back (long a, long b)

  To provide interface to STL vector's emplace_back() function.
- auto push_back (Point &p)

  To provide interface to STL vector's push_back() function.
- auto size ()

  To provide interface to STL vector's size() function.
- auto begin ()

  To provide interface to STL vector's begin() function which returns iterator to the start of container.
- auto end ()

  To provide interface to STL vector's end() function which returns iterator to the end of container.
- auto erase (size_t idx1, size_t idx2)

  erase the points present in the given range of indices
- void print ()

  print the points contained in this PointList object
- void print (int s, int e)

  print the points contained in this PointList in the specified range
- Point & operator[ ] (long long idx)

  To provide interface to STL vector's access operator []. Also considers sentinel nodes if index is <0 or >=size of PointList.

4.2.1 Detailed Description

Structure to store a list of Point class objects(points)
This class gives functionality to store a list of Point objects with additional support of having sentinel nodes which are auto determined by the class based on points present in its list.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 PointList()

```
PointList::PointList (
            vector< Point > points )  [inline]
```

vector based constructor for Point class

Parameters

points	vector of Point objects used to initialize PointList

4.2.3 Member Function Documentation

4.2.3.1 erase()

```
auto PointList::erase (
            size_t idx1,
            size_t idx2 )  [inline]
```

erase the points present in the given range of indices

Parameters

idx1	index from which points must be erased
idx2	index until which all points will be erased (inclusive)

4.2.3.2 operator[]()

```
Point& PointList::operator[] (
            long long idx )  [inline]
```

To provide interface to STL vector's access operator []. Also considers sentinel nodes if index is <0 or $>=$size of PointList.

Parameters

idx	index which is to be accessed.

4.2.3.3 print()

```
void PointList::print (
            int s,
            int e )  [inline]
```

print the points contained in this PointList in the specified range

Parameters

s	starting index to start printing from
e	ending index to end printing at (inclusive)

4.2.3.4 setSentinels()

```
void PointList::setSentinels (
            bool value,
            bool upperHull )  [inline]
```

add or remove sentinel nodes

Parameters

value	boolean value which when set to true, sets the sentinel nodes for this PointList
upperHull	boolean value to be set to true if the current operations are to be performed on upper hull

The documentation for this class was generated from the following file:

- PointList.h

Chapter 5

File Documentation

5.1 ConvexHull.h File Reference

This file contains the main Convex Hull algorithm.

```
#include <vector>
#include <algorithm>
#include "Point.h"
#include "Tools.h"
#include "PointList.h"
```

Functions

- bool xPointComparator (const Point &a, const Point &b)

  comparator function used to compare x coordinates while sorting
- void xAxisSort (PointList &input)

  function to sort given list of points in increasing order of x coordinate
- pair< long long, long long > upperTangent (PointList &lHull, PointList &rHull)

  compute the upper tangent for the given left hull and right hull
- pair< long long, long long > lowerTangent (PointList &lHull, PointList &rHull)

  compute the lower tangent for the given left hull and right hull
- PointList combineHulls (PointList &uHull, PointList &lHull)

  Function to combine the upper hull and the lower hull.
- PointList upperConvexHull (PointList &input, int start, int end)

  Recursive function to compute the upper hull of the left half and right half points then merge them with upper tangent.
- PointList lowerConvexHull (PointList &input, int start, int end)

  Recursive function to compute the lower hull of the left half and right half points then merge them with lower tangent.
- PointList convexHull (PointList &input)

  driver function which computes upper hull, lower hull and then combines them

5.1.1 Detailed Description

This file contains the main Convex Hull algorithm.
This header file (ConvexHull.h) file consists of main convexHull() function along with it's helper functions for calculating upper and lower tangents. This file also depends on these headers: Point.h, Tools.h and PointList.h

5.1.2 Function Documentation

5.1.2.1 combineHulls()

```
PointList combineHulls (
            PointList & uHull,
            PointList & lHull )
```

Function to combine the upper hull and the lower hull.

Parameters

uHull	PointList object with points of upper hull in clockwise order
lHull	PointList object with points of lower hull in clockwise order

Returns

PointList object with the final list of points present on the complete convex hull in clockwise order

5.1.2.2 convexHull()

```
PointList convexHull (
            PointList & input )
```

driver function which computes upper hull, lower hull and then combines them

Parameters

input	input points given by user over which convex hull is computed

Returns

Points present on the convex hull of given points in clockwise order

5.1.2.3 lowerConvexHull()

```
PointList lowerConvexHull (
            PointList & input,
            int start,
            int end )
```

Recursive function to compute the lower hull of the left half and right half points then merge them with lower tangent.

Parameters

input	input points which are sorted w.r.t x coordinate
start	start index of interval over which lower hull is computed
end	ending index of interval over which lower hull is computed

Returns

PointList with all Points present on lower hull of given interval (in clockwise order)

5.1.2.4 lowerTangent()

```
pair<long long, long long> lowerTangent (
            PointList & lHull,
            PointList & rHull )
```

compute the lower tangent for the given left hull and right hull

Parameters

lHull	PointList object contaning the points on the left hull in clockwise order
rHull	PointList object containig the points on the right hull in clockwise order

Returns

a STL pair object with the points which are part of lower tangent between both hulls

5.1.2.5 upperConvexHull()

```
PointList upperConvexHull (
            PointList & input,
            int start,
            int end )
```

Recursive function to compute the upper hull of the left half and right half points then merge them with upper tangent.

Parameters

input	input points which are sorted w.r.t x coordinate
start	start index of interval over which upper hull is computed
end	ending index of interval over which upper hull is computed

Returns

PointList with all Points present on upper hull of given interval (in clockwise order)

5.1.2.6 upperTangent()

```
pair<long long, long long> upperTangent (
            PointList & lHull,
            PointList & rHull )
```

compute the upper tangent for the given left hull and right hull

Parameters

lHull	PointList object contaning the points on the left hull in clockwise order
rHull	PointList object containig the points on the right hull in clockwise order

Returns

a STL pair object with the points which are part of upper tangent between both hulls

5.1.2.7 xAxisSort()

```
void xAxisSort (
            PointList & input )
```

function to sort given list of points in increasing order of x coordinate

Parameters

input	PointList object containing the user input data points

5.1.2.8 xPointComparator()

```
bool xPointComparator (
            const Point & a,
            const Point & b )
```

comparator function used to compare x coordinates while sorting

Parameters

a	first Point object
b	second Point object

Returns

true if a comes before b

5.2 main.cpp File Reference

Main file which takes input, computes convex hull and returns output.

```
#include <chrono>
#include "Tools.h"
#include "ConvexHull.h"
```

Functions

- int **main** (int argc, char ∗argv[ ])

5.2.1 Detailed Description

Main file which takes input, computes convex hull and returns output.
This is the main runner file which uses the convexHull() function from the header ConvexHull.h to compute the convex hull. This file also uses std::chrono from the chrono c++ header to estimate the duration of the convex hull computation

5.3 Point.h File Reference

This file contains the implementation of the Point class.

```
#include <iostream>
#include <vector>
```

Classes

- class Point

  Structure to efficiently store and compare points.

Functions

- std::ostream & **operator**<< (std::ostream &os, const Point &p)

Variables

- std::string **red** = "\033[31;1m"
- std::string **green** = "\033[32;1m"
- std::string **reset** = "\033[0m"

5.3.1 Detailed Description

This file contains the implementation of the Point class.

5.4 PointList.h File Reference

This file contains the implementation of PointList class.

```
#include <vector>
#include <iterator>
#include "Point.h"
```

Classes

- class PointList

  Structure to store a list of Point class objects(points)

5.4.1 Detailed Description

This file contains the implementation of PointList class.
PointList class, although similar to c++ vector tries to extend the default vector capabilities. The main reason for this class to exist is due to the requirement of sentinel nodes while computing the upper and lower tangents.

5.5 Tools.h File Reference

The main purpose of this file is to provide a data input function.

```
#include <vector>
#include <fstream>
#include <sstream>
#include "Point.h"
#include "PointList.h"
```

Functions

- int orient (const Point &p, const Point &q, const Point &r)

  To check the orientation of the given 3 points.
- void **log** (string s)
- PointList readPoints (int argc, char ∗argv[ ])

  Read input points from stdin or from file (if provided in args)

5.5.1 Detailed Description

The main purpose of this file is to provide a data input function.
This file provides the readPoints() function which helps to read file and take input from it. In case any file is not provided, It tries to take input from the user via stdin.

5.5.2 Function Documentation

5.5.2.1 orient()

```
int orient (
            const Point & p,
            const Point & q,
            const Point & r )
```

To check the orientation of the given 3 points.

Parameters

p	Point object
q	Point object
r	Point object

Returns

Orientation: -1 if clockwise, +1 if counter-clockwise and 0 if colinear

5.5.2.2 readPoints()

```
PointList readPoints (
            int argc,
            char * argv[] )
```

Read input points from stdin or from file (if provided in args)

Parameters

argc	no of command-line arguments
argv	command-line arguments

Returns

PointList object containing the input points in the given order

Index