DAA Assignment 1

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

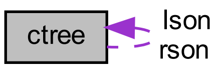
Chapter 3

Class Documentation

3.1 ctree Class Reference

A class to represent a binary tree.

Collaboration diagram for ctree:



Public Member Functions

- ctree ()

  Default constructor to create a ctree object.
- ctree (T x, string side, ctree ∗lson, ctree ∗rson)

  Constructor to create a ctree object with given initialisation values.

Public Attributes

- T x

  x-coordinate of the vertical edge
- string side

  type of side
- ctree ∗ lson

  pointer to left child of current node of tree
- ctree ∗ rson

  pointer to right child of current node of tree

3.1.1 Detailed Description

A class to represent a binary tree.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 ctree() [1/2]

```
ctree::ctree ( )  [inline]
```

Default constructor to create a ctree object.

Returns

Empty object of class ctree

3.1.2.2 ctree() [2/2]

```
ctree::ctree (
             T x,
             string side,
             ctree * lson,
             ctree * rson )  [inline]
```

Constructor to create a ctree object with given initialisation values.

Parameters

x	Value for x-coordinate of an edge
side	Value for type of side
lson	Value for left Pointer
rson	Value for left Pointer

Returns

Object of class ctree initialised with given values

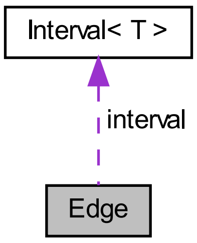
The documentation for this class was generated from the following file:

- src/contour.cpp

3.2 Edge Class Reference

A class to represent an edge in two dimensional space.

Collaboration diagram for Edge:



Public Member Functions

- Edge (Interval< T > interval, T coord, string side)

    Constructor for creating an Edge type object.
- bool operator< (const Edge &other) const

    Defines the less-than operator for set insertion and comparision.
- Edge (Interval< T > interval, T coord, string side)

    Constructor for creating an Edge type object.
- bool operator< (const Edge &other) const

    Defines the less-than operator for set insertion and comparision.

Public Attributes

- Interval< T > interval

    Interval of the edge.
- T coord

    coordinate of the edge that remains constant between the Interval of the edge
- string side

    Represents what side of the figure the edge is - {'left', 'right', 'top', 'bottom'}.

3.2.1 Detailed Description

A class to represent an edge in two dimensional space.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Edge() [1/2]

```
Edge::Edge (
            Interval< T > interval,
            T coord,
            string side )  [inline]
```

Constructor for creating an Edge type object.

Parameters

interval	Value for interval
coord	Value for coord
side	Value for side

Returns

An empty Interval type object

3.2.2.2 Edge() [2/2]

```
Edge::Edge (
            Interval< T > interval,
            T coord,
            string side )  [inline]
```

Constructor for creating an Edge type object.

Parameters

interval	Value for interval
coord	Value for coord
side	Value for side

Returns

An empty Interval type object

3.2.3 Member Function Documentation

3.2.3.1 operator<() [1/2]

```
bool Edge::operator< (
            const Edge & other ) const  [inline]
```

Defines the less-than operator for set insertion and comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object less than other, else false

3.2.3.2 operator<() `[2/2]`

```
bool Edge::operator< (
            const Edge & other ) const  [inline]
```

Defines the less-than operator for set insertion and comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object less than other, else false

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

- src/contour.cpp
- src/measure.cpp

3.3 Interval Class Reference

A class to represent an interval between two lines in the 2D plane.

Public Member Functions

- Interval ()

  Default constructor for creating an empty Interval type object.
- Interval (T bottom, T top)

  Constructor for creating an Interval type object.
- bool operator< (const Interval &other) const

  Defines the less-than operator for set insertion and comparision.
- bool operator== (const Interval &other) const

  Defines the equals-to operator for comparision.
- Interval ()

  Default constructor for creating an empty Interval type object.

- Interval (T bottom, T top)

    Constructor for creating an Interval type object.
- bool operator< (const Interval &other) const

    Defines the less-than operator for set insertion and comparision.
- bool operator== (const Interval &other) const

    Defines the equals-to operator for comparision.

Public Attributes

- T top

    upper limit of the interval
- T bottom

    lower limit of the interval

3.3.1 Detailed Description

A class to represent an interval between two lines in the 2D plane.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Interval() [1/4]

```
Interval::Interval ( )  [inline]
```

Default constructor for creating an empty Interval type object.

Returns

An empty Interval type object

3.3.2.2 Interval() [2/4]

```
Interval::Interval (
            T bottom,
            T top )  [inline]
```

Constructor for creating an Interval type object.

Parameters

bottom	Value for bottom
top	Value for top

Returns

An empty Interval type object

3.3.2.3 Interval() [3/4]

```
Interval::Interval ( )  [inline]
```

Default constructor for creating an empty Interval type object.

Returns

An empty Interval type object

3.3.2.4 Interval() [4/4]

```
Interval::Interval (
             T bottom,
             T top )  [inline]
```

Constructor for creating an Interval type object.

Parameters

bottom	Value for bottom
top	Value for top

Returns

An empty Interval type object

3.3.3 Member Function Documentation

3.3.3.1 operator<() [1/2]

```
bool Interval::operator< (
             const Interval & other ) const  [inline]
```

Defines the less-than operator for set insertion and comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object less than other, else false

3.3.3.2 operator<() [2/2]

```
bool Interval::operator< (
            const Interval & other ) const  [inline]
```

Defines the less-than operator for set insertion and comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object less than other, else false

3.3.3.3 operator==() [1/2]

```
bool Interval::operator== (
            const Interval & other ) const  [inline]
```

Defines the equals-to operator for comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object is equal to the other, else false

3.3.3.4 operator==() [2/2]

```
bool Interval::operator== (
            const Interval & other ) const  [inline]
```

Defines the equals-to operator for comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object is equal to the other, else false

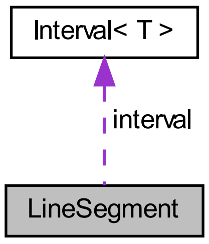
The documentation for this class was generated from the following files:

- src/contour.cpp
- src/measure.cpp

3.4 LineSegment Class Reference

A class to represent a Line Segment between the given interval of two points with coord as the offset from the axes.

Collaboration diagram for LineSegment:



Public Member Functions

- LineSegment (Interval< T > interval, T coord)

  Constructor for creating an LineSegment type object.
- bool operator< (const LineSegment &other) const

  Defines the less-than operator for set insertion and comparision.
- LineSegment (Interval< T > interval, T coord)

  Constructor for creating an LineSegment type object.
- bool operator< (const LineSegment &other) const

  Defines the less-than operator for set insertion and comparision.

Public Attributes

- Interval< T > interval

  interval between the two end points of the Line Segment
- T coord

  coordinate that remains constant between the endpoints of the line segment

3.4.1 Detailed Description

A class to represent a Line Segment between the given interval of two points with coord as the offset from the axes.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 LineSegment() [1/2]

```
LineSegment::LineSegment (
            Interval< T > interval,
            T coord )  [inline]
```

Constructor for creating an LineSegment type object.

Parameters

interval	Value for bottom
coord	Value for coord

Returns

An empty Interval type object

3.4.2.2 LineSegment() [2/2]

```
LineSegment::LineSegment (
            Interval< T > interval,
            T coord )  [inline]
```

Constructor for creating an LineSegment type object.

Parameters

interval	Value for bottom
coord	Value for coord

Returns

An empty Interval type object

3.4.3 Member Function Documentation

3.4.3.1 operator<() [1/2]

```
bool LineSegment::operator< (
             const LineSegment & other ) const  [inline]
```

Defines the less-than operator for set insertion and comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object less than other, else false

3.4.3.2 operator<() [2/2]

```
bool LineSegment::operator< (
             const LineSegment & other ) const  [inline]
```

Defines the less-than operator for set insertion and comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object less than other, else false

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

- src/contour.cpp
- src/measure.cpp

3.5 Point Class Reference

A simple class to represent a point in a two dimensional space.

Public Member Functions

- Point (T x, T y)
  This constructor is used to initialise the object with given x and y coordinates.
- bool operator< (const Point &other) const
  Defines the less-than operator for set insertion and comparision.
- Point (T x, T y)
  This constructor is used to initialise the object with given x and y coordinates.
- bool operator< (const Point &other) const
  Defines the less-than operator for set insertion and comparision.

Public Attributes

- T x

  x-coordinate of the point represented by the object
- T y

  y-coordinate of the point represented by the object

3.5.1 Detailed Description

A simple class to represent a point in a two dimensional space.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Point() [1/2]

```
Point::Point (
            T x,
            T y )  [inline]
```

This constructor is used to initialise the object with given x and y coordinates.

Parameters

x	x-coordinate
y	y-coordinate

Returns

The object initialised with the given coordinates

3.5.2.2 Point() [2/2]

```
Point::Point (
            T x,
            T y )  [inline]
```

This constructor is used to initialise the object with given x and y coordinates.

Parameters

x	x-coordinate
y	y-coordinate

Returns

The object initialised with the given coordinates

3.5.3 Member Function Documentation

3.5.3.1 operator<() `[1/2]`

```
bool Point::operator< (
             const Point & other ) const  [inline]
```

Defines the less-than operator for set insertion and comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object less than other, else false

3.5.3.2 operator<() `[2/2]`

```
bool Point::operator< (
             const Point & other ) const  [inline]
```

Defines the less-than operator for set insertion and comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object less than other, else false

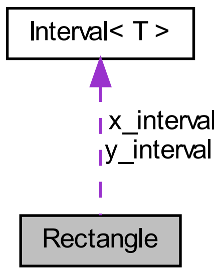
The documentation for this class was generated from the following files:

- src/contour.cpp
- src/measure.cpp

3.6 Rectangle Class Reference

A class to represent a rectangle in a two dimensional space.

Collaboration diagram for Rectangle:



Public Member Functions

- Rectangle ()

  This is the default constructor for creating an empty Interval type object.
- Rectangle (T x1, T x2, T y1, T y2)

  This constructor is used to initialise the object with given x and y coordinates.
- bool operator< (const Rectangle &other) const

  Defines the less-than operator for set insertion and comparision.
- Rectangle ()

  This is the default constructor for creating an empty Interval type object.
- Rectangle (T x1, T x2, T y1, T y2)

  This constructor is used to initialise the object with given x and y coordinates.
- bool operator< (const Rectangle &other) const

  Defines the less-than operator for set insertion and comparision.

Public Attributes

- T x_left

  x-coordinate of left side
- T x_right

  x-coordinate of right side
- T y_bottom

  y-coordinate of left side
- T y_top

  y-coordinate of right side
- Interval< T > x_interval

  Interval on x-axis.
- Interval< T > y_interval

  Interval on y-axis.

3.6.1 Detailed Description

A class to represent a rectangle in a two dimensional space.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 Rectangle() [1/4]

```
Rectangle::Rectangle ( )  [inline]
```

This is the default constructor for creating an empty Interval type object.

Returns

An empty Interval type object

3.6.2.2 Rectangle() [2/4]

```
Rectangle::Rectangle (
             T x1,
             T x2,
             T y1,
             T y2 )  [inline]
```

This constructor is used to initialise the object with given x and y coordinates.

Parameters

x1	Value for x_left
x2	Value for x_right
y1	Value for y_bottom
y2	Value for y_top

Returns

The object initialised with the given coordinates

3.6.2.3 Rectangle() [3/4]

```
Rectangle::Rectangle ( )  [inline]
```

This is the default constructor for creating an empty Interval type object.

Returns

An empty Interval type object

3.6.2.4 Rectangle() [4/4]

```
Rectangle::Rectangle (
             T x1,
             T x2,
             T y1,
             T y2 )  [inline]
```

This constructor is used to initialise the object with given x and y coordinates.

Parameters

x1	Value for x_left
x2	Value for x_right
y1	Value for y_bottom
y2	Value for y_top

Returns

The object initialised with the given coordinates

3.6.3 Member Function Documentation

3.6.3.1 operator<() [1/2]

```
bool Rectangle::operator< (
             const Rectangle & other ) const  [inline]
```

Defines the less-than operator for set insertion and comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object less than other, else false

3.6.3.2 operator<() [2/2]

```
bool Rectangle::operator< (
             const Rectangle & other ) const  [inline]
```

Defines the less-than operator for set insertion and comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object less than other, else false

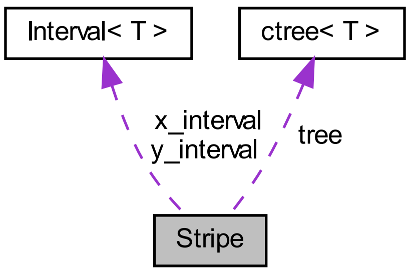
The documentation for this class was generated from the following files:

- src/contour.cpp
- src/measure.cpp

3.7 Stripe Class Reference

A class to represent a horizontal stripe in two dimensions.

Collaboration diagram for Stripe:



Public Member Functions

- Stripe ()

    Default constructor to create a Stripe object.
- Stripe (Interval< T > x_interval, Interval< T > y_interval, ctree< T > ∗tree)

    Constructor to create a Stripe object with given initialisation values.
- bool operator< (const Stripe &other) const

Defines the less-than operator for set insertion and comparision.

- Stripe ()

  Default constructor to create a Stripe object.
- Stripe (Interval< T > x_interval, Interval< T > y_interval, T x_measure)

  Constructor to create a Stripe object with given initialisation values.
- bool operator< (const Stripe &other) const

  Defines the less-than operator for set insertion and comparision.

Public Attributes

- Interval< T > x_interval

  Interval of the stripe on the x-axis.
- Interval< T > y_interval

  Interval of the stripe on the y-axis.
- ctree< T > ∗ tree

  Pointer to root of a binary tree.
- T x_measure

  Total length of intervals contained in stripes on x-axis.

3.7.1 Detailed Description

A class to represent a horizontal stripe in two dimensions.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 Stripe() [1/4]

```
Stripe::Stripe ( )  [inline]
```

Default constructor to create a Stripe object.

Returns

Empty object of class Stripe

3.7.2.2 Stripe() [2/4]

```
Stripe::Stripe (
            Interval< T > x_interval,
            Interval< T > y_interval,
            ctree< T > ∗ tree )  [inline]
```

Constructor to create a Stripe object with given initialisation values.

Parameters

x_interval	Value for x_interval
y_interval	Value for y_interval
tree	Value for root Pointer

Returns

Object of class Stripe initialised with given values

3.7.2.3 Stripe() [3/4]

```
Stripe::Stripe ( )  [inline]
```

Default constructor to create a Stripe object.

Returns

Empty object of class Stripe

3.7.2.4 Stripe() [4/4]

```
Stripe::Stripe (
            Interval< T > x_interval,
            Interval< T > y_interval,
            T x_measure )  [inline]
```

Constructor to create a Stripe object with given initialisation values.

Parameters

x_interval	Value for x_interval
y_interval	Value for y_interval
x_measure	Value for x_measure

Returns

Object of class Stripe initialised with given values

3.7.3 Member Function Documentation

3.7.3.1 operator<() [1/2]

```
bool Stripe::operator< (
             const Stripe & other ) const  [inline]
```

Defines the less-than operator for set insertion and comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object less than other, else false

3.7.3.2 operator<() [2/2]

```
bool Stripe::operator< (
             const Stripe & other ) const  [inline]
```

Defines the less-than operator for set insertion and comparision.

Parameters

other	object with which comparision needs to be done

Returns

true if object less than other, else false

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

- src/contour.cpp
- src/measure.cpp

Chapter 4

File Documentation

4.1 src/contour.cpp File Reference

Computation of the contour for a set of iso rectangles using divide-and-conquer.

```
#include <bits/stdc++.h>
```

Classes

- class Point

  A simple class to represent a point in a two dimensional space.
- class Interval

  A class to represent an interval between two lines in the 2D plane.
- class LineSegment

  A class to represent a Line Segment between the given interval of two points with coord as the offset from the axes.
- class Rectangle

  A class to represent a rectangle in a two dimensional space.
- class Edge

  A class to represent an edge in two dimensional space.
- class ctree

  A class to represent a binary tree.
- class Stripe

  A class to represent a horizontal stripe in two dimensions.

Macros

- #define **tplate** template <typename T = long double>

Functions

- template<class T >
  set< T > operator- (set< T > a, set< T > b)

  Defines the minus operator for computing set difference of set A and set B.
- template<class T >
  set< T > operator+ (set< T > a, set< T > b)

  Defines the plus operator for computing union of set A and set B.
- template<class T >
  set< T > operator^ (set< T > a, set< T > b)

  Defines the intersection operator for computing set intersection of two sets.
- tplate void getNodes (ctree< T > *root, vector< Edge< T >> &v, T start, T end)

  Performs inorder traversal on the tree represented by the root node passed to it.
- tplate bool isEnclosed (ctree< T > *root, T start, T end)

  Checks if a horizontal edge should be included in the contour.
- tplate vector< Edge< T > > filter (vector< Edge< T >> v)

  Removes vertical edges that are strictly enclosed within the contour.
- tplate set< LineSegment< T > > intervals (Edge< T > h, ctree< T > *tree)

  Finds set of horizontal line segments that are part of the contour.
- tplate set< LineSegment< T > > contour_pieces (Edge< T > h, set< Stripe< T >> S)

  Finds pieces of an edge belonging to the contour.
- tplate set< LineSegment< T > > contour (vector< Edge< T >> H, set< Stripe< T >> S)

  Amalgamates all the pieces of the contour.
- tplate set< Interval< T > > partition (set< T > Y)

  Finds intervals created by a set of coordinates.
- tplate set< Stripe< T > > Copy (set< Stripe< T >> S, set< T > P, Interval< T > x_int)

  Copies a set of stripes into the stripes created by partitions.
- tplate void Blacken (set< Stripe< T >> &S, set< Interval< T >> J)

  Removes the edges that are covered by other rectangles for a particular stripe.
- tplate set< Stripe< T > > Concat (set< Stripe< T >> S1, set< Stripe< T >> S2, set< T > P, Interval< T > x_int)

  Combine the results from two sets of stripes.
- tplate set< Stripe< T > > STRIPES (vector< Edge< T >> &V, Interval< T > &x_ext, set< Interval< T >> &L, set< Interval< T >> &R, set< T > &P)

  Creates the stripes required for finding the contour.
- tplate set< Stripe< T > > RECTANGLE_DAC (set< Rectangle< T >> RECT)

  A helper function that converts the Rectangle into edges and calls the STRIPES function on those intervals.
- int **main** (int argc, char const *argv[ ])

Variables

- tplate const T inf = numeric_limits<T>::infinity()

  Constant to represent infinity.

4.1.1 Detailed Description

Computation of the contour for a set of iso rectangles using divide-and-conquer.

4.1.2 Function Documentation

4.1.2.1 Blacken()

```
tplate void Blacken (
             set< Stripe< T >> & S,
             set< Interval< T >> J )
```

Removes the edges that are covered by other rectangles for a particular stripe.

Parameters

S	Set of stripes
J	Set of Intervals

4.1.2.2 Concat()

```
tplate set<Stripe<T> > Concat (
             set< Stripe< T >> S1,
             set< Stripe< T >> S2,
             set< T > P,
             Interval< T > x_int )
```

Combine the results from two sets of stripes.

Parameters

S1	First set of stripes
S2	Second set of stripes
P	Set of coordinates
x_int	Interval on x-axis for both sets of stripes

Returns

A set of stripes after concatenation

4.1.2.3 contour()

```
tplate set<LineSegment<T> > contour (
             vector< Edge< T >> H,
             set< Stripe< T >> S )
```

Amalgamates all the pieces of the contour.

Parameters

H	vector of all the edges of the rectangles
S	set of Stripes

Returns

A set of line segments that define the contour for the given set of rectangles defined by the edges

4.1.2.4 contour_pieces()

```
tplate set<LineSegment<T> > contour_pieces (
              Edge< T > h,
              set< Stripe< T >> S )
```

Finds pieces of an edge belonging to the contour.

Parameters

h	edge of a rectangle
S	stripe adjacent to edge

Returns

A set of line segments on the edge belonging to the contour

4.1.2.5 Copy()

```
tplate set<Stripe<T> > Copy (
              set< Stripe< T >> S,
              set< T > P,
              Interval< T > x_int )
```

Copies a set of stripes into the stripes created by partitions.

Parameters

S	Set of stripes
P	Set of coordinates
x_int	Interval of stripes on x-axis

Returns

A set of stripes

4.1.2.6 filter()

```
tplate vector<Edge<T> > filter (
             vector< Edge< T >> v )
```

Removes vertical edges that are strictly enclosed within the contour.

Parameters

v	Set of vertical edges of a stripe

Returns

New set of edges with enclosed edges removed

4.1.2.7 getNodes()

```
tplate void getNodes (
             ctree< T > * root,
             vector< Edge< T >> & v,
             T start,
             T end )
```

Performs inorder traversal on the tree represented by the root node passed to it.

Parameters

root	root node of the tree
v	a vector of edges passed by reference
start	start coordinate of edge
end	end coordinate of edge

4.1.2.8 intervals()

```
tplate set<LineSegment<T> > intervals (
             Edge< T > h,
             ctree< T > * tree )
```

Finds set of horizontal line segments that are part of the contour.

Parameters

h	Edge of the rectangle
tree	root of binary tree

Returns

A set of horizontal line segments on the edge belonging to the contour

4.1.2.9 isEnclosed()

```
tplate bool isEnclosed (
            ctree< T > * root,
            T start,
            T end )
```

Checks if a horizontal edge should be included in the contour.

Parameters

root	root node of the tree
start	of interval
end	of interval

Returns

true if the edge should not included and false otherwise

4.1.2.10 operator+()

```
template<class T >
set<T> operator+ (
            set< T > a,
            set< T > b )
```

Defines the plus operator for computing union of set A and set B.

Parameters

a	set a
b	set b

Returns

a set with the union of set a and set b

4.1.2.11 operator-()

```
template<class T >
set<T> operator- (
                set< T > a,
                set< T > b )
```

Defines the minus operator for computing set difference of set A and set B.

Parameters

a	the set from which to elements are to be removed
b	the set of items to be removed

Returns

a set with items of set b removed

4.1.2.12 operator^()

```
template<class T >
set<T> operator^ (
                set< T > a,
                set< T > b )
```

Defines the intersection operator for computing set intersection of two sets.

Parameters

a	set a
b	set b

Returns

intersection of set a and set b

4.1.2.13 partition()

```
tplate set<Interval<T> > partition (
                set< T > Y )
```

Finds intervals created by a set of coordinates.

Parameters

Y	set of y-coordinates

Returns

A set of intervals

4.1.2.14 RECTANGLE_DAC()

```
tplate set<Stripe<T> > RECTANGLE_DAC (
              set< Rectangle< T >> RECT )
```

A helper function that converts the Rectangle into edges and calls the STRIPES function on those intervals.

Parameters

RECT	A set of Rectangles

Returns

A set of stripes

4.1.2.15 STRIPES()

```
tplate set<Stripe<T> > STRIPES (
              vector< Edge< T >> & V,
              Interval< T > & x_ext,
              set< Interval< T >> & L,
              set< Interval< T >> & R,
              set< T > & P )
```

Creates the stripes required for finding the contour.

Parameters

V	Set of edges
x_ext	Interval on x-axis for set of stripes
L	Intervals consisting of 'left' edges
R	Intervals consisting of 'right' edges
P	Set of coordinates

Returns

A set of stripes

4.2 src/measure.cpp File Reference

Computation of the measure for a set of iso rectangles using divide-and-conquer.

```
#include <bits/stdc++.h>
```

Classes

- class Point

  A simple class to represent a point in a two dimensional space.
- class Interval

  A class to represent an interval between two lines in the 2D plane.
- class LineSegment

  A class to represent a Line Segment between the given interval of two points with coord as the offset from the axes.
- class Rectangle

  A class to represent a rectangle in a two dimensional space.
- class Edge

  A class to represent an edge in two dimensional space.
- class Stripe

  A class to represent a horizontal stripe in two dimensions.

Macros

- #define **tplate** template <typename T = long double>

Functions

- template<class T >
  set< T > operator- (set< T > a, set< T > b)

  Defines the minus operator for computing set difference of set A and set B.
- template<class T >
  set< T > operator+ (set< T > a, set< T > b)

  Defines the plus operator for computing union of set A and set B.
- template<class T >
  set< T > operator^ (set< T > a, set< T > b)

  Defines the intersection operator for computing set intersection of two sets.
- tplate set< Interval< T > > partition (set< T > Y)

  Finds intervals created by a set of coordinates.
- tplate set< Stripe< T > > Copy (set< Stripe< T >> S, set< T > P, Interval< T > x_int)

  Copies a set of stripes into the stripes created by partitions.
- tplate void Blacken (set< Stripe< T >> &S, set< Interval< T >> J)

  Removes the edges that are covered by other rectangles for a particular stripe.
- tplate set< Stripe< T > > Concat (set< Stripe< T >> S1, set< Stripe< T >> S2, set< T > P, Interval< T > x_int)

  Combine the results from two sets of stripes.
- tplate set< Stripe< T > > STRIPES (vector< Edge< T >> &V, Interval< T > &x_ext, set< Interval< T >> &L, set< Interval< T >> &R, set< T > &P)

  Creates the stripes required for finding the contour.
- tplate set< Stripe< T > > RECTANGLE_DAC (set< Rectangle< T >> RECT)

  A helper function that converts the Rectangle into edges and calls the STRIPES function on those intervals.
- int **main** (int argc, char const ∗argv[ ])

Variables

- tplate const T inf = numeric_limits<T>::infinity()

  Constant to represent infinity.

4.2.1 Detailed Description

Computation of the measure for a set of iso rectangles using divide-and-conquer.

4.2.2 Function Documentation

4.2.2.1 Blacken()

```
tplate void Blacken (
             set< Stripe< T >> & S,
             set< Interval< T >> J )
```

Removes the edges that are covered by other rectangles for a particular stripe.

Parameters

S	Set of stripes
J	Set of Intervals

4.2.2.2 Concat()

```
tplate set<Stripe<T> > Concat (
             set< Stripe< T >> S1,
             set< Stripe< T >> S2,
             set< T > P,
             Interval< T > x_int )
```

Combine the results from two sets of stripes.

Parameters

S1	First set of stripes
S2	Second set of stripes
P	Set of coordinates
x_int	Interval on x-axis for both sets of stripes

Returns

A set of stripes after concatenation

4.2.2.3 Copy()

```
tplate set<Stripe<T> > Copy (
              set< Stripe< T >> S,
              set< T > P,
              Interval< T > x_int )
```

Copies a set of stripes into the stripes created by partitions.

Parameters

S	Set of stripes
P	Set of coordinates
x_int	Interval of stripes on x-axis

Returns

A set of stripes

4.2.2.4 operator+()

```
template<class T >
set<T> operator+ (
              set< T > a,
              set< T > b )
```

Defines the plus operator for computing union of set A and set B.

Parameters

a	set a
b	set b

Returns

a set with the union of set a and set b

4.2.2.5 operator-()

```
template<class T >
set<T> operator- (
            set< T > a,
            set< T > b )
```

Defines the minus operator for computing set difference of set A and set B.

Parameters

a	the set from which to elements are to be removed
b	the set of items to be removed

Returns

a set with items of set b removed

4.2.2.6 operator^()

```
template<class T >
set<T> operator^ (
            set< T > a,
            set< T > b )
```

Defines the intersection operator for computing set intersection of two sets.

Parameters

a	set a
b	set b

Returns

intersection of set a and set b

4.2.2.7 partition()

```
tplate set<Interval<T> > partition (
            set< T > Y )
```

Finds intervals created by a set of coordinates.

Parameters

Y	set of y-coordinates

Returns

A set of intervals

4.2.2.8 RECTANGLE_DAC()

```
tplate set<Stripe<T> > RECTANGLE_DAC (
              set< Rectangle< T >> RECT )
```

A helper function that converts the Rectangle into edges and calls the STRIPES function on those intervals.

Parameters

RECT	A set of Rectangles

Returns

A set of stripes

4.2.2.9 STRIPES()

```
tplate set<Stripe<T> > STRIPES (
              vector< Edge< T >> & V,
              Interval< T > & x_ext,
              set< Interval< T >> & L,
              set< Interval< T >> & R,
              set< T > & P )
```

Creates the stripes required for finding the contour.

Parameters

V	Set of edges
x_ext	Interval on x-axis for set of stripes
L	Intervals consisting of 'left' edges
R	Intervals consisting of 'right' edges
P	Set of coordinates

Returns

A set of stripes