Divide and Conquer approach to find the Measure and Contour for a set of iso-rectangles

The following is the list of team members:

S.No	ID No	Name
1.	2017B2A71604H	Pranav V Grandhi
2.	2017B3A70878H	Rahul R Shevade
3.	2017B3A70740H	Vamshi Kasam
4.	2017B3A71386H	Shanmukh Kali Prasad

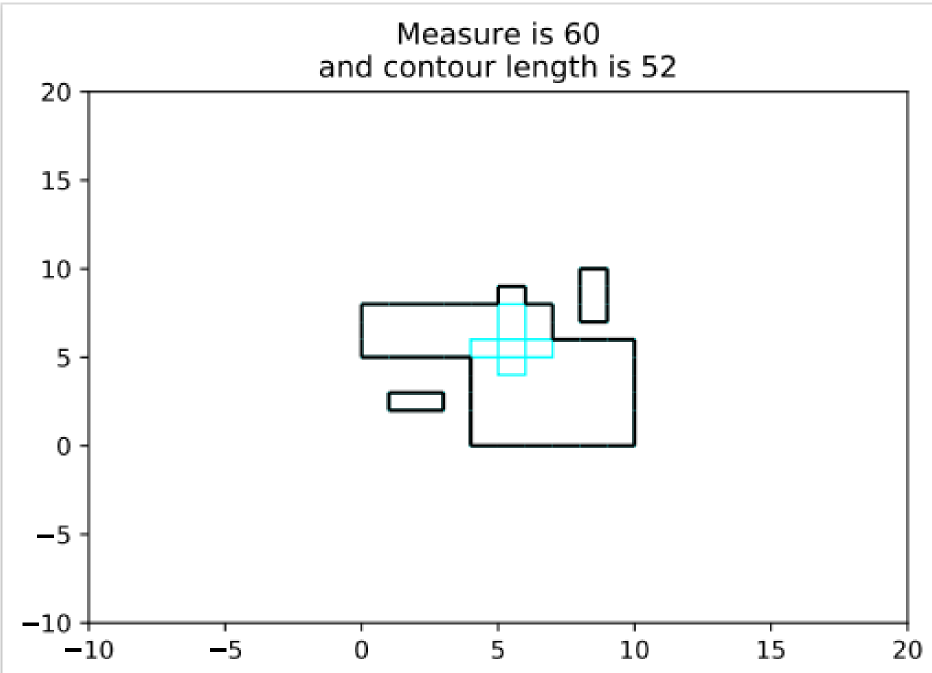
Experimental Results

We have implemented the divide and conquer algorithm which was discussed in the paper written by Ralf Hartmut Guting. The objective is to find out the measure and the contour of a set of iso-rectangles. The measure is the area that is bounded inside the set of rectangles and contour is the outermost edges of the set of rectangles.
The algorithm described in a paper is a divide and conquer algorithm which runs in O(nlogn) time.

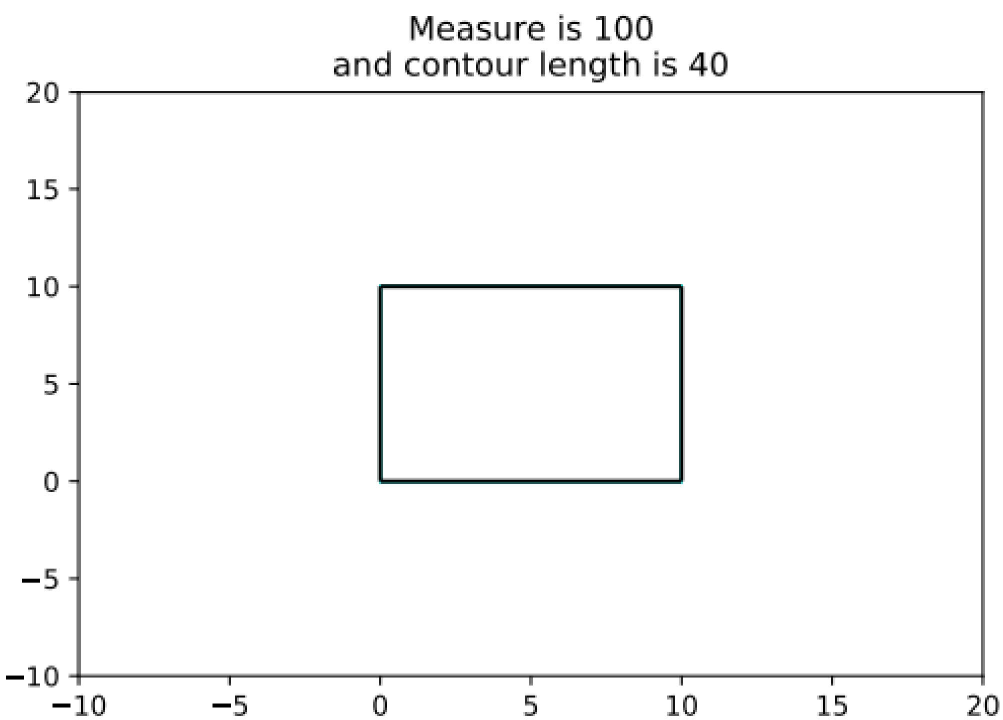
We have implemented the algorithm described in the paper using C++. We have also visualized the outputs using the matplotlib library in python. The program has been tested for smaller inputs and the answers were validated. Then we have also implemented it using larger datasets to check the robustness of the algorithm.

The following are the visualization results for the tests we have run for both contour and the measure:

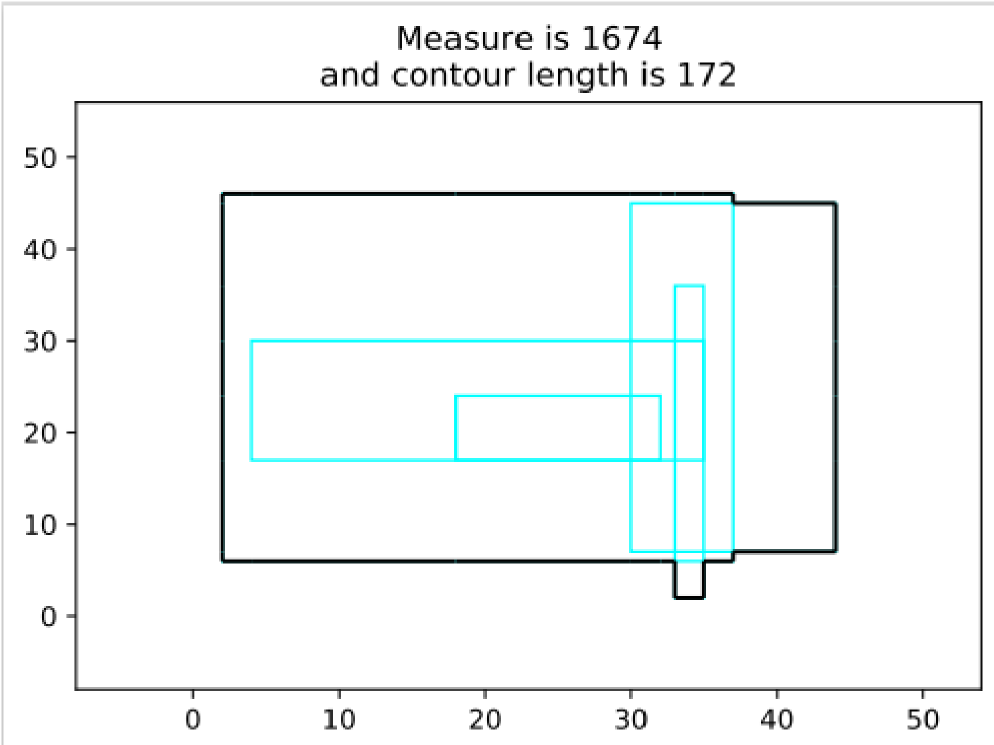
Test case 1



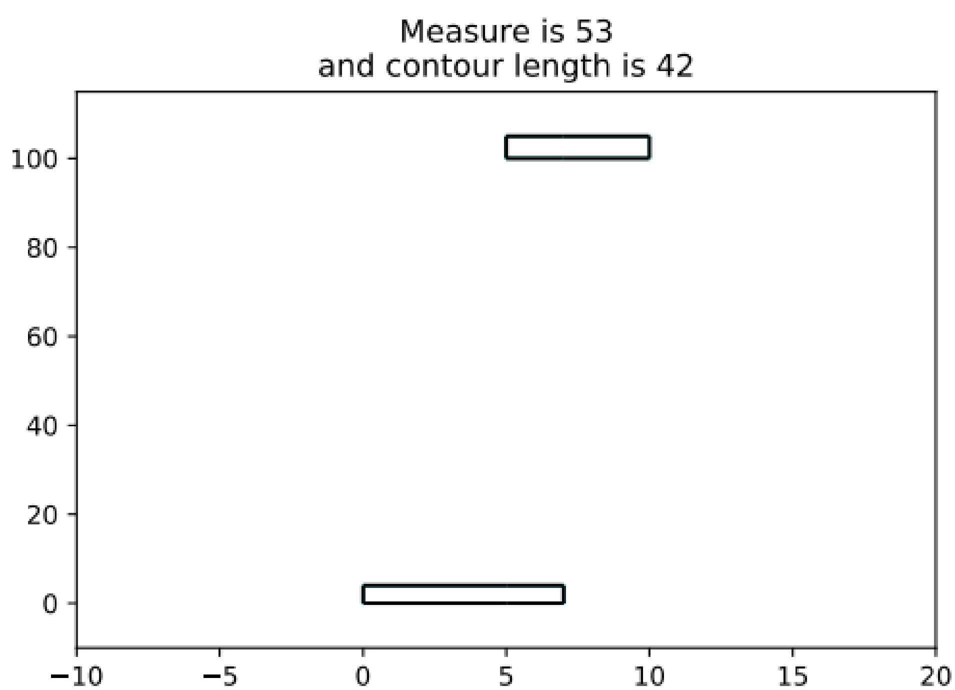
Test case 2



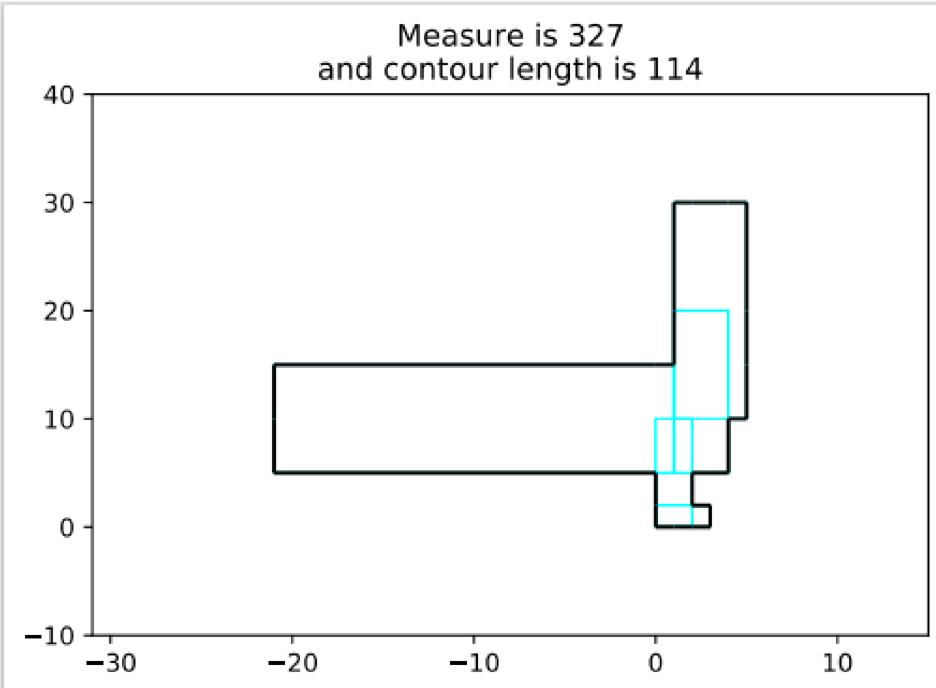
Test Case 3



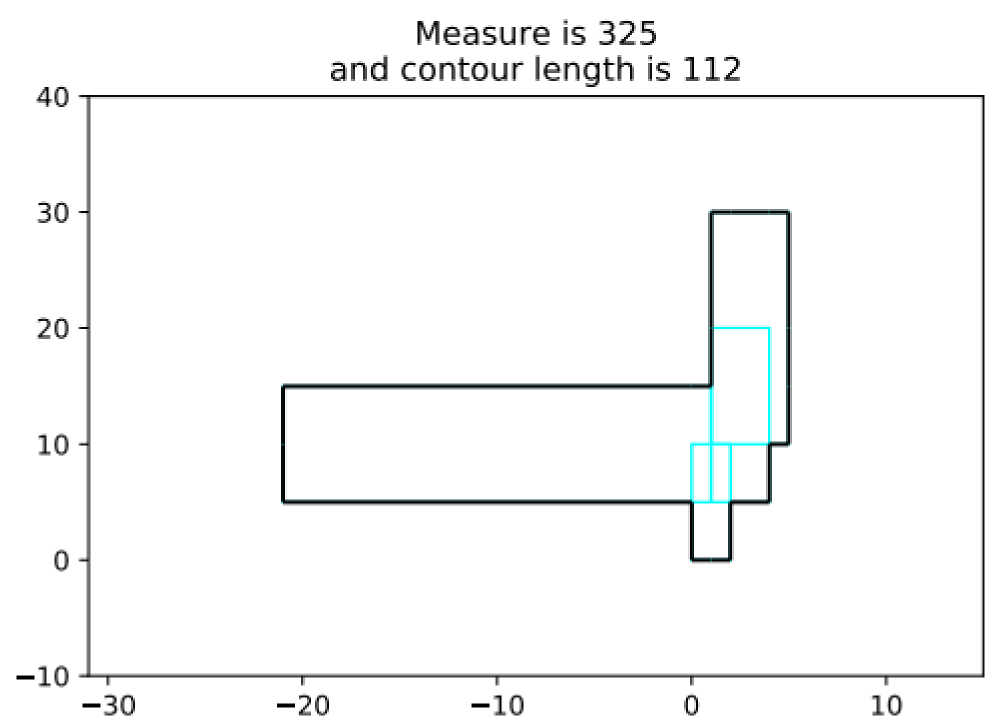
Test Case 4



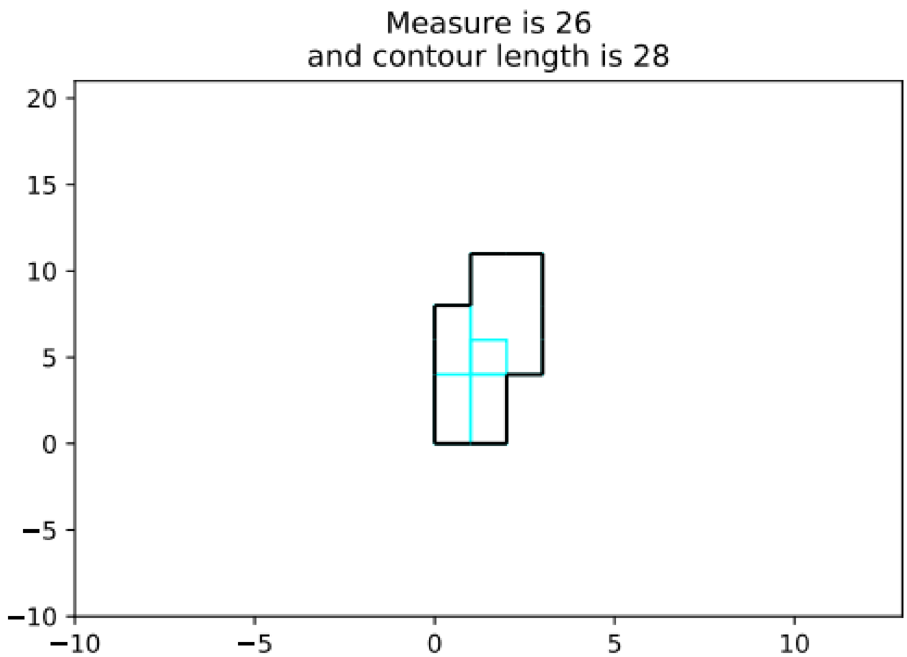
Test Case 5



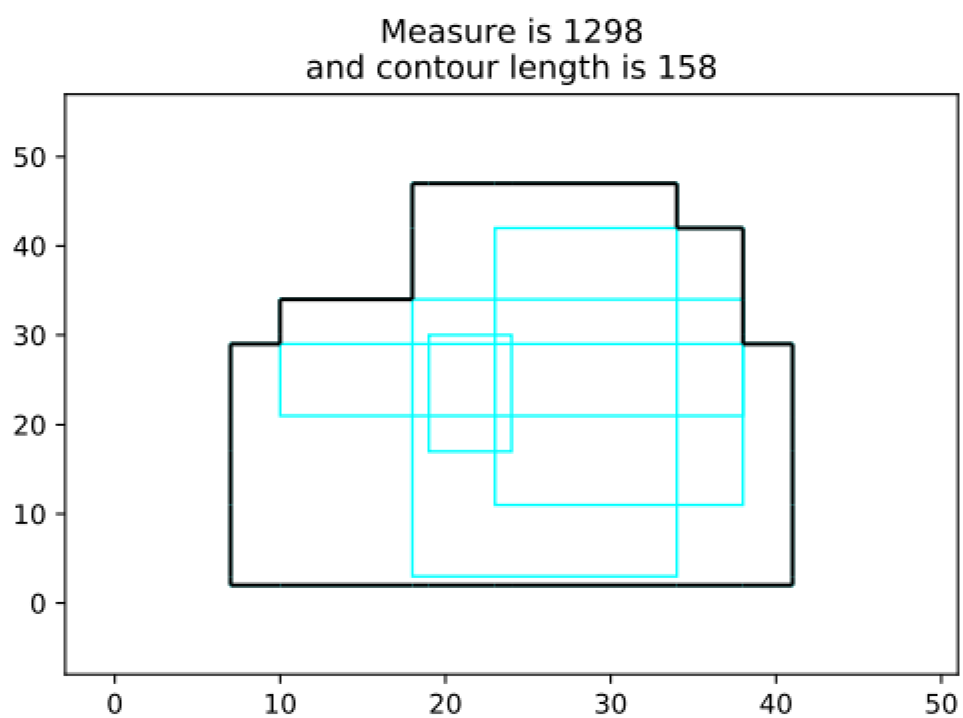
Test Case 6



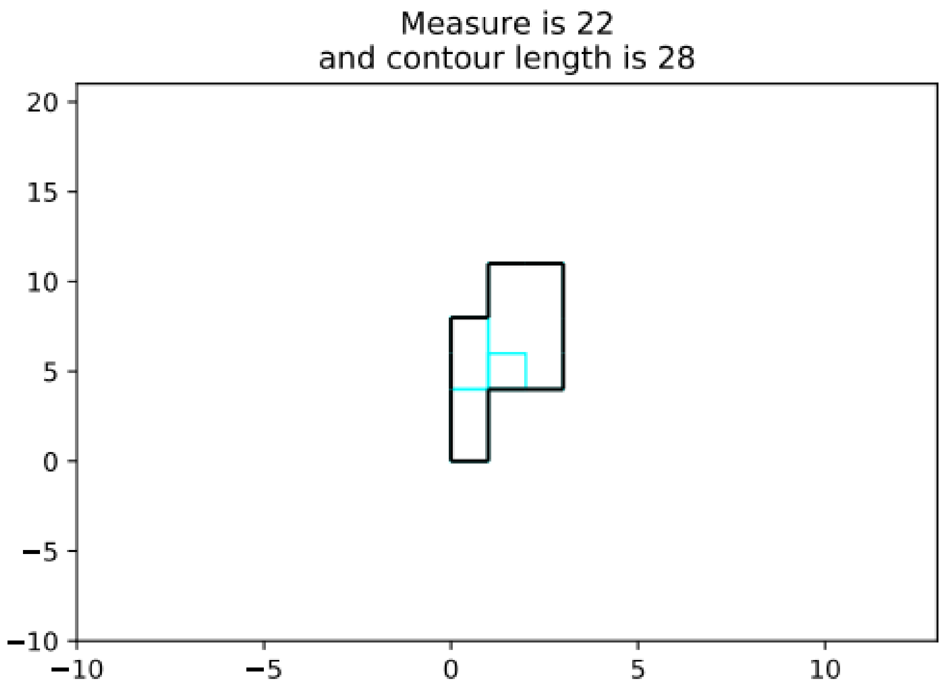
Test Case 7



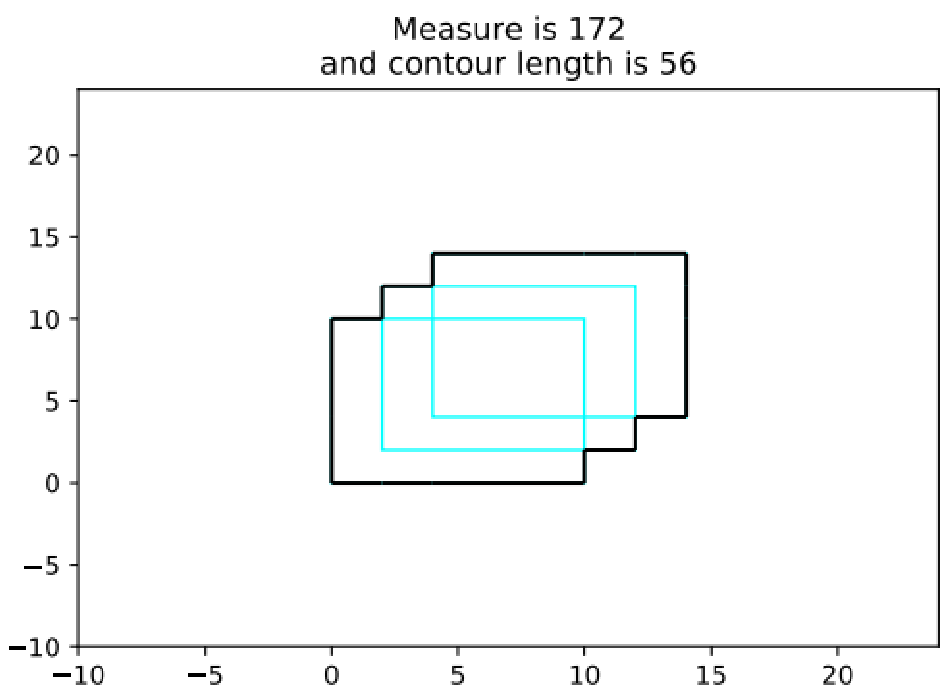
Test Case 8



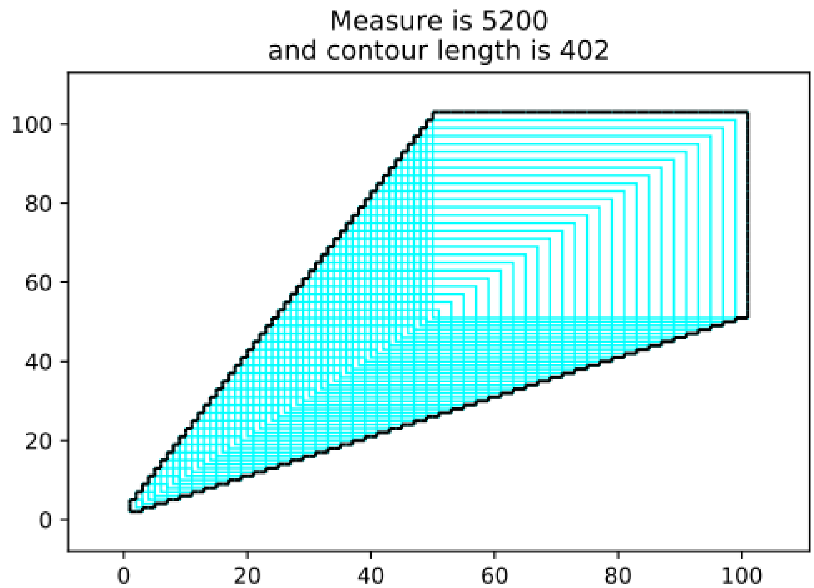
Test Case 9



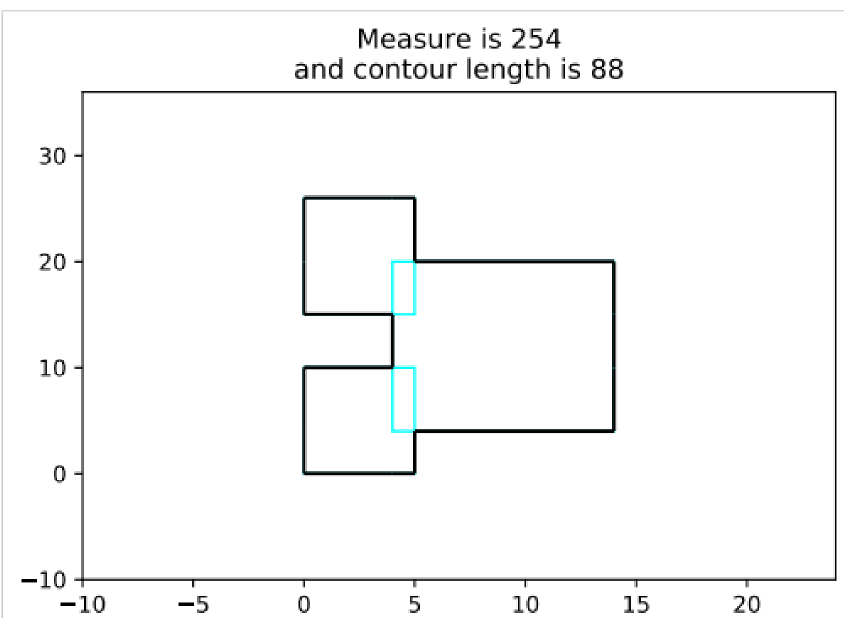
Test Case 10



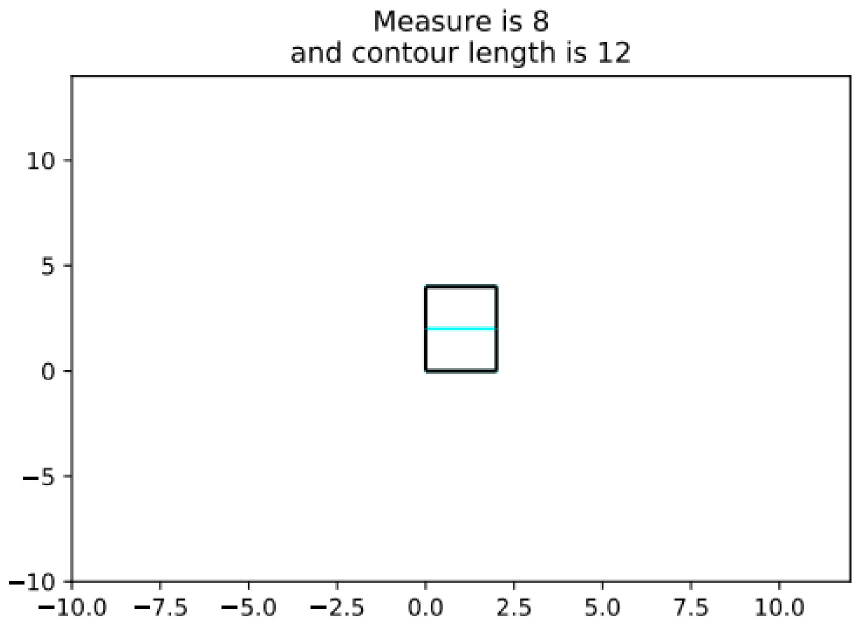
Test Case 15



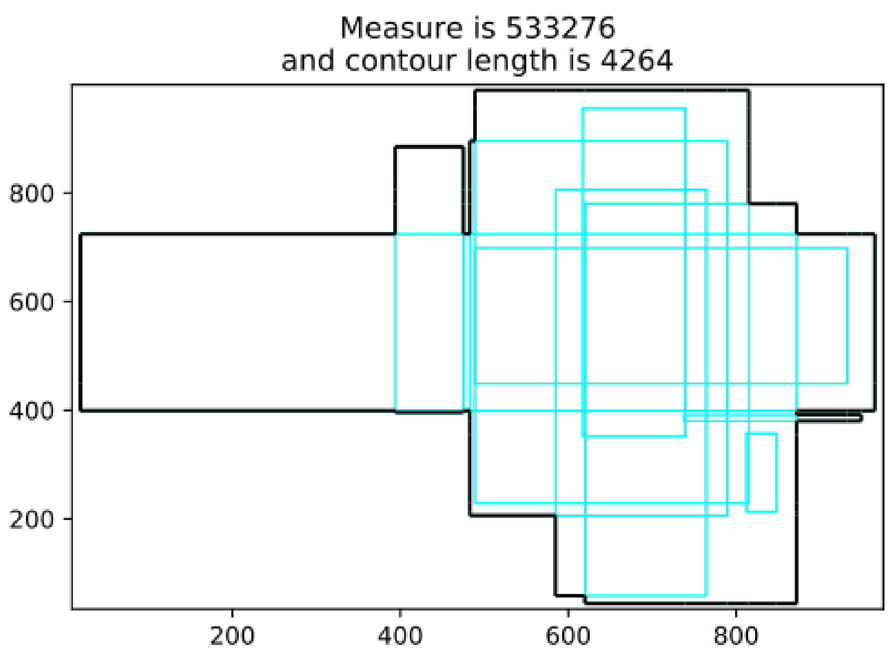
Test Case 11



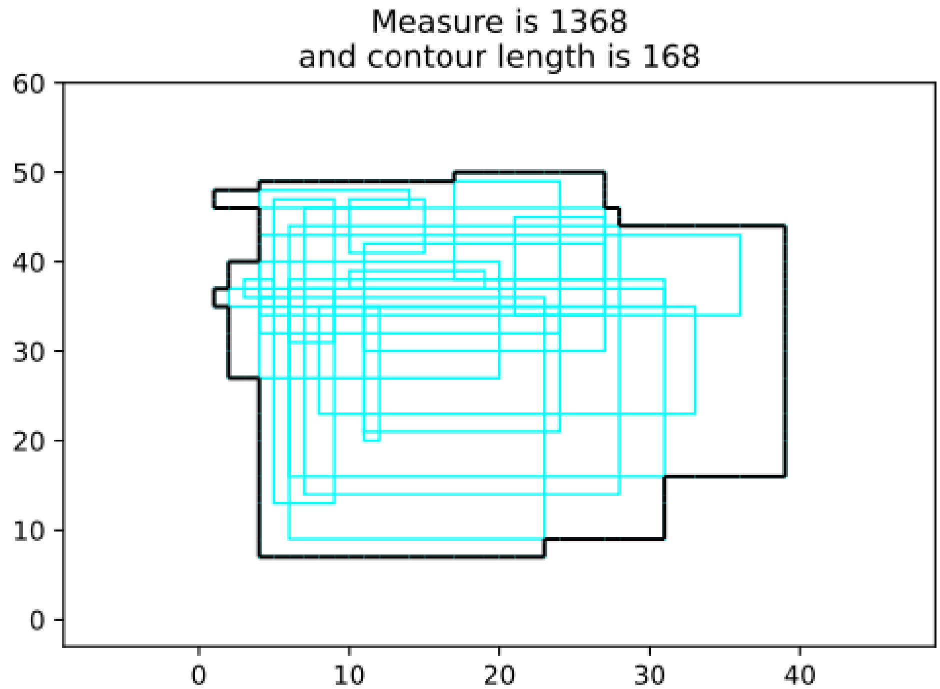
Test Case 12



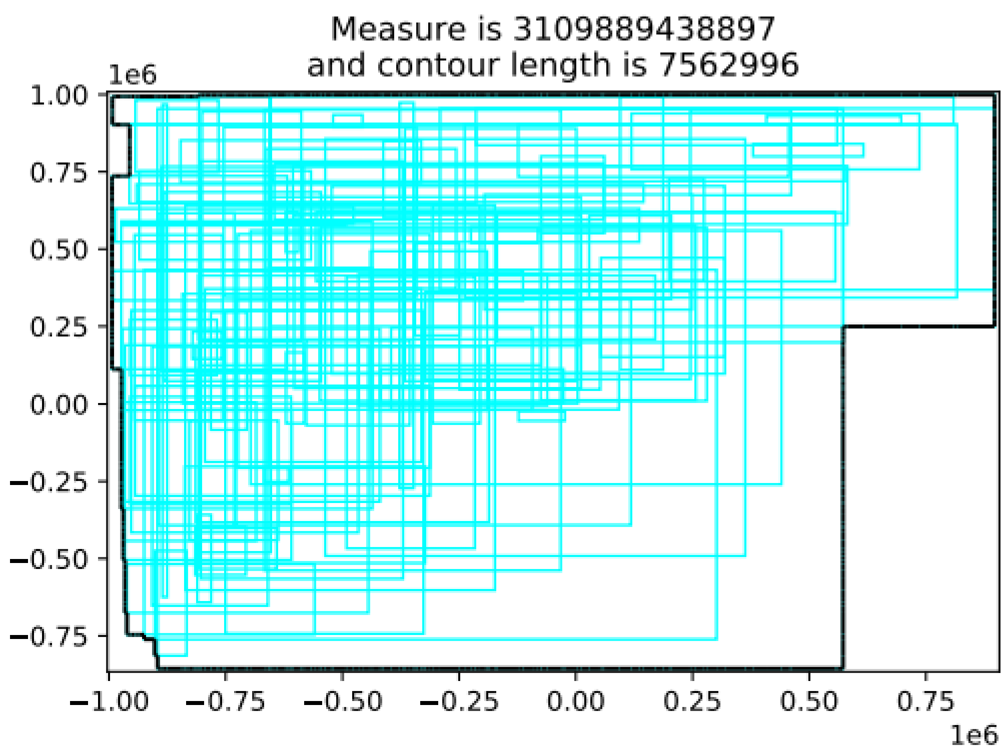
Test Case 13



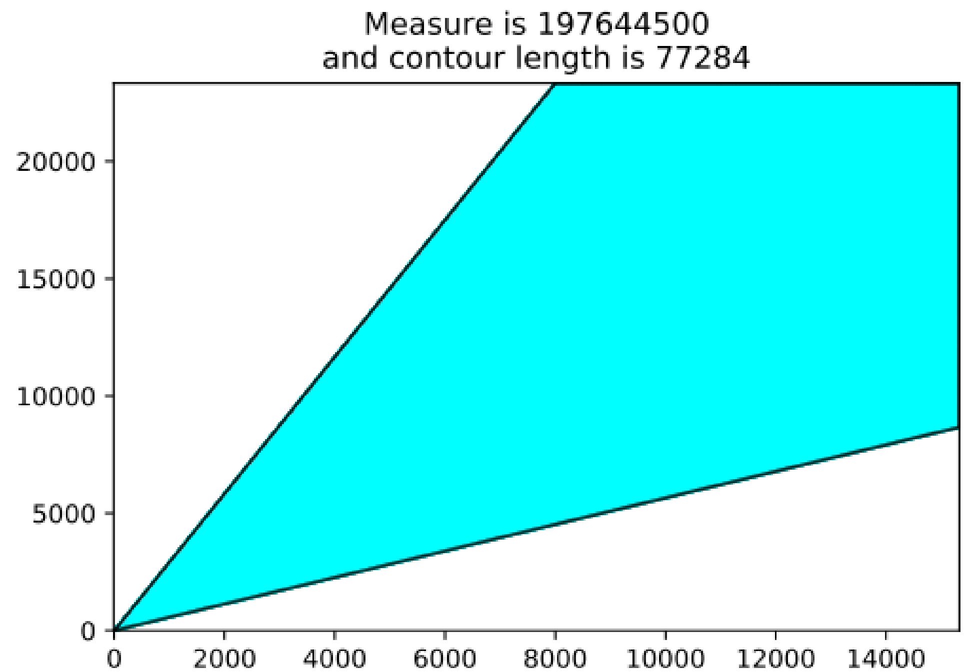
Test Case 14



Test Case 16



Test Case 17



General Discussion on the algorithm-

This is an alternate approach to finding the measure and contour of a set of iso rectangles. This algorithm was previously tackled using line sweep algorithms. A divide and conquer approach involves the breaking of complex problems into smaller ones and tackling them independently. A new technique called 'seperational representation' has been used which extends the applicability of the divide and conquer approach to orthogonal planar objects. Rectangles play a major role in applications like VLSI design, geography and computer graphics. Therefore, this field of study is pretty important.

The following is a concise version of the algorithm which was implemented by us. The main function that implements the divide and conquer algorithm is the function called STRIPES. This function, during the divide part, cuts the horizontal edges into two parts and sends each part to the STRIPES function recursively. This keeps happening till the base case of the recursion is reached. The base case of the recursion is when the number of edges in each half is one. The stripes are generated for the base case for both the halves. Then the conquer part of the algorithm comes into play. During the conquer phase, the following functions are called:

- Copy

- Blacken
- Concat

These functions effectively merge both the stripes of the individual halves into one stripe. After the entire recursion is executed, we utilize the final stripes to calculate the measure and contour of the set of iso-rectangles given as input.

It is important to note that the stripes structures for both measure and contour are very different. The measure has a component called x_measure and contour has a component called ctree. The measure and coconut pieces can be calculated as discussed in the paper.

Issues in Coding-

We needed to implement a divide and conquer algorithm which has the same complexity as the original line-sweep algorithm of O(nlogn). We have implemented using the C++ language. The STL library was used to implement many structures.

Visualization for large test cases is not very feasible as plotting greater than 50 rectangles fills the whole plot completely. It was difficult to match the complexity suggested by the paper and certain flexibilities were undertaken for a slightly more convenient implementation by using vectors instead of sets. Therefore, we checked the correctness of the algorithm for small cases, while visualizing them. The larger test cases could not be easily visualized.

SInce the structures for measure and the contour program are different, two different programs were written for them. They have to be implemented serially. Since visualization libraries are easier to use in python, the visualization was implemented in python. The C++ programs output the measure and the contour pieces to a file which are taken as input by the python file to plot the graphs.

Timing Analysis

The paper has proven that the algorithm is O(nlogn) complexity. The base case is of complexity O(1). Then the recurrence relation is as follows:

$$T(n) = O(1) + 2\,T(n/2) + O(n)$$

After solving this, we get the final complexity to be O(nlogn). Similarly, the space complexity is also O(nlogn).

The following is a table depicting the time taken to run with respect to the number of rectangles:

Rectangles	2	10	50	2000
Time(seconds)	~0	~0	~0	1 second

References

- https://www.doxygen.nl/index.html
- https://www.w3schools.com/html/
- https://matplotlib.org/
- https://en.cppreference.com/w/
- https://cp-algorithms.com/geometry/intersecting_segments.html
- https://www.hackerearth.com/practice/math/geometry/line-sweep-technique/tutorial/
- Güting, R. H. (1984). Optimal divide-and-conquer to compute measure and contour for a set of iso-rectangles. *Acta Informatica*, *21*(3), 271-291.