Calibre D2S R&D

C++ Internship Project

Rectangles Intersection Problem

Submitted By:
Shehab Hosny Ibrahim

Submitted To:
Eng. Michael Samy

1) Platform used:

- Operating System ➡ **Windows 10**
- (Integrated Development Environment) IDE ➡ **Visual Studio 2017**

2) Assumptions and Important Points:

- Touching Rectangles are not considered as Overlapping Rectangles
  I'm not sure about this point but I've done some reasearch and found that there is a difference betwwen Touching Rectangles and Overlapping Rectangles.

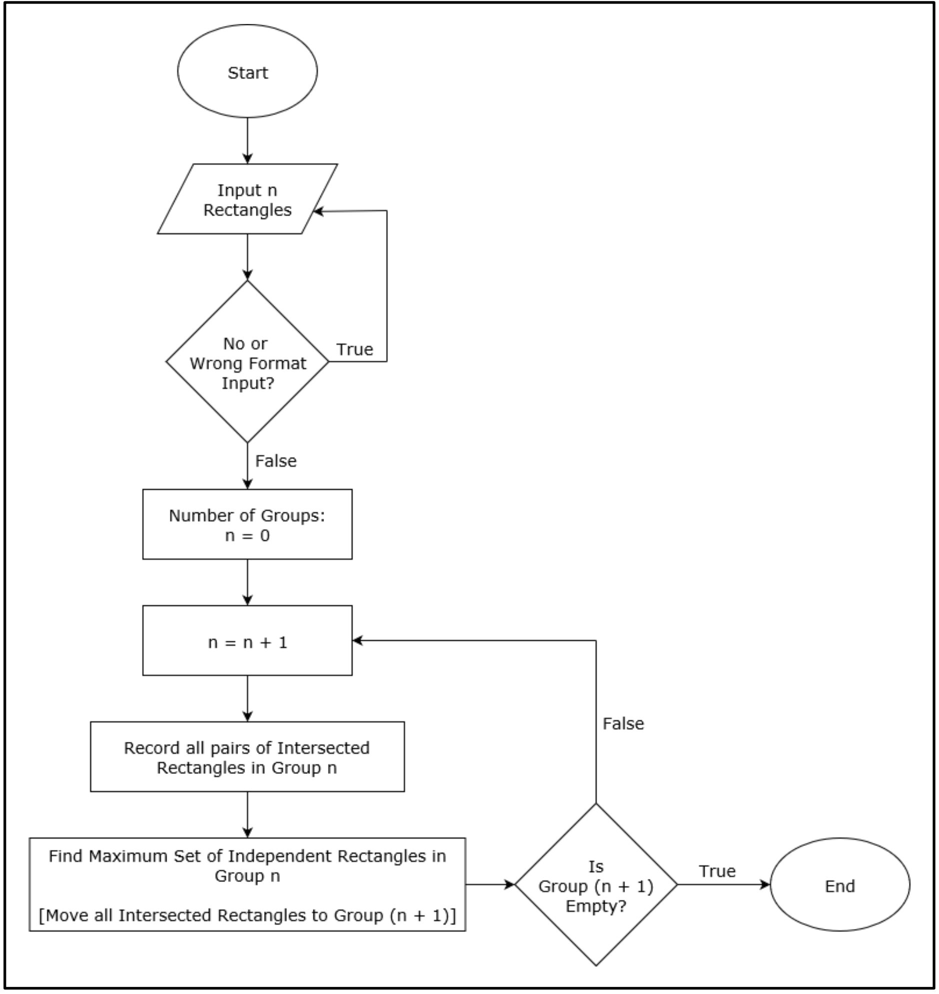
- I tried my best to run Dataset 16 but everytime I try to run I get nothing except runtime error. I was thinking about splitting the file and merging it later on but I think that this type of solution is not allowed.

- I used 2 different algorithms to solve this problem, and I compared the results.
  I used Brute Force as my first algorithm but it was not generating the minimum number of sets (except for Dataset 9 which was 12 Files). However, this algorithm is naive and time consuming. So, I thought about the Divide and Conquer algorithm. This algorithm was much better in all datasets except for dataset 9, where the output was more files (14 exactly) and it was a little bit time consuming. I think that because this problem is NP-hard so it may be because of the approximations involved in the algorithm.

3) Flowcharts of the main program and algorithms involved:

- Main Program:

❖ Flowchart Diagram:


Start
Input n Rectangles
No or Wrong Format Input?
True
False
Number of Groups: n = 0
n = n + 1
Record all pairs of Intersected Rectangles in Group n
Find Maximum Set of Independent Rectangles in Group n
[Move all Intersected Rectangles to Group (n + 1)]
Is Group (n + 1) Empty?
False
True
End

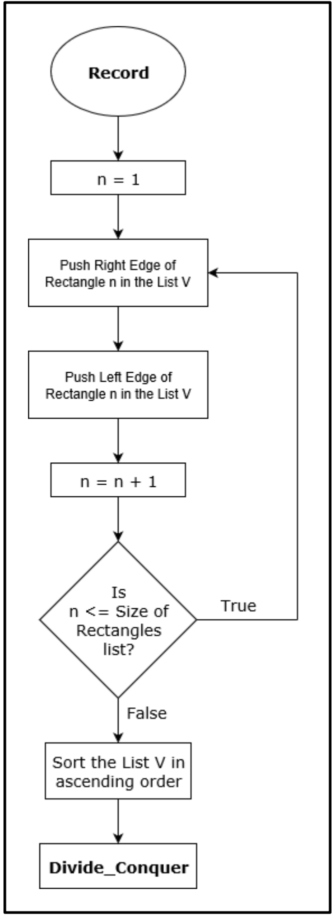
❖ Pseudocode:

```
procedure main()
    Read a file to Input n Rectangles
    Check that the file is not empty and that the format is correct
    Do {
            Record all pairs of intersected rectangles in Group n
            Find the Maximum Set of Independent rectangles in Group n
    } while (Group (n + 1) is not empty)
```

- **Algorithm** to Record all pairs of Intersected Rectangles in Group n:
  Since this problem involves massive input data, one of the best cases to solve such a problem can be achieved by using Divide and Conquer algorithm. This may to help us to finish this process in average time complexity of O(n logn) rather than O(n^2) by using Brute Force.
  In this algorithm, we usually use the following 3 steps:

  a) Divide: Break the given problem into subproblems of same type.
  b) Conquer: **Recursively** solve these subproblems.
  c) Combine: Appropriately combine the answers.

❖ Flowchart Diagram: procedure **Record**(S, n):

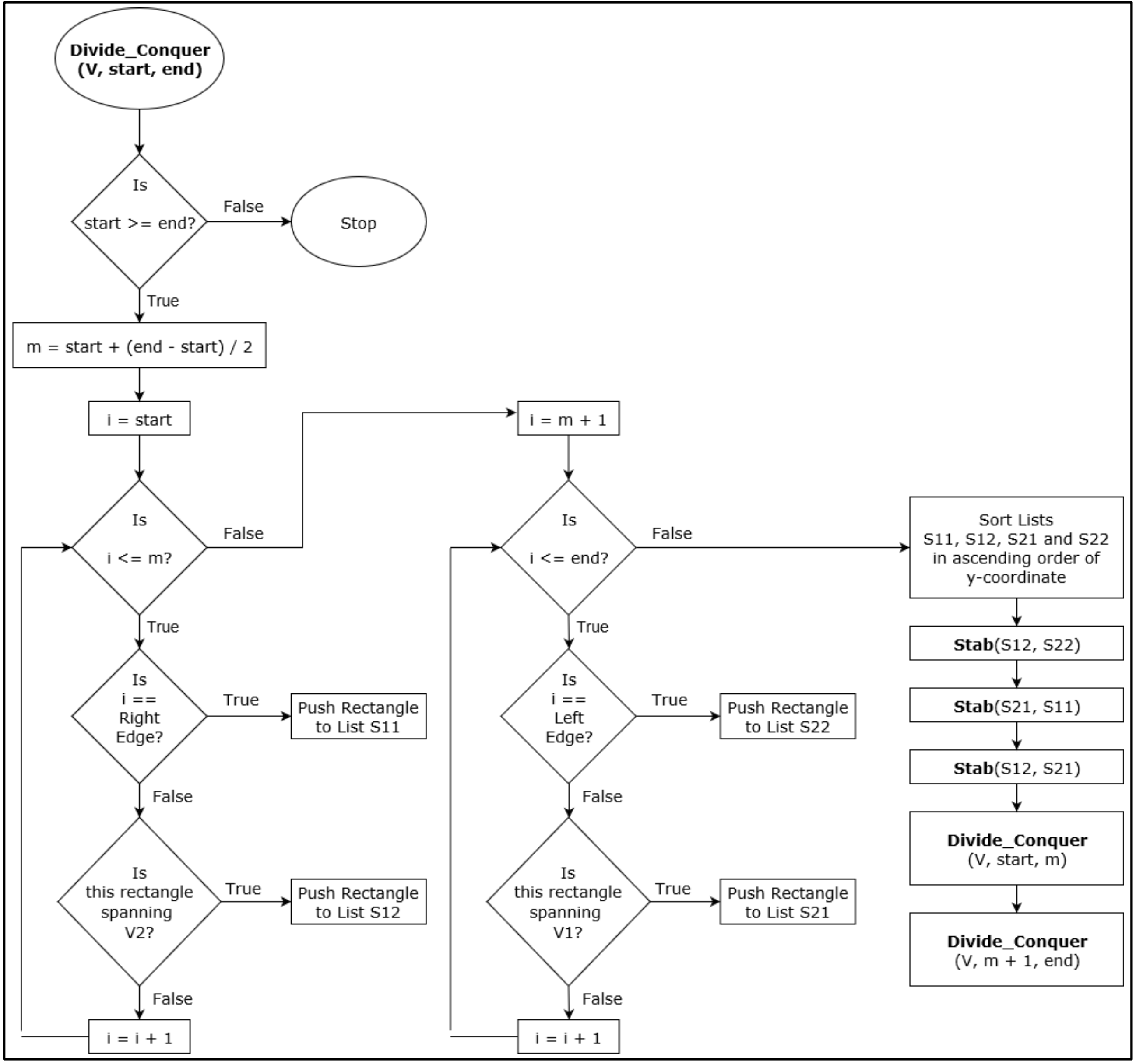

Record
n = 1
Push Right Edge of Rectangle n in the List V
Push Left Edge of Rectangle n in the List V
n = n + 1
Is n <= Size of Rectangles list?
True
False
Sort the List V in ascending order
Divide_Conquer

❖ Pseudocode: procedure **Record**(S, n):

Let S be the set of n iso-oriented rectangles from a specific dataset.

```
procedure Record(S, n)
    Let V be the list of x-coordinates of the 2n vertical edges in S.
    Sort V in ascending order of the x-coordinate values.
    Call procedure Divide_Conquer(V, 0, 2n).
```

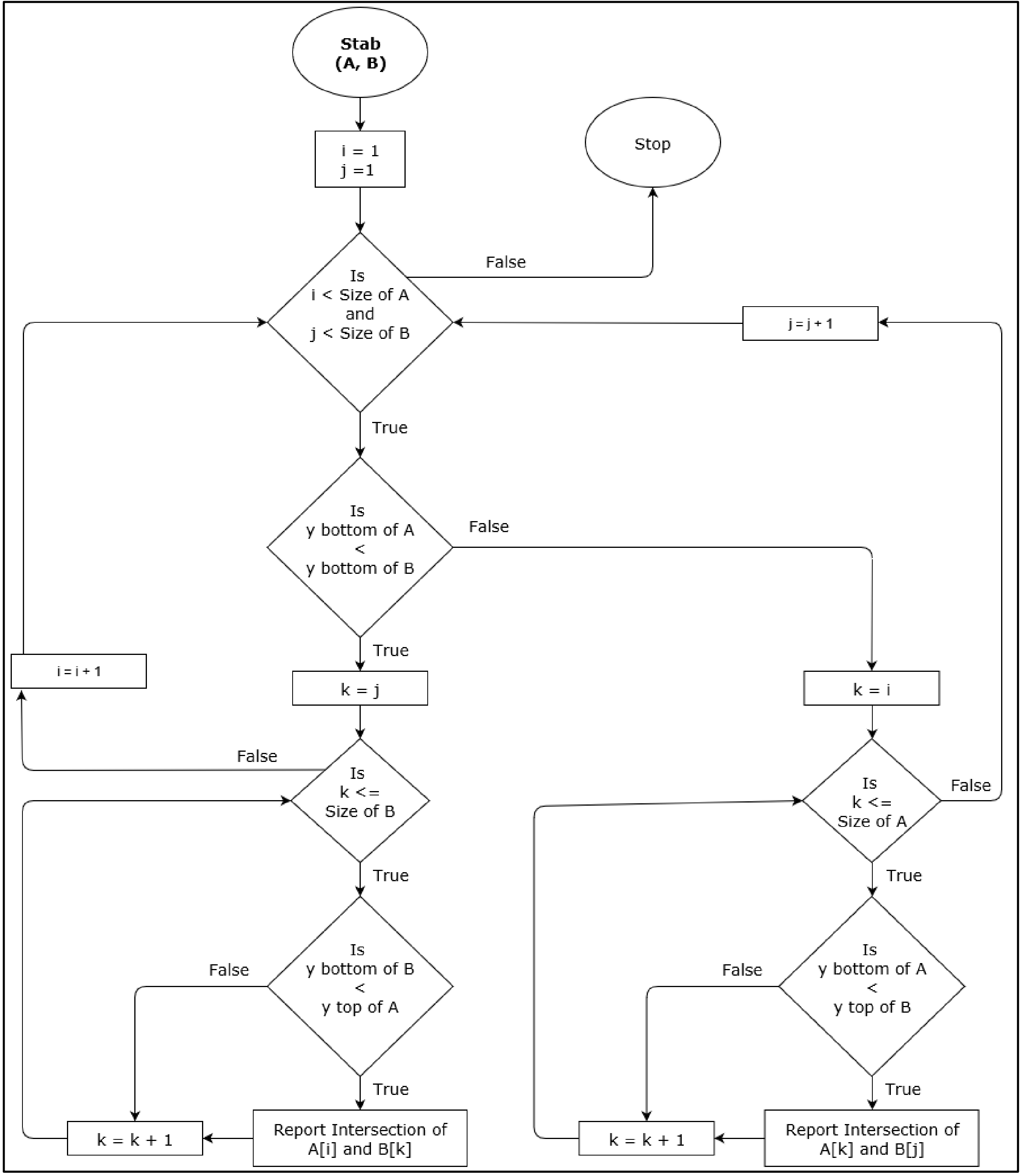
❖ Flowchart Diagram: procedure **Divide_Conquer**(V, start, end):



- Pseudocode: procedure **Divide_Conquer**(V, start, end):

```
procedure Divide_Conquer(V, start, end)
    if start ≥ end:
            return
    else
            m = start + (end - start) / 2;

            Let V1 be the first m elements of the set V.
            Let V2 be the rest of the elements remaining in the set V.

            Scan the list V1:
                    if the corresponding element is a Right Edge:
                            Push the corresponding rectangle to the list S11.
                    else if the corresponding element is to the right of V2:
                            Push the corresponding rectangle to the list S12.
            End Scanning V1.

            Scan the list V2:
                    if the corresponding element is a Left Edge:
                            Push the corresponding rectangle to the list S22.
                    else if the corresponding element is to the left of V1:
                            Push the corresponding rectangle to the list S21.
            End Scanning V2.

            Sort the list S11, S12, S22, and  S21 in ascending order of the bottom y-values

            Call procedure Stab(S12, S22).
            Call procedure Stab(S21, S11).
            Call procedure Stab(S12, S21).

            Call procedure Divide_Conquer(V1, start, m).
            Call procedure Divide_Conquer(V2, m + 1, end).
```

❖ Flowchart Diagram: procedure **Stab**(A, B):

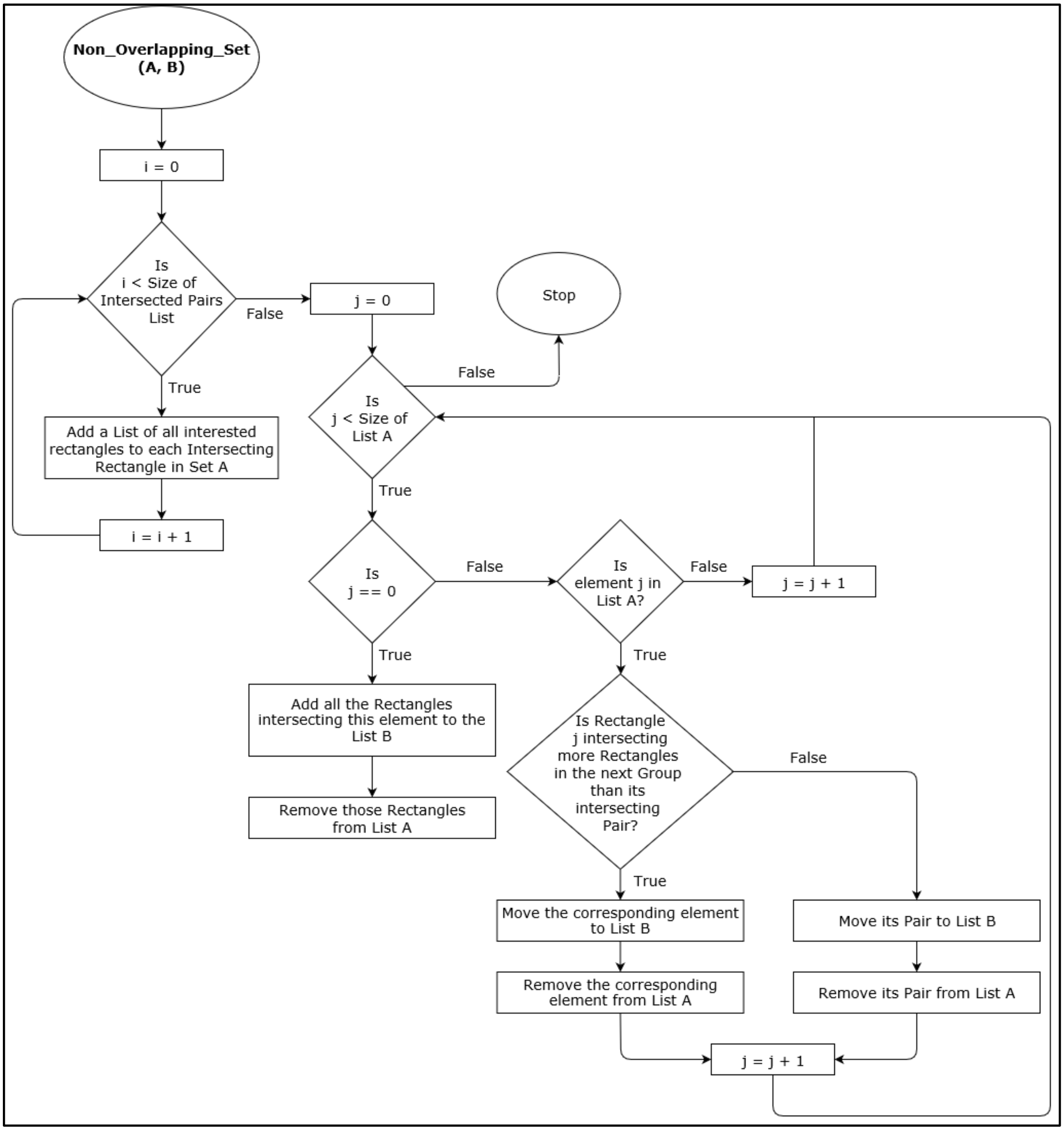


❖ Pseudocode: procedure **Stab**(A, B):

```
procedure Stab(A, B)
        i := 1 and j := 1
        while (Size of A ≥ i) and (Size of B ≥ j):
            if Bottom y-value of A[i] < Bottom y-value of B[j]:
                k := j
                while (Size of B ≥ k):
                    if Bottom y-value of B[k] < Top y-value of A[i]:
                        Report_Intersection(A[i], B[k])
                    k := k + 1
                    End if
                i := i + 1
                End while
            else
                k := i
                while (Size of A ≥ k):
                    if Bottom y-value of A[k] < Top y-value of B[j]:
                        Report_Intersection(A[k], B[j])
                    k := k + 1
                    End if
                j := j + 1
                End while

        End while
```

- **Algorithm** to Find the Maximum Set of Independent Rectangles in Group n:
  Since this is a NP-hard Problem, the approach of detecting the maximum set of non-overlapping rectangles in a single set can be base on a kind of approximation. My approximation was mainly based on picking TWO intersected rectangles in Group n and compare it to the rectangles added to the Group (n + 1) and then the rectangle with the fewer number of intersection with rectangles in Group (n + 1) should be removed from Group n and added to the Group (n + 1).

❖ Flowchart Diagram: procedure **Non_Overlapping_Set**(A, B):


Non_Overlapping_Set (A, B)
i = 0
Is i < Size of Intersected Pairs List
False
True
Add a List of all interested rectangles to each Intersecting Rectangle in Set A
i = i + 1
j = 0
Stop
False
Is j < Size of List A
True
Is j == 0
False
True
Add all the Rectangles intersecting this element to the List B
Remove those Rectangles from List A
Is element j in List A?
False
j = j + 1
True
Is Rectangle j intersecting more Rectangles in the next Group than its intersecting Pair?
False
True
Move the corresponding element to List B
Remove the corresponding element from List A
Move its Pair to List B
Remove its Pair from List A
j = j + 1

- Pseudocode: procedure **Non_Overlapping_Set**(A, B):

```
Let A be the set of n iso-oriented rectangles in Group n.
Let A be the set of n iso-oriented rectangles in Group (n + 1).

procedure Non_Overlapping_Set(A, B)
    Scan the list of Intersected Pairs:
            For each INTERSECTING rectangle in Set A add a list of all the Rectangles IDs
            intersecting A.
    End Scanning the list of Intersected Pairs.

    Scan the list A:
            if the corresponding element is the First Rectangle in the List:
                    Add all the Rectangles intersecting this element to the List B
                    Remove those Rectangles from List A
            End if.

            else if the corresponding element is in List A:
                    Scan the list of all Rectangles Intersecting this element:
                            if the Rectangle n in the corresponding List intersects more
                            Rectangles in List B than the corresponding element:
                                    Move the corresponding element to List B
                                    Remove the corresponding element from List A
                            End if

                            else
                                    Move the Rectangle n to List B
                                    Remove the Rectangle n from List A
                            End else
                    End Scanning.
            End else.
    End Scanning A.
```

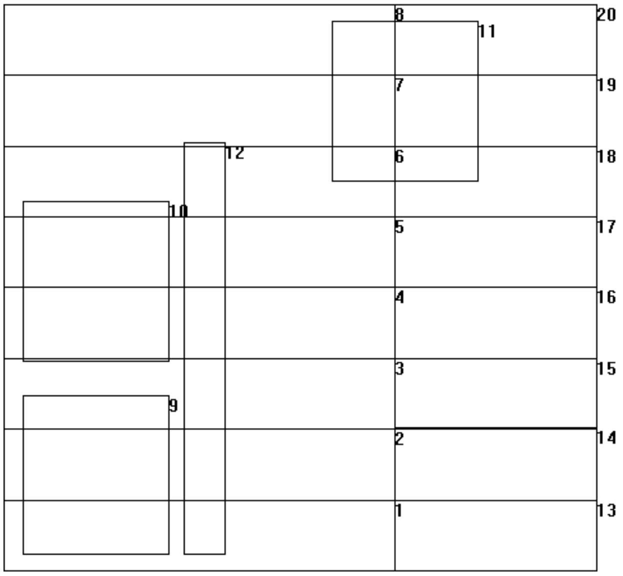
4) Table of Information:

Dataset	Number of Inputs	Number of Output Groups	Runtime (sec)	Memory usage (MB)
data_set_1.txt	5	1	0.005	0.9
data_set_2.txt	7	2	0.009	0.9
data_set_3.txt	20	2	0.017	0.9
data_set_4.txt	39	3	0.03	0.9
data_set_5.txt	77	3	0.048	0.9
data_set_6.txt	136	4	0.234	1
data_set_7.txt	216	5	0.443	1
data_set_8.txt	460	5	1.134	2
data_set_9.txt	741	14	7.523	4
data_set_10.txt	981	5	2.82	2
data_set_11.txt	5793	6	44.65	8
data_set_12.txt	6775	6	56.88	9
data_set_13.txt	7538	6	61.194	9
data_set_14.txt	8774	6	110.234	10
data_set_15.txt	9188	6	125.341	11
data_set_16.txt	25,000,000	-	-	-

5) Testing methodology:

In order to test my algorithm efficiently. I prepared a C++ Project (Including the Graphics Library ccc_win.h) in order to be able to plot the input file as well as the output files of the dataset. This will make it more clear to identify the minimum number of sets and run my program to compare it with my expectations. I provided the project folder and the source file for this program in the deliverable files. I also provided some screenshots of the program output below:

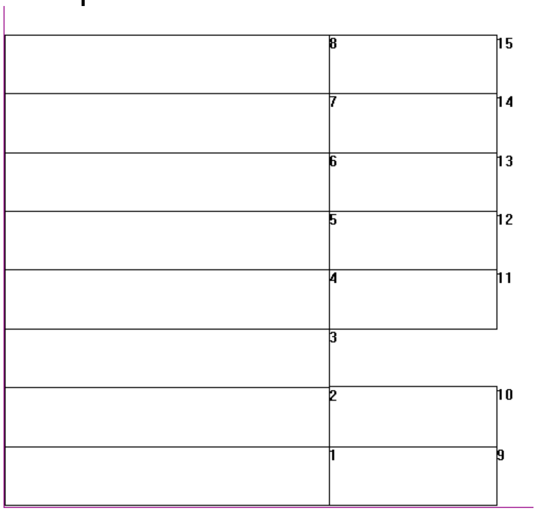
- Input plot of Dataset 3:



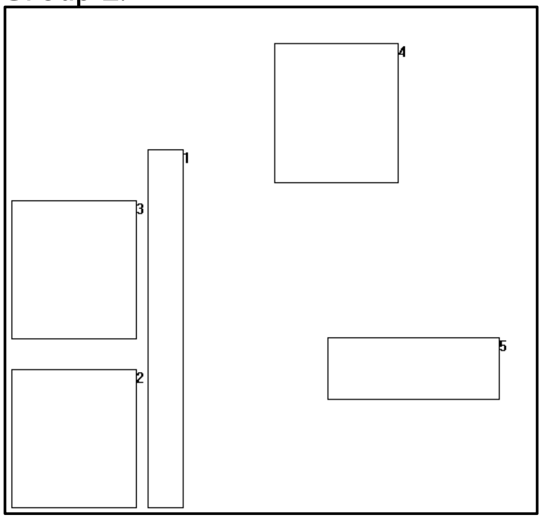
By Observing this figure, we can see the intersected rectangles (9, 10, 11, 12 and 15) and hence none of these rectangles intersect with each other. We can guess that the minimum number of non-overlapping groups will be 2. So, let's check it and plot the output files.

- Output Plots of Dataset 3:

Group 1:

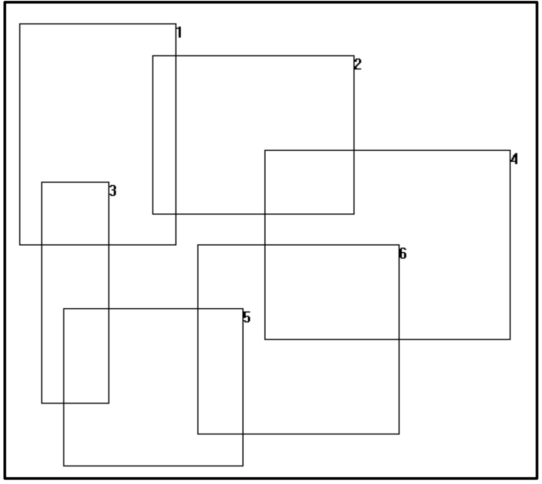


Group 2:

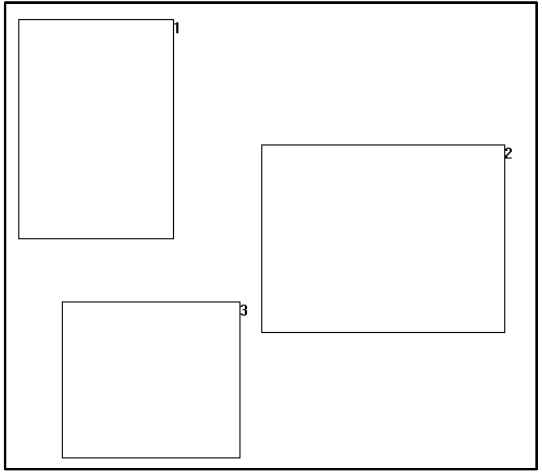


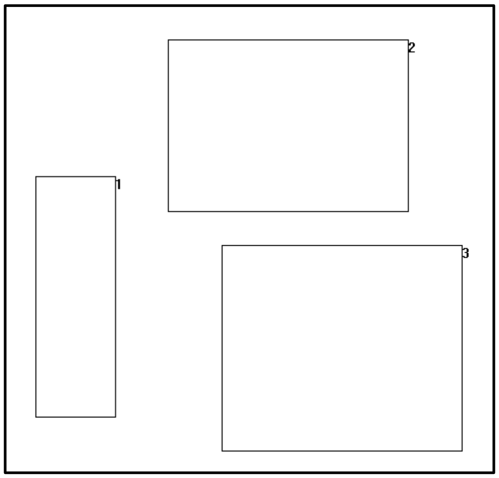
As shown, the output is as expected and thus. We tested our algorithm efficiently. Of course this will be more efficiently using 'small' dataset so that we can ensure our algorithm and implement it our larger datasets.

- Input plot of example given in project description:



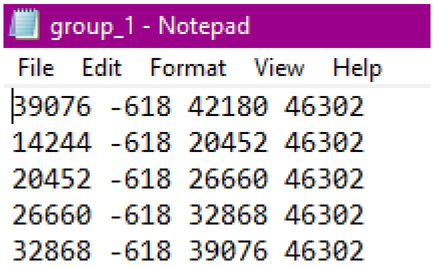
Output Plots of example given in project description:





6) Screen shots for the output of your program:

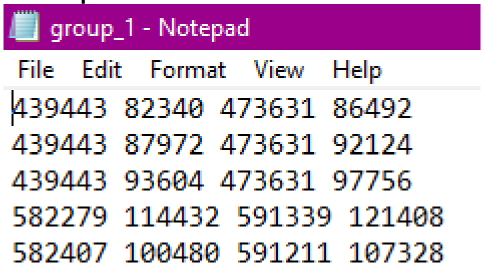
- Dataset 1:



```
39076 -618 42180 46302
14244 -618 20452 46302
20452 -618 26660 46302
26660 -618 32868 46302
32868 -618 39076 46302
```

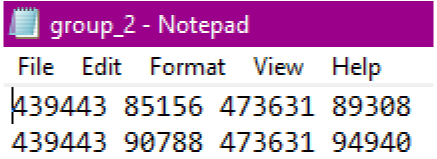
- Dataset 2:

Group 1:



```
439443 82340 473631 86492
439443 87972 473631 92124
439443 93604 473631 97756
582279 114432 591339 121408
582407 100480 591211 107328
```

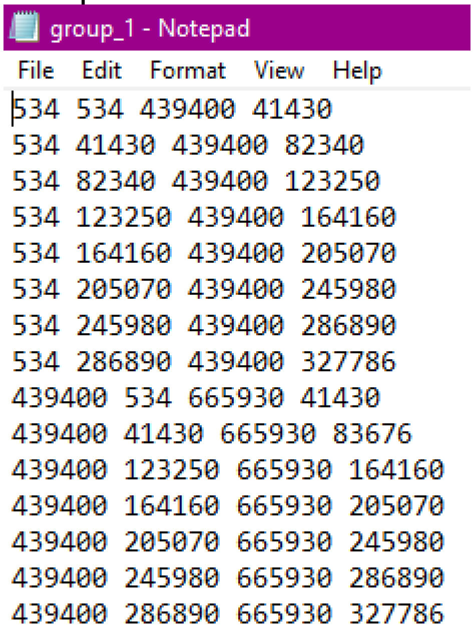
Group 2:



```
439443 85156 473631 89308
439443 90788 473631 94940
```

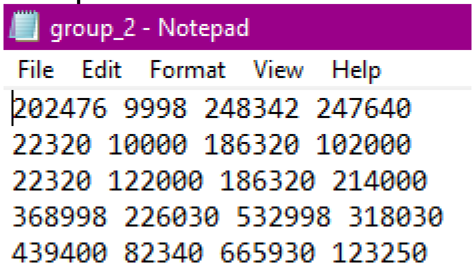
- Dataset 3:

Group 1:



```
534 534 439400 41430
534 41430 439400 82340
534 82340 439400 123250
534 123250 439400 164160
534 164160 439400 205070
534 205070 439400 245980
534 245980 439400 286890
534 286890 439400 327786
439400 534 665930 41430
439400 41430 665930 83676
439400 123250 665930 164160
439400 164160 665930 205070
439400 205070 665930 245980
439400 245980 665930 286890
439400 286890 665930 327786
```

Group 2:



```
202476 9998 248342 247640
22320 10000 186320 102000
22320 122000 186320 214000
368998 226030 532998 318030
439400 82340 665930 123250
```

7) Final Notes:

- I hope that you liked both of the programs and the Report.
- I read some papers in order to achieve the best algorithm for Rectangles Intersection problem.
- If you didn't like anything about my program or the report, I would love to hear your feedback so that I can improve in the long run.
- Finally, Thank you for taking the time to check my work and rate it. It was a great pleasure for me to work on this project.