```go
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

package main

type Coord struct {
  X, Y int
}

type Direction int

const (
  None Direction = iota - 1
  Up
  Down
  Left
  Right
)

// O(n^2) time | O(n) space - where n is the number of coordinates
func RectangleMania(coords []Coord) int {
  coordsTable := getCoordsTable(coords)
  return getRectangleCount(coords, coordsTable)
}

type CoordSet map[Coord]struct{}
type CoordsTable struct {
  Xs, Ys map[int]CoordSet
}

func getCoordsTable(coords []Coord) CoordsTable {
  table := CoordsTable{
    Xs: map[int]CoordSet{},
    Ys: map[int]CoordSet{},
  }
  for _, coord := range coords {
    x, y := coord.X, coord.Y
    if _, found := table.Xs[x]; !found {
      table.Xs[x] = CoordSet{}
    }
    table.Xs[x][coord] = struct{}{}
    if _, found := table.Ys[y]; !found {
      table.Ys[y] = CoordSet{}
    }
    table.Ys[y][coord] = struct{}{}
  }
  return table
}

func getRectangleCount(coords []Coord, coordsTable CoordsTable) int {
  count := 0
  for _, coord := range coords {
    lowerLeftY := coord.Y
    count += clockwiseCountRectangles(coord, coordsTable, Up, lowerLeftY)
  }
  return count
}

func clockwiseCountRectangles(coord Coord, coordsTable CoordsTable, direction Direction, lowerLeftY int) int {
  if direction == Down {
    relevantCoords := coordsTable.Xs[coord.X]
    for coord2 := range relevantCoords {
      lowerRightY := coord2.Y
      if lowerRightY == lowerLeftY {
        return 1
      }
    }
    return 0
  }

  if direction == Up {
    rectangleCount := 0
    relevantCoords := coordsTable.Xs[coord.X]
    for coord2 := range relevantCoords {
      if coord2.Y > coord.Y {
        rectangleCount += clockwiseCountRectangles(coord2, coordsTable, Right, lowerLeftY)
      }
    }
    return rectangleCount
  }

  if direction == Right {
    rectangleCount := 0
    relevantCoords := coordsTable.Ys[coord.Y]
    for coord2 := range relevantCoords {
      if coord2.X > coord.X {
        rectangleCount += clockwiseCountRectangles(coord2, coordsTable, Down, lowerLeftY)
      }
    }
    return rectangleCount
  }
  return 0
}

```