```
using PlutoUI
```

```
parse_line (generic function with 1 method)
```

```
function parse_line(line)
    return (split(line, " -> ")
            .|> x -> split(x, ",")
            .|> n -> parse(Int16, n))
end
```

```
points_in_between (generic function with 1 method)
```

```
function points_in_between(p1, p2)
    x1, y1 = p1[1], p1[2]
    x2, y2 = p2[1], p2[2]

    if x1 == x2
        return y1 < y2 ? tuple.(x1, y1:y2) : tuple.(x1, y2:y1)
    elseif y1 == y2
        return x1 < x2 ? tuple.(x1:x2, y1) : tuple.(x2:x1, y1)
    else
        # Its a diagonal, solves #Problem 2
        xrange = x1:(x1 < x2 ? 1 : -1):x2
        yrange = y1:(y1 < y2 ? 1 : -1):y2
        return tuple.(xrange, yrange)
    end
end
```

Problem 1

```
is_horizontal_or_vertical (generic function with 1 method)
```

```
function is_horizontal_or_vertical(locs)
    p1, p2 = locs[1], locs[2]

    return p1[1] == p2[1] || p1[2] == p2[2]
end
```

```
get_dangerous_areas (generic function with 1 method)
```

```
function get_dangerous_areas(vents_locs; LW = 1000)
    vent_points = zeros(Int16, LW*LW)

    # Removed dict as it was taking upto 1.0M allocation and 30 MiB.
    # With array its taking only 2K allocation and 5.2 MiB
    # vent_points = Dict()

    function update_point_value!(p)
        key = p[2] * LW + p[1]
        vent_points[key] += 1
    end

    for locs in vents_locs
        (points_in_between(locs[1], locs[2])
        .|> update_point_value!)
    end

    return count(n -> n >= 2, values(vent_points))
end
```

```
7436
```

```
  0.000926 seconds (1.29 k allocations: 3.298 MiB)
```

```
with_terminal() do
    open("./Day5/prob_input.txt") do io
         vents_locs = [parse_line(line) for line in eachline(io)]
         vents_locs = filter(is_horizontal_or_vertical, vents_locs)
         @time get_dangerous_areas(vents_locs)
    end
end
```

Problem 2

```
21104
```

```
  0.001603 seconds (2.00 k allocations: 5.261 MiB)
```

```
with_terminal() do
    open("./Day5/prob_input.txt") do io
         vents_locs = [parse_line(line) for line in eachline(io)]
         @time get_dangerous_areas(vents_locs)
    end
end
```