Team 7

IP Rules Regulator

Li Li
Yudong Cao
Yong Chen

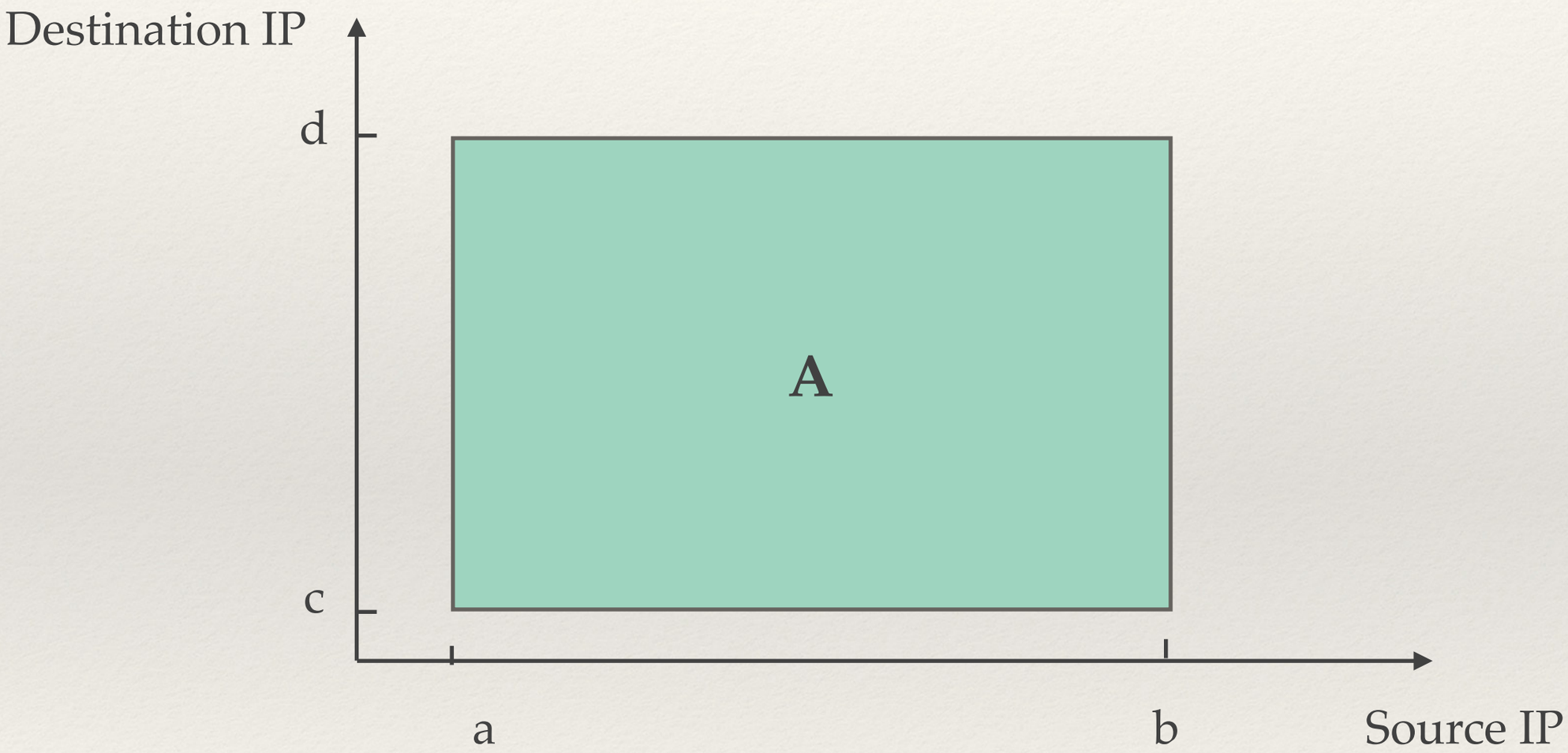
Problem Definition

Source IP	Destination IP		Action
a.b.c.d/k	1.b.c.d/k	k<=24	allow/block

Problem Definition

1. When a packet arrives at the filter, we look for the first rule that matches the sourceIP and destIP fields of the incoming packet. The action field determines whether the packet is blocked or forwarded to the destination.
2. Filter rules in practice also specify a "port number" field which is used to block specific protocols. We will ignore that for our project.

Data Structure



Data Structure

Int32

a.b.c.d/k

Int32

Data Structure

Rules Class

index

Source IP Start

Source IP End

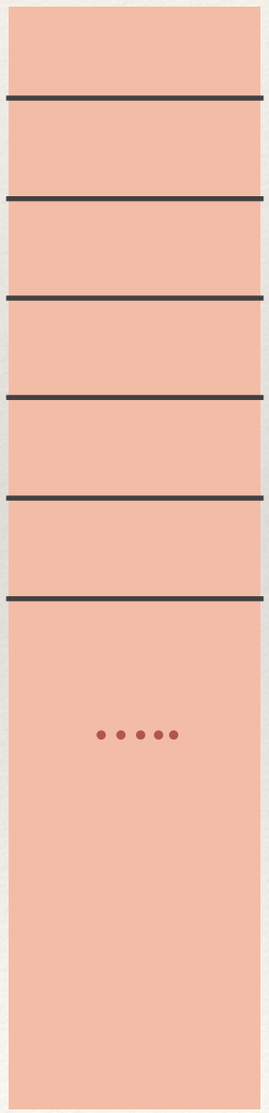
Destination IP Start

Destination IP End

Action

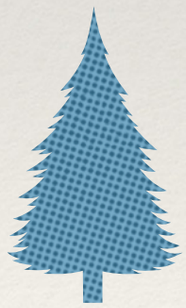
Data Structure

List<rectangles>



Source IP Range ETree

Destination IP Range ETree



Revers-Source IP Range ETree

O(nlogn)

n: # of rectangles

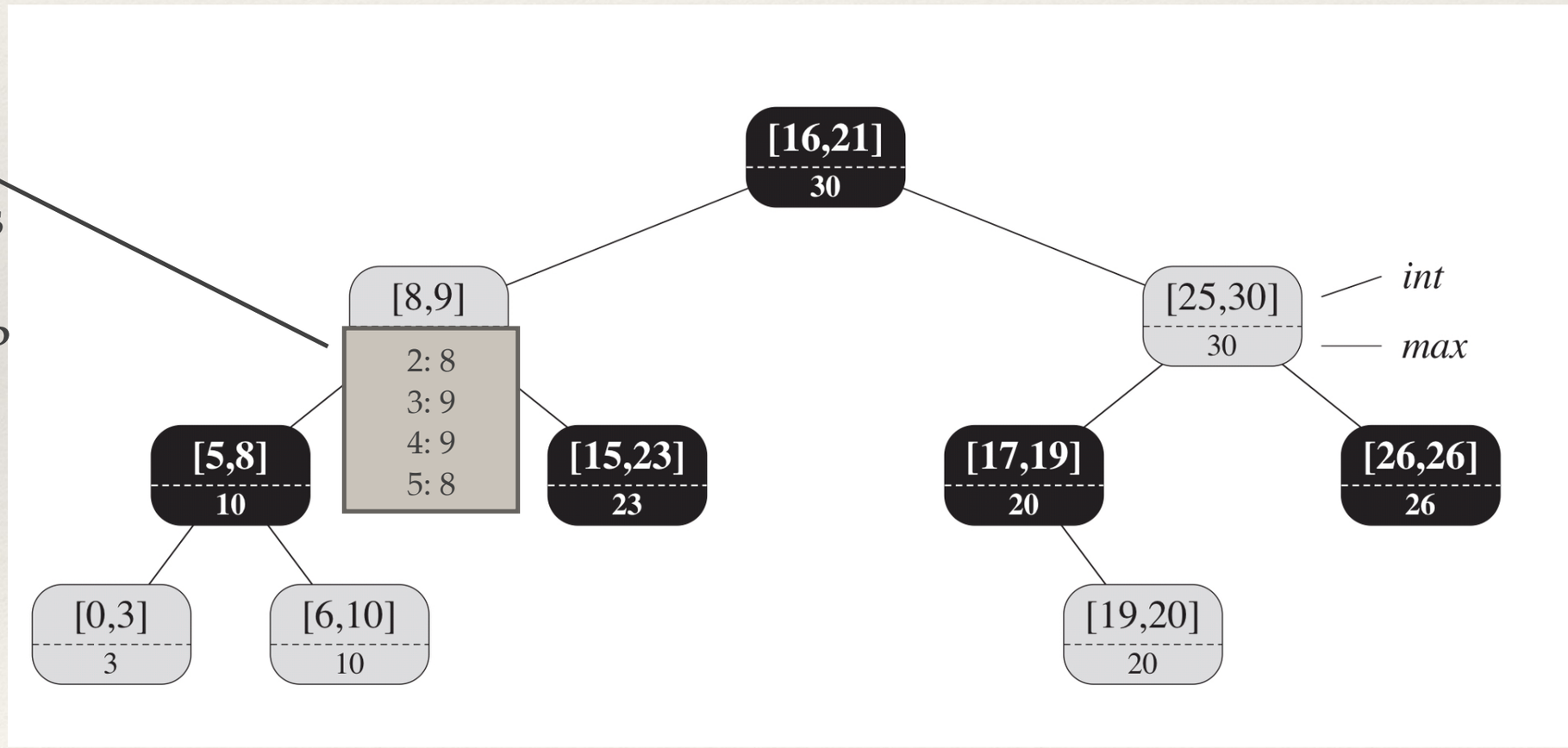
Extended Tree: interval search tree

Source IP Range ETree

2: 8

2: indexes of rectangles

8: end point of source IP



When adding a new rule:
1.Find all intervals(a little trick)

List<rectangles>

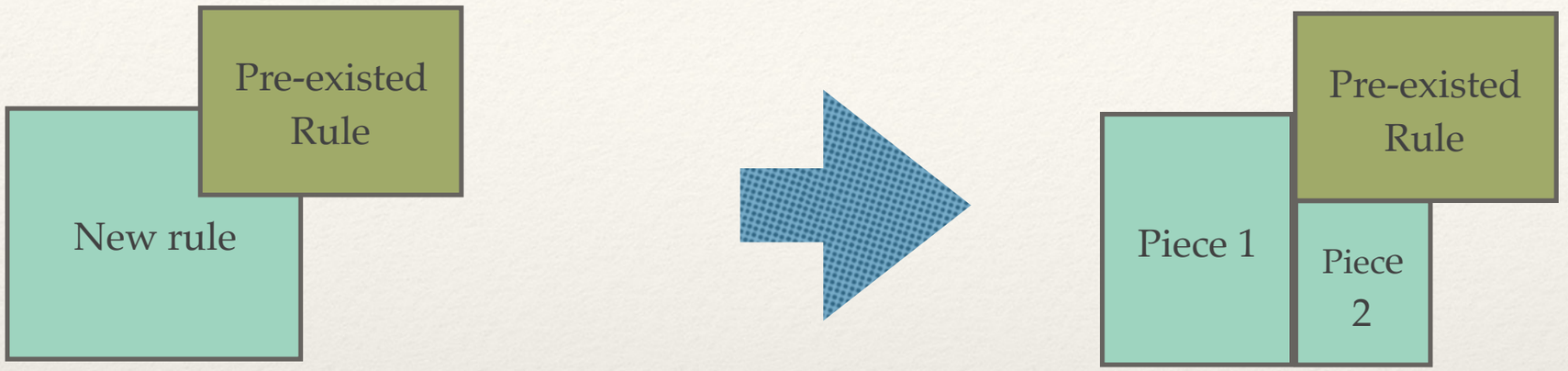
Source IP Range ETree

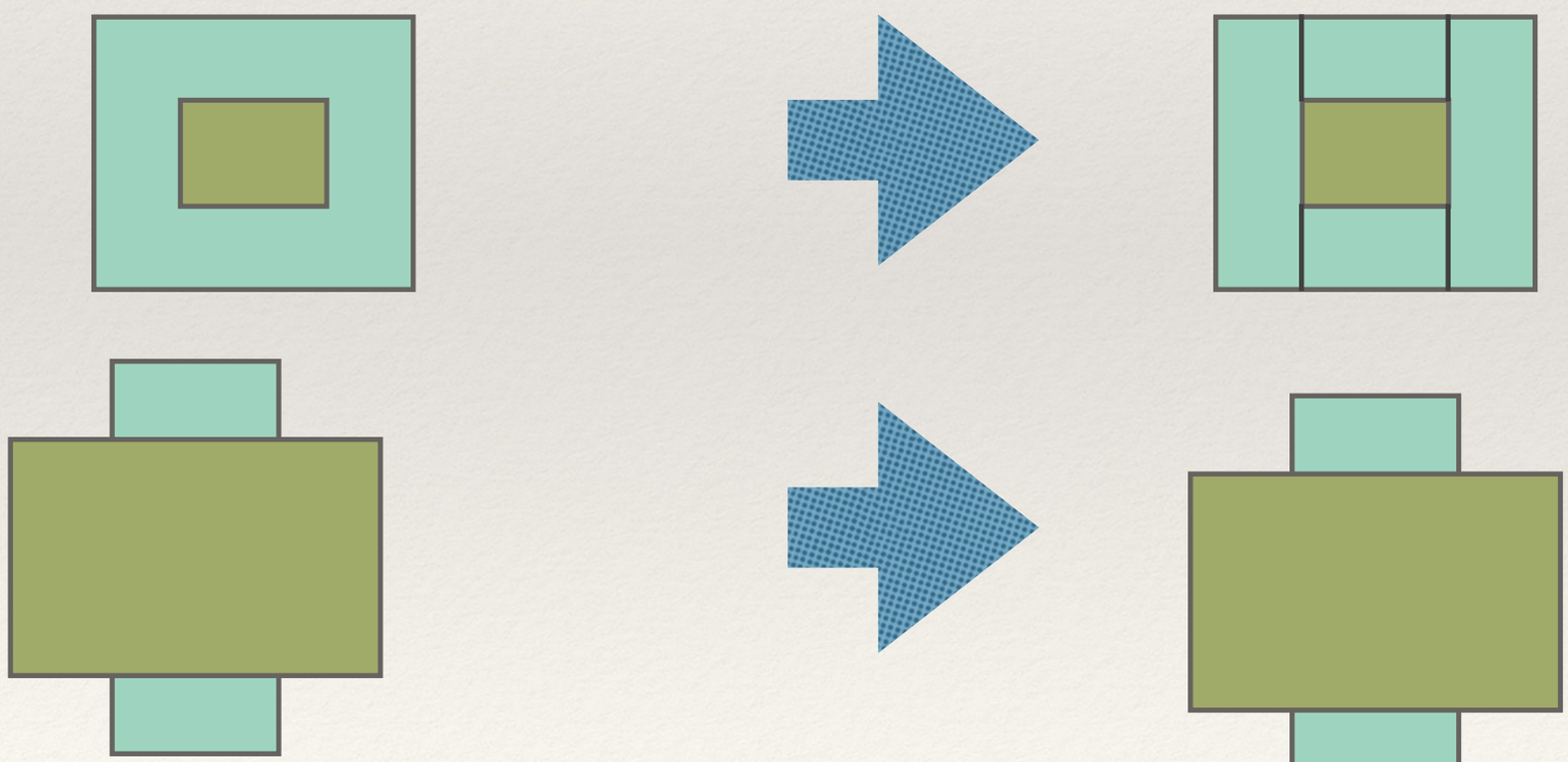
Destination IP Range ETree

Suppose we found x SourceRange intervals, and y DestRange intervals

1. Mark these x rectangles
2. Check if these y rectangles are marked. If marked, report
3. Unmark those x rectangles

When adding a new rule:
2.Breaking into pieces(scan from left to right)





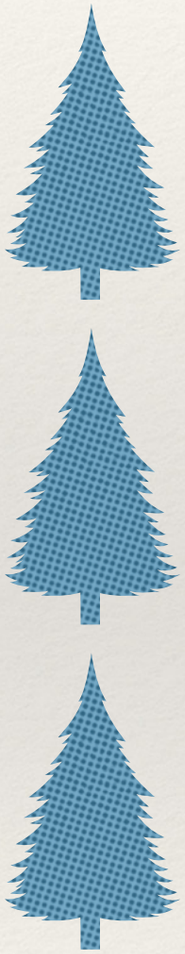
The goal of Step 2

All these pieces of rules would have no intervals with any Pre-existed rules at all.

Also, if no pieces generated, then the original new rule is redundant.

When adding a new rule :

3.1 For each piece, find its neighbors at right or left



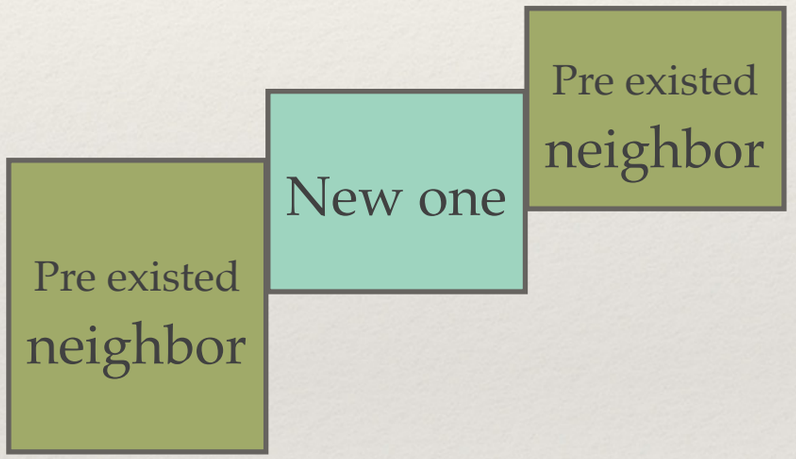
Source IP Range Tree

Right neighbor

Destination IP Range Tree

Revers-Source IP Range Tree

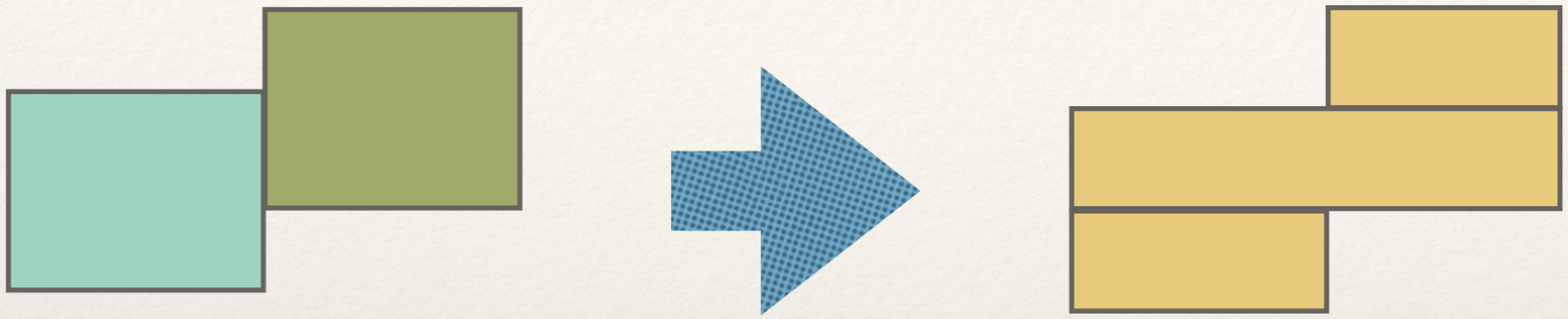
Left neighbor

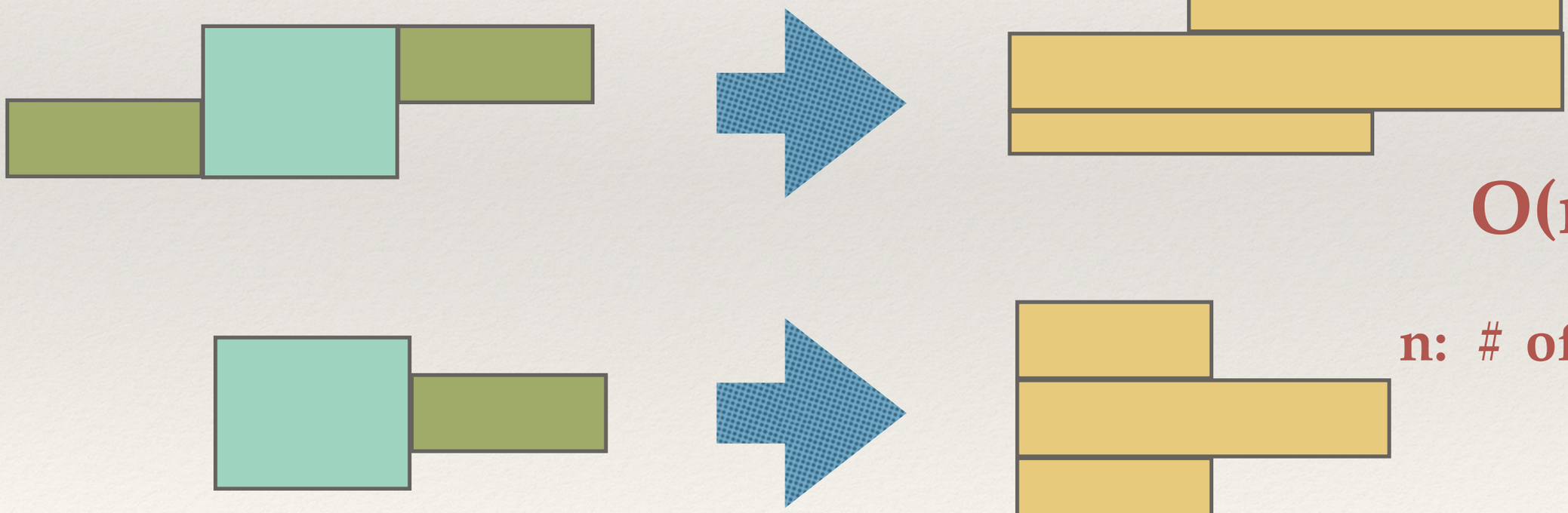


Find neighbors status takes :O(logn)

When adding a new rule:

3.2. For each piece of rule, break and combine with neighbors



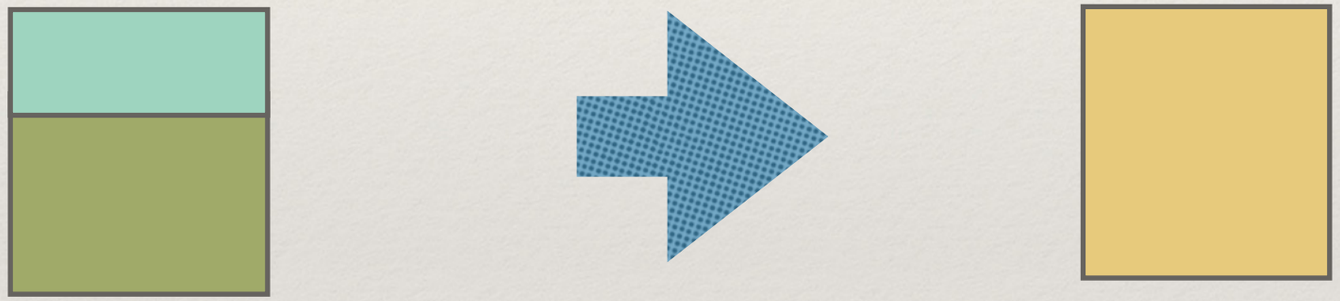


O(nlogn)

n: # of neighbors

When adding a new rule:

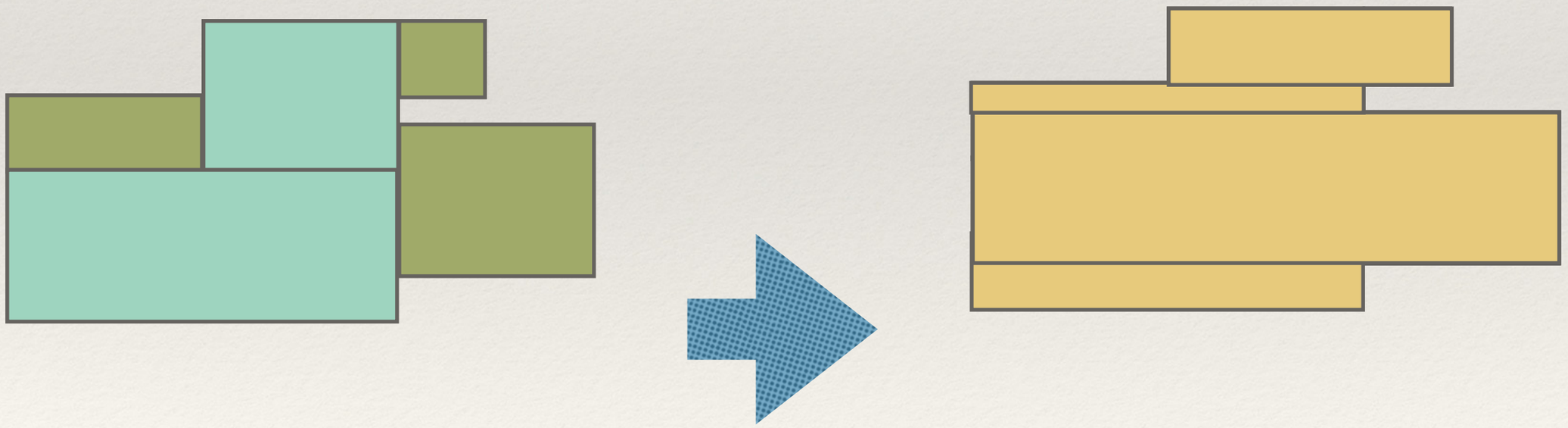
3.3. For each piece of rule, check if could have a vertical combination

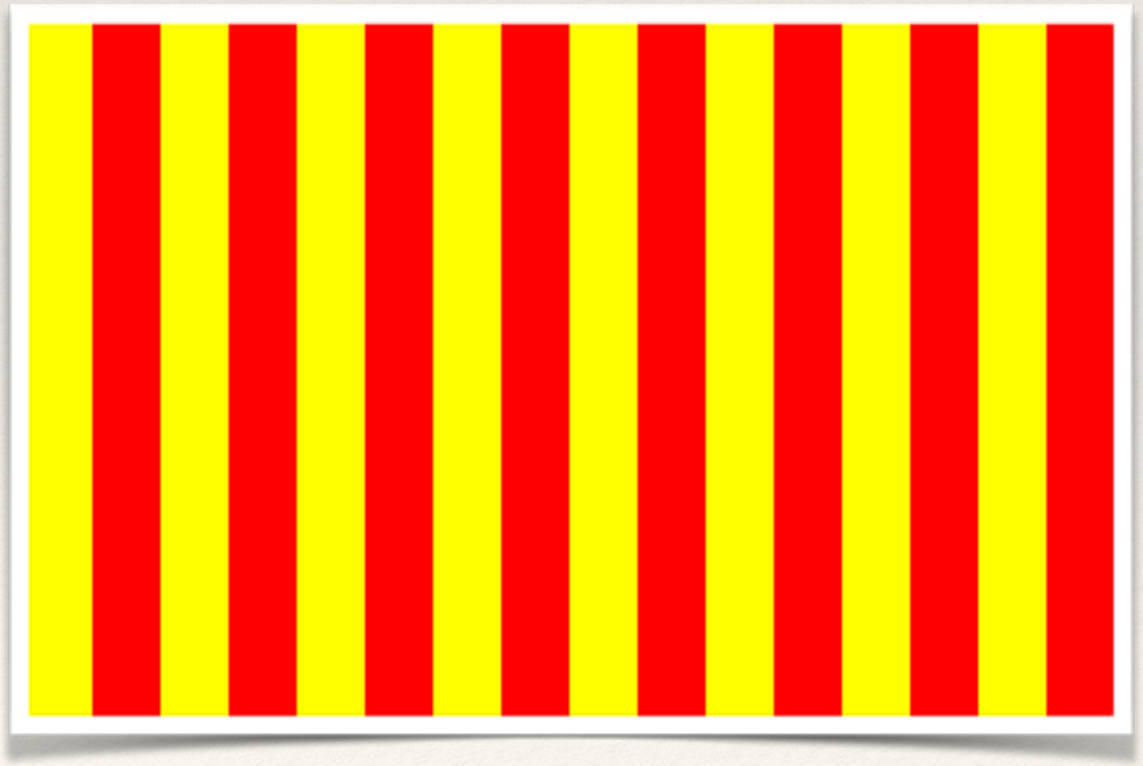


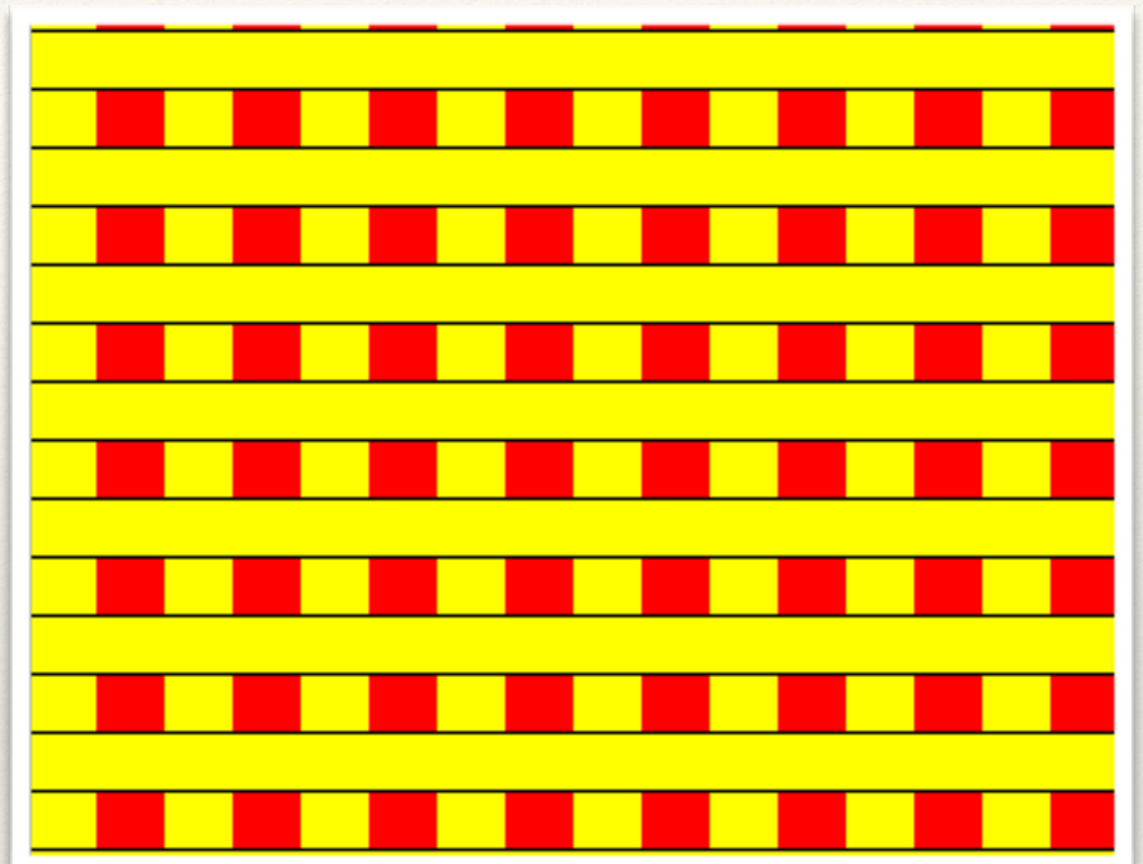
The goal of Step 3

All these breaking new rule into pieces, try to combine with left and right neighbors, then try to combine vertically, is to make sure

That no matter how bizzard the range might look,
it will be saved with the same group of rectangles.
Therefore we can check if two group of rules are the same.







Thank You!