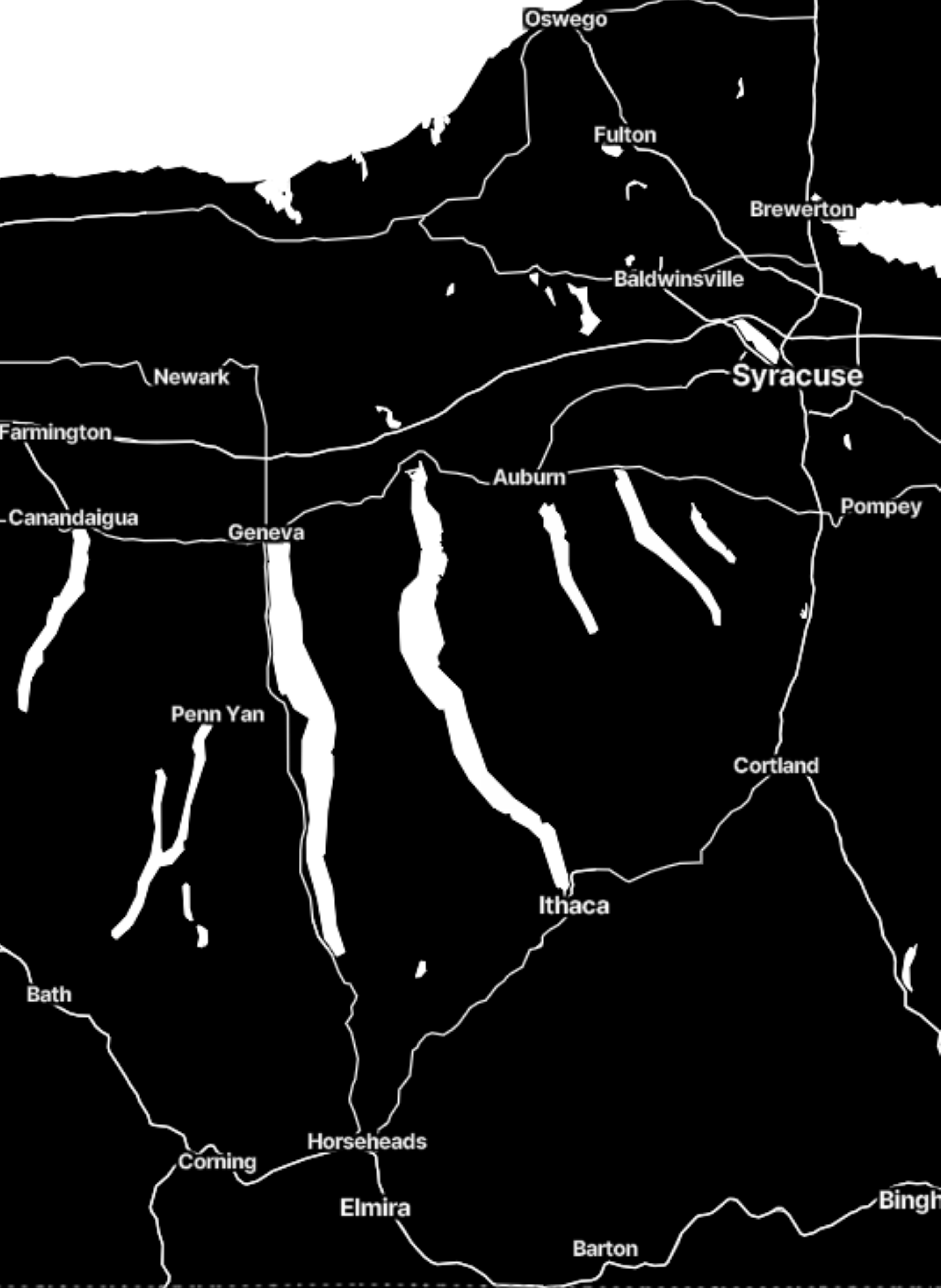
Formal Abstractions for Packet Scheduling

A. Mohan, Y. Liu, N. Foster, T. Kappé, D. Kozen

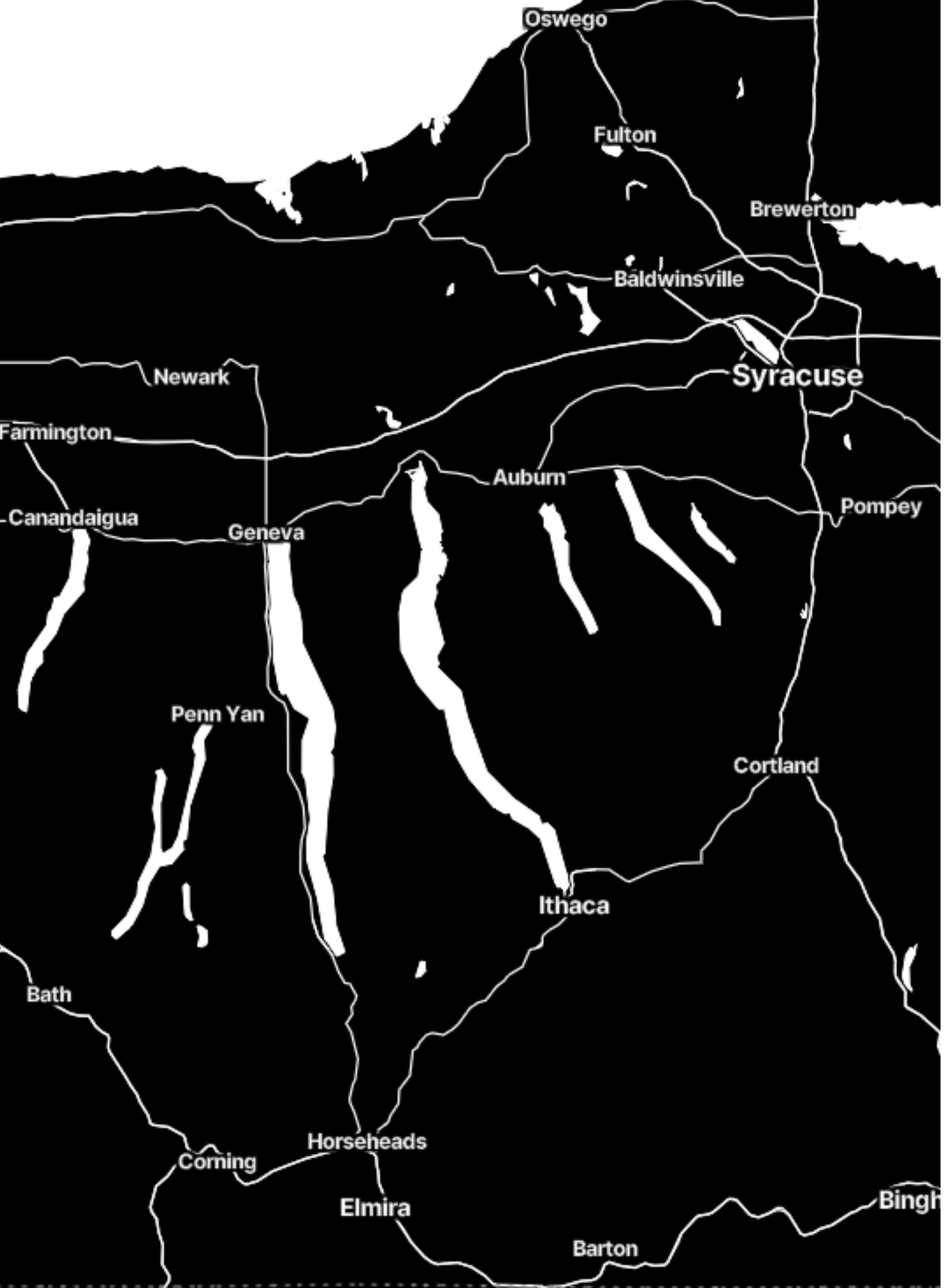
Formal Abstractions for Packet Scheduling

A. Mohan, Y. Liu, N. Foster, T. Kappé, D. Kozen

Has let me use his slides!

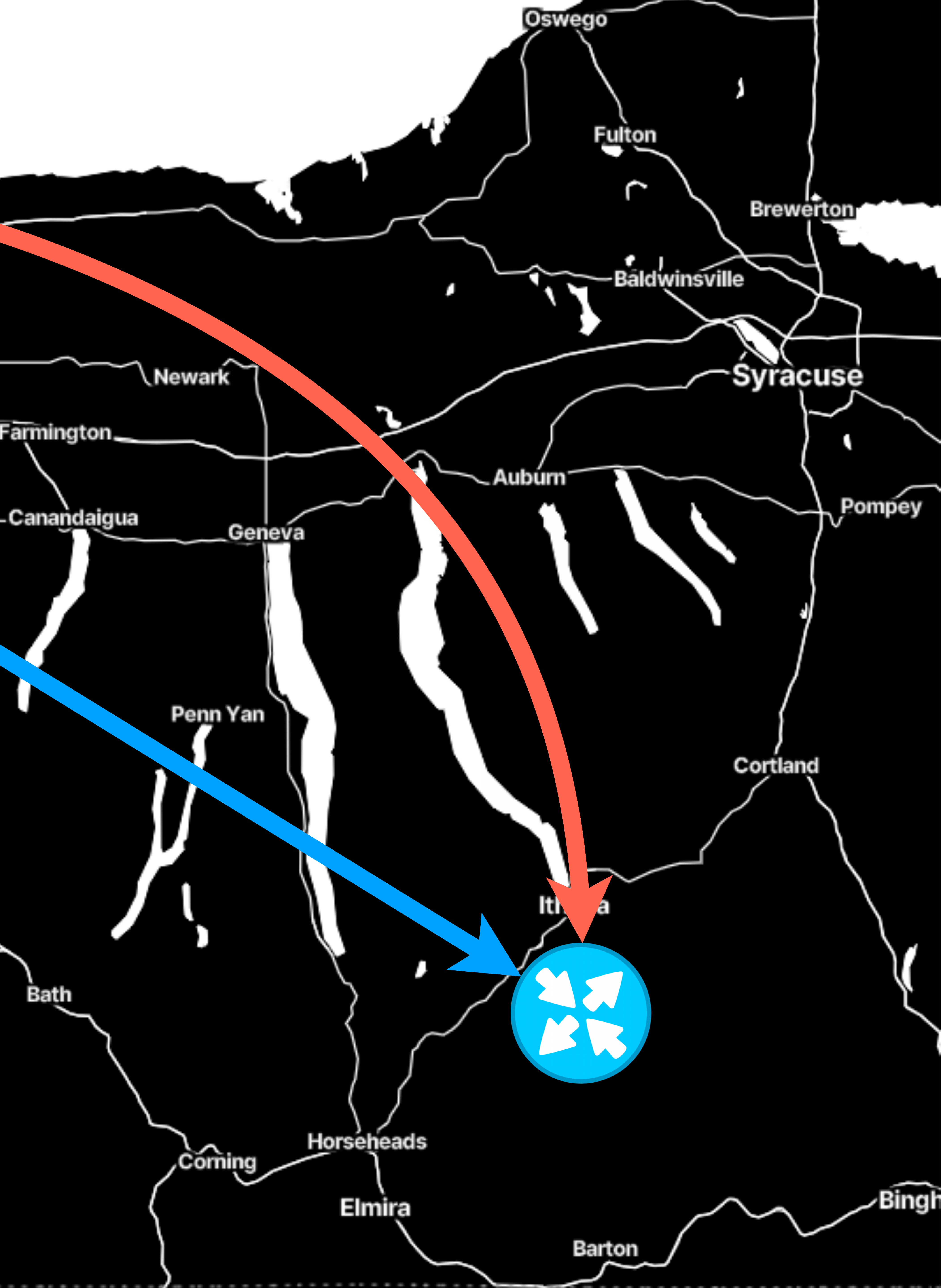


Software-defined networking made networks programmable.



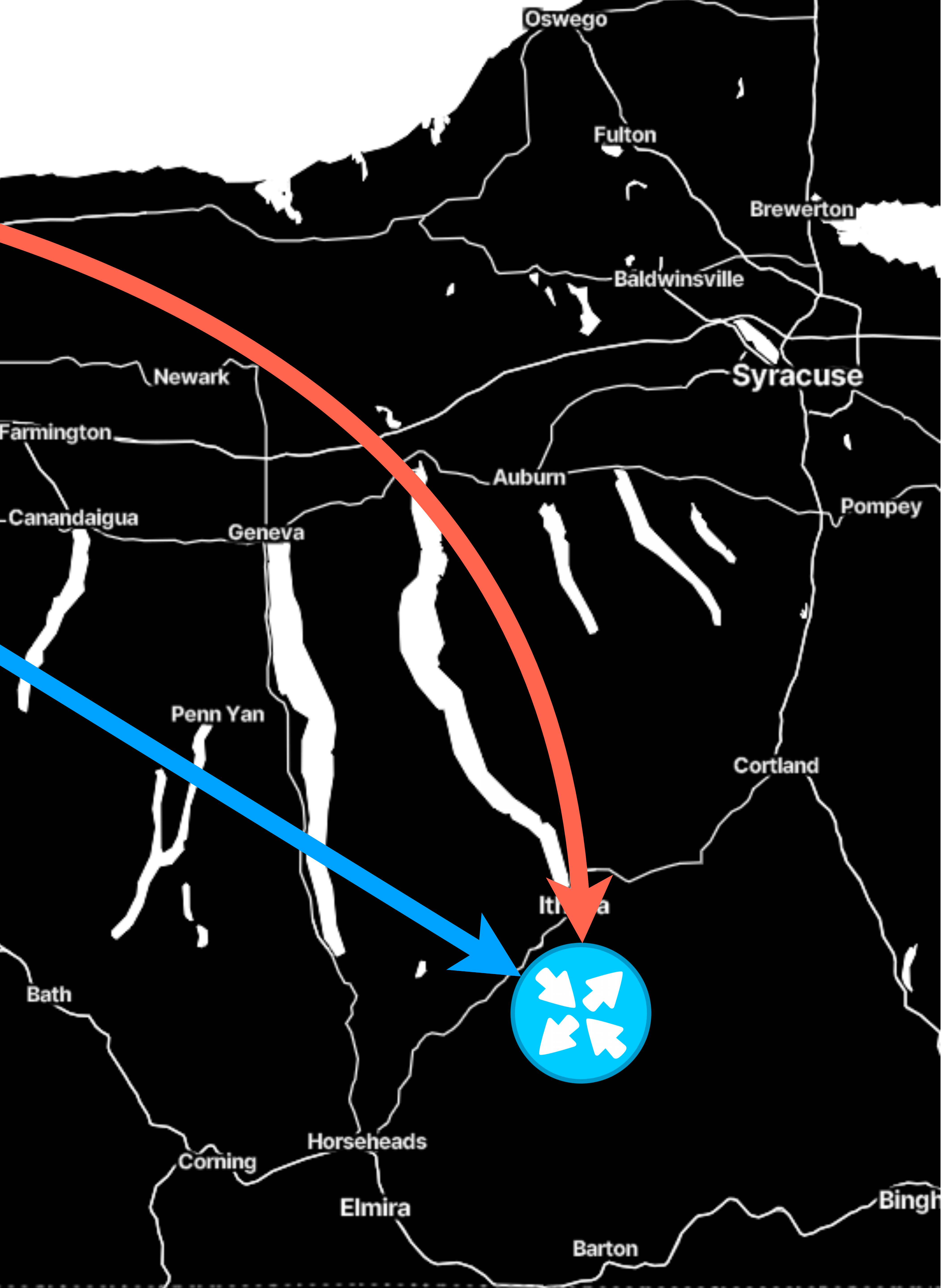
Software-defined networking made networks programmable.

We want control over *packet scheduling.*



Software-defined networking made networks programmable.

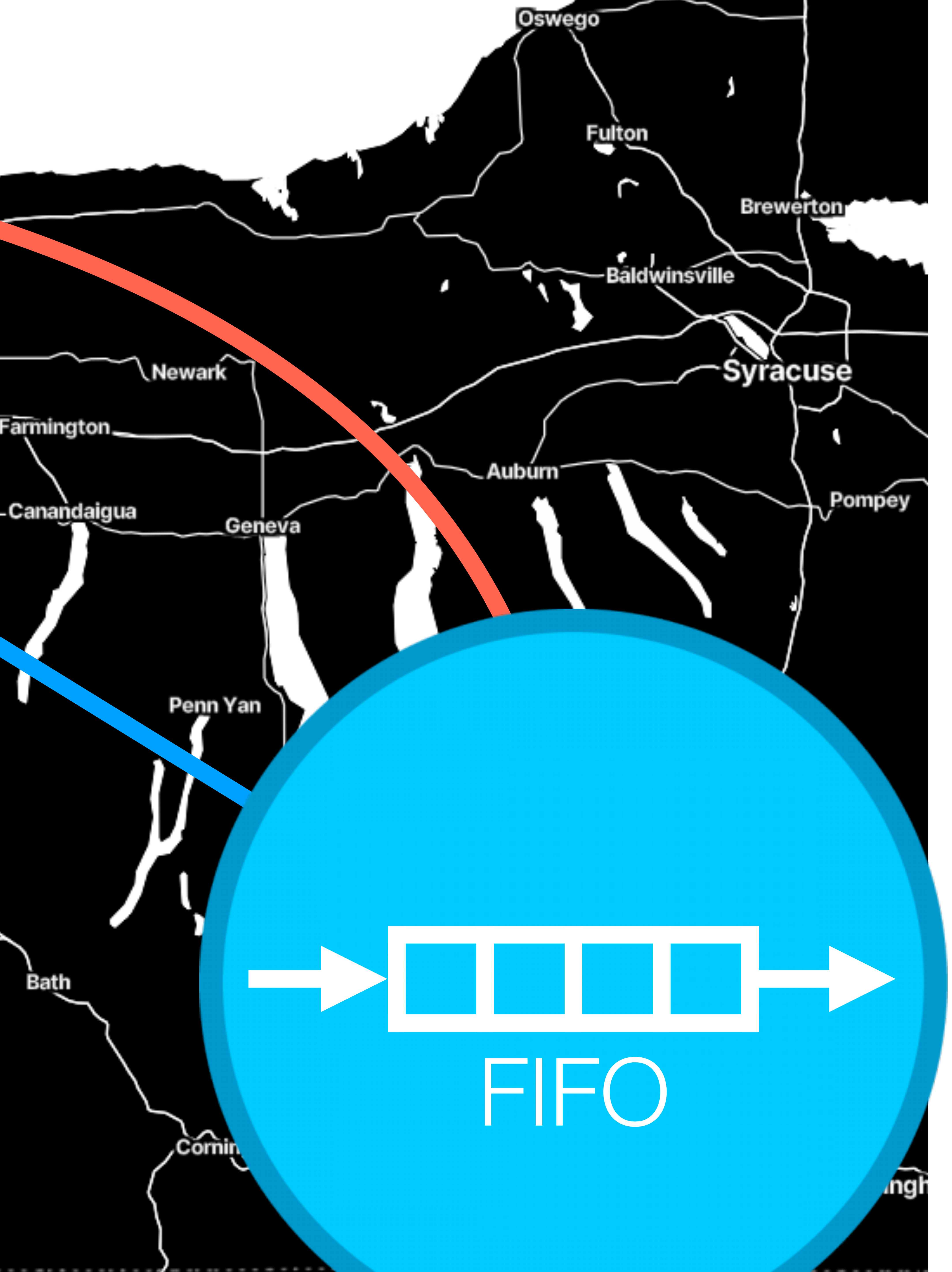
We want control over *packet scheduling.*



Software-defined networking made networks programmable.

We want control over *packet scheduling.*

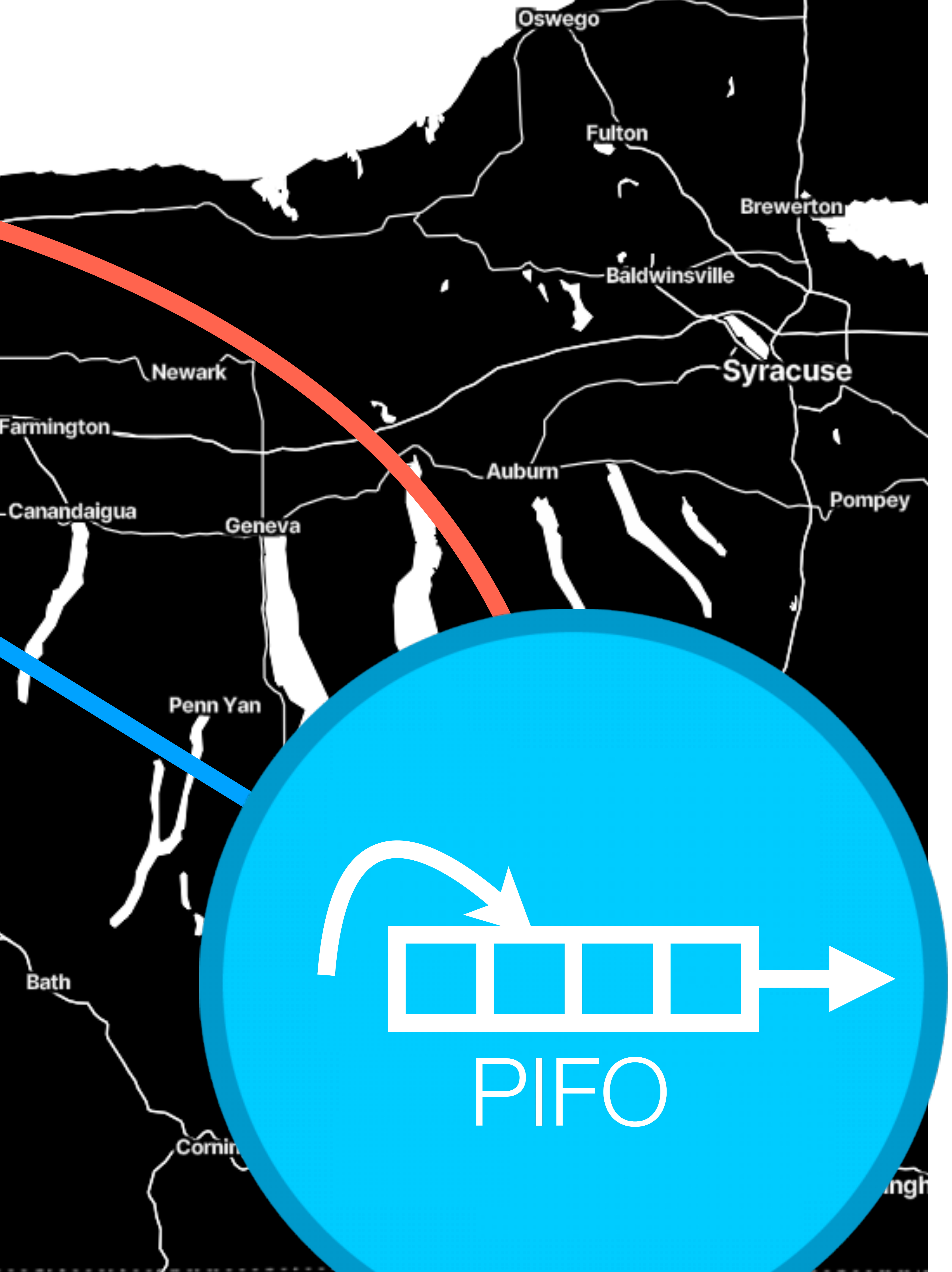
Basic tools work fine…



Software-defined networking made networks programmable.

We want control over *packet scheduling.*

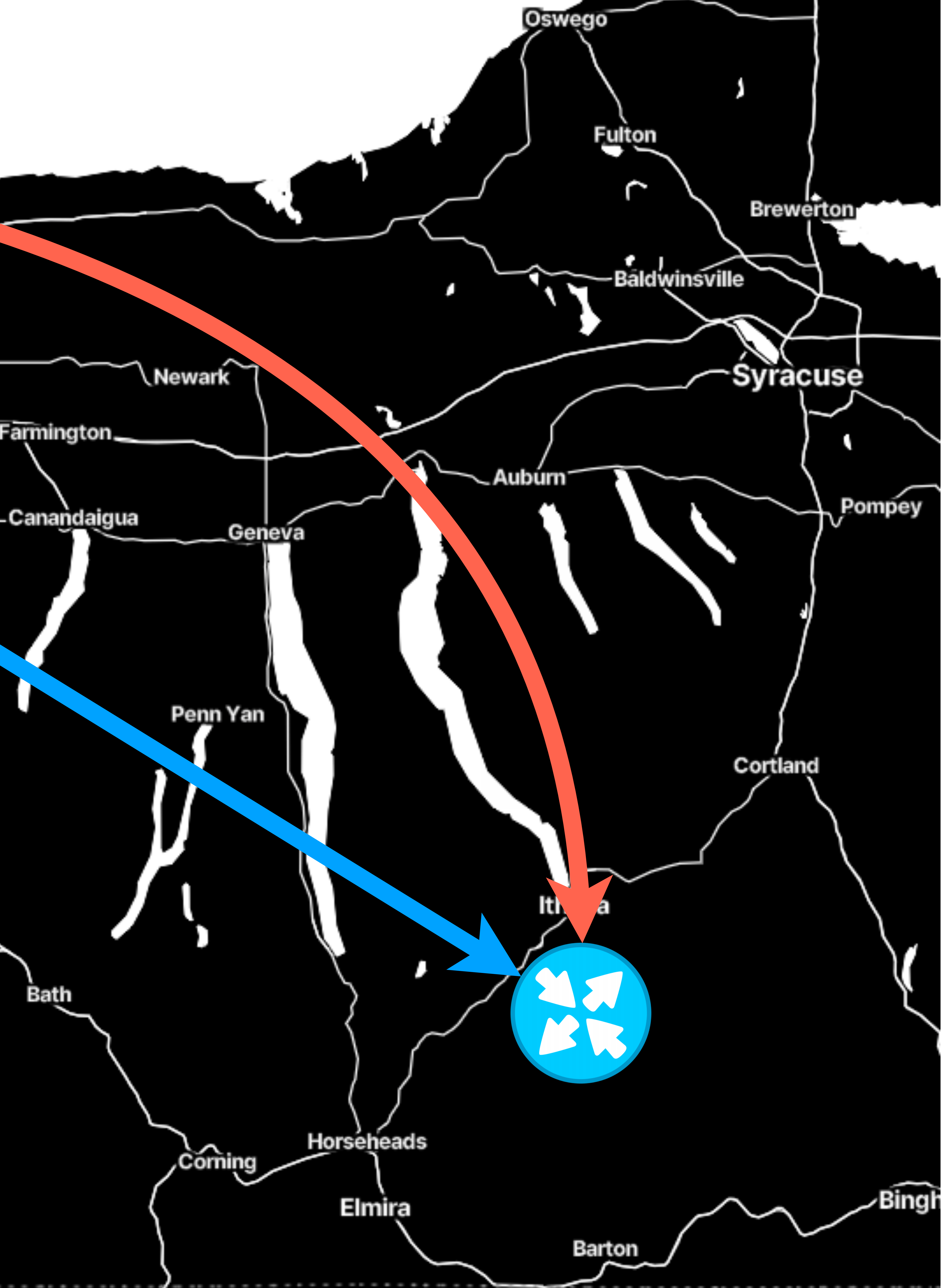
Basic tools work fine…



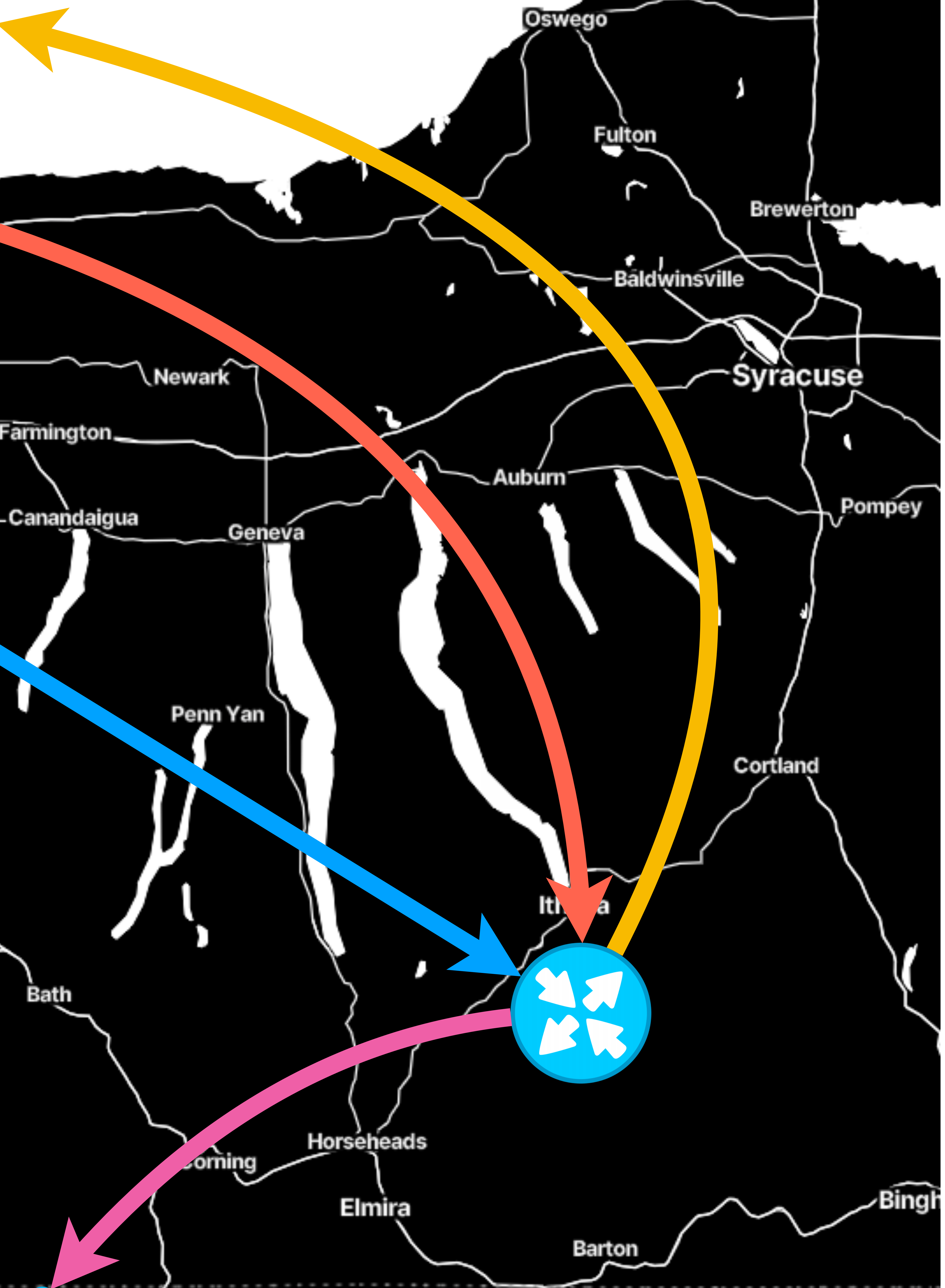
Software-defined networking made networks programmable.

We want control over *packet scheduling.*

Basic tools work fine…

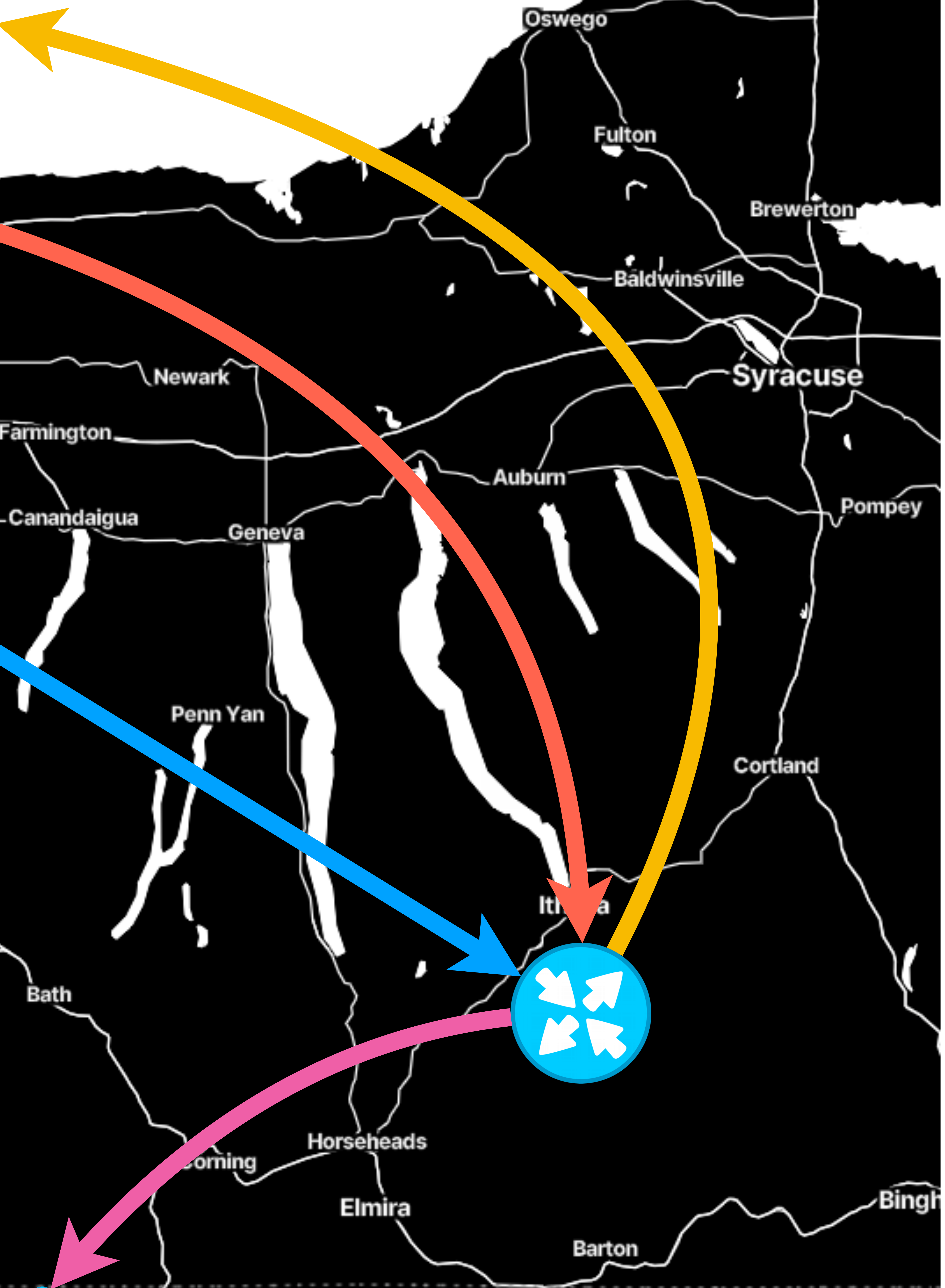


…but modern scheduling requires more.



…but modern scheduling requires more.

R traffic goes to either Pittsburgh or Toronto.

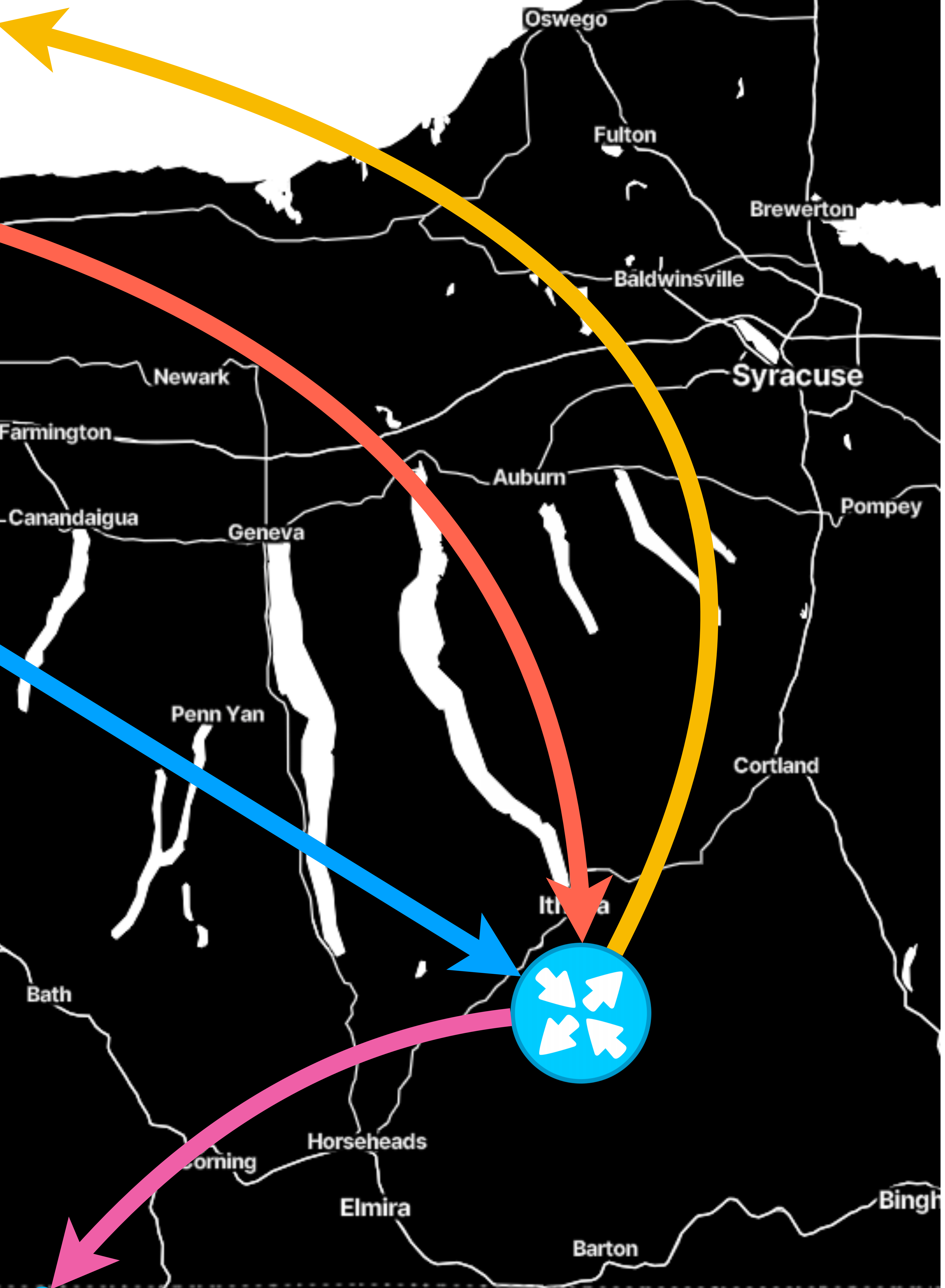


…but modern scheduling requires more.

R traffic goes to either Pittsburgh or Toronto.

Goal:

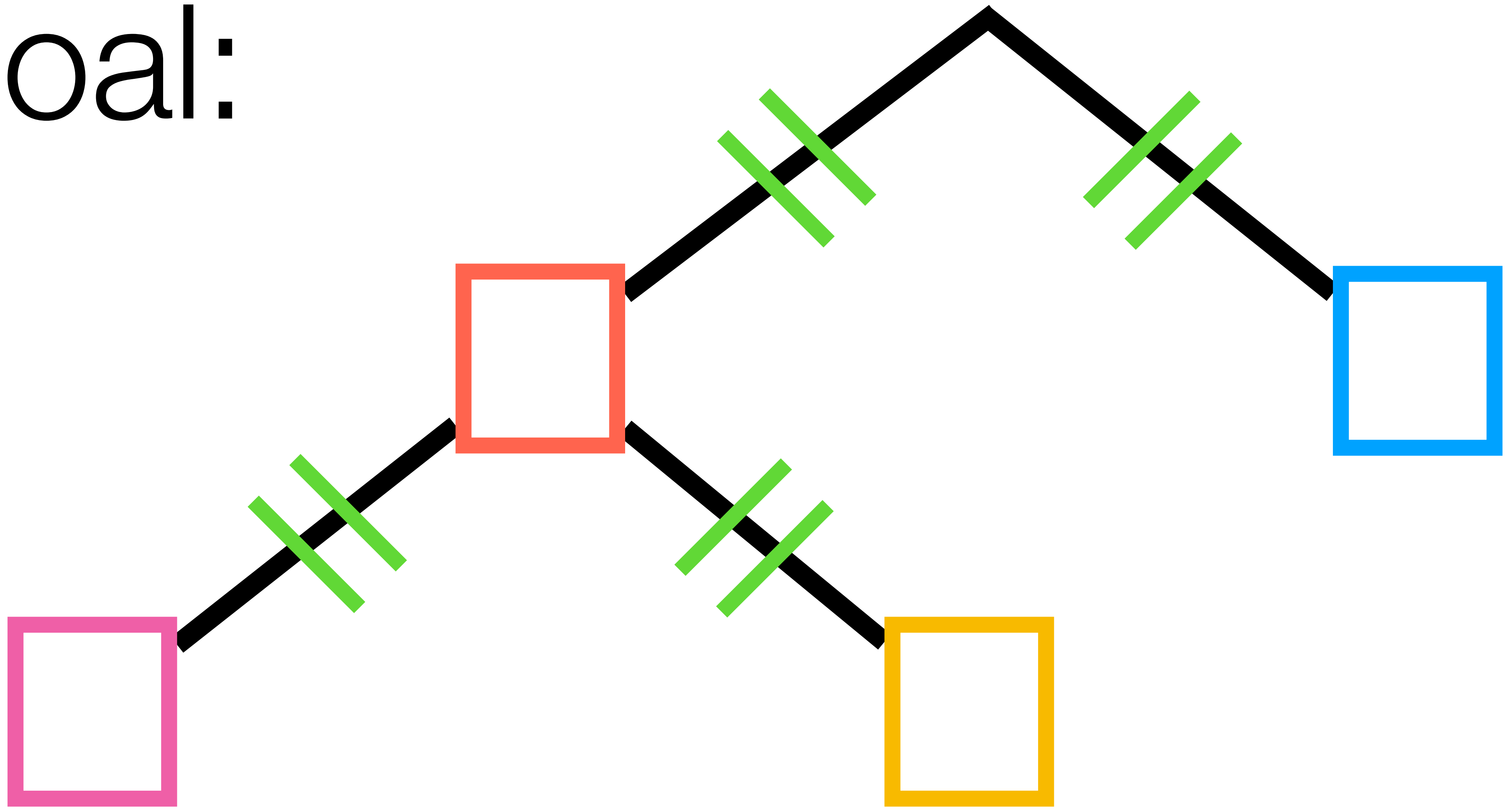
Interleave R and B;
interleave P and T.



…but modern scheduling requires more.

R traffic goes to either Pittsburgh or Toronto.

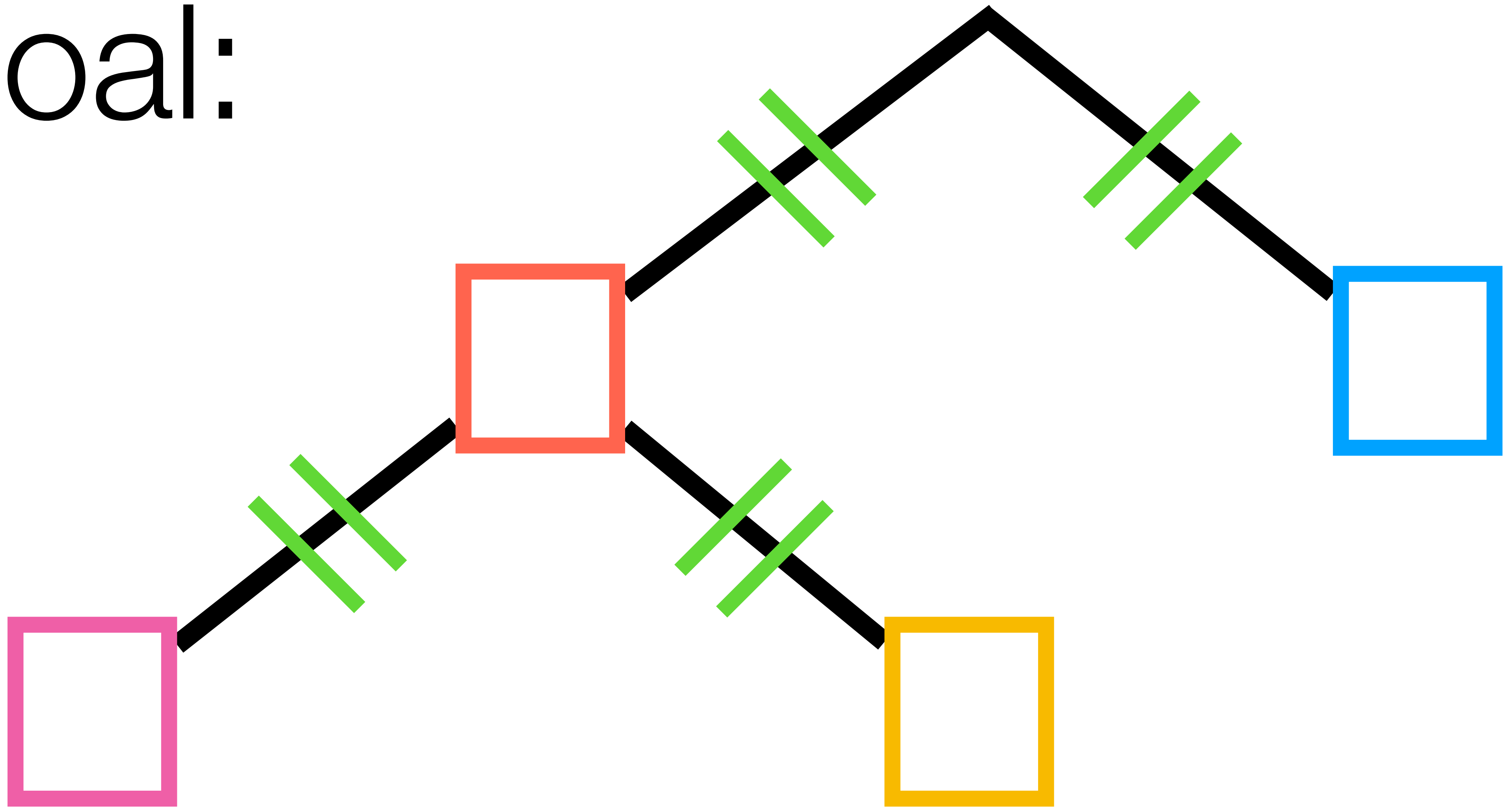
Goal:



…but modern scheduling requires more.

R traffic goes to either Pittsburgh or Toronto.

Goal:



Oswego
Fulton
Brewerton
Baldwinsville
Syracuse
Newark
Farmington
Auburn
Canandaigua
Geneva
Pompey
Penn Yan
Bath
PIFO Tree

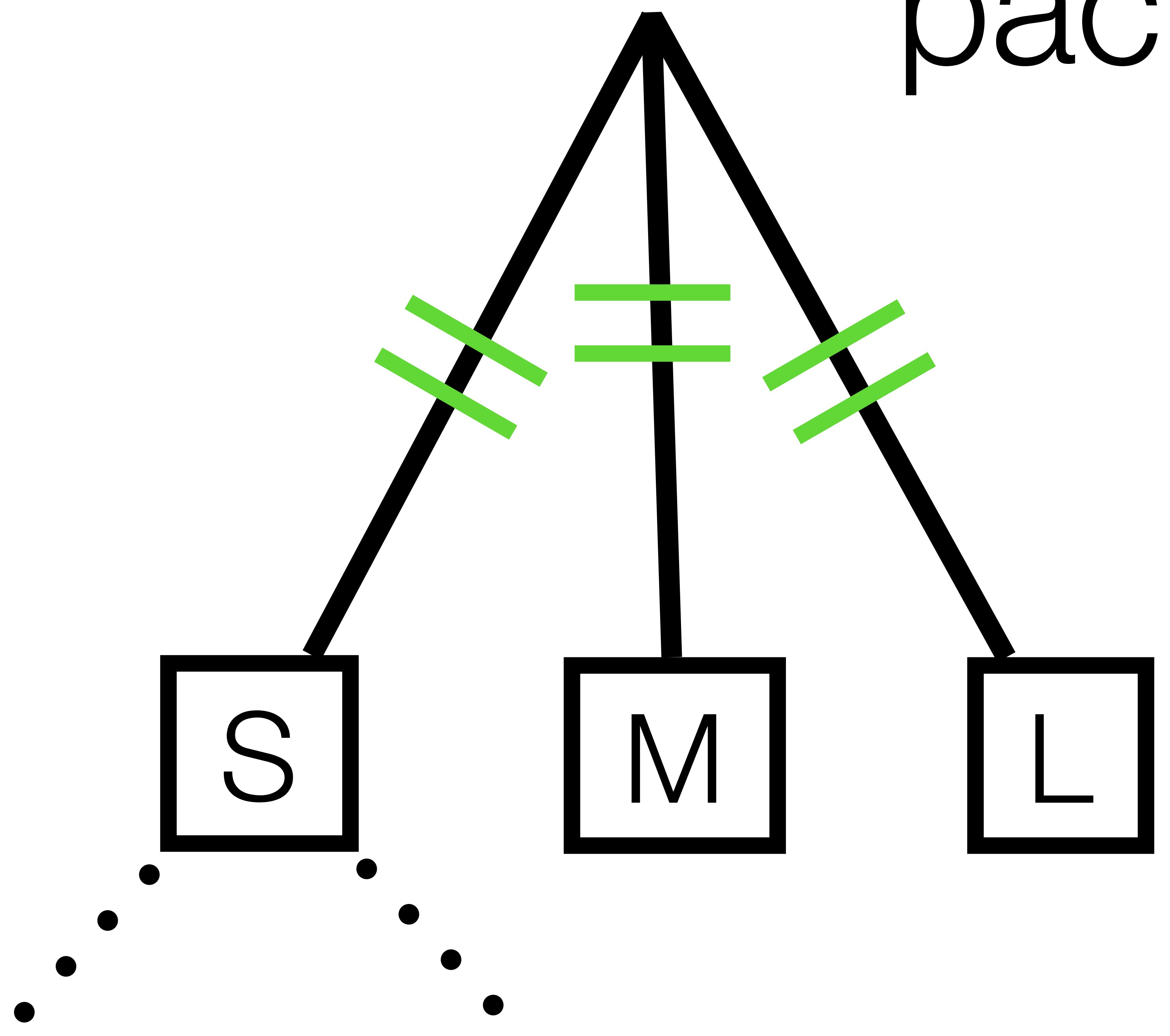
New plan!

New plan!

Interleave
small, medium, and large
packets.

New plan!

Interleave
small, medium, and large
packets.



No general way to deploy our gadget.

No general way to deploy our gadget.

A human needs a *range* of trees.

No general way to deploy our gadget.

A human needs a *range* of trees.

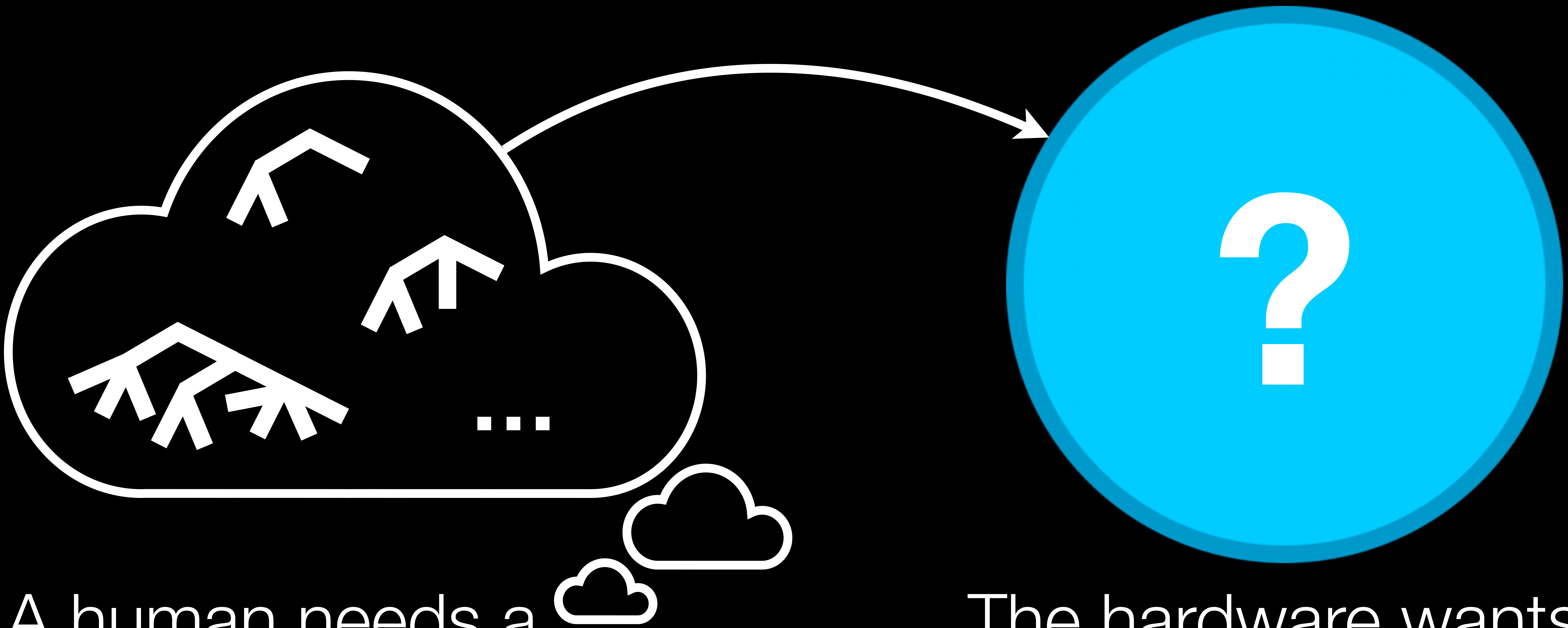
The hardware wants to support *one* tree.

No general way to deploy our gadget.

A human needs a *range* of trees.

The hardware wants to support *one* tree.

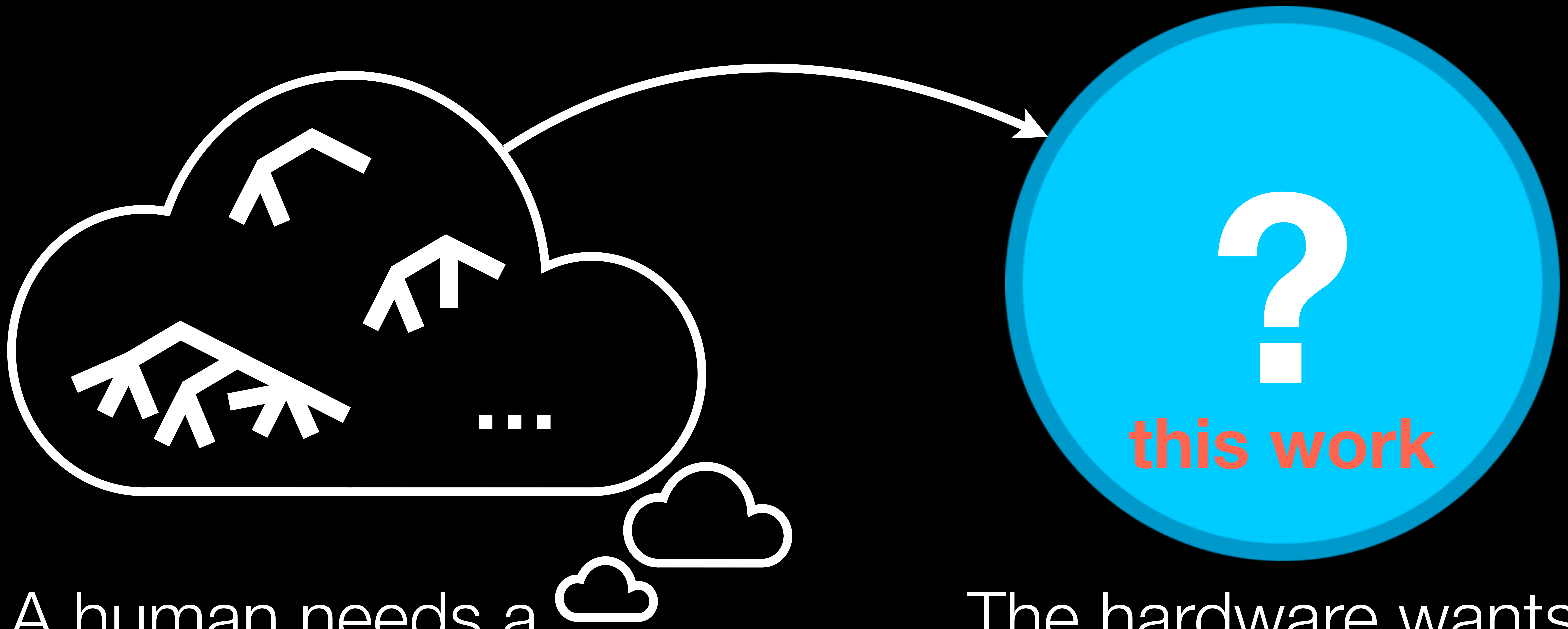
No general way to deploy our gadget.



A human needs a *range* of trees.

The hardware wants to support *one* tree.

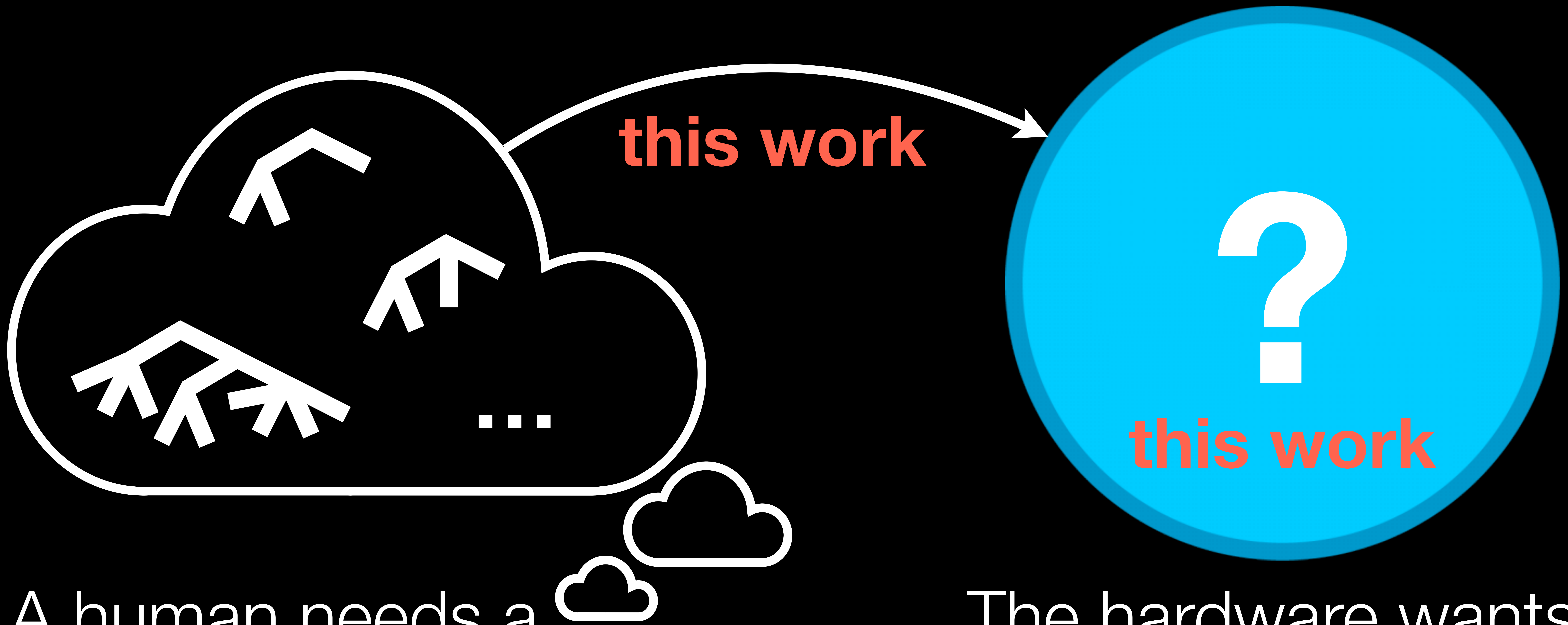
No general way to deploy our gadget.



A human needs a *range* of trees.

The hardware wants to support *one* tree.

No general way to deploy our gadget.



A human needs a *range* of trees.

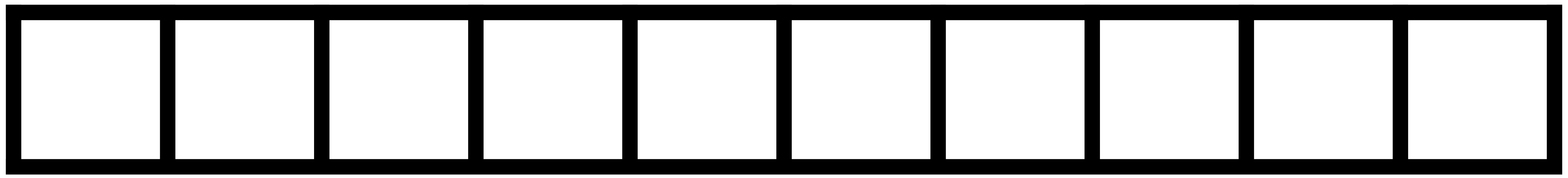
The hardware wants to support *one* tree.

Aside: PIFO Trees

Sivaraman et al. at SIGCOMM ’16

Review: FIFO

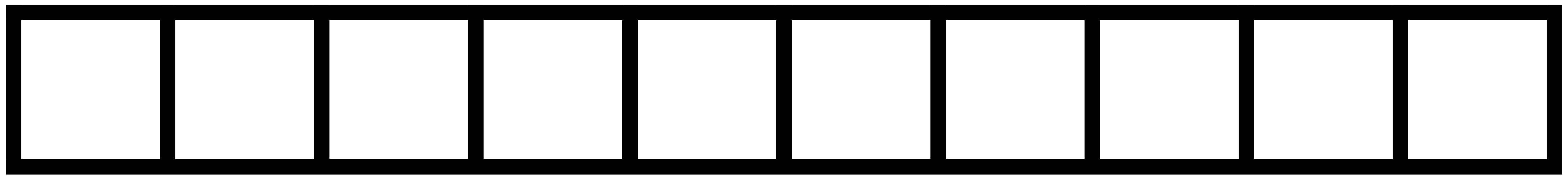
Just an ordered collection.



Review: FIFO

Just an ordered collection.

Two ways of interacting with the collection:

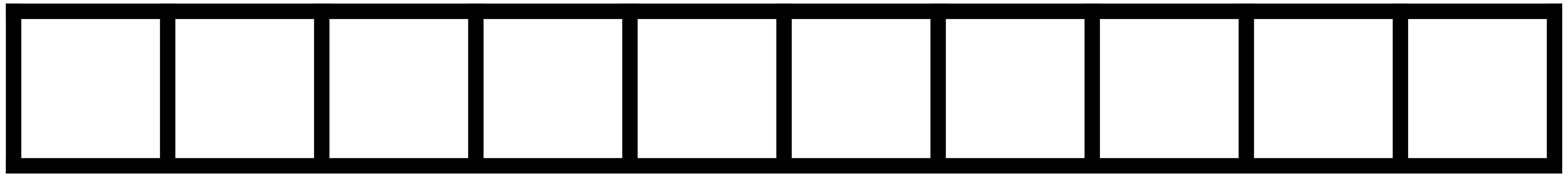


Review: FIFO

Just an ordered collection.

Two ways of interacting with the collection:

push

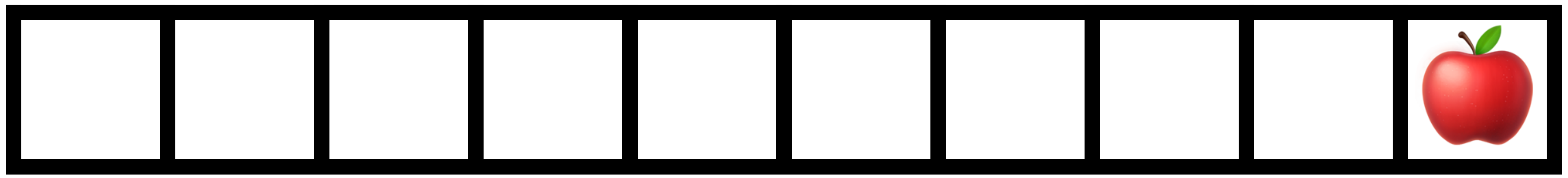


Review: FIFO

Just an ordered collection.

Two ways of interacting with the collection:

push

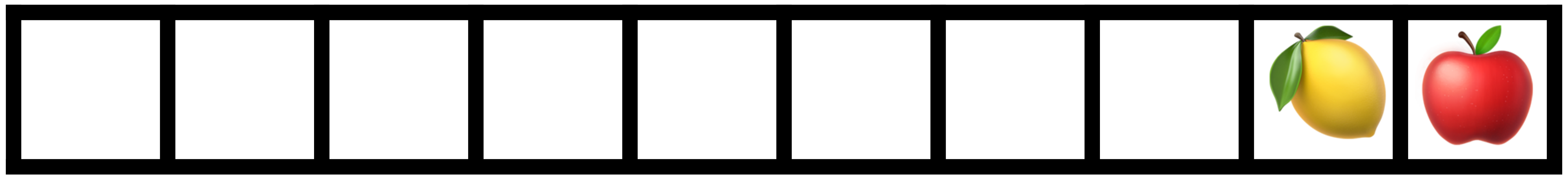


Review: FIFO

Just an ordered collection.

Two ways of interacting with the collection:

push



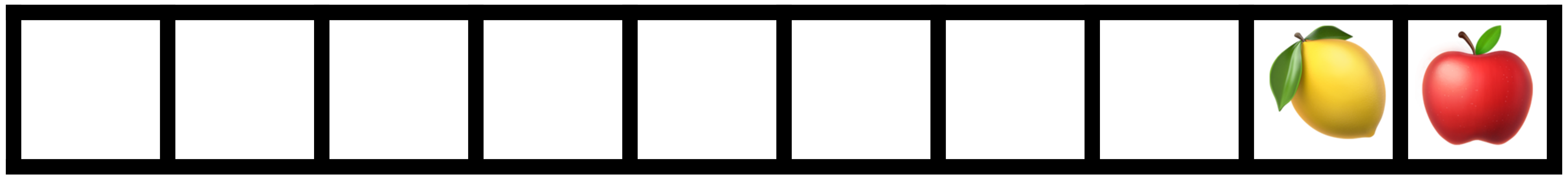
Review: FIFO

Just an ordered collection.

Two ways of interacting with the collection:

push

pop

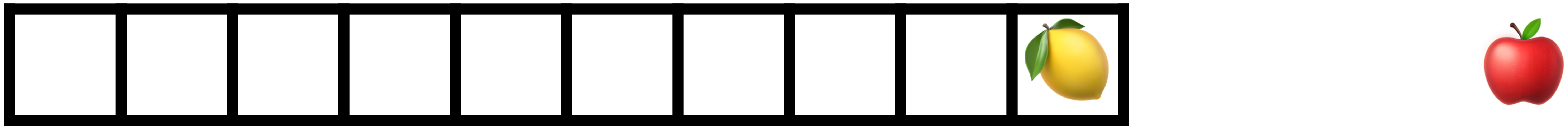


Review: FIFO

Just an ordered collection.

Two ways of interacting with the collection:

push pop

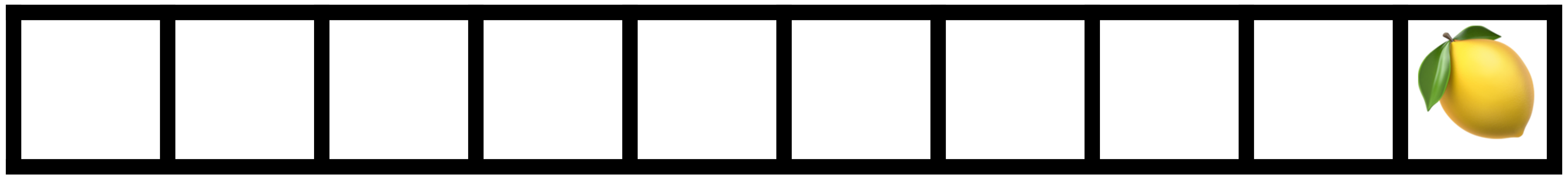


Review: FIFO

Just an ordered collection.

Two ways of interacting with the collection:

push pop



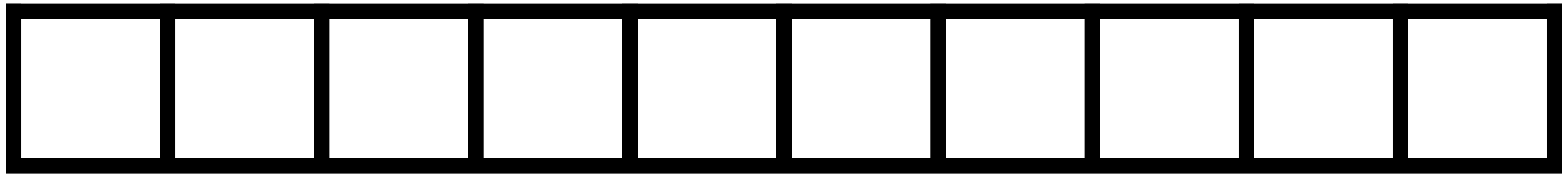
Review: FIFO

Just an ordered collection.

Two ways of interacting with the collection:

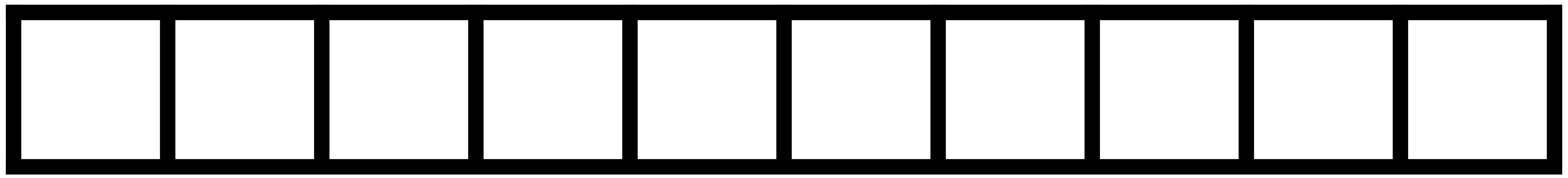
push

pop



Review: priority queue

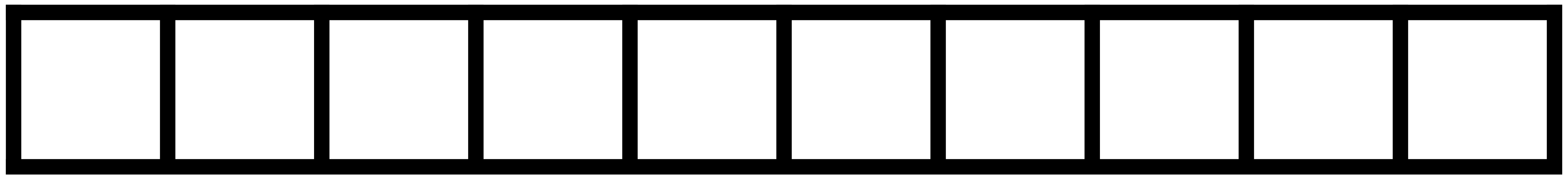
Everything from before holds,
but we have a little more control.



Review: priority queue

Everything from before holds,
but we have a little more control.

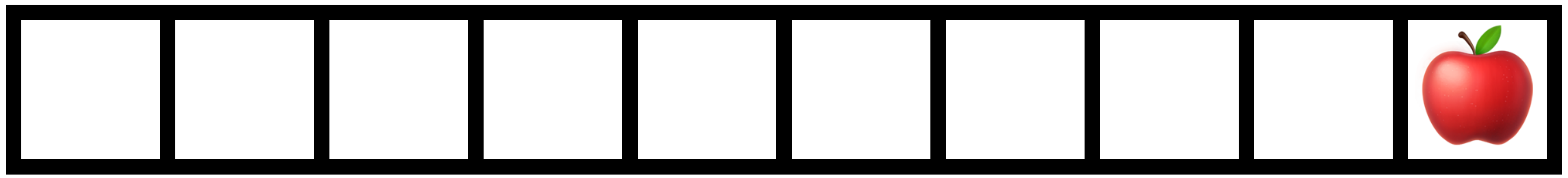
Say we have a queue *prioritized by pH.*



Review: priority queue

Everything from before holds,
but we have a little more control.

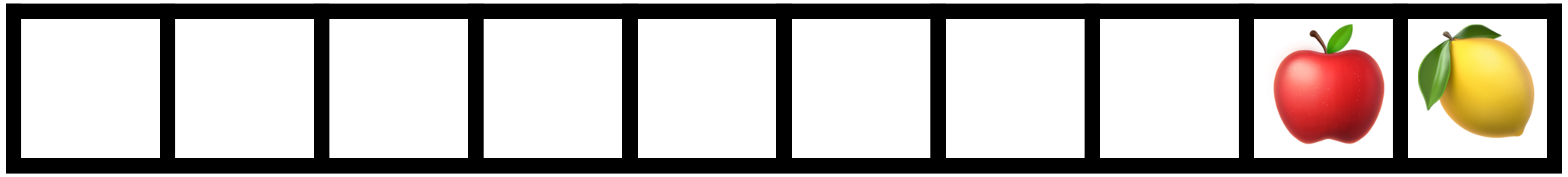
Say we have a queue *prioritized by pH.*



Review: priority queue

Everything from before holds,
but we have a little more control.

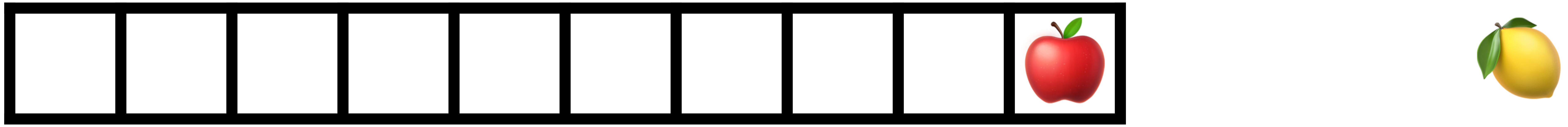
Say we have a queue *prioritized by pH.*



Review: priority queue

Everything from before holds,
but we have a little more control.

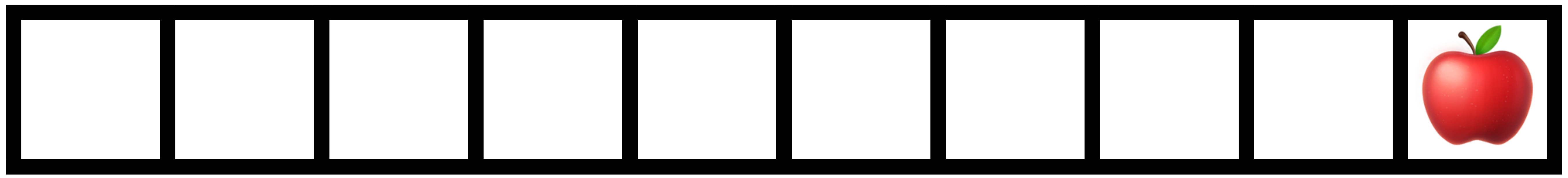
Say we have a queue *prioritized by pH.*



Review: priority queue

Everything from before holds,
but we have a little more control.

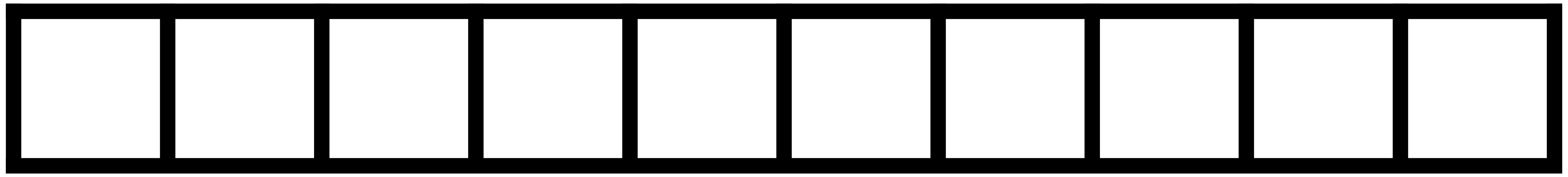
Say we have a queue *prioritized by pH.*



Review: priority queue

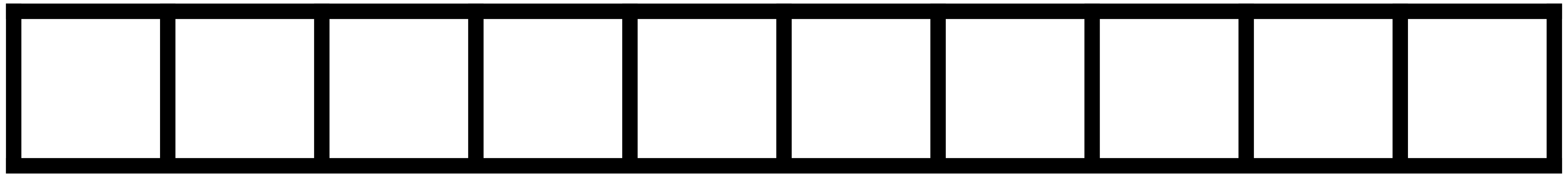
Everything from before holds,
but we have a little more control.

Say we have a queue *prioritized by pH.*



Review: priority queue

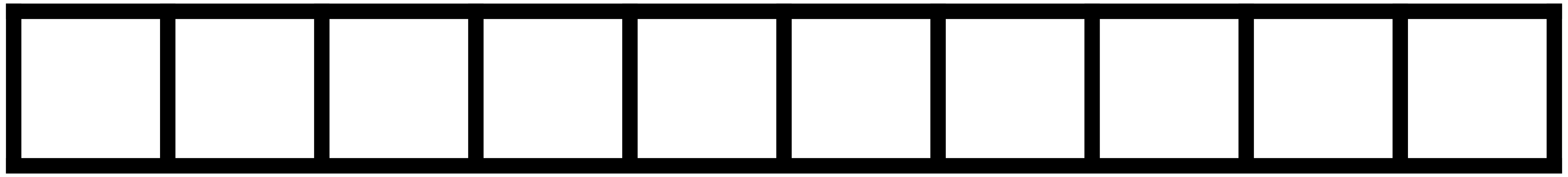
The priority need not be inherent to the item!



Review: priority queue

The priority need not be inherent to the item!

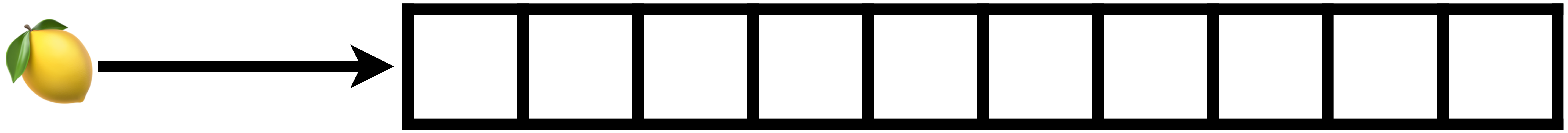
We can have a *ranking function:*



Review: priority queue

The priority need not be inherent to the item!

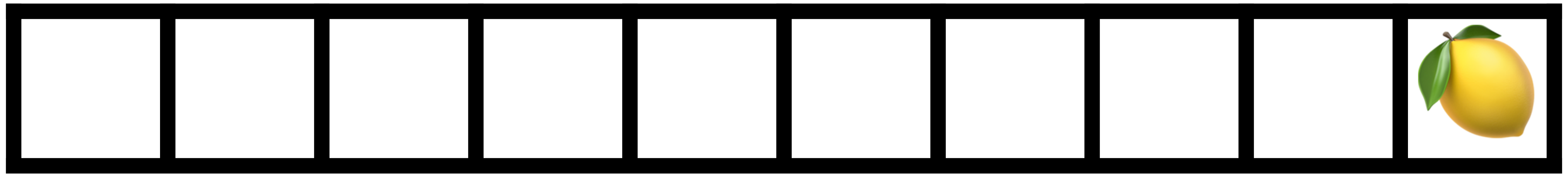
We can have a *ranking function:*



Review: priority queue

The priority need not be inherent to the item!

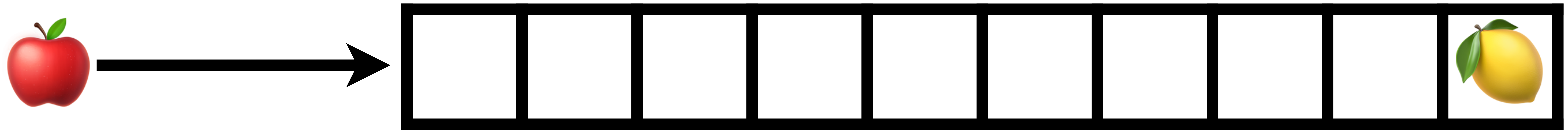
We can have a *ranking function:*



Review: priority queue

The priority need not be inherent to the item!

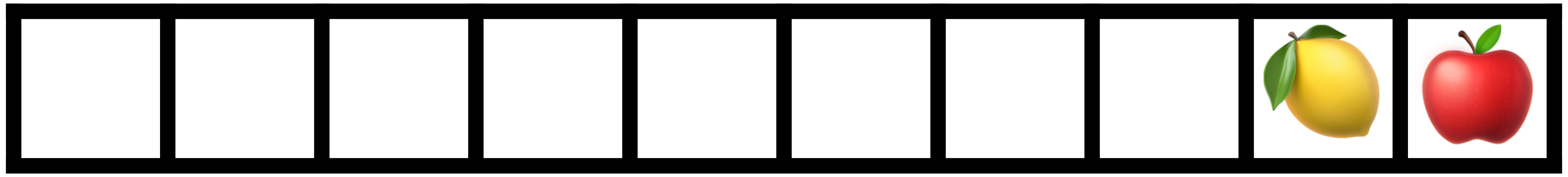
We can have a *ranking function:*



Review: priority queue

The priority need not be inherent to the item!

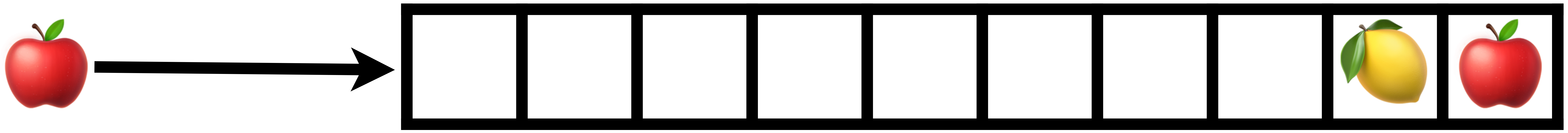
We can have a *ranking function:*



Review: priority queue

The priority need not be inherent to the item!

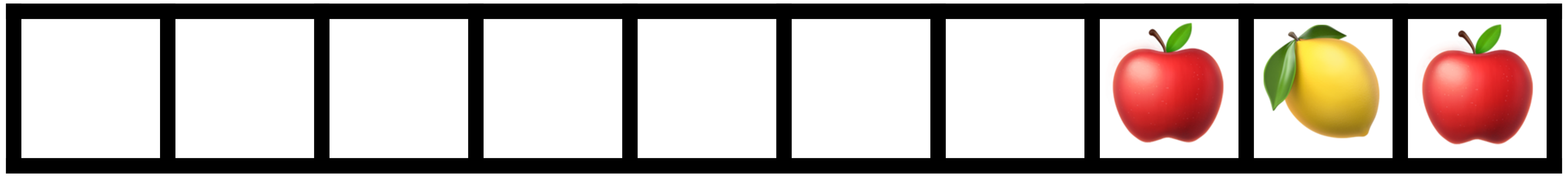
We can have a *ranking function:*



Review: priority queue

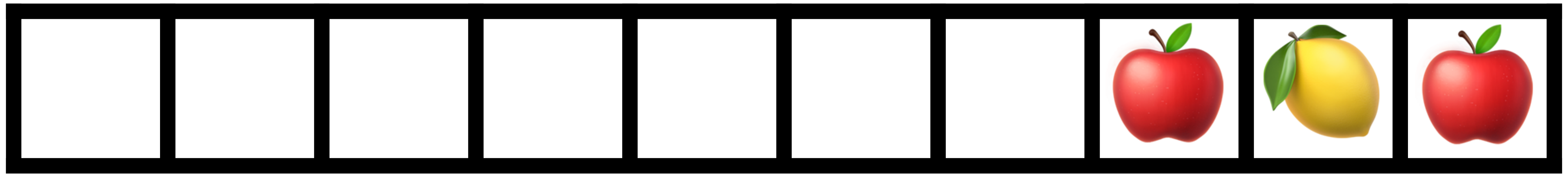
The priority need not be inherent to the item!

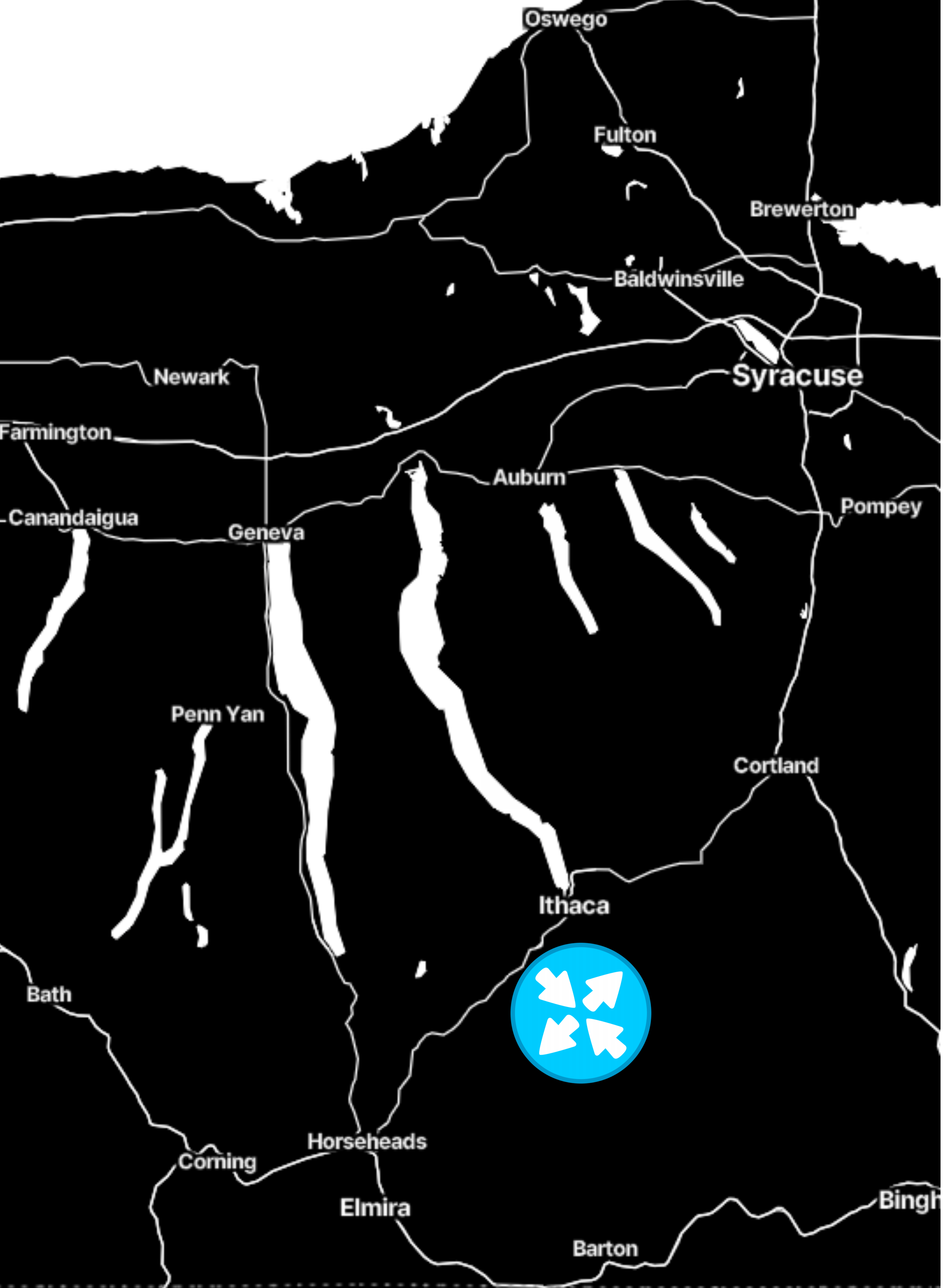
We can have a *ranking function:*

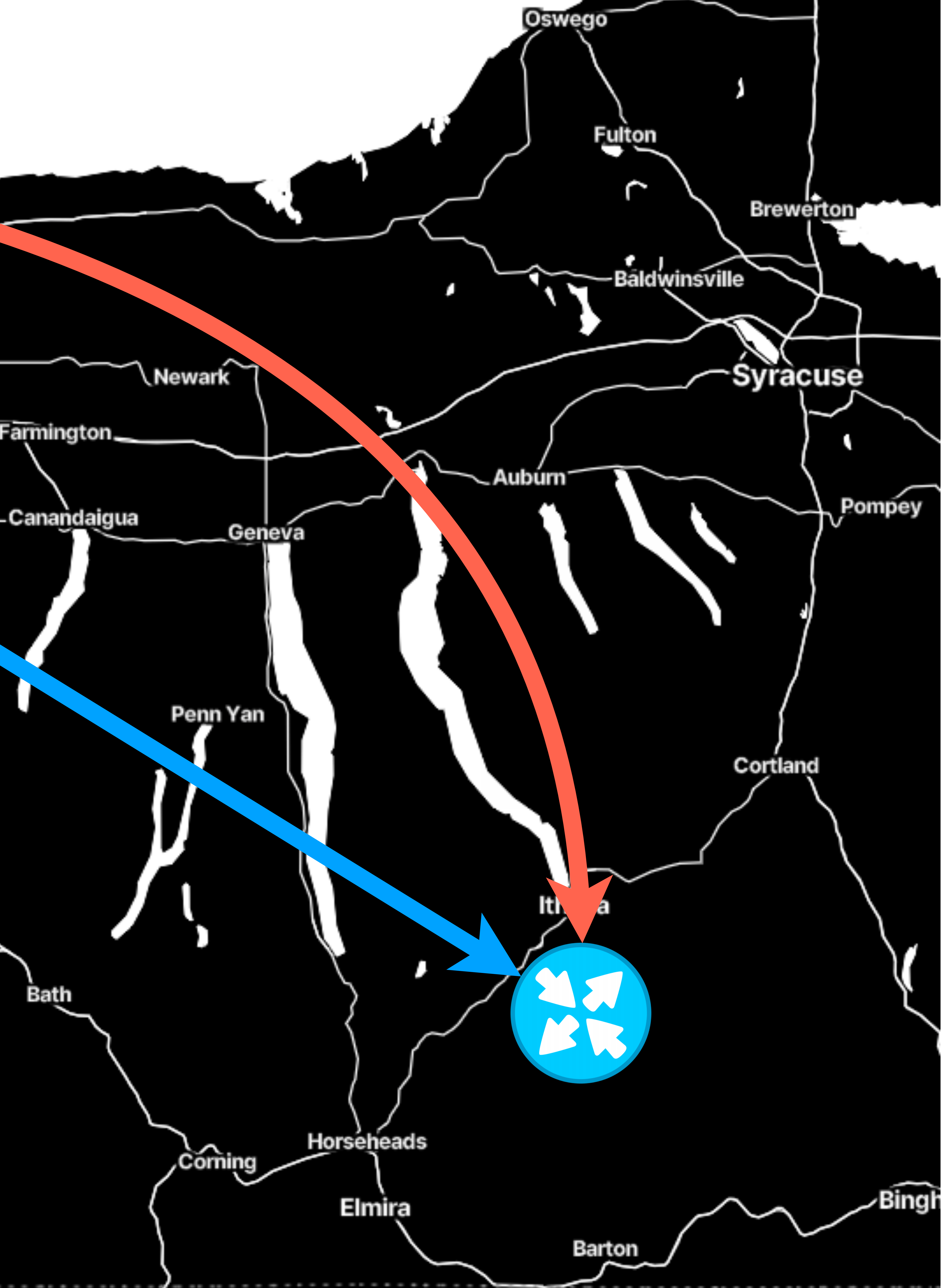


Introducing: PIFO

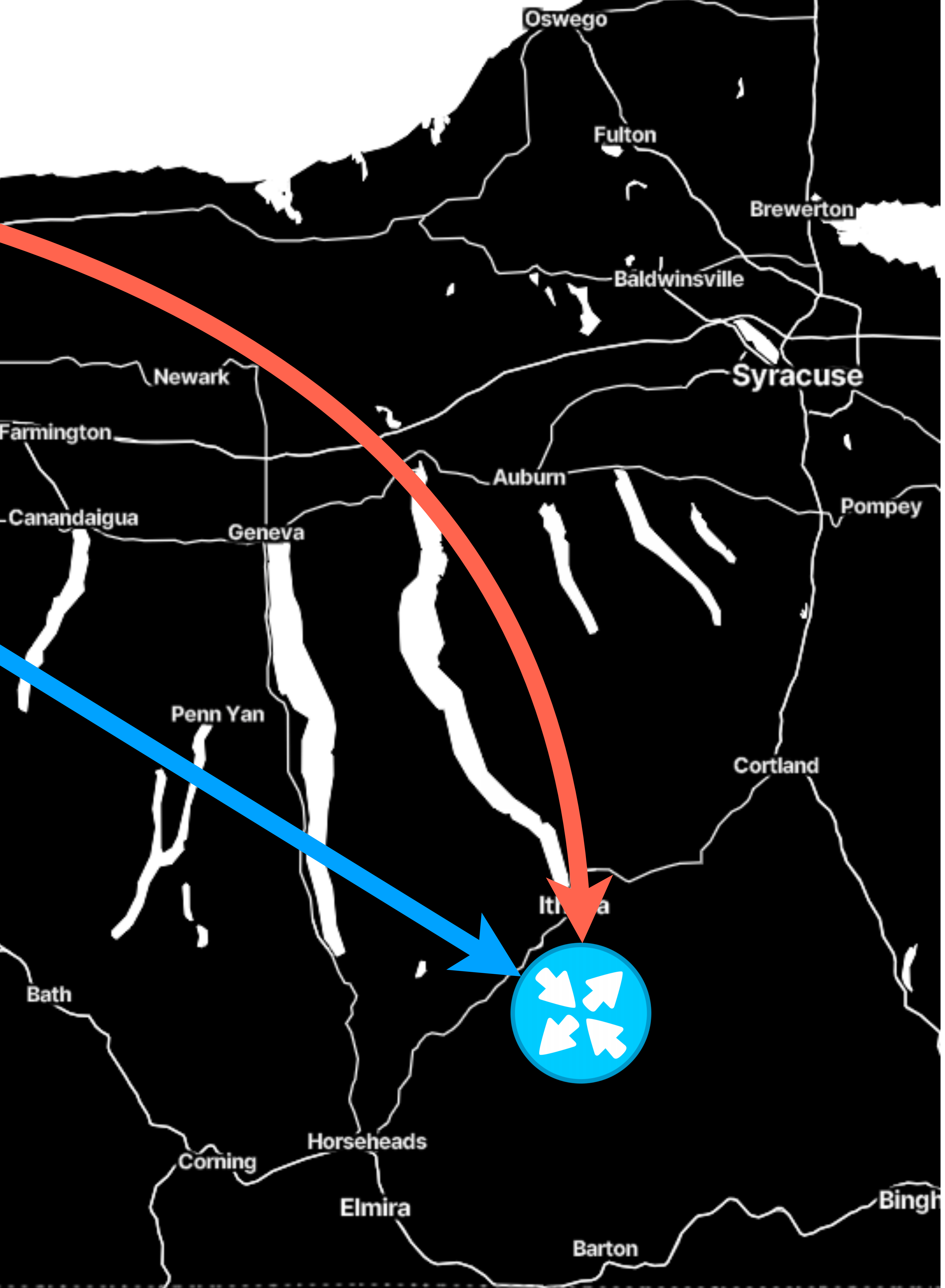
Just a PQ, with a ranking function,
but with *rank-ties* broken in FIFO order.




Oswego
Fulton
Brewerton
Baldwinsville
Newark
Syracuse
Farmington
Auburn
Pompey
Canandaigua
Geneva
Penn Yan
Cortland
Ithaca
Bath
Horseheads
Corning
Elmira
Barton

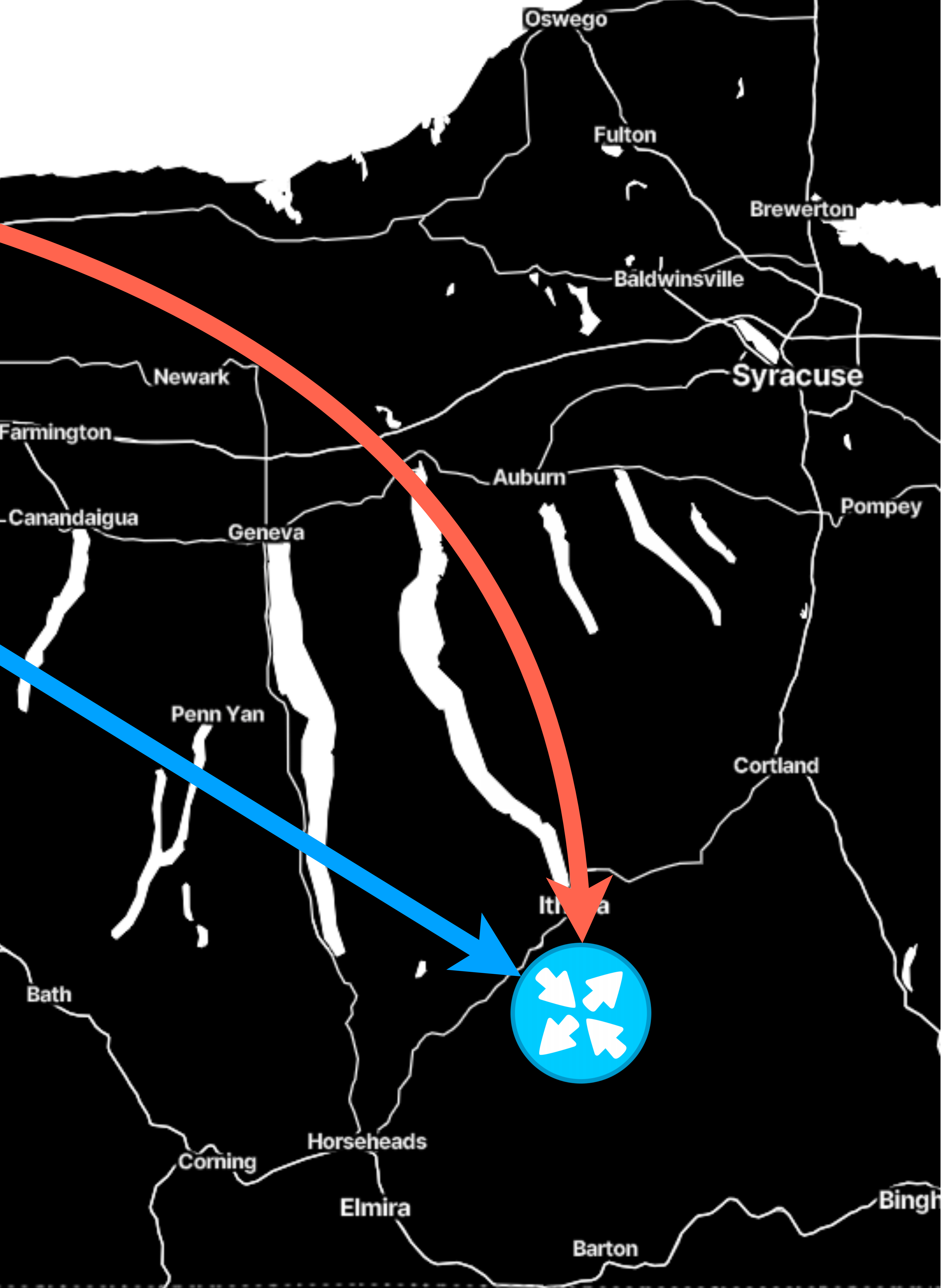


Baby goal:
interleave R and B.



Baby goal:
interleave R and B.

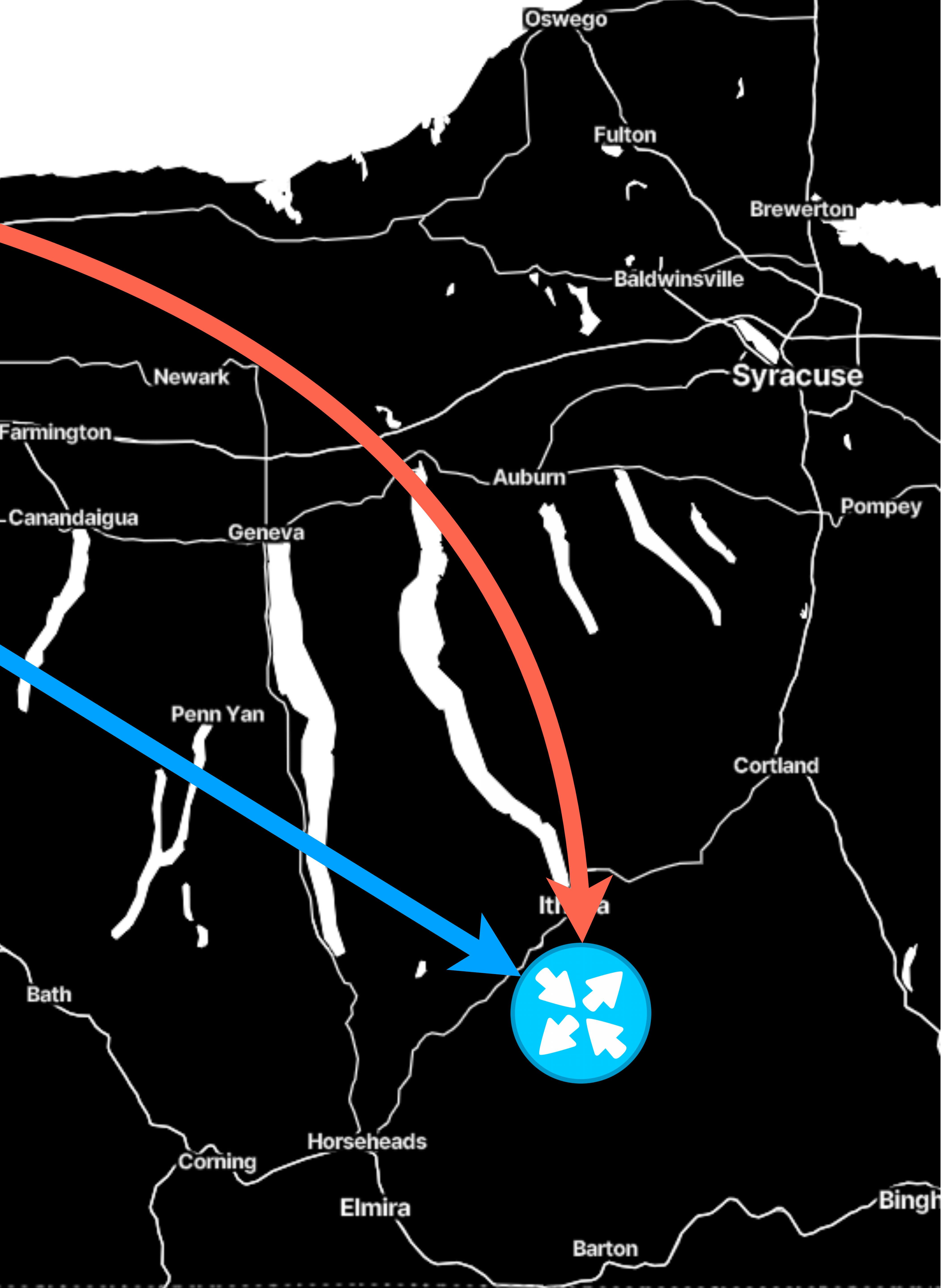
A PIFO will suffice.



Baby goal:
interleave R and B.

A PIFO will suffice.

B_n, …, B_1, (R,B)*

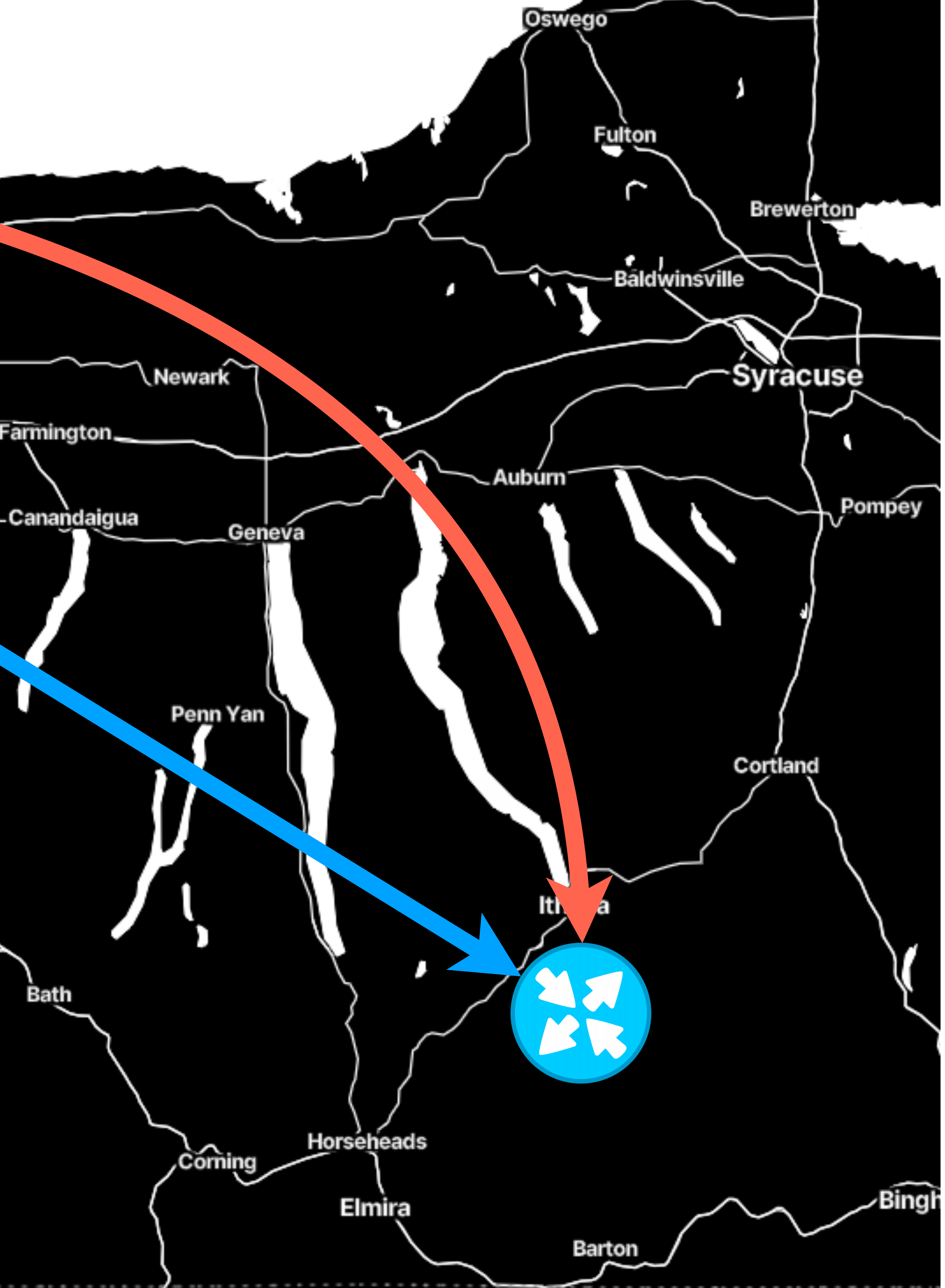


Baby goal:
interleave R and B.

A PIFO will suffice.

R

B_n, . . ., B_1, (R,B)*

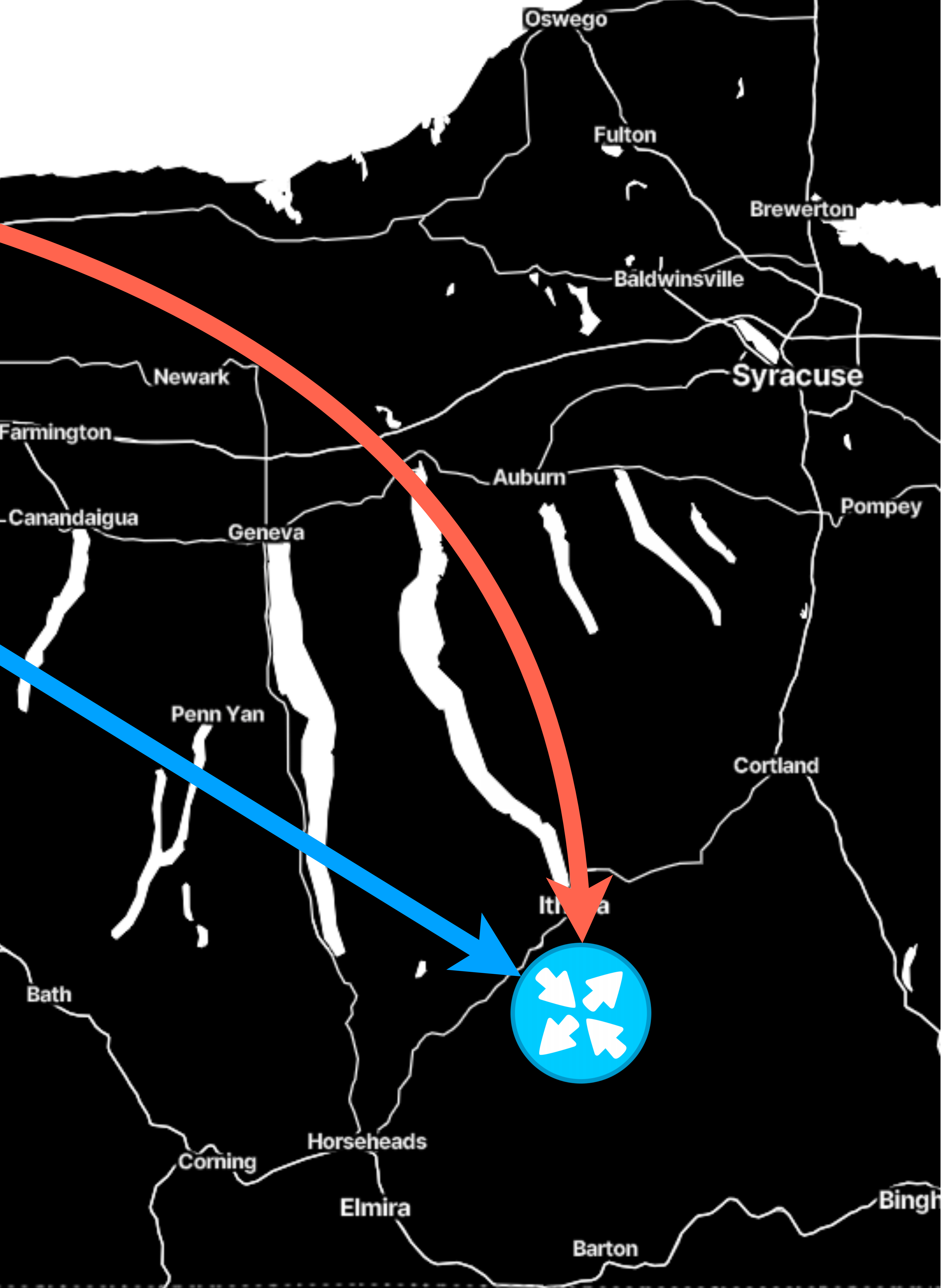


Baby goal:
interleave R and B.

A PIFO will suffice.

B

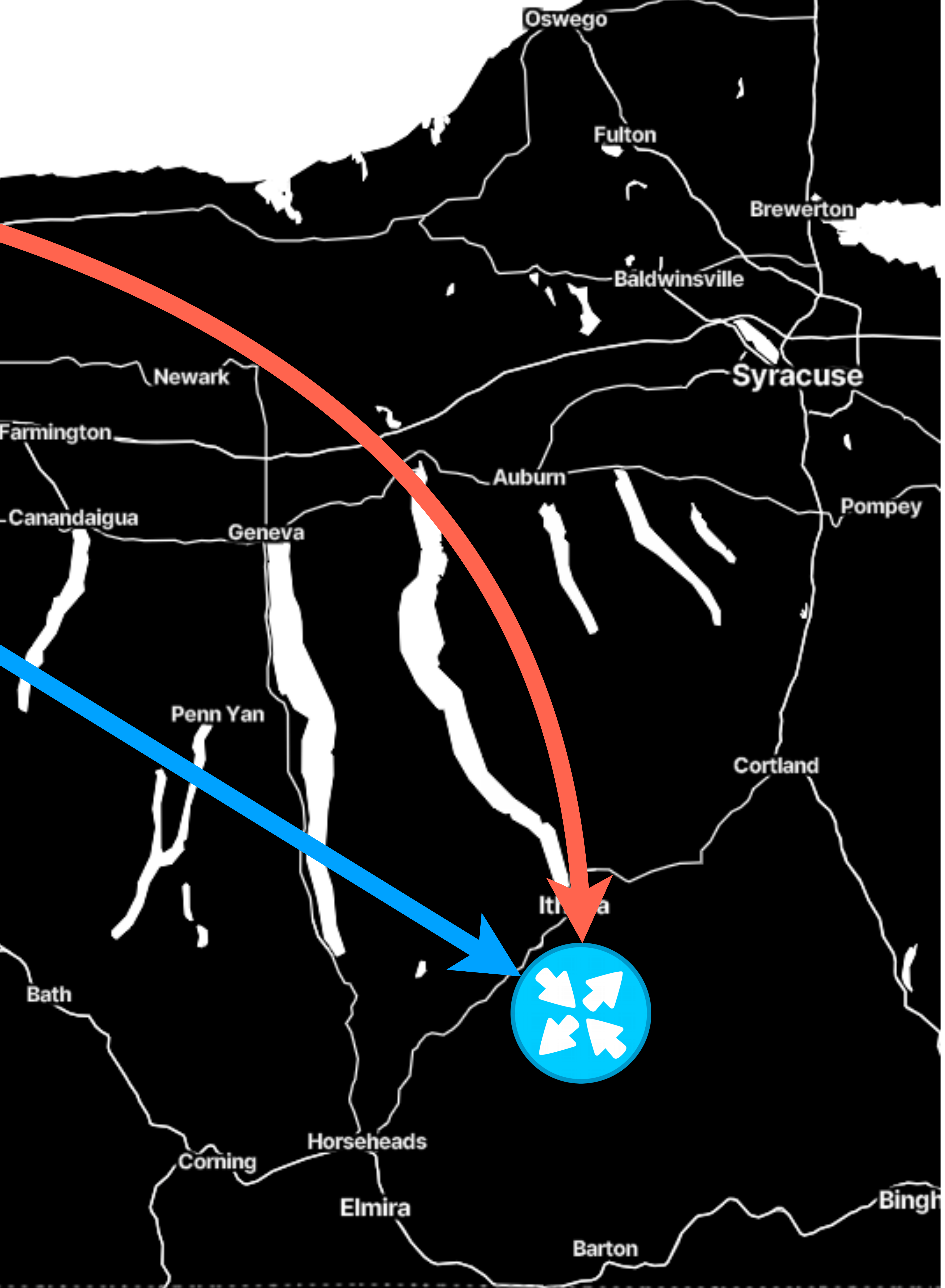
B_n, …, B_1, (R,B)*



Baby goal:
interleave R and B.

A PIFO will suffice.

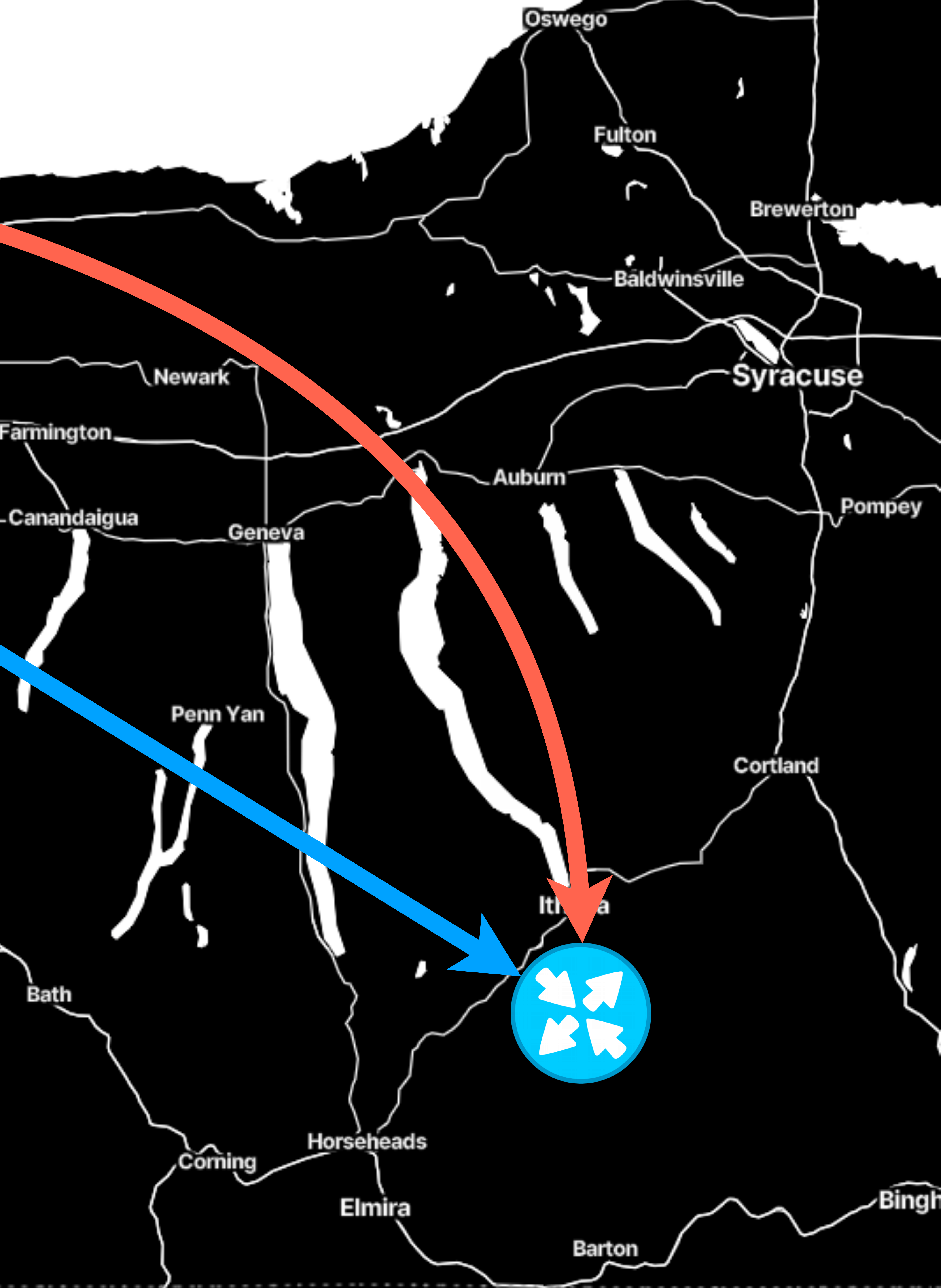
$B_n, \ldots, B_1, (R,B)^*$



Baby goal:
interleave R and B.

A PIFO will suffice.

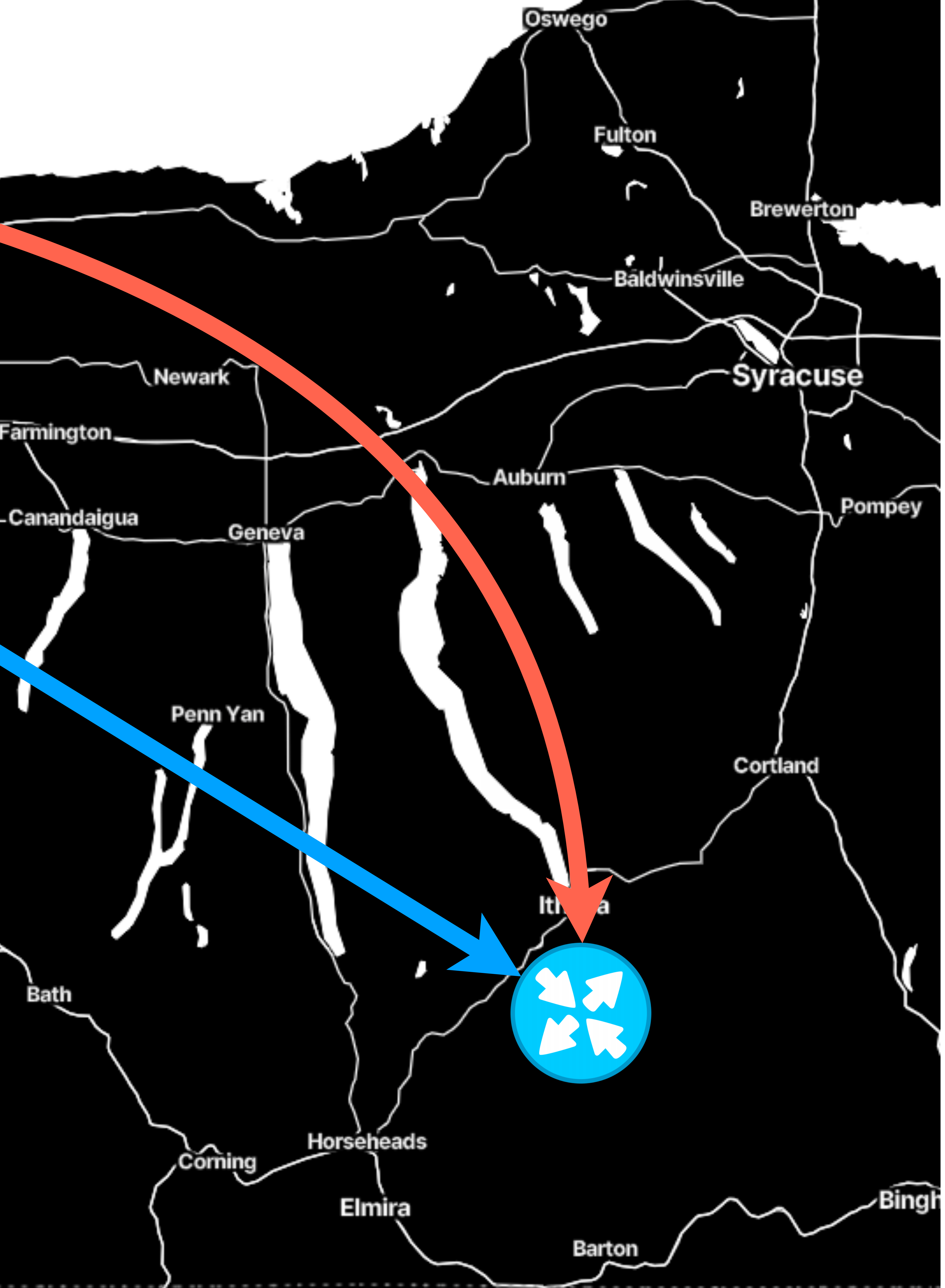
B_n, …, B_1, (R,B)*
R_n, …, R_1, (R,B)*

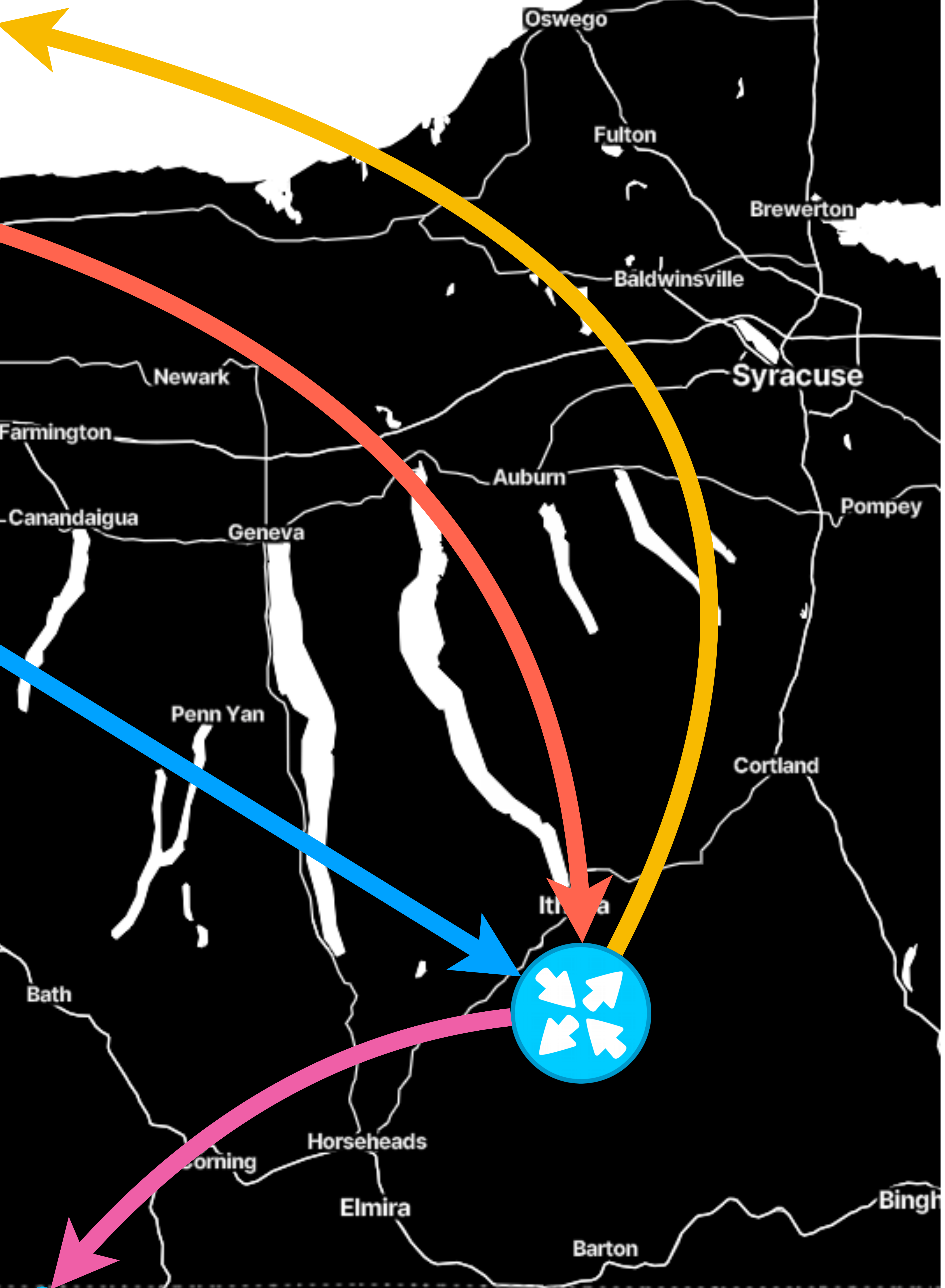


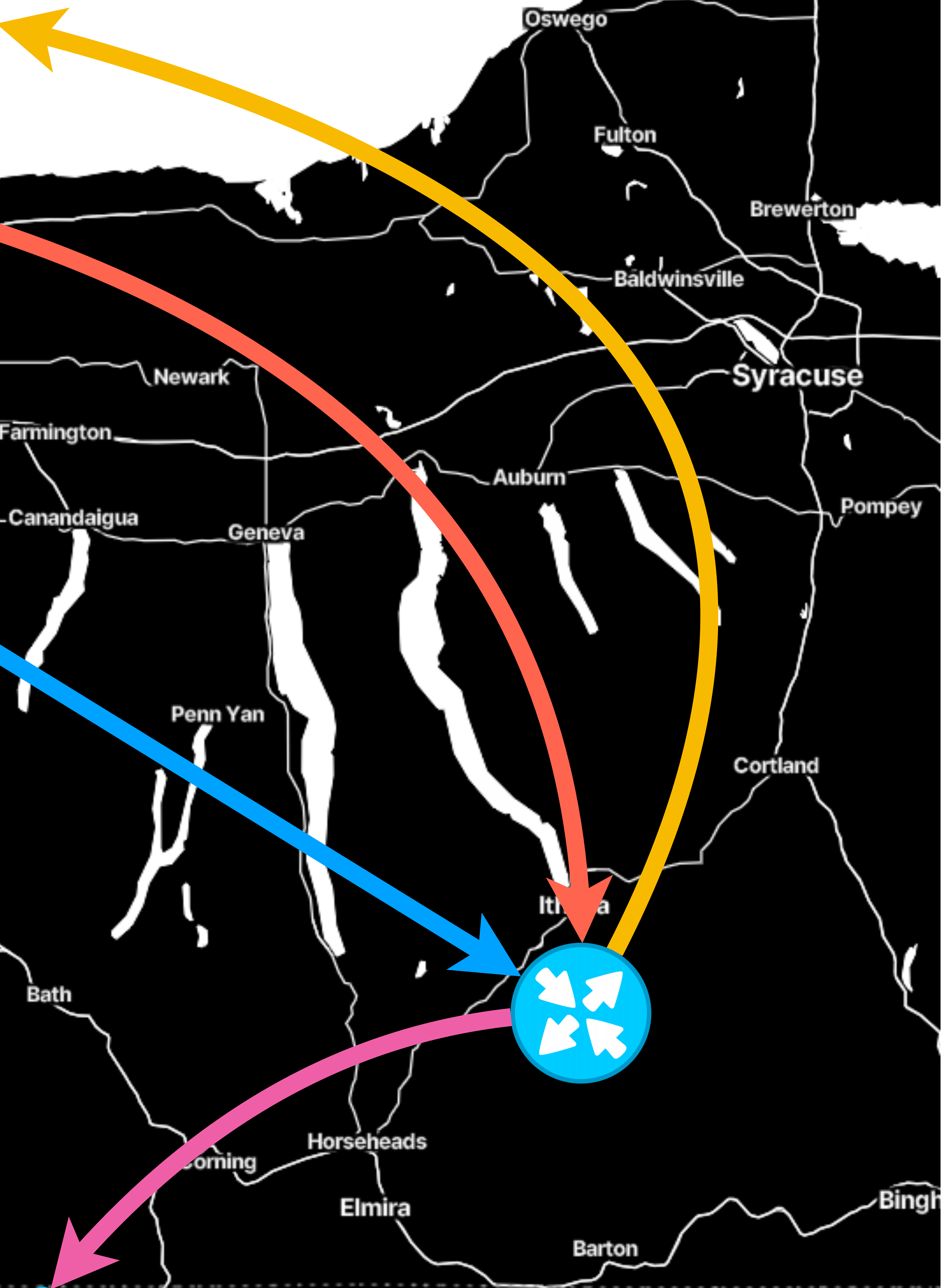
Baby goal:
interleave R and B.

A PIFO will suffice.

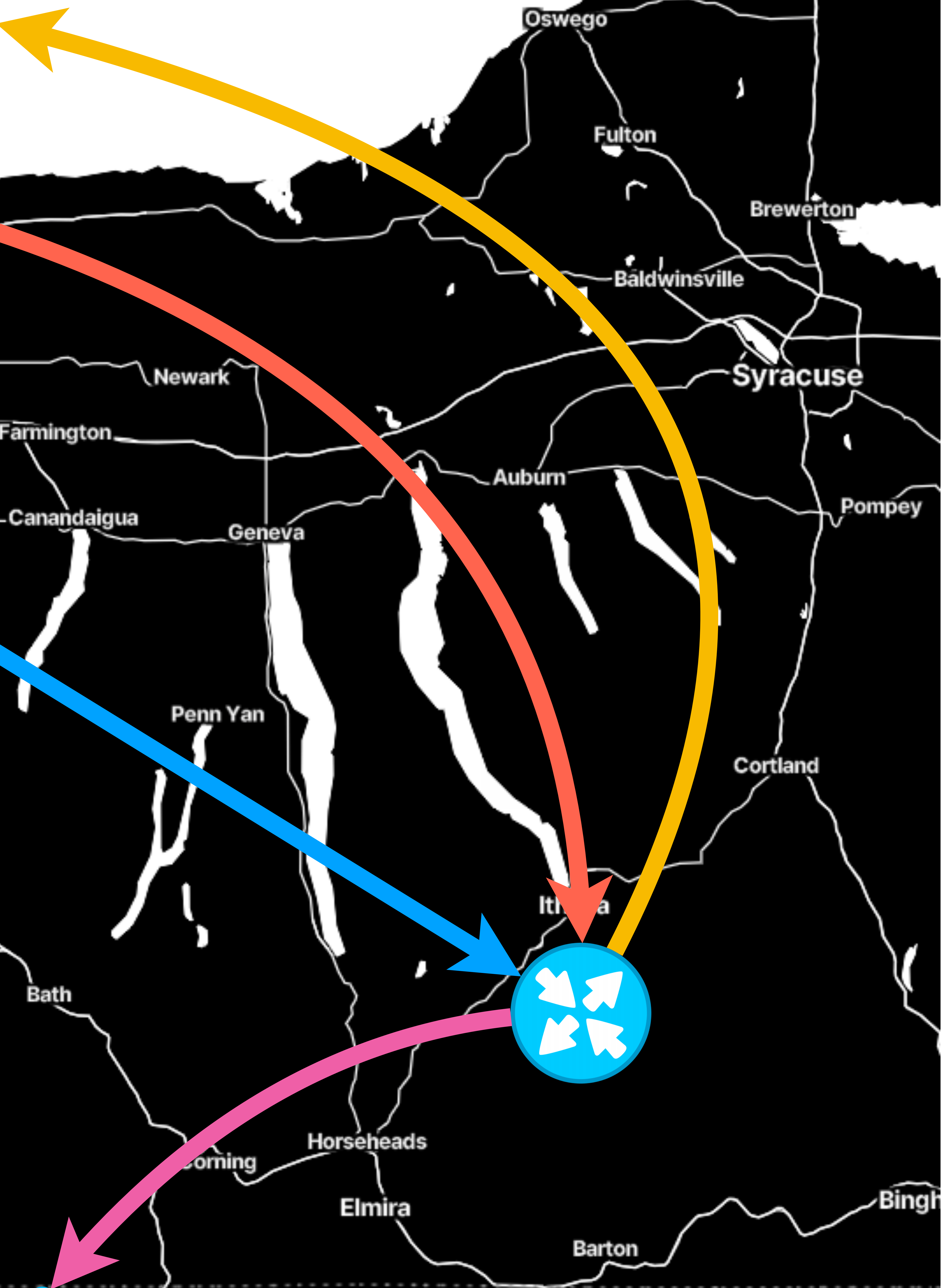
$B_n, \ldots, B_1, (R,B)^*$
$R_n, \ldots, R_1, (R,B)^*$
$(R,B)^*$


Oswego
Fulton
Brewerton
Baldwinsville
Newark
Syracuse
Farmington
Auburn
Pompey
Canandaigua
Geneva
Penn Yan
Cortland
Bath
Horseheads
Corning
Elmira
Barton

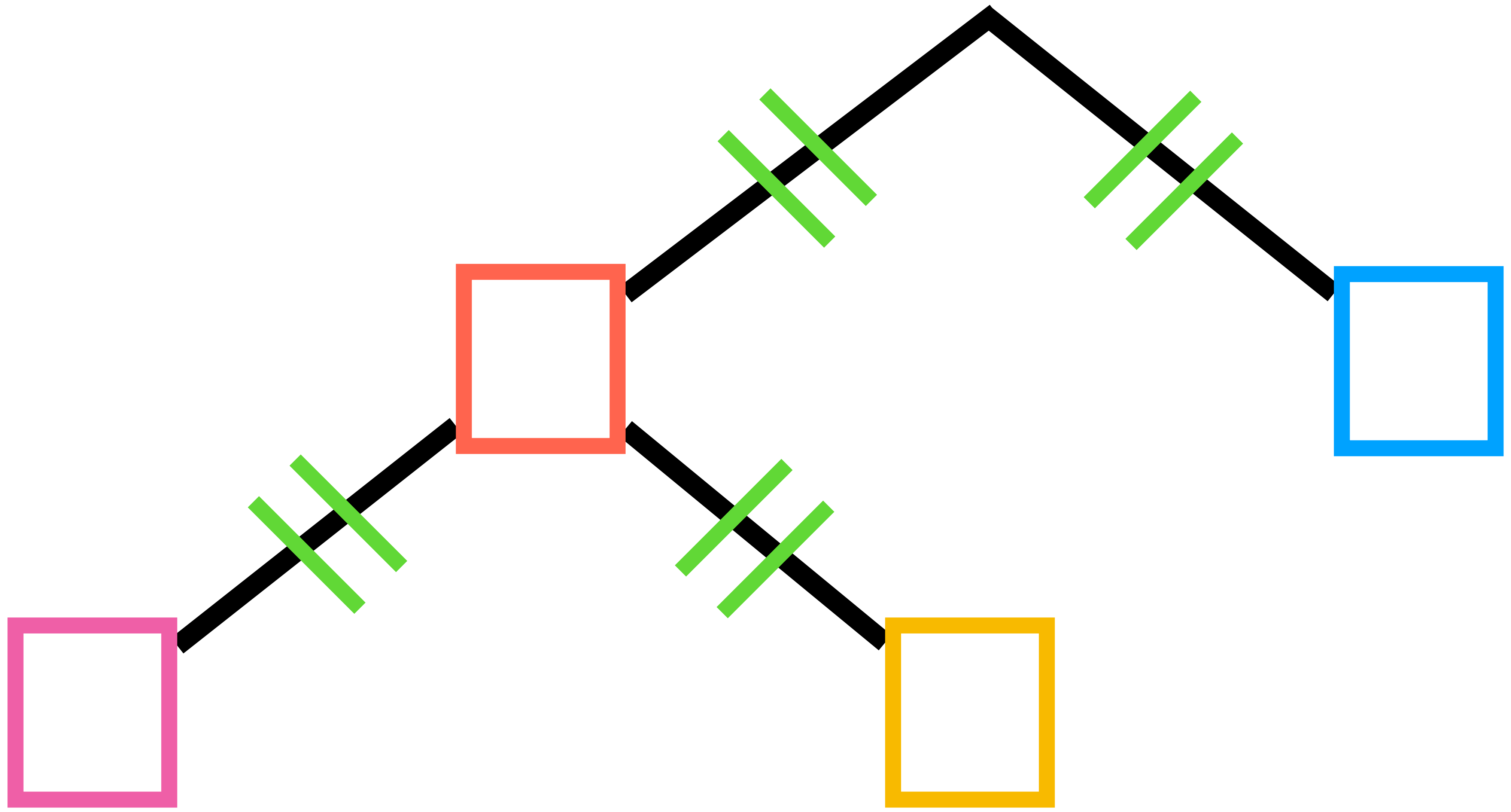

Oswego
Fulton
Brewerton
Baldwinsville
Syracuse
Newark
Farmington
Auburn
Pompey
Canandaigua
Geneva
Penn Yan
Cortland
Bath
Horseheads
Elmira
Barton



Goal:
interleave R and B;
interleave P and T.



Goal:
interleave R and B;
interleave P and T.



Goal:

interleave R and B;

  interleave P and T.

Goal:

interleave R and B;

  interleave P and T.

B_3, B_2, P_2, B_1, P_1

Goal:

interleave R and B;

interleave P and T.

$T_1 \longrightarrow B_3, B_2, P_2, B_1, P_1$

Goal:

interleave R and B;

interleave P and T.

$T_1 \longrightarrow B_3, B_2, P_2, B_1, P_1$

$B_3, B_2, P_2, T_1, B_1, P_1$

Goal:
interleave R and B;
  interleave P and T.

$T_1 \longrightarrow B_3, B_2, P_2, B_1, P_1$

$B_3, B_2, P_2, T_1, B_1, P_1$

$B_3, B_2, P_2, B_1, T_1, P_1$

Goal:

interleave R and B;

  interleave P and T.

$T_1 \longrightarrow B_3, B_2, P_2, B_1, P_1$

$B_3, B_2, \blacksquare, \blacksquare, B_1, \blacksquare$

$B_3, B_2, \blacksquare, B_1, \blacksquare, \blacksquare$

Goal:
interleave R and B;
interleave P and T.

$B_3, B_2, P_2, T_1, B_1, P_1$
$B_3, B_2, P_2, B_1, T_1, P_1$

$T_1 \longrightarrow B_3, B_2, P_2, B_1, P_1$

Goal:

interleave R and B;

interleave P and T.

~~$B_3, B_2, P_2, T_1, B_1, P_1$~~

~~$B_3, B_2, P_2, B_1, T_1, P_1$~~

$T_1 \longrightarrow B_3, B_2, P_2, B_1, P_1$

Goal:
interleave R and B;
interleave P and T.

~~B_3, B_2, P_2, T_1, B_1, P_1~~
~~B_3, B_2, P_2, B_1, T_1, P_1~~

$T_1 \longrightarrow B_3, B_2, P_2, B_1, P_1$

B_3, P_2, B_2, T_1, B_1, P_1

Goal:

interleave R and B;

interleave P and T.

~~$B_3, B_2, P_2, T_1, B_1, P_1$~~

~~$B_3, B_2, P_2, B_1, T_1, P_1$~~

$T_1 \longrightarrow B_3, B_2, P_2, B_1, P_1$

B_3, , B_2, , B_1,

Goal:
interleave R and B;
interleave P and T.

~~$B_3, B_2, P_2, T_1, B_1, P_1$~~
~~$B_3, B_2, P_2, B_1, T_1, P_1$~~

$T_1 \longrightarrow B_3, B_2, P_2, B_1, P_1$

$B_3, P_2, B_2, T_1, B_1, P_1$

Goal:
interleave R and B;
interleave P and T.

~~$B_3, B_2, P_2, T_1, B_1, P_1$~~

~~$B_3, B_2, P_2, B_1, T_1, P_1$~~

$T_1 \longrightarrow B_3, B_2, P_2, B_1, P_1$

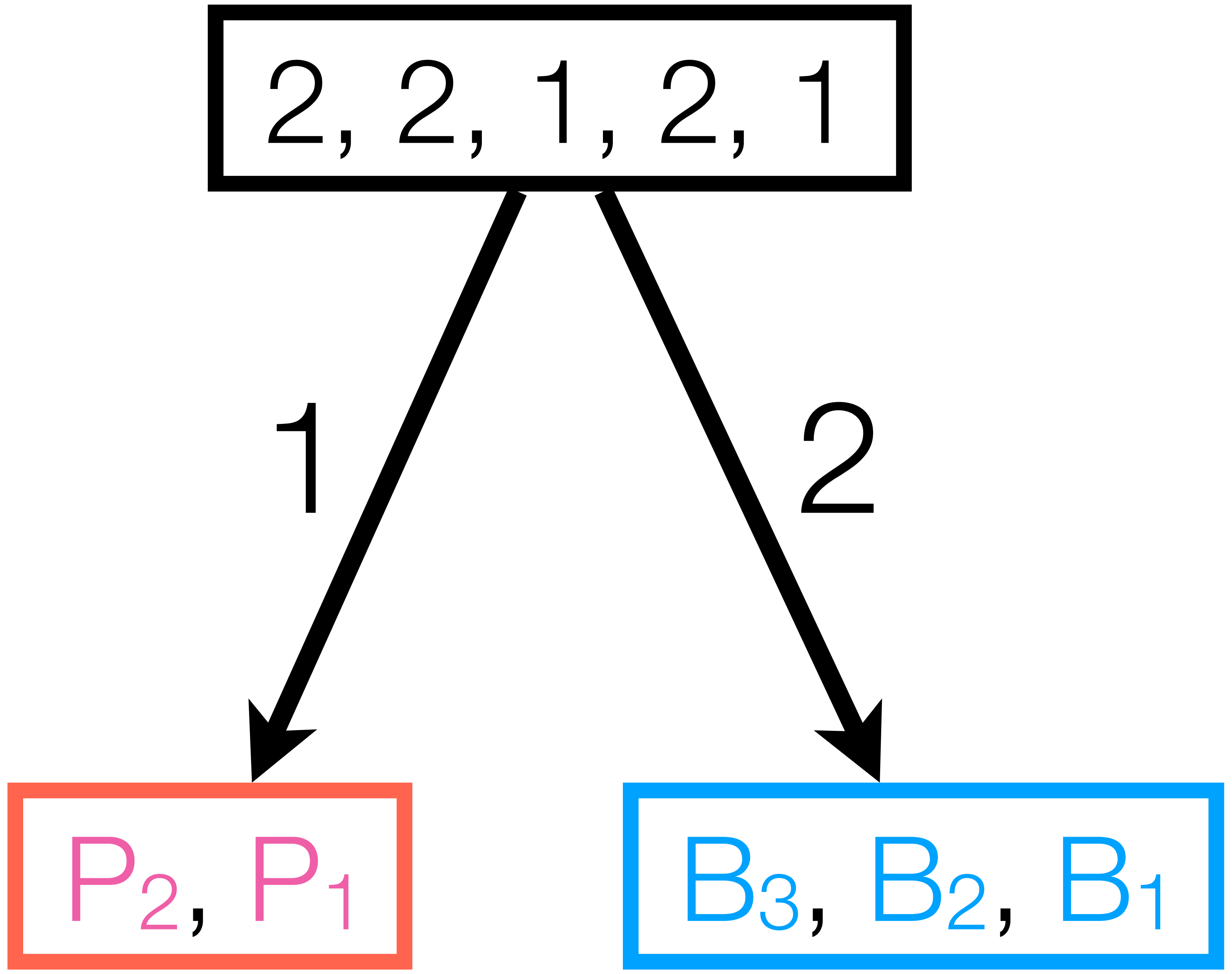
$B_3, P_2, B_2, T_1, B_1, P_1$

Enqueueing a packet can require the reordering of buffered packets.

No PIFO can do this.

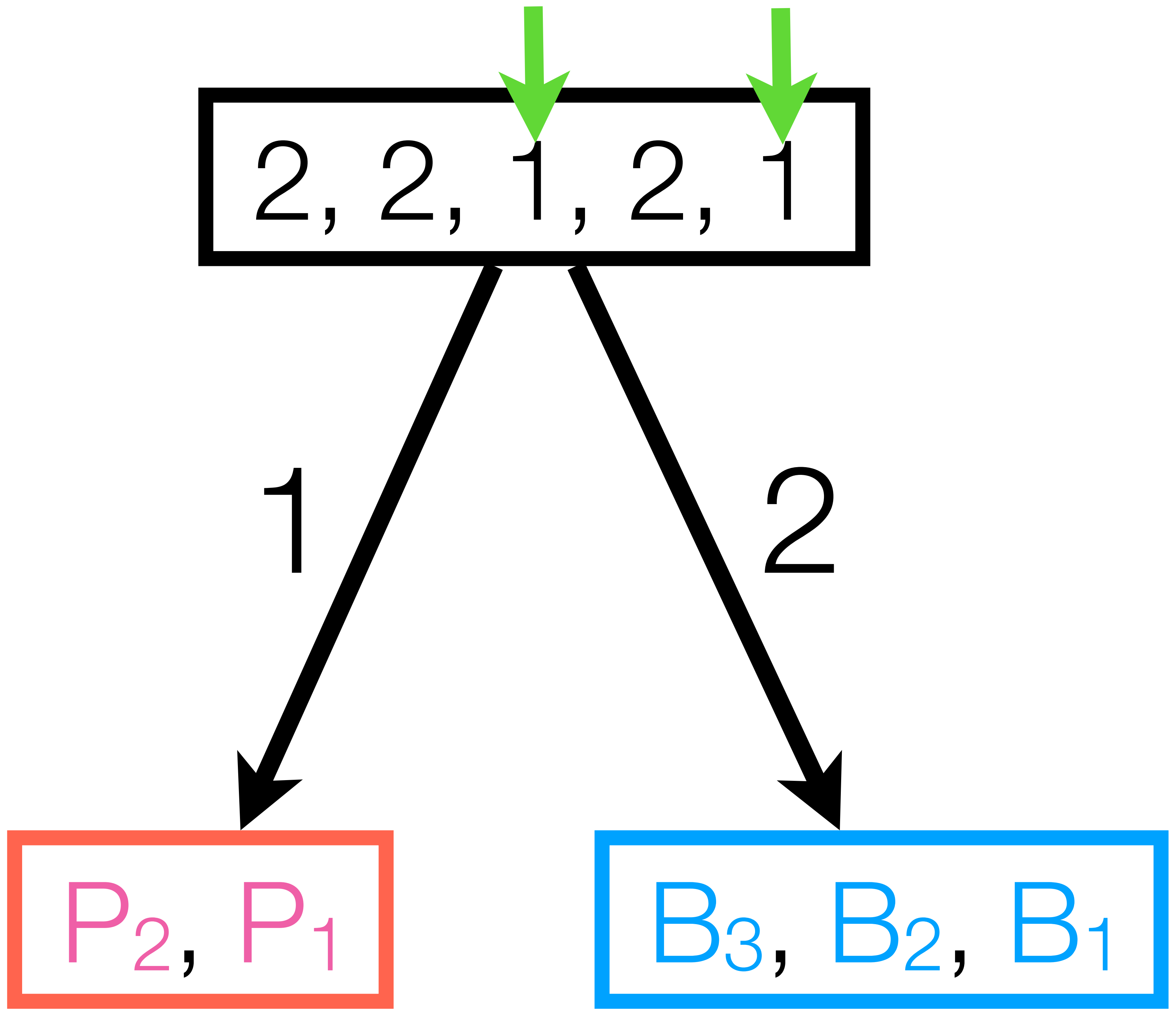
Introducing: PIFO trees

Interleave R and B;
interleave P and T.



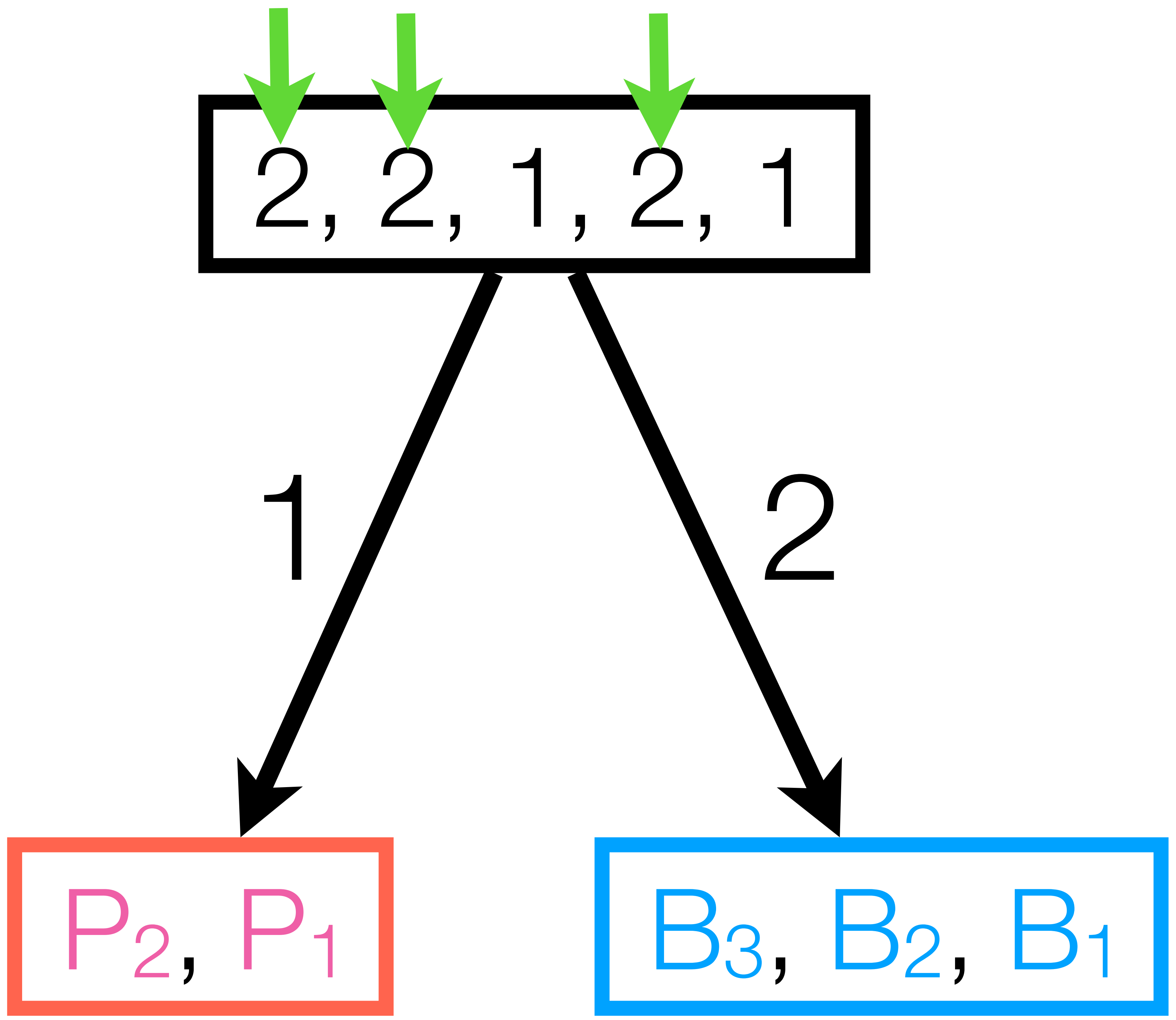
Introducing: PIFO trees

Interleave R and B;
interleave P and T.



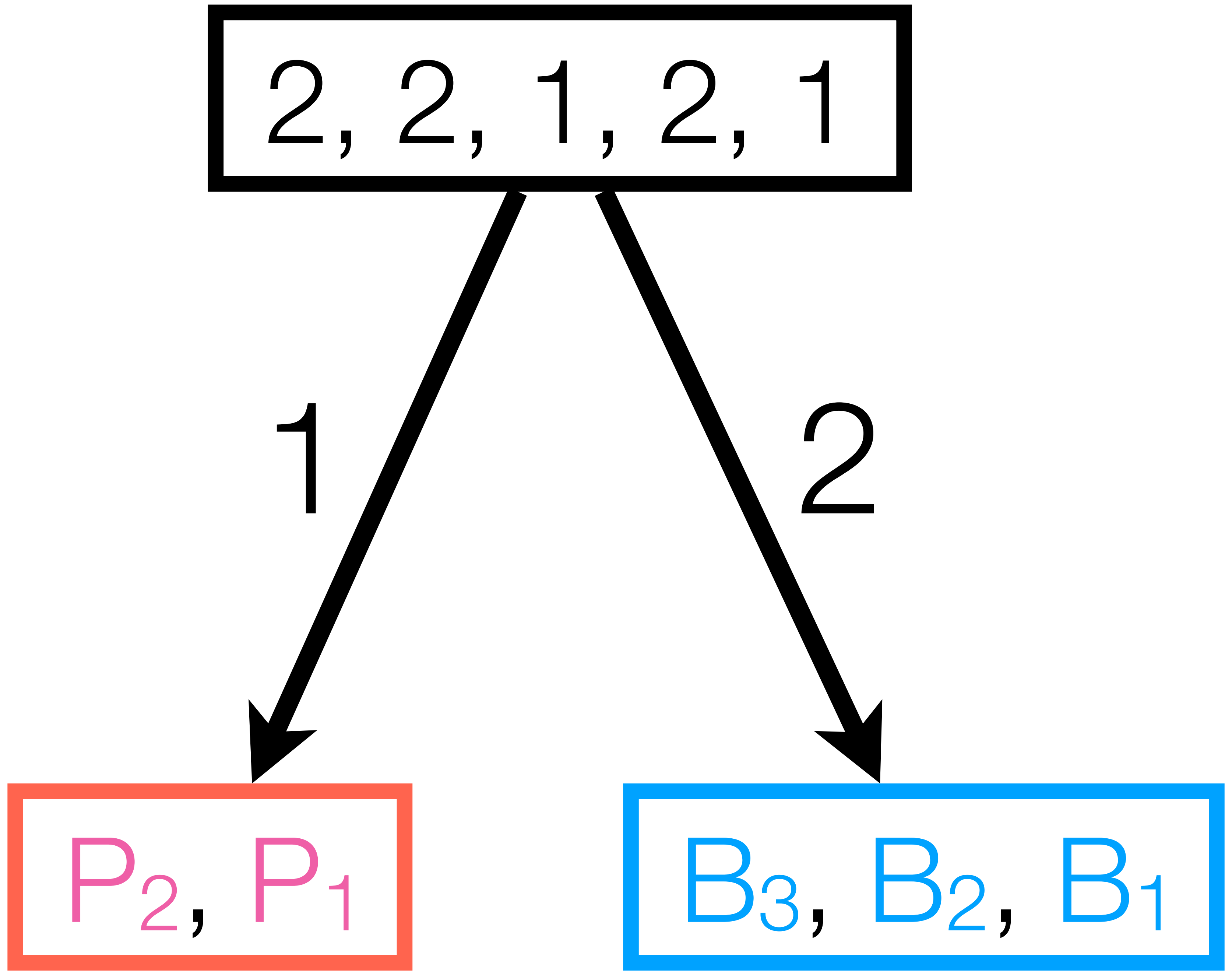
Introducing: PIFO trees

Interleave R and B;
interleave P and T.



Introducing: PIFO trees

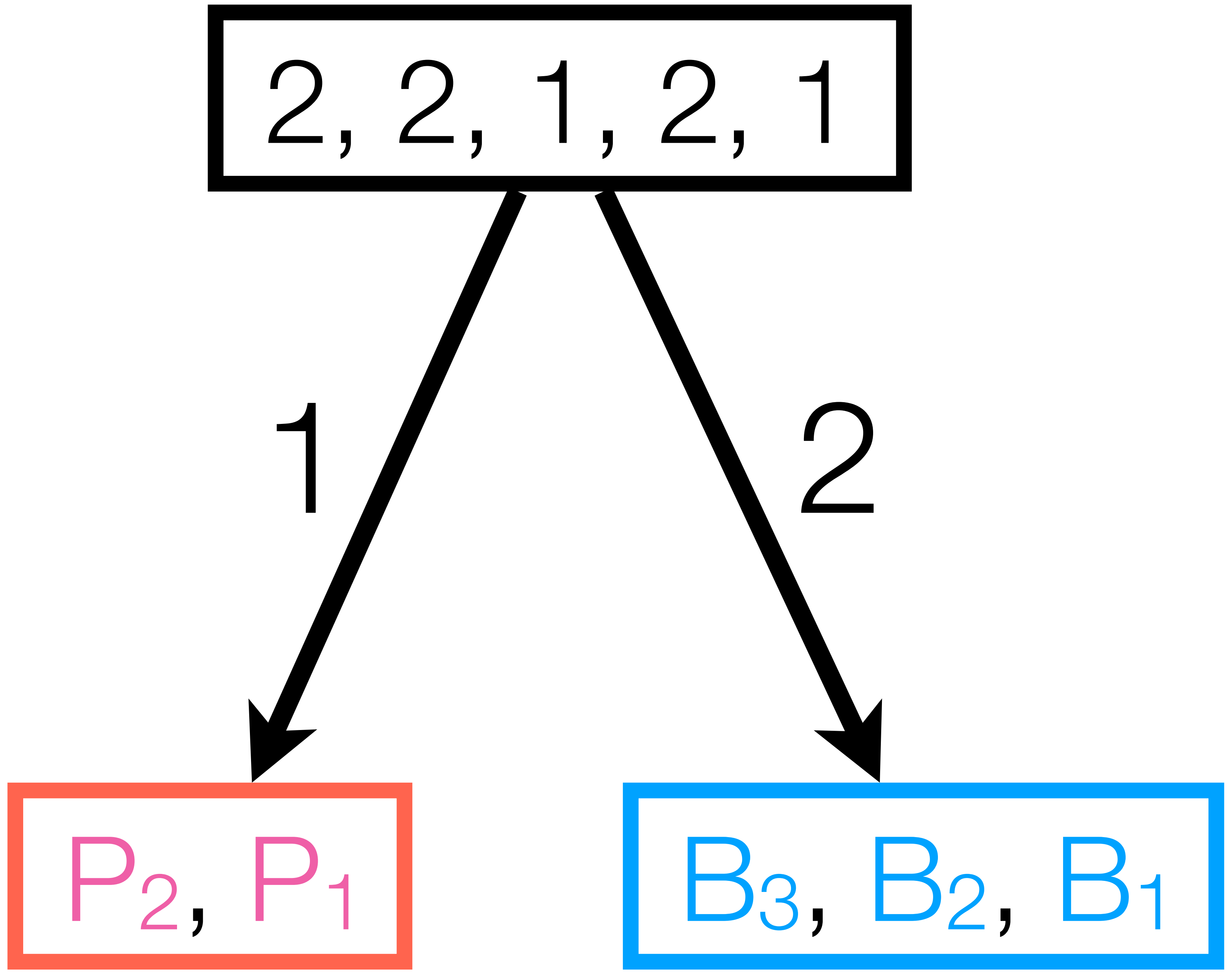
Interleave R and B;
interleave P and T.



This behaves like a queue!

Introducing: PIFO trees

Interleave R and B;
interleave P and T.

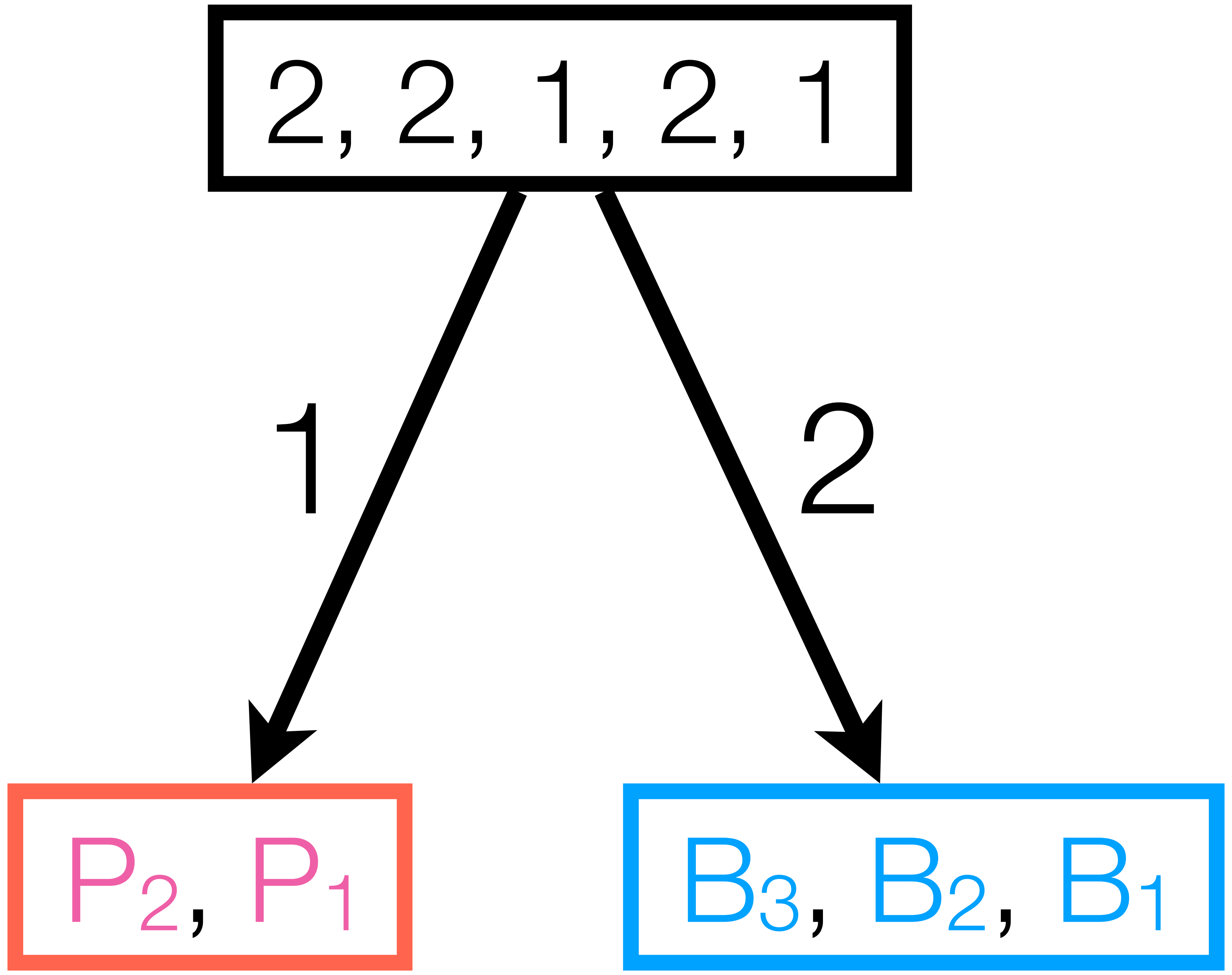


This behaves like a queue!

How do we pop it?

Introducing: PIFO trees

Interleave R and B;
interleave P and T.

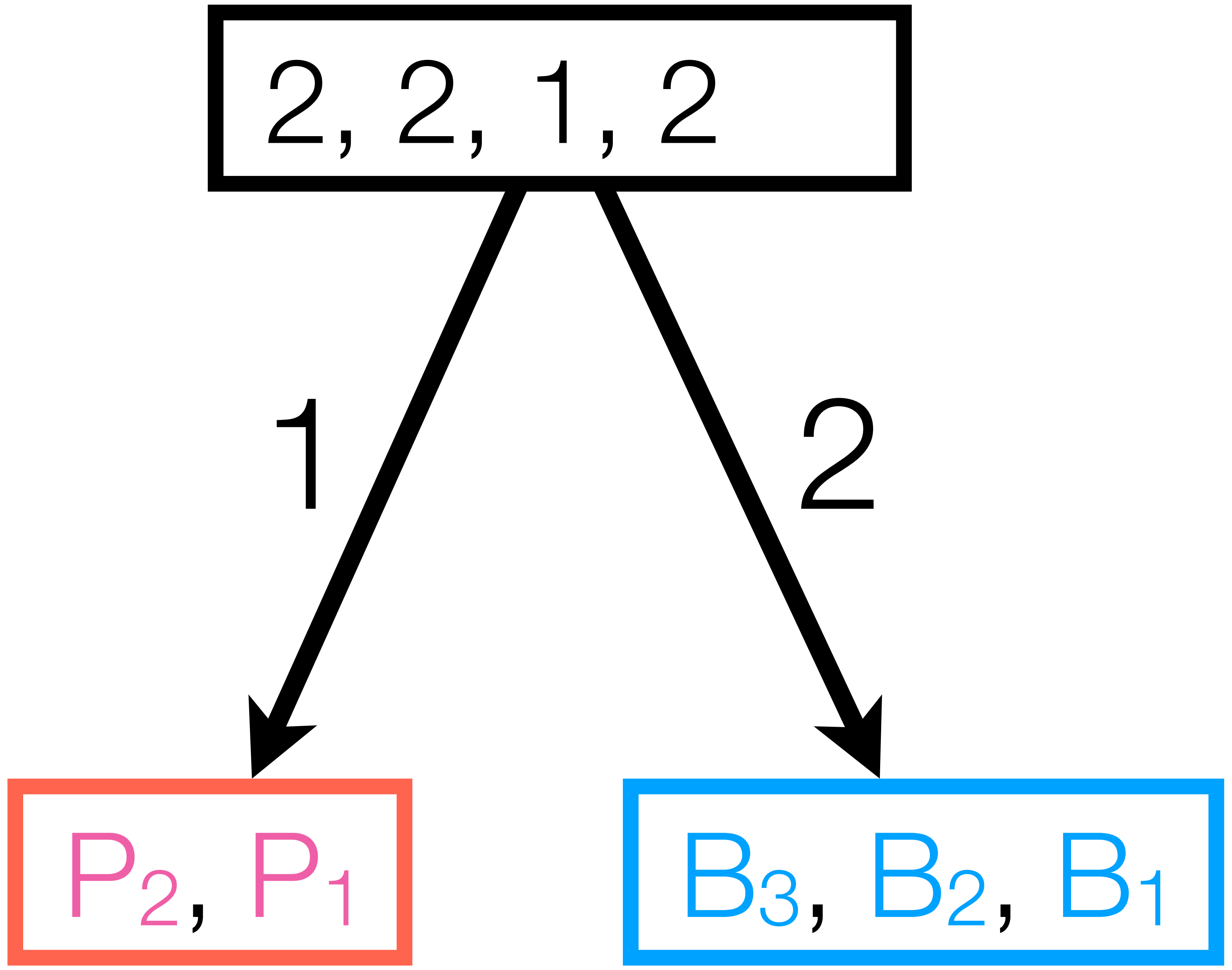


This behaves like a queue!

How do we pop it?

Introducing: PIFO trees

Interleave R and B;
interleave P and T.

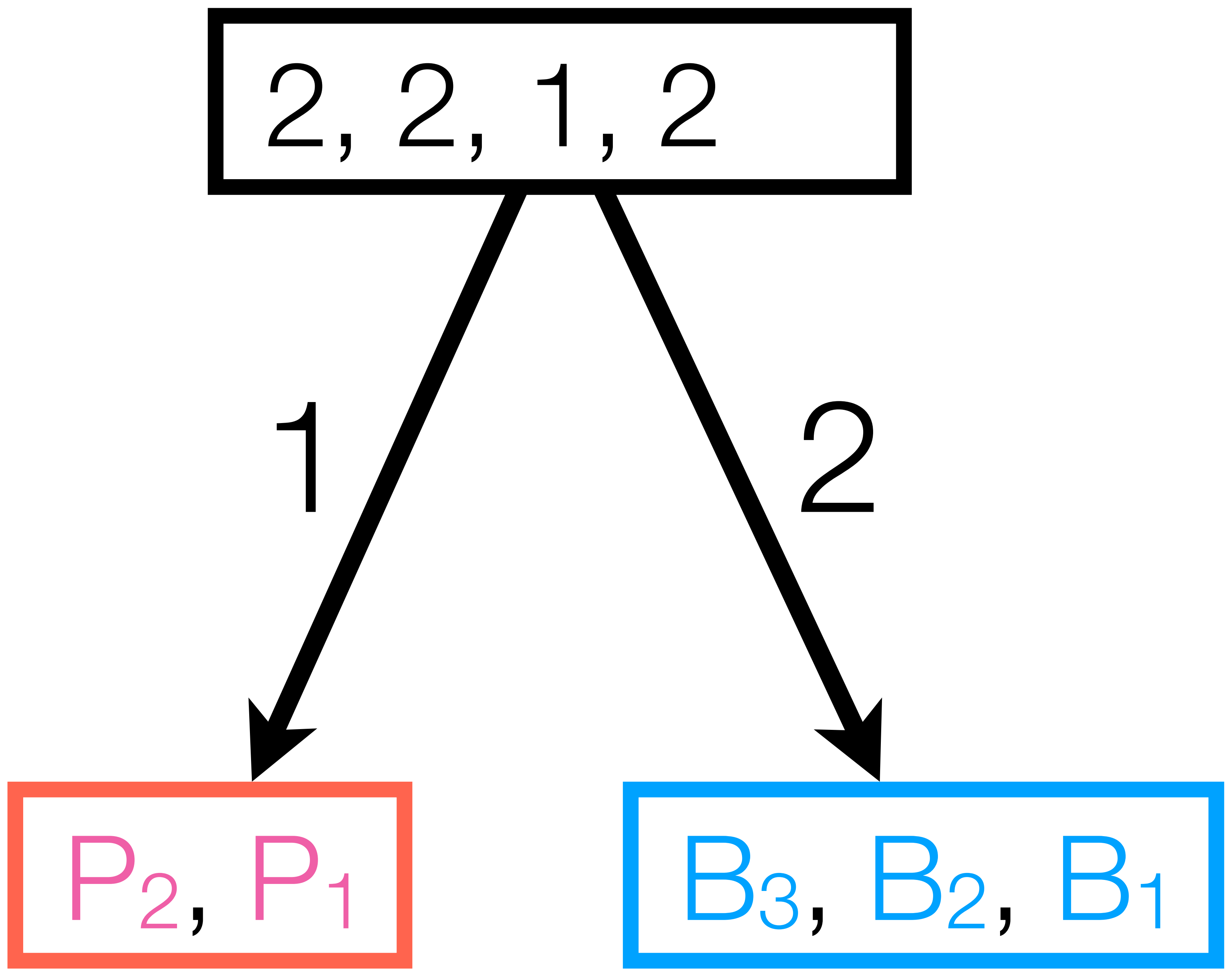


This behaves like a queue!

How do we pop it?

Introducing: PIFO trees

Interleave R and B;
interleave P and T.

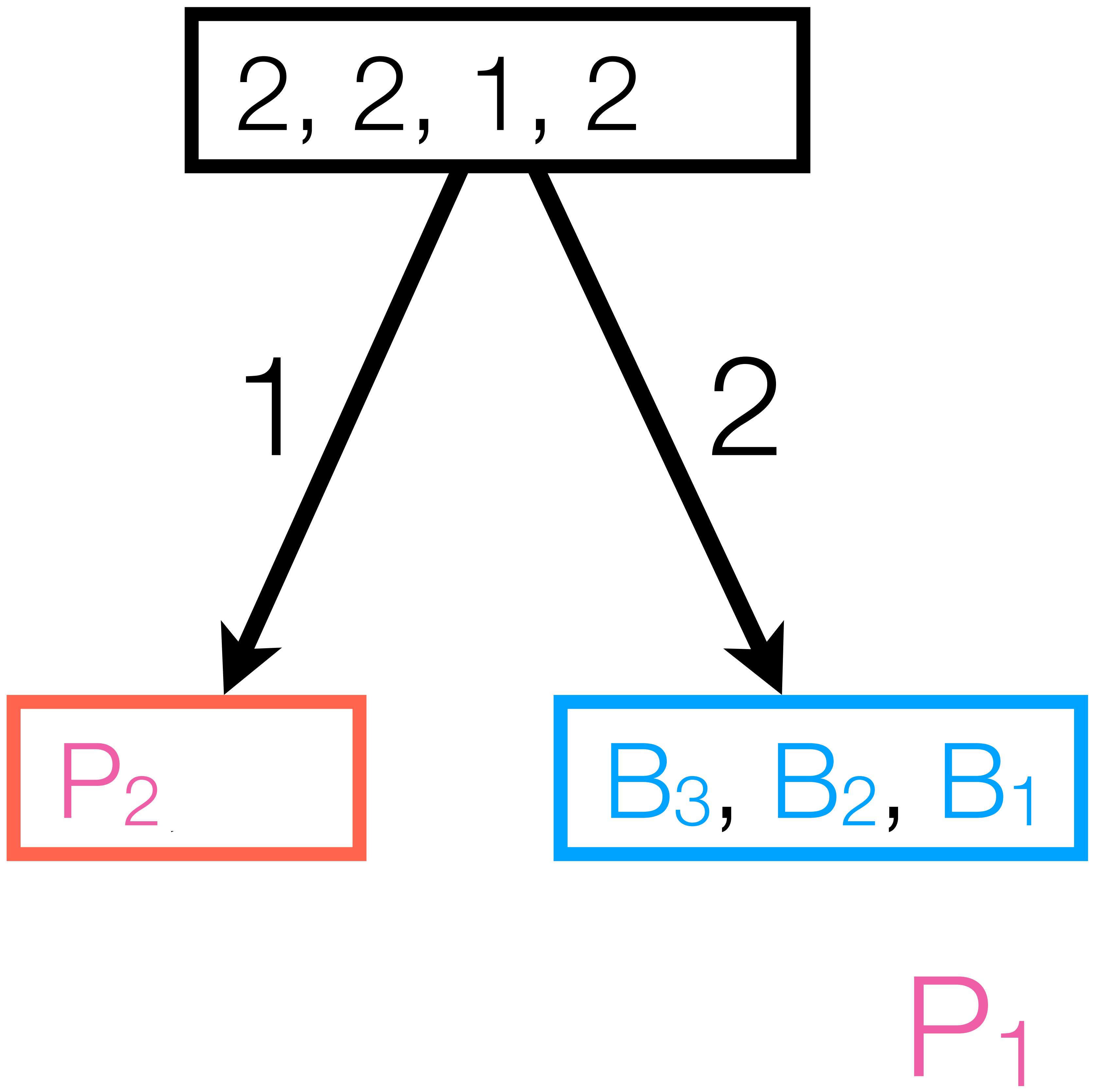


This behaves like a queue!

How do we pop it?

Introducing: PIFO trees

Interleave R and B;
interleave P and T.

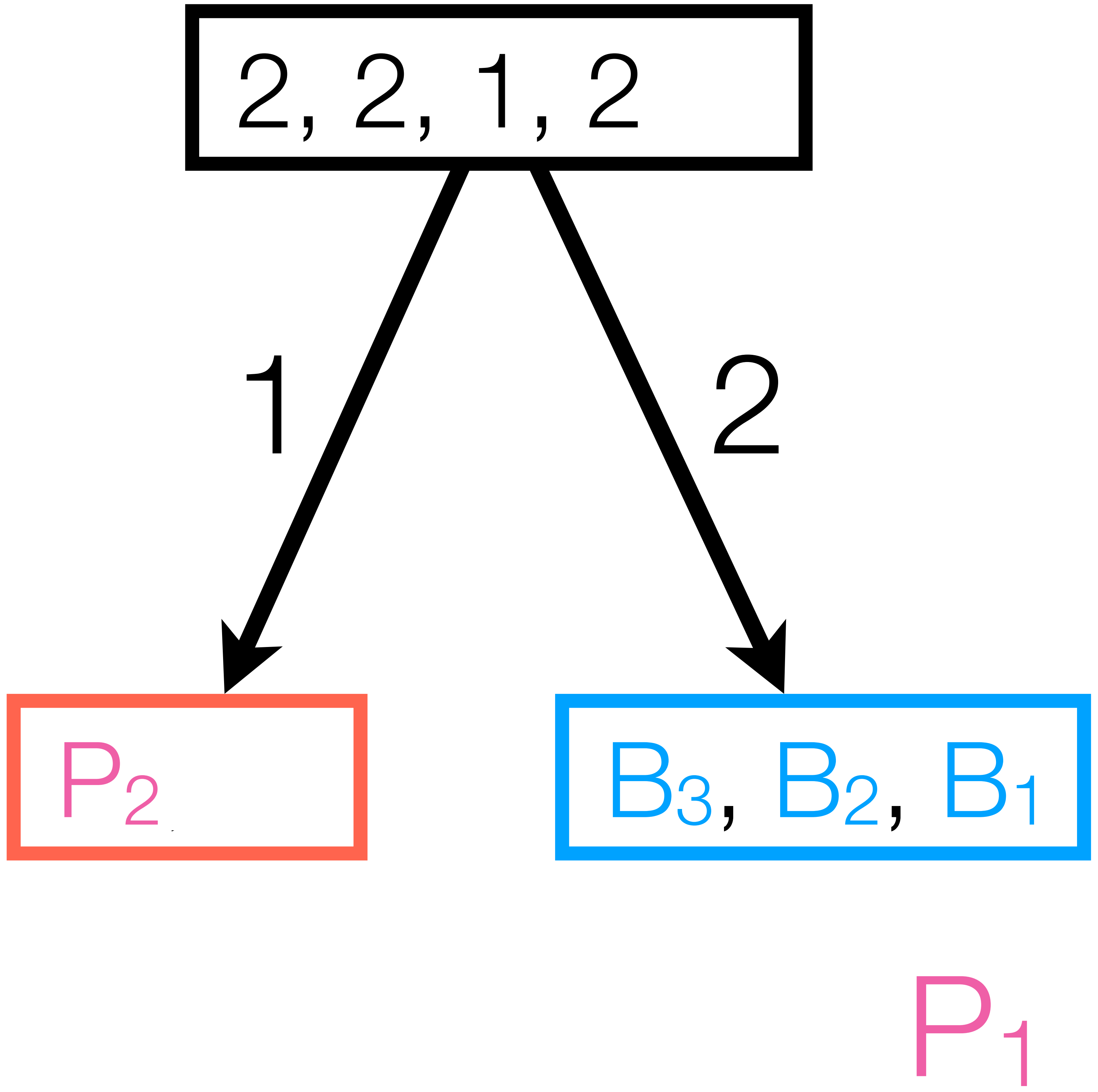


This behaves like a queue!

How do we pop it?

Introducing: PIFO trees

Interleave R and B;
interleave P and T.

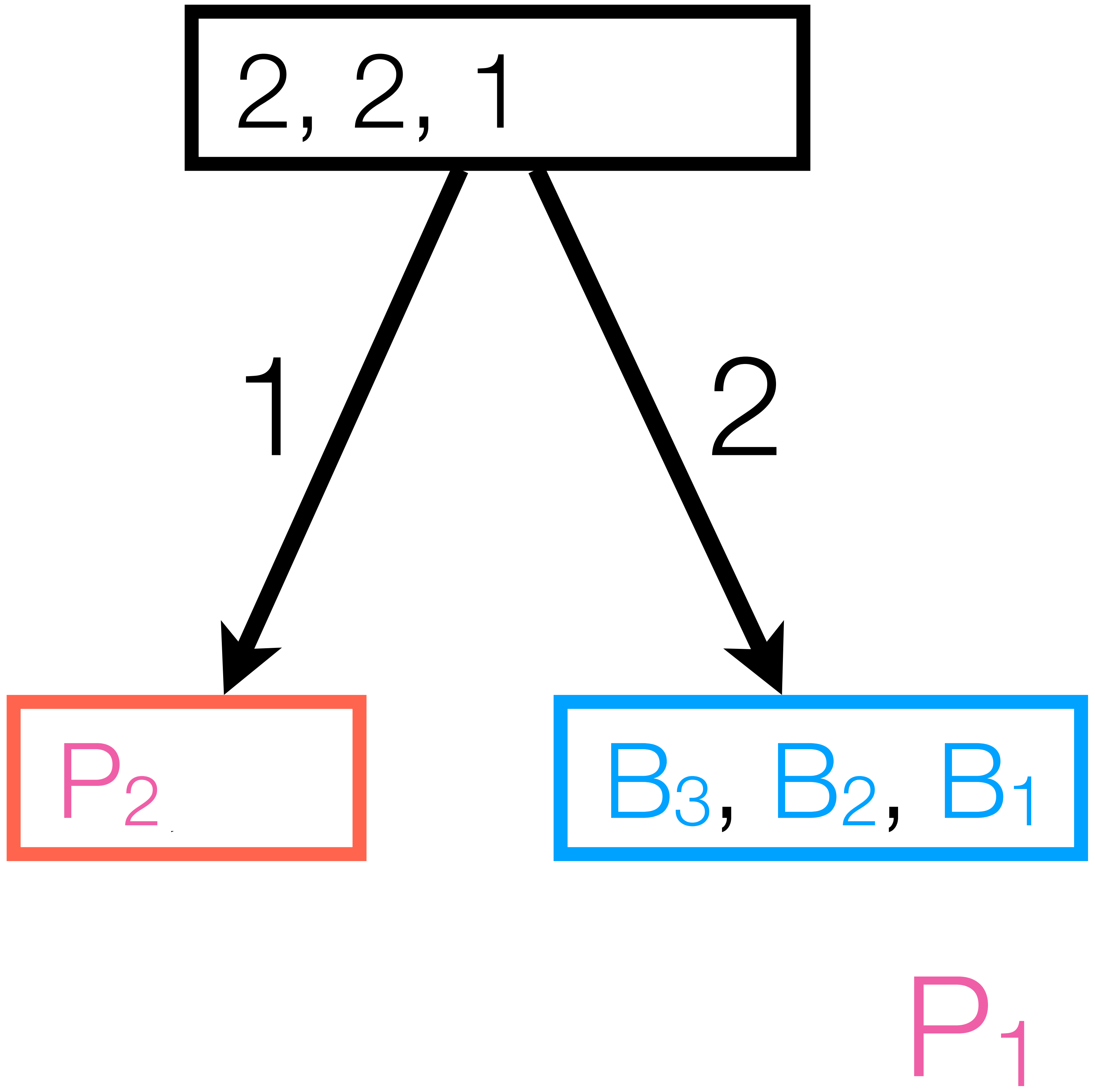


This behaves like a queue!

How do we pop it?

Introducing: PIFO trees

Interleave R and B;
interleave P and T.

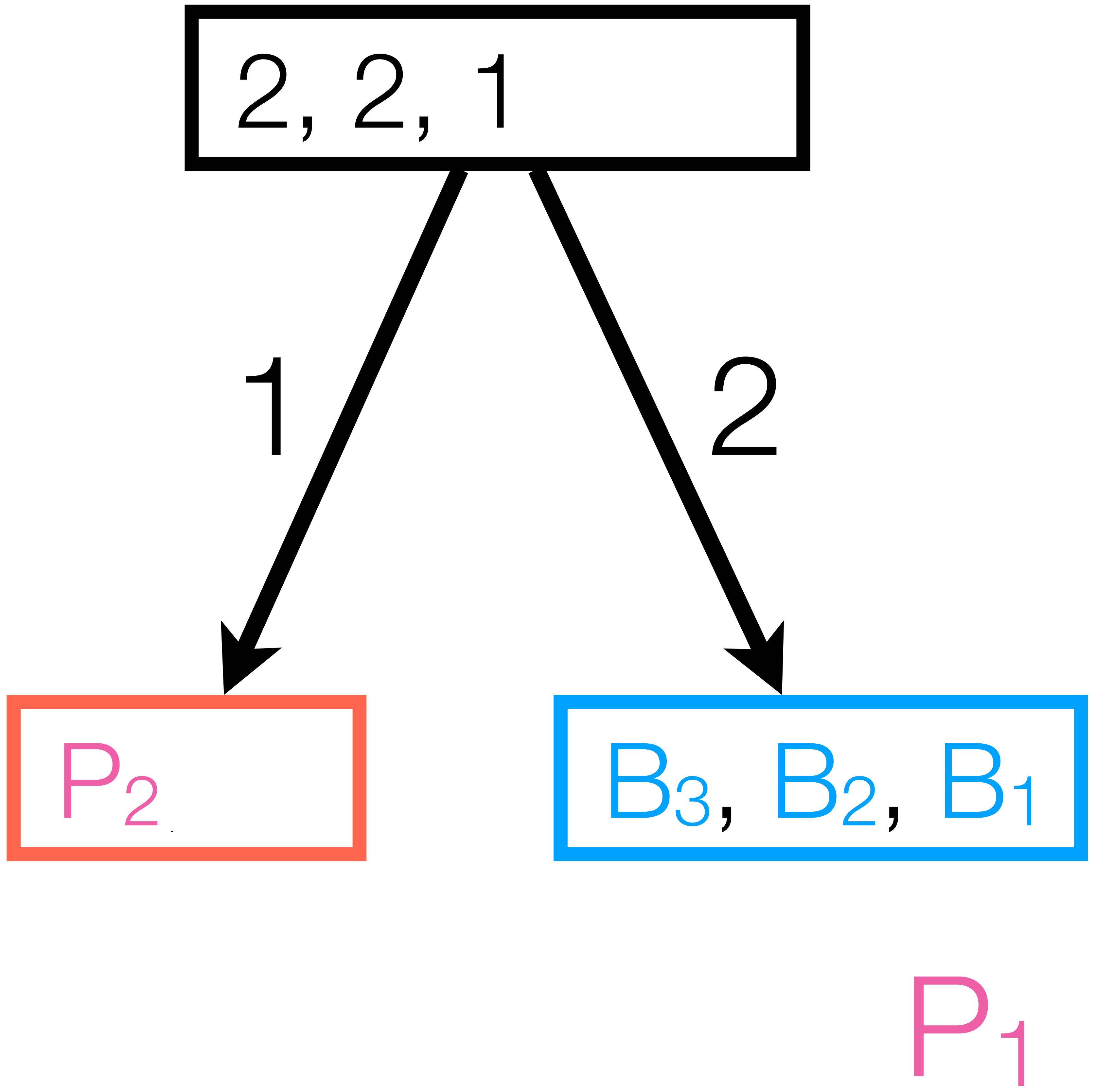


This behaves like a queue!

How do we pop it?

Introducing: PIFO trees

Interleave R and B;
interleave P and T.

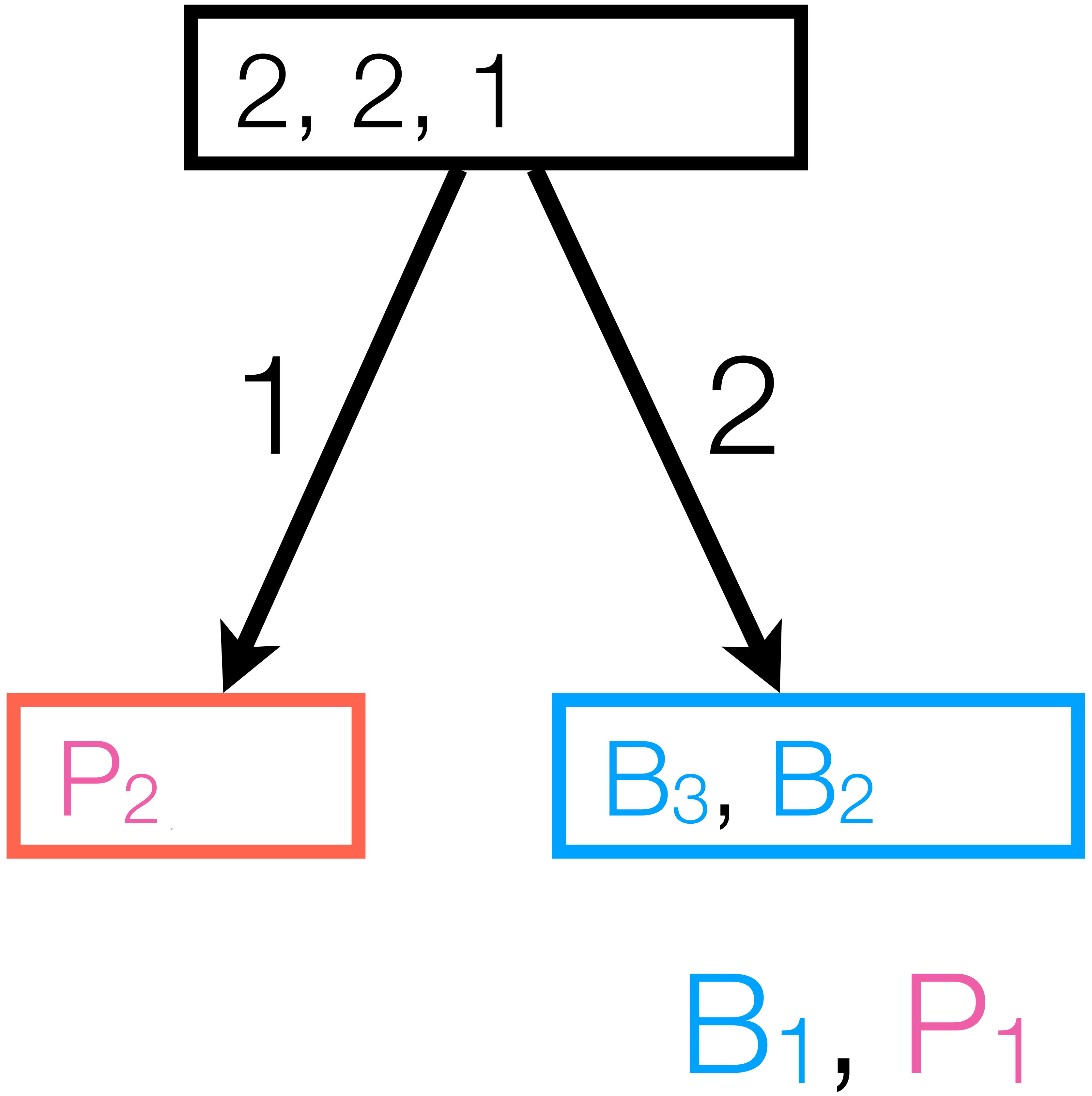


This behaves like a queue!

How do we pop it?

Introducing: PIFO trees

Interleave R and B;
interleave P and T.

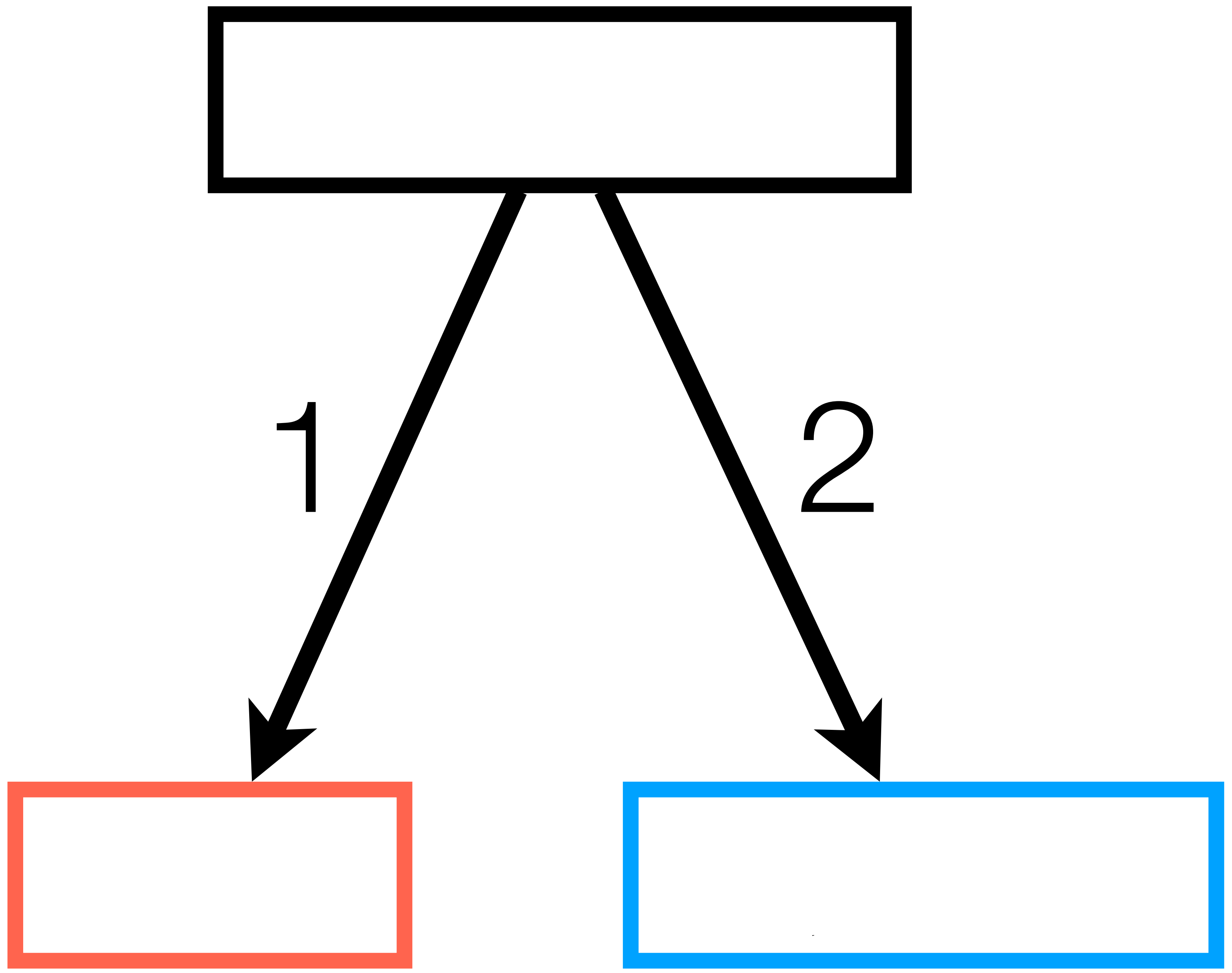


This behaves like a queue!

How do we pop it?

Introducing: PIFO trees

Interleave R and B;
interleave P and T.



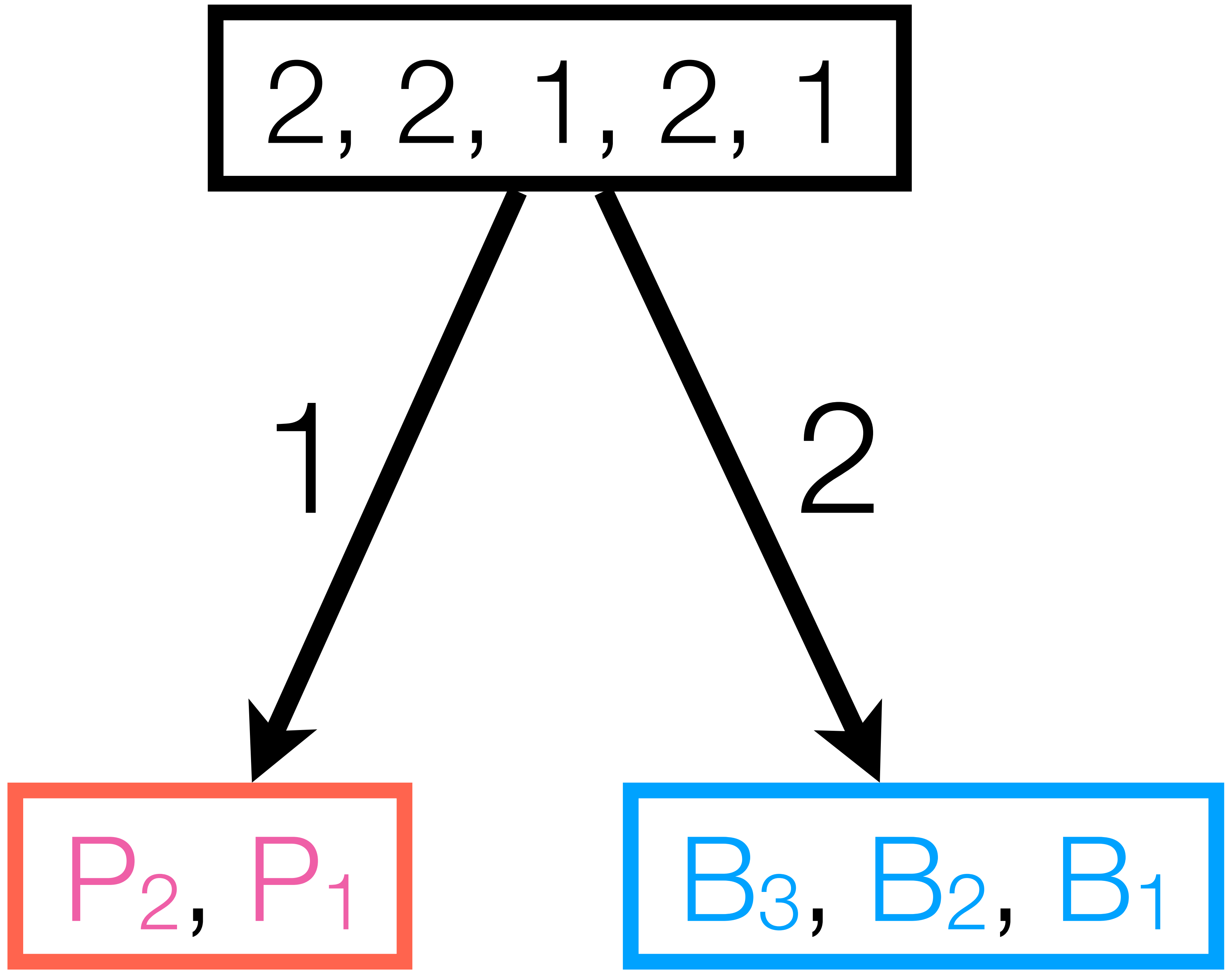
B_3, B_2, P_2, B_1, P_1

This behaves like a queue!

How do we pop it?

Introducing: PIFO trees

Interleave R and B;
interleave P and T.



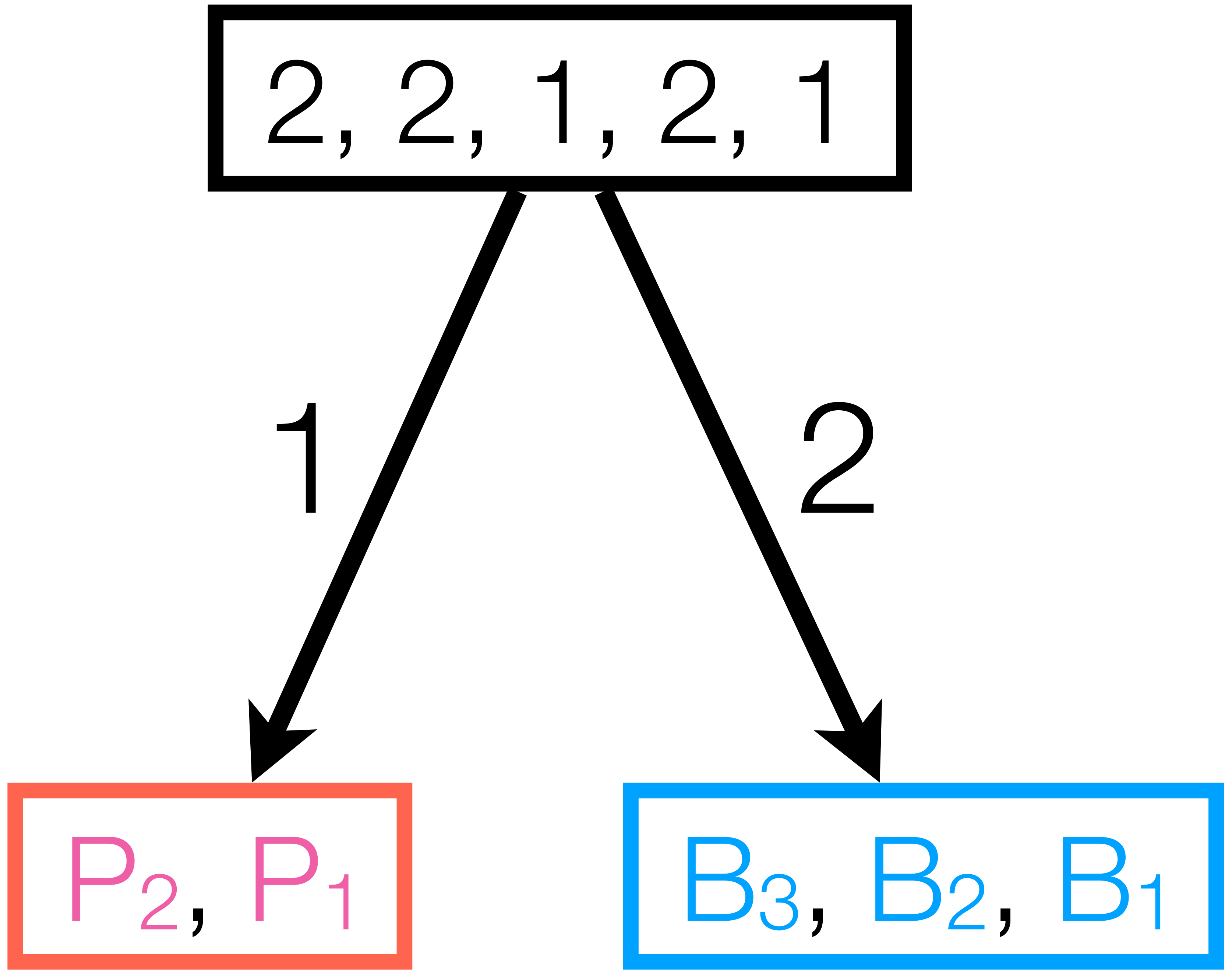
B_3, B_2, P_2, B_1, P_1

This behaves like a queue!

How do we pop it?

Introducing: PIFO trees

Interleave R and B;
interleave P and T.



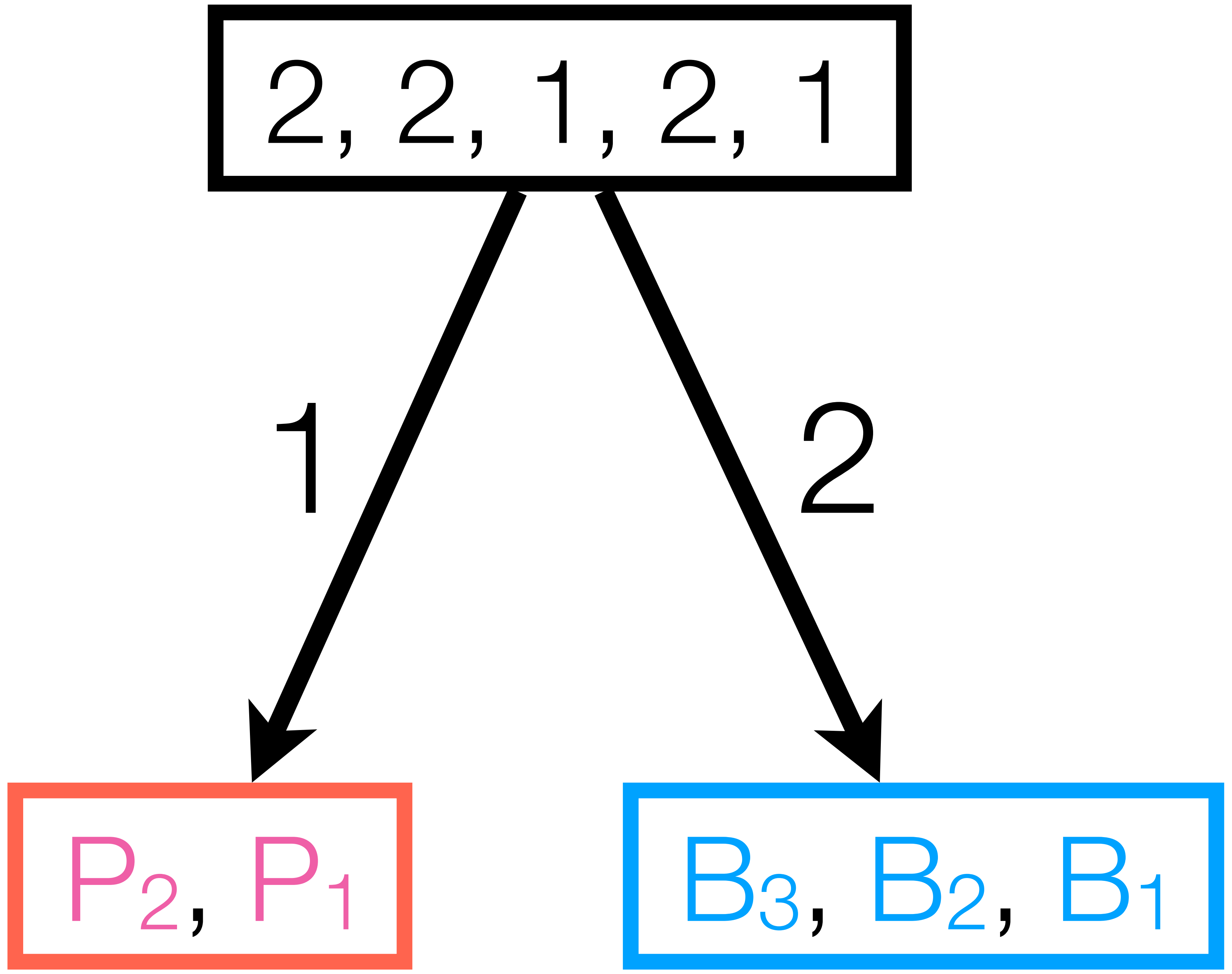
B_3, B_2, $\overline{P_2}$, B_1, $\overline{P_1}$

This behaves like a queue!

How do we pop it?

Introducing: PIFO trees

Interleave R and B;
interleave P and T.



$B_3, B_2, \overline{P_2}, B_1, \overline{P_1}$

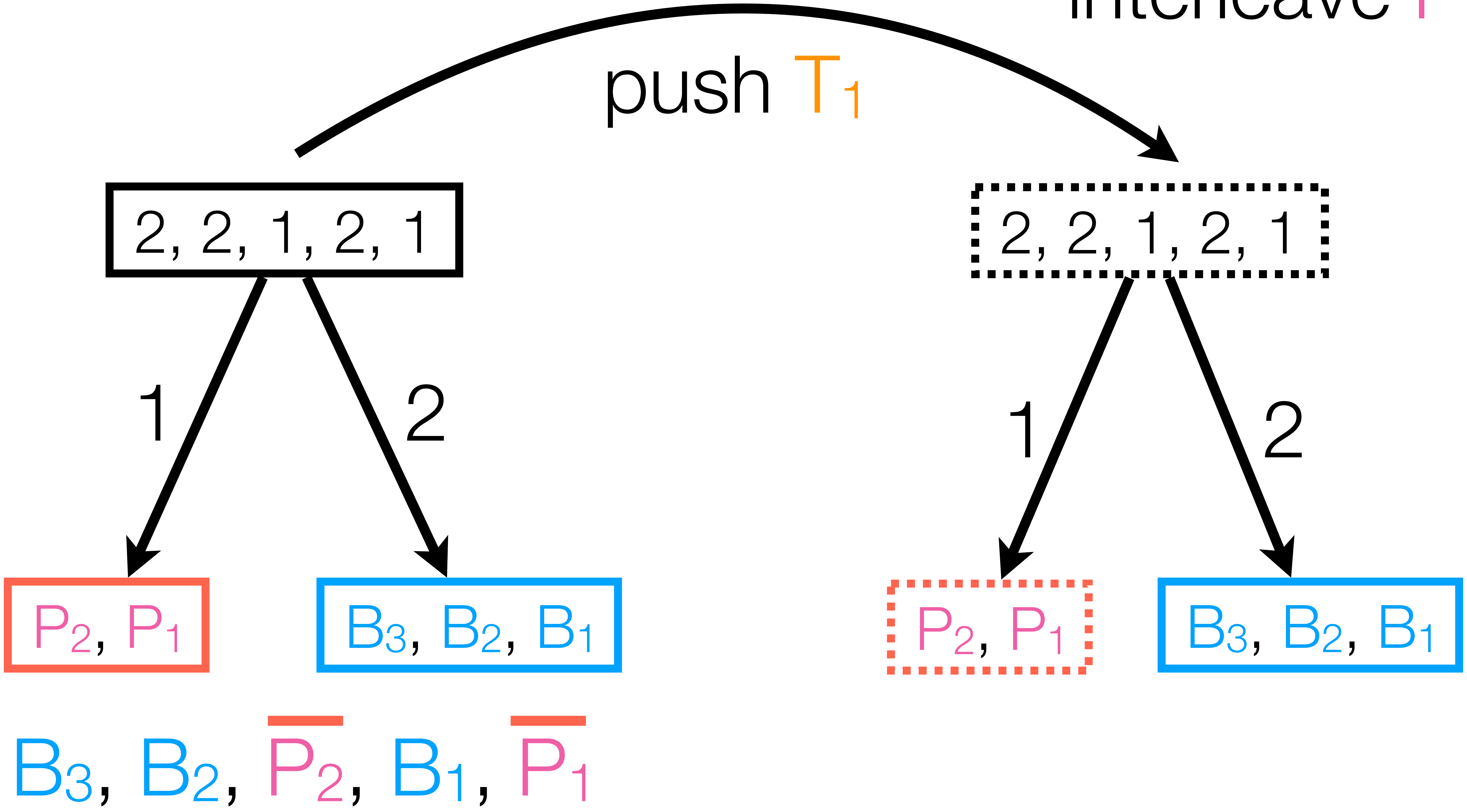
This behaves like a queue!

How do we pop it?

How do we push into it?

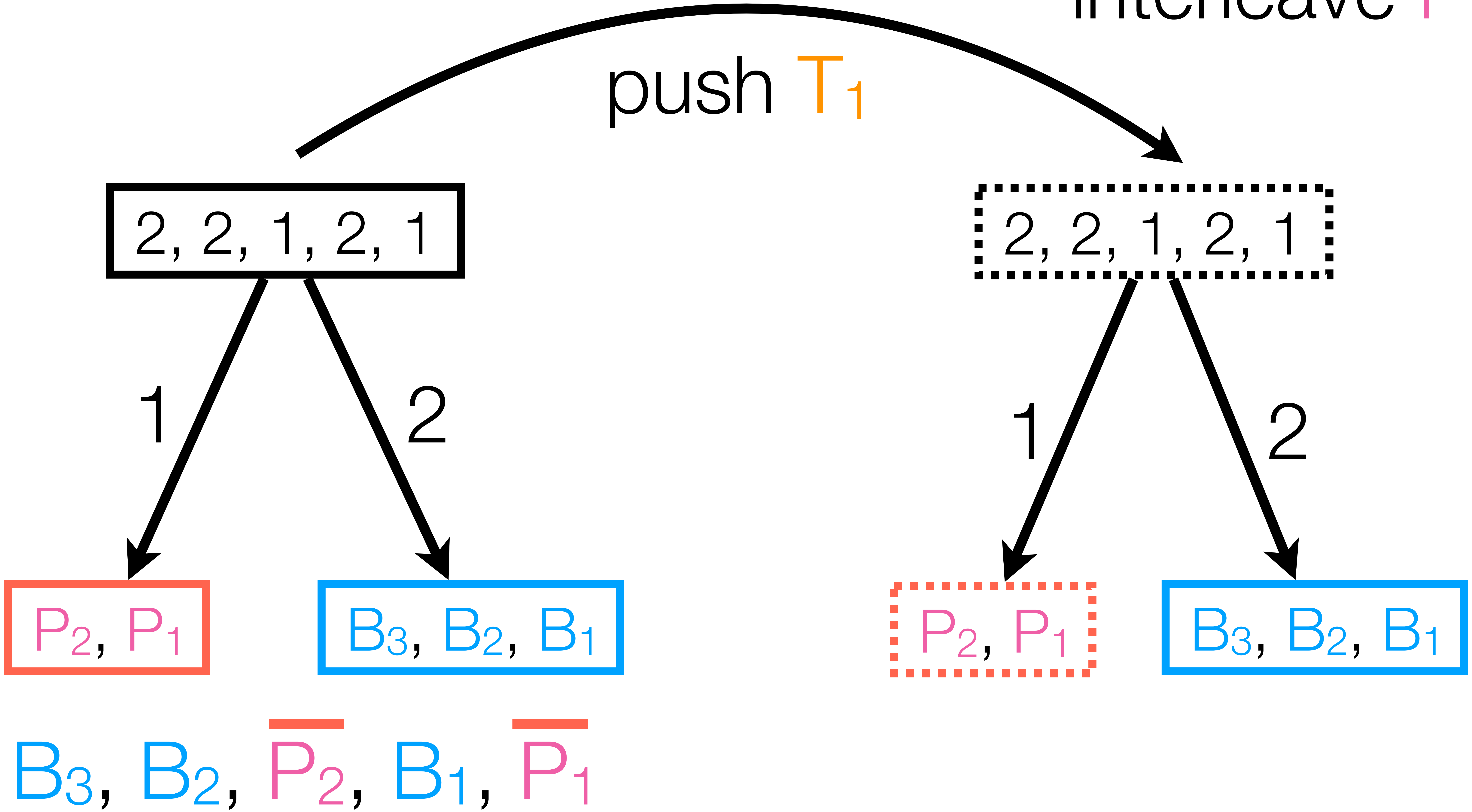
Introducing: PIFO trees

Interleave R and B;
interleave P and T.



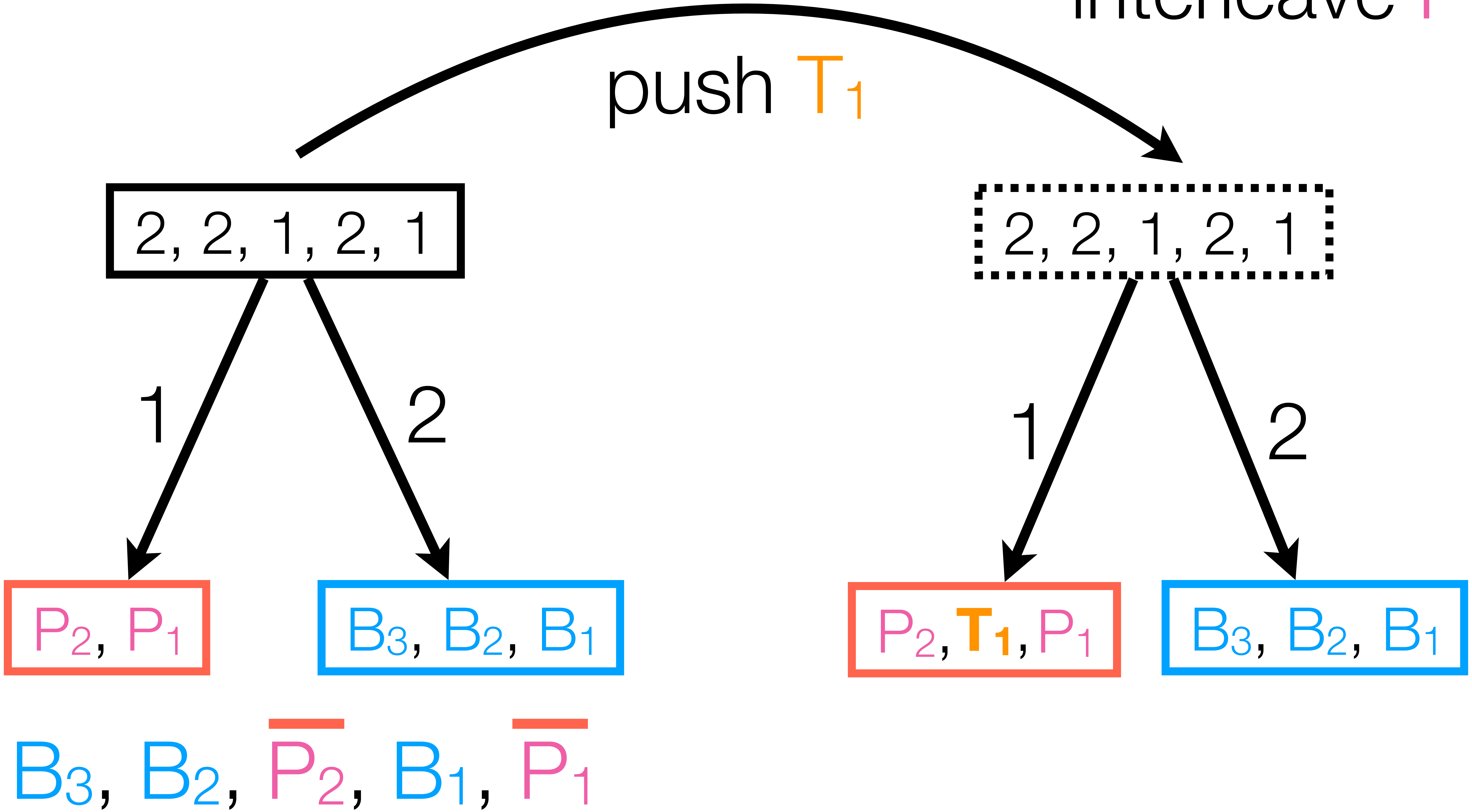
Introducing: PIFO trees

Interleave R and B;
interleave P and T.



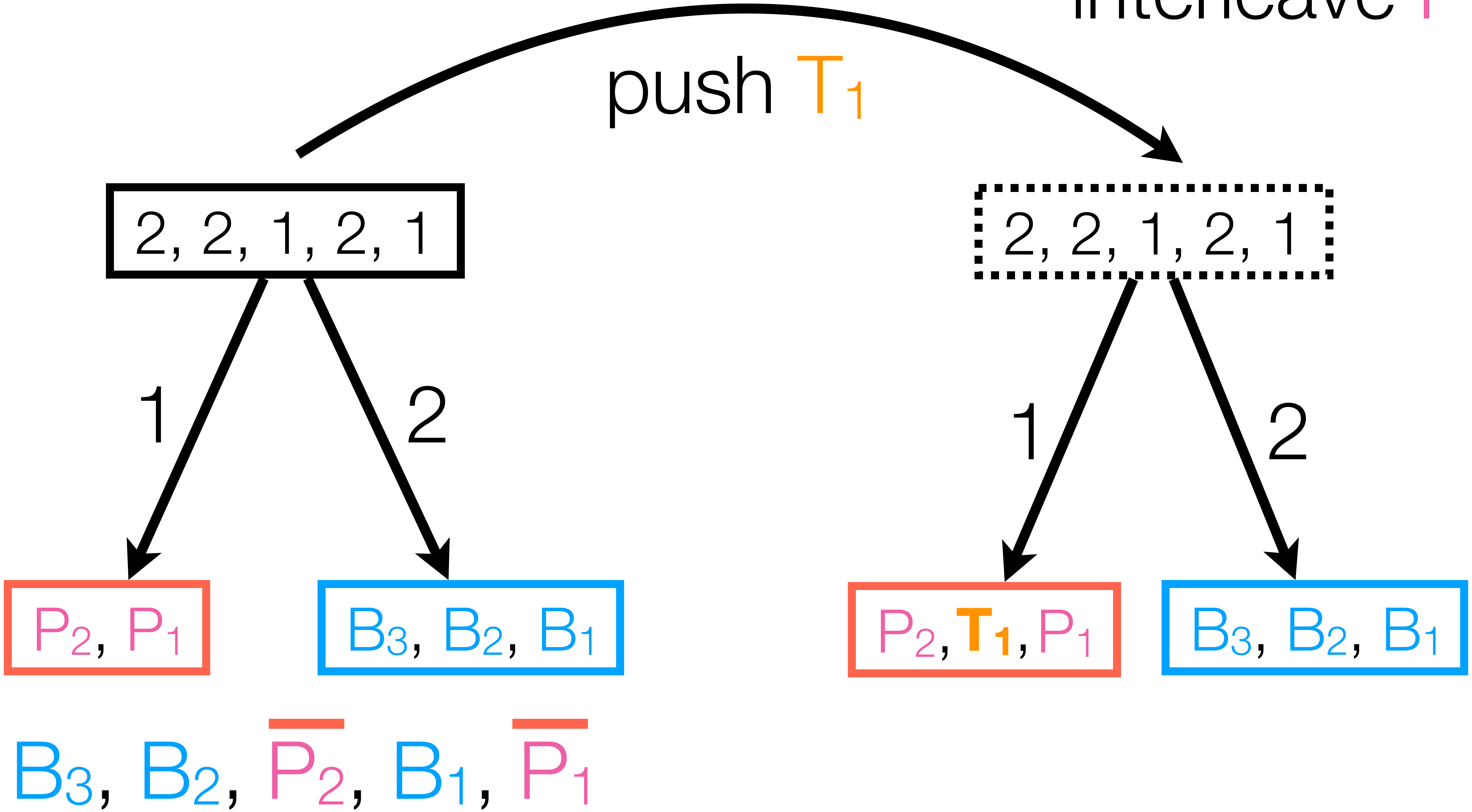
Introducing: PIFO trees

Interleave R and B;
interleave P and T.



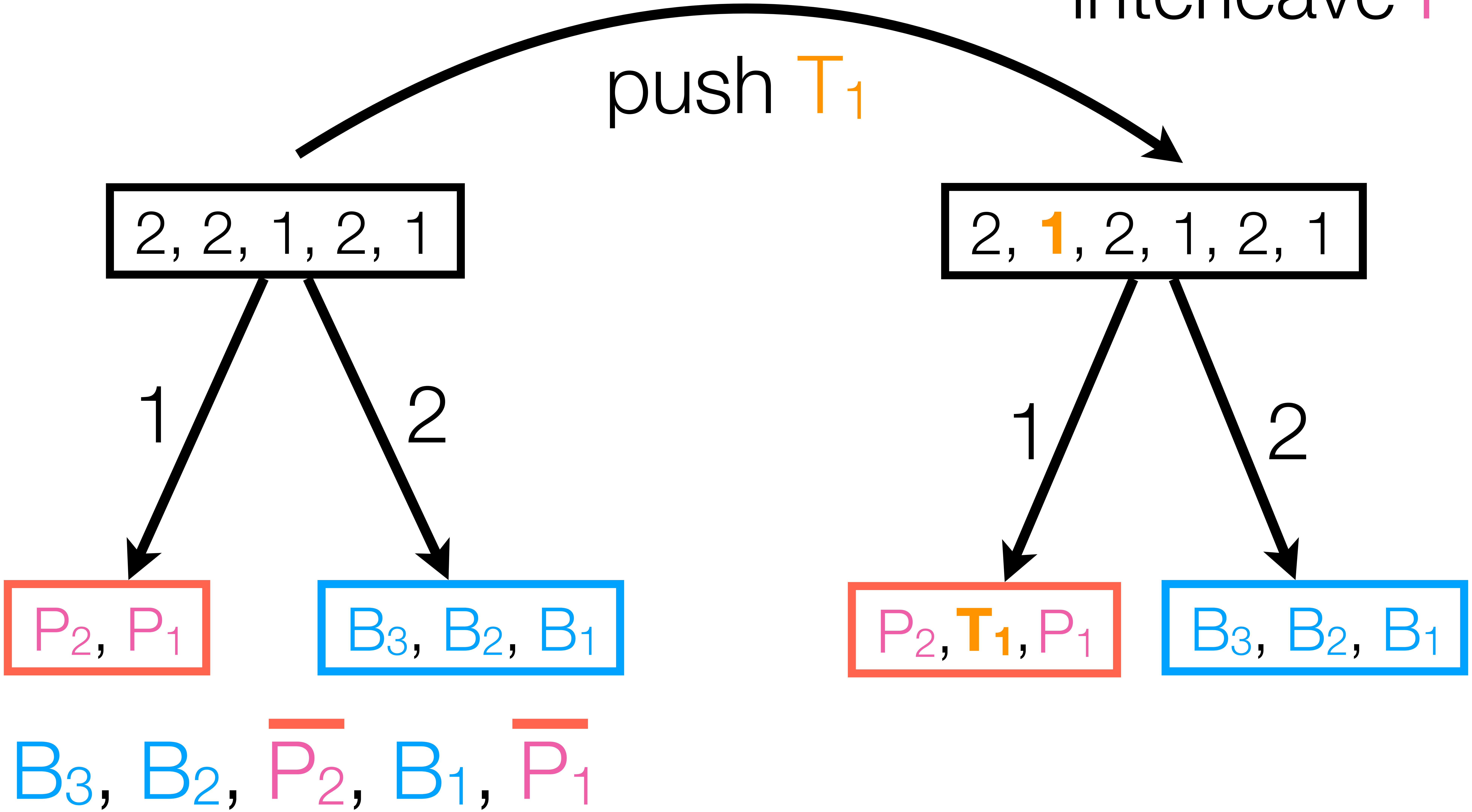
Introducing: PIFO trees

Interleave R and B;
interleave P and T.



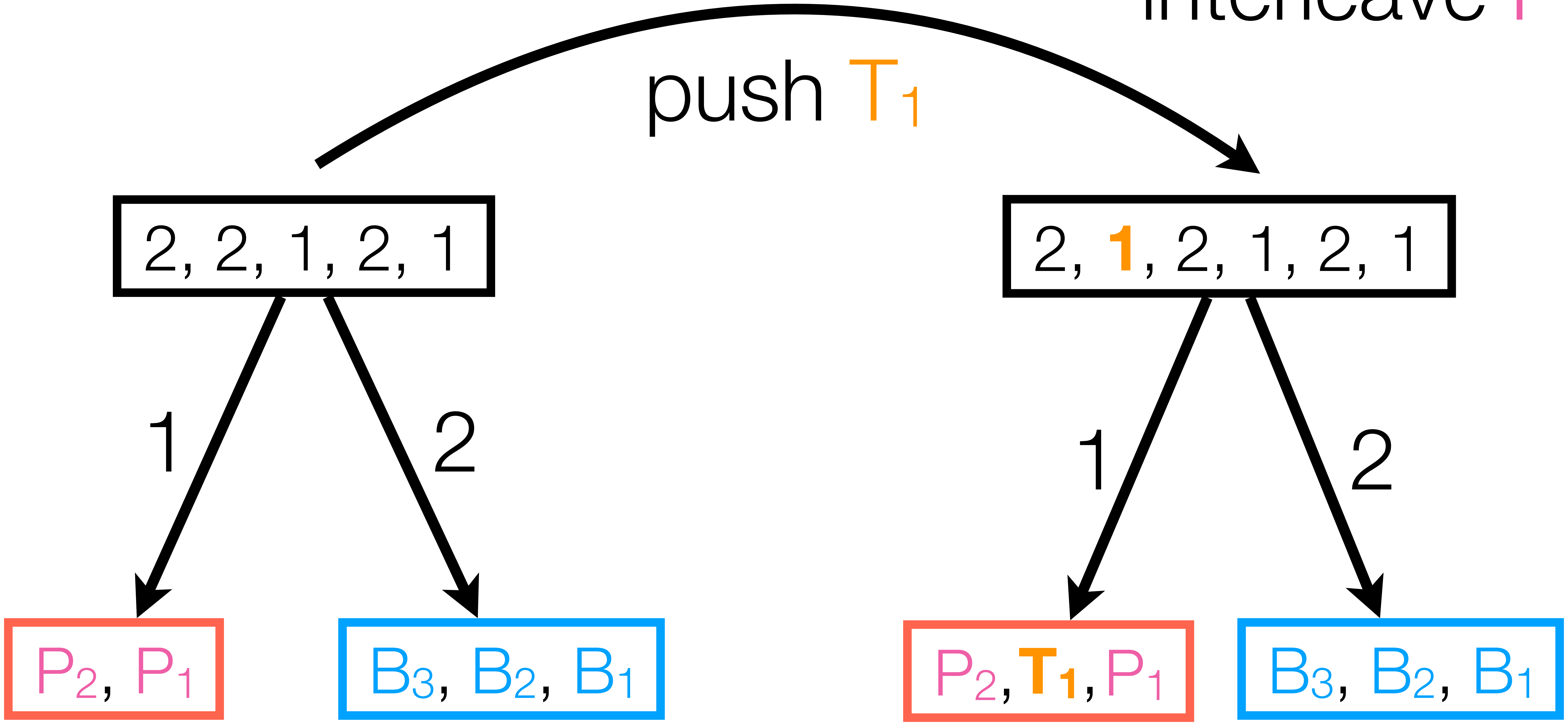
Introducing: PIFO trees

Interleave R and B;
interleave P and T.



Introducing: PIFO trees

Interleave R and B;
interleave P and T.

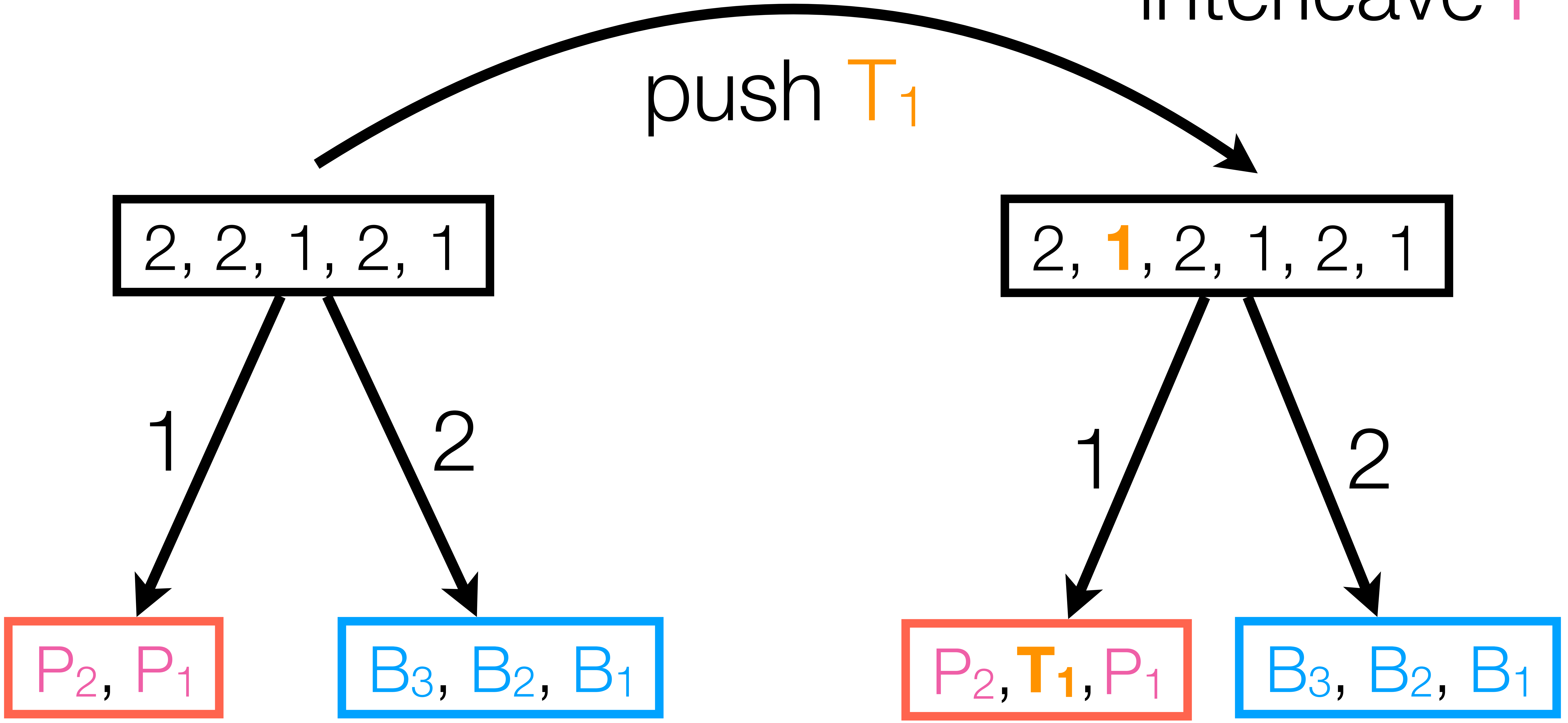


B_3, B_2, $\overline{P_2}$, B_1, $\overline{P_1}$

B_3, P_2, B_2, $\mathbf{T_1}$, B_1, P_1

Introducing: PIFO trees

Interleave R and B;
interleave P and T.

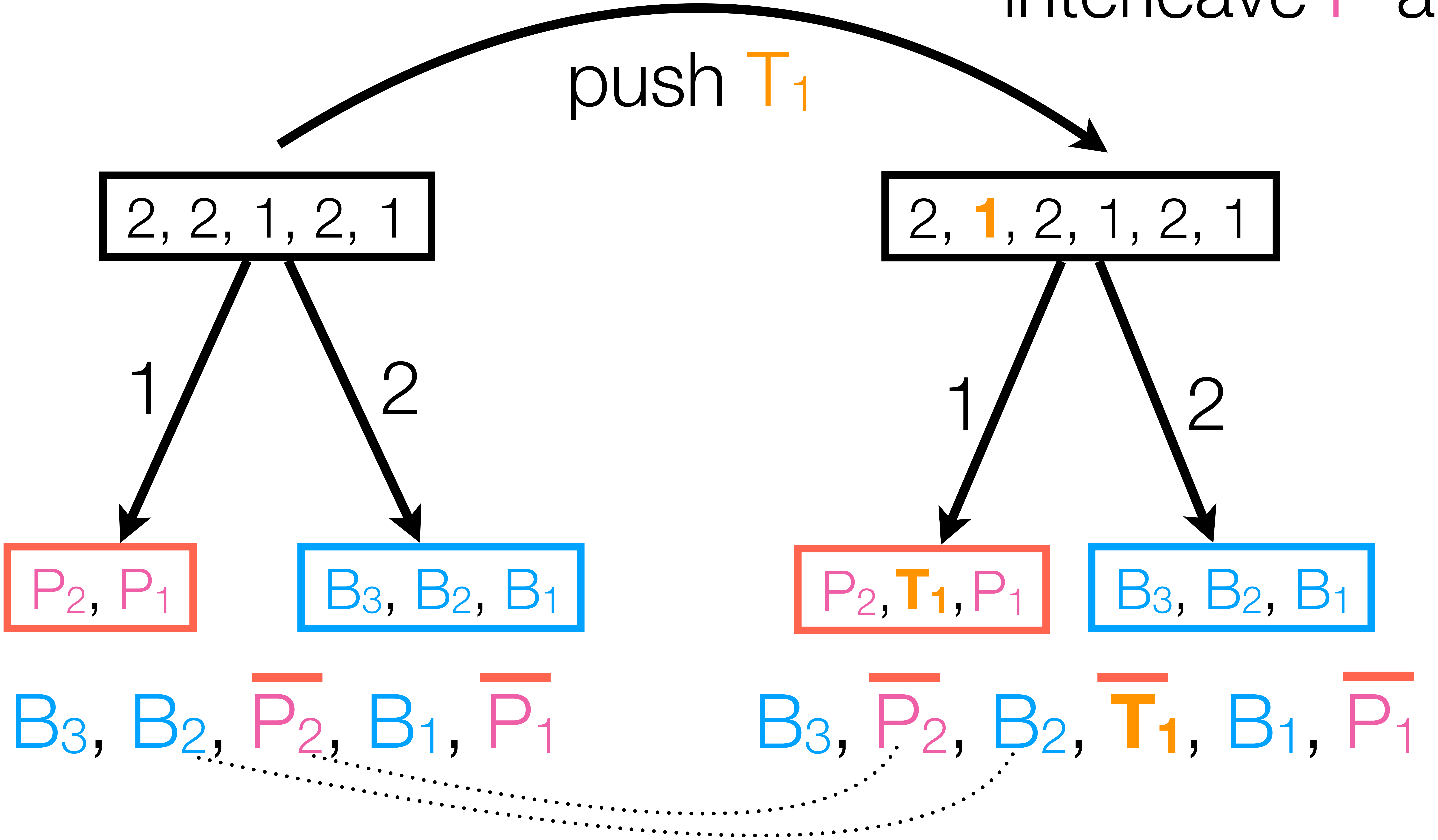


$B_3, B_2, \overline{P_2}, B_1, \overline{P_1}$

$B_3, \overline{P_2}, B_2, \overline{\mathbf{T_1}}, B_1, \overline{P_1}$

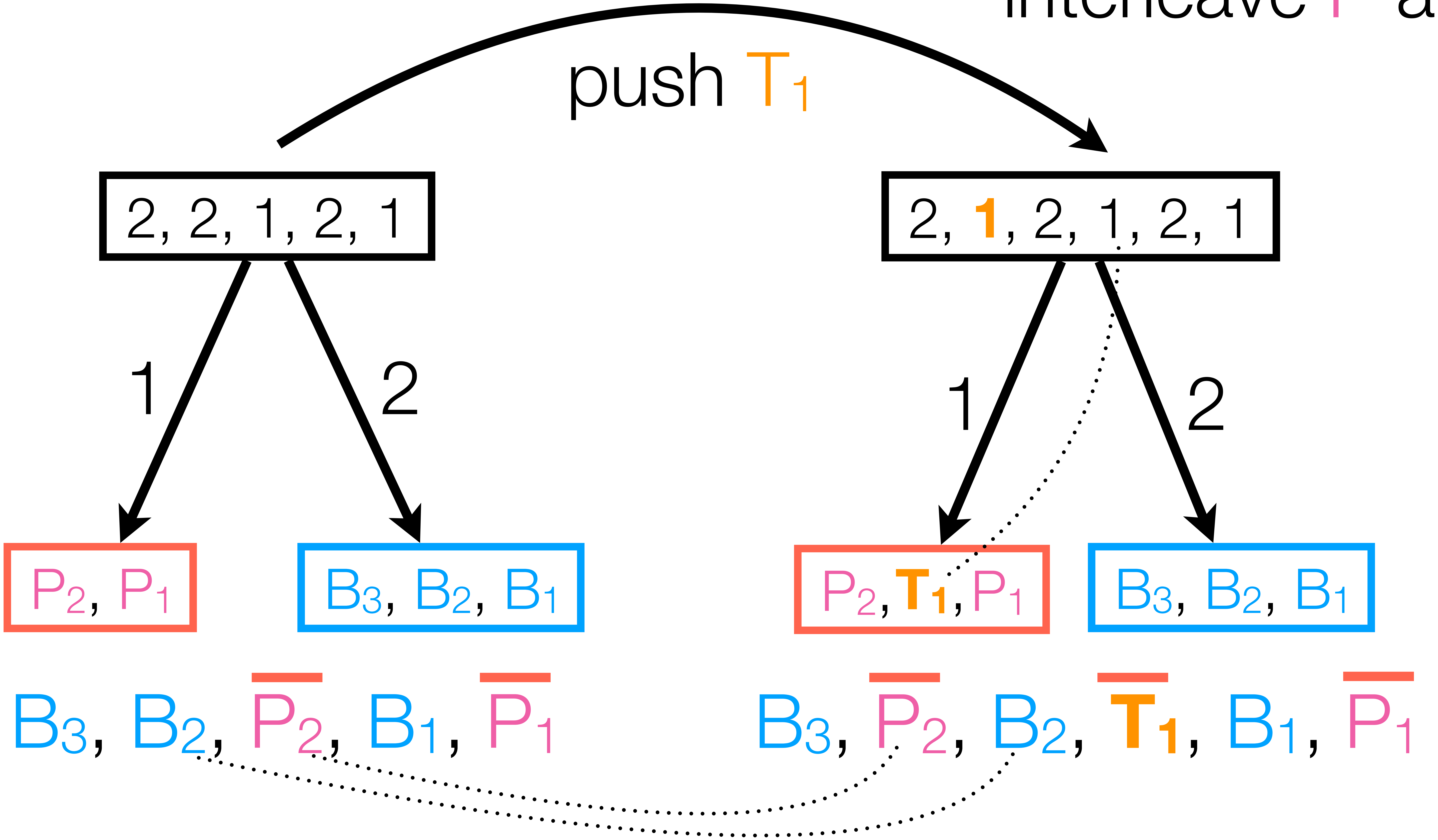
Introducing: PIFO trees

Interleave R and B;
interleave P and T.



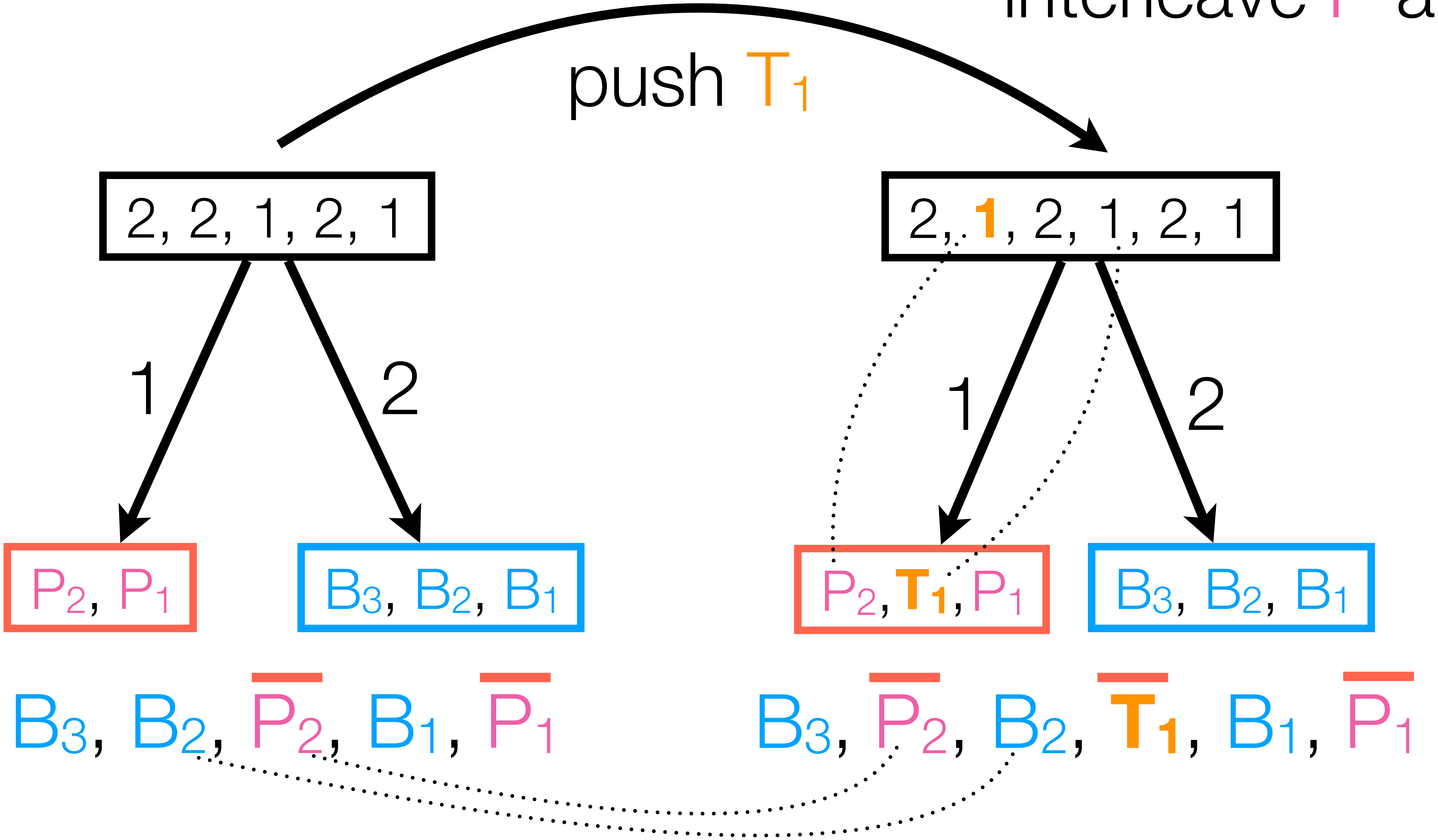
Introducing: PIFO trees

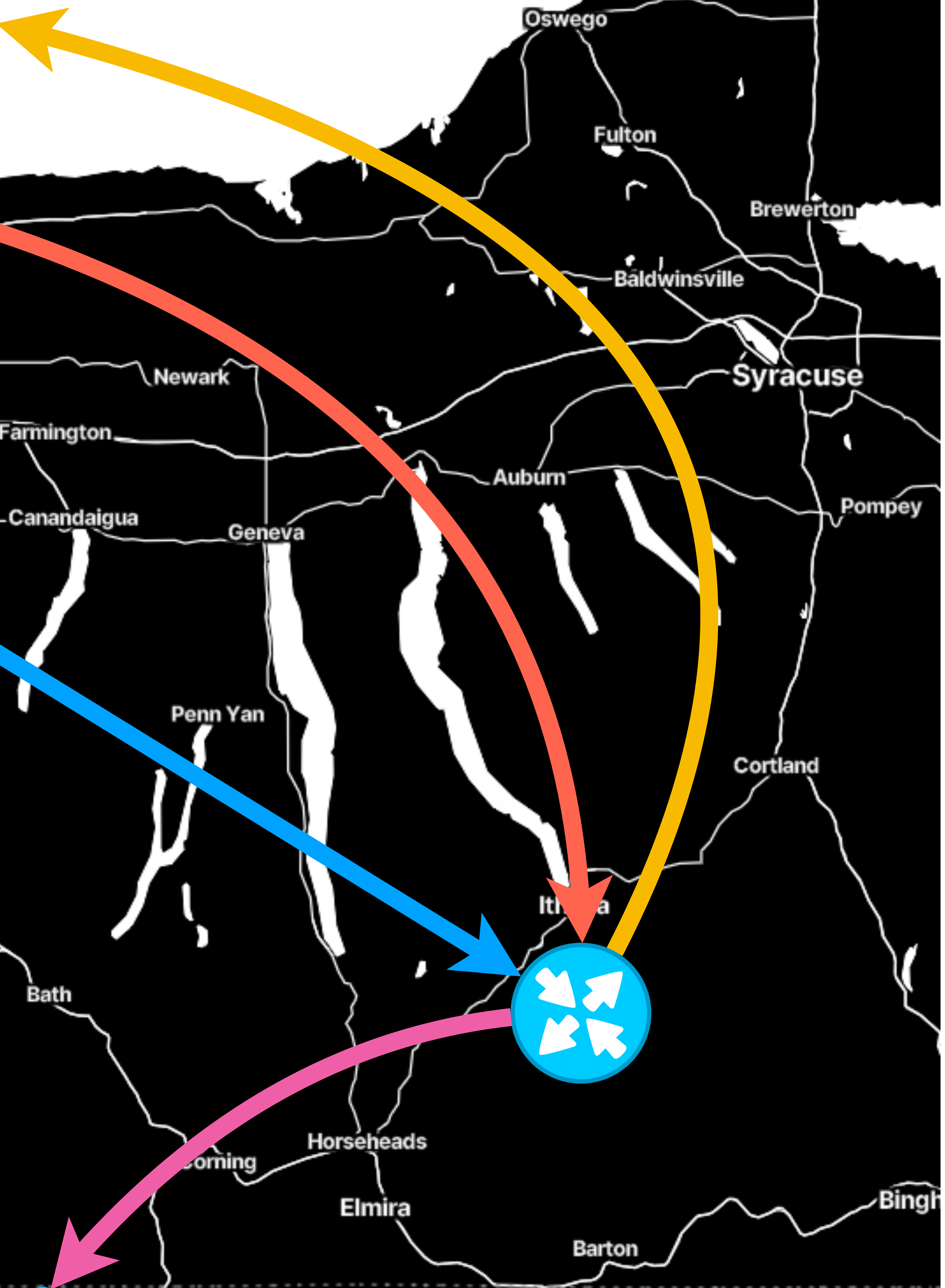
Interleave R and B;
interleave P and T.



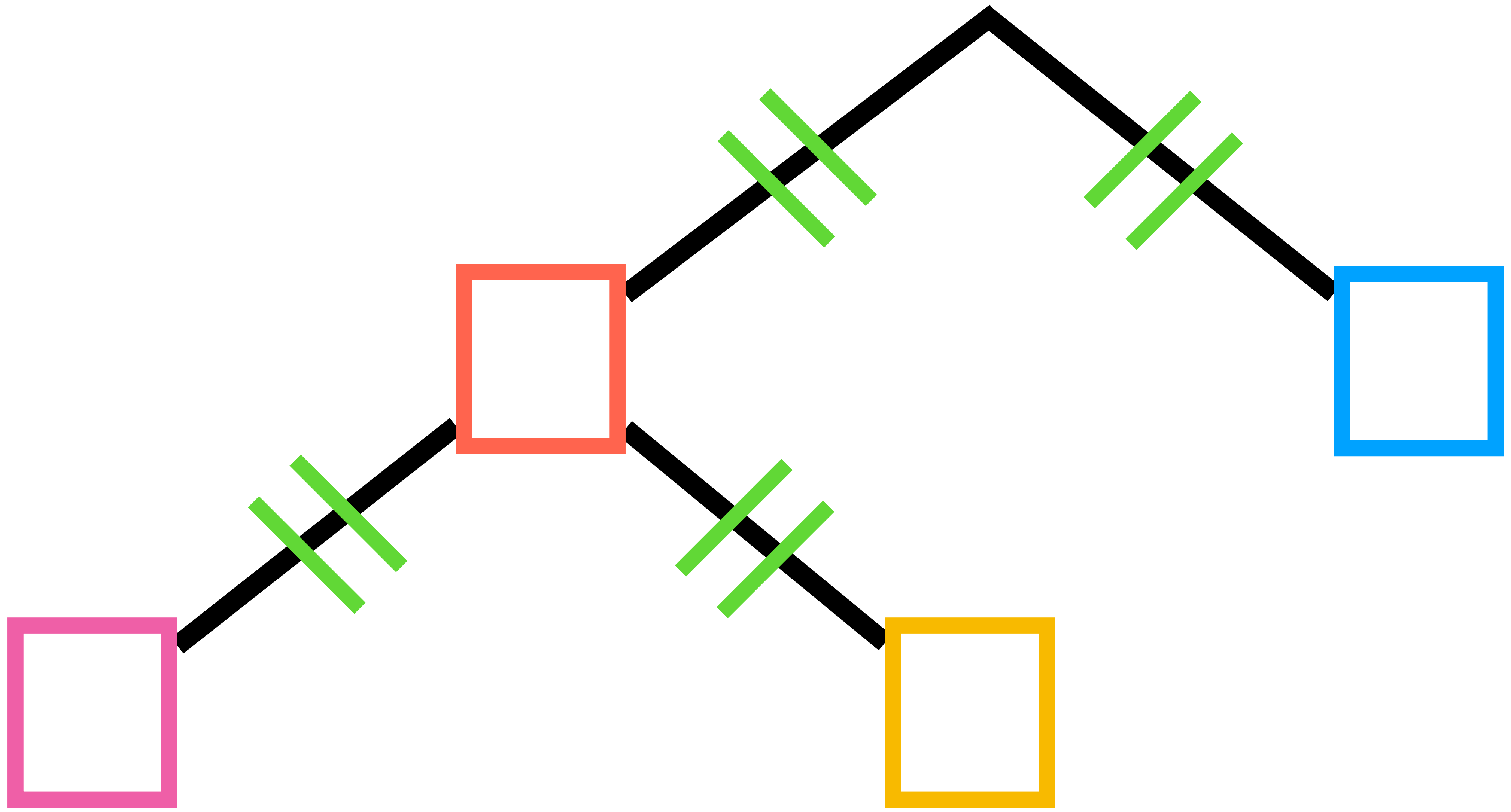
Introducing: PIFO trees

Interleave R and B;
interleave P and T.

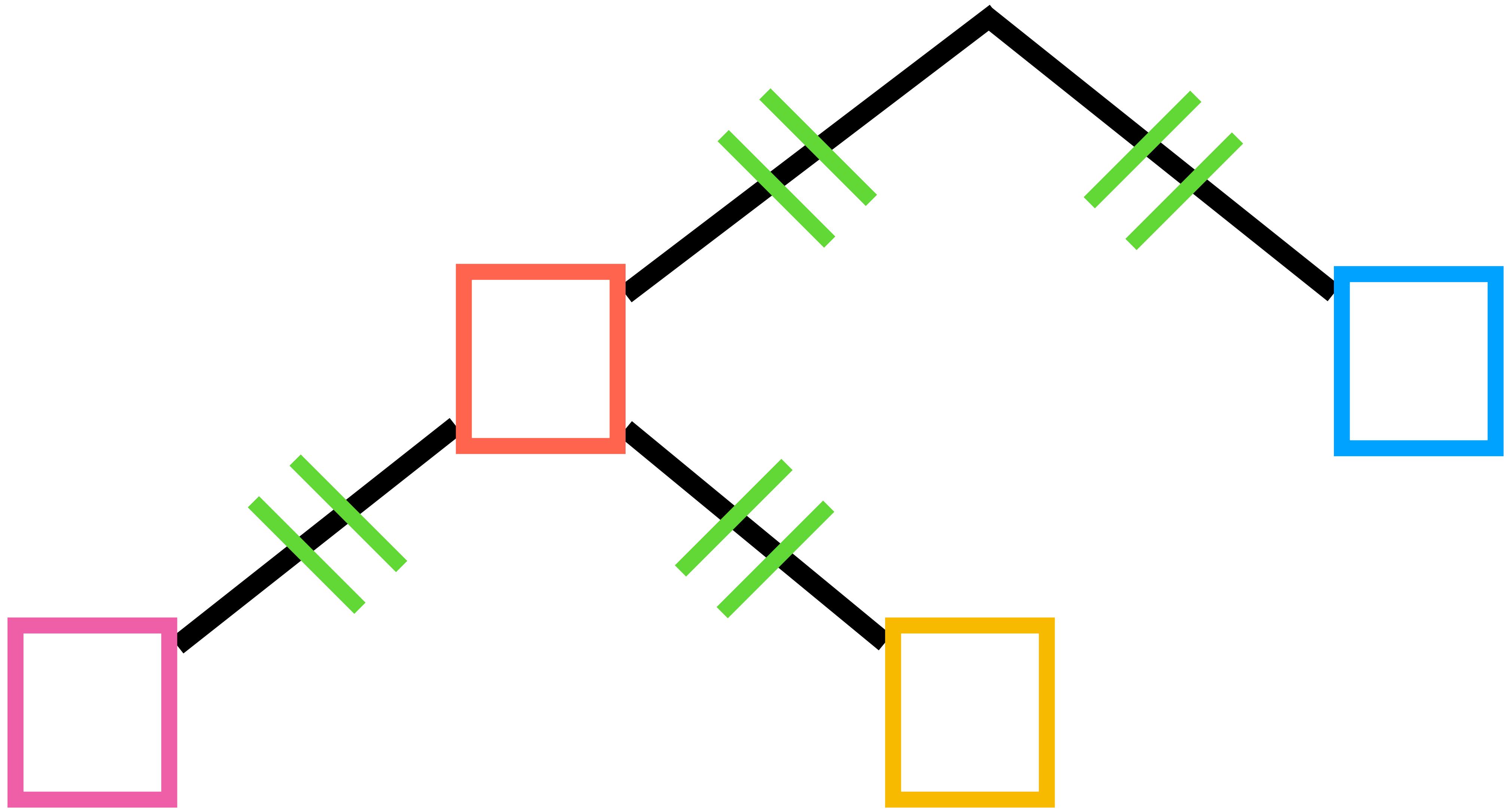




Goal:
interleave R and B;
interleave P and T.



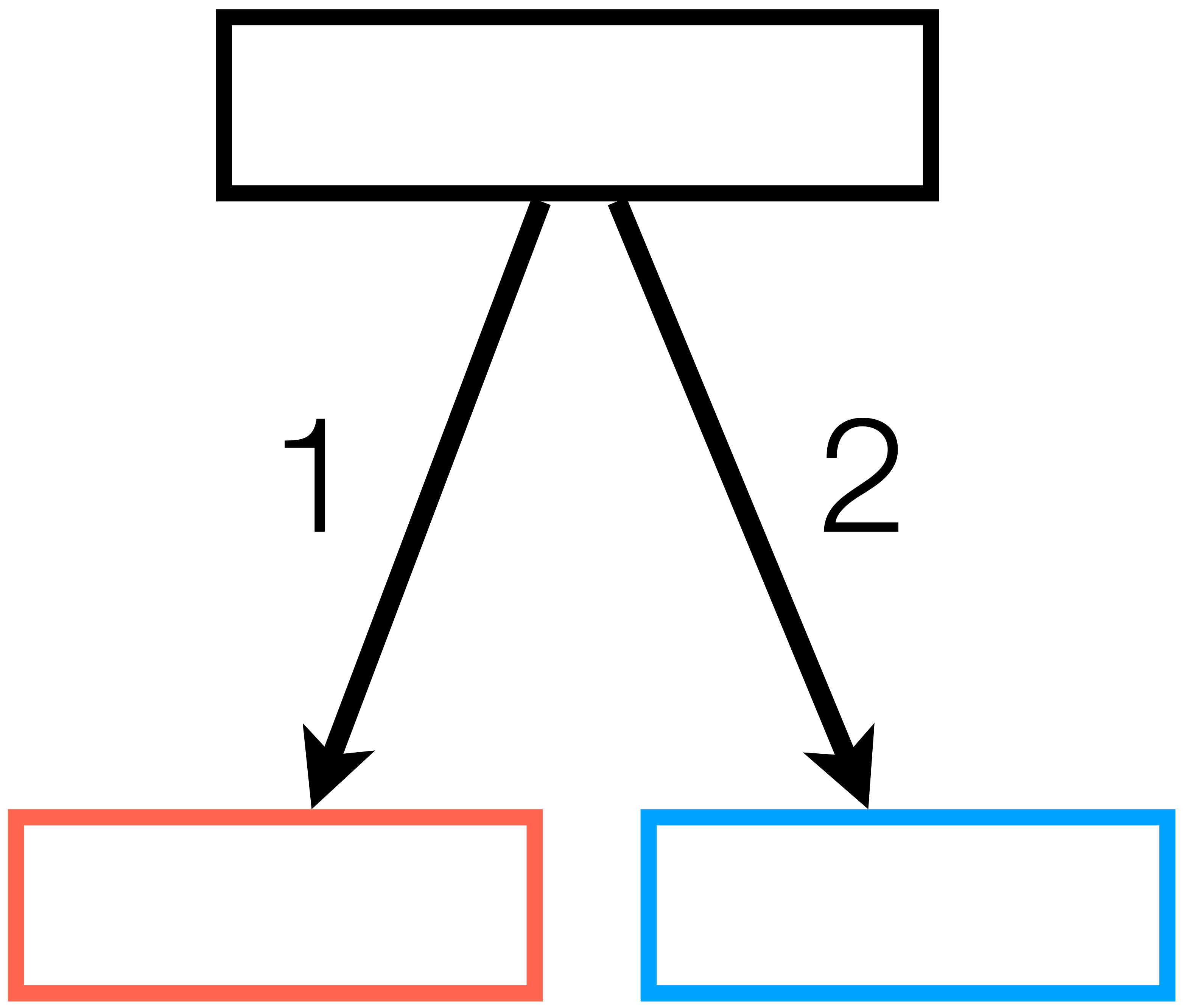
Goal:
interleave R and B;
interleave P and T.



Aside: PIFO Trees

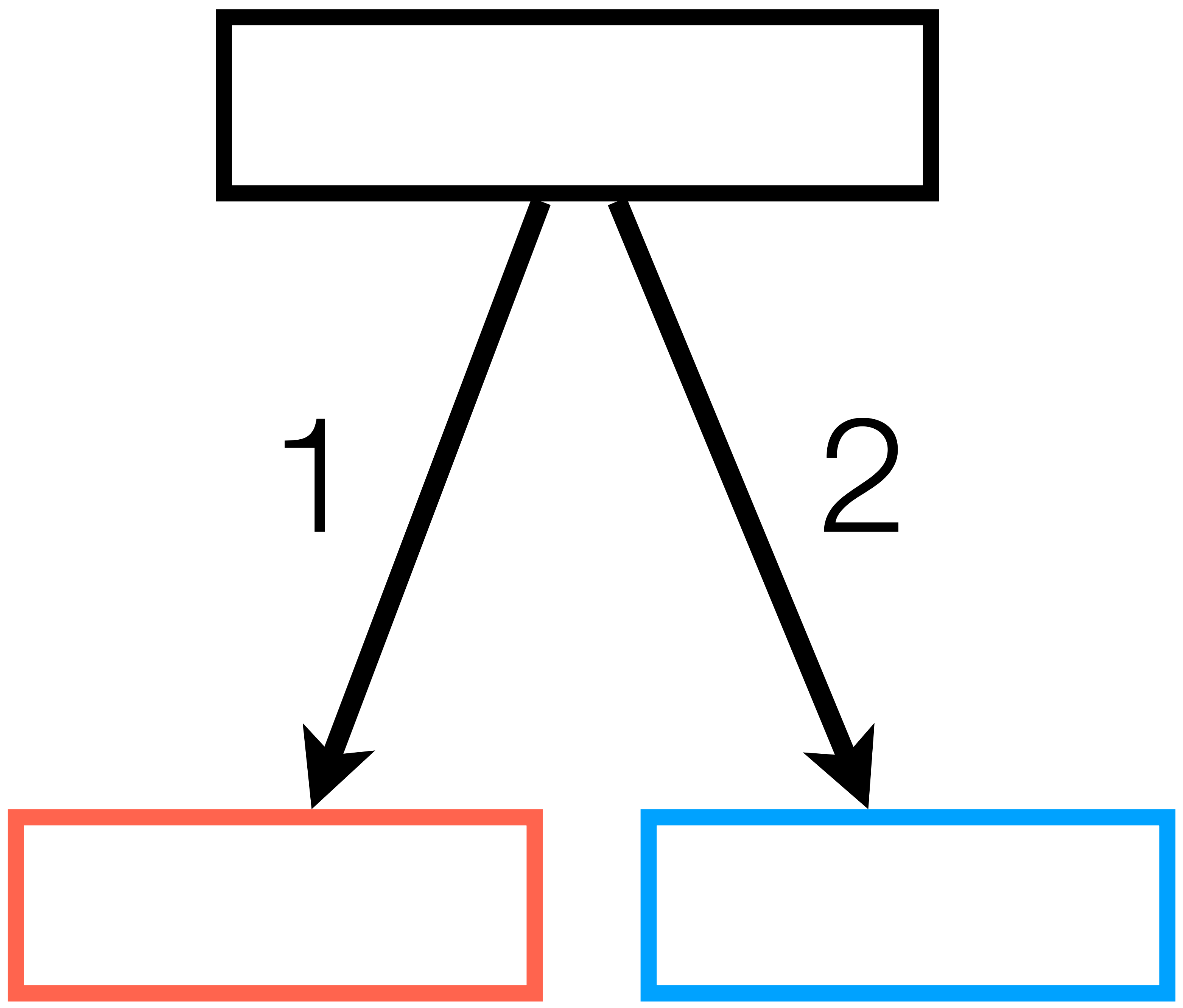
Sivaraman et al. at SIGCOMM ’16

Key Insight



Key Insight

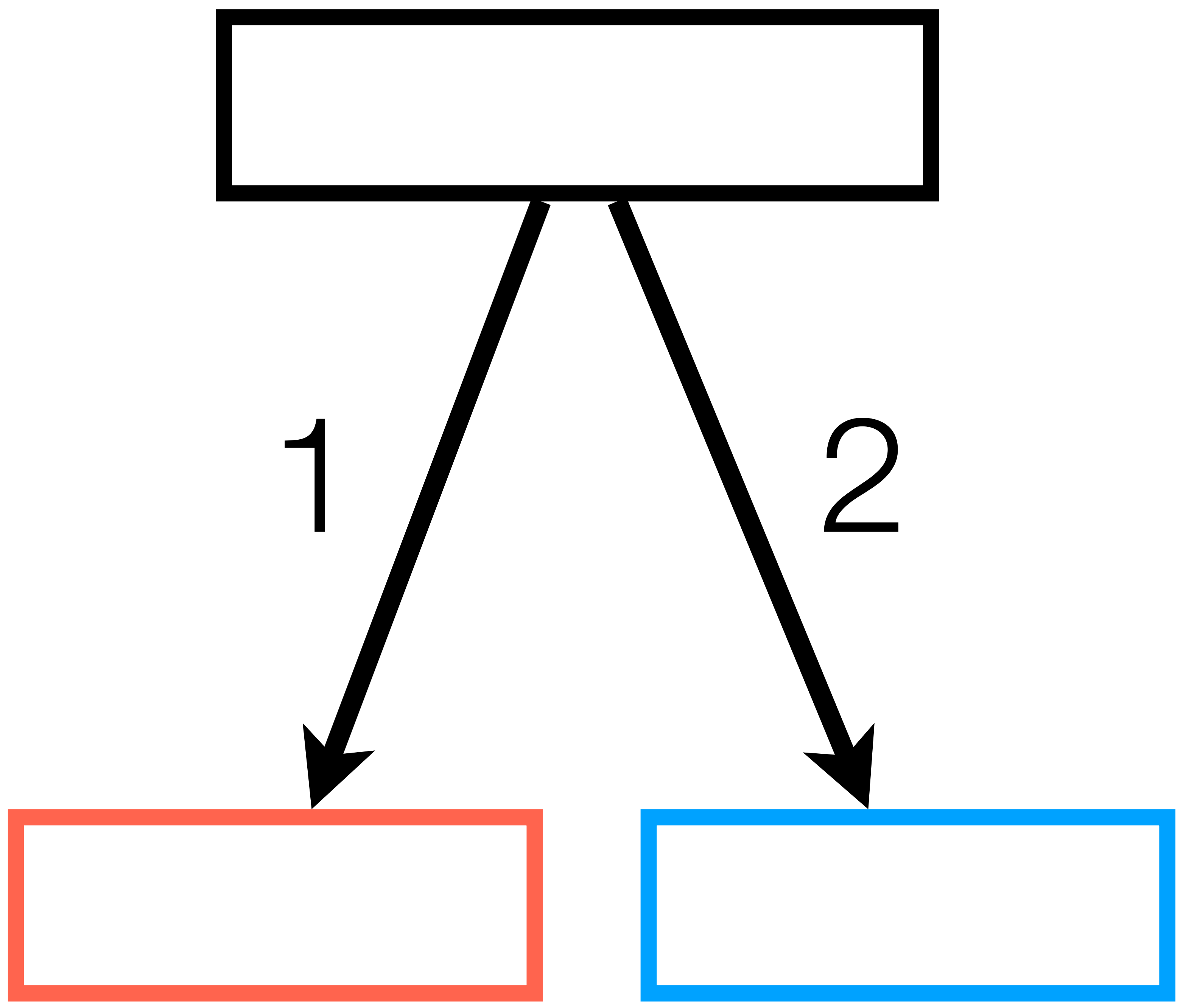
A PIFO tree manifests a *programming language.*



Key Insight

A PIFO tree manifests a *programming language.*

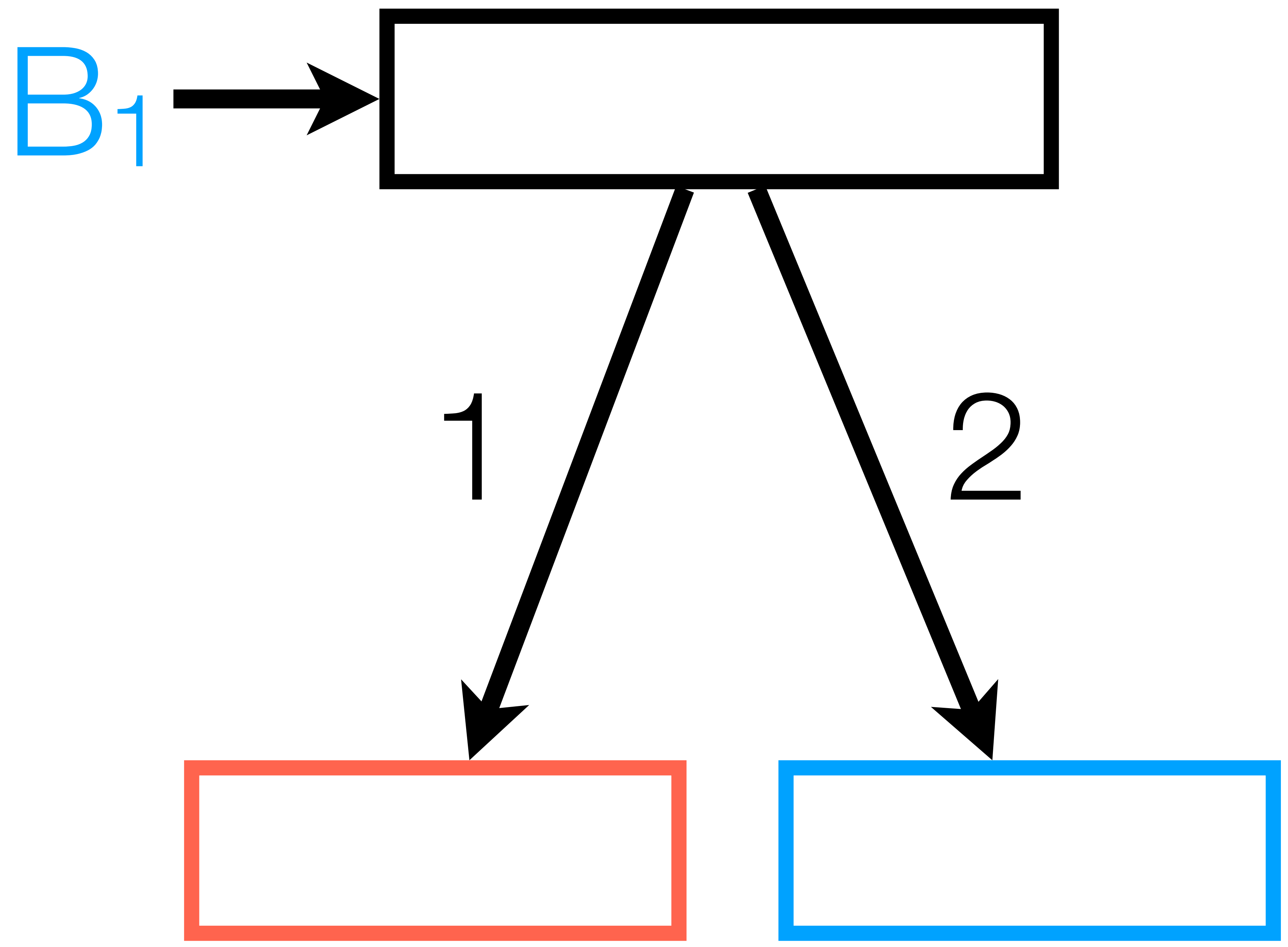
A program is precisely a *scheduling algorithm.*



Key Insight

A PIFO tree manifests a *programming language.*

A program is precisely a *scheduling algorithm.*

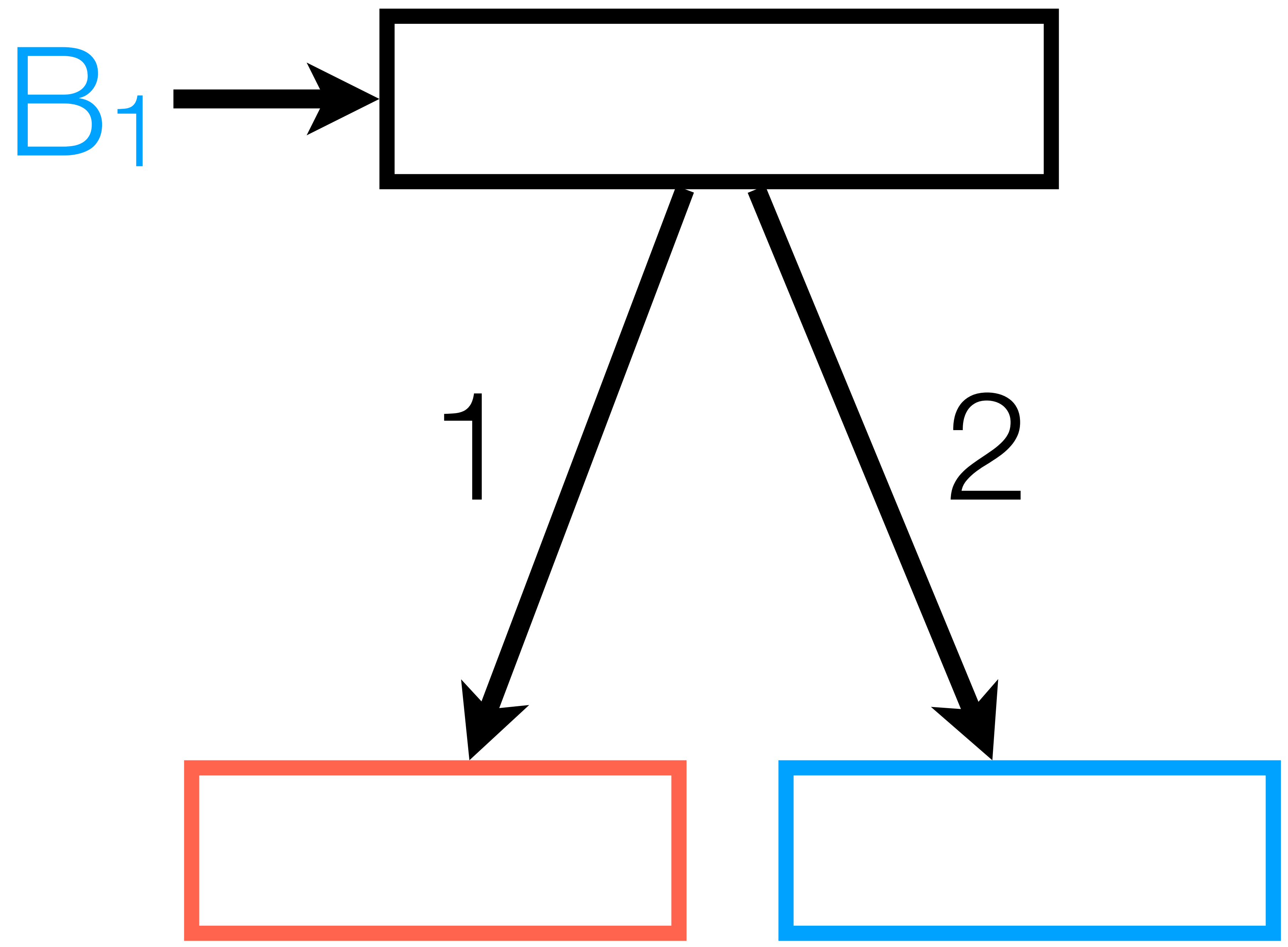


Key Insight

A PIFO tree manifests a *programming language.*

A program is precisely a *scheduling algorithm.*

Path: $[(2, r_1), (B_1, r_2)]$

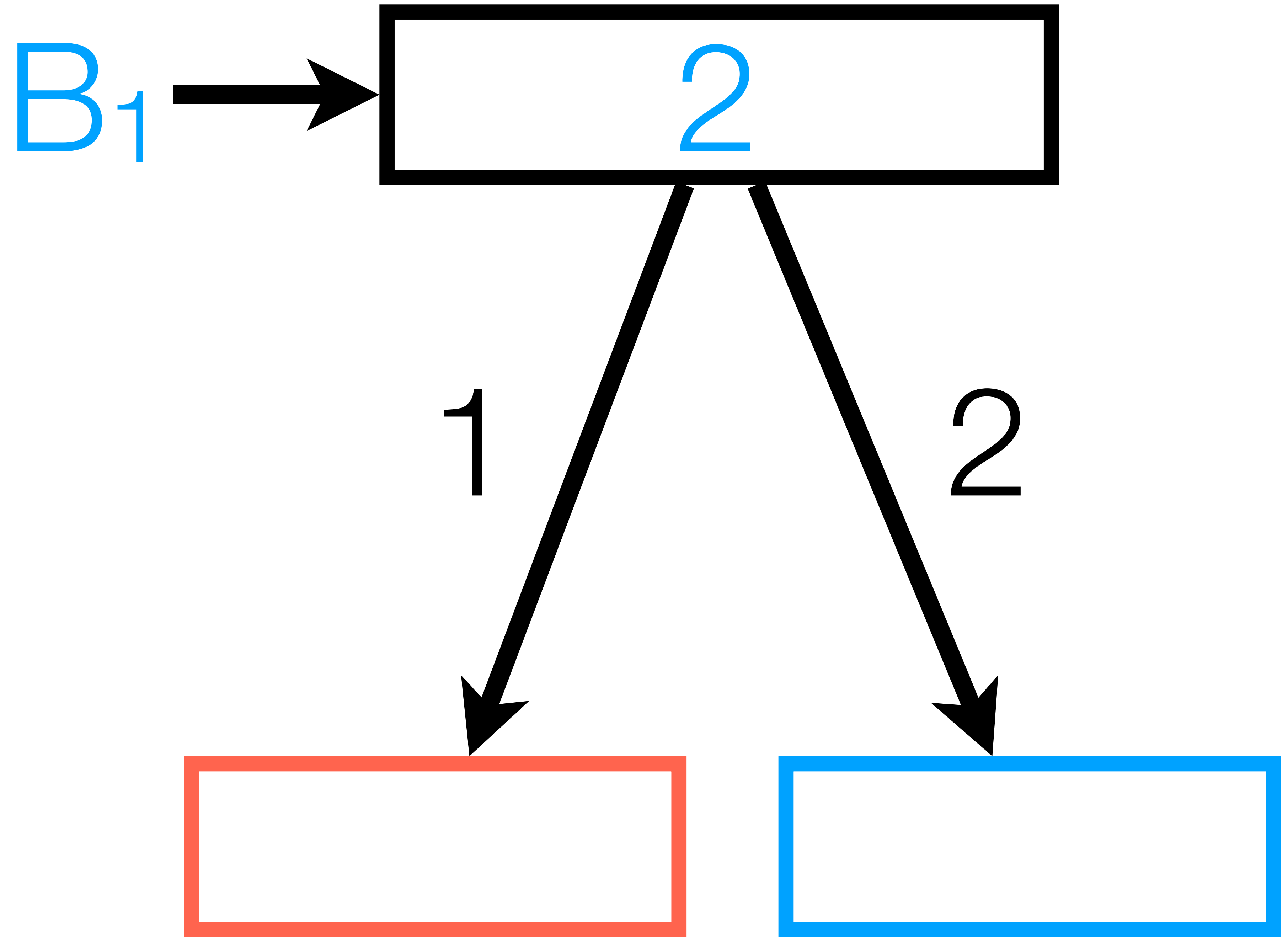


Key Insight

A PIFO tree manifests a *programming language.*

A program is precisely a *scheduling algorithm.*

Path: $[(2, r_1), (B_1, r_2)]$

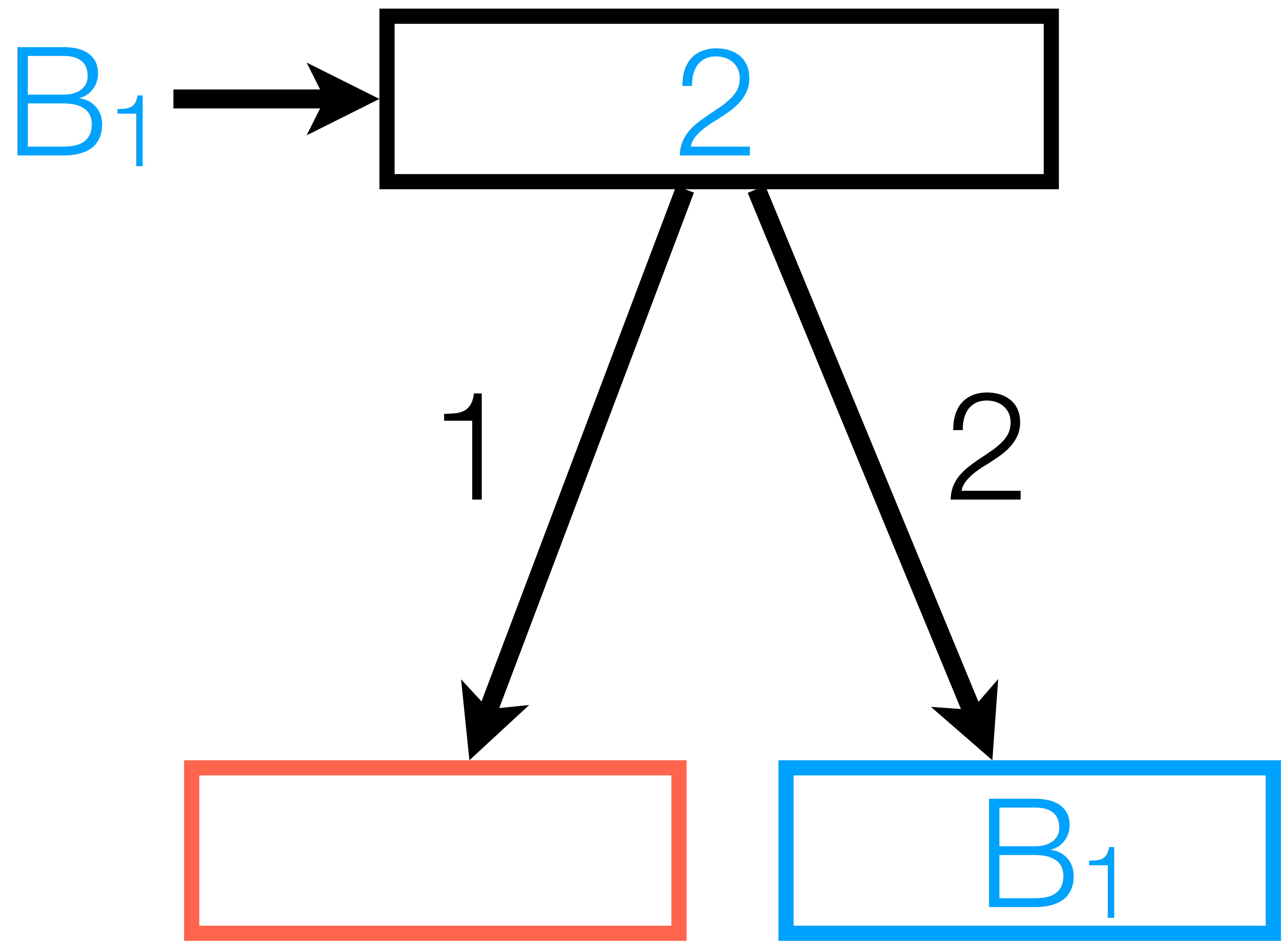


Key Insight

A PIFO tree manifests a *programming language.*

A program is precisely a *scheduling algorithm.*

Path: $[(2, r_1), (B_1, r_2)]$



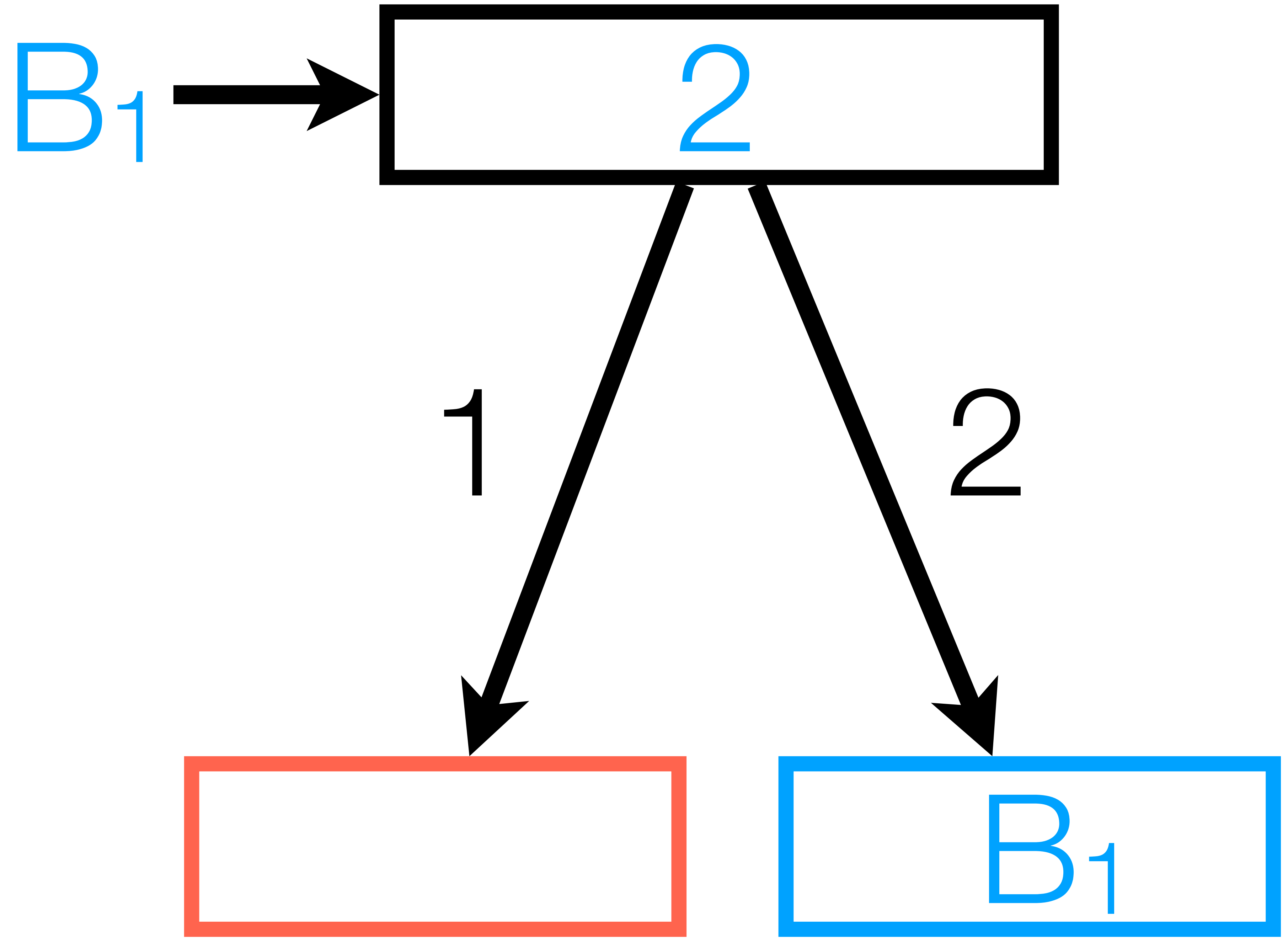
Key Insight

A PIFO tree manifests a *programming language.*

A program is precisely a *scheduling algorithm.*

tree shape ⟷ language expressivity

Path: $[(2, r_1), (B_1, r_2)]$



Which leads to some very PL-ey questions:

tree shape ⟷ language expressivity

Which leads to some very PL-ey questions:

tree shape ⟷ language expressivity

Compare expressivity of languages?

Which leads to some very PL-ey questions:

tree shape $\longleftrightarrow$ language expressivity

Compare expressivity of languages?
Compare expressivity of *trees*?

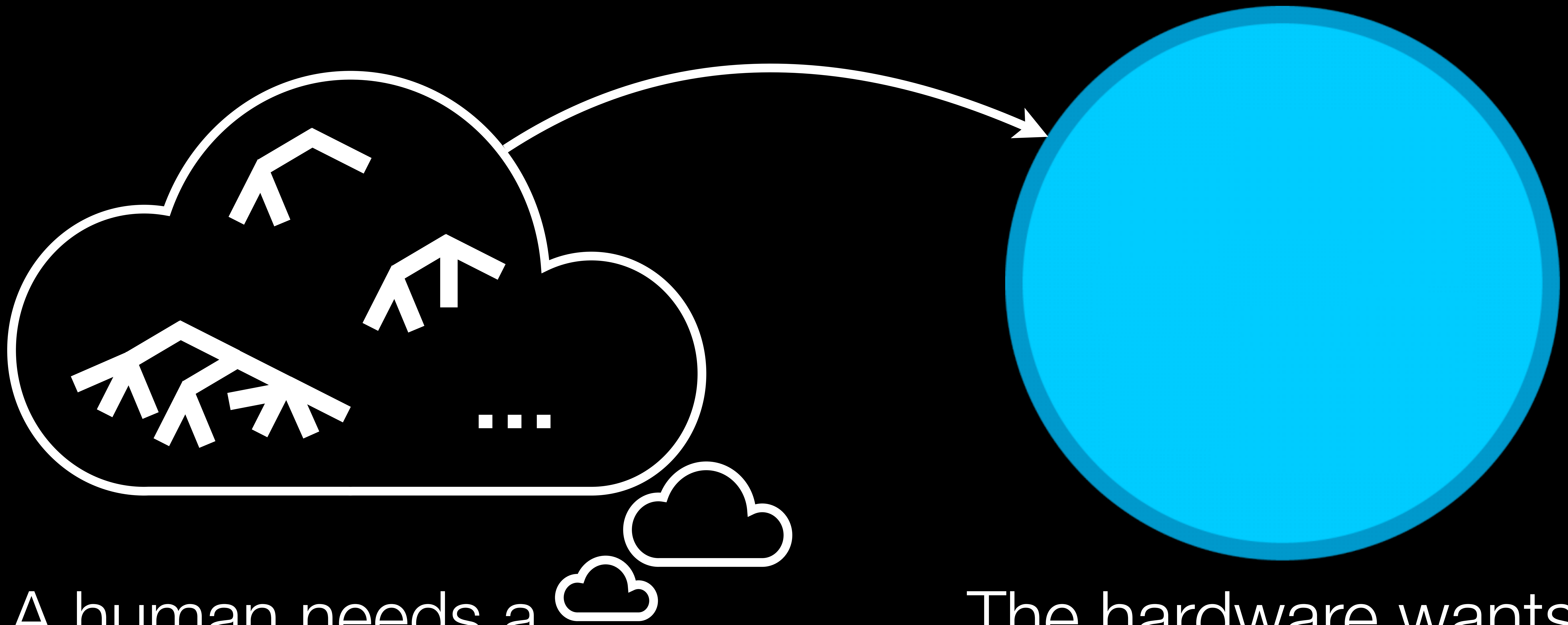
Which leads to some very PL-ey questions:

tree shape $\longleftrightarrow$ language expressivity

Compare expressivity of languages?
Compare expressivity of *trees*?

Compile a program so it runs against a new tree?

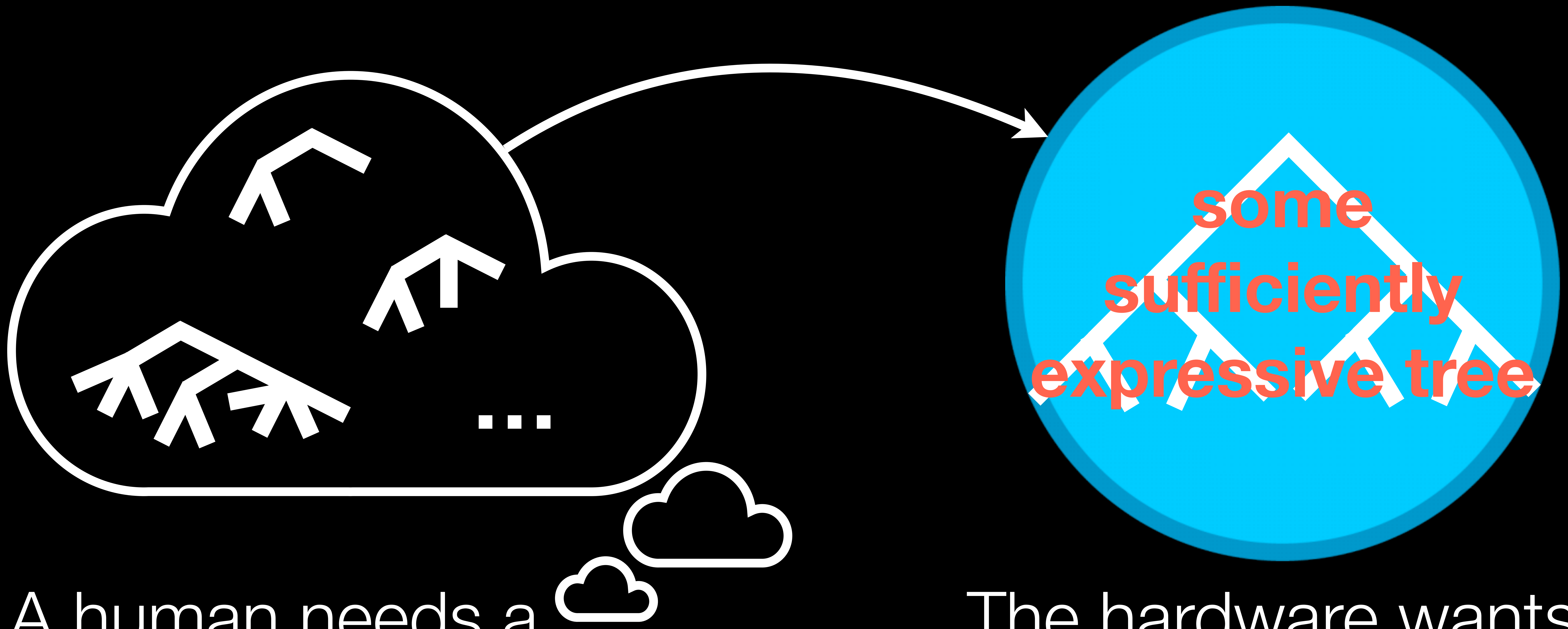
~~No general way to deploy our gadget.~~



A human needs a *range* of trees.

The hardware wants to support *one* tree.

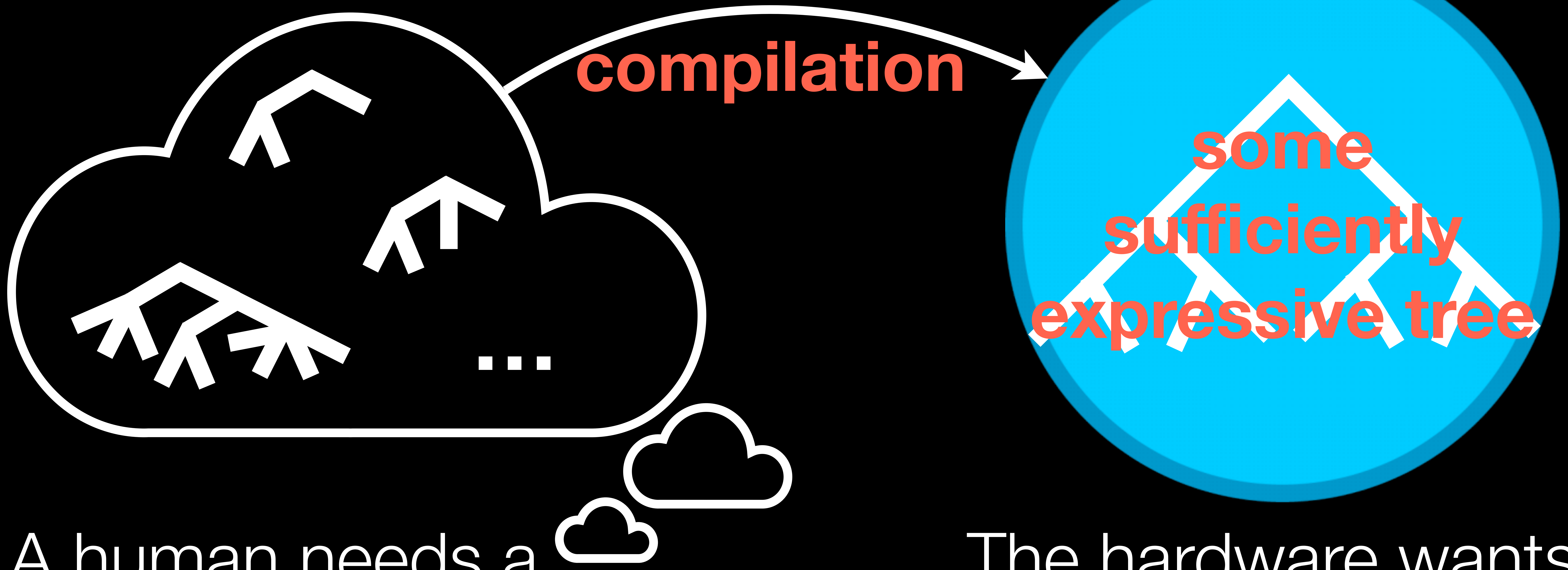
~~No general way to deploy our gadget.~~



A human needs a *range* of trees.

The hardware wants to support *one* tree.

~~No general way to deploy our gadget.~~



A human needs a *range* of trees.

The hardware wants to support *one* tree.

Contributions

Contributions

Formal model of PIFO trees

Contributions

Formal model of PIFO trees

General theorems of expressiveness w.r.t. tree shape

Contributions

Formal model of PIFO trees

General theorems of expressiveness w.r.t. tree shape

Compiler

Contributions

Formal model of PIFO trees

General theorems of expressiveness w.r.t. tree shape

Compiler

Simulator

Expressivity of trees

Trees with more leaves are more expressive.

Taller trees are more expressive.

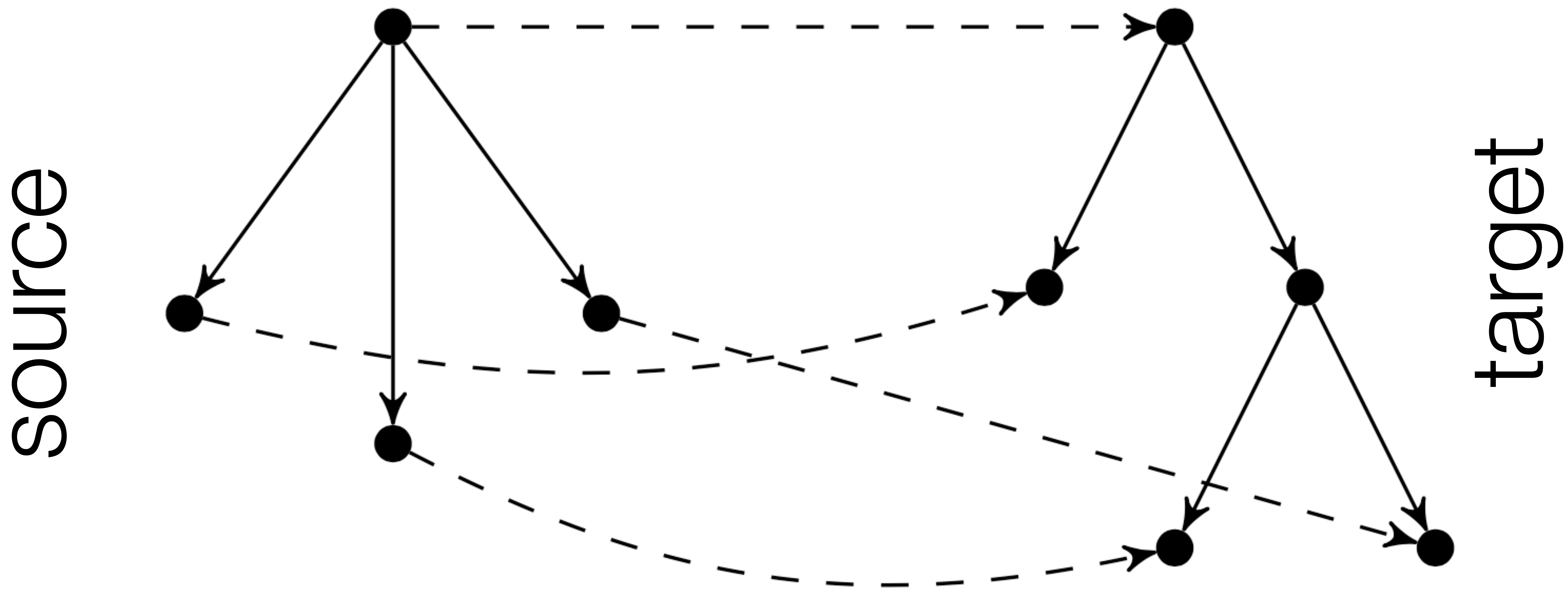
Expressivity of trees

Trees with more leaves are more expressive.
Taller trees are more expressive.

Captured elegantly by:

Homomorphic embedding.
Map root to root, leaves to leaves. Respect ancestry.

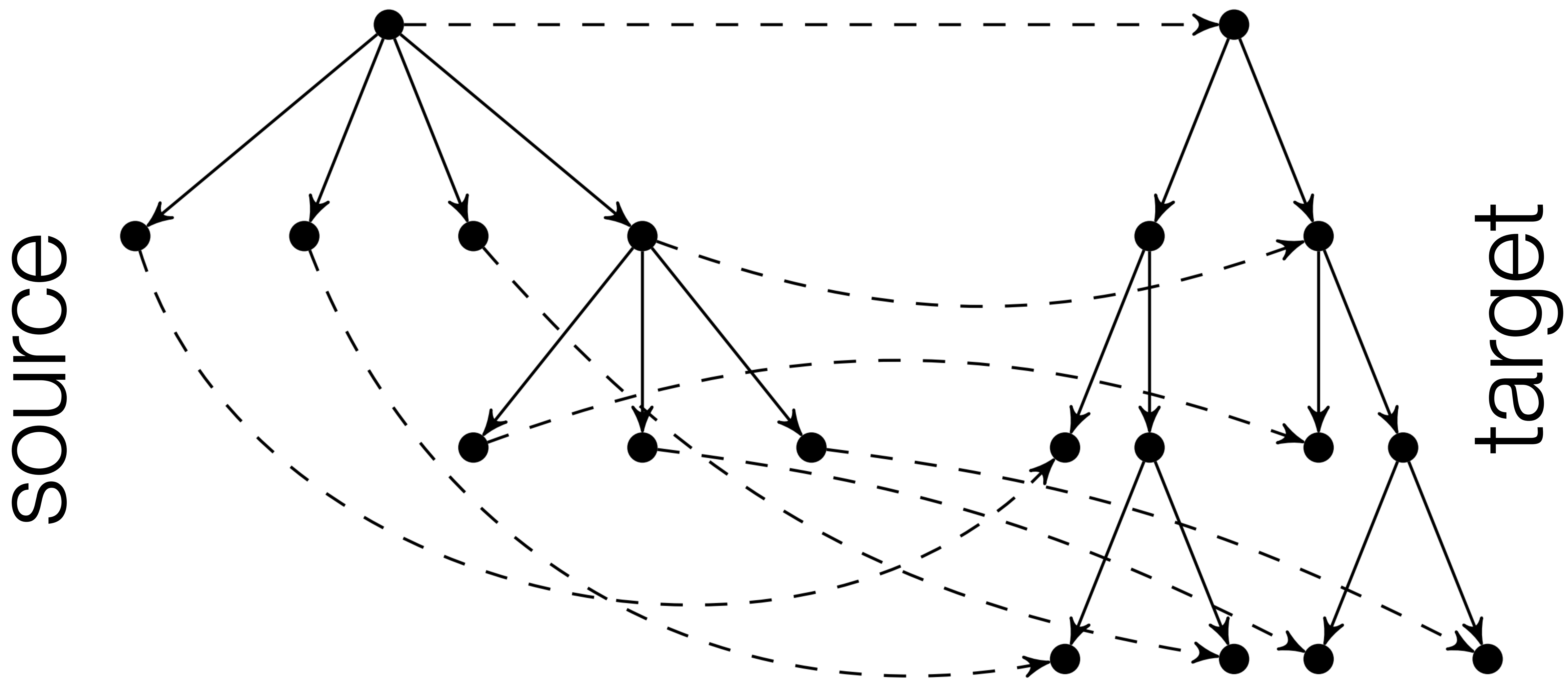
Expressivity of trees



Homomorphic embedding.

Map root to root, leaves to leaves. Respect ancestry.

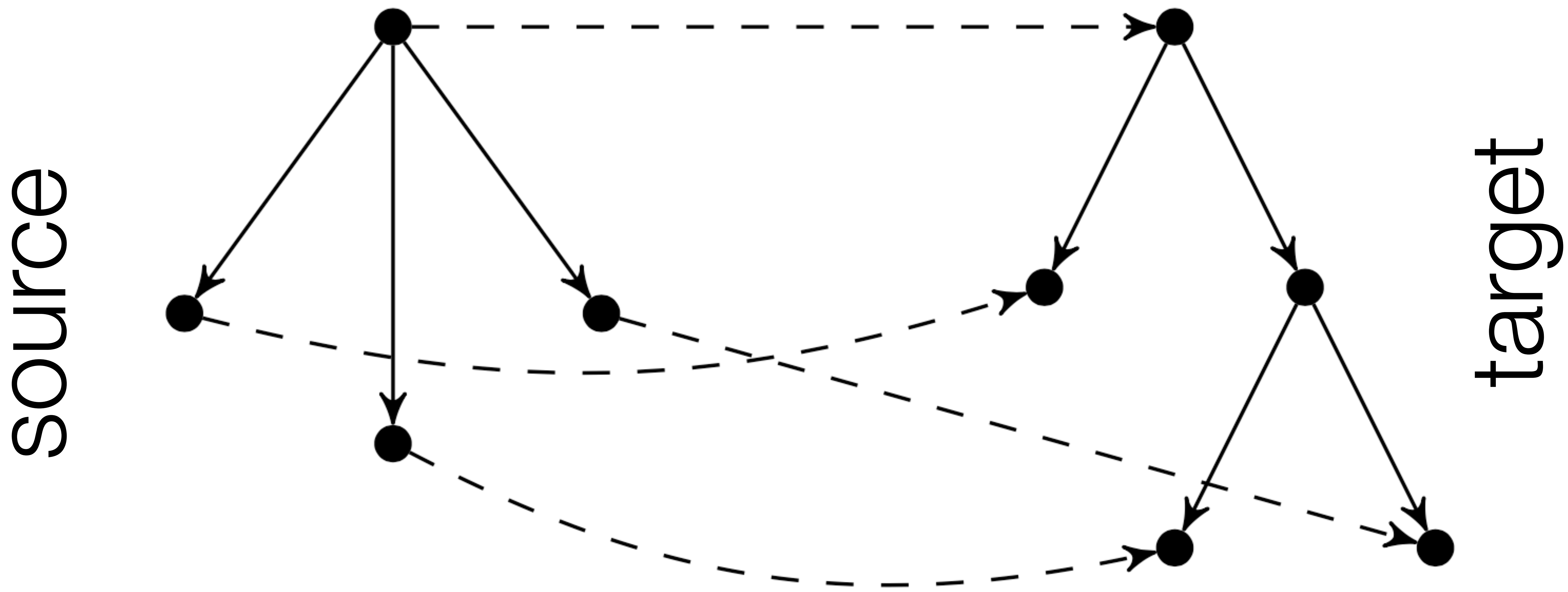
Expressivity of trees



Homomorphic embedding.

Map root to root, leaves to leaves. Respect ancestry.

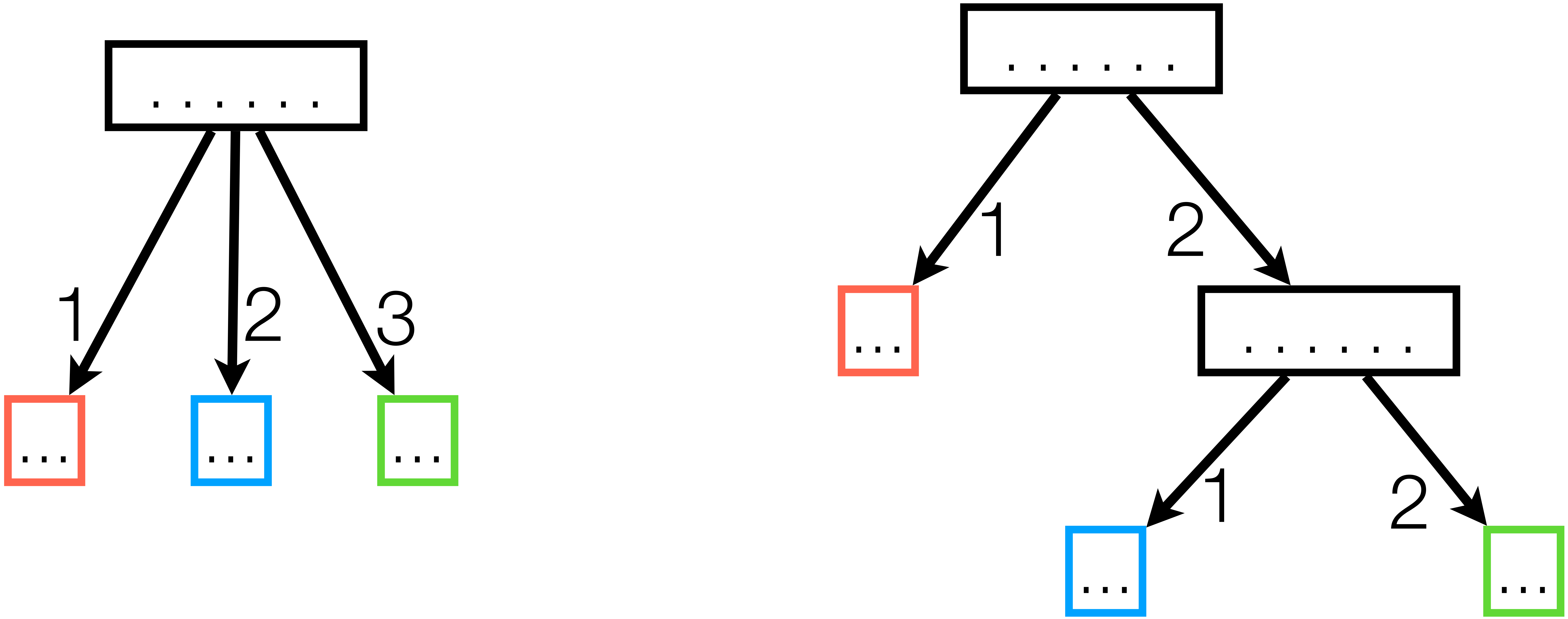
Expressivity of trees



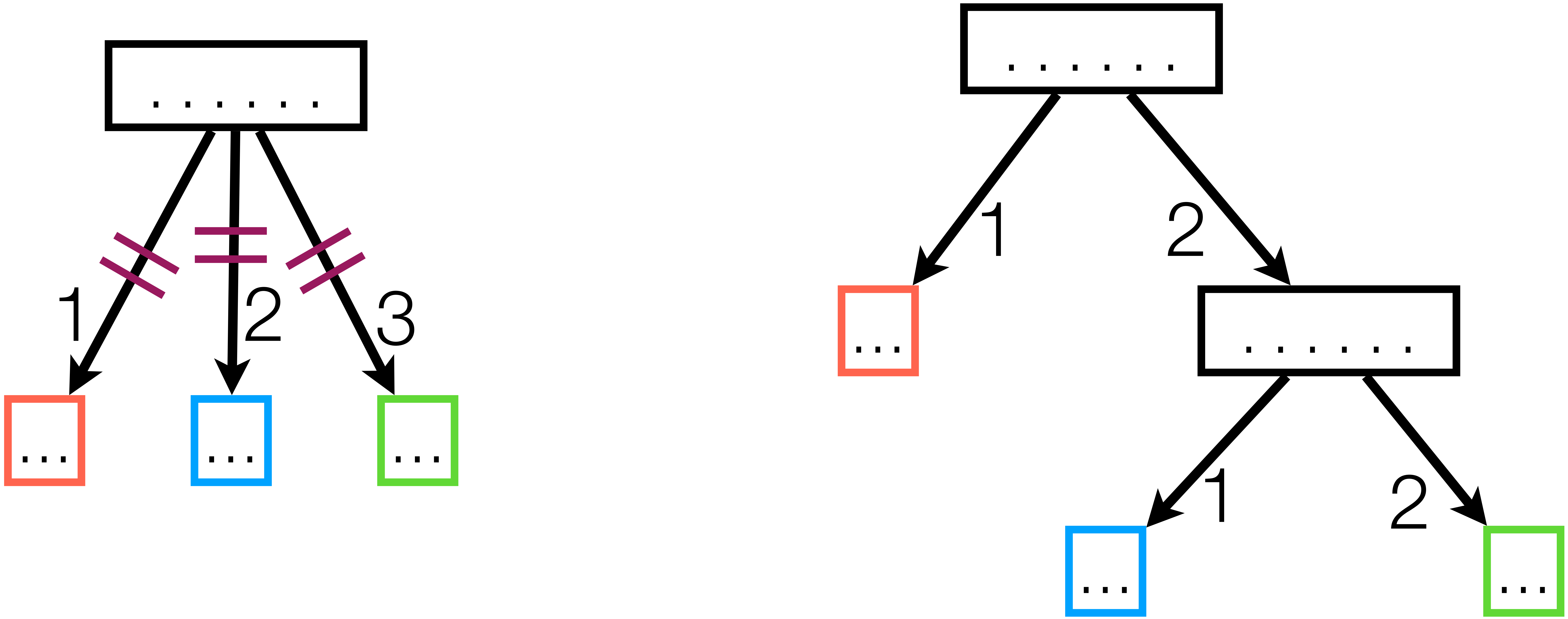
Homomorphic embedding.

Map root to root, leaves to leaves. Respect ancestry.

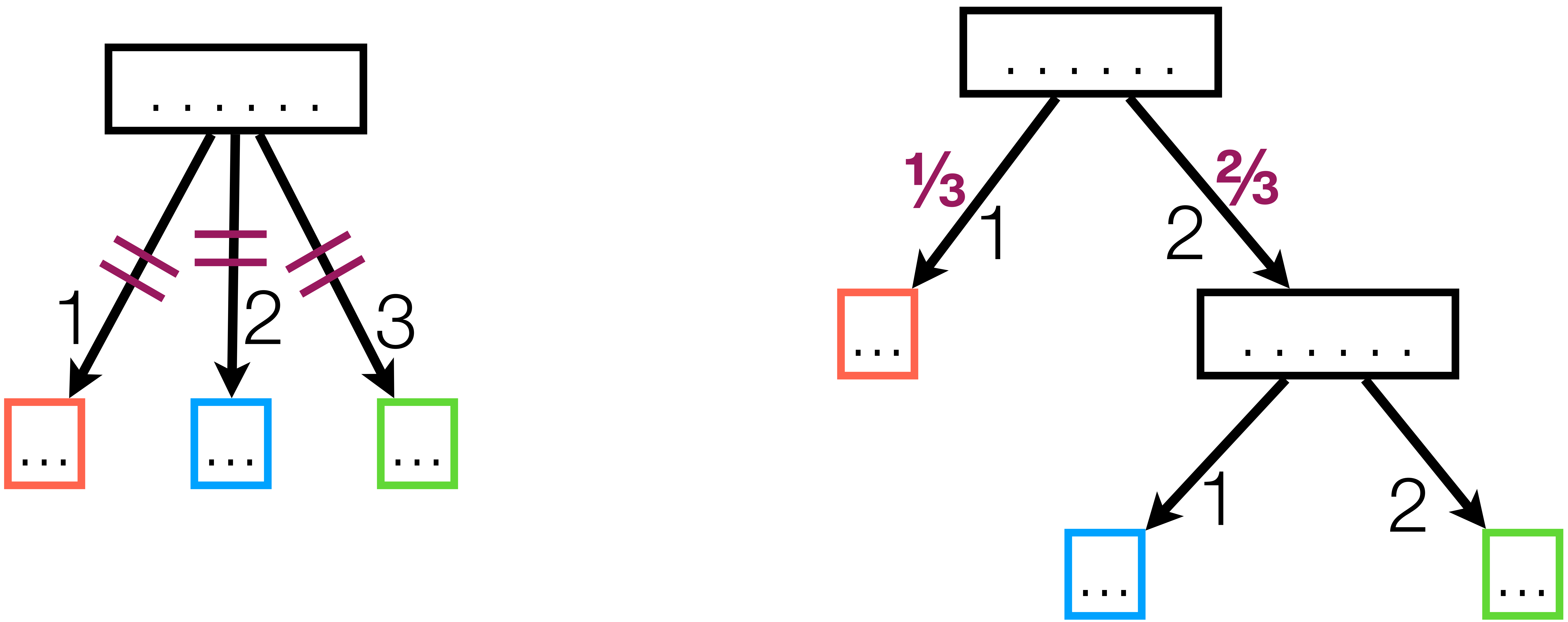
Compiling programs



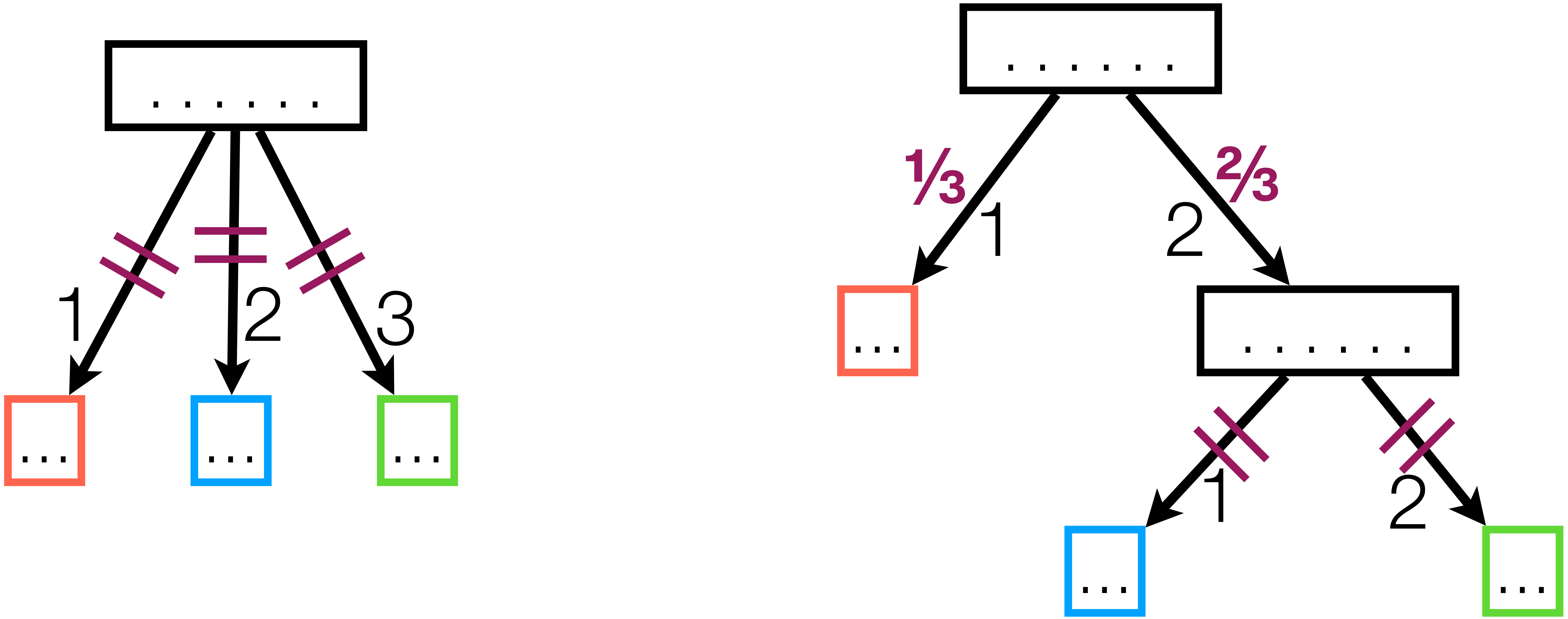
Compiling programs



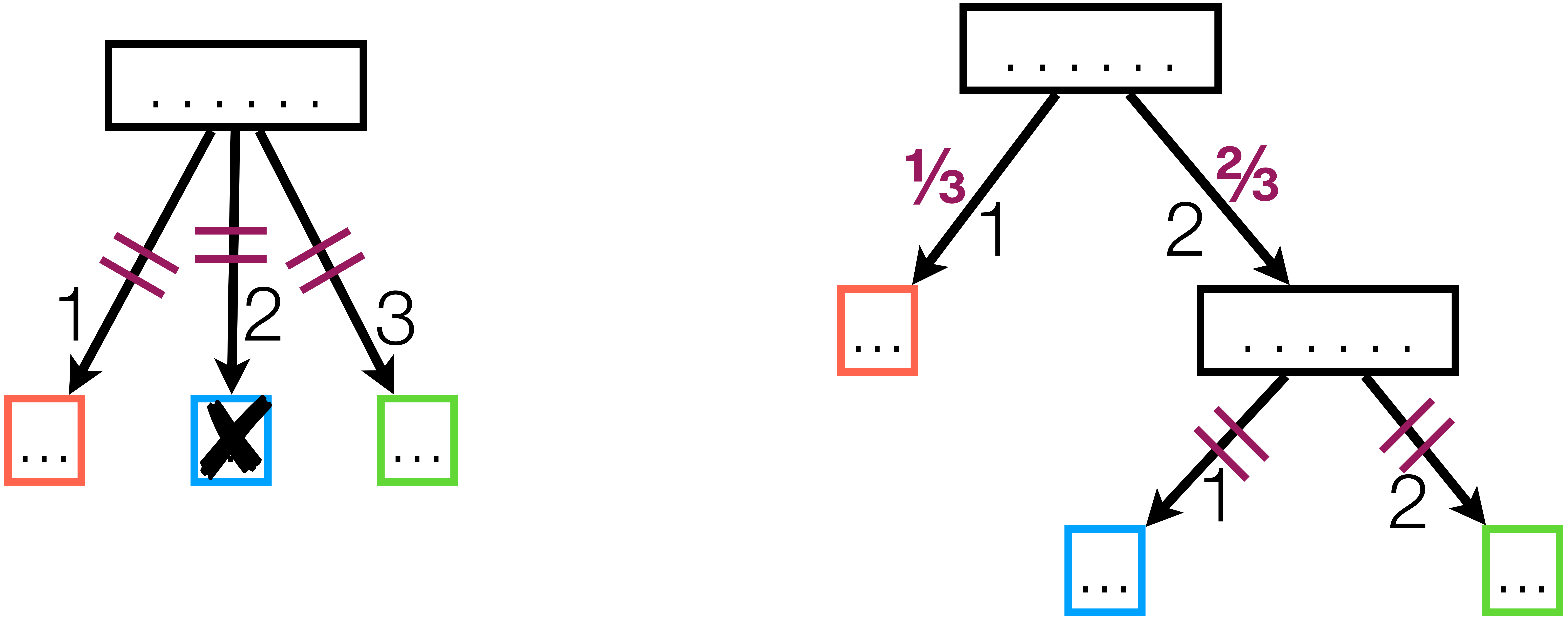
Compiling programs



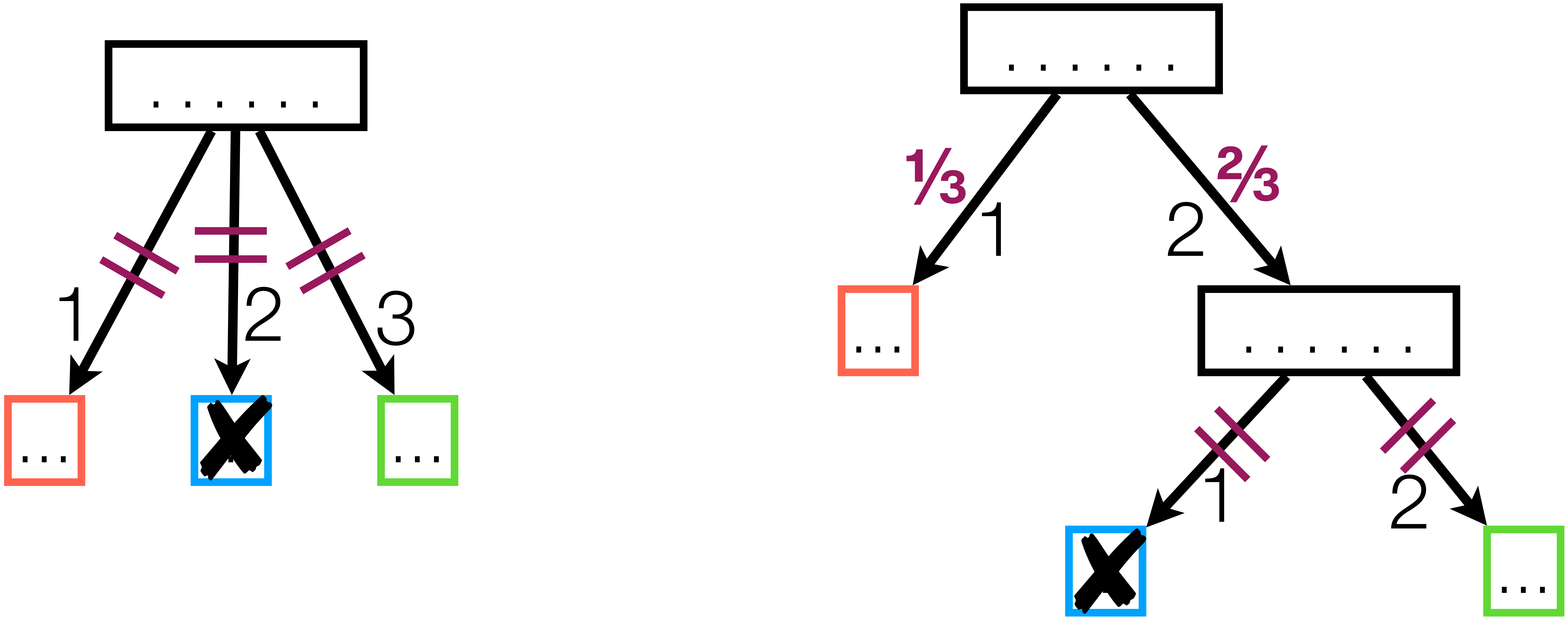
Compiling programs



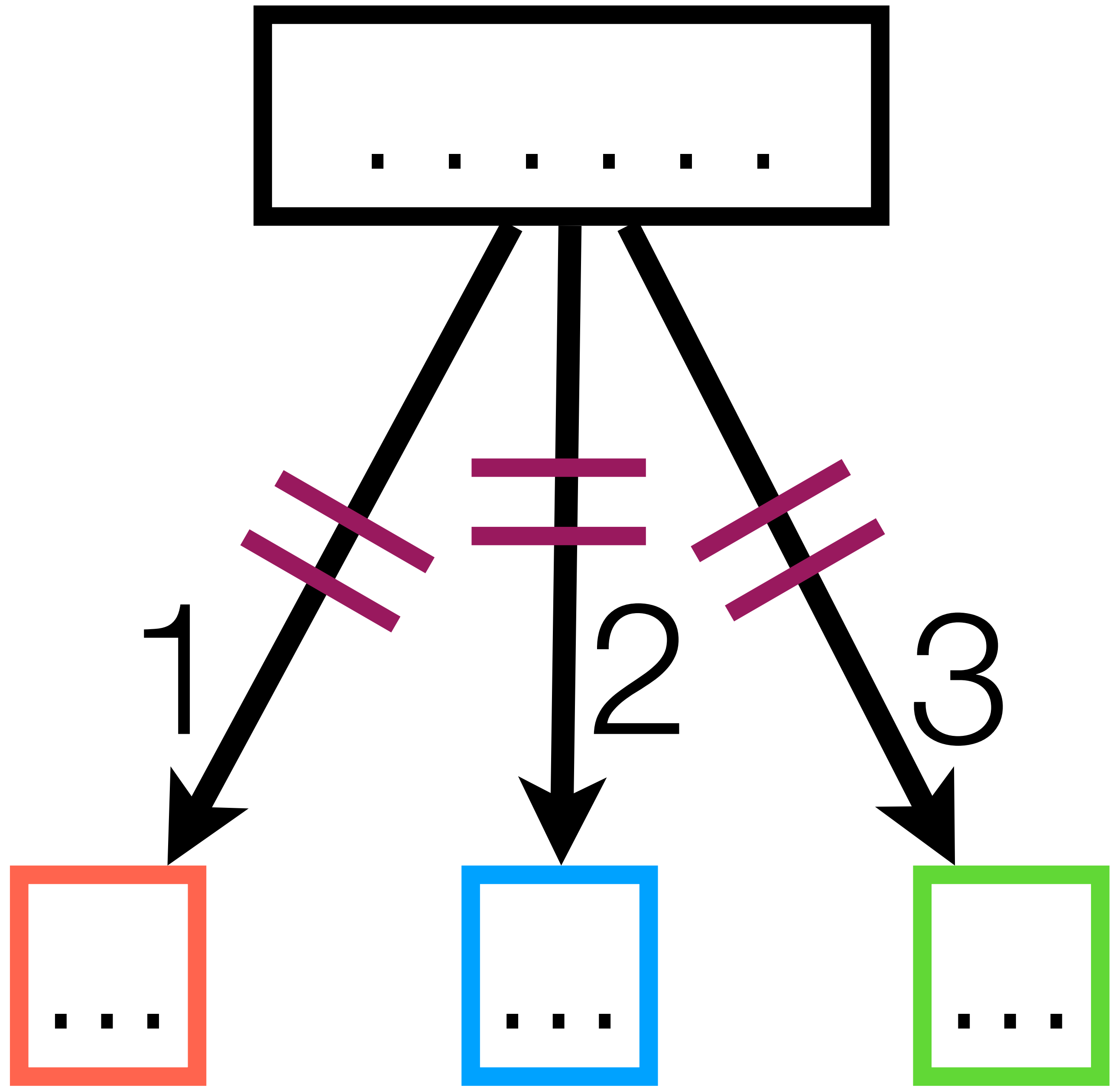
Compiling programs

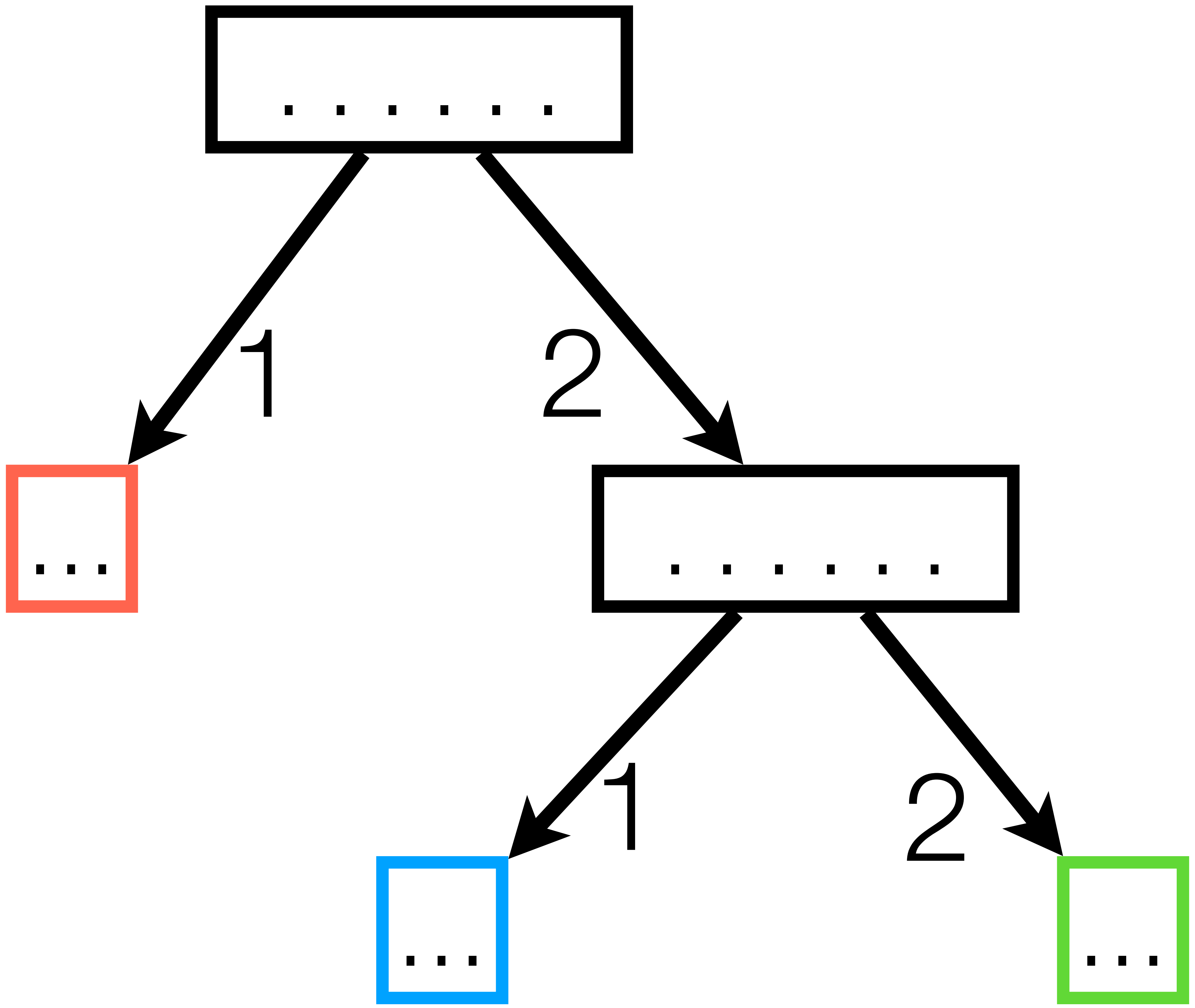


Compiling programs

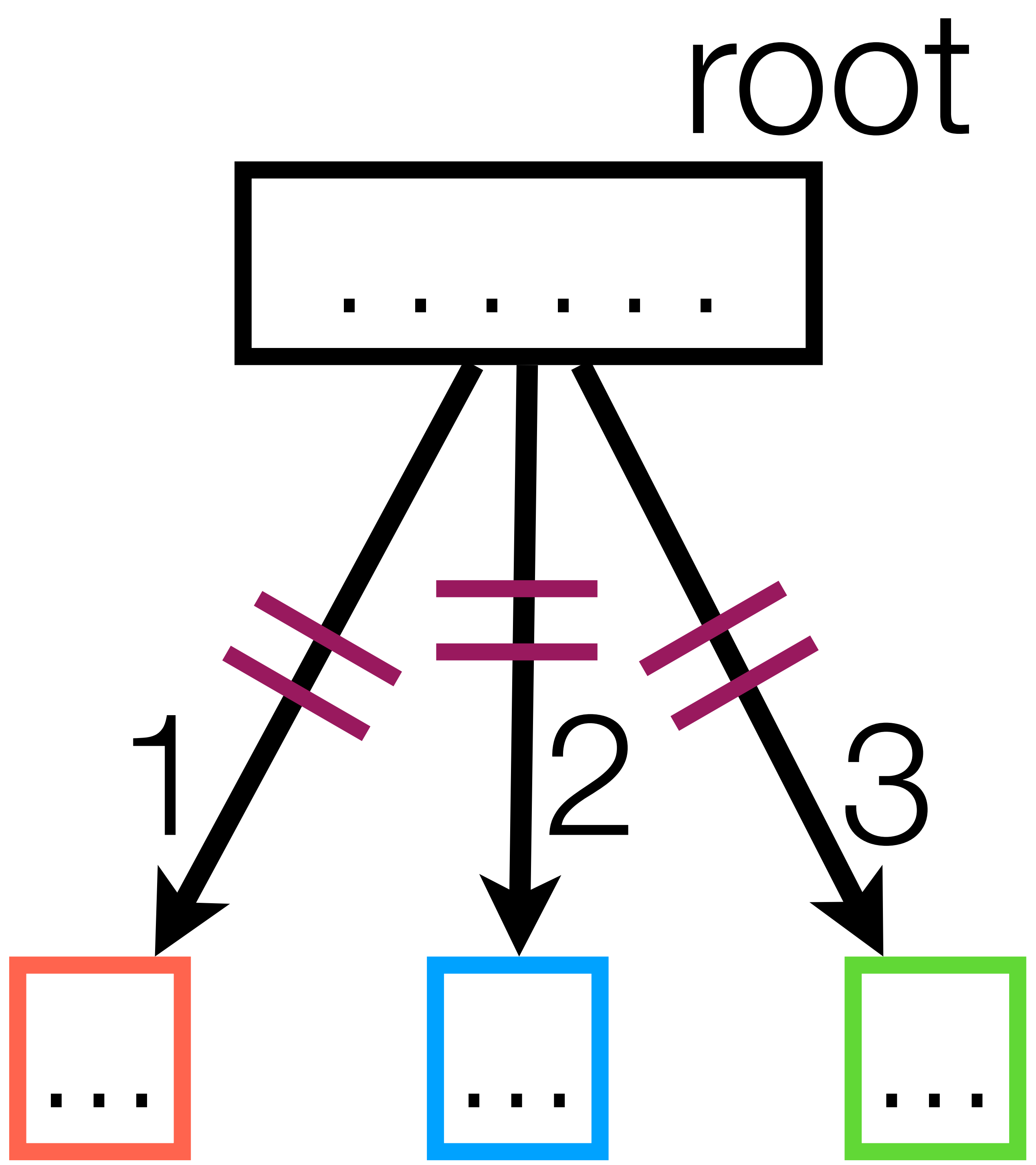


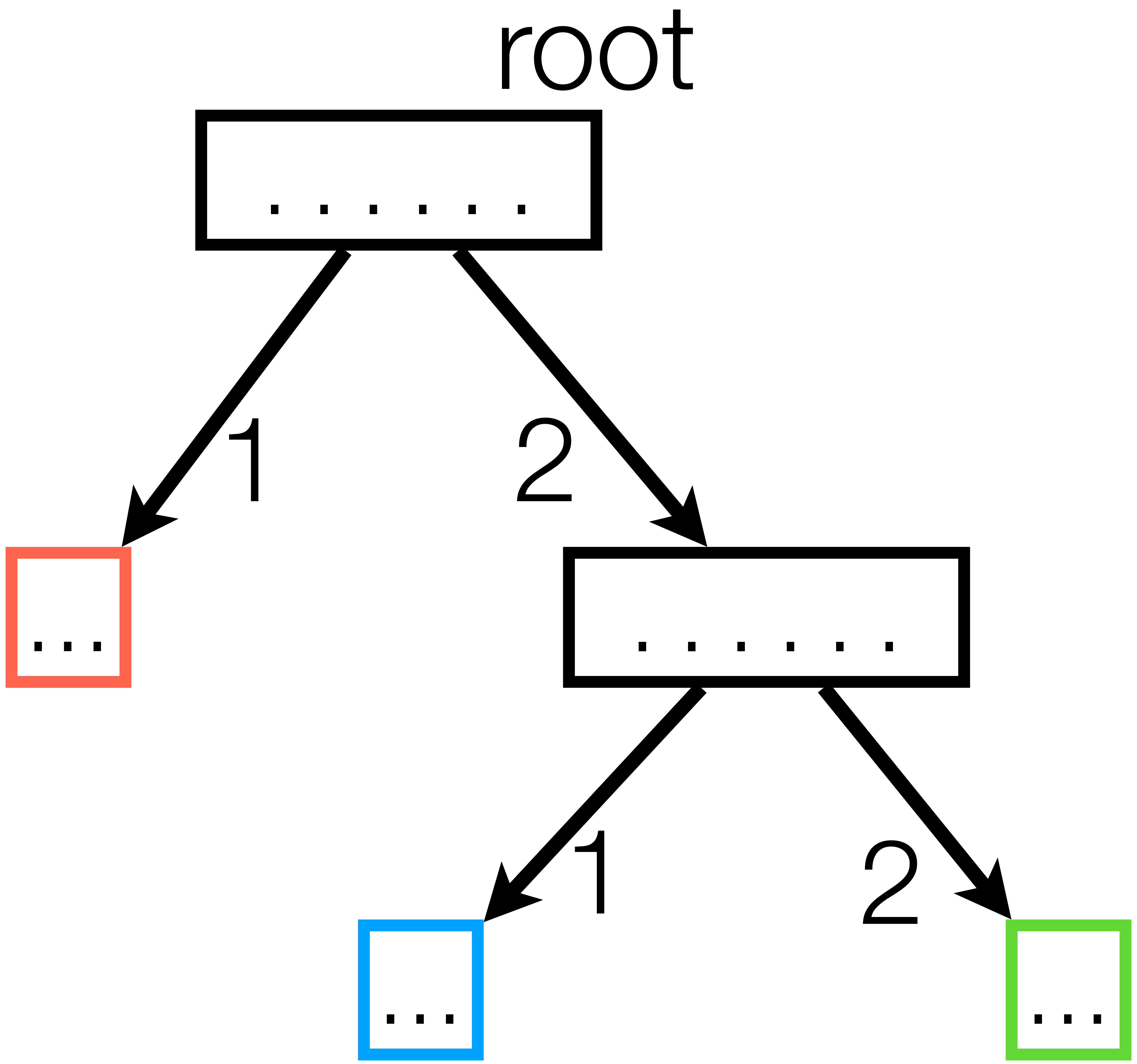
Compiling programs



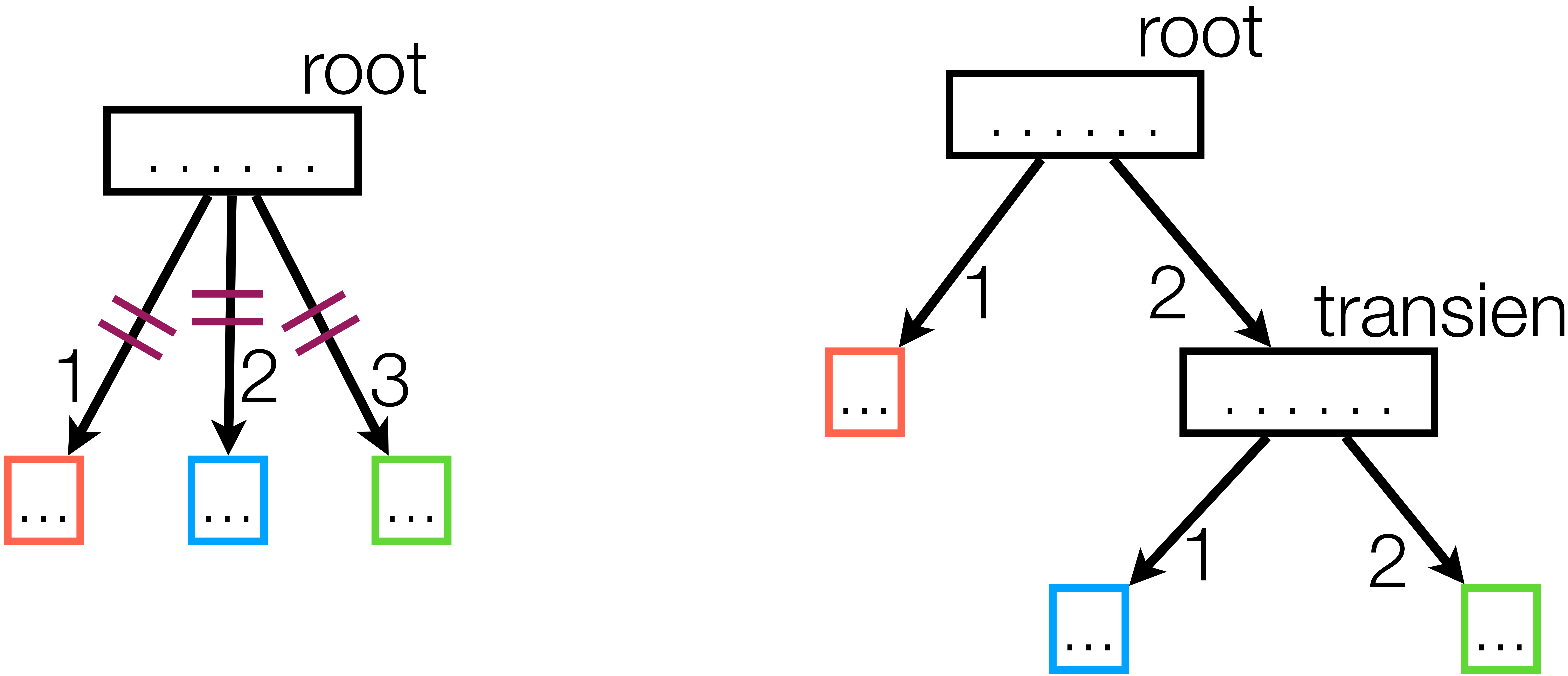


Compiling programs

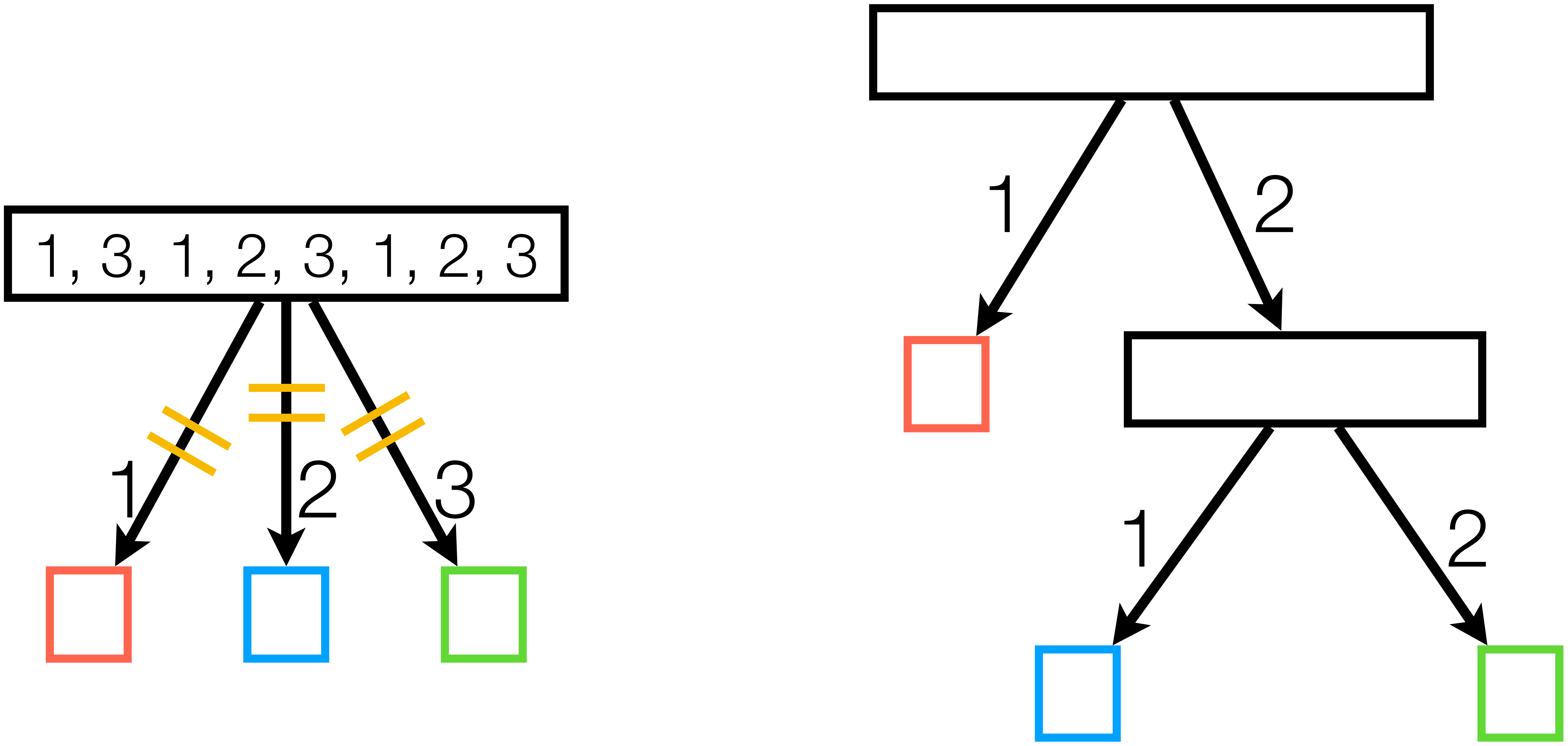




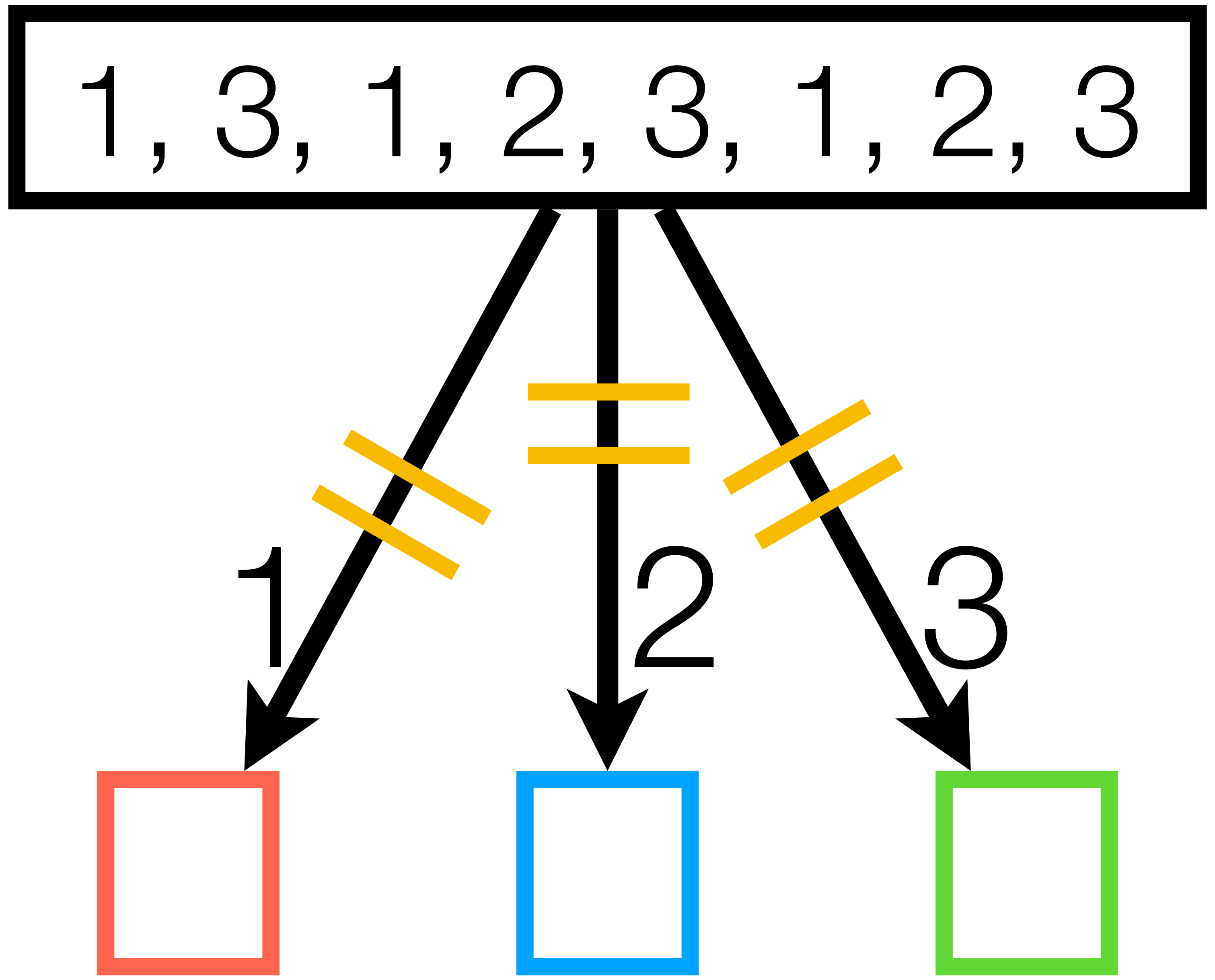
Compiling programs

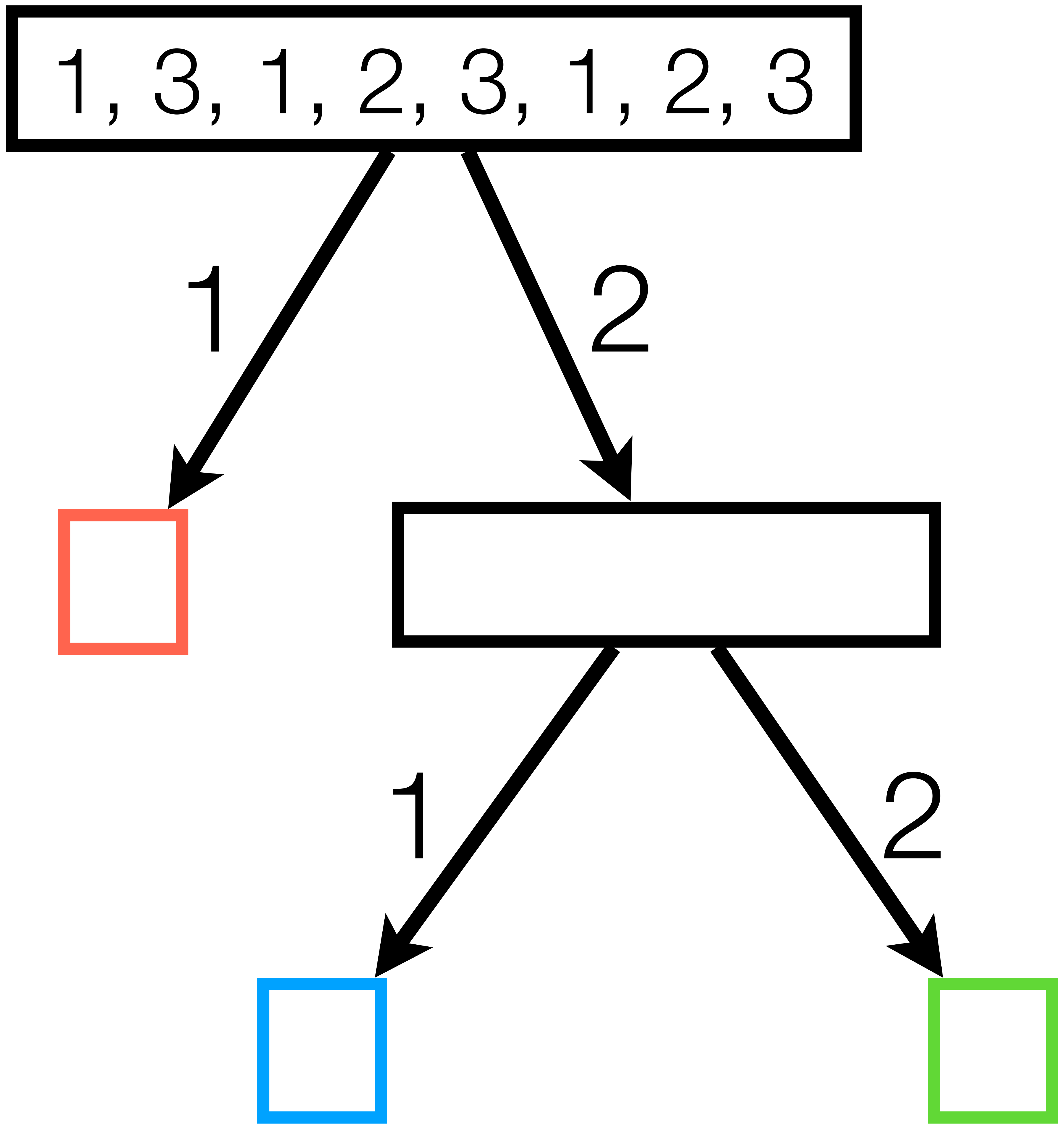


Compiling programs

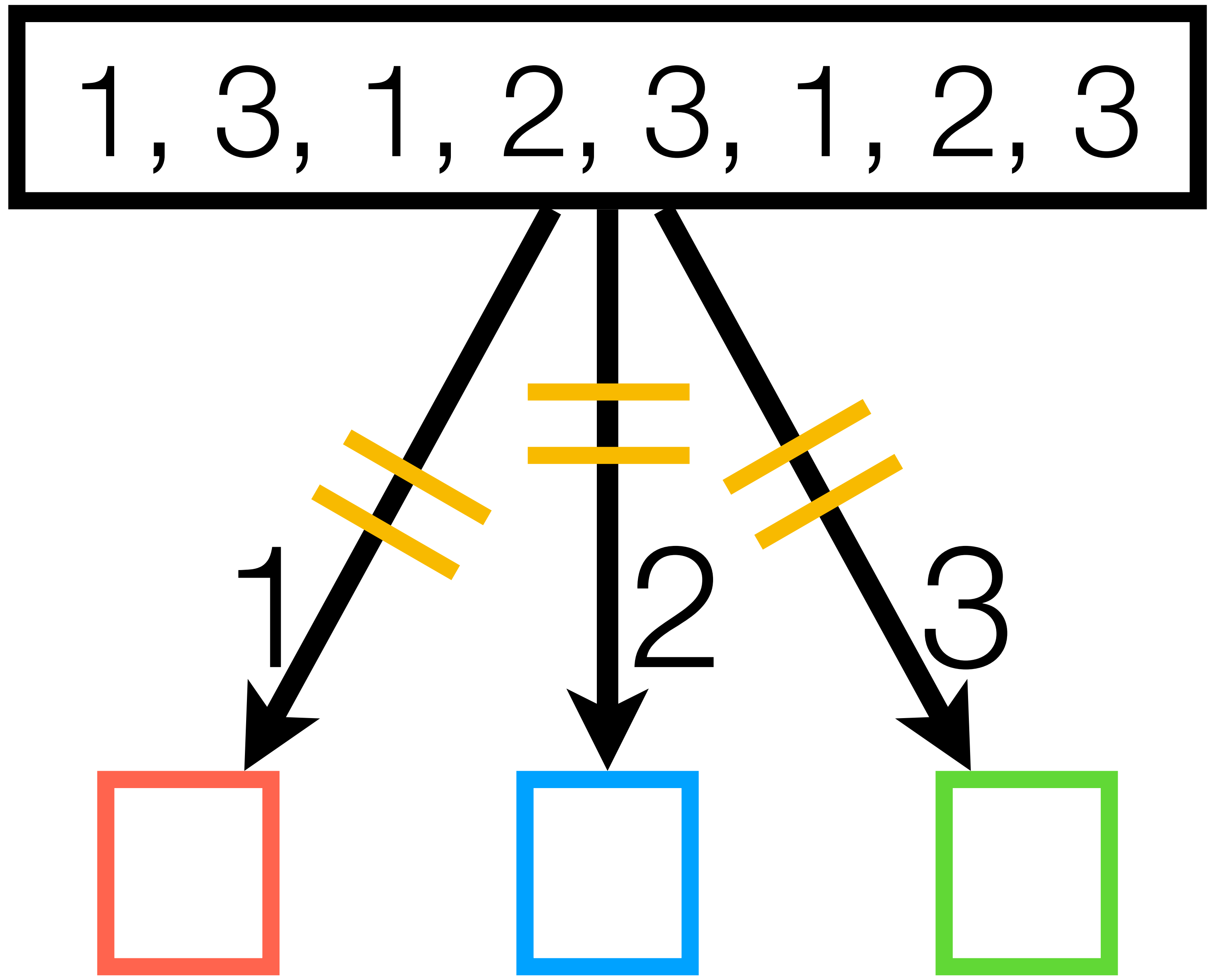


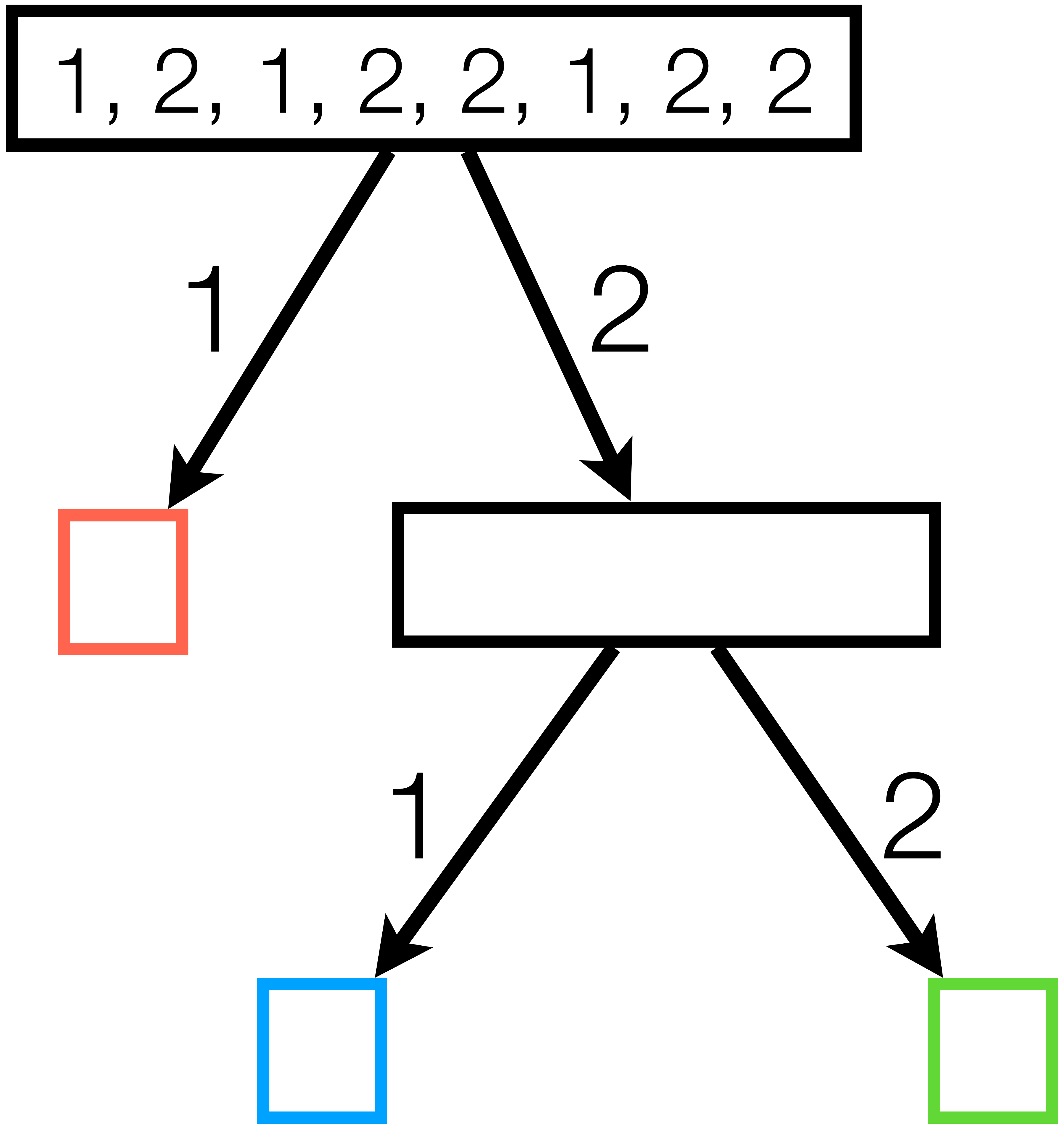
Compiling programs



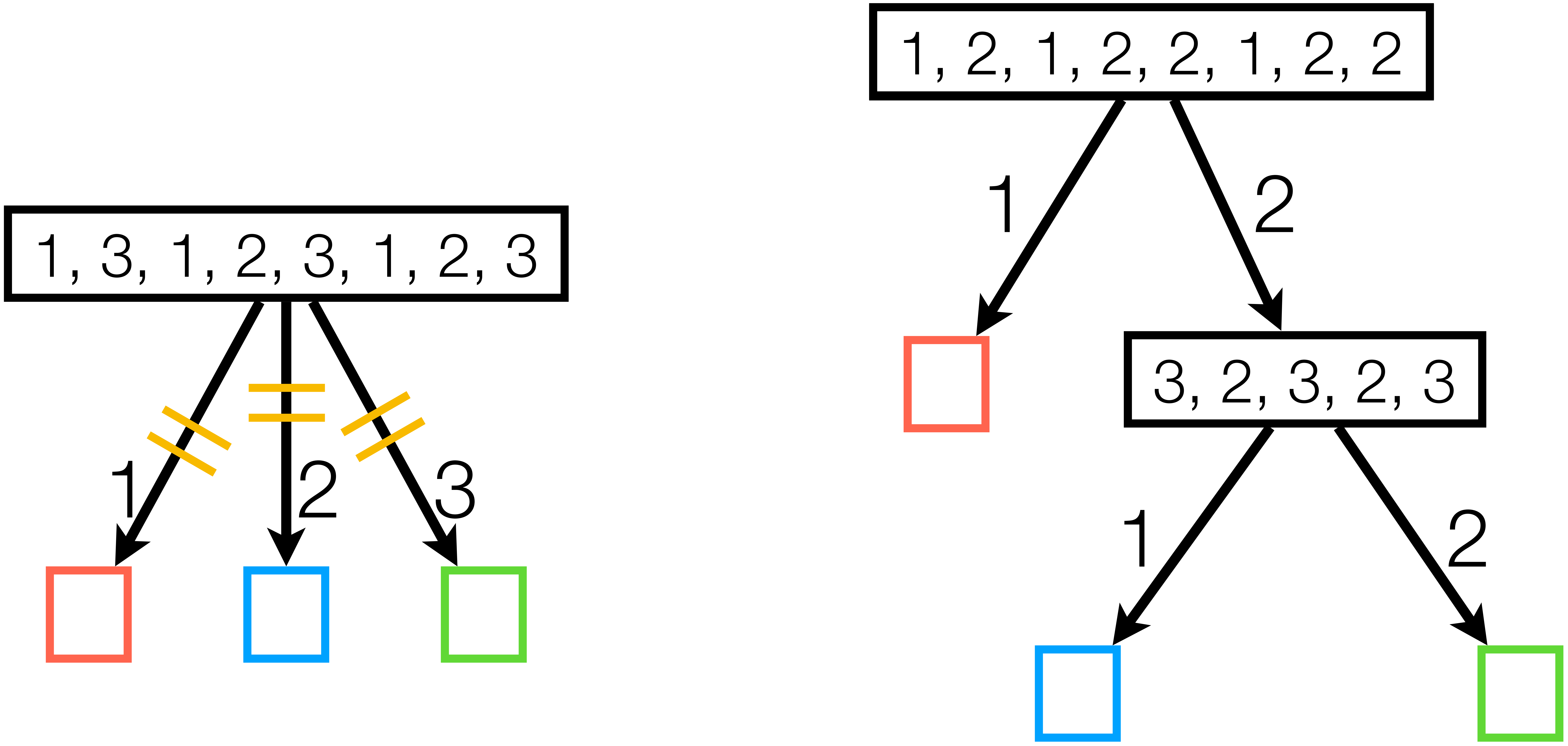


Compiling programs

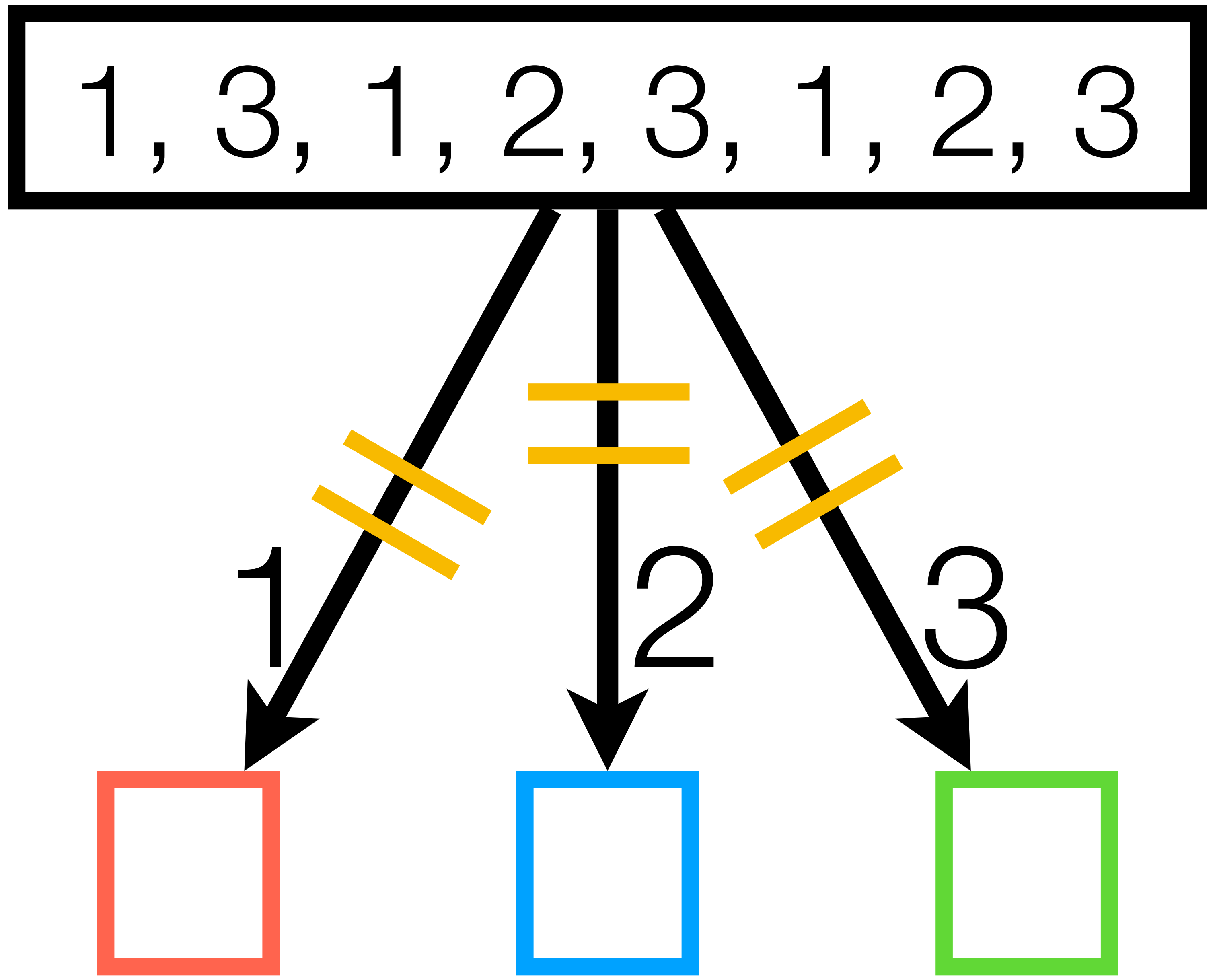


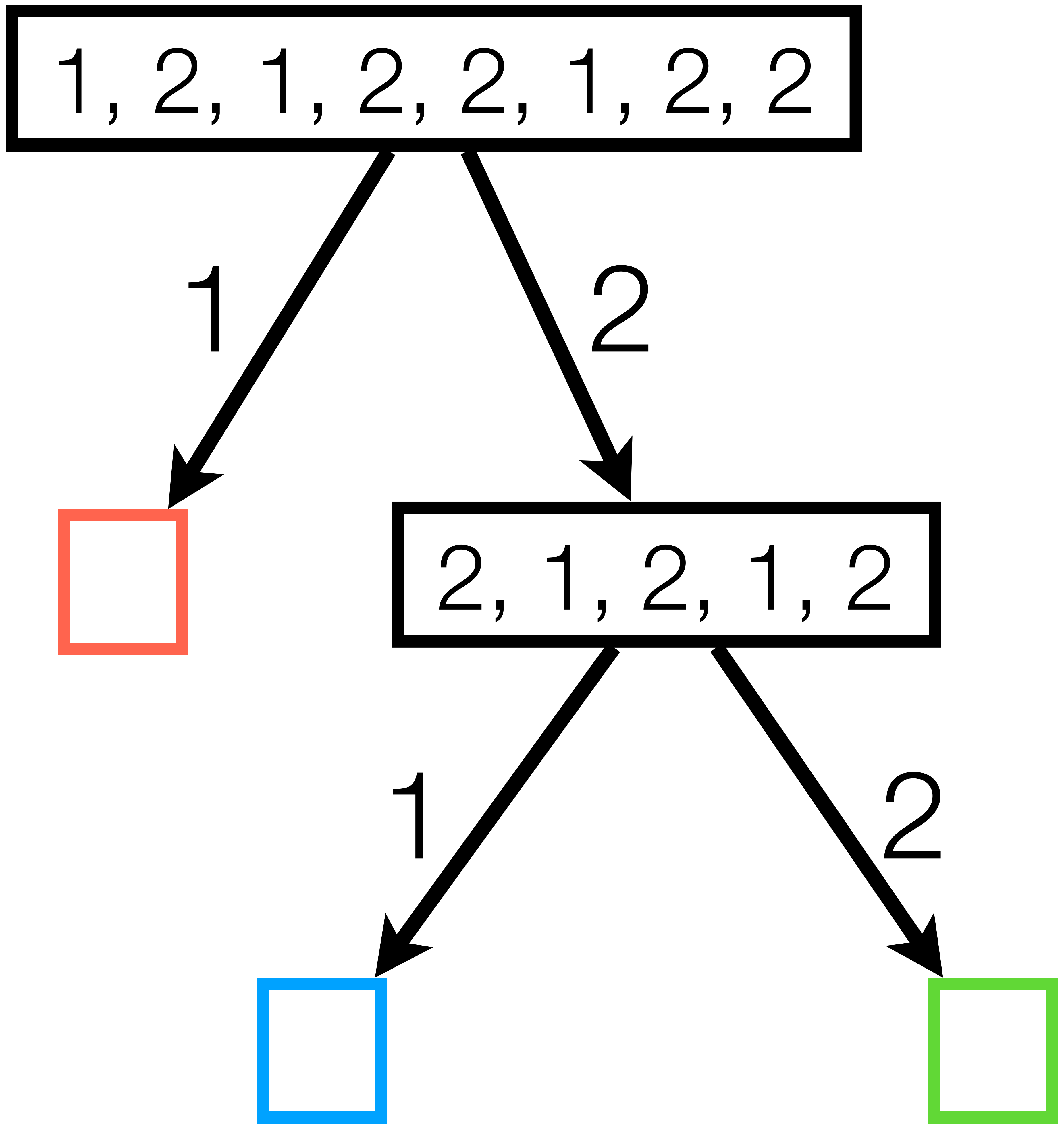


Compiling programs

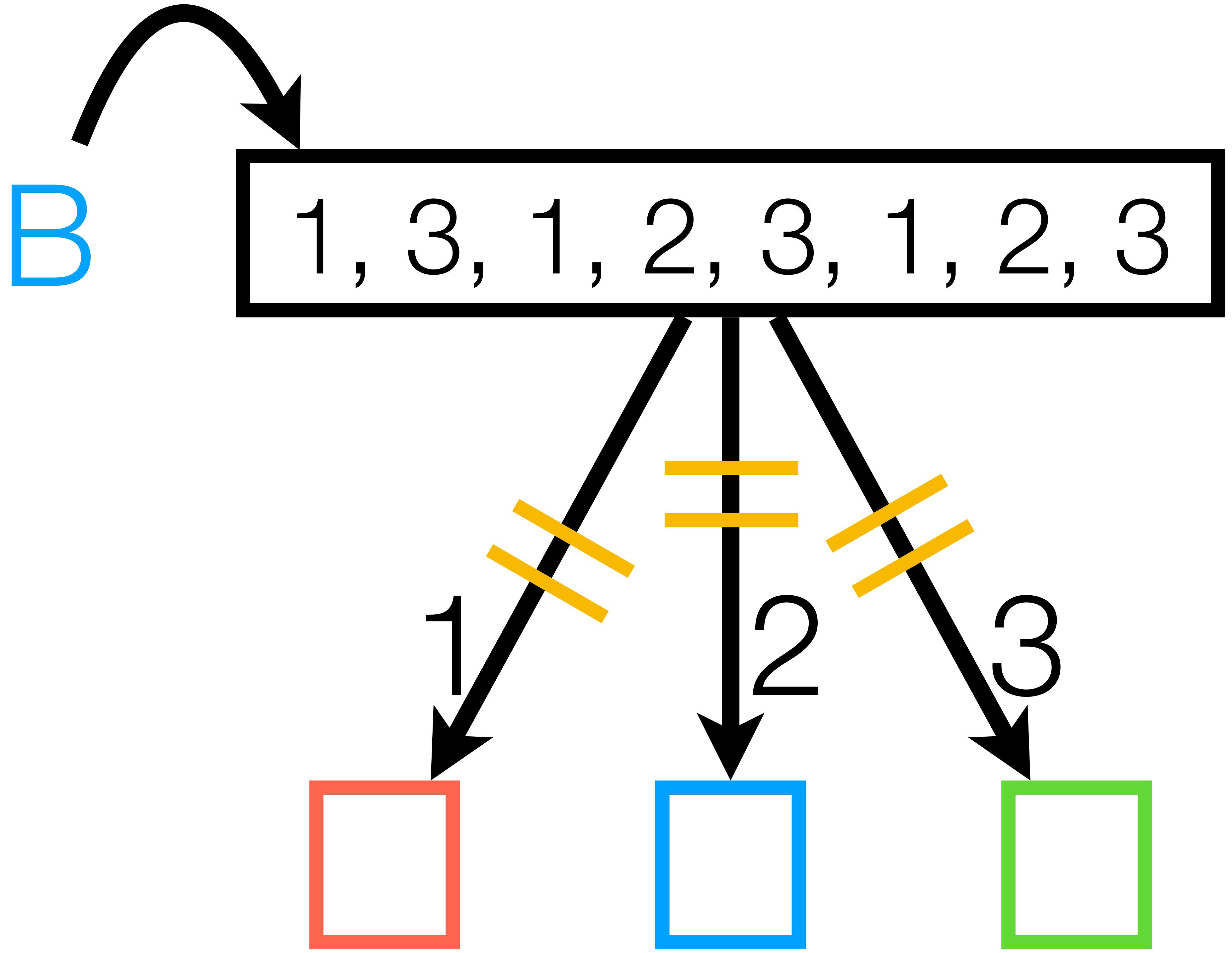


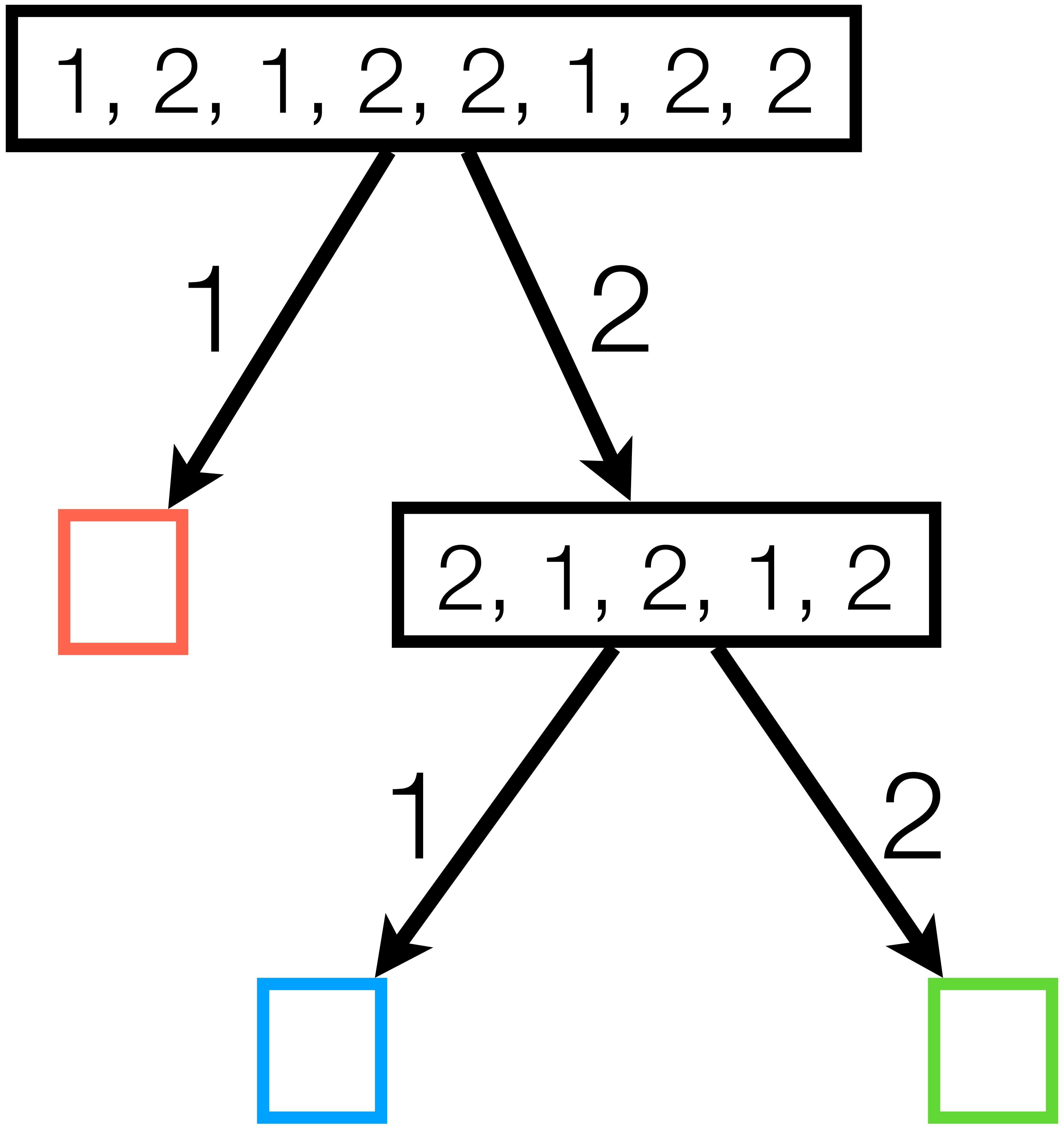
Compiling programs





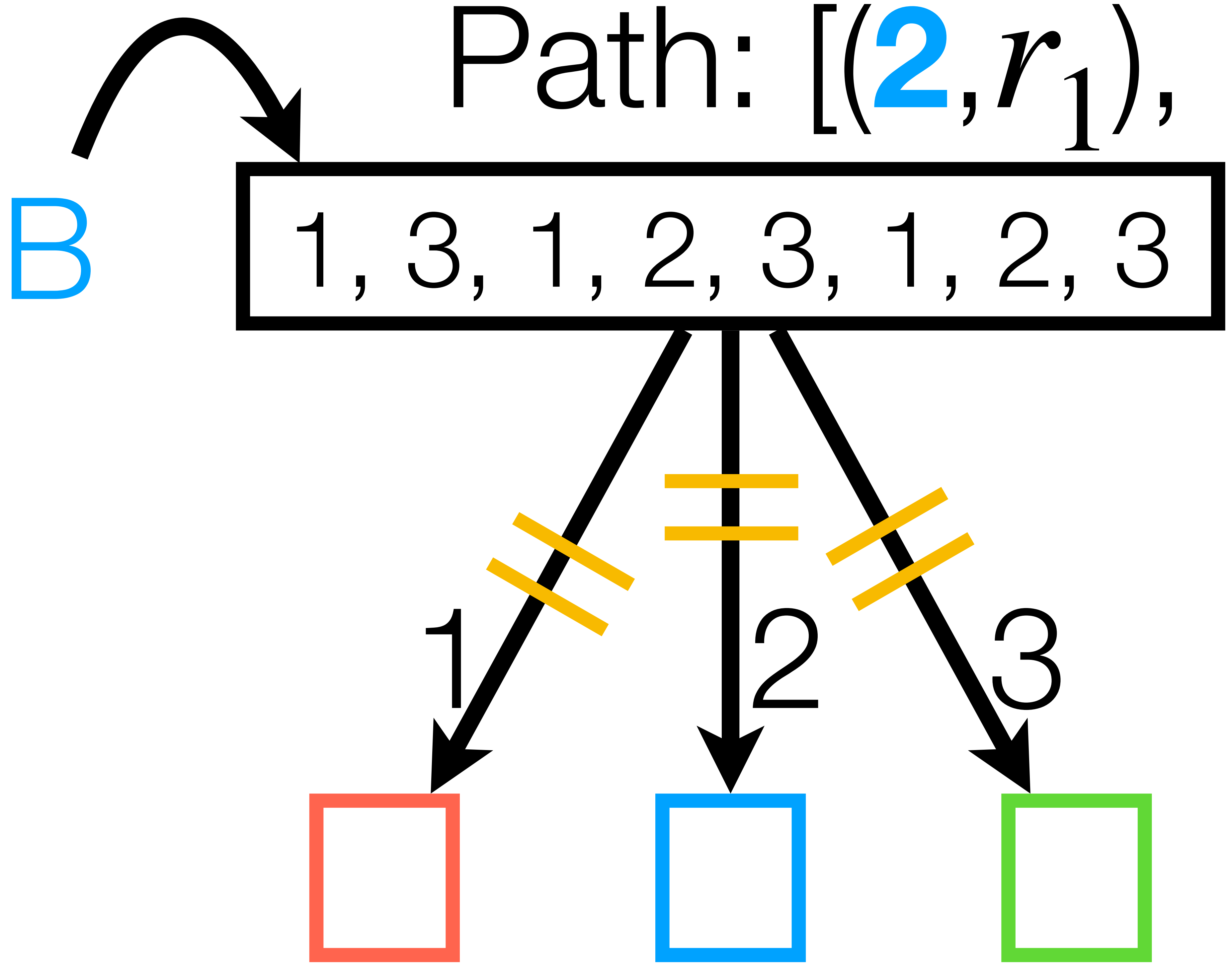
Compiling programs

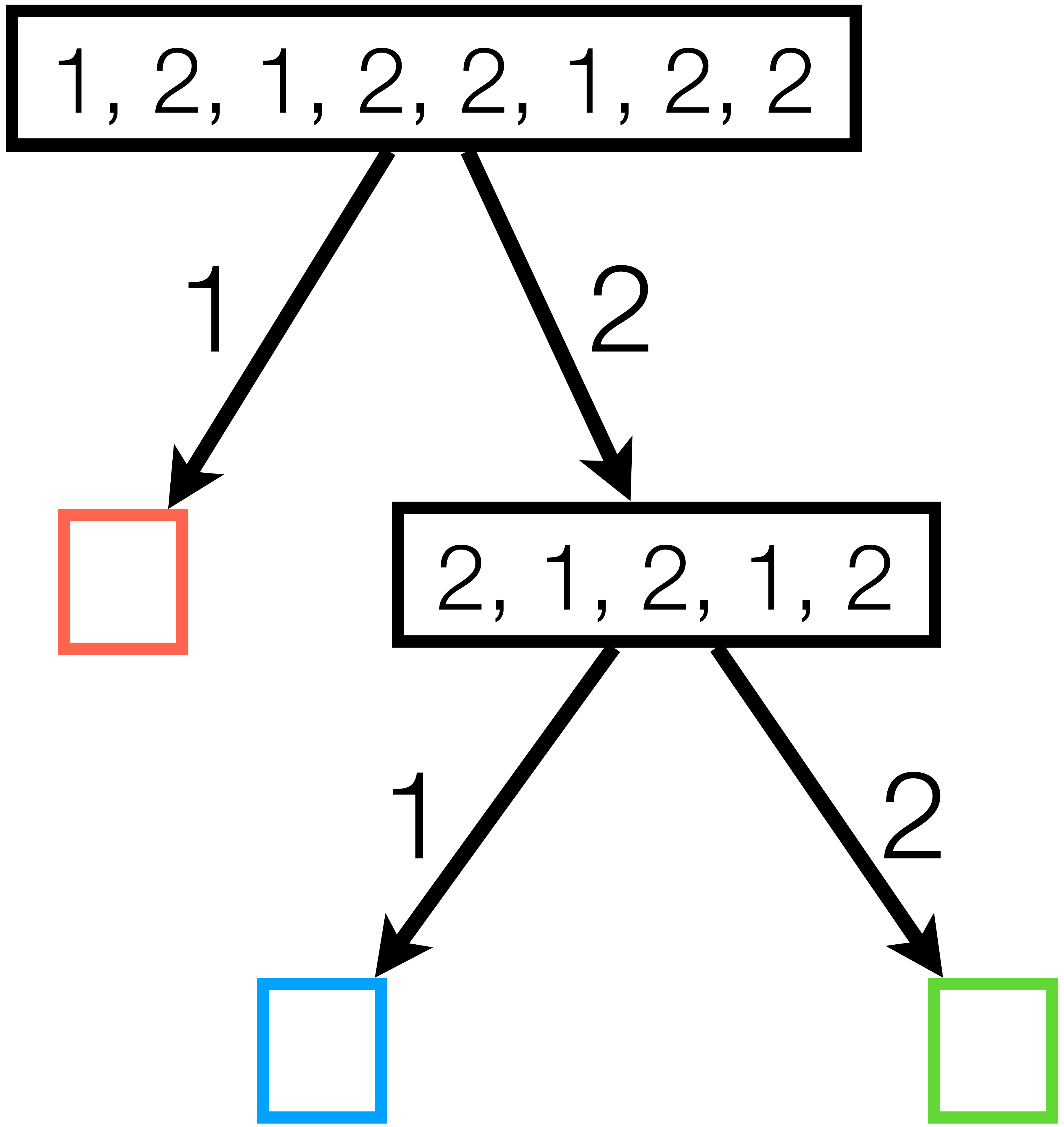




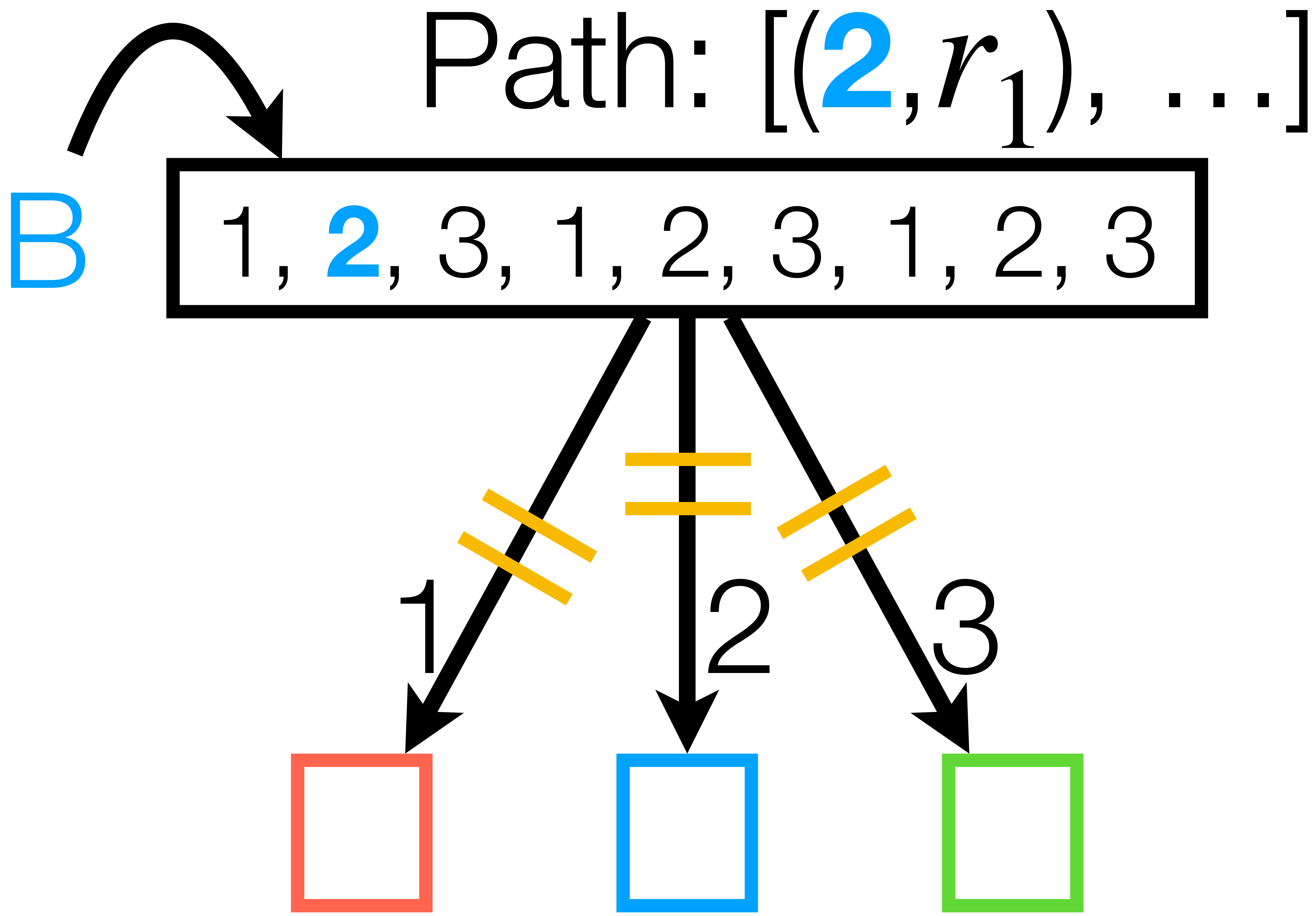
Compiling programs

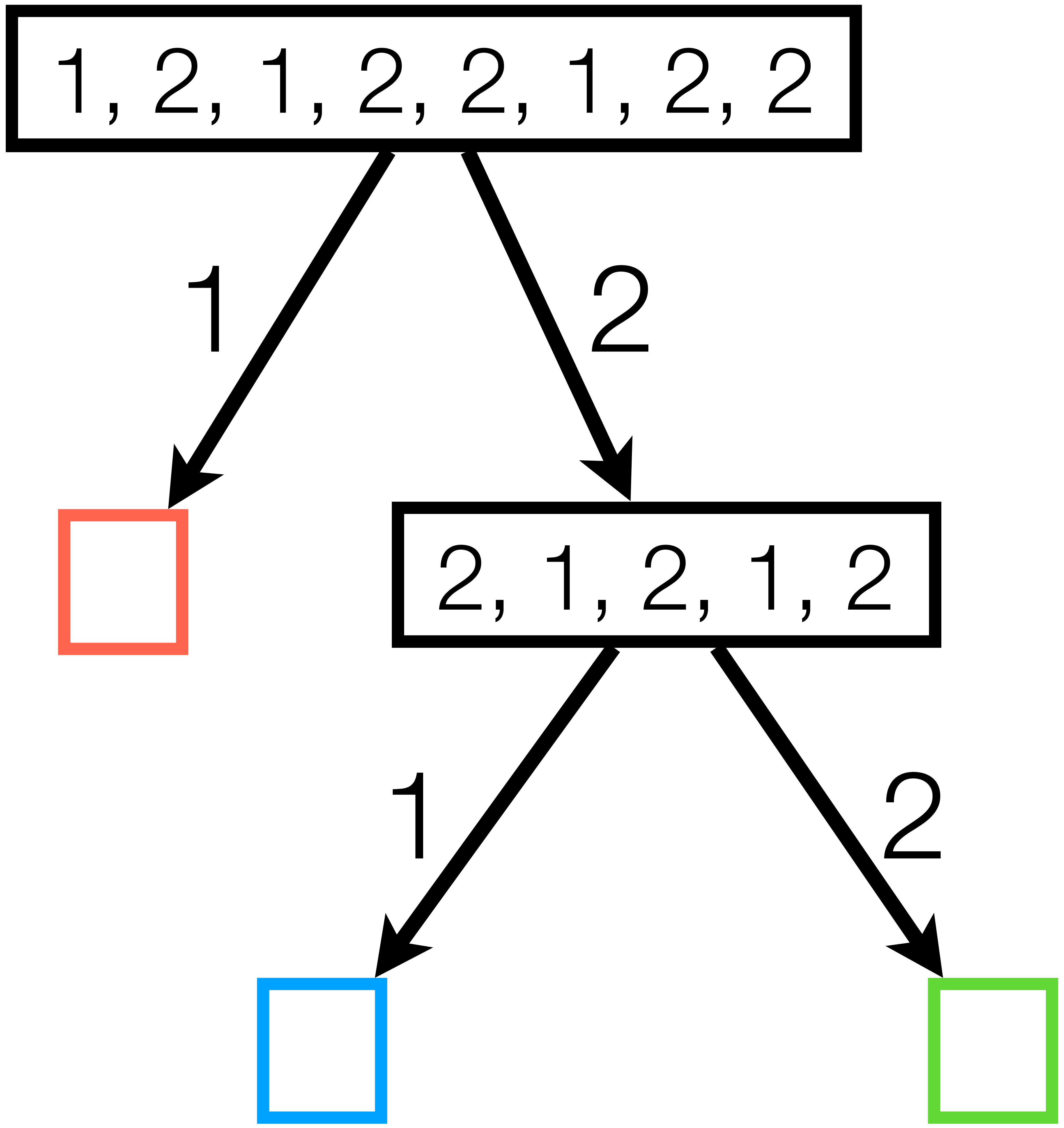
Path: [($\mathbf{2}$,r_1), …]



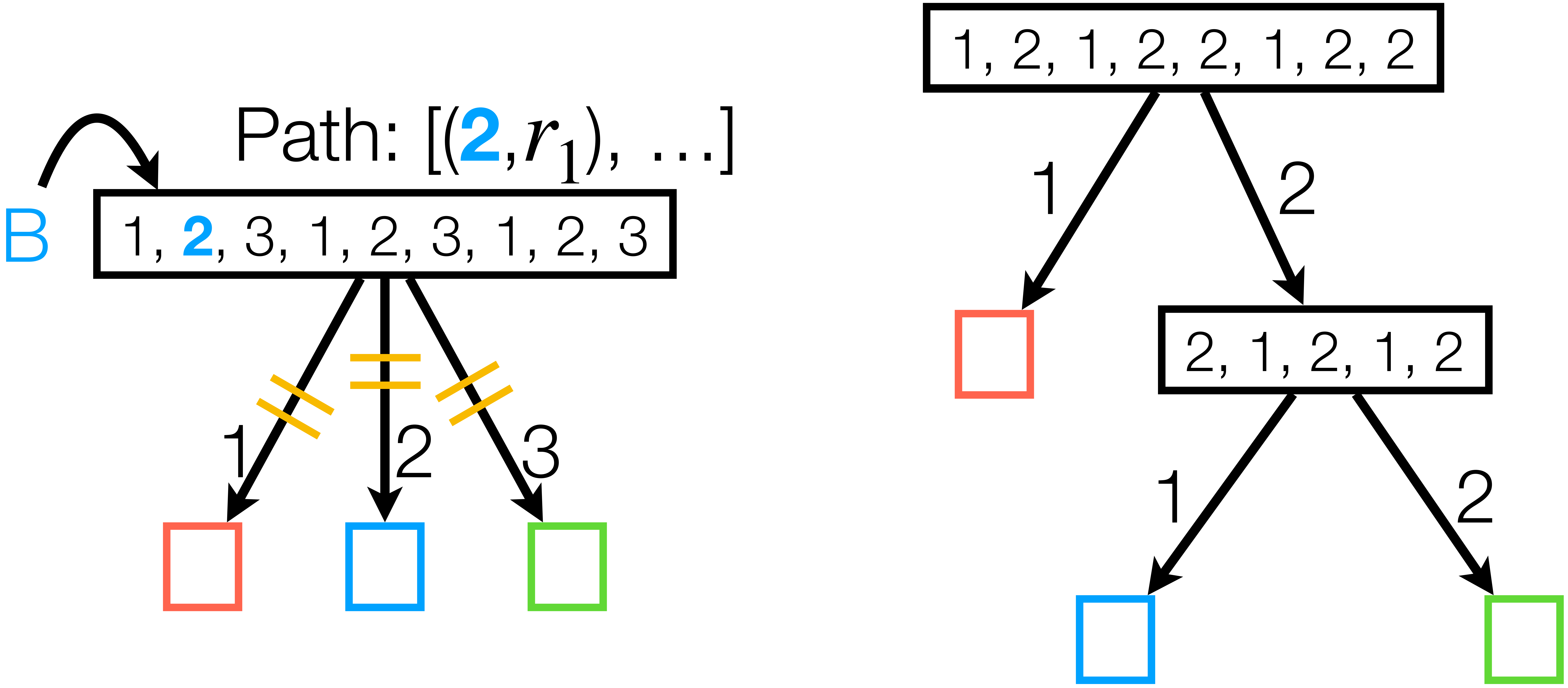


Compiling programs

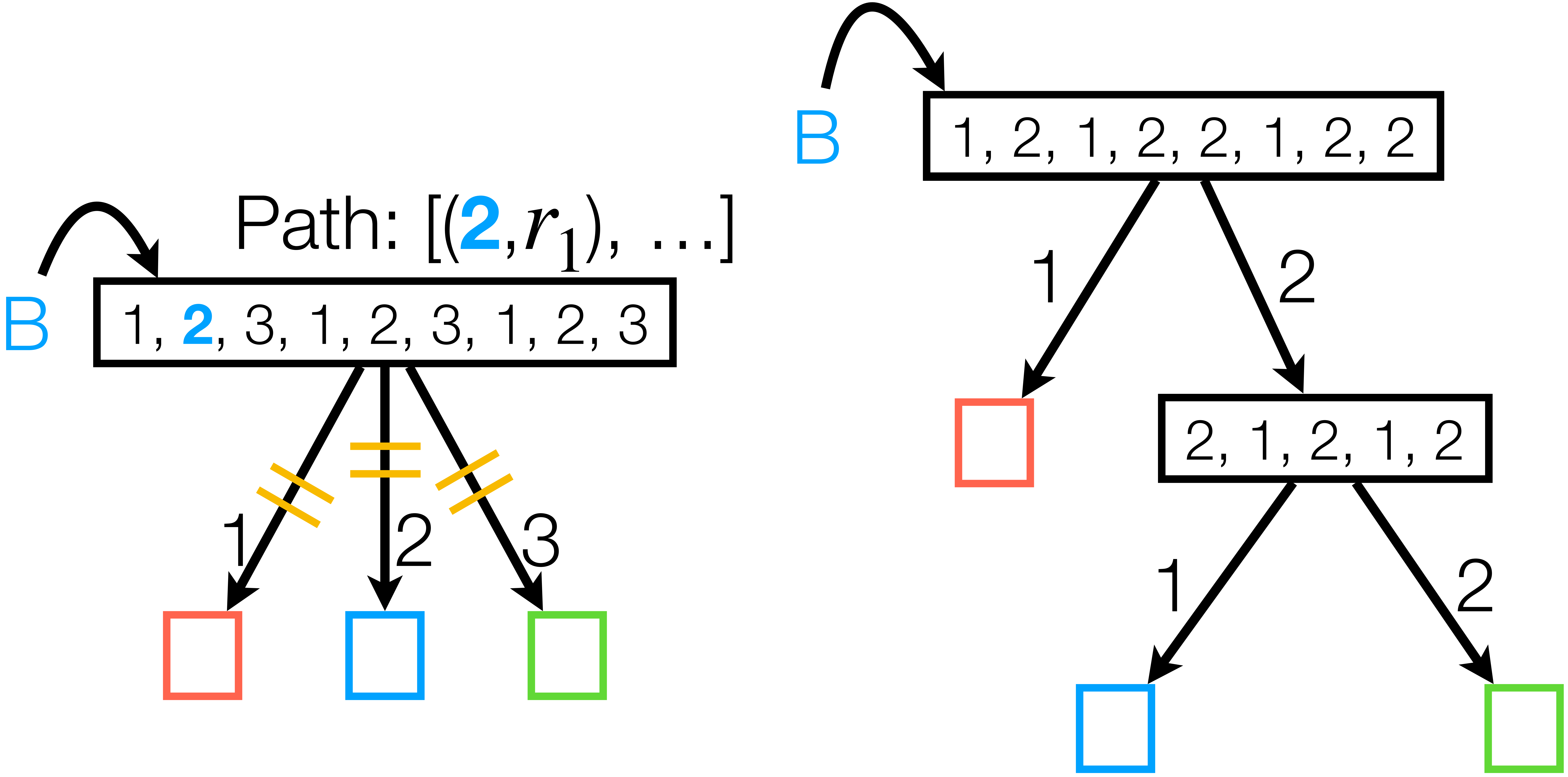




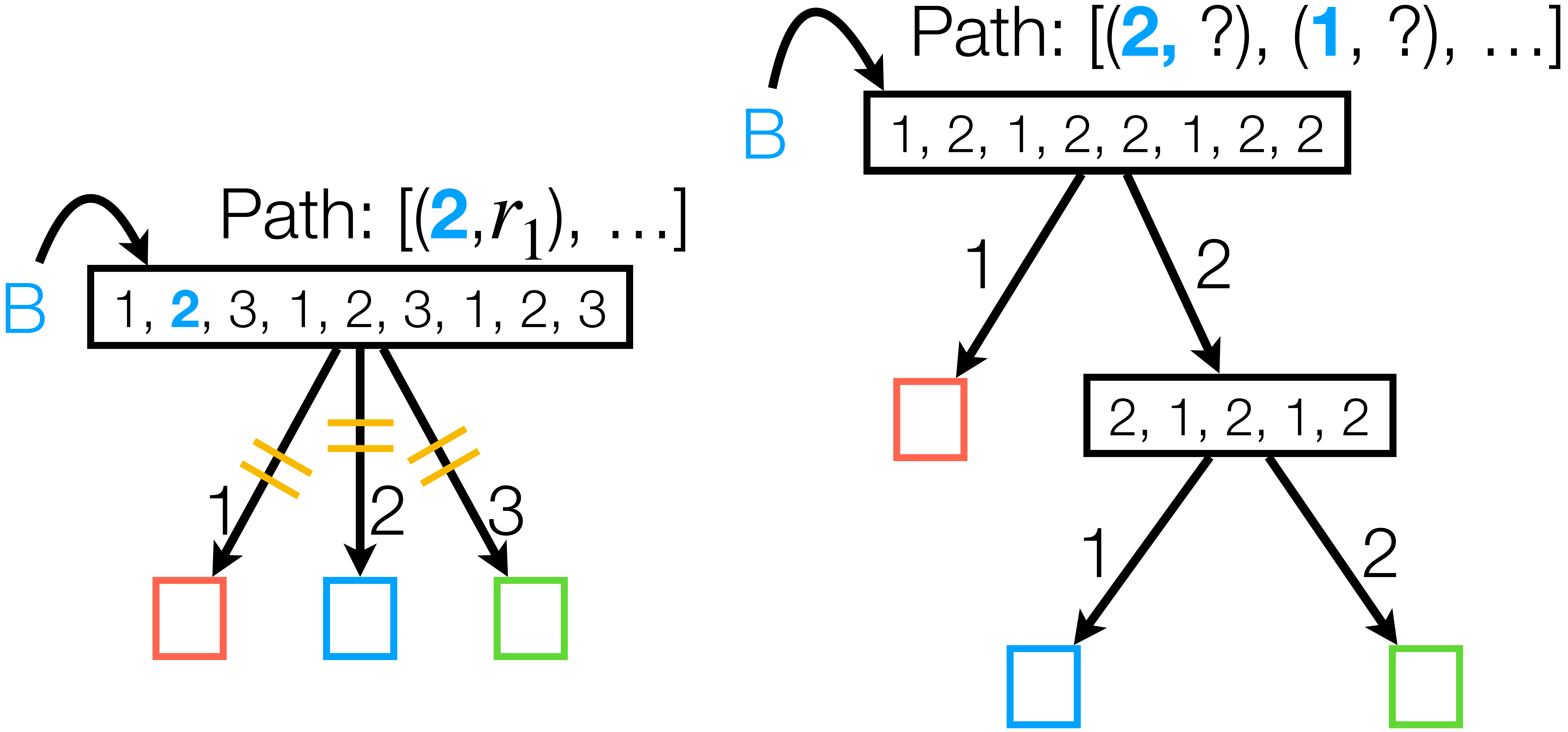
Compiling programs



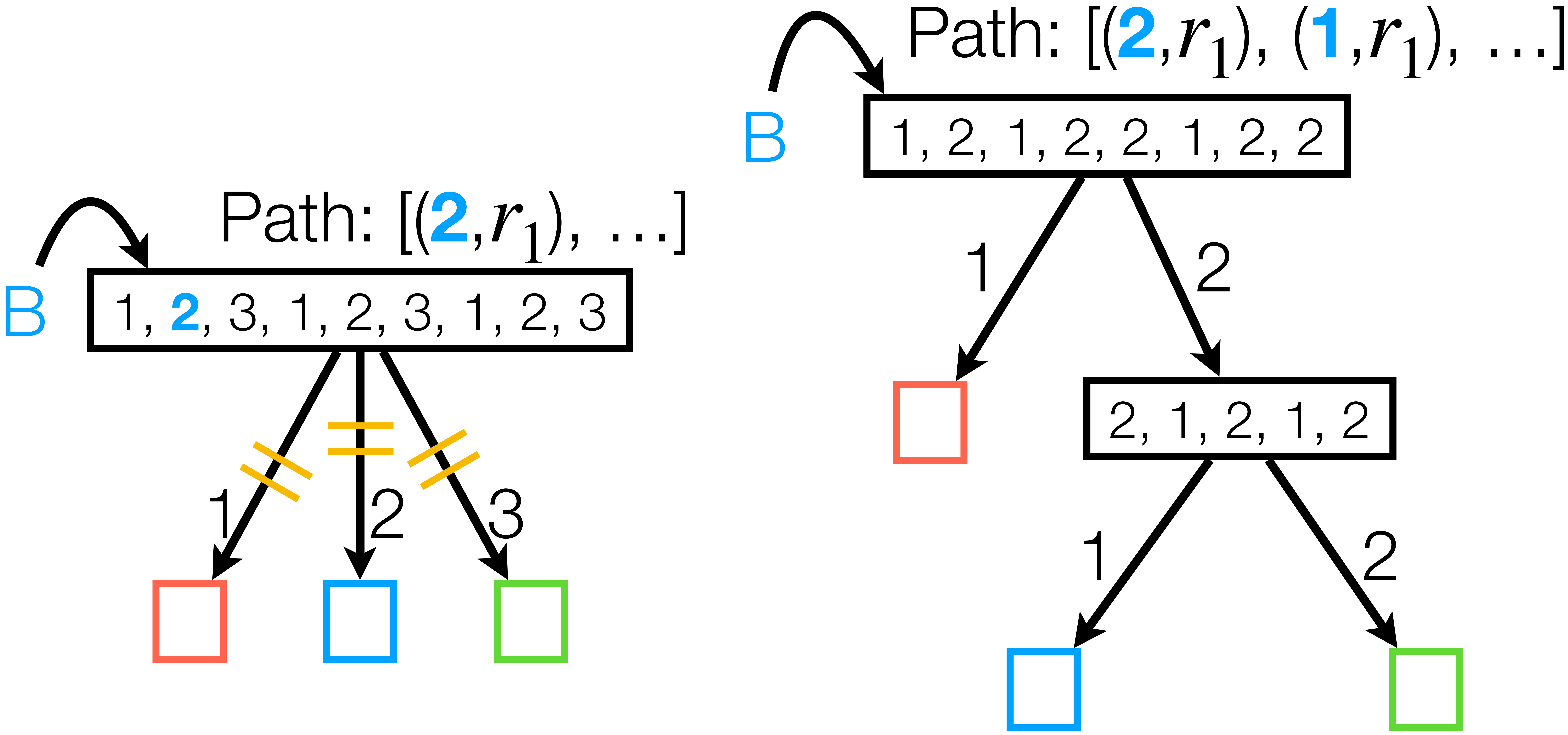
Compiling programs



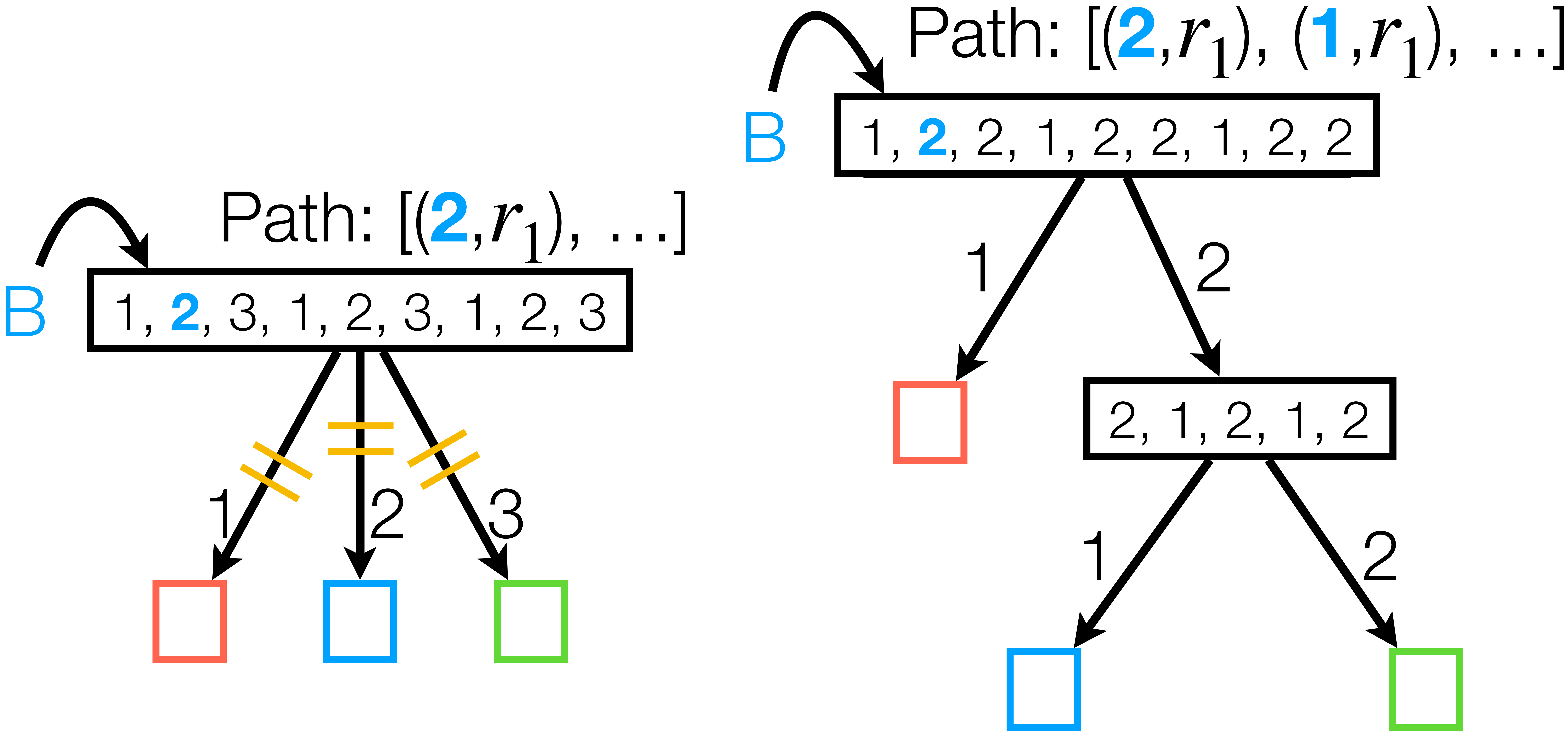
Compiling programs



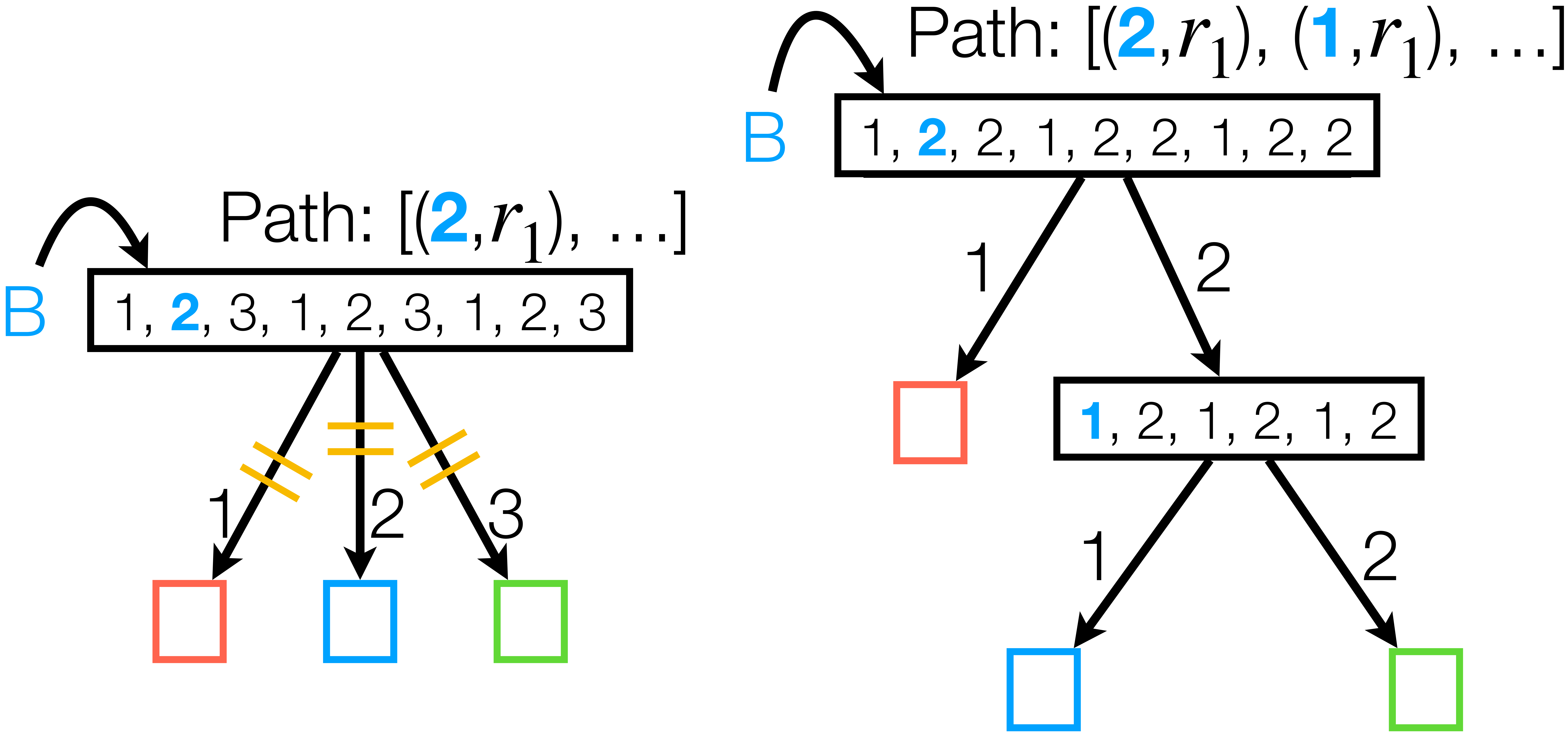
Compiling programs



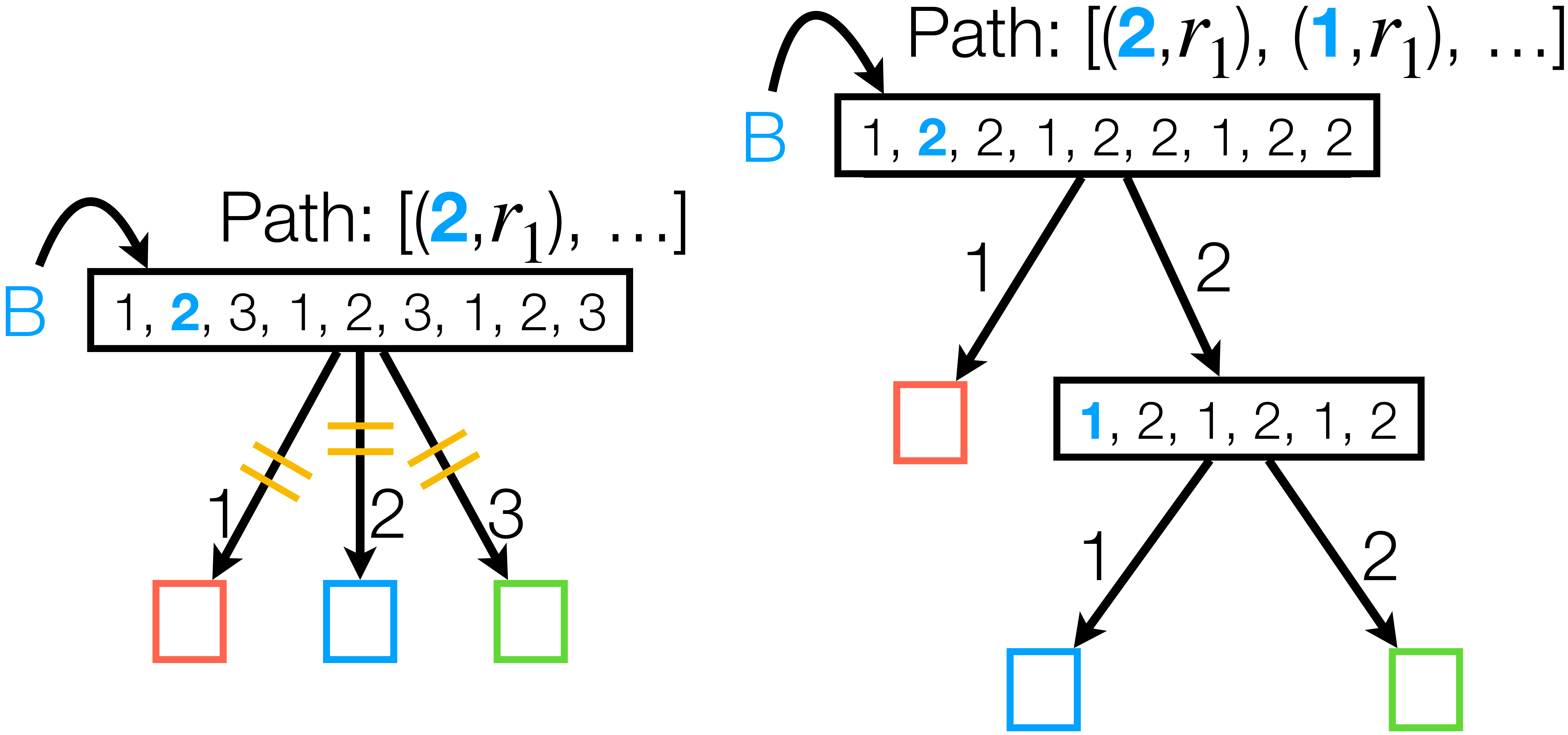
Compiling programs



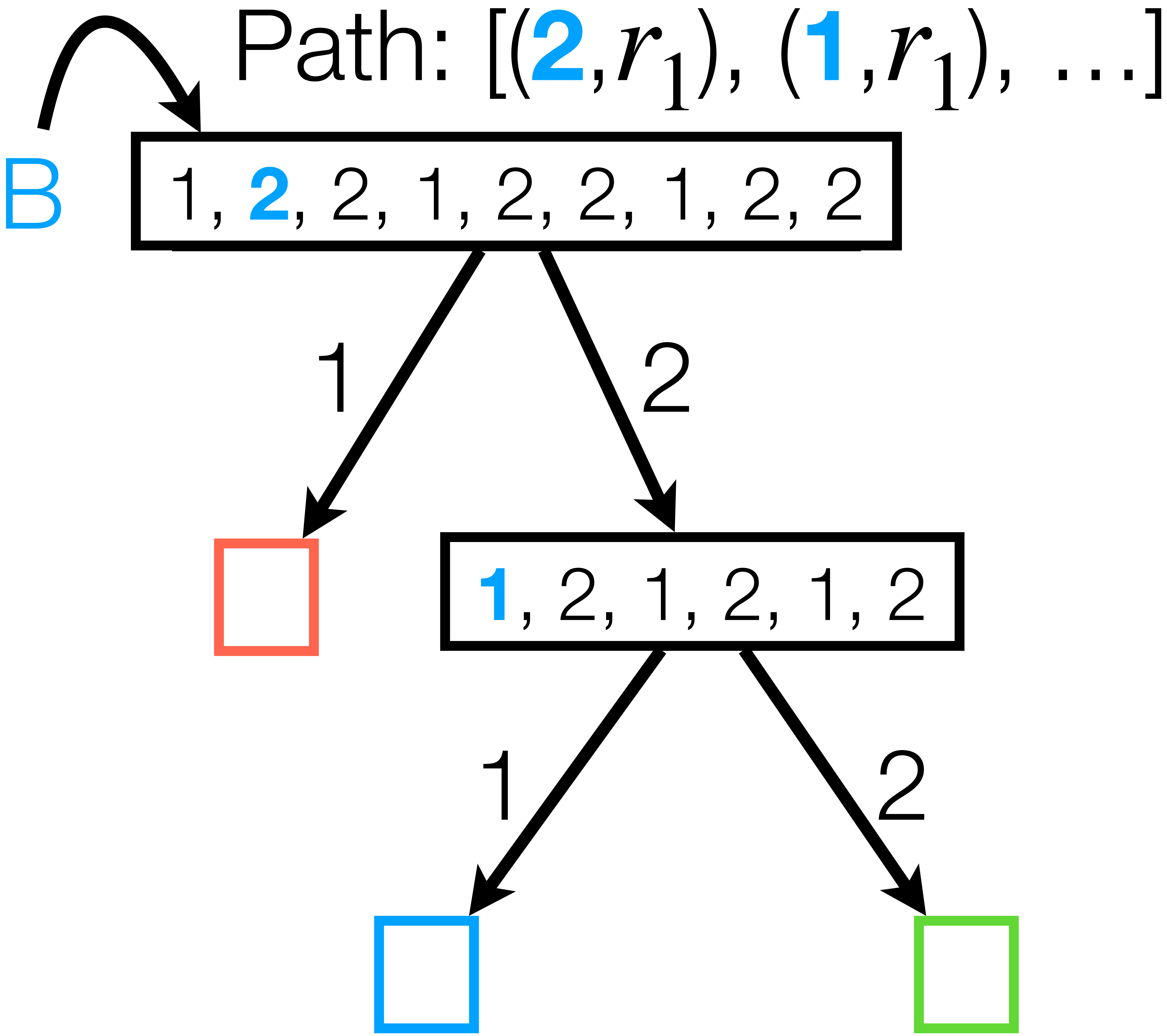
Compiling programs

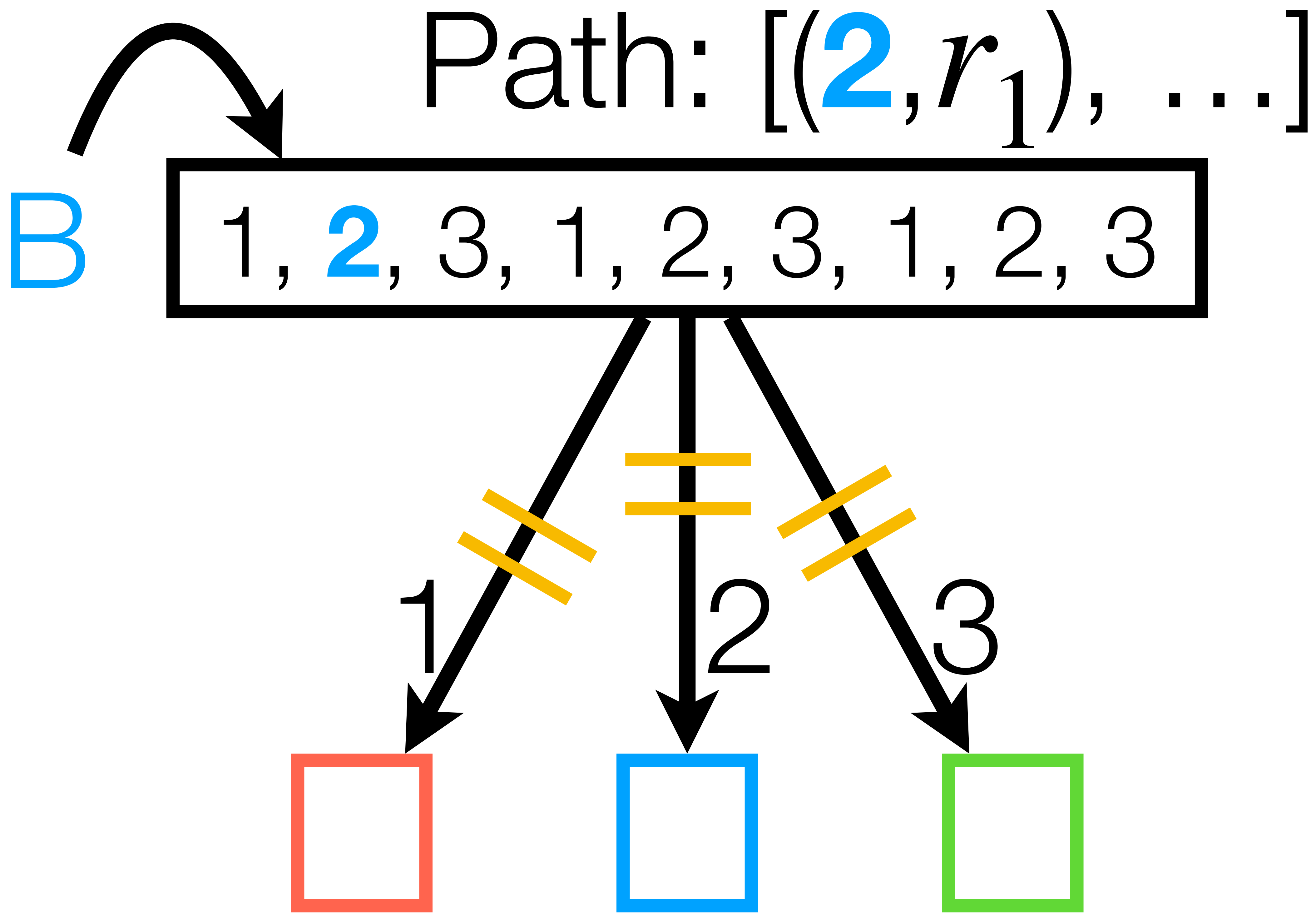


Compiling programs



Given an embedding, we *lift* it to arrive at a compiler.

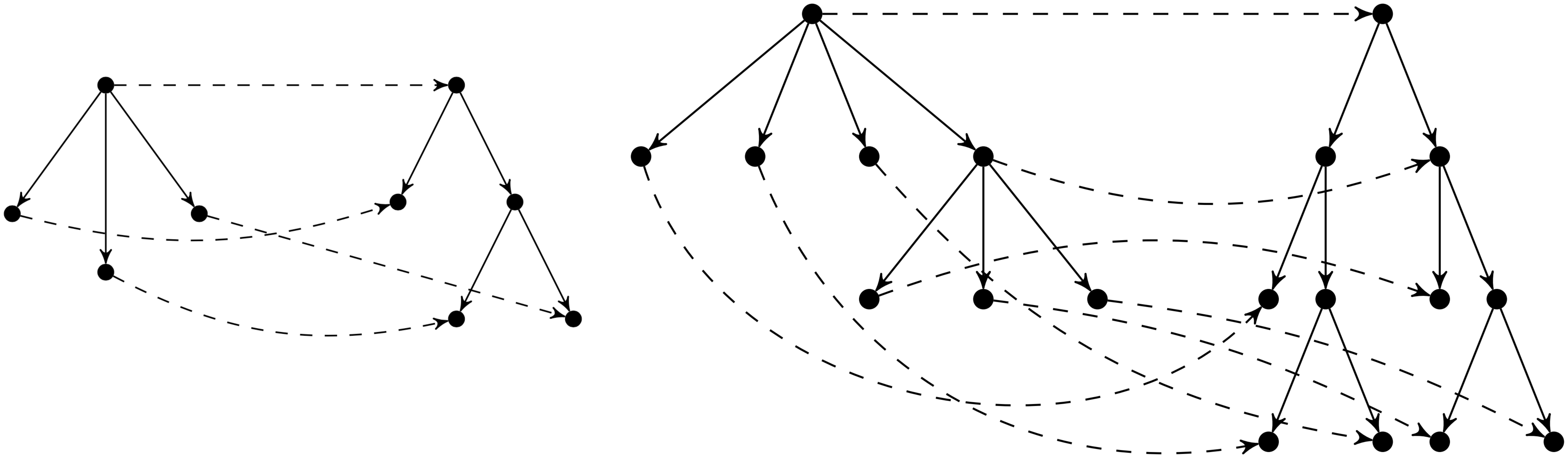




Generating embeddings automatically!

Generating embeddings automatically!

Homomorphic embedding.
Map root to root, leaves to leaves. Respect ancestry.



Generating embeddings automatically!

Homomorphic embedding.
Map root to root, leaves to leaves. Respect ancestry.

Two new algorithms,
both starting with heterogeneous source trees.

Generating embeddings automatically!

Homomorphic embedding.
Map root to root, leaves to leaves. Respect ancestry.

Two new algorithms,
both starting with heterogeneous source trees.

1. If target tree is regular d-ary for some d.

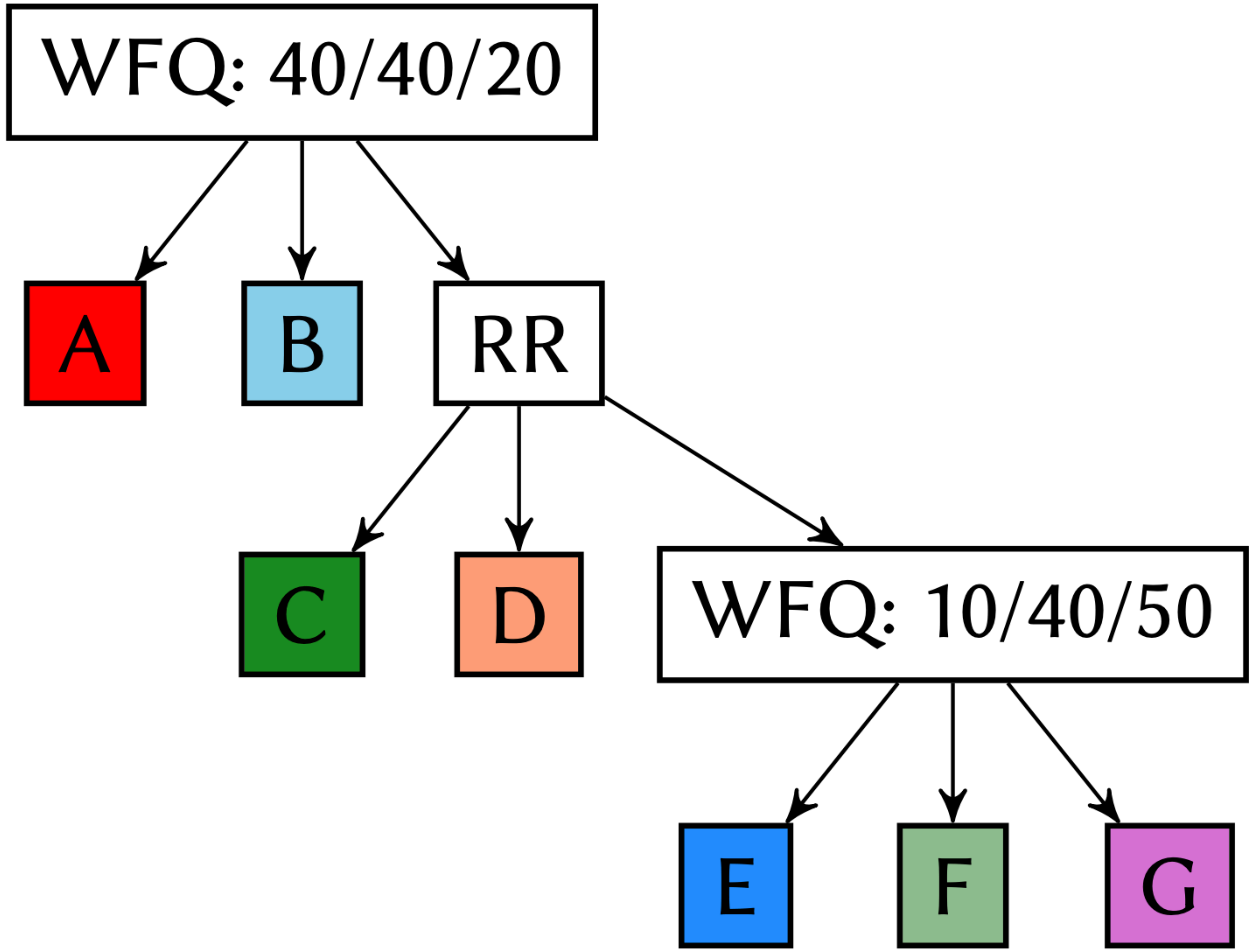
Generating embeddings automatically!

Homomorphic embedding.
Map root to root, leaves to leaves. Respect ancestry.

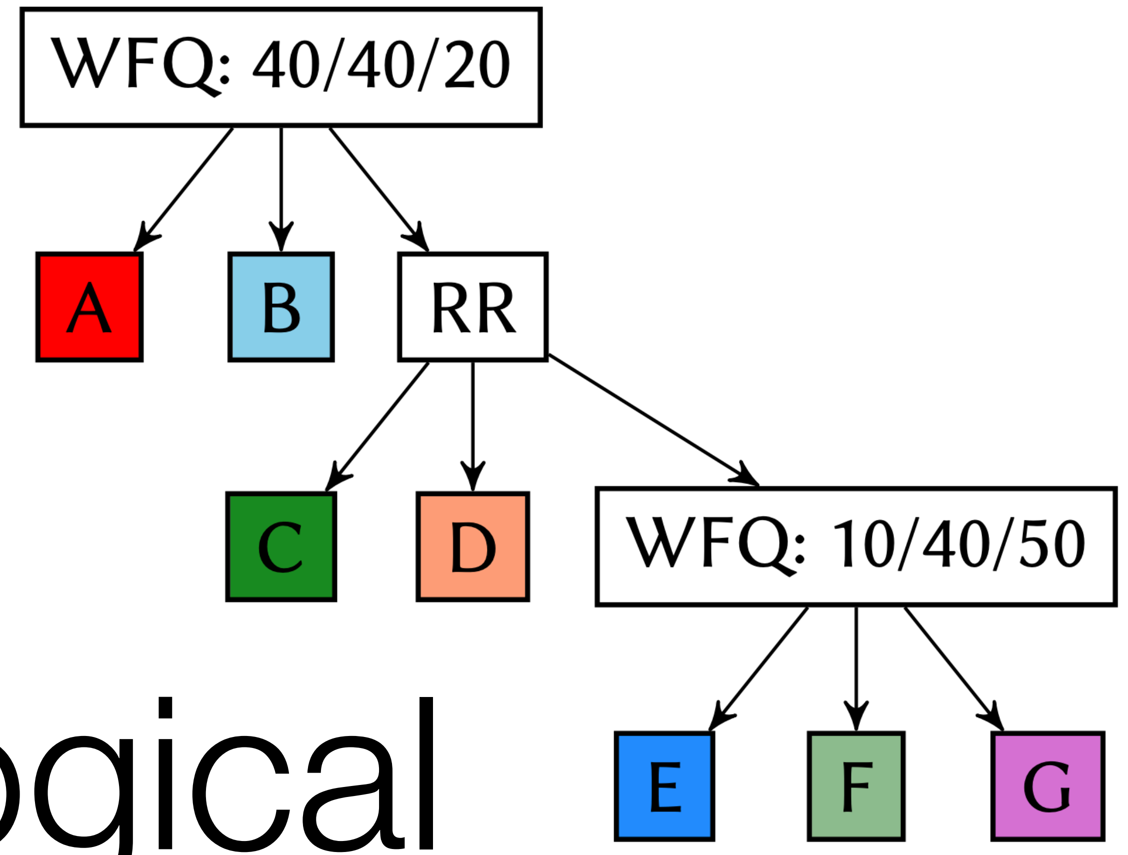
Two new algorithms,
both starting with heterogeneous source trees.

1. If target tree is regular *d*-ary for some *d.*
2. If target tree is itself heterogeneous.

Workflow

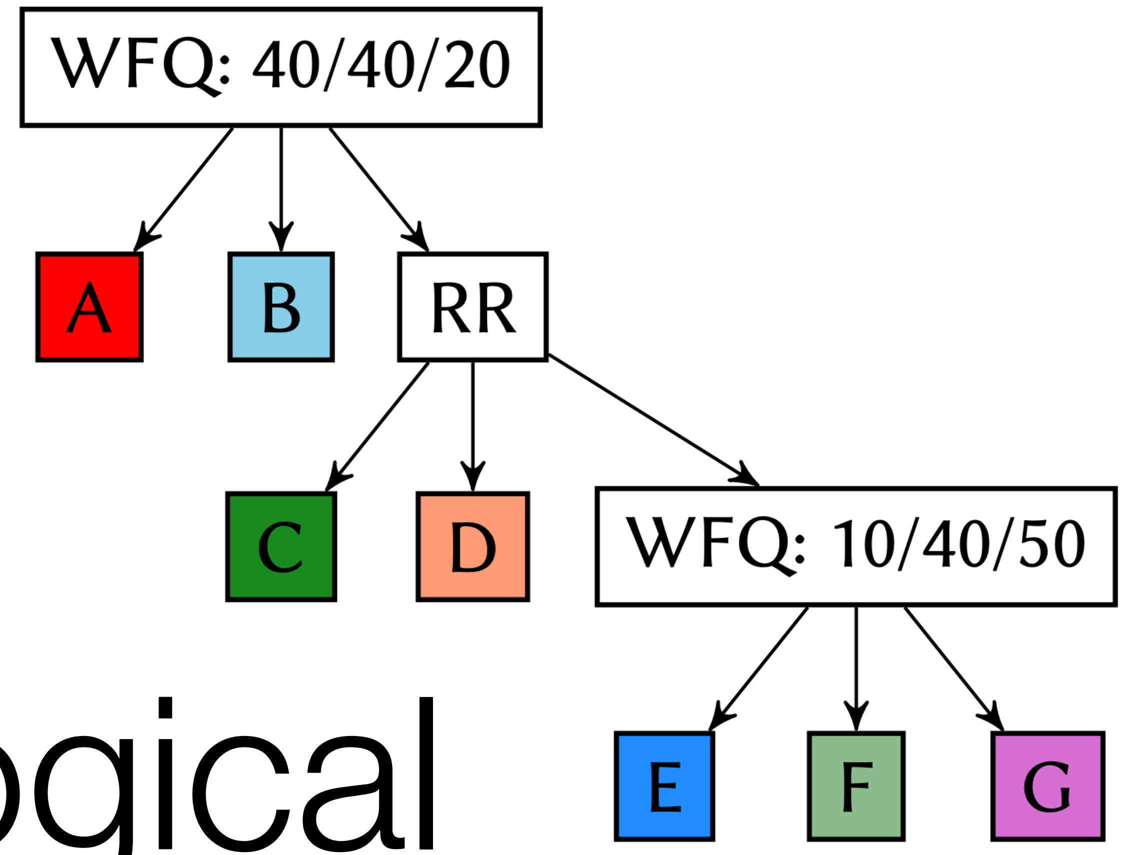


Workflow



logical

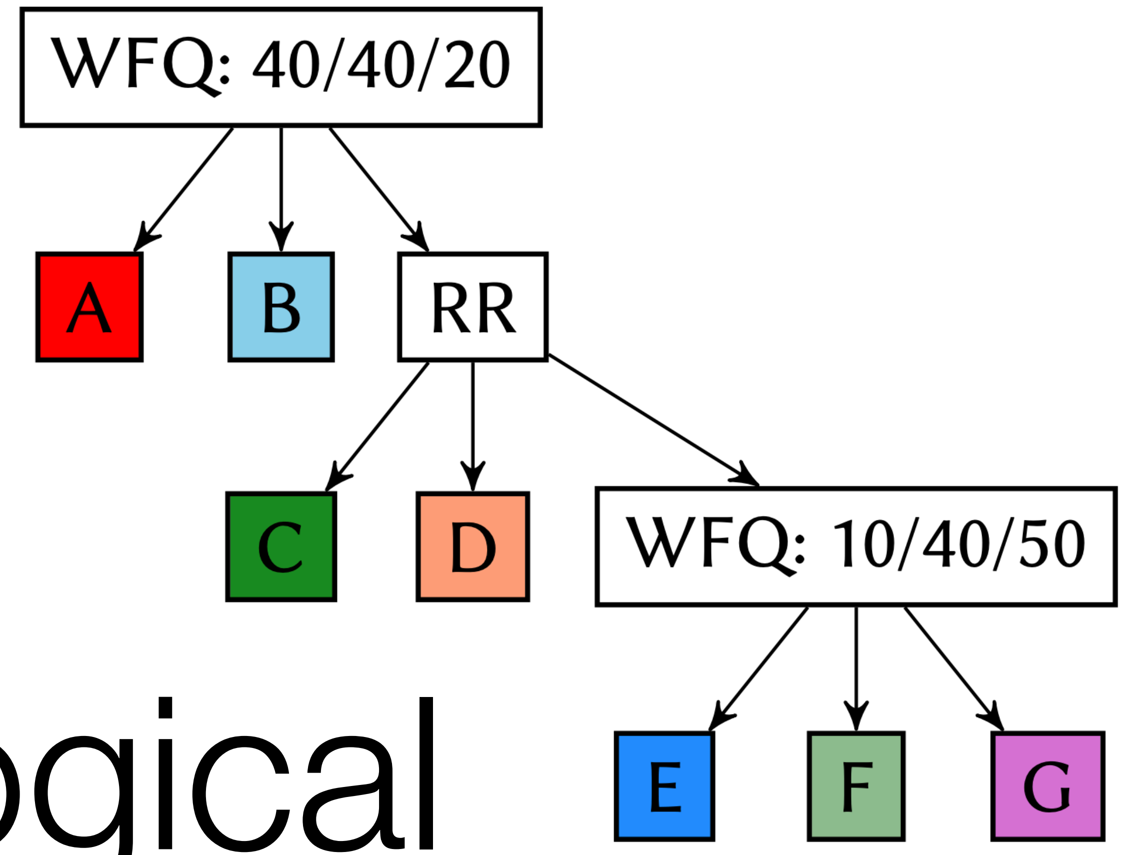
Workflow



logical

But the hardware supports
a regular-branching binary tree.

Workflow

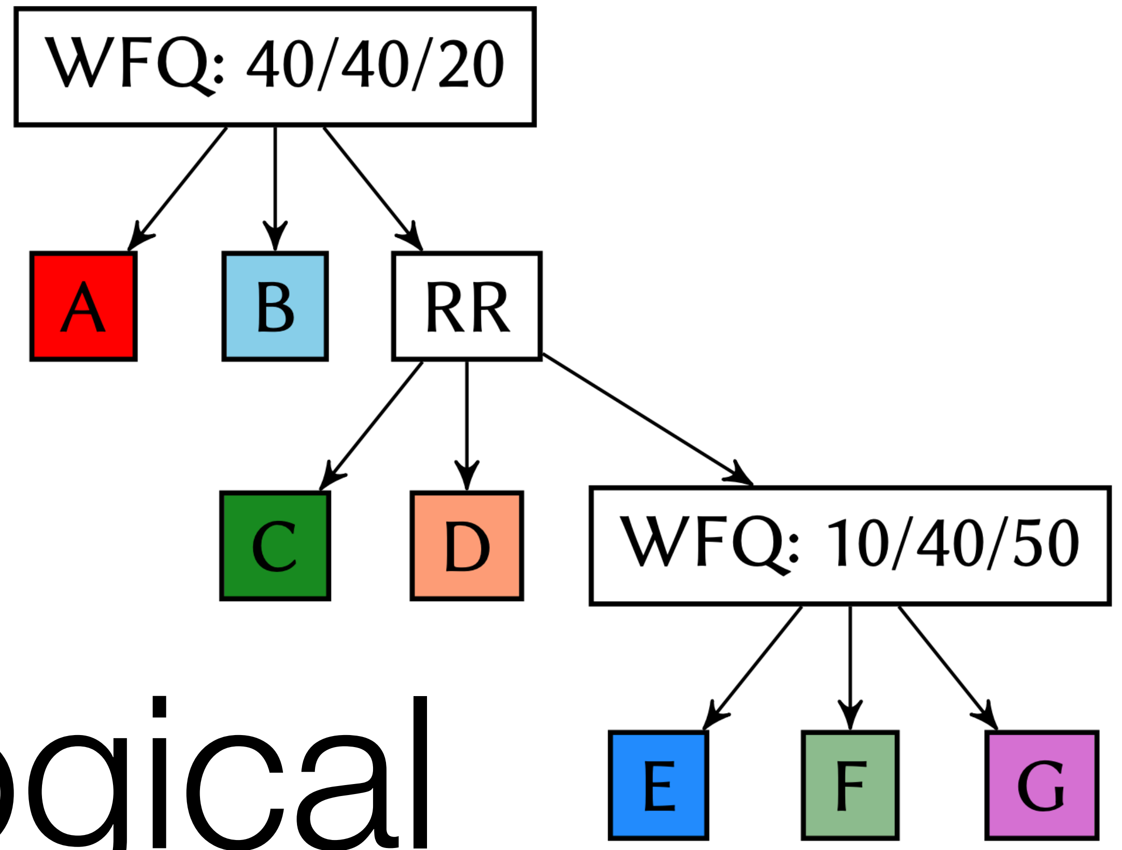


logical

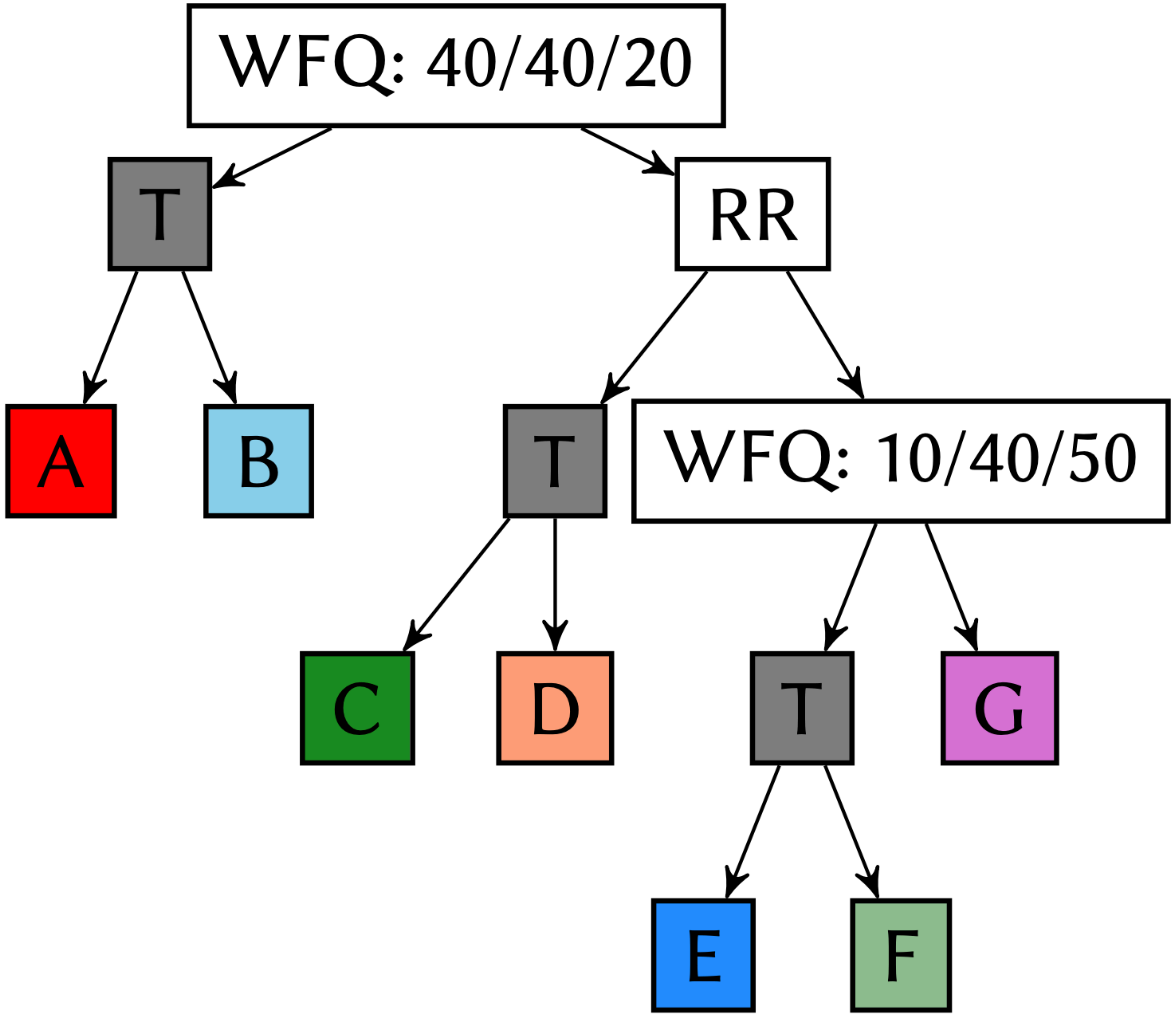
But the hardware supports
a regular-branching binary tree.

Here’s how I’ll use that tree.

Workflow

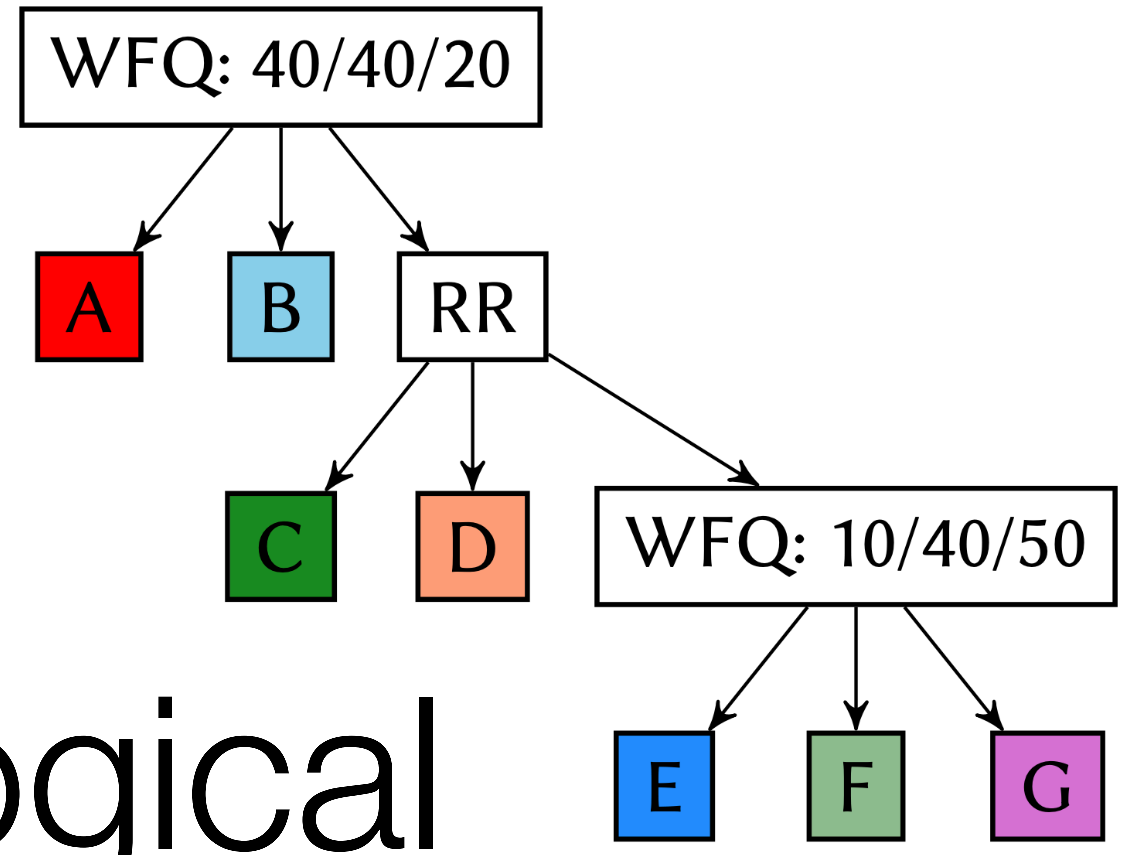


logical

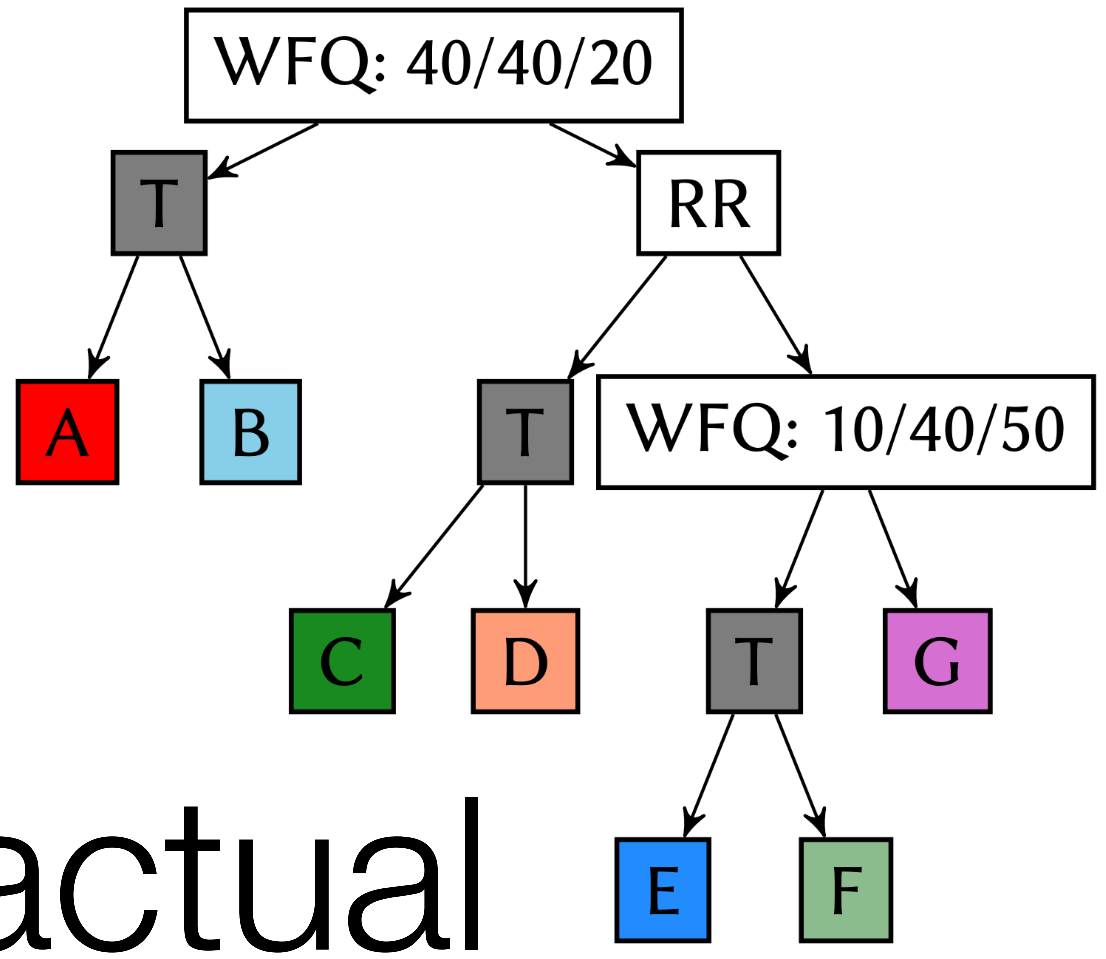


Here's how I'll use that tree.

Workflow

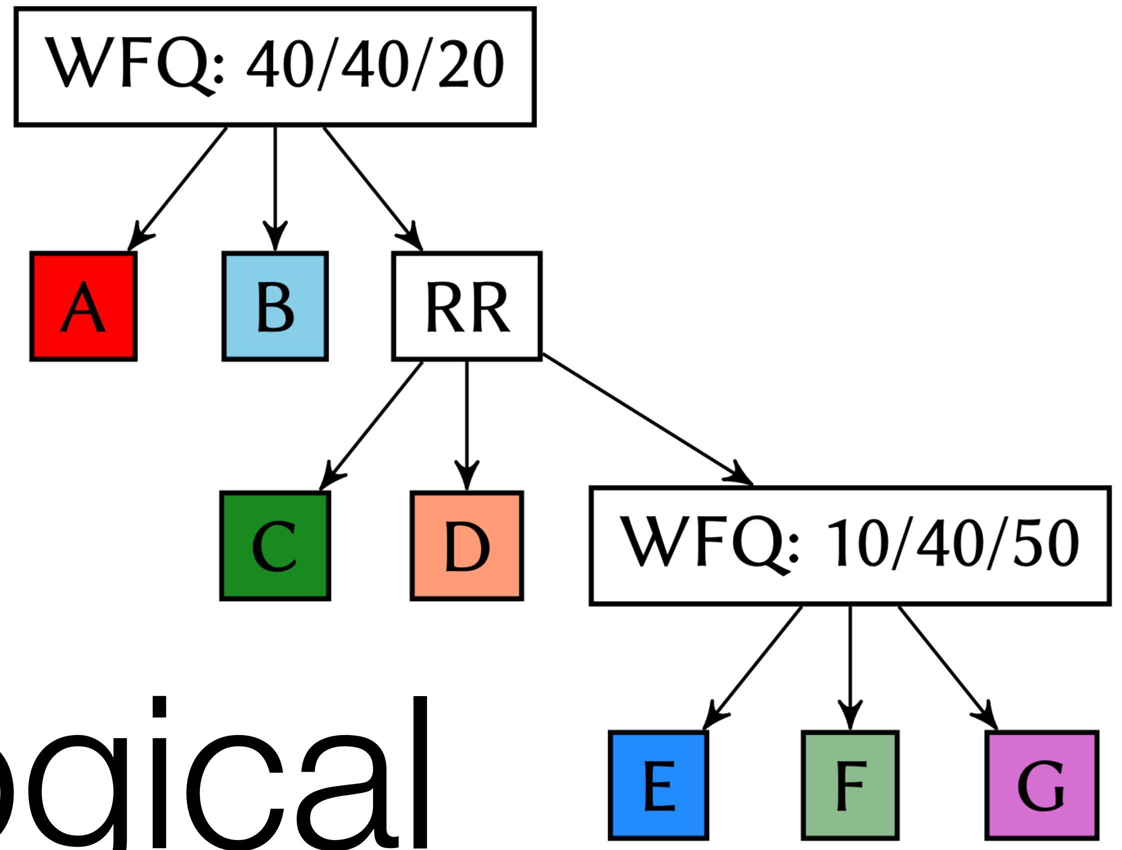


logical

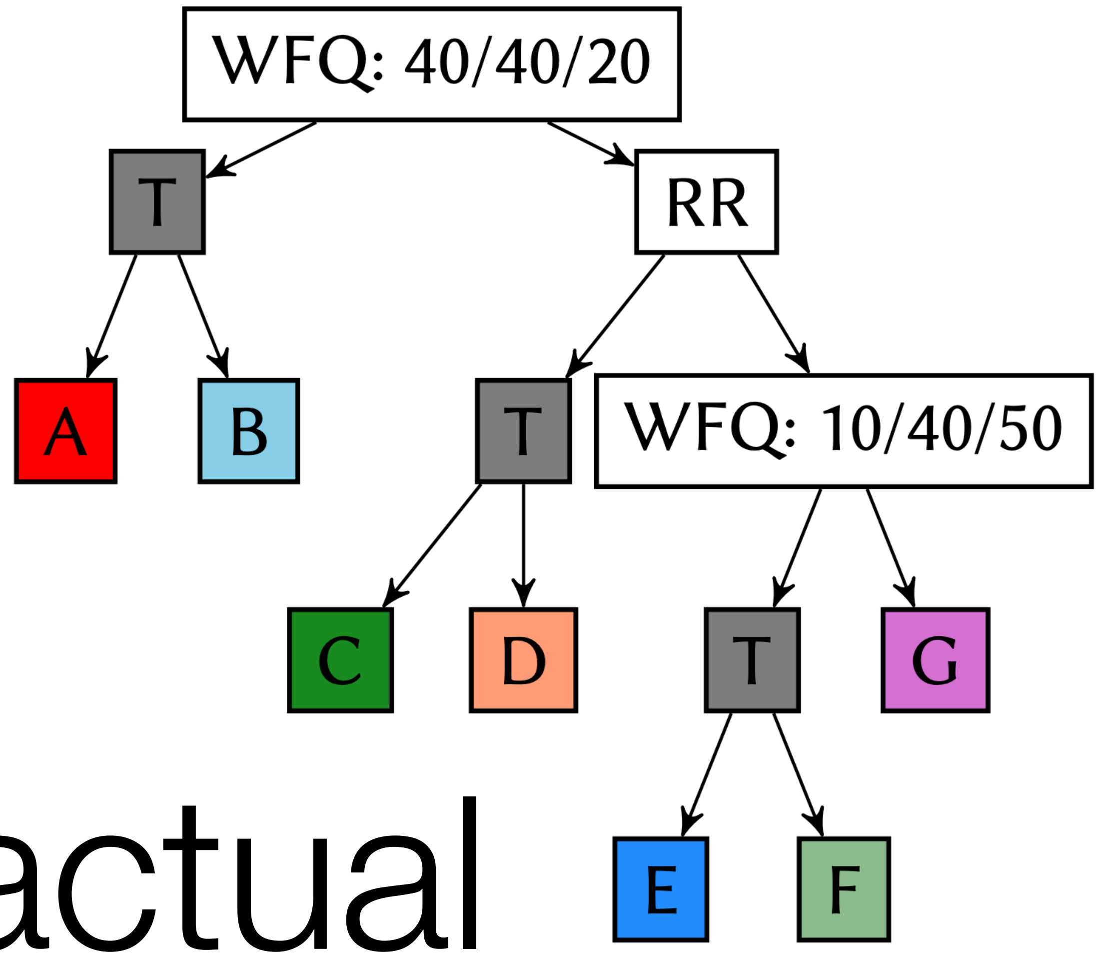


actual

Simulation

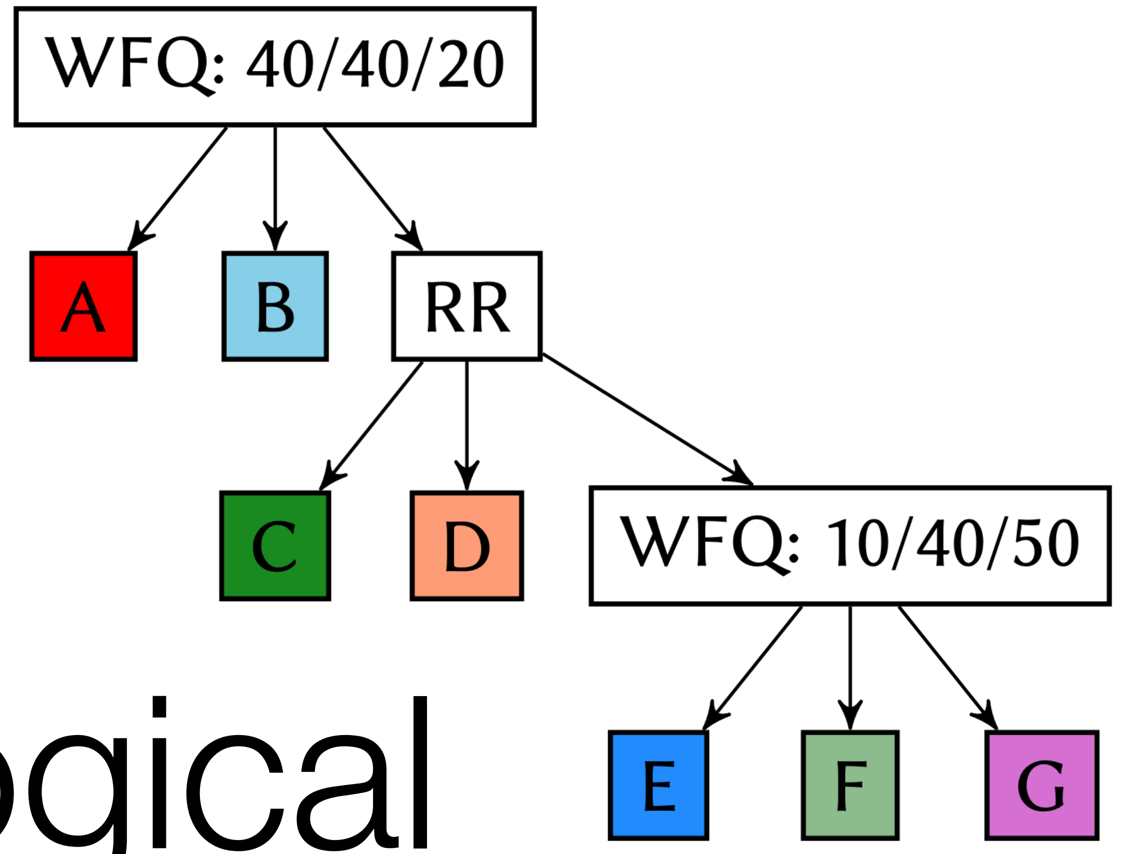


logical

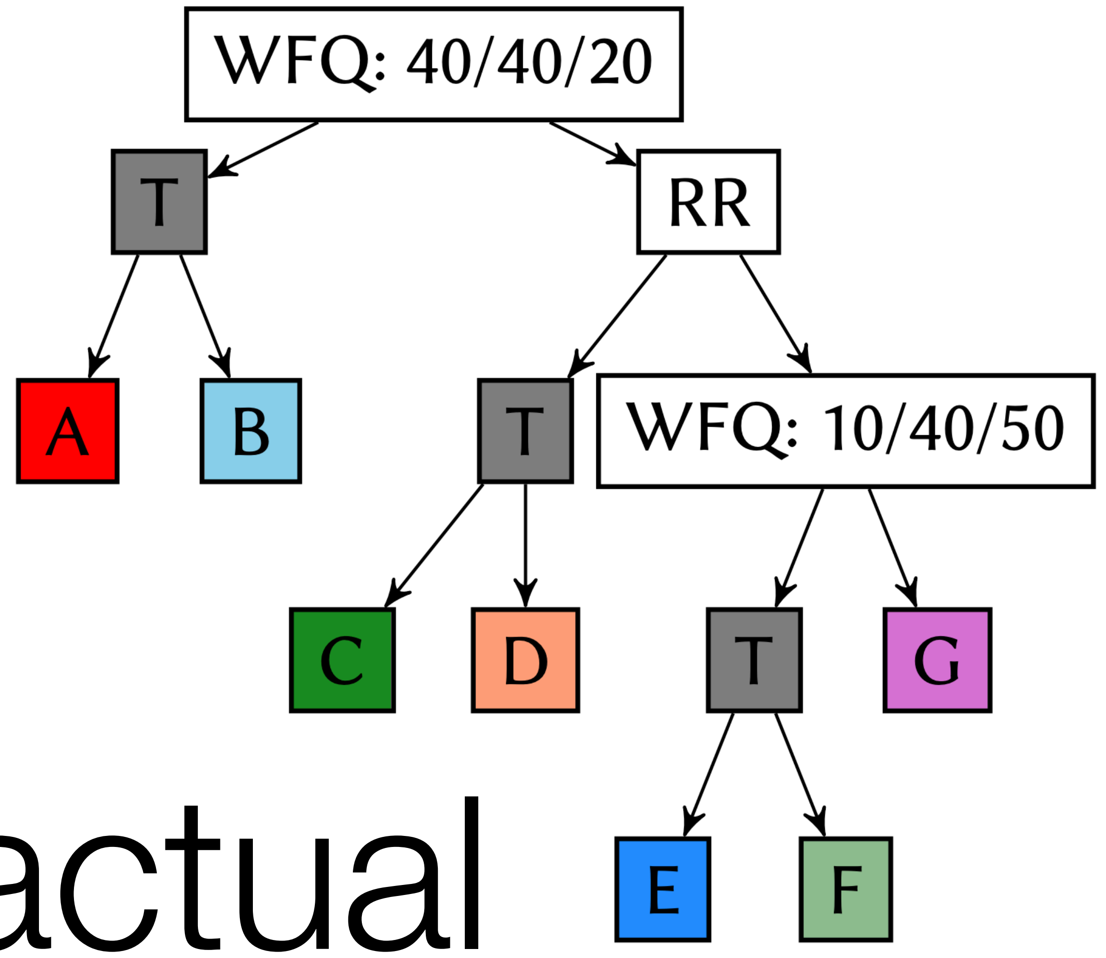


actual

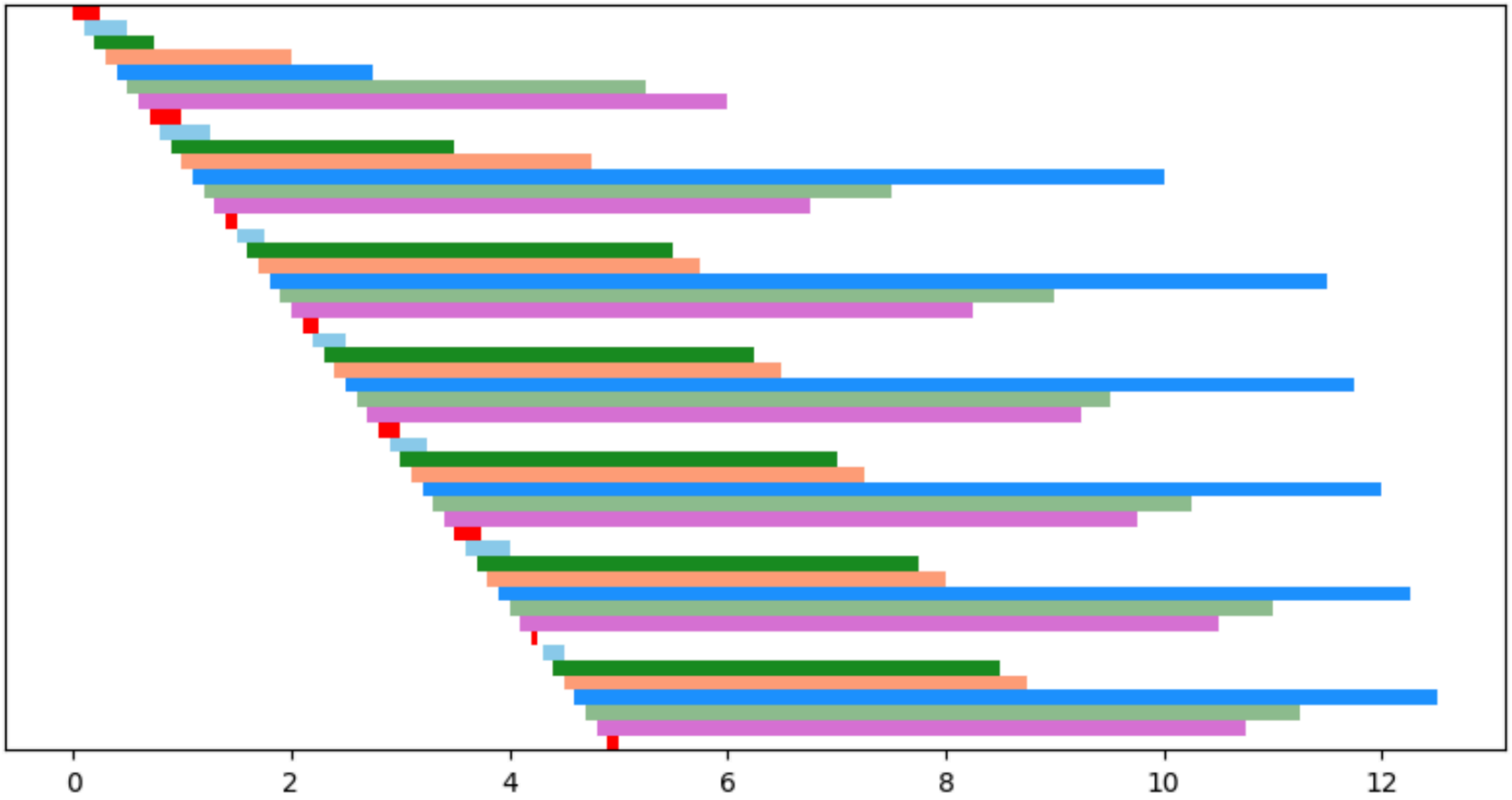
Simulation



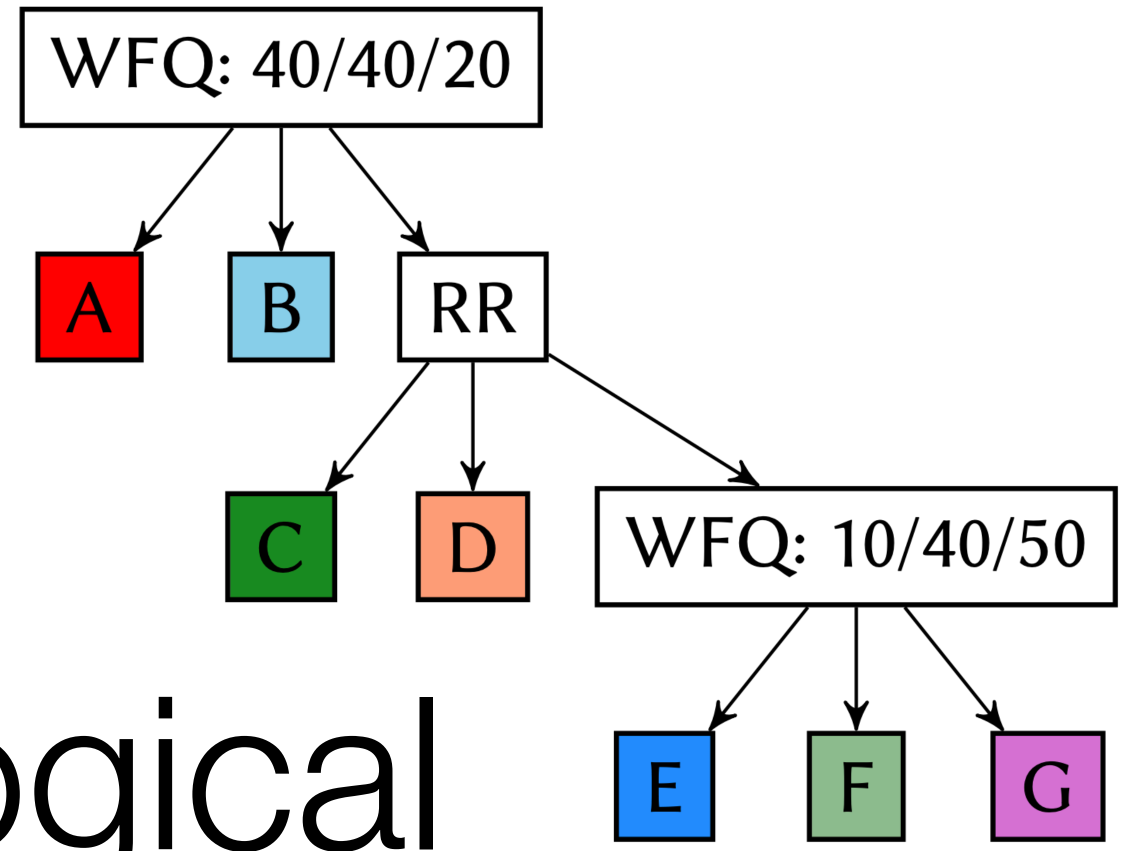
logical



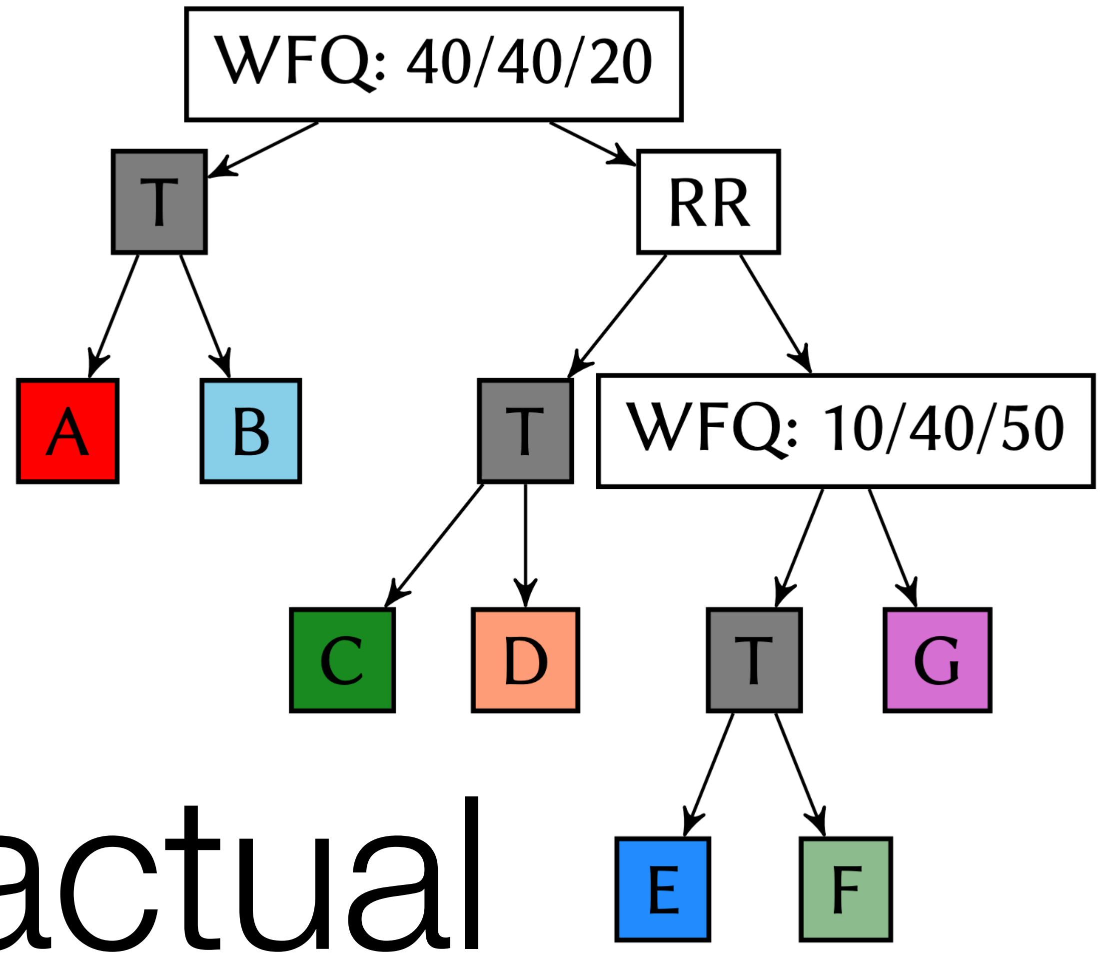
actual



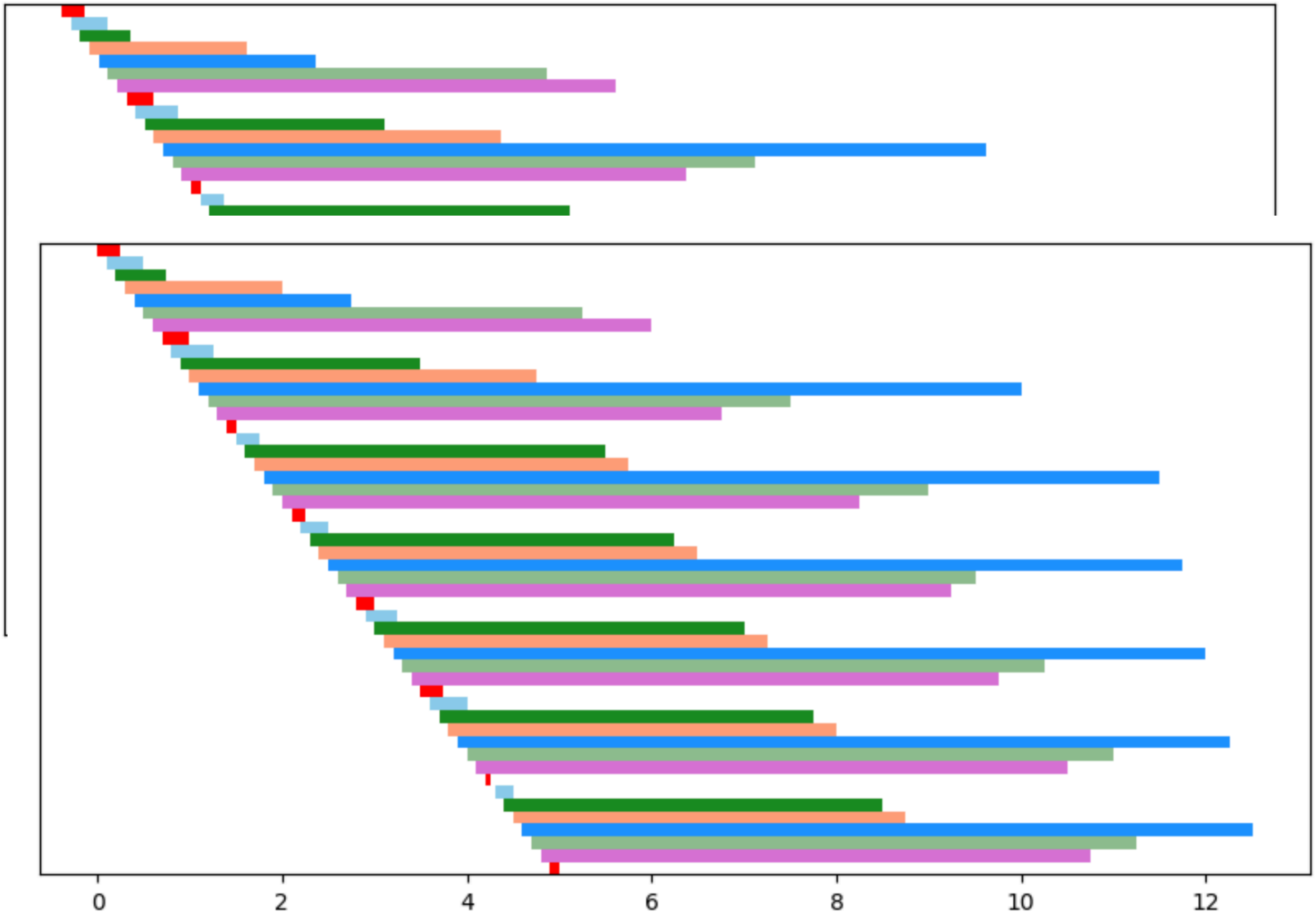
Simulation



logical



actual



Underlying formalism

$$\frac{}{* \in \mathsf{Topo}} \qquad \frac{n \in \mathbb{N} \qquad ts \in \mathsf{Topo}^n}{\mathsf{Node}(ts) \in \mathsf{Topo}}$$

$$\frac{p \in \mathsf{PIFO}(\mathsf{Pkt})}{\mathsf{Leaf}(p) \in \mathsf{PIFOTree}\underbrace{(*)}_{\text{Topo}}} \qquad \frac{n \in \mathbb{N} \qquad ts \in \mathsf{Topo}^n \qquad p \in \mathsf{PIFO}(\{1, \ldots, n\}) \\ \forall 1 \leq i \leq n.\ qs[i] \in \mathsf{PIFOTree}(ts[i])}{\mathsf{Internal}(qs, p) \in \mathsf{PIFOTree}(\underbrace{\mathsf{Node}(ts)}_{\text{Topo}})}$$

$$\frac{r \in \mathsf{Rk}}{r \in \mathsf{Path}\underbrace{(*)}_{\text{Topo}}} \qquad \frac{ts \in \mathsf{Topo}^n \qquad 1 \leq i \leq n \qquad r \in \mathsf{Rk} \qquad pt \in \mathsf{Path}(ts[i])}{(i, r) :: pt \in \mathsf{Path}(\underbrace{\mathsf{Node}(ts)}_{\text{Topo}})}$$

$$\frac{\textsc{push}(p, pkt, r) = p'}{\mathsf{push}(\mathsf{Leaf}(p), pkt, r) = \underbrace{\mathsf{Leaf}(p')}_{\text{PIFOTree}}} \qquad \frac{\mathsf{push}(qs[i], pkt, pt) = q' \qquad \textsc{push}(p, i, r) = p'}{\mathsf{push}(\mathsf{Internal}(qs, p), pkt, \underbrace{(i, r) :: pt)}_{\text{Path}} = \underbrace{\mathsf{Internal}(qs[i/q'], p')}_{\text{PIFOTree}}}$$

A general way to deploy PIFO trees.

A general way to deploy PIFO trees.

Let the hardware support some tree.

A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.

A general way to deploy PIFO trees.

Let the human
program against some tree.

Let the hardware
support some tree.

A general way to deploy PIFO trees.

Let the human program against some tree.

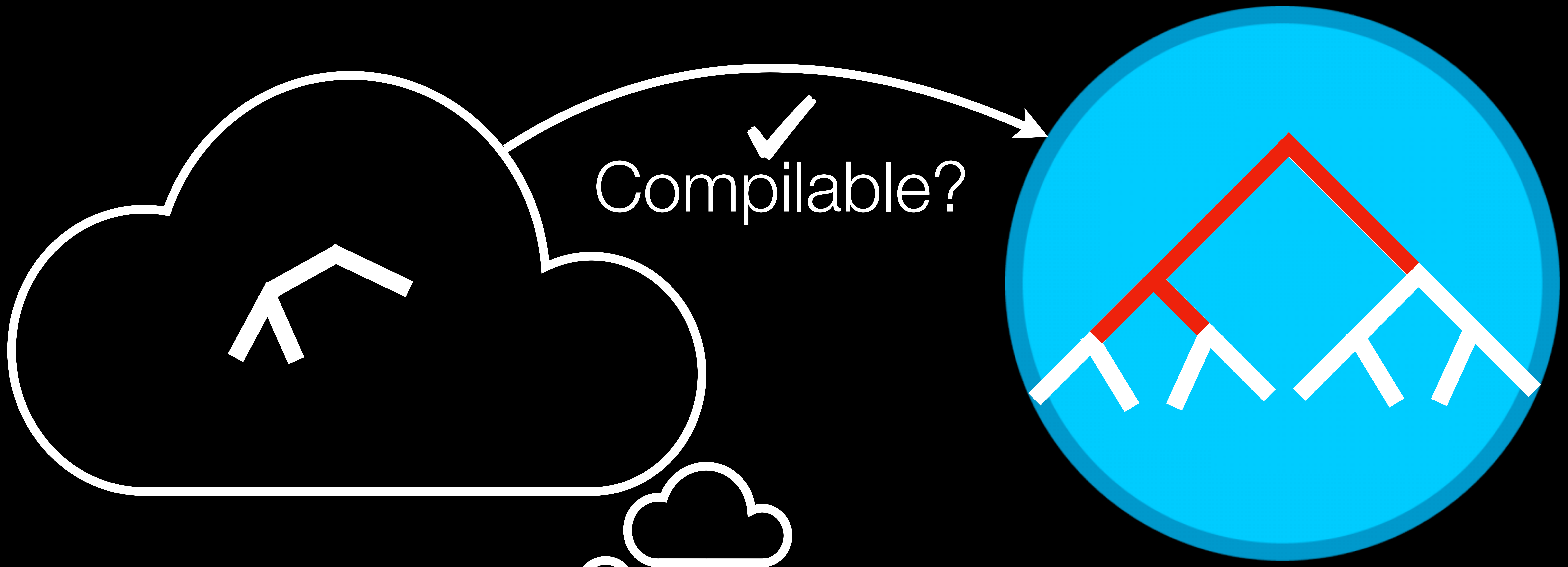
Let the hardware support some tree.

A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.

A general way to deploy PIFO trees.



Let the human program against some tree.

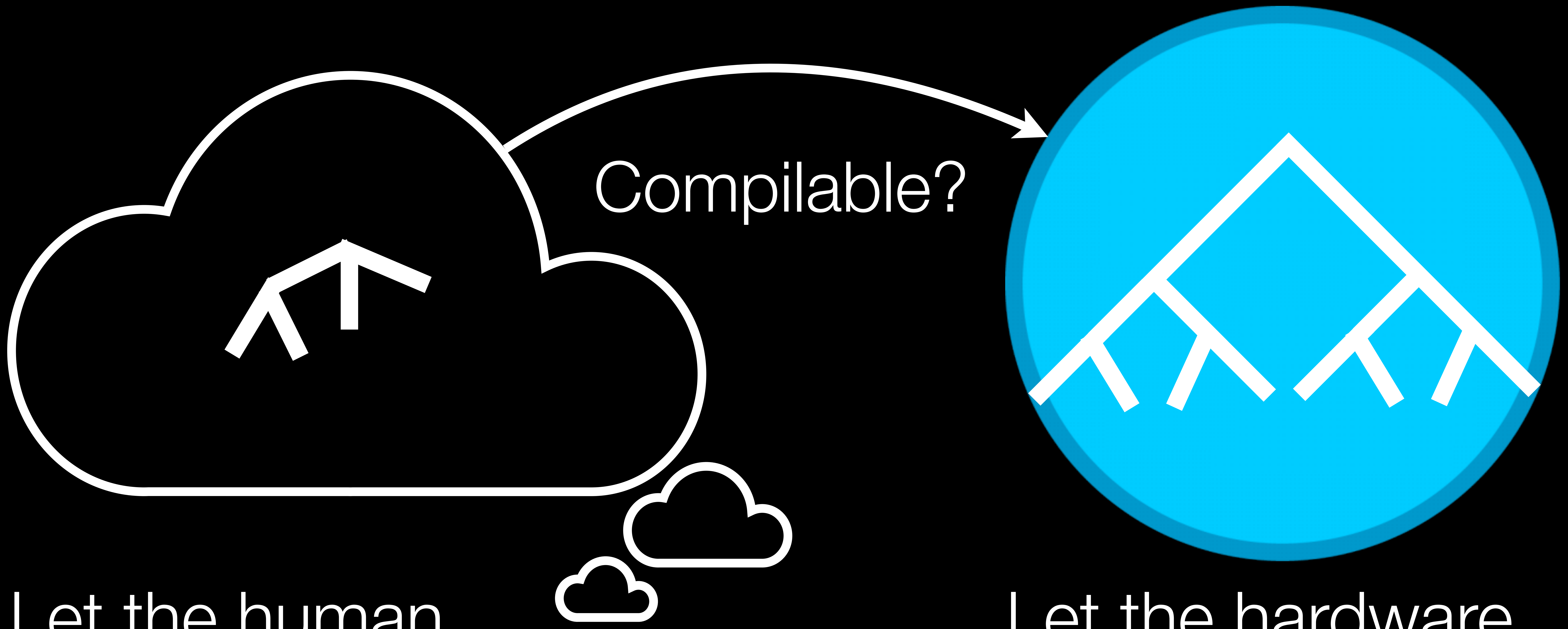
Let the hardware support some tree.

A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.

A general way to deploy PIFO trees.



Let the human program against some tree.

Let the hardware support some tree.

A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.

A general way to deploy PIFO trees.

Let the human program against some tree.

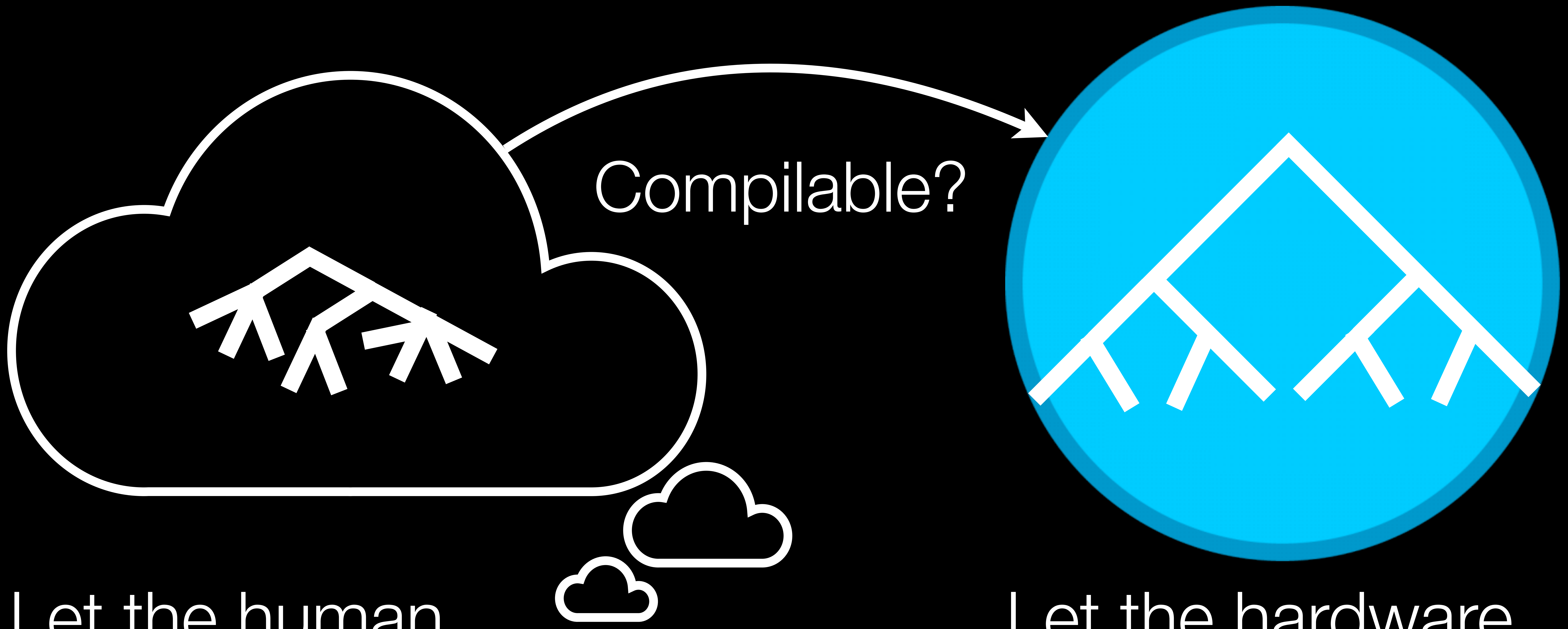
Let the hardware support some tree.

A general way to deploy PIFO trees.

Let the human
program against some tree.

Let the hardware
support some tree.

A general way to deploy PIFO trees.



Let the human program against some tree.

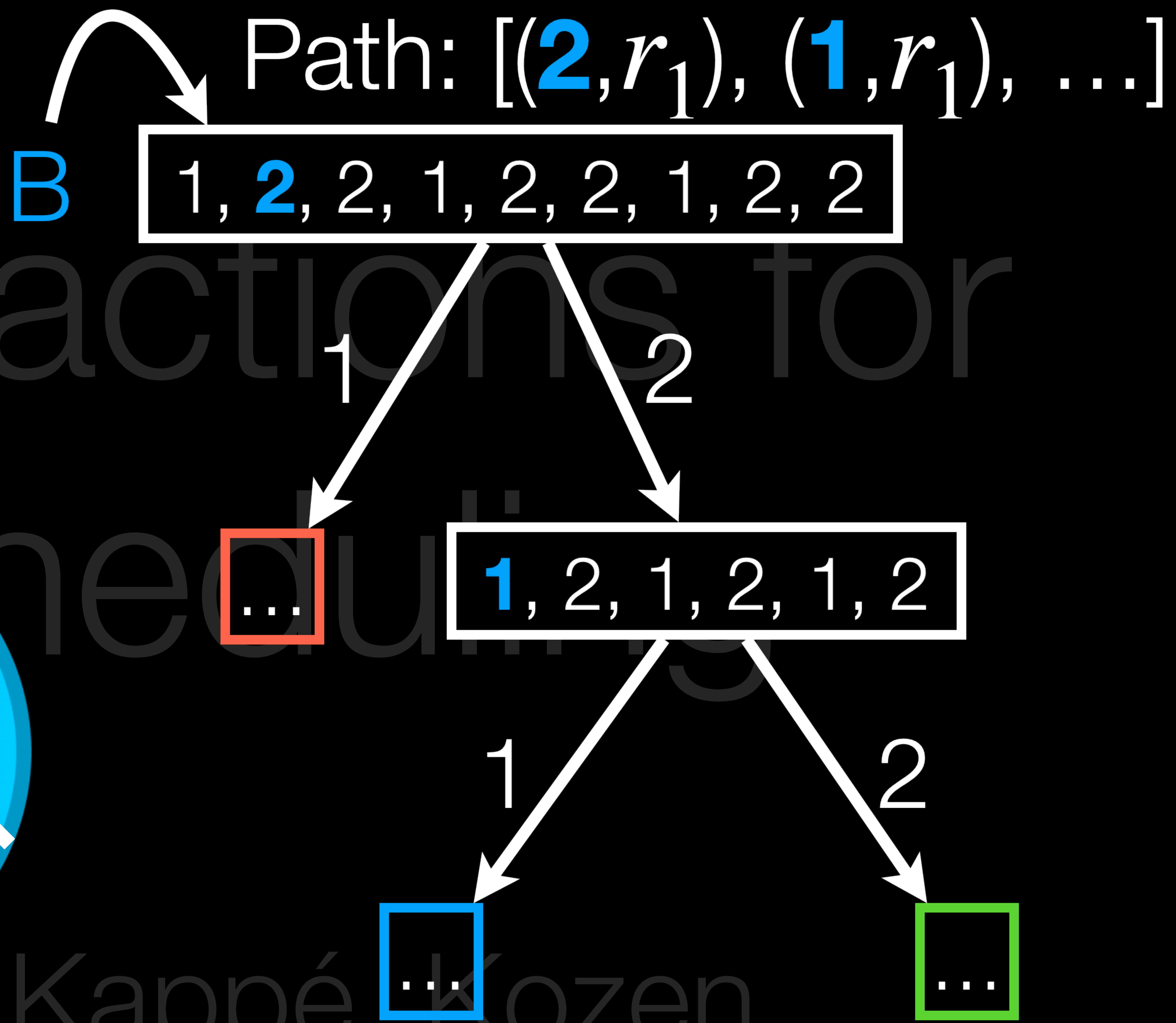
Let the hardware support some tree.

A general way to deploy PIFO trees.

Let the human program against some tree.

Let the hardware support some tree.

tree shape ⟷ language expressivity



Formal Abstractions for Packet Scheduling

Mohan, Liu, Foster, Kappé, Kozen

cs.cornell.edu/~amohan