https://hao-ai-lab.github.io/dsc204a-f25/

# DSC 204A: Scalable Data Systems
# Fall 2025


Staff

Instructor: Hao Zhang

TAs: Mingjia Huo, Yuxuan Zhang

@haozhangml @haoailab

haozhang@ucsd.edu

# Large Message

Communication Model: $\alpha + n\,\beta, \beta = \frac{1}{B}$

- The second term dominates – we want to minimize the second term
  - We want to utilize the bandwidth as much as possible

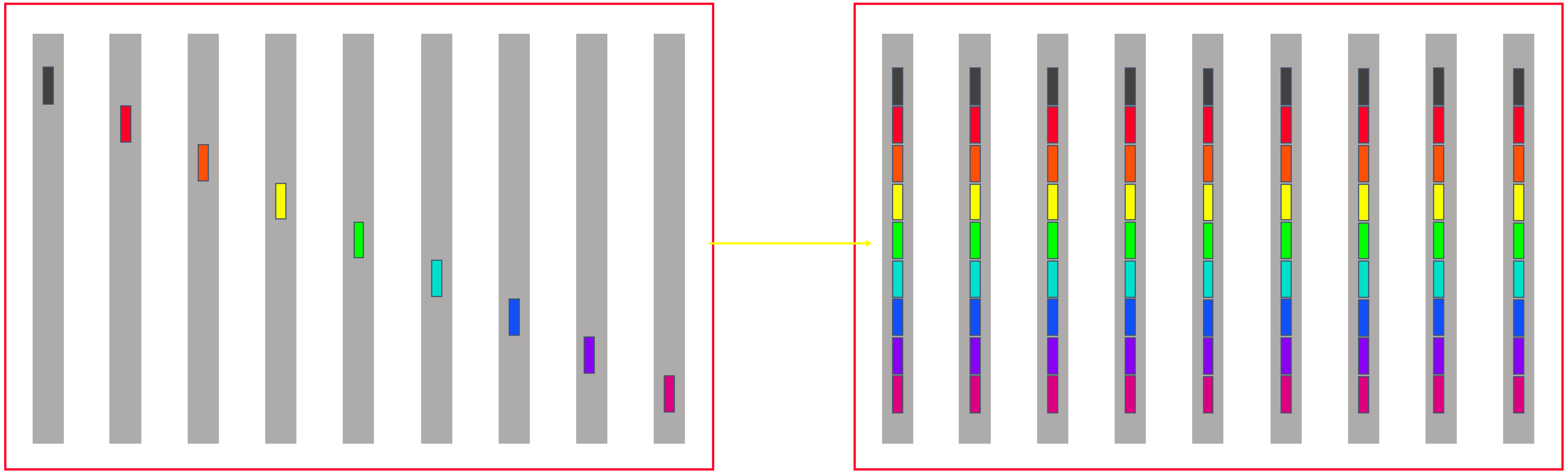
# General principles

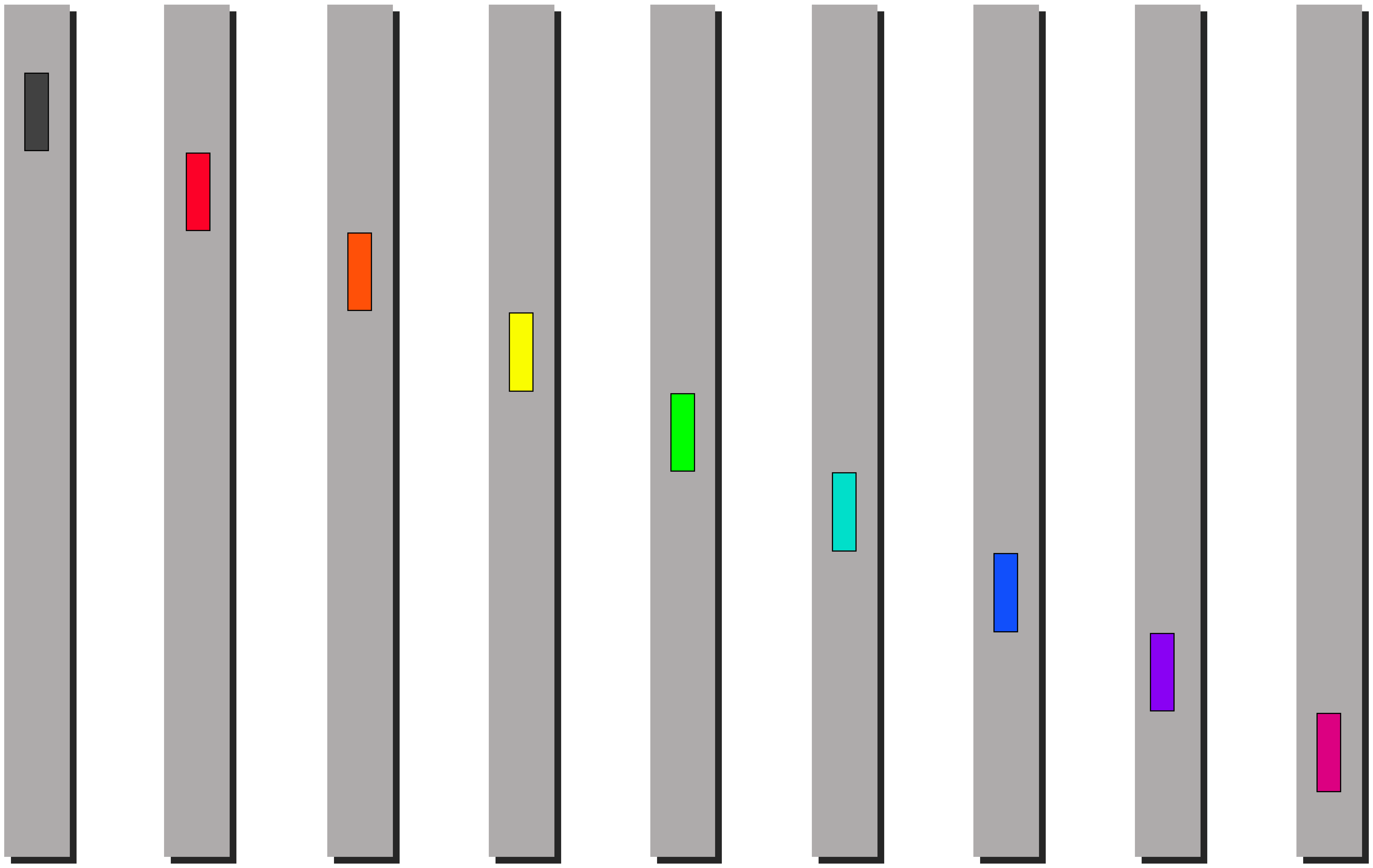
Ring algorithm has the following advantages

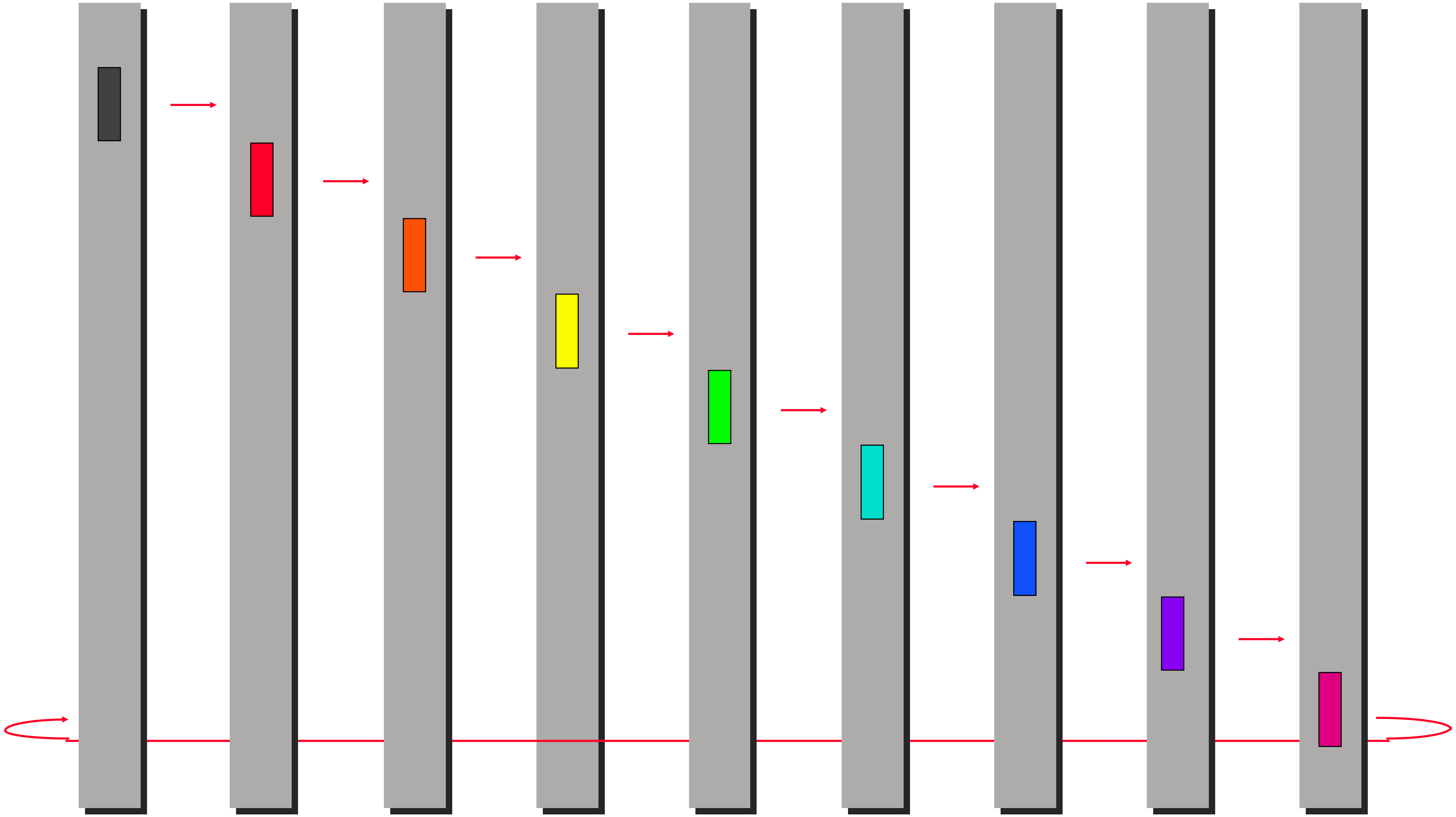
- Fully utilize the bandwidth (bandwidth optimal)
- implementation for arbitrary numbers of node

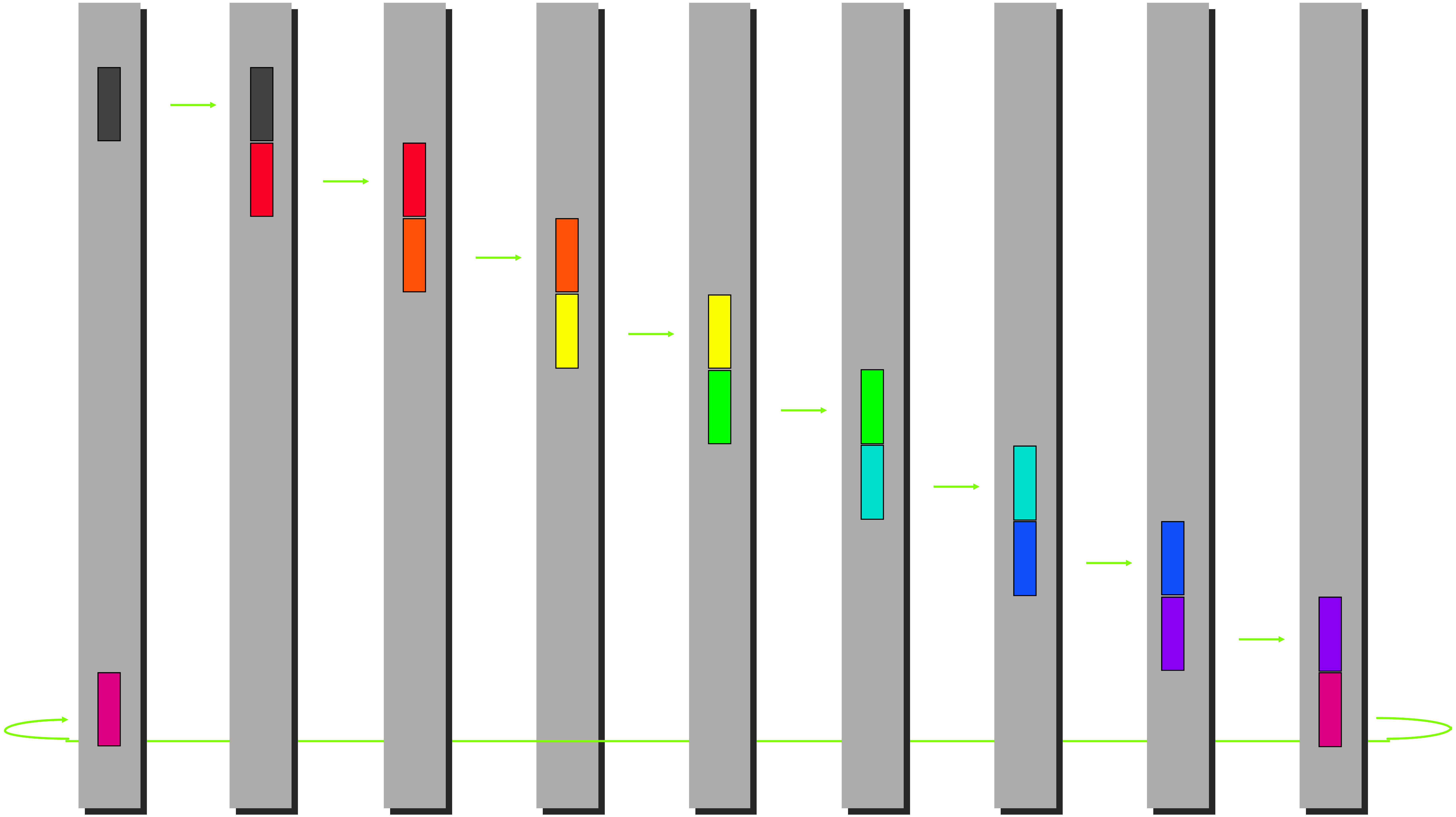
# Allgather

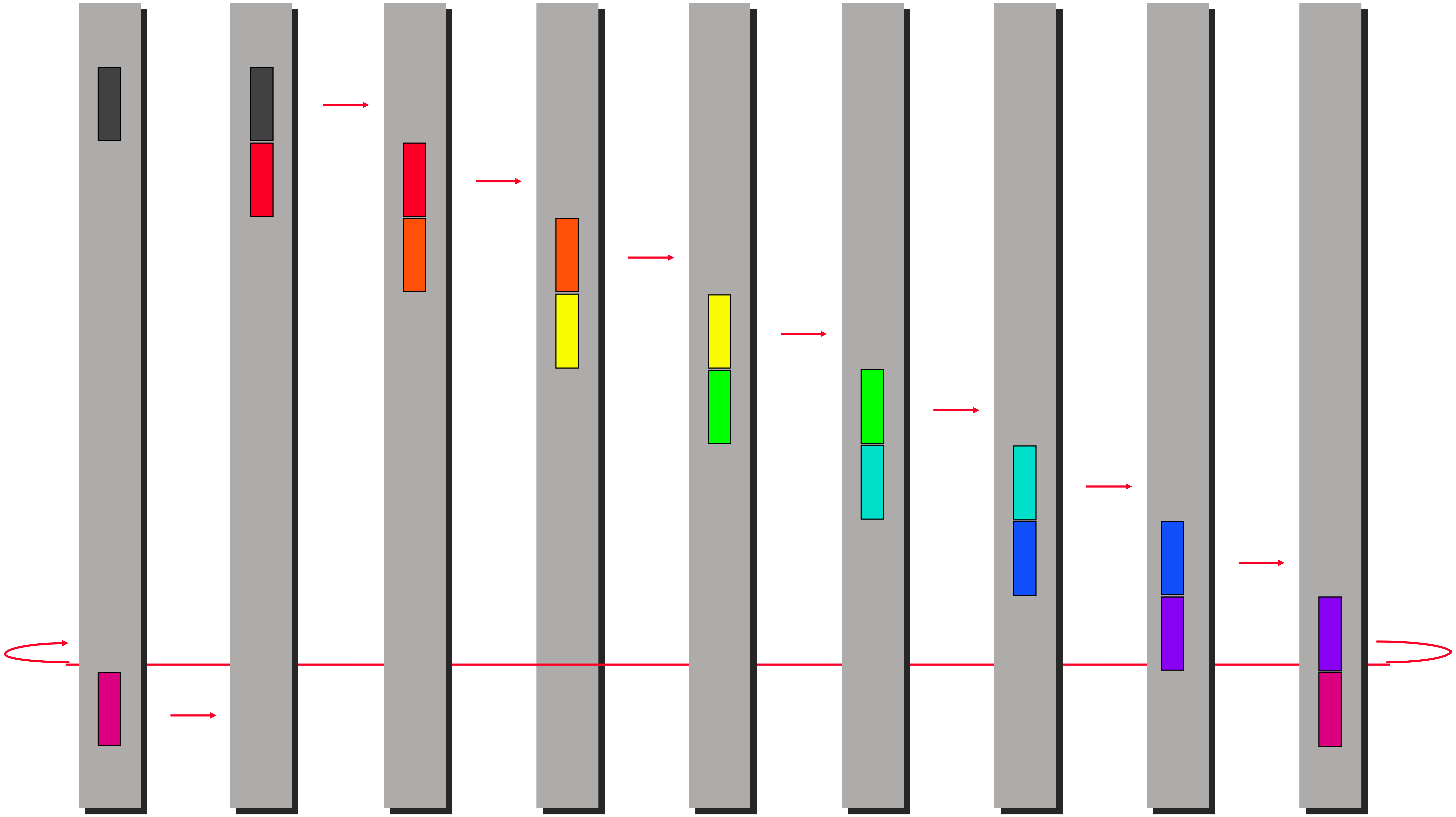
Before | After

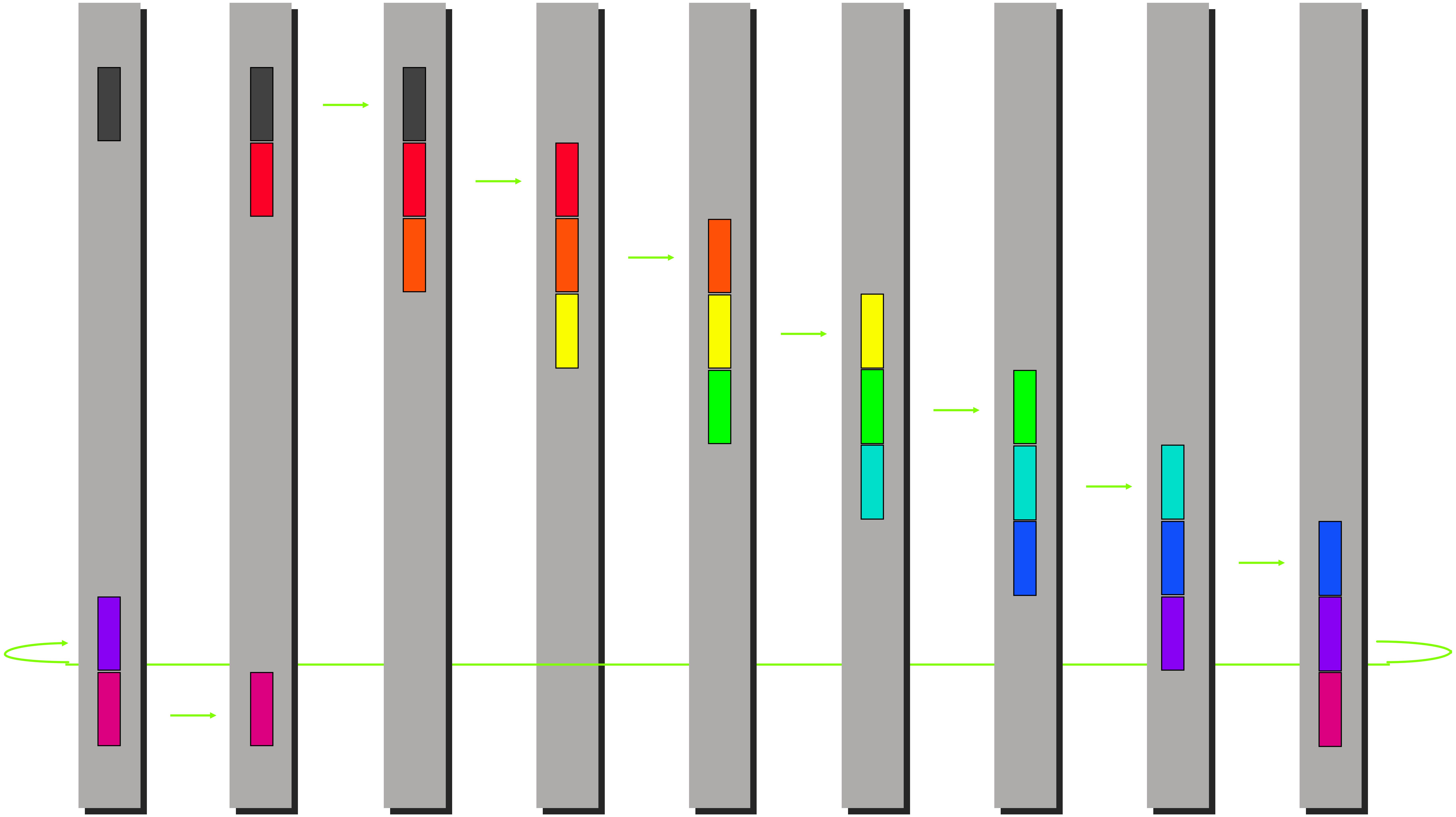


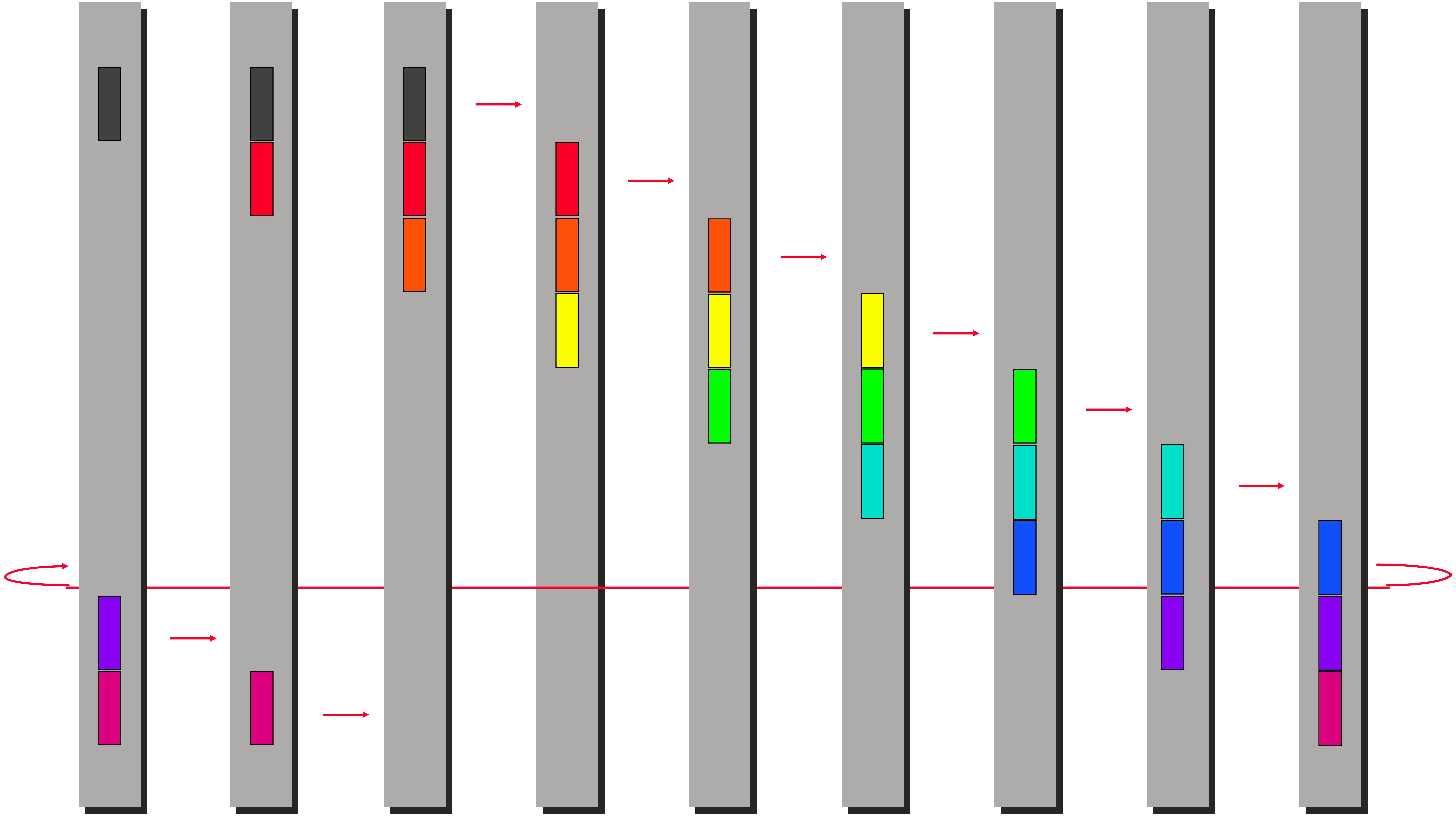


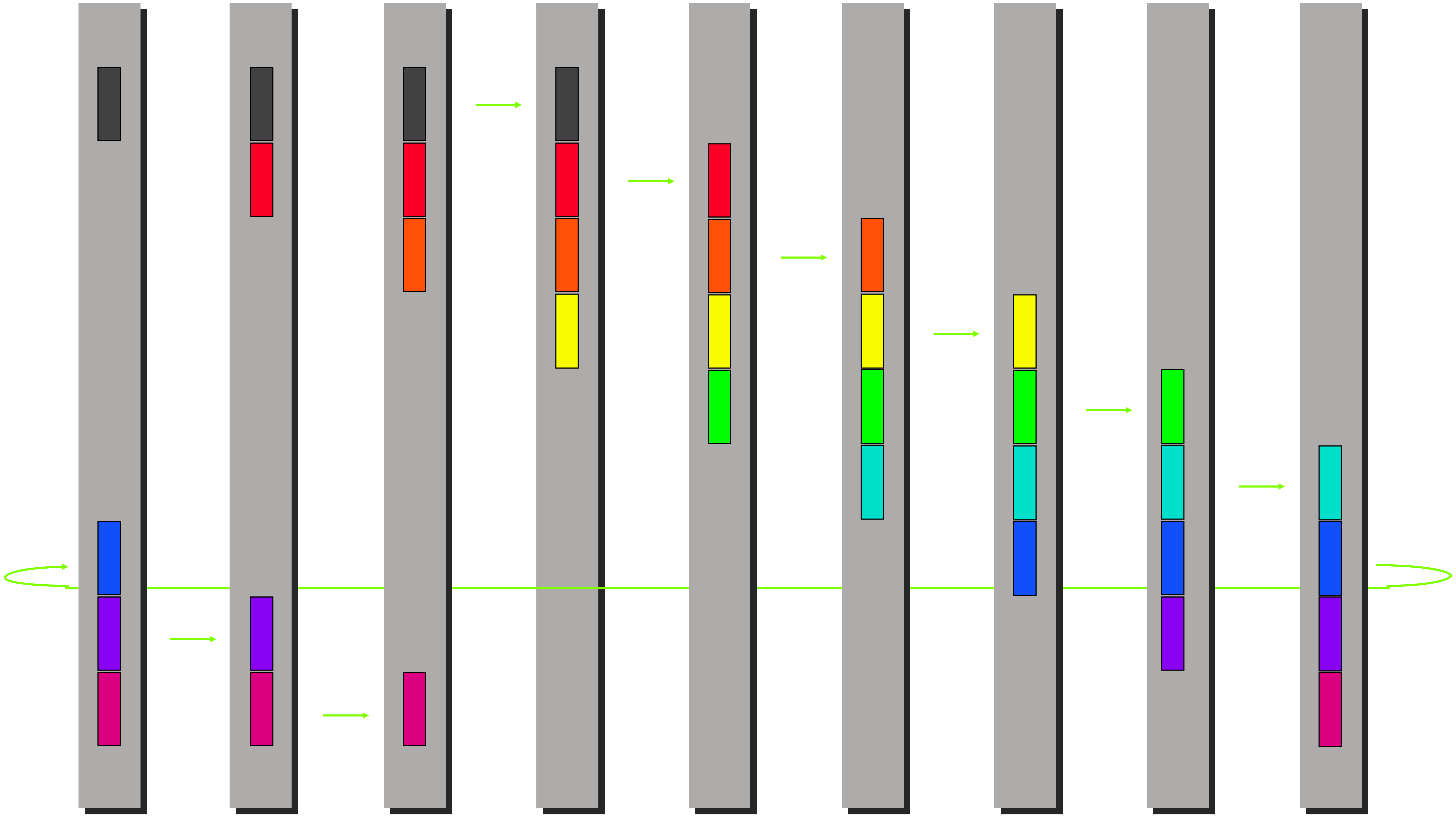


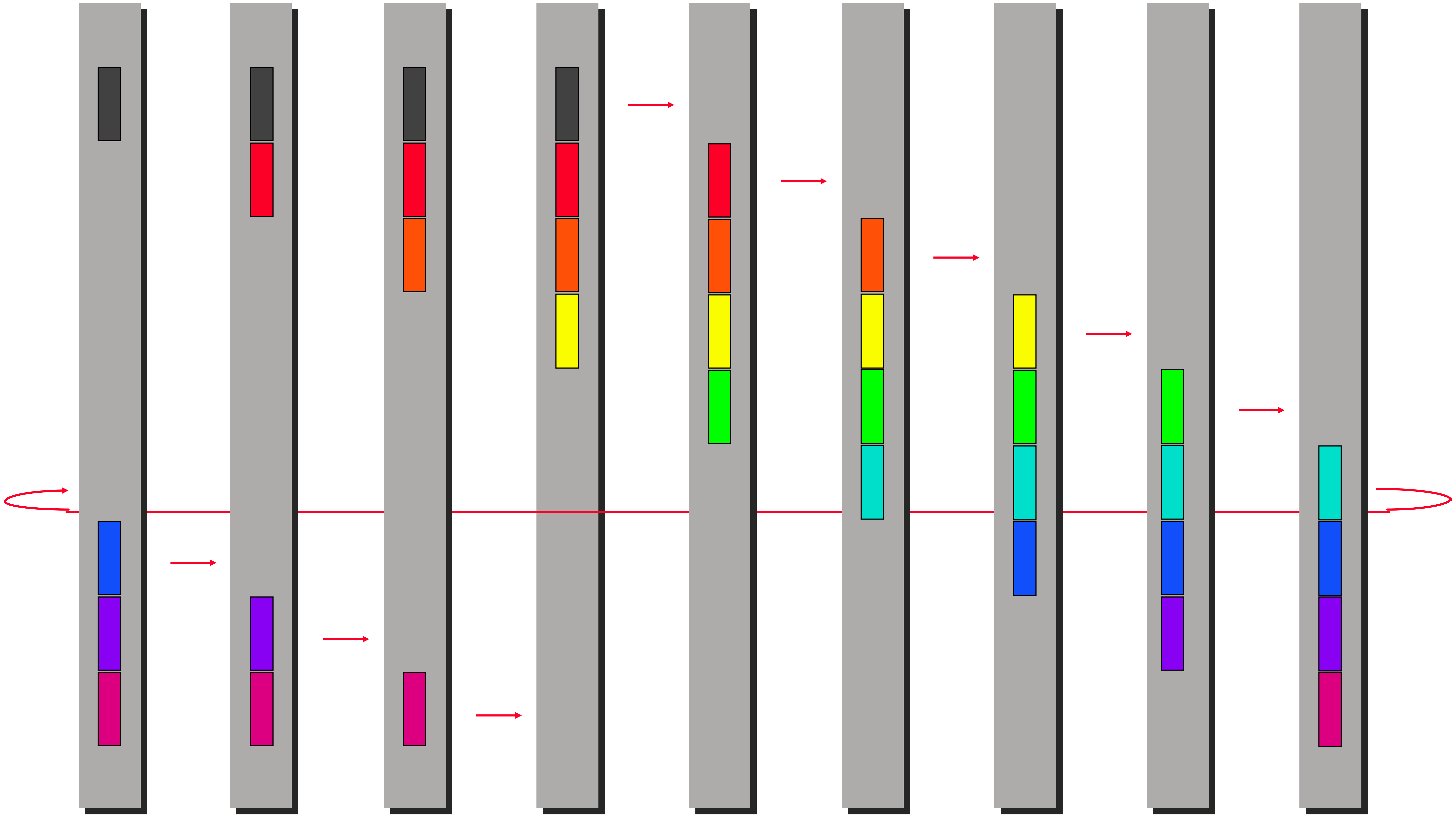


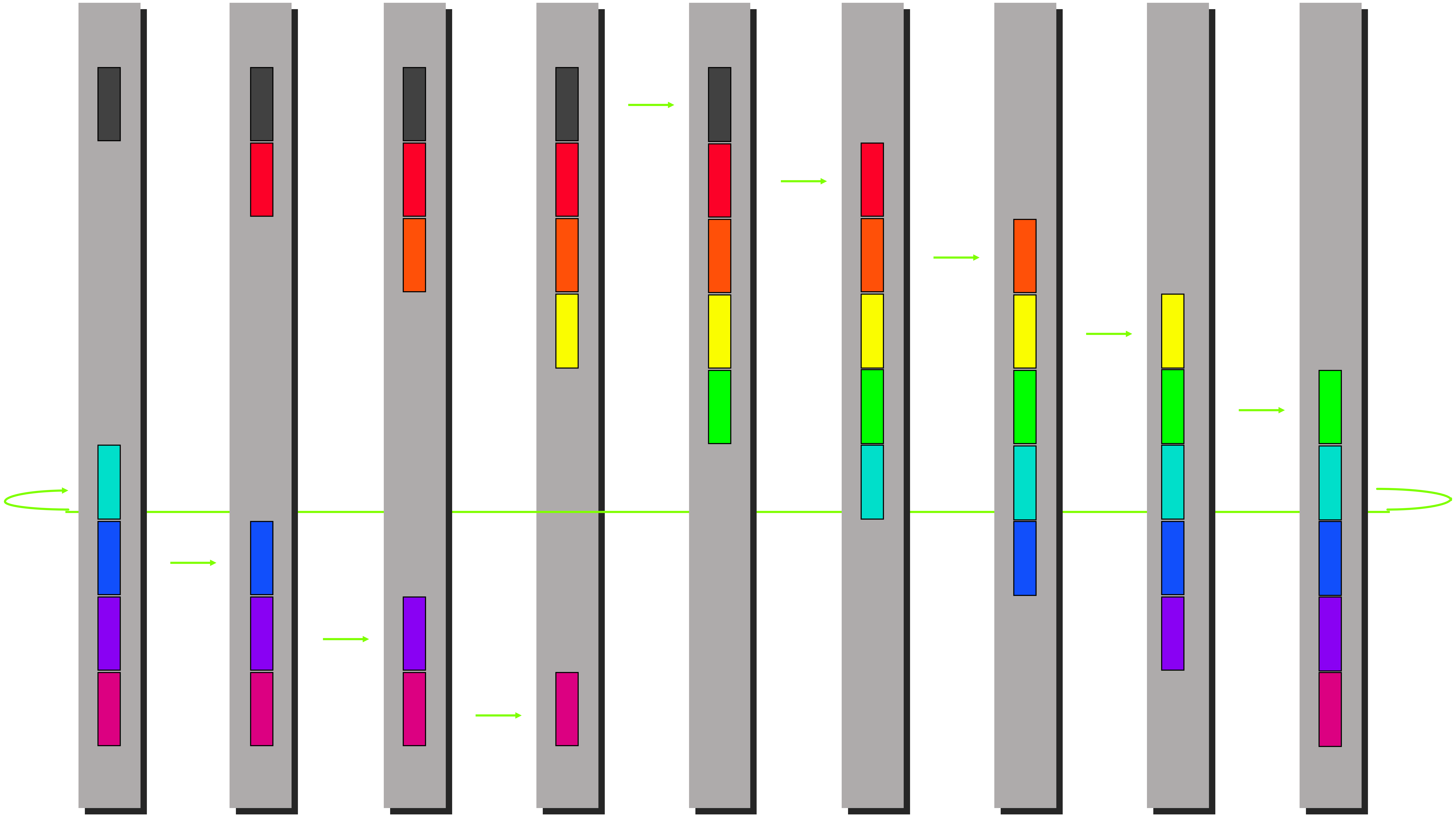


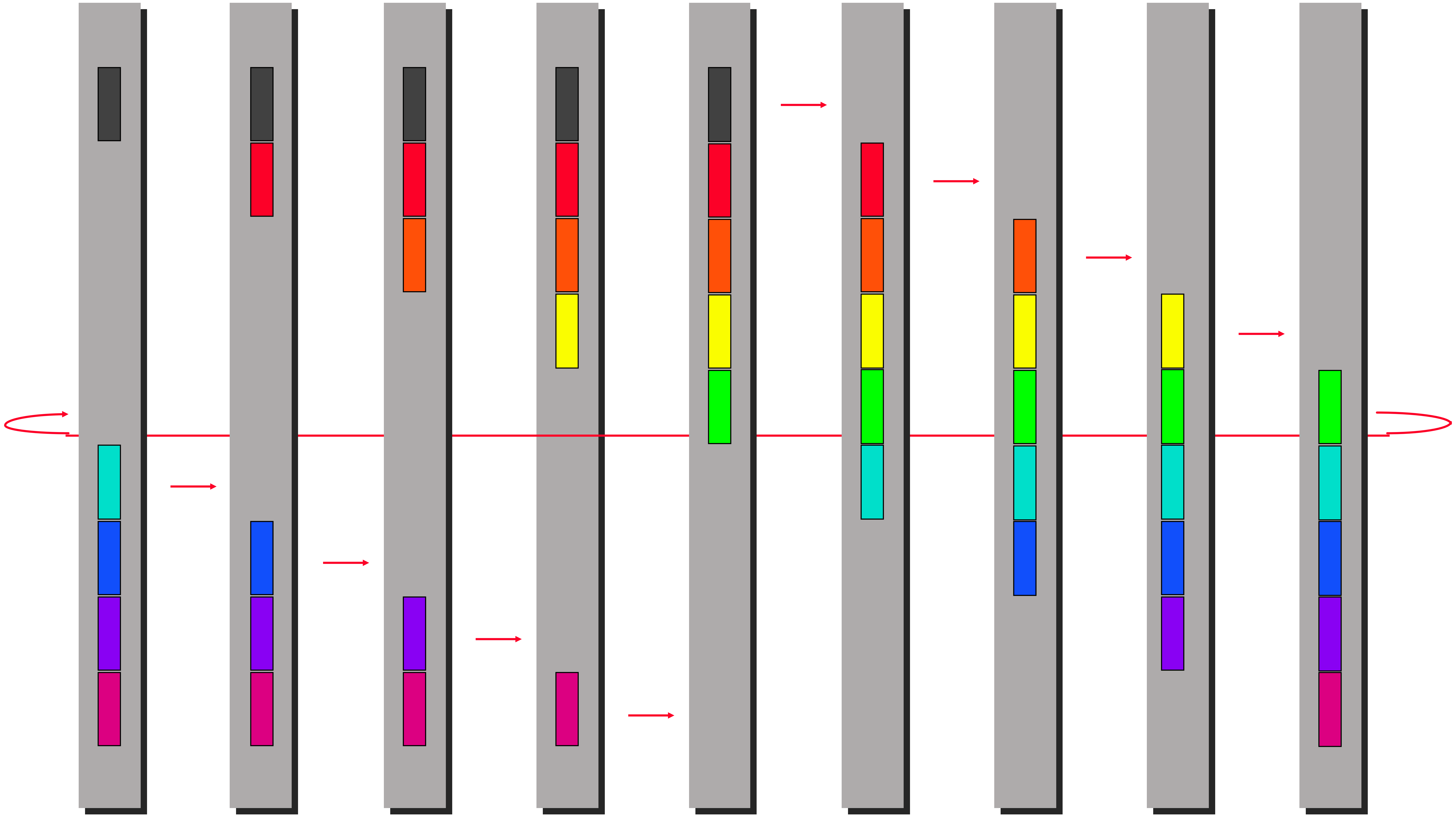


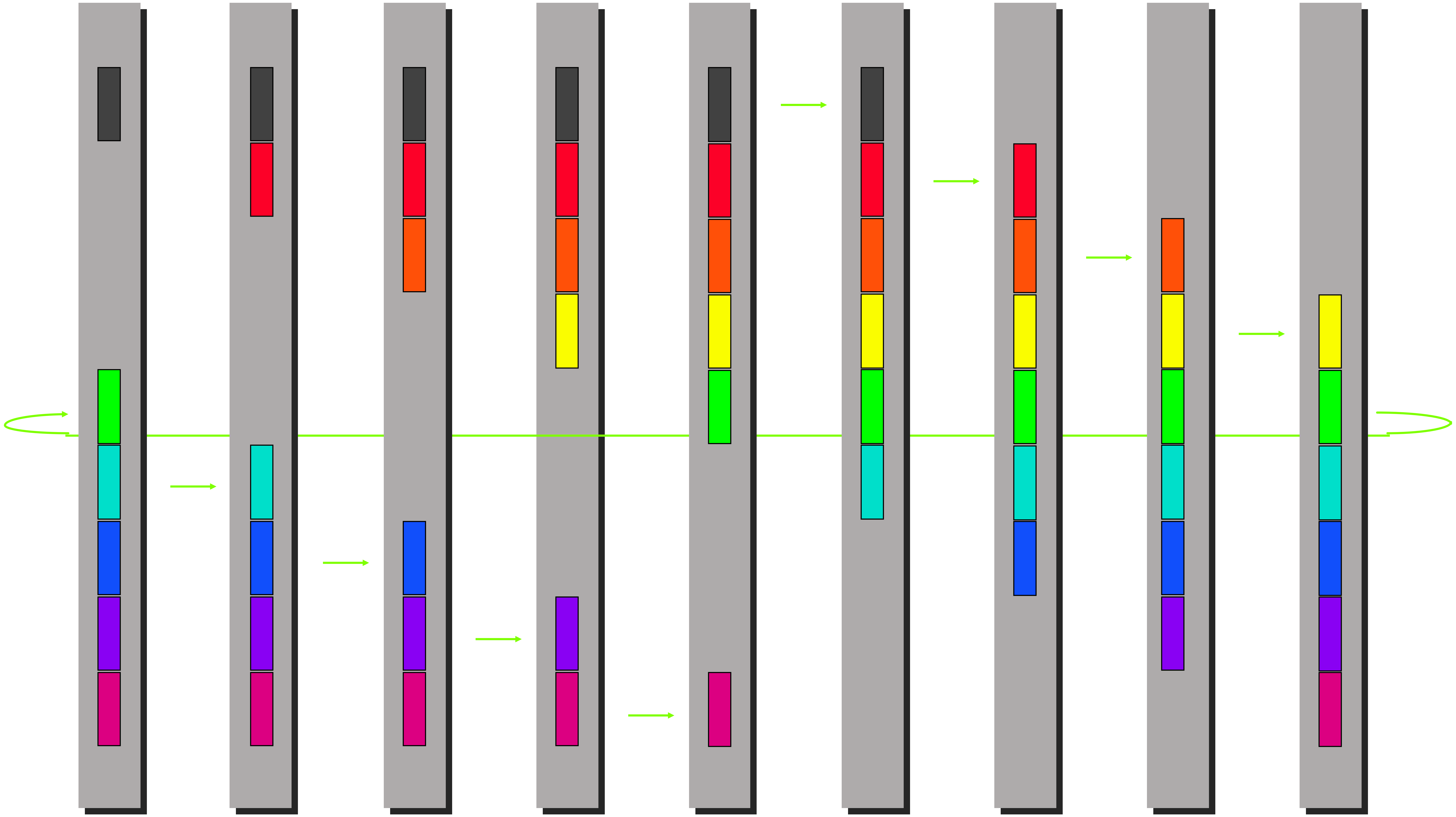


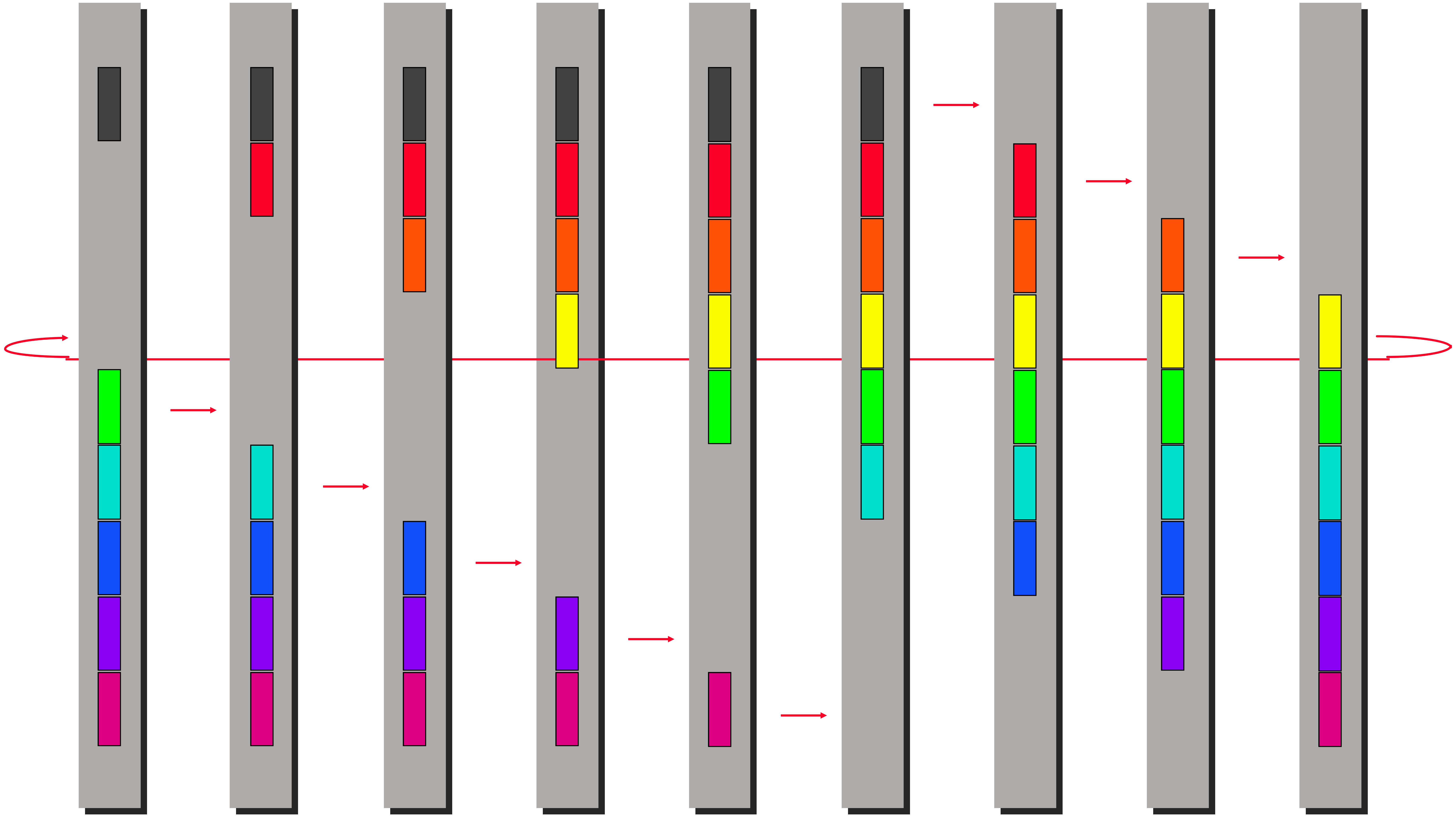


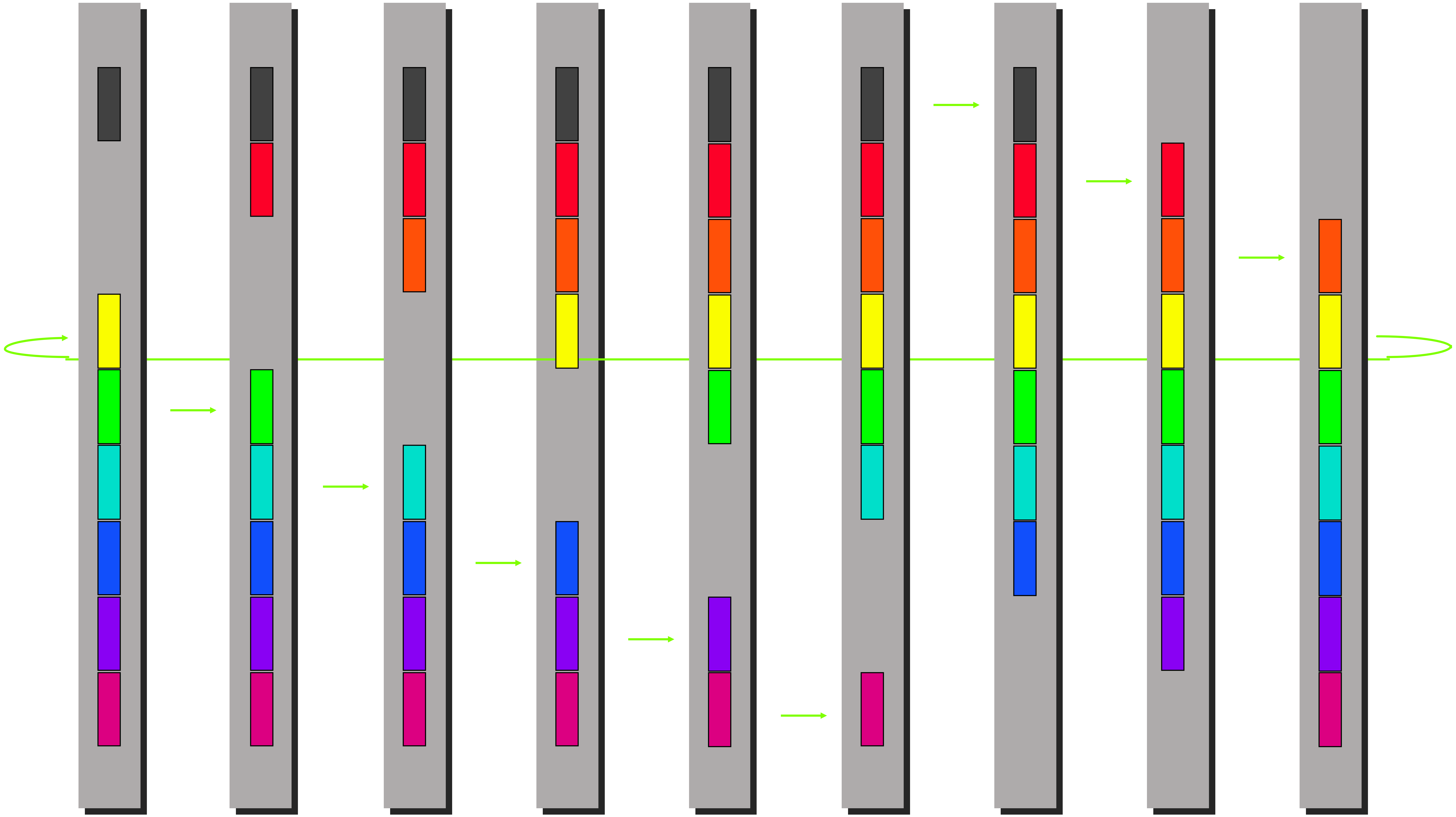


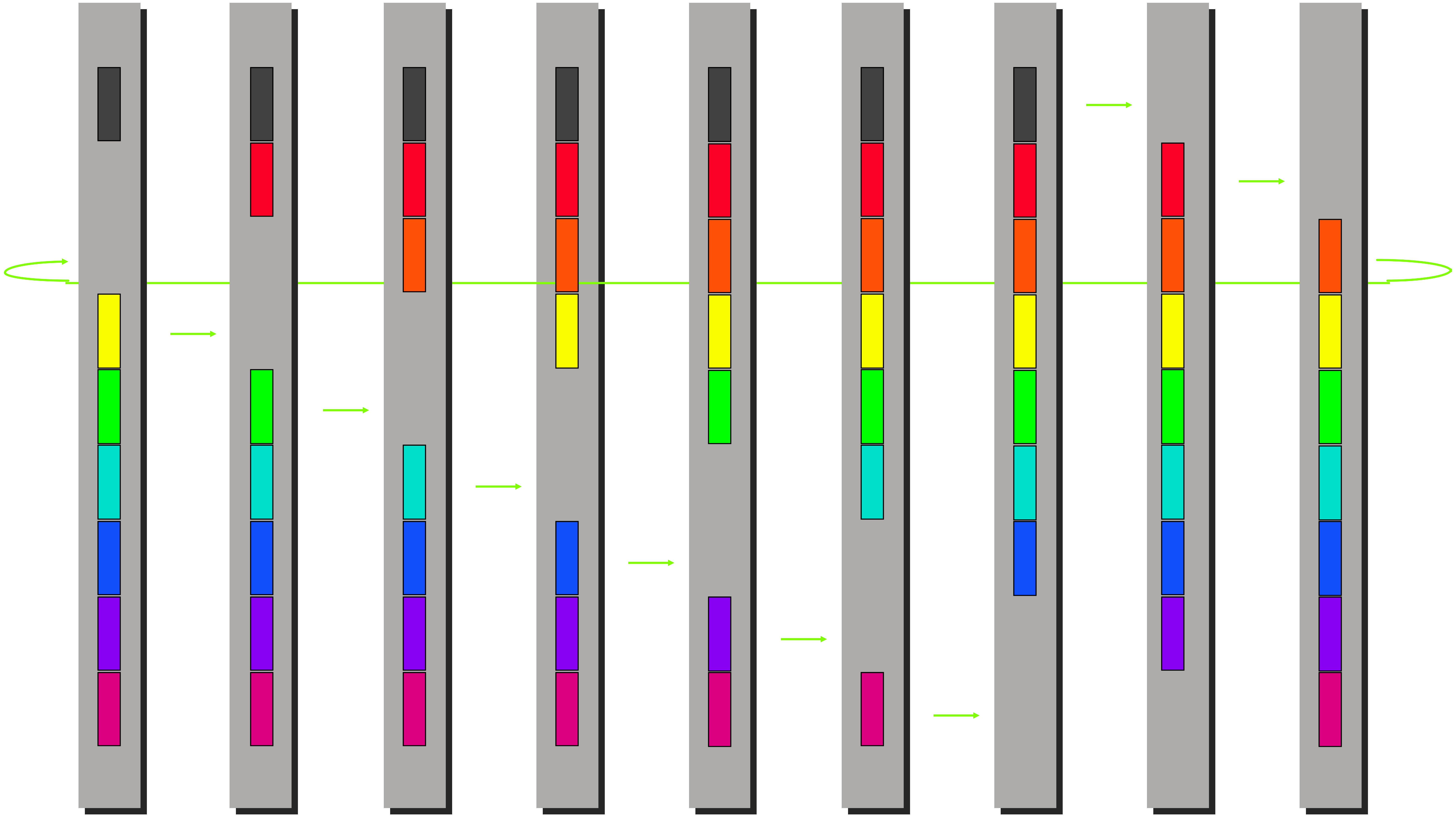


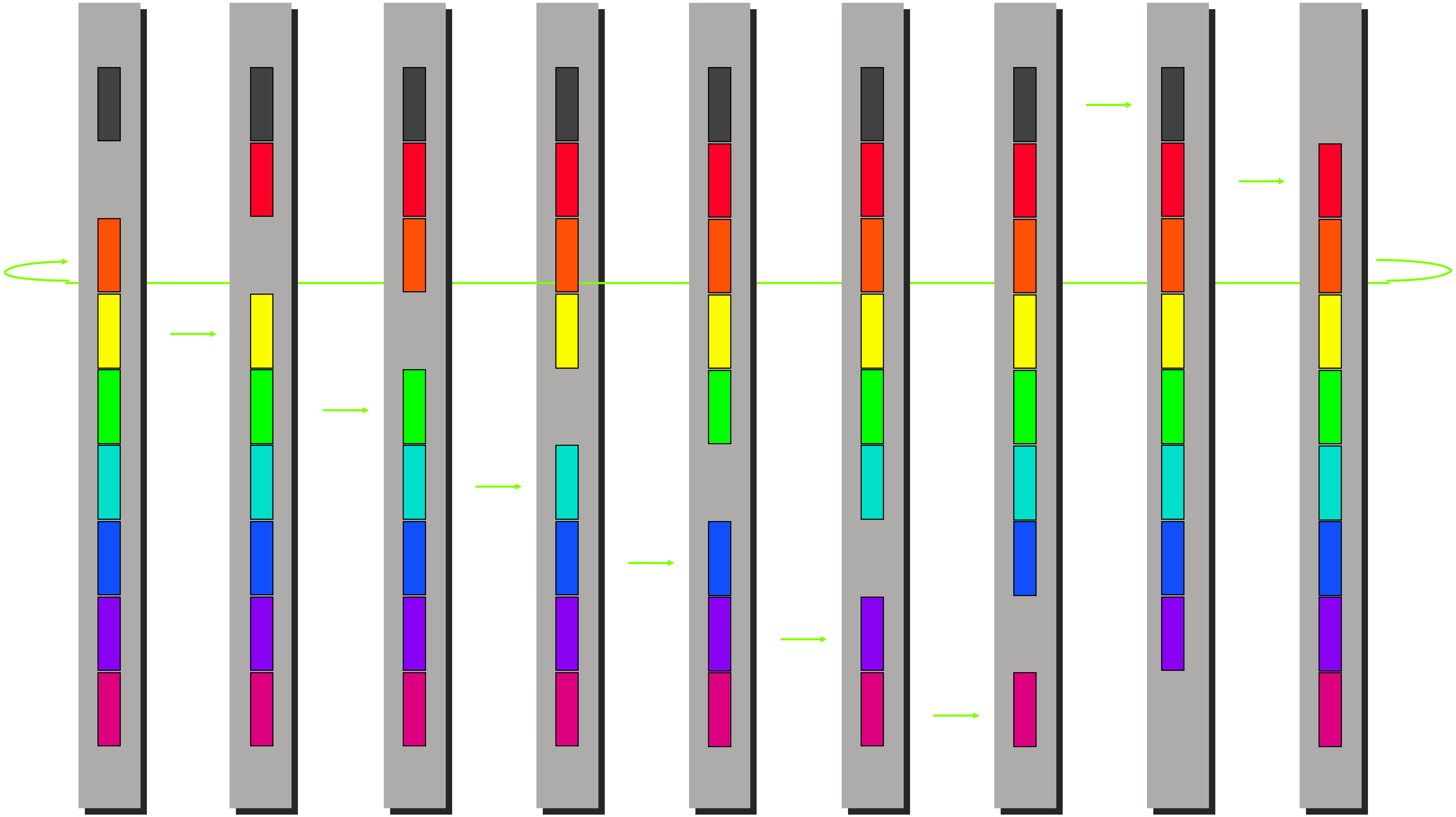


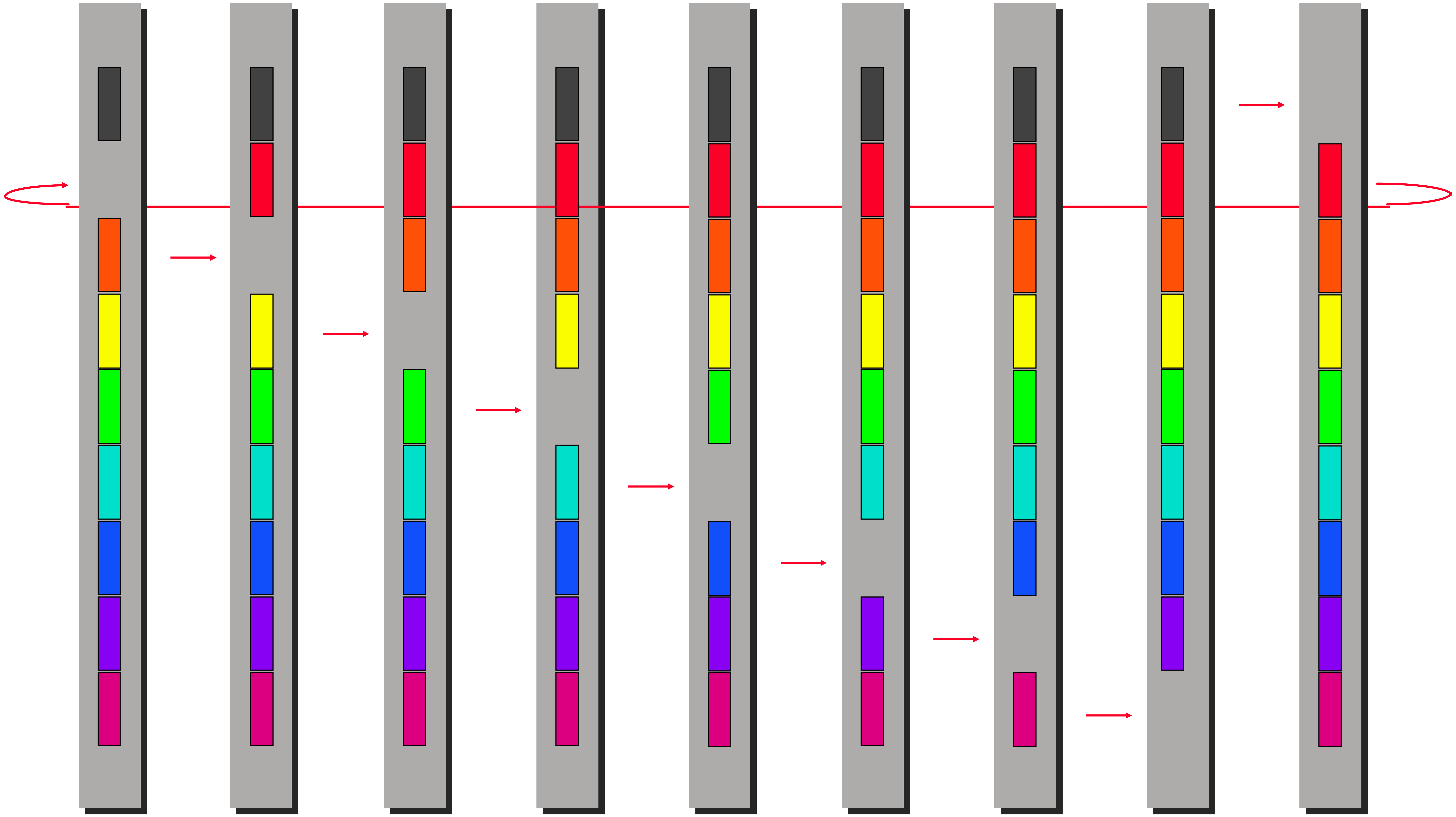


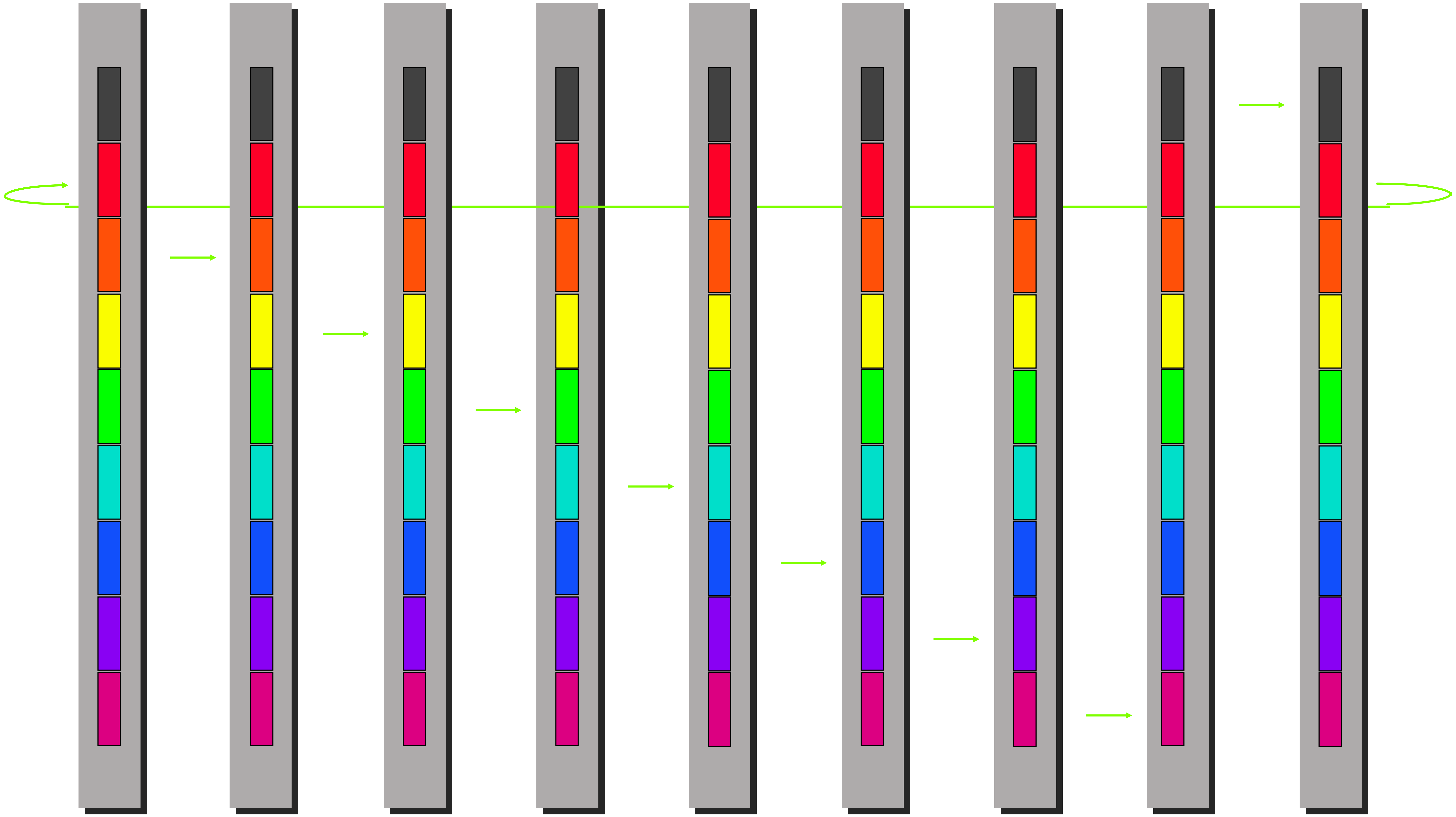


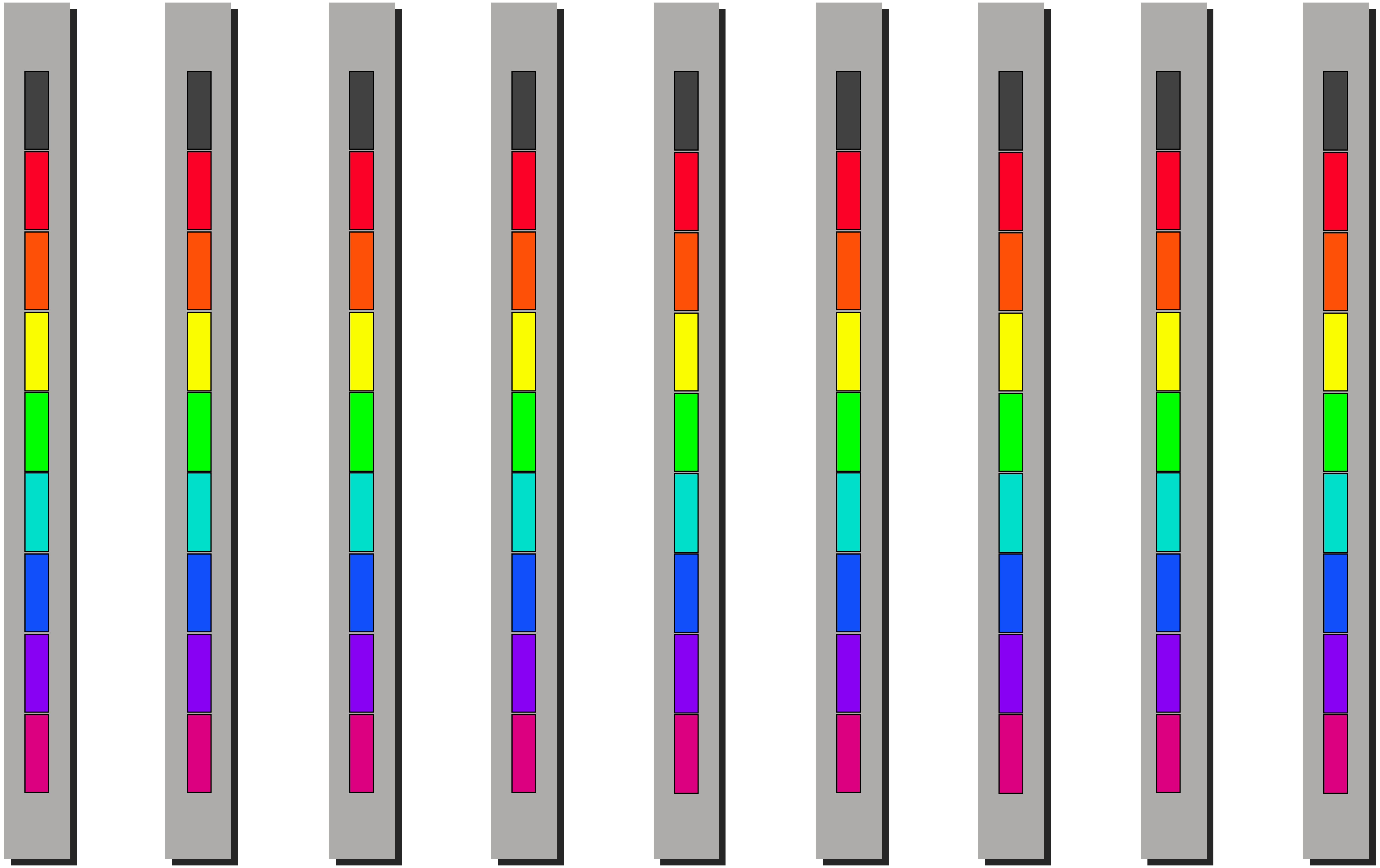












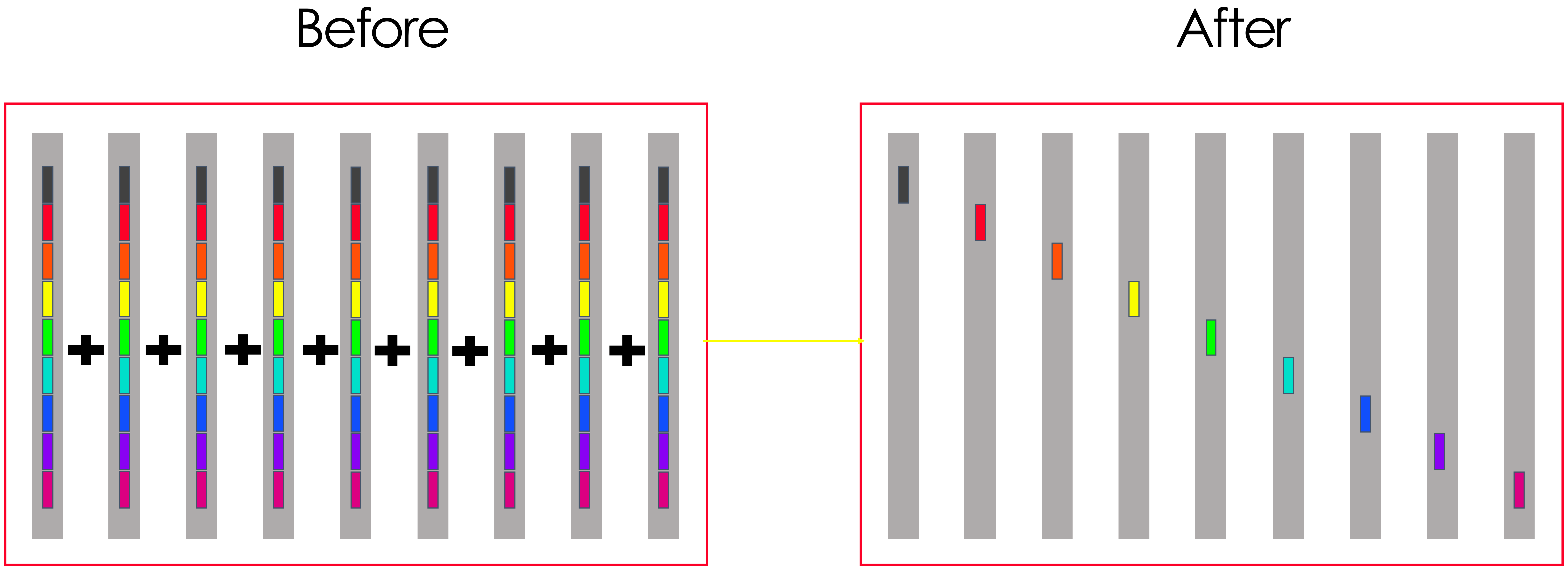
# Cost of bucket Allgather

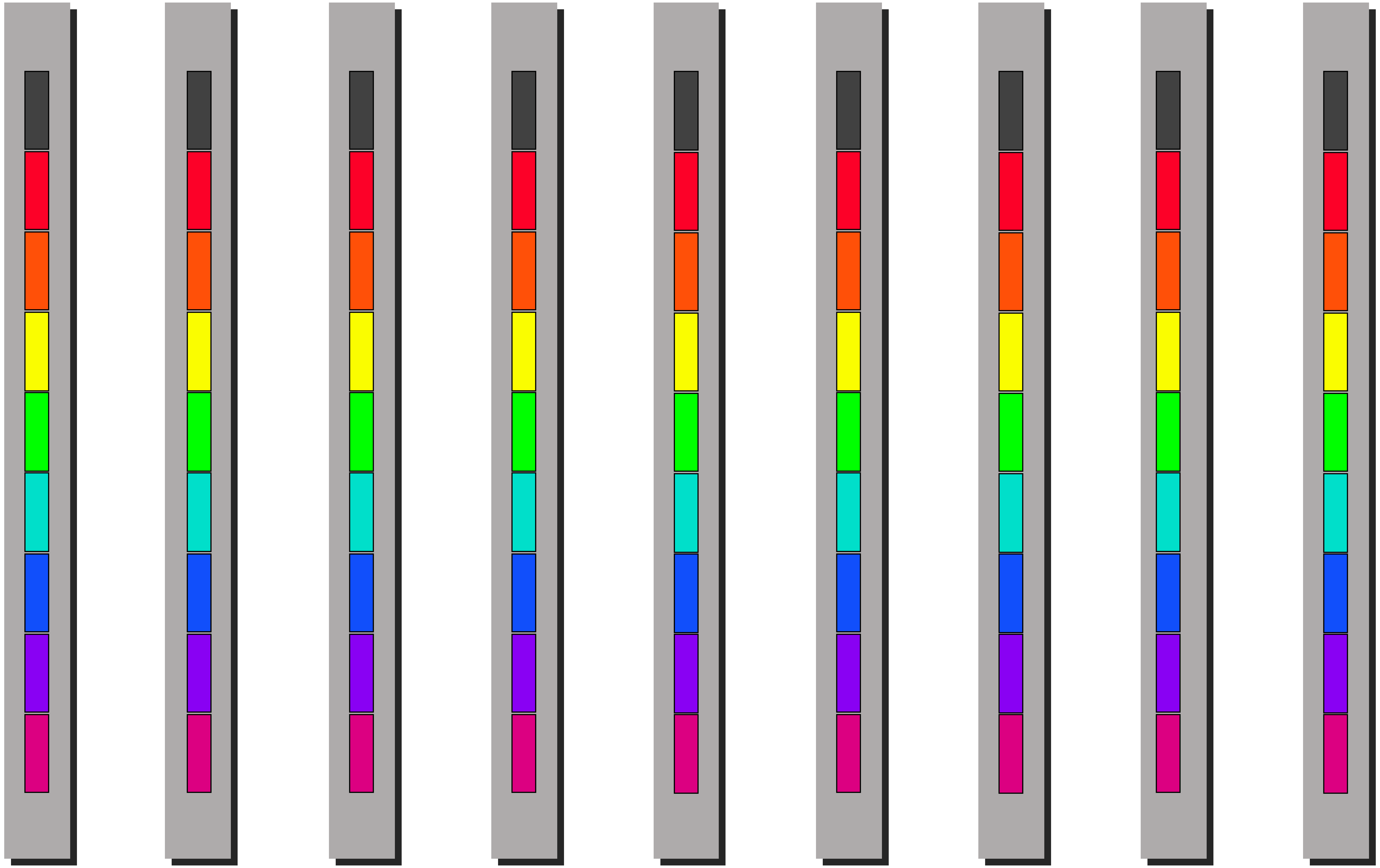
$$\underbrace{\boxed{(p-1)}}_{\text{number of steps}}\ \underbrace{\boxed{\left(\alpha+\frac{n}{p}\beta\right)}}_{\text{cost per steps}}$$

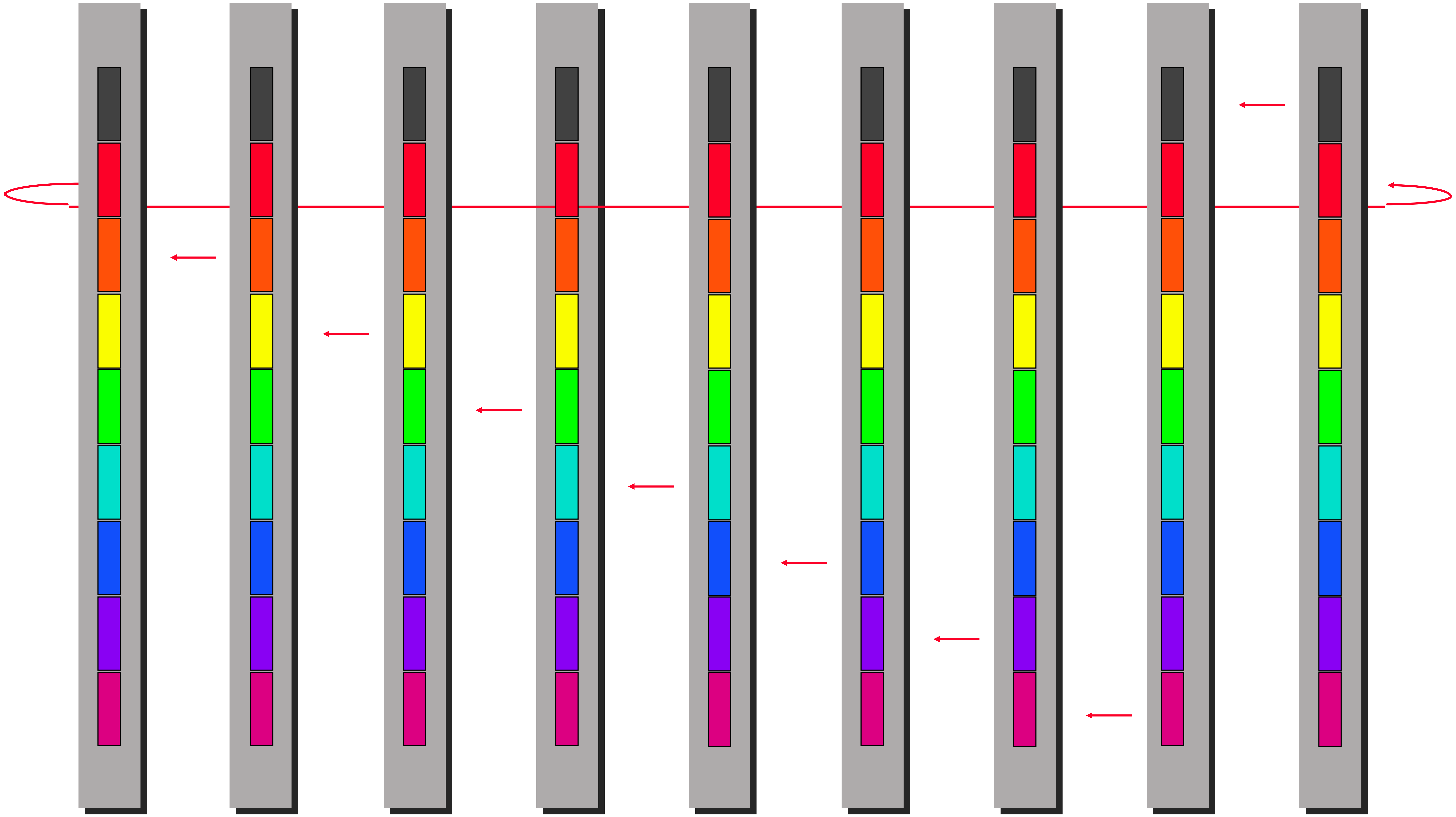
$$=$$

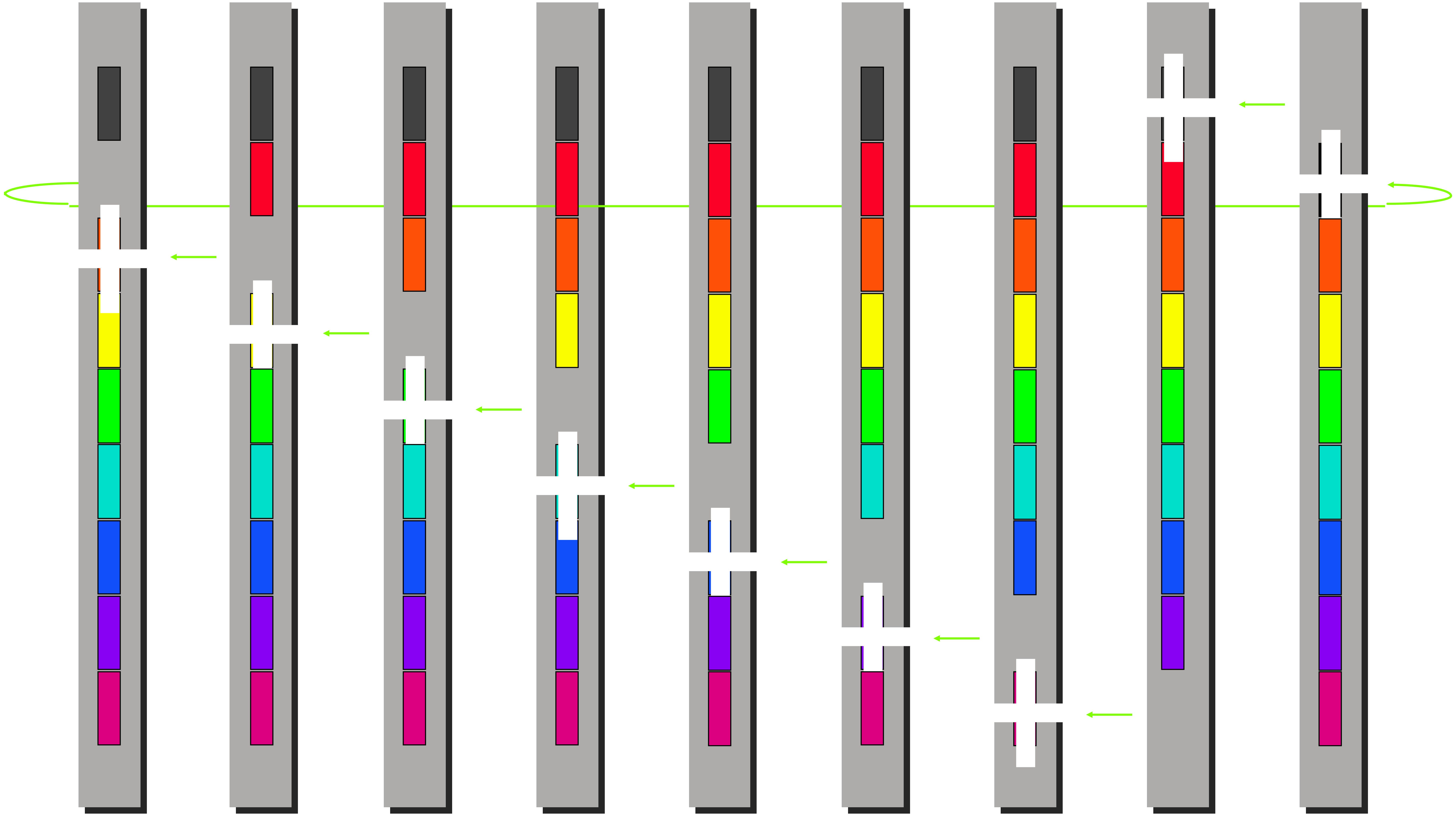
$$(p-1)\alpha+\frac{p-1}{p}n\beta$$

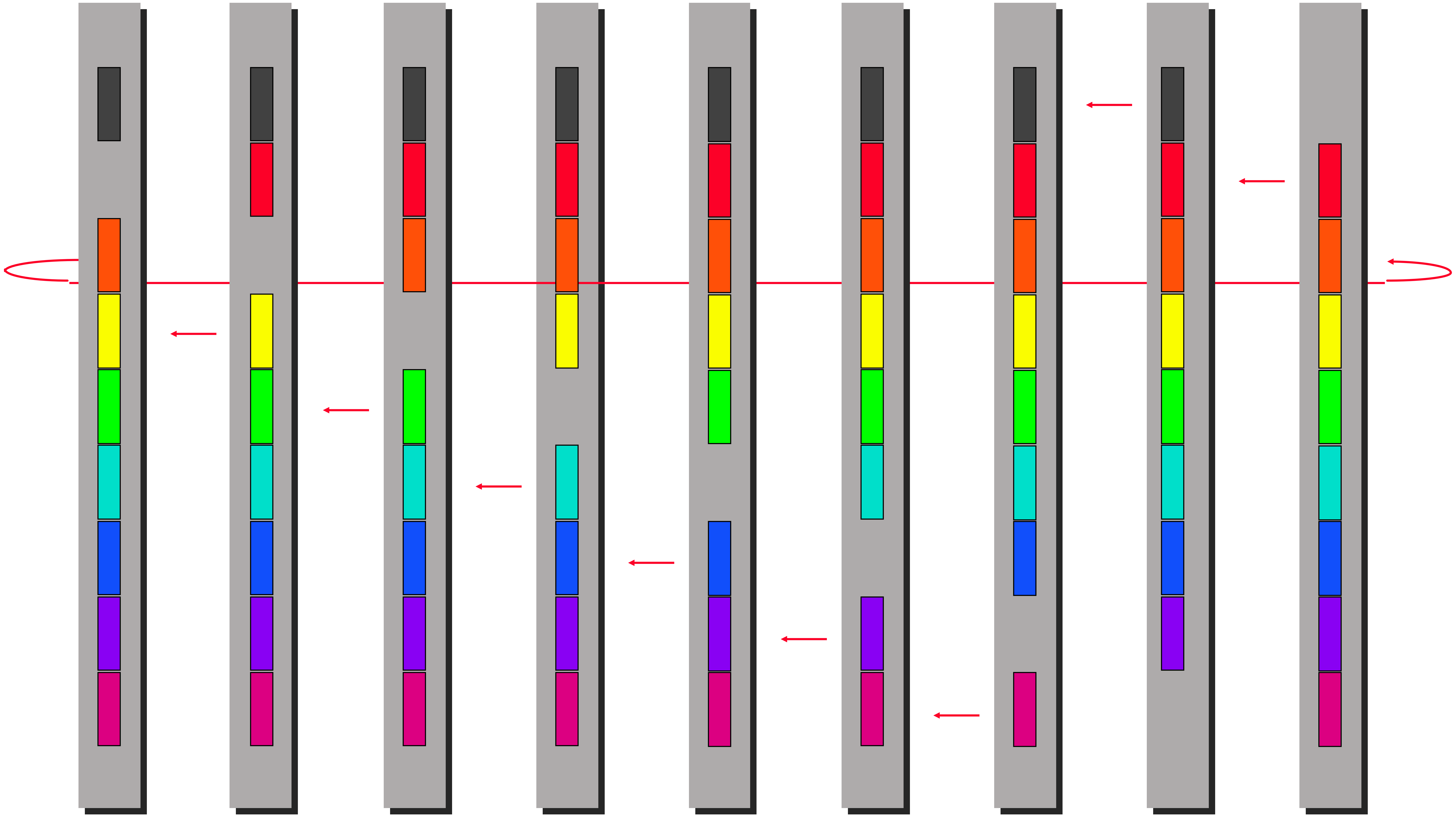
# Reduce-scatter

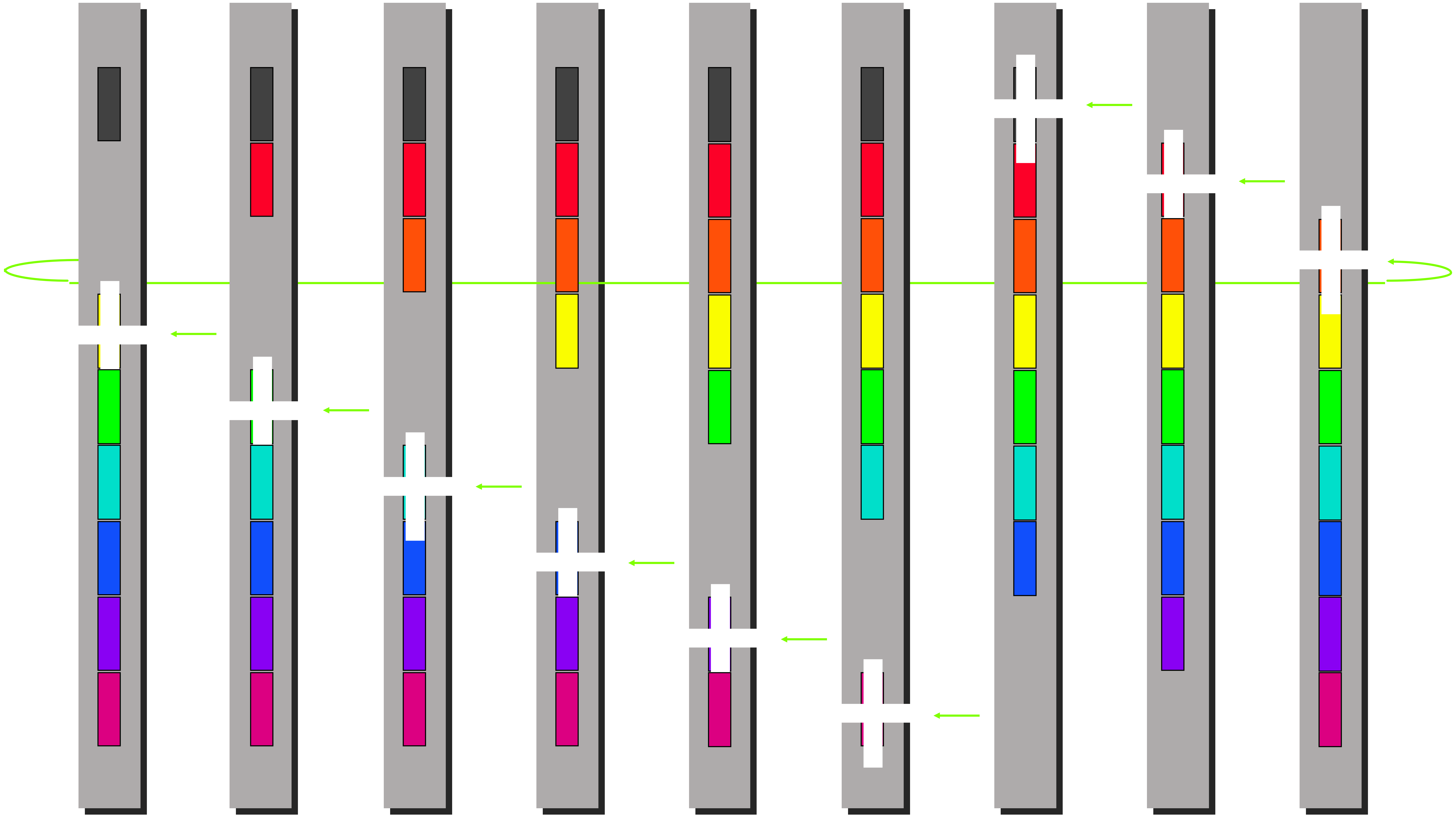


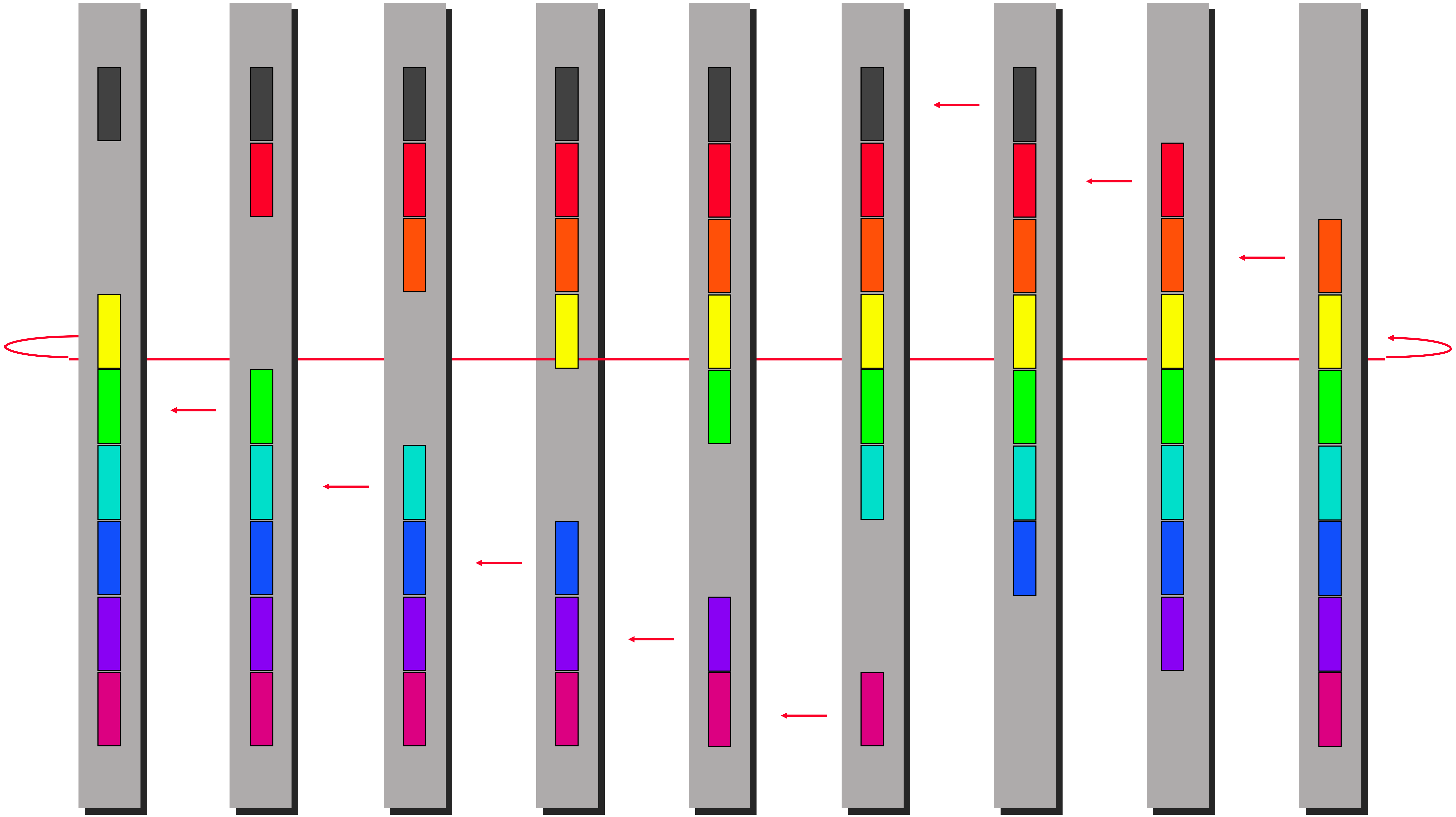


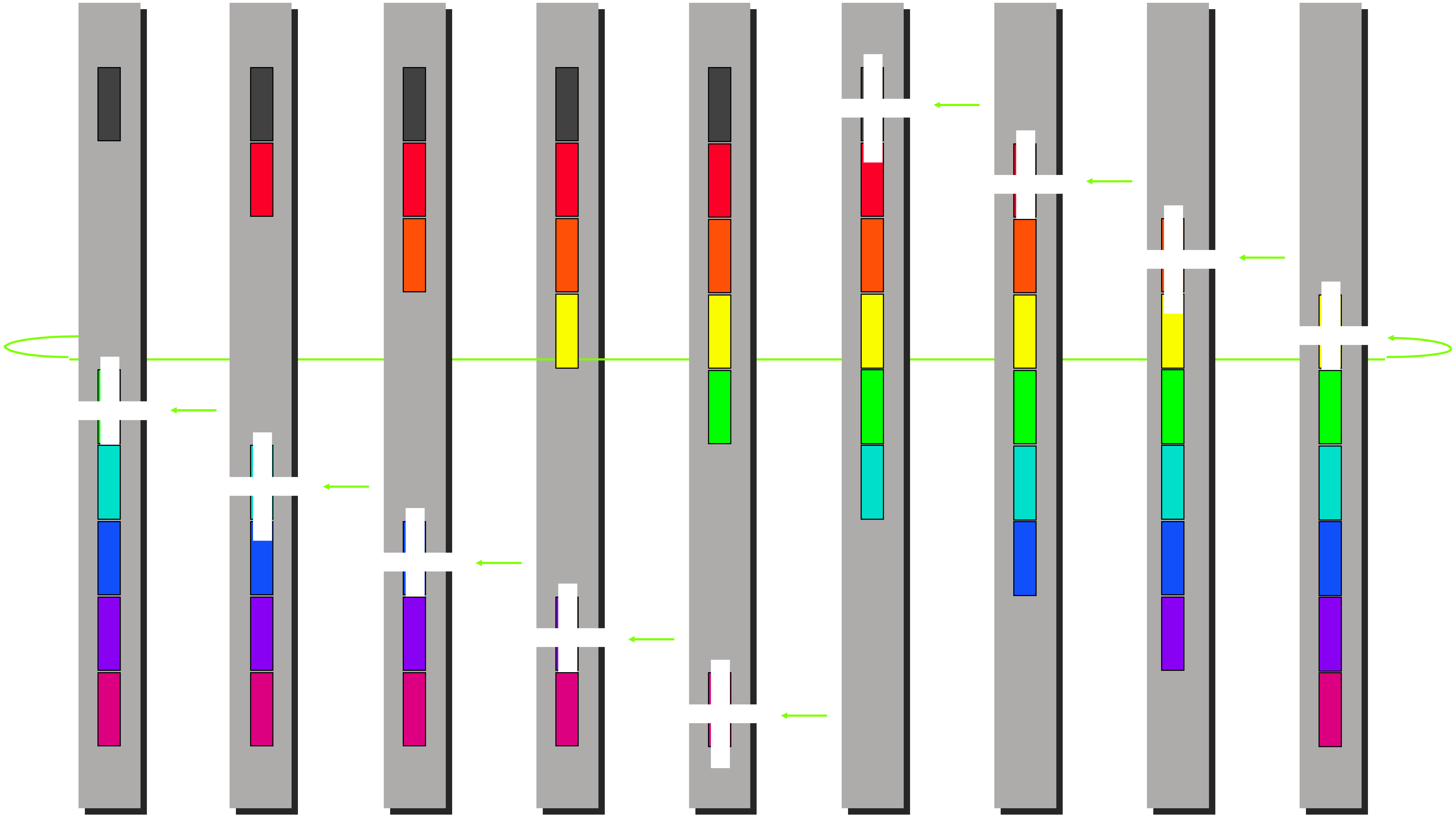


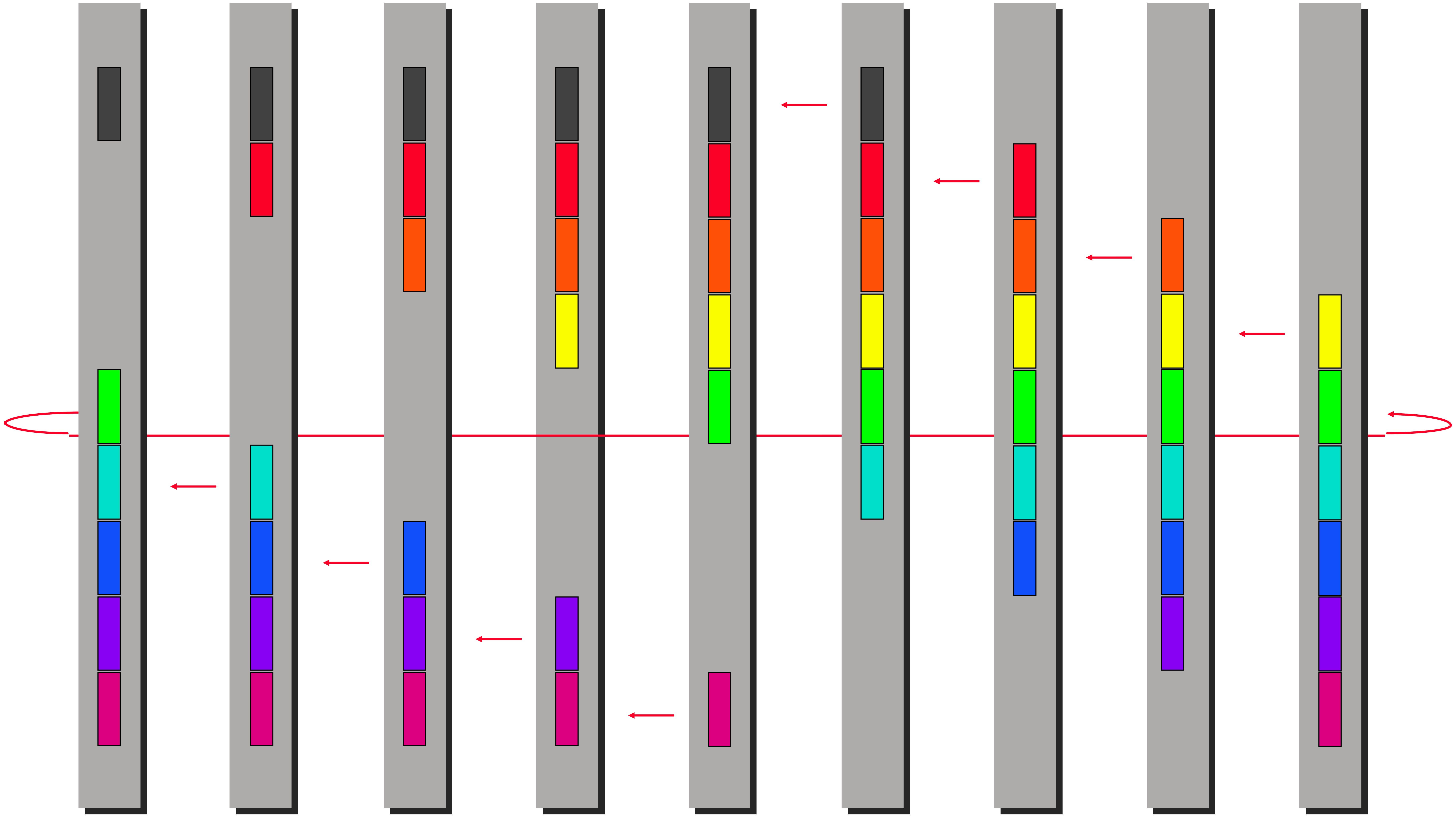


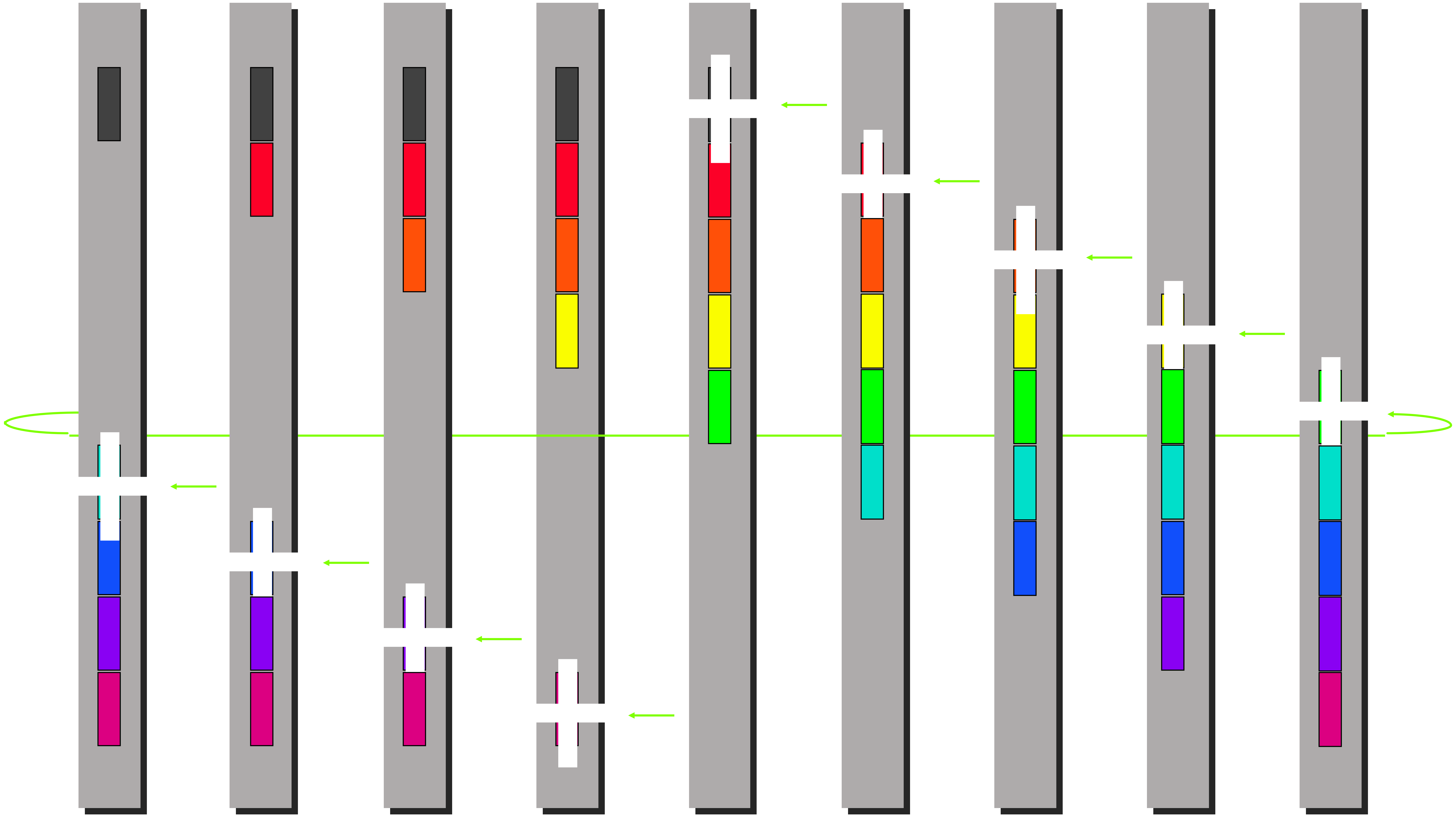


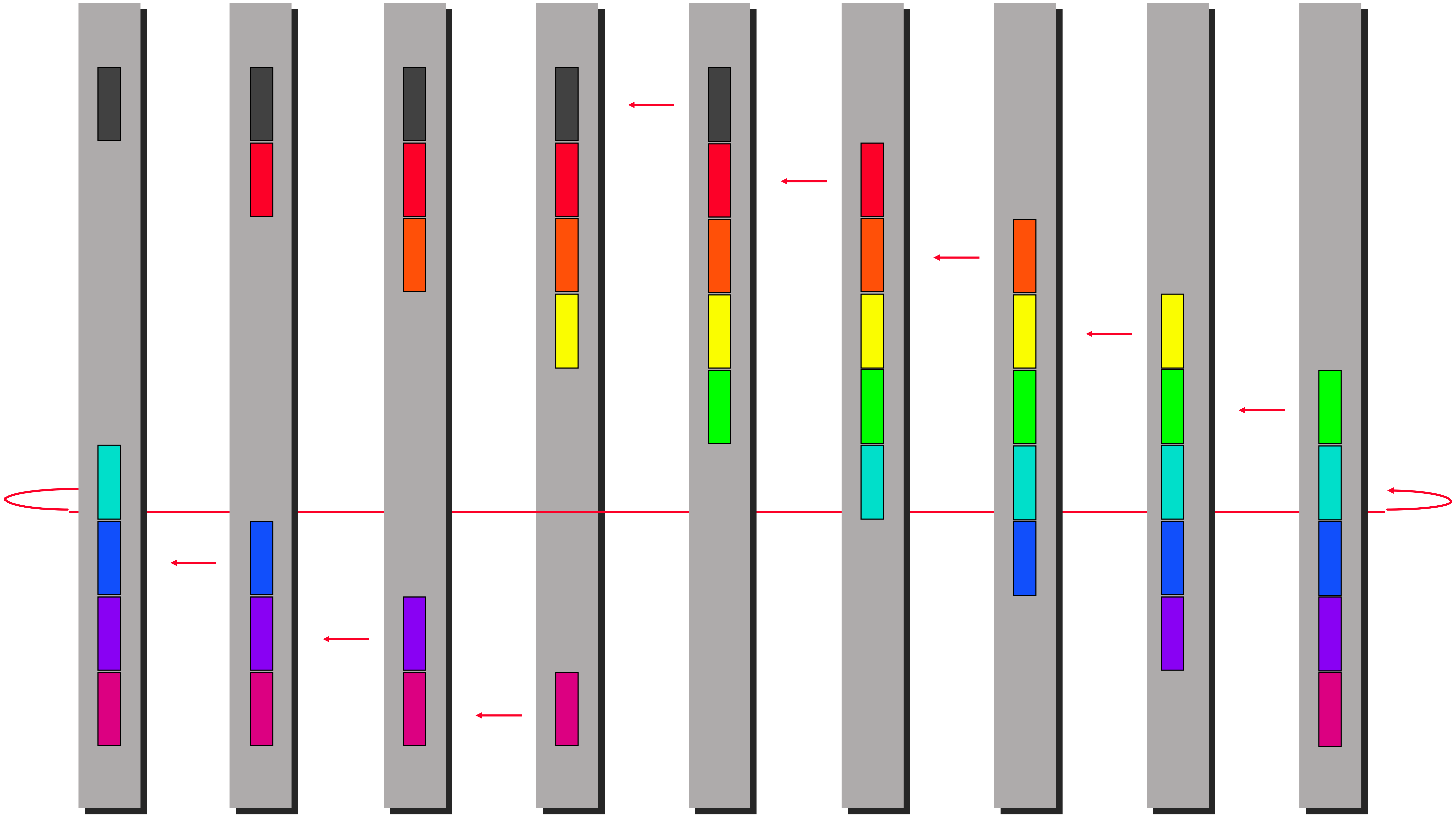


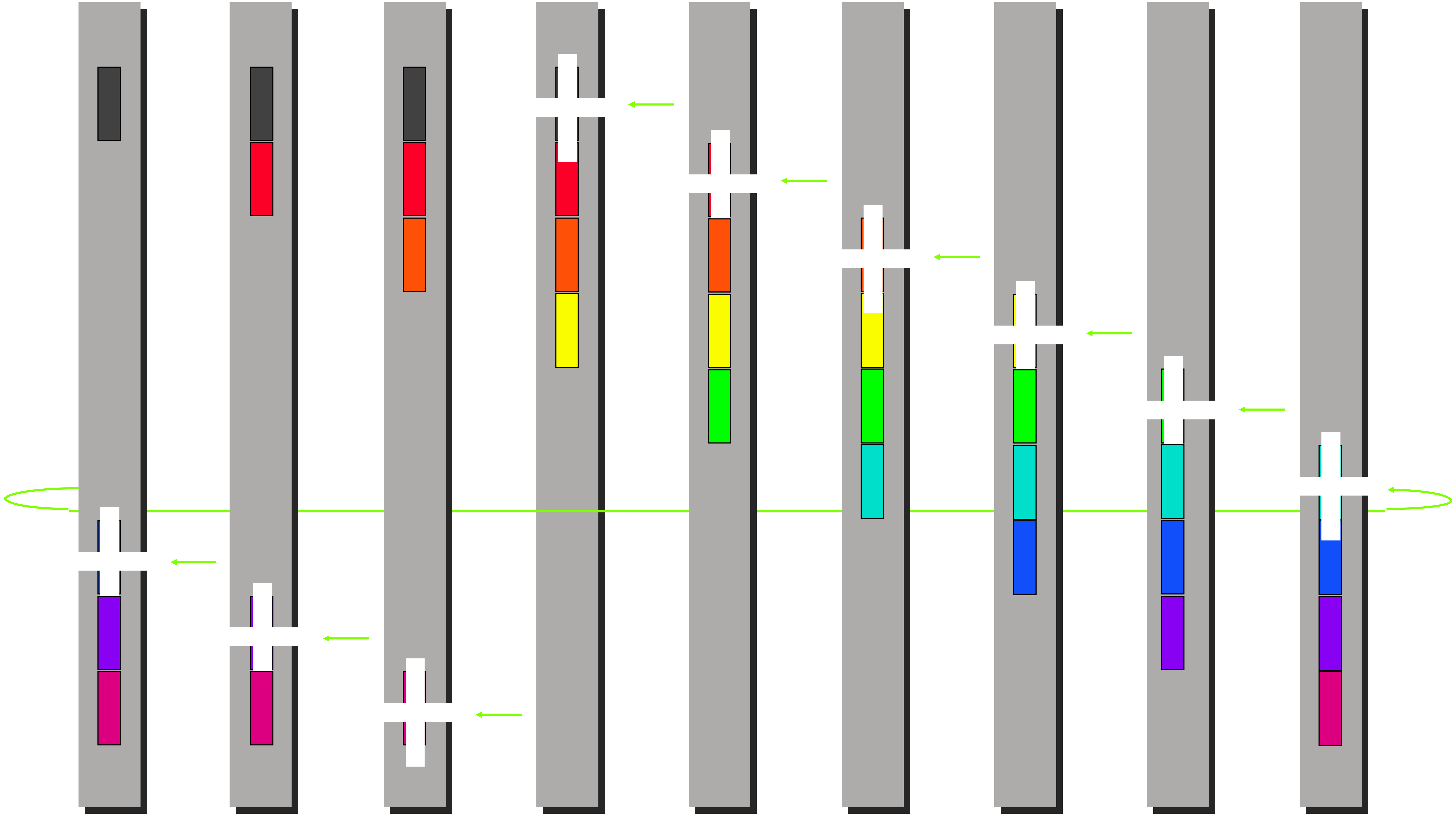


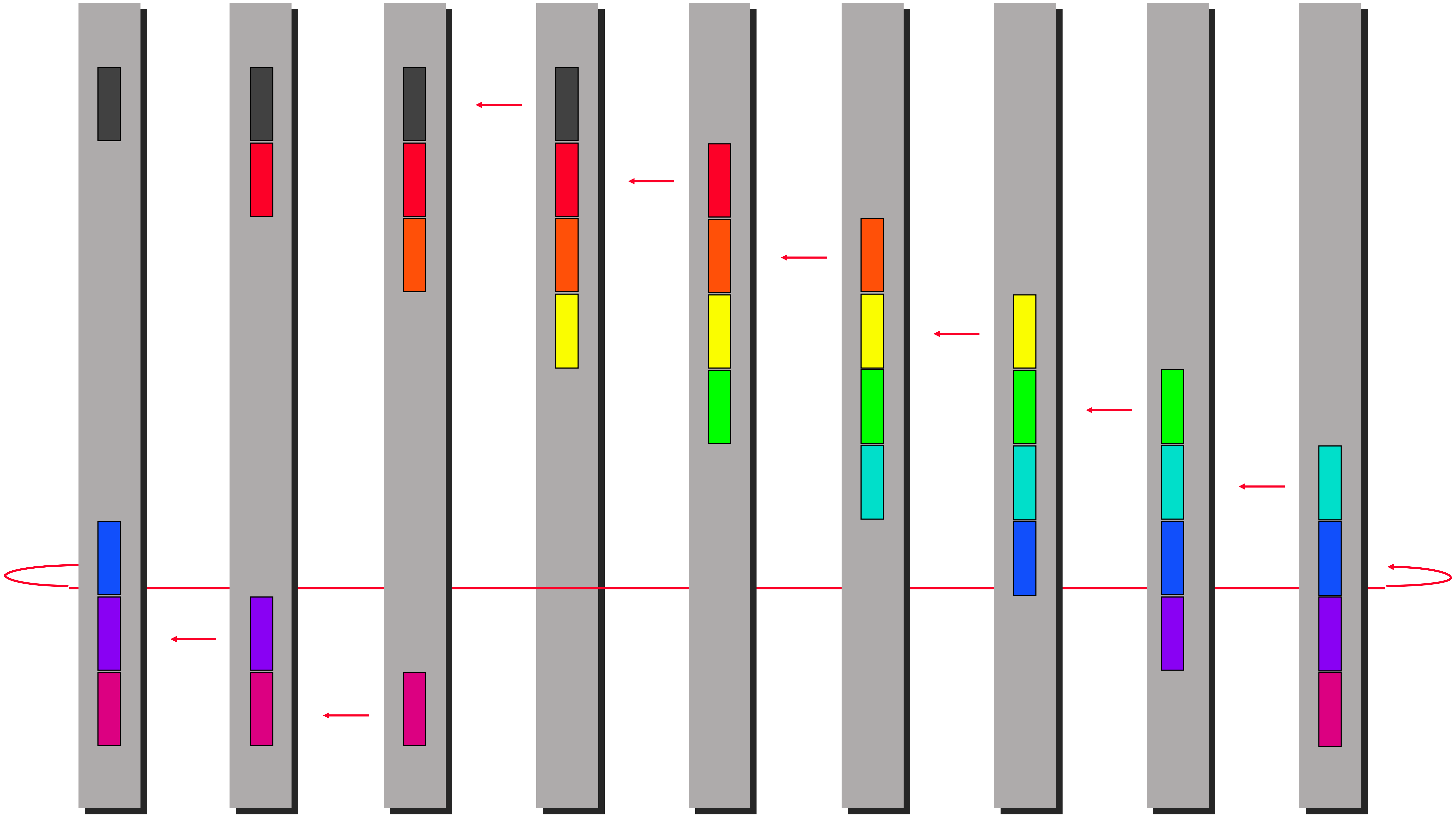


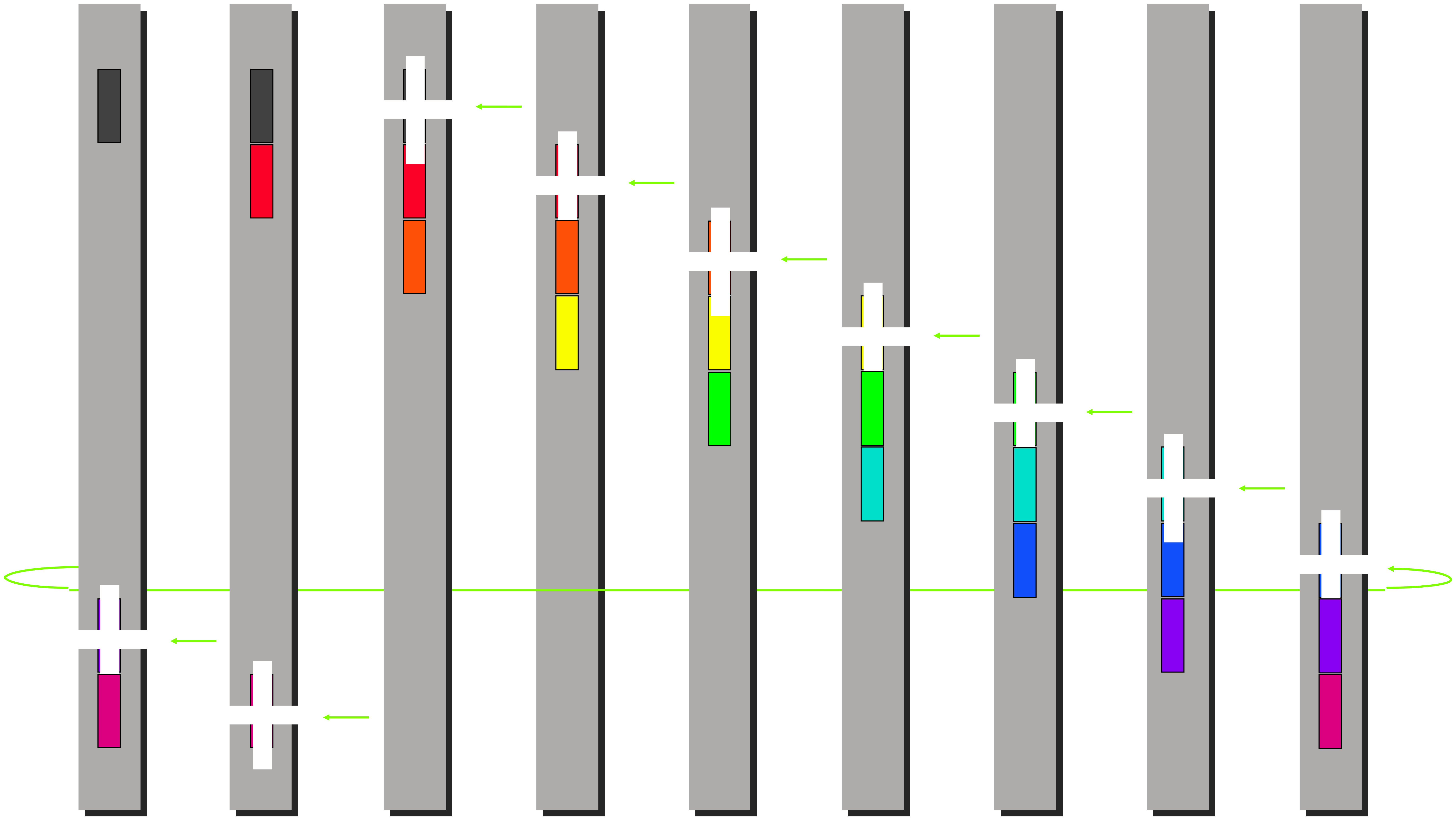


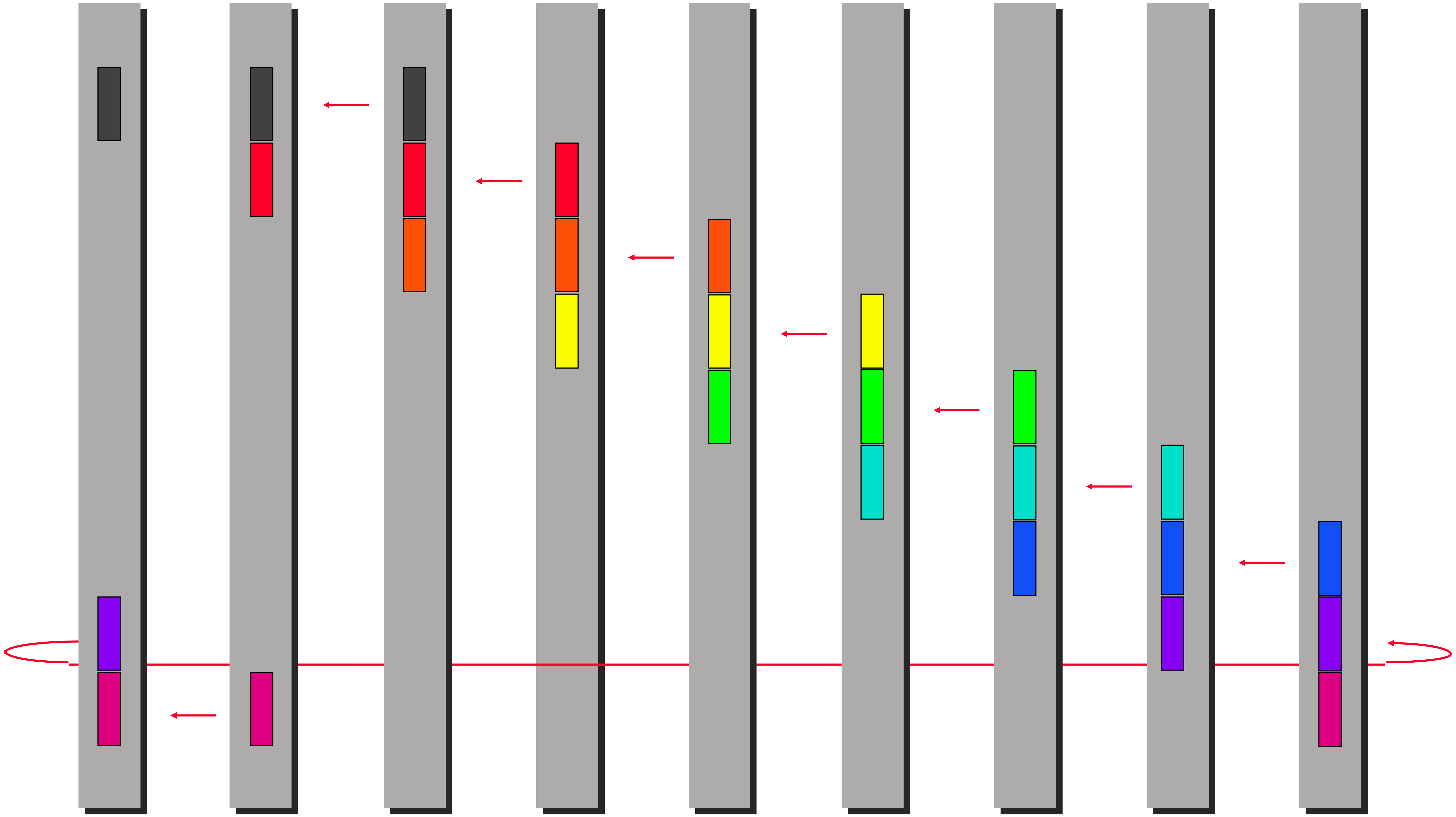


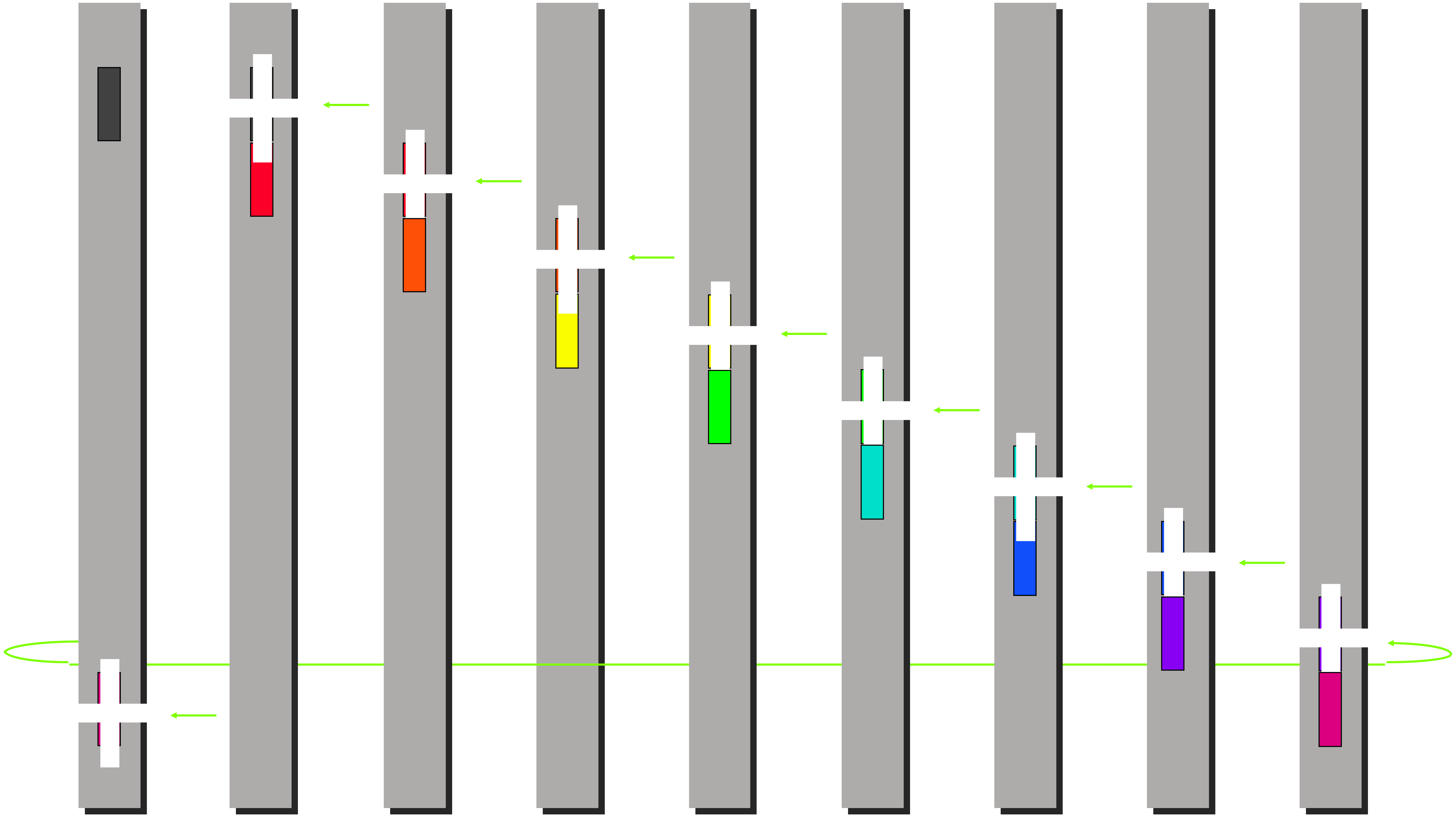


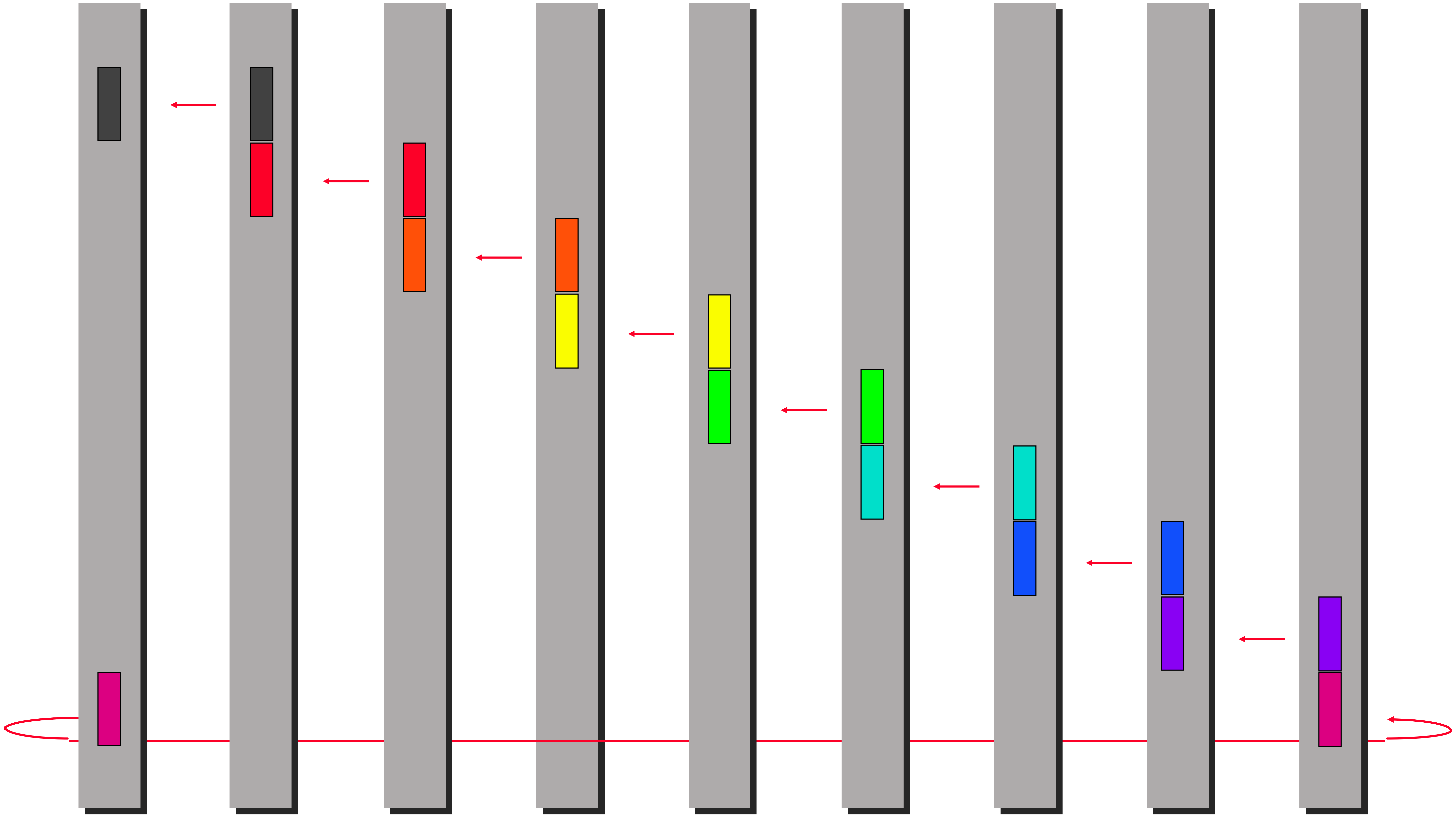


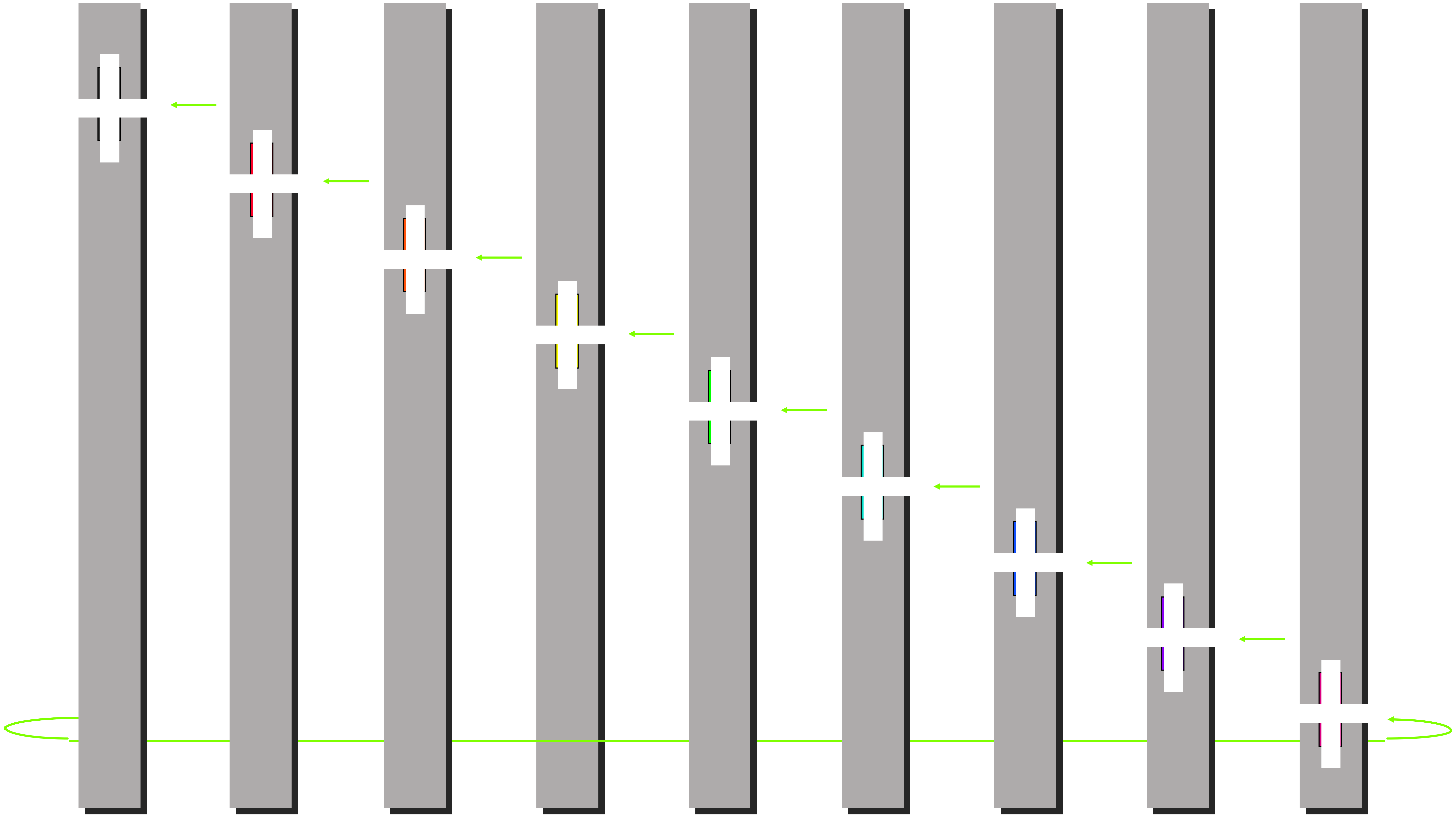


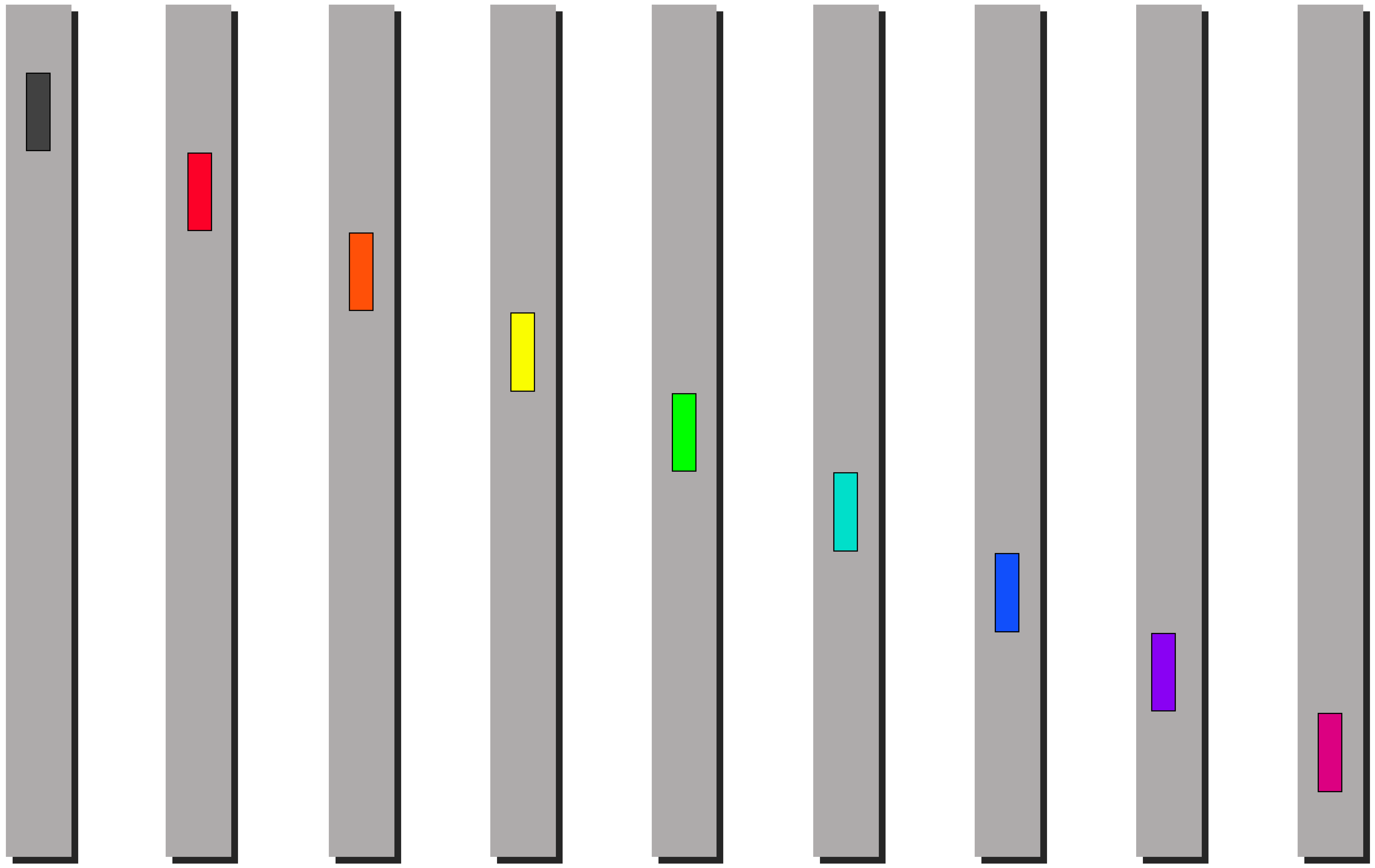












# Cost

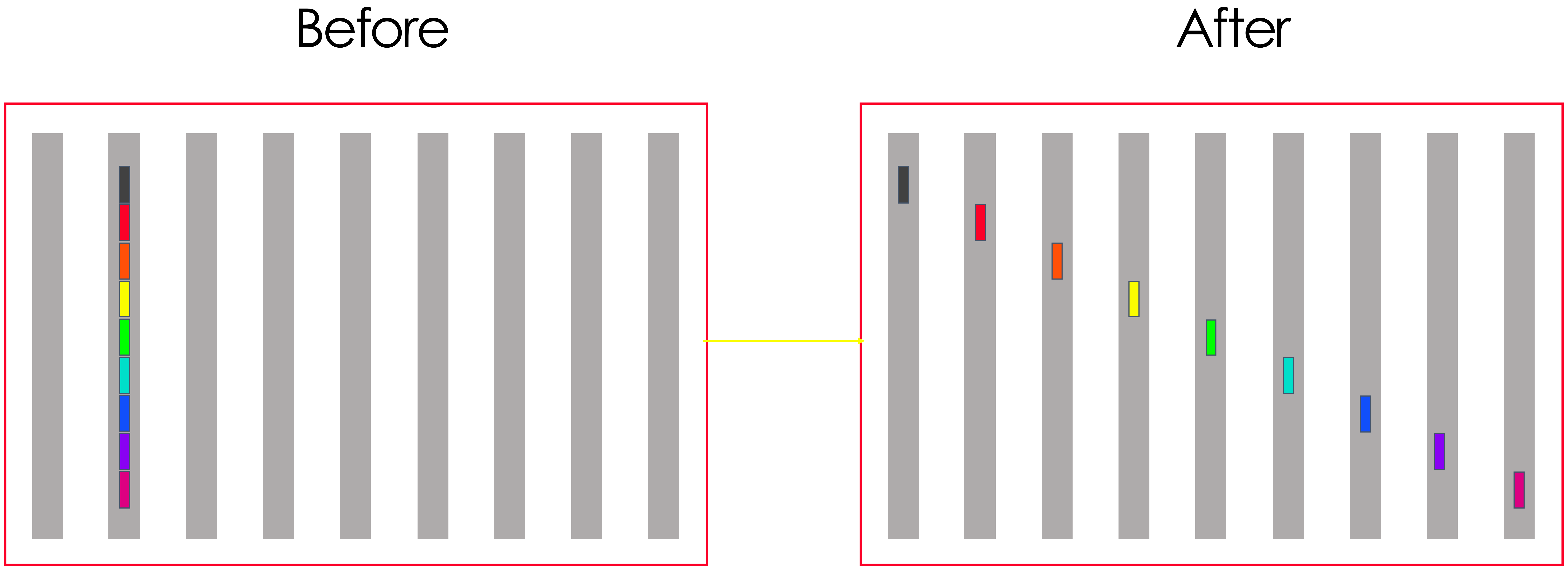
$$\underbrace{(p-1)}_{\text{number of steps}}\underbrace{\left(\alpha+\frac{n}{p}\beta+\frac{n}{p}\gamma\right)}_{\text{cost per steps}}$$

$$=$$

$$(p-1)\alpha+\frac{p-1}{p}n\beta+\frac{p-1}{p}n\gamma$$

# Scatter: Can Ring Be Better?

Notice: Scatter as implemented before using MST was optimal in Bandwidth as well (How to Prove?)

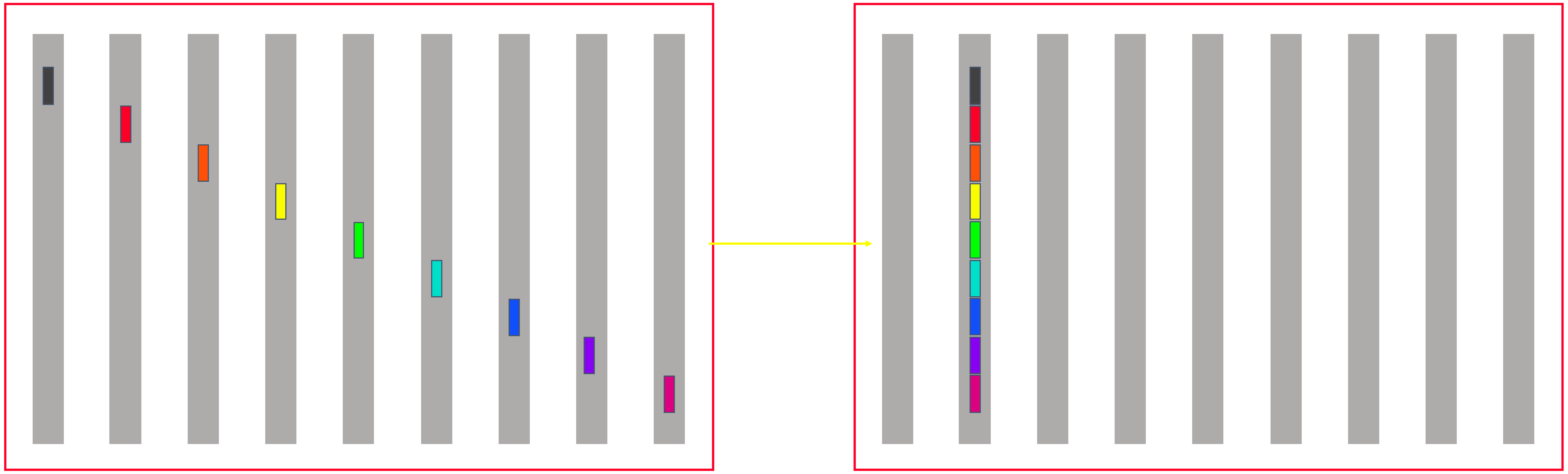


# Gather

Notice: Gather as implemented before using MST was optimal in bandwidth as well (how to prove?)

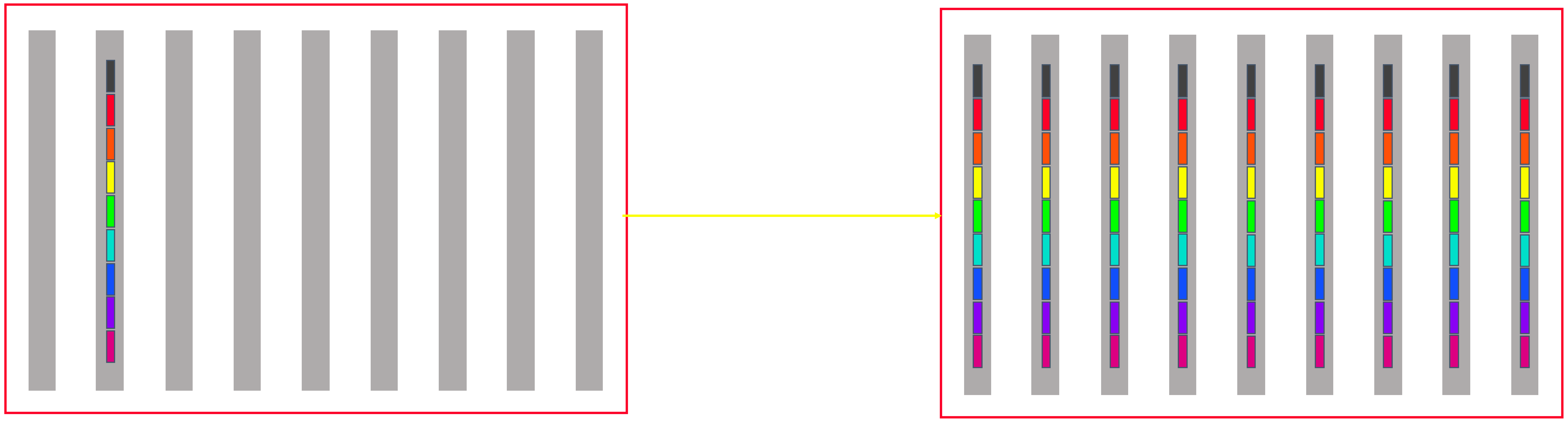
Before

After

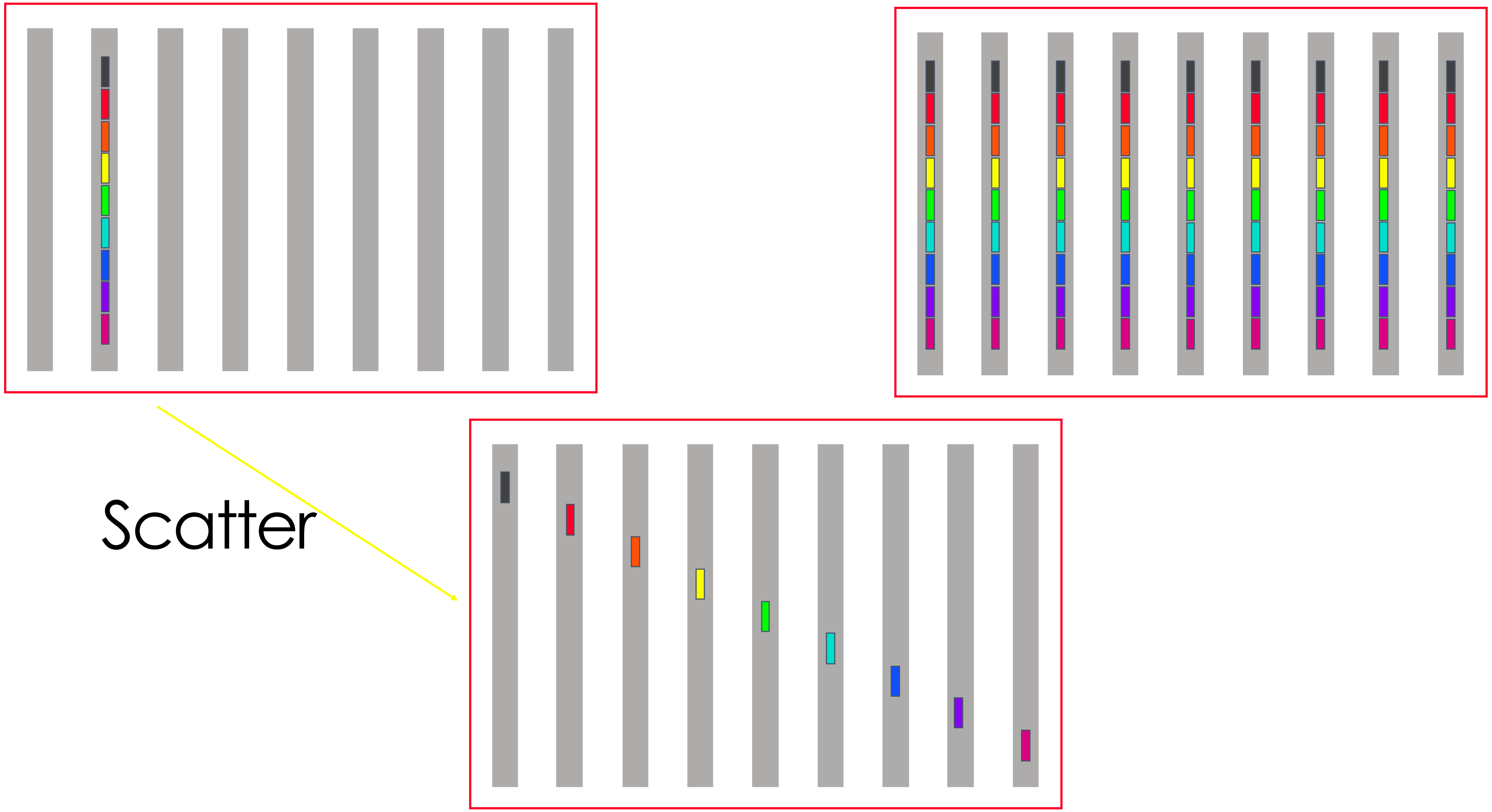


# Using the building blocks

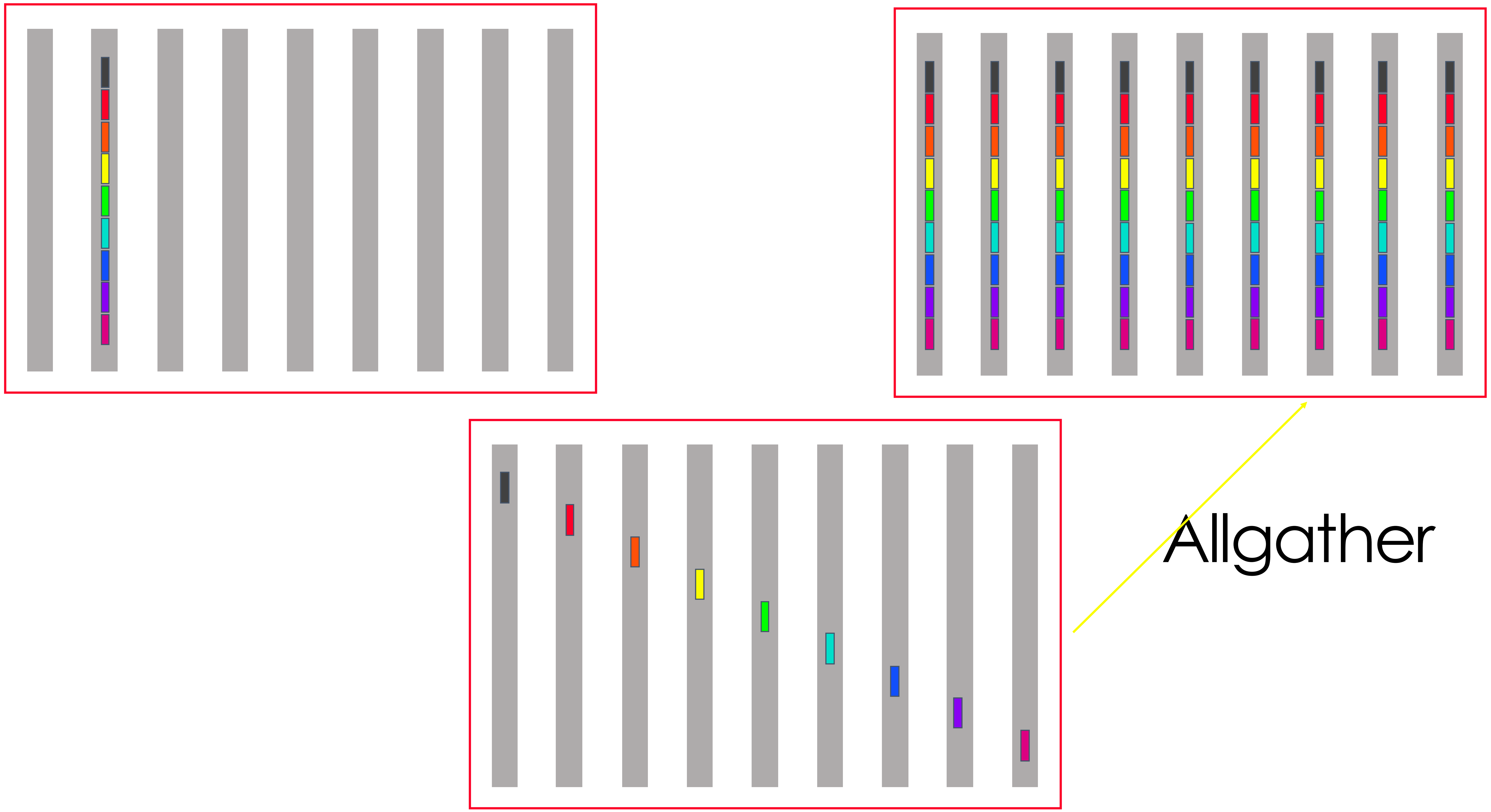
# Broadcast (Large Message)



# Broadcast (Large Message)



# Broadcast (Large Message)



# Cost of scatter/allgather broadcast

- Assumption: power of two number of nodes

scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

allgather

$$(p-1)\alpha + \frac{p-1}{p}n\beta$$

$$(\log(p) + p - 1)\alpha + 2\frac{p-1}{p}n\beta$$

# Cost of scatter/allgather broadcast

- Assumption: power of two number of nodes

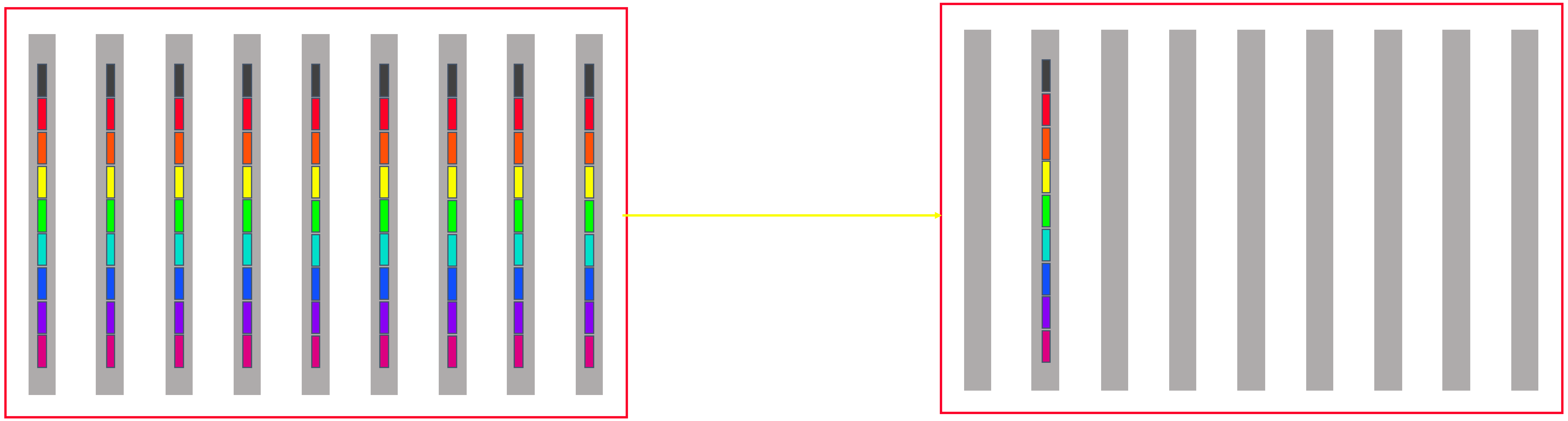
scatter
$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

allgather
$$(p-1)\alpha + \frac{p-1}{p} n\beta$$

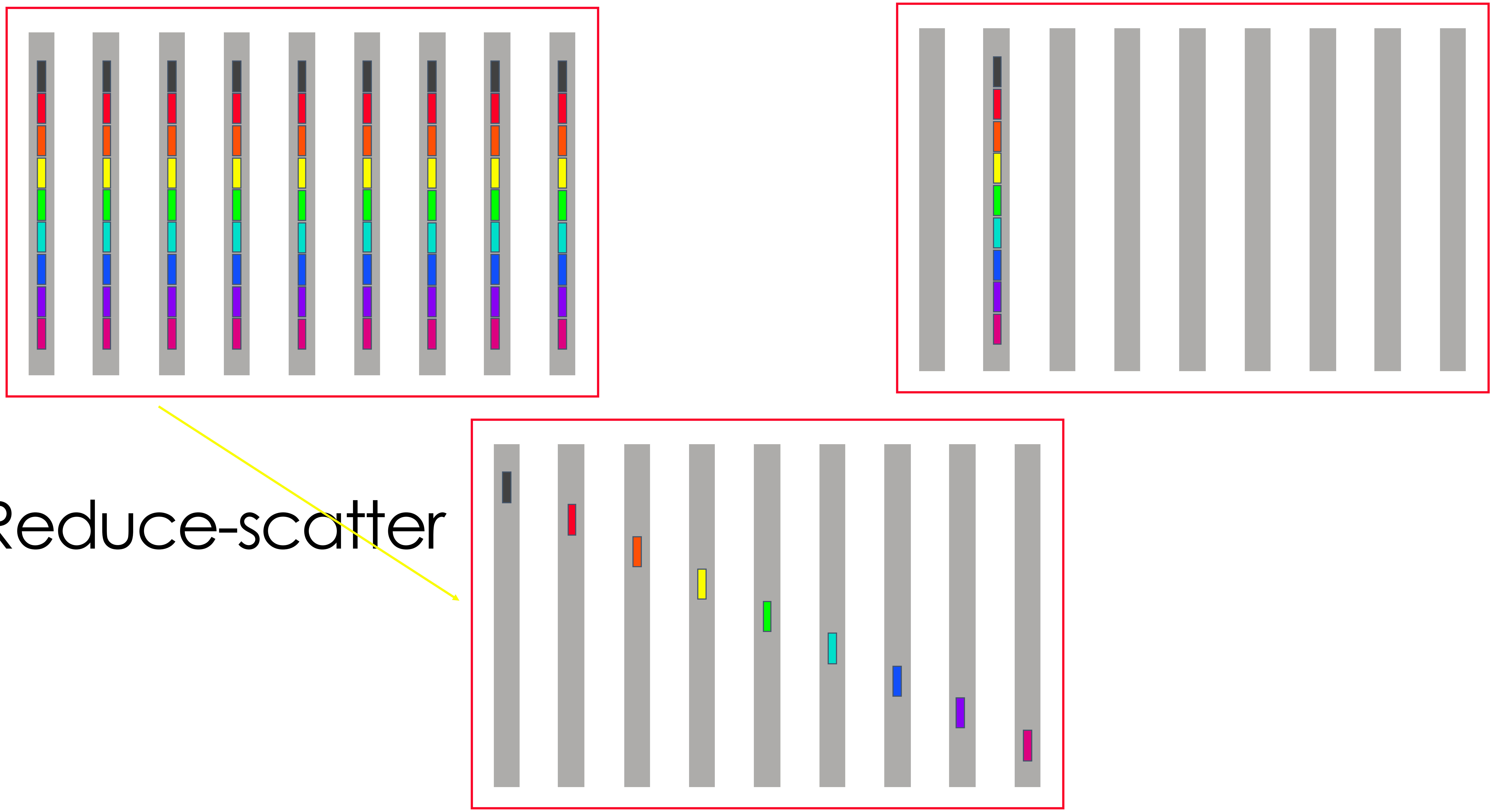
$$(\log(p) + p - 1)\alpha + 2\frac{p-1}{p} n\beta$$

Vs. MST broadcast: $\lceil \log(p) \rceil (\alpha + n\beta)$

# Reduce(-to-one) (Large Message)

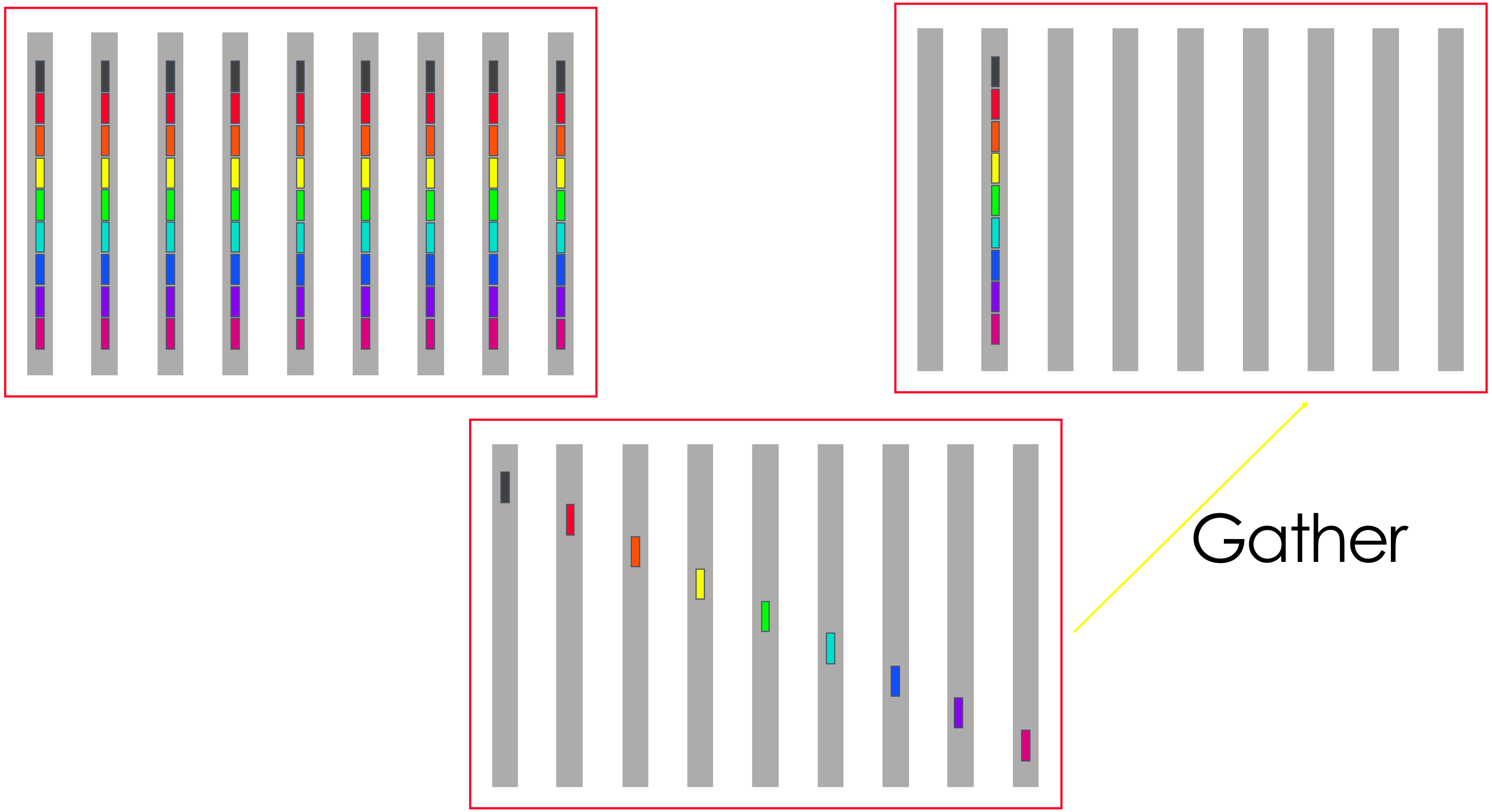


# Reduce (long vector)



Reduce-scatter

# Combine-to-one (long vector)



# Cost of Reduce-scatter/Gather Reduce(-to-one)

- Assumption: power of two number of nodes

Reduce-scatter $$(p-1)\alpha + \frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma$$

gather $$\log(p)\alpha + \frac{p-1}{p} n\beta$$

---

$$(\log(p) + p - 1)\alpha + 2\frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma$$

# Cost of Reduce-scatter/Gather Reduce(-to-one)

- Assumption: power of two number of nodes

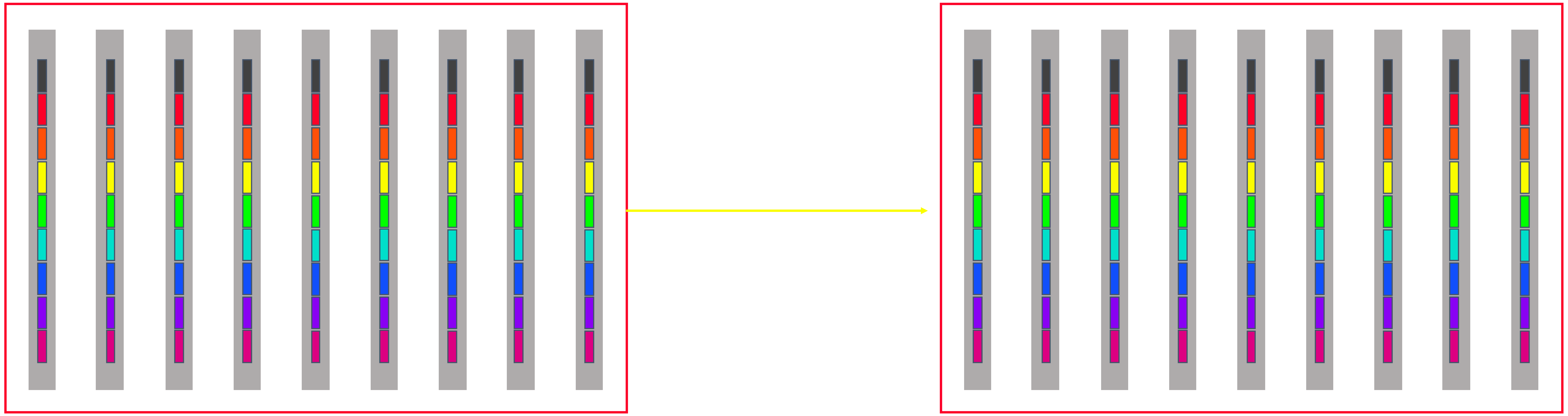
Reduce-scatter $$(p-1)\alpha + \frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma$$

gather $$\log(p)\alpha + \frac{p-1}{p} n\beta$$

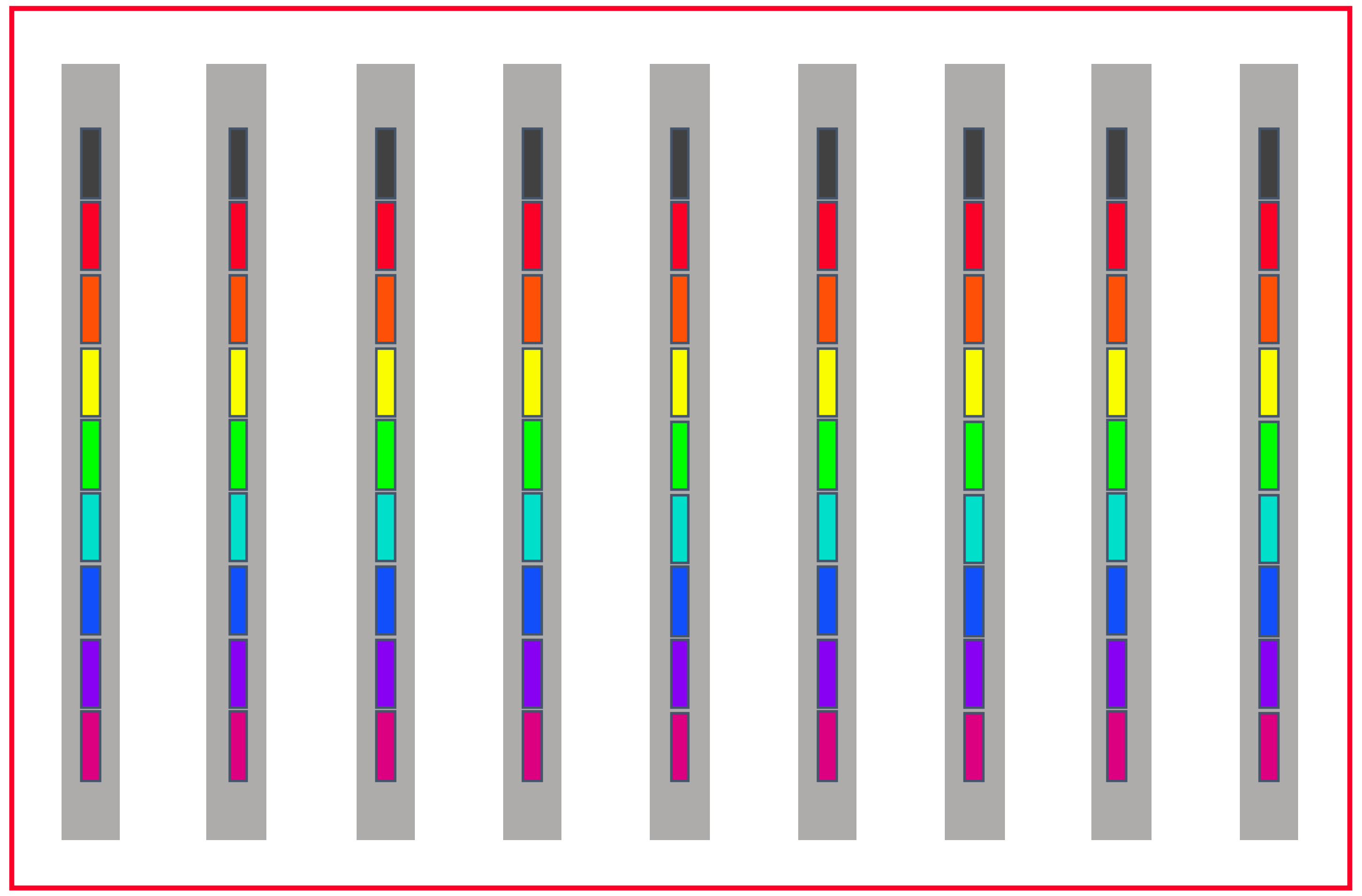
$$(\log(p) + p - 1)\alpha + 2\frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma$$

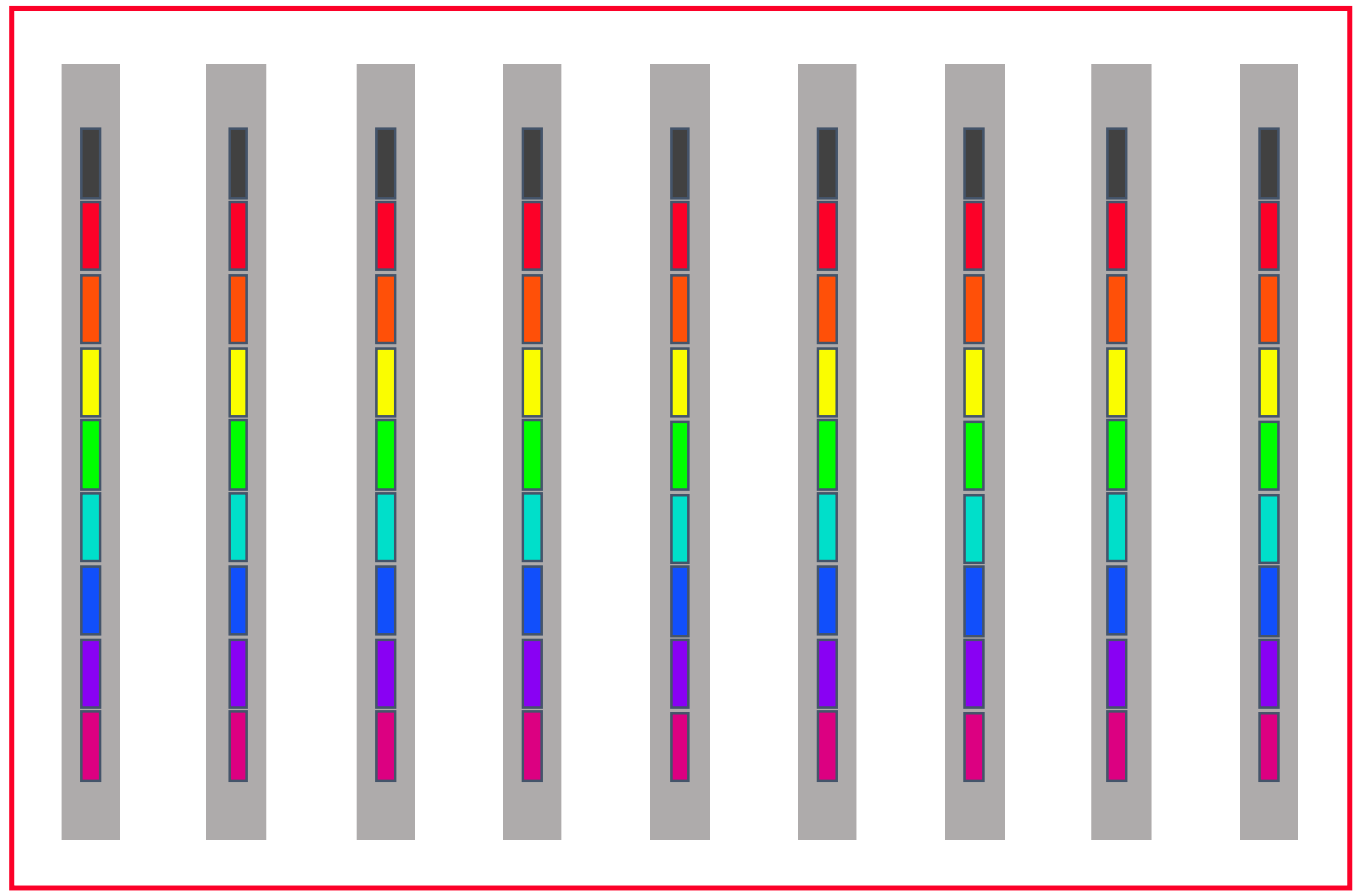
Vs. MST reduce: $$\lceil \log(p) \rceil \left(\alpha + n\beta + n\gamma\right)$$

# Allreduce (Large Message)

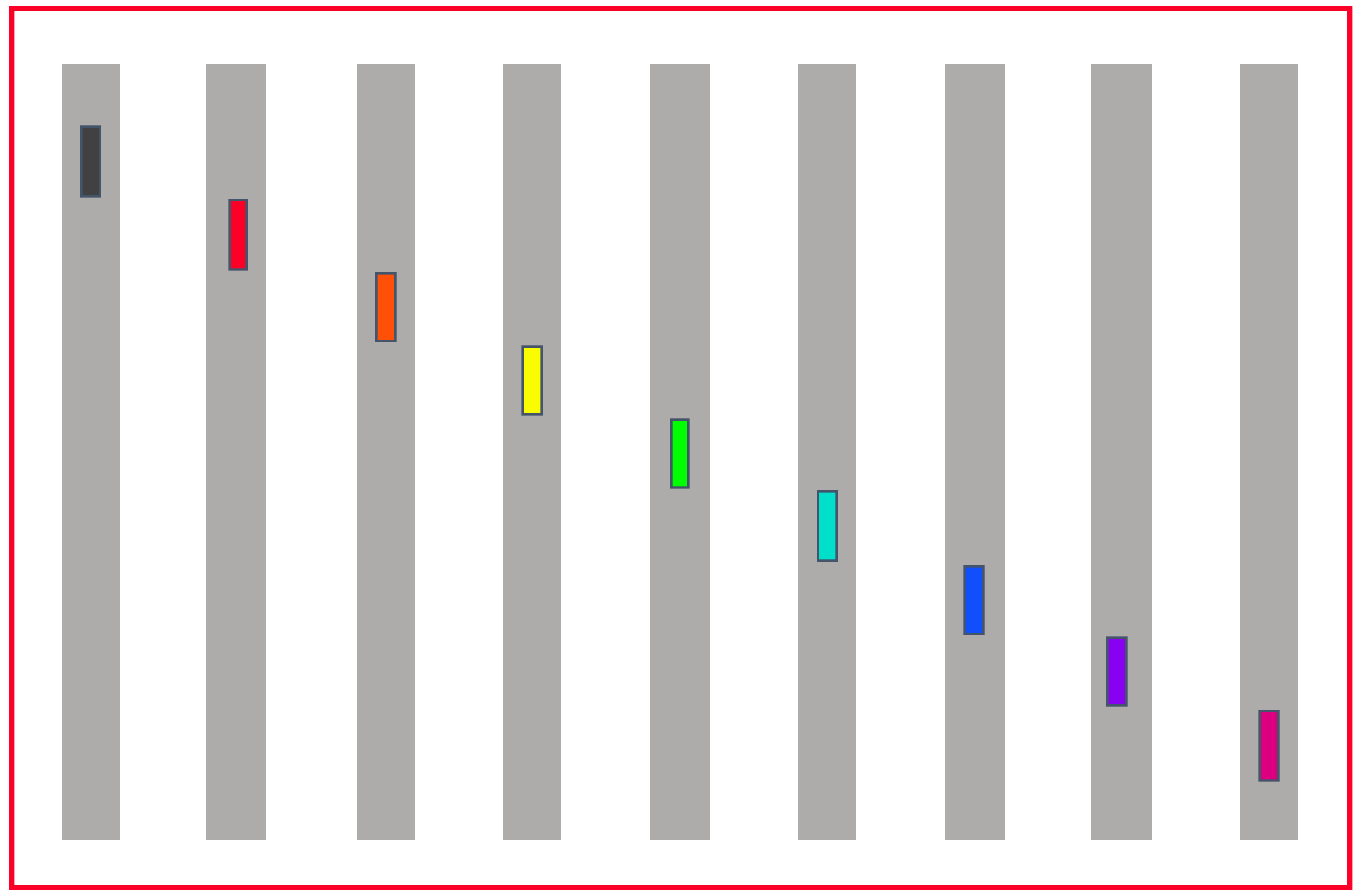


# Allreduce (Large Message)

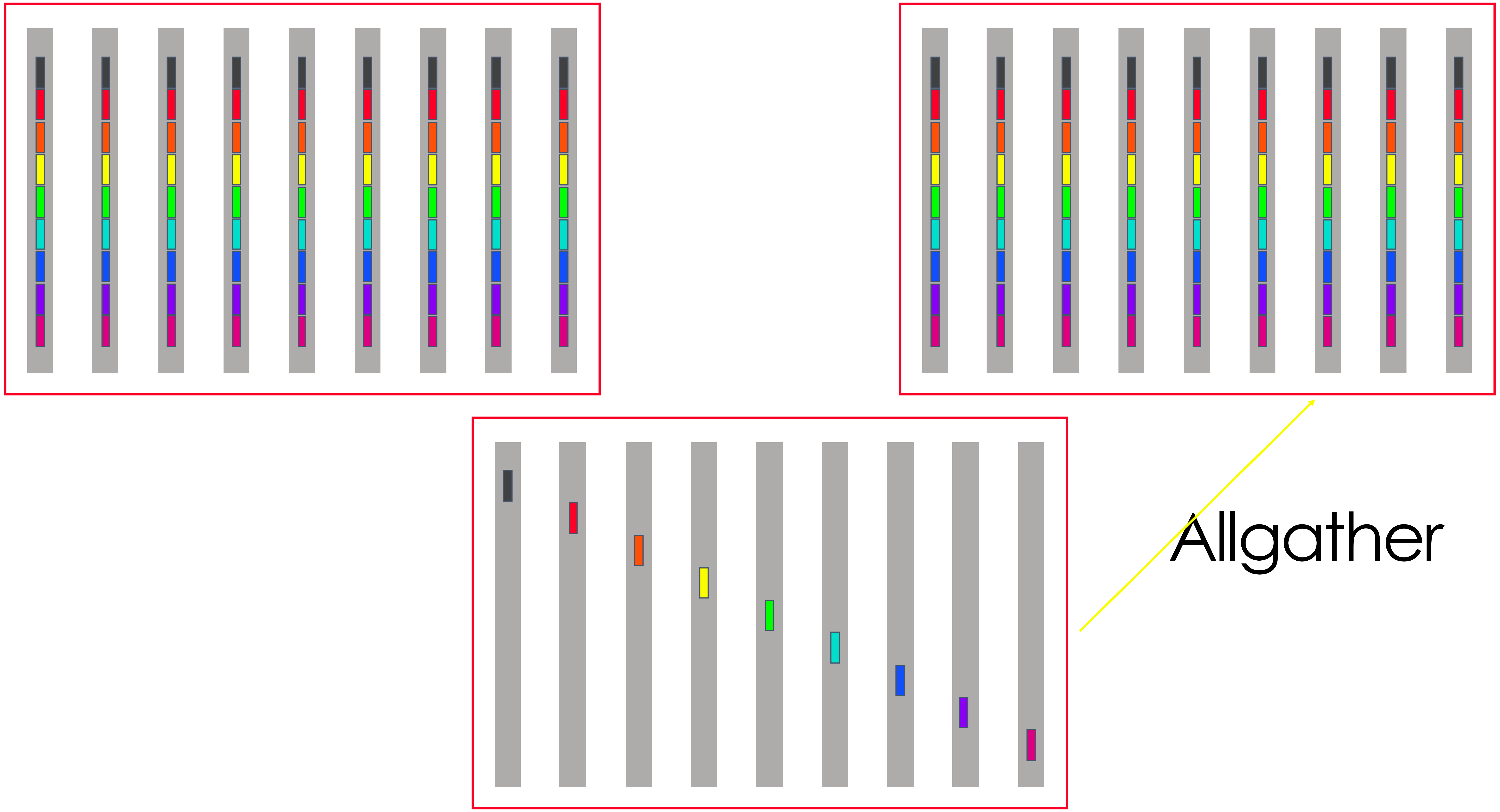




Reduce-scatter



# Allreduce
(long vector)



# Cost of Reduce-scatter/Allgather Allreduce

- Assumption: power of two number of nodes

Reduce-scatter $(p-1)\alpha + \frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma$

Allgather $(p-1)\alpha + \frac{p-1}{p} n\beta$

$$2(p-1)\alpha + 2\frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma$$

## Cost of Reduce-scatter/Allgather Allreduce

- Assumption: power of two number of nodes

Reduce-scatter $$(p-1)\alpha + \frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$$

Allgather $$(p-1)\alpha + \frac{p-1}{p}n\beta$$

---

$$2(p-1)\alpha + 2\frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$$

Vs. Reduce-broadcast allreduce $$2log(p)\alpha + 2log(p)n\beta + log(p)n\gamma$$

# Recap

Reduce-scatter

$$(p-1)\alpha + \frac{p-1}{p} n(\beta+\gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

Allgather

$$(p-1)\alpha + \frac{p-1}{p} n\beta$$

Reduce(-to-one)

Allreduce

Broadcast

# Recap

**Reduce-scatter**

$$(p-1)\alpha + \frac{p-1}{p} n(\beta+\gamma)$$

**Scatter**

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

**Gather**

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

**Allgather**

$$(p-1)\alpha + \frac{p-1}{p} n\beta$$

**Reduce(-to-one)**

$$(p-1+\log(p))\alpha + \frac{p-1}{p} n(2\beta+\gamma)$$

**Allreduce**

**Broadcast**

# Recap

## Reduce-scatter

$$(p-1)\alpha + \frac{p-1}{p} n(\beta+\gamma)$$

## Scatter

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

## Gather

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

## Allgather

$$(p-1)\alpha + \frac{p-1}{p} n\beta$$

## Reduce(-to-one)

$$(p-1+\log(p))\alpha + \frac{p-1}{p} n(2\beta+\gamma)$$

## Allreduce

$$2(p-1)\alpha + \frac{p-1}{p} n(2\beta+\gamma)$$

## Broadcast

$$(\log(p)+p-1)\alpha + 2\frac{p-1}{p} n\beta$$

# Recap

**Reduce-scatter**

$$(p-1)\alpha + \frac{p-1}{p} n(\beta+\gamma)$$

**Scatter**

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

**Gather**

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

**Allgather**

$$(p-1)\alpha + \frac{p-1}{p} n\beta$$

**Reduce(-to-one)**

$$(p-1+\log(p))\alpha + \frac{p-1}{p} n(2\beta+\gamma)$$

**Allreduce**

$$2(p-1)\alpha + \frac{p-1}{p} n(2\beta+\gamma)$$

**Broadcast**

# Recap

## Reduce-scatter

$$(p-1)\alpha + \frac{p-1}{p} n(\beta+\gamma)$$

## Scatter

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

## Gather

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

## Allgather

$$(p-1)\alpha + \frac{p-1}{p} n\beta$$

## Reduce(-to-one)

$$(p-1+\log(p))\alpha + \frac{p-1}{p} n(2\beta+\gamma)$$

## Allreduce

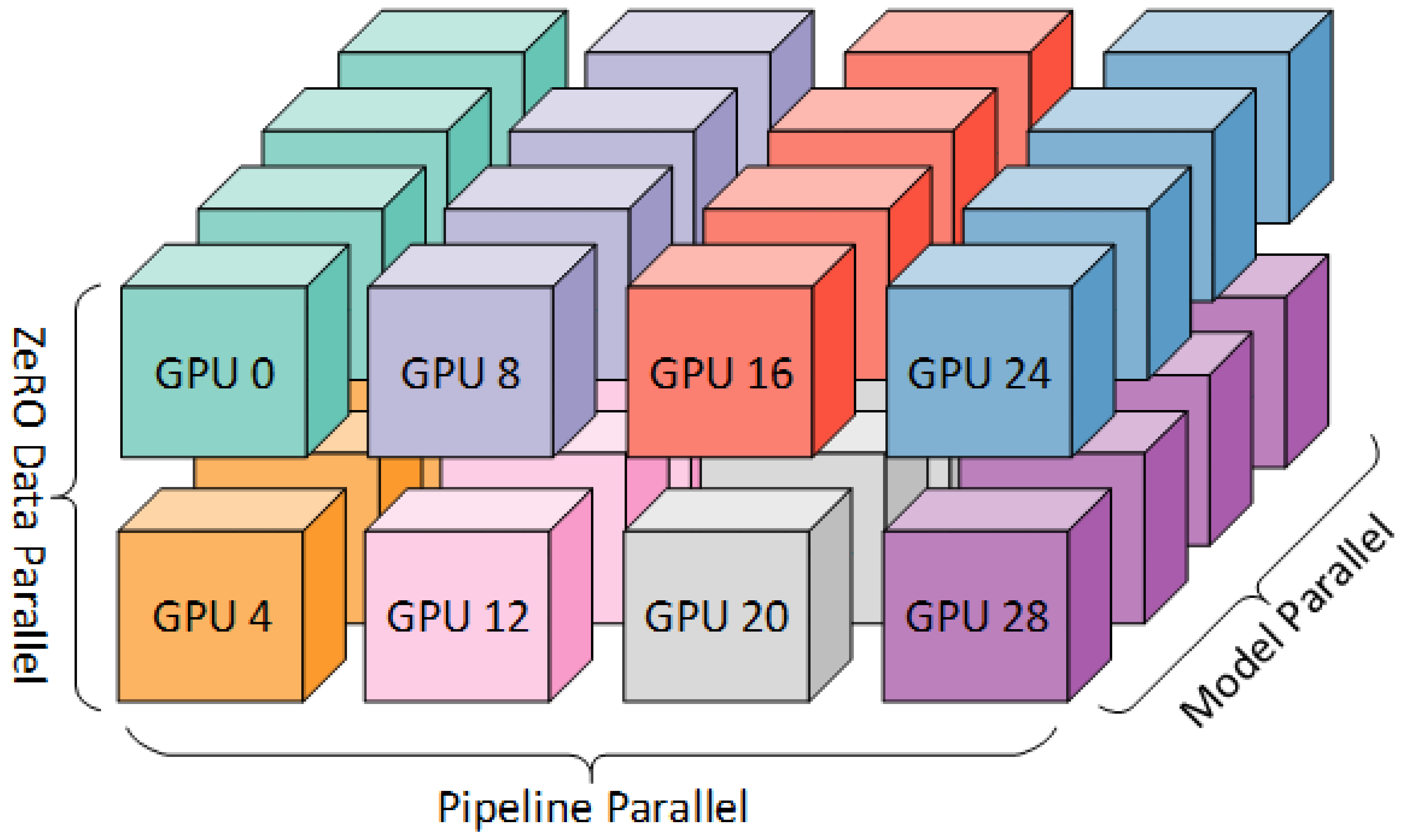
$$2(p-1)\alpha + \frac{p-1}{p} n(2\beta+\gamma)$$

## Broadcast

$$(\log(p)+p-1)\alpha + 2\frac{p-1}{p} n\beta$$

# A More Complicate Case

- Real Cluster to train ChatGPT:
  - If using GPU: 2D Mesh
  - If using TPU: 3D Mesh, see figure below



# Summary and Question

- MST -> when alpha dominates
- Ring -> when n*beta dominates
- 2D can be composed using 1D, 3D can be composed using 2D, …
- Latency / Bandwidth trade-offs

# Recap

- Q1: Which collective primitive maps to the distributed SGD gradient synchronization step?
- Q2: How many messages do we need to transfer over the network for a single iteration of GPT-3 SGD update assuming 8-gpu parallelism?
- Q3: For Q2, assuming 1D mesh, should we use MST or Ring?

# Collective Pros

- A set of structured / well-defined communication primitives
- Extremely well-optimized
- Beautiful math, easy to analyze, and easy to understand its performance

# Collective Cons

- Lack of Fault Tolerance
  - What if one node (in the ring) is dead?
- Requires Homogeneity
  - What if one node computes slower than all other nodes?
  - What if one link has lower bandwidth than the other node?

Real Cluster:

- Need Fault tolerance
- Heterogeneous hardware setup

# Where we are

Motivations, Economics, Ecosystems, Trends

Cloud

Skip this

Networking

~~Datacenter networking~~

~~Collective communication~~

Storage

(Distributed) File Systems / Database

Cloud storage

Compute

Distributed Computing

Big data processing

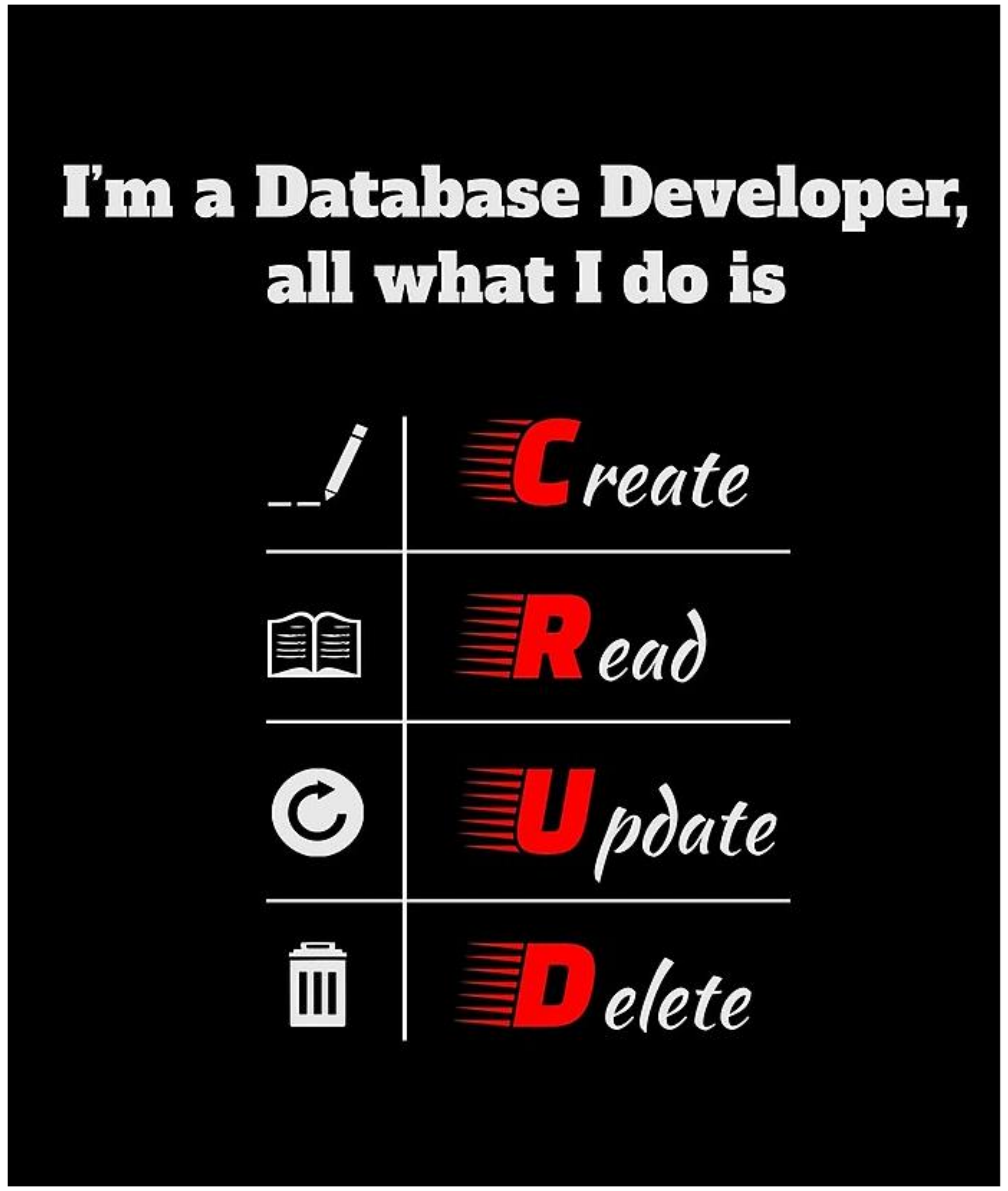
# But Some Basic Knowledge Check

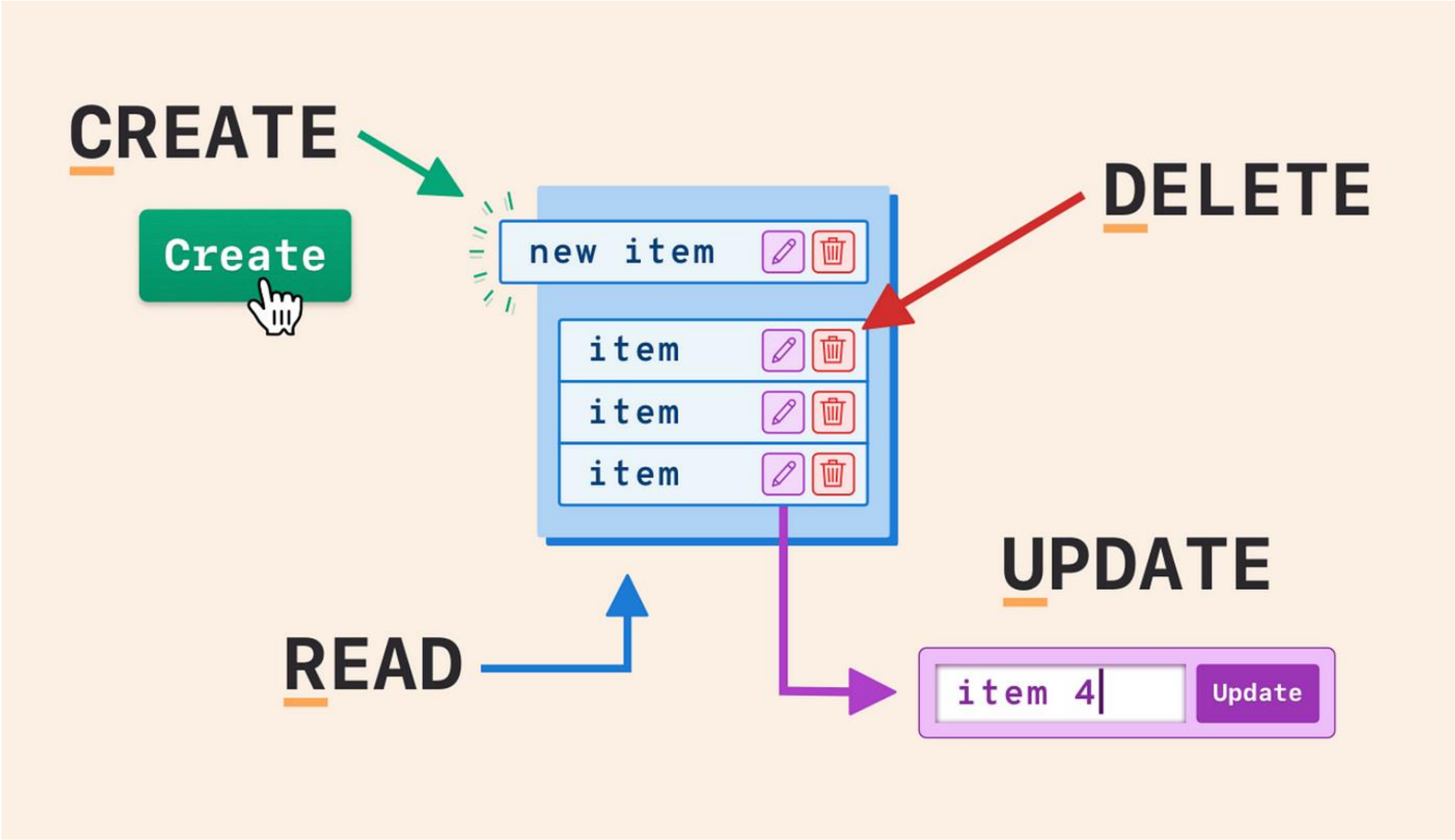
- **What is a Database**
- Have you heard these terms?
  - HashTable
  - SSTable and LSM-Trees
  - BTree?
- Optional readings will cover this – highly recommend to read

# But Some Important Concepts

- OLTP v.s. OLAP
- Data warehousing
- Schemas for Analytics
- Column-oriented storage
- Data cubes and materialized views

# CRUD


I'm a Database Developer,
all what I do is
Create
Read
Update
Delete

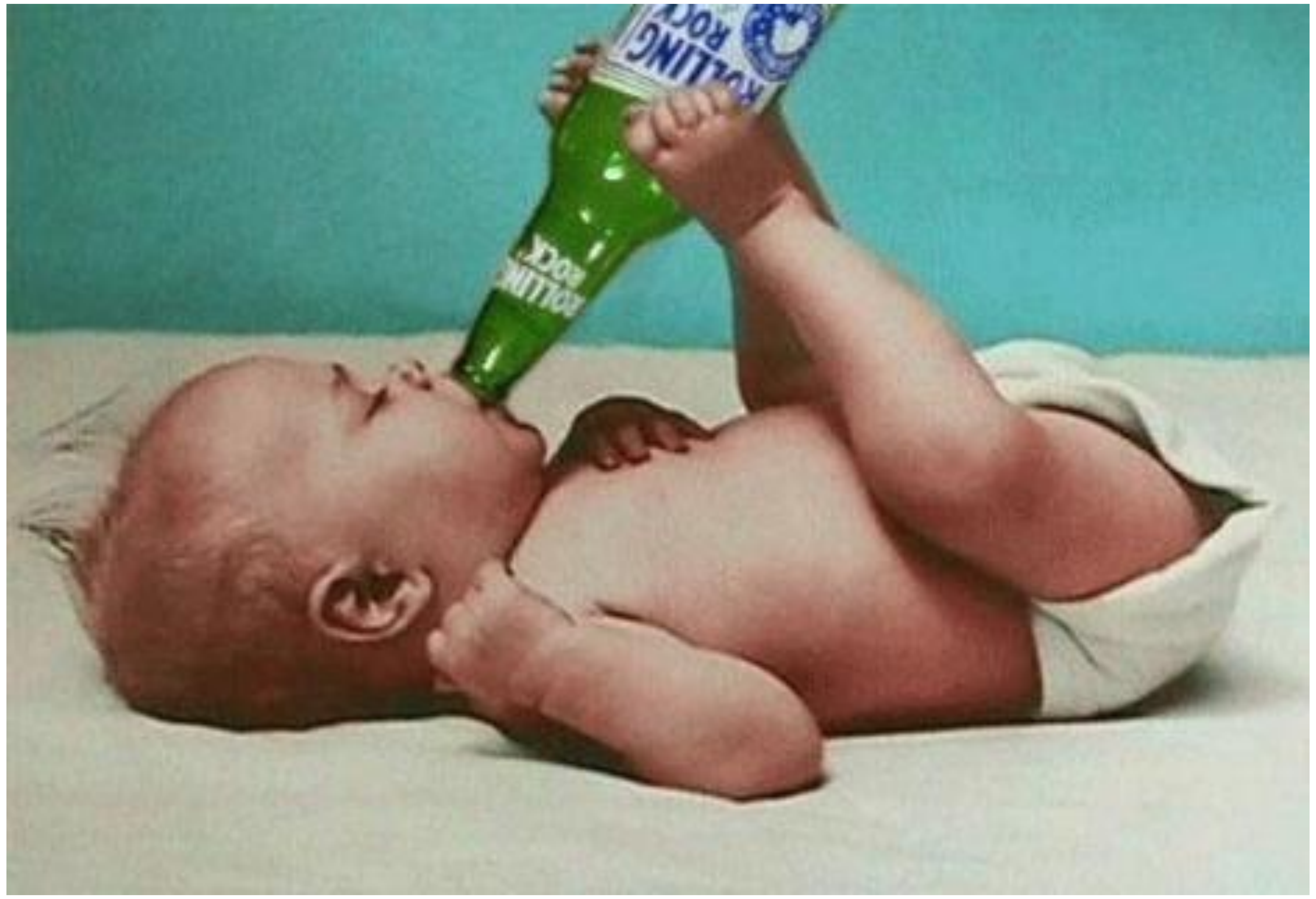

CREATE
Create
new item
item
item
item
DELETE
UPDATE
READ
item 4
Update

# Database transactions

- Make sale
- Place an order
- Pay an employee’s salary
- Comment a blog post
- Act in games
- Add/remove contract to an address book

Online transaction processing (OLTP)

# Walmart Beer and Diaper (1988)




Forbes 1988

- Unexpected correlation:
  - Sales of diapers and beer



https://www.forbes.com/forbes/1998/0406/6107128a.html?sh=2574a9316260

# Data analytics

- What was the total revenue of each of our stores in Jan?
- How many more bananas than usual did we sell during our latest data?
- Which brand of baby food is most often purchased together with brand X diapers?

Online analytic processing (OLAP)

# OLTP v.s. OLAP

| Property | Transaction processing systems (OLTP) | Analytic systems (OLAP) |
|---|---|---|
| Main read pattern | Small number of records per query, fetched by key | Aggregate over large number of records |

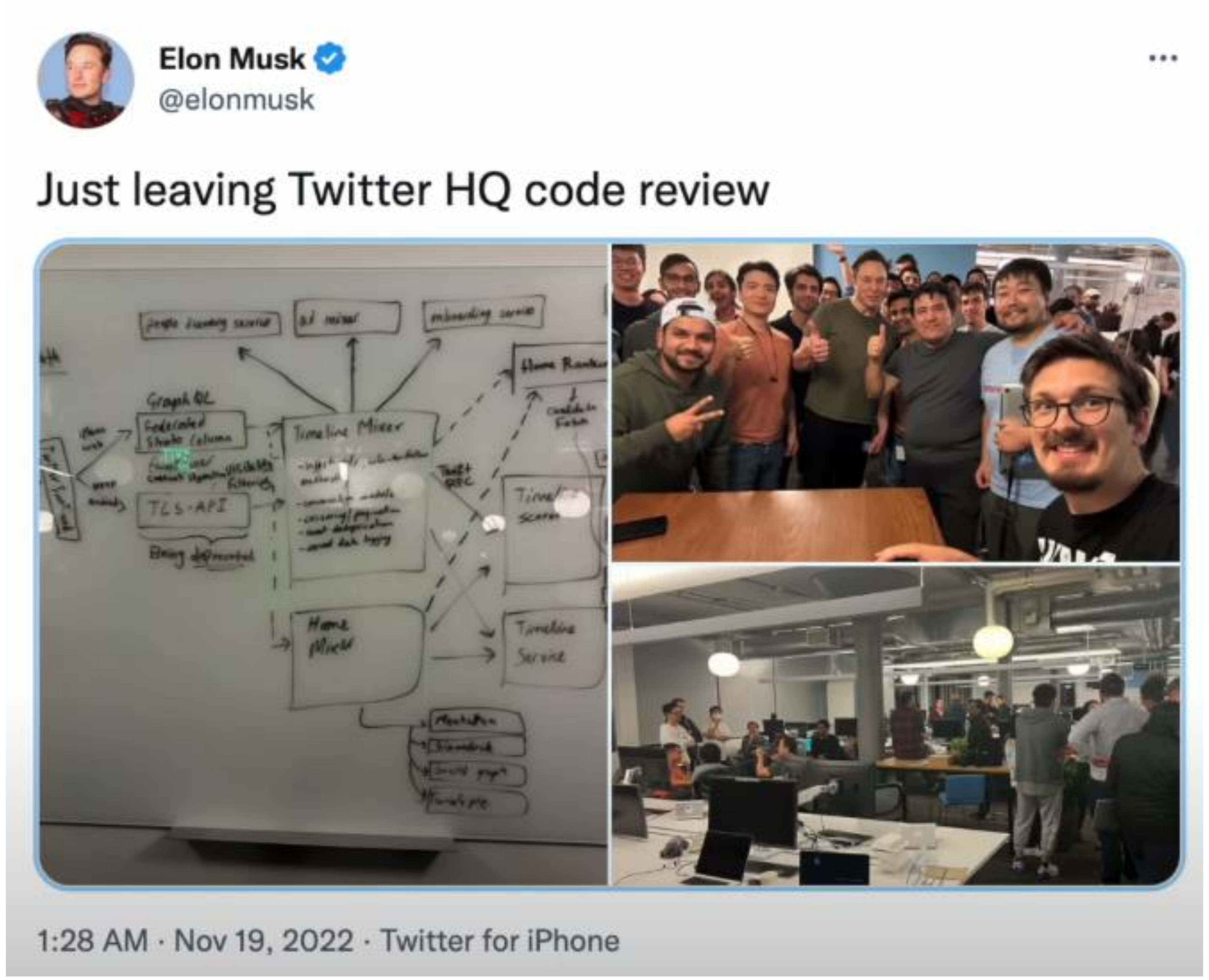
# OLTP v.s. OLAP

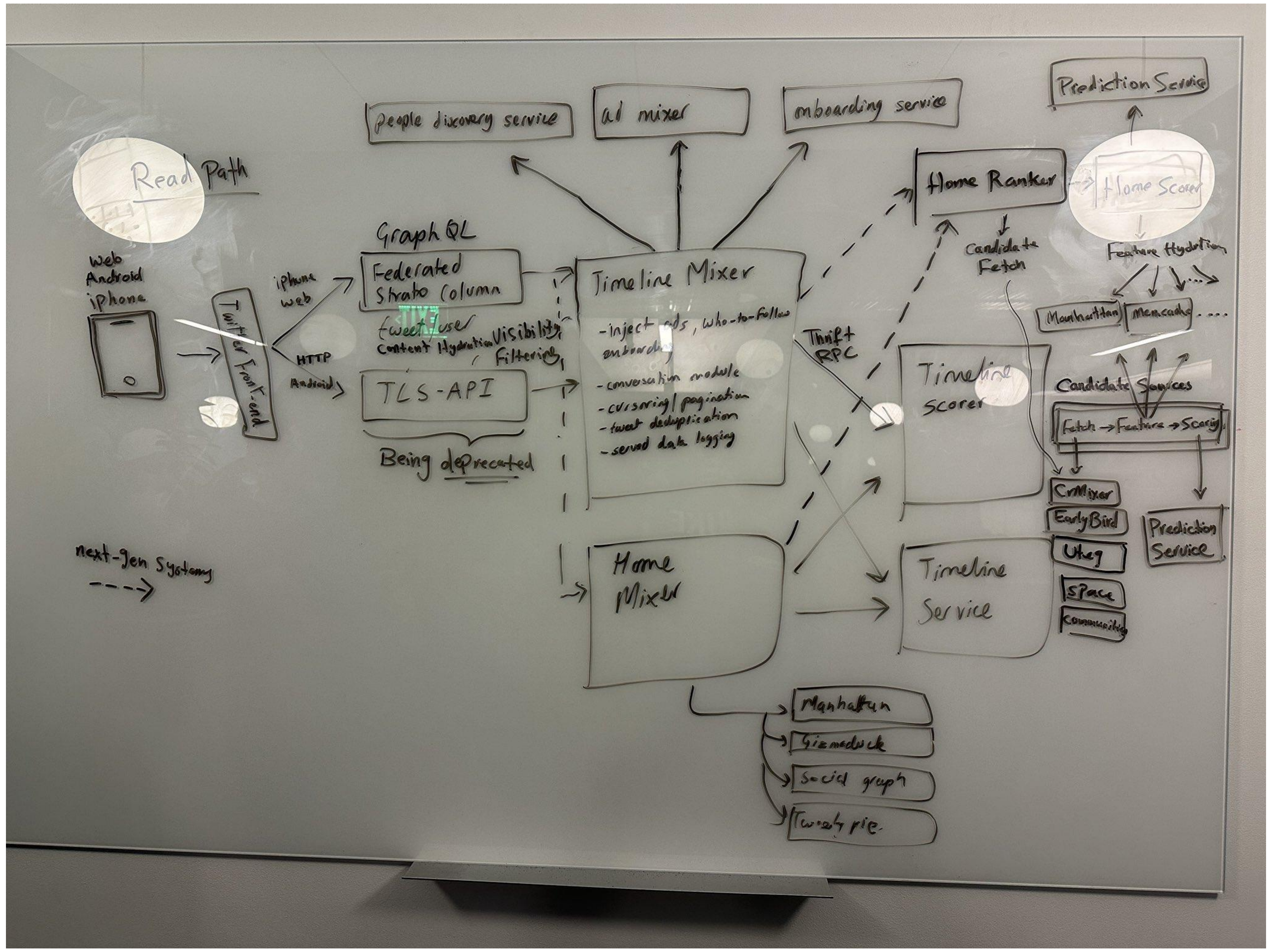
| Property | Transaction processing systems (OLTP) | Analytic systems (OLAP) |
|---|---|---|
| Main read pattern | Small number of records per query, fetched by key | Aggregate over large number of records |
| Main write pattern | Random-access, low-latency writes from user input | Bulk import (ETL) or event stream |
| Primarily used by | End user/customer, via web application | Internal analyst, for decision support |
| What data represents | Latest state of data (current point in time) | History of events that happened over time |
| Dataset size | Gigabytes to terabytes | Terabytes to petabytes |

# Today’s topic

- OLTP v.s. OLAP
- Data warehousing
- Schemas for Analytics
- Column-oriented storage

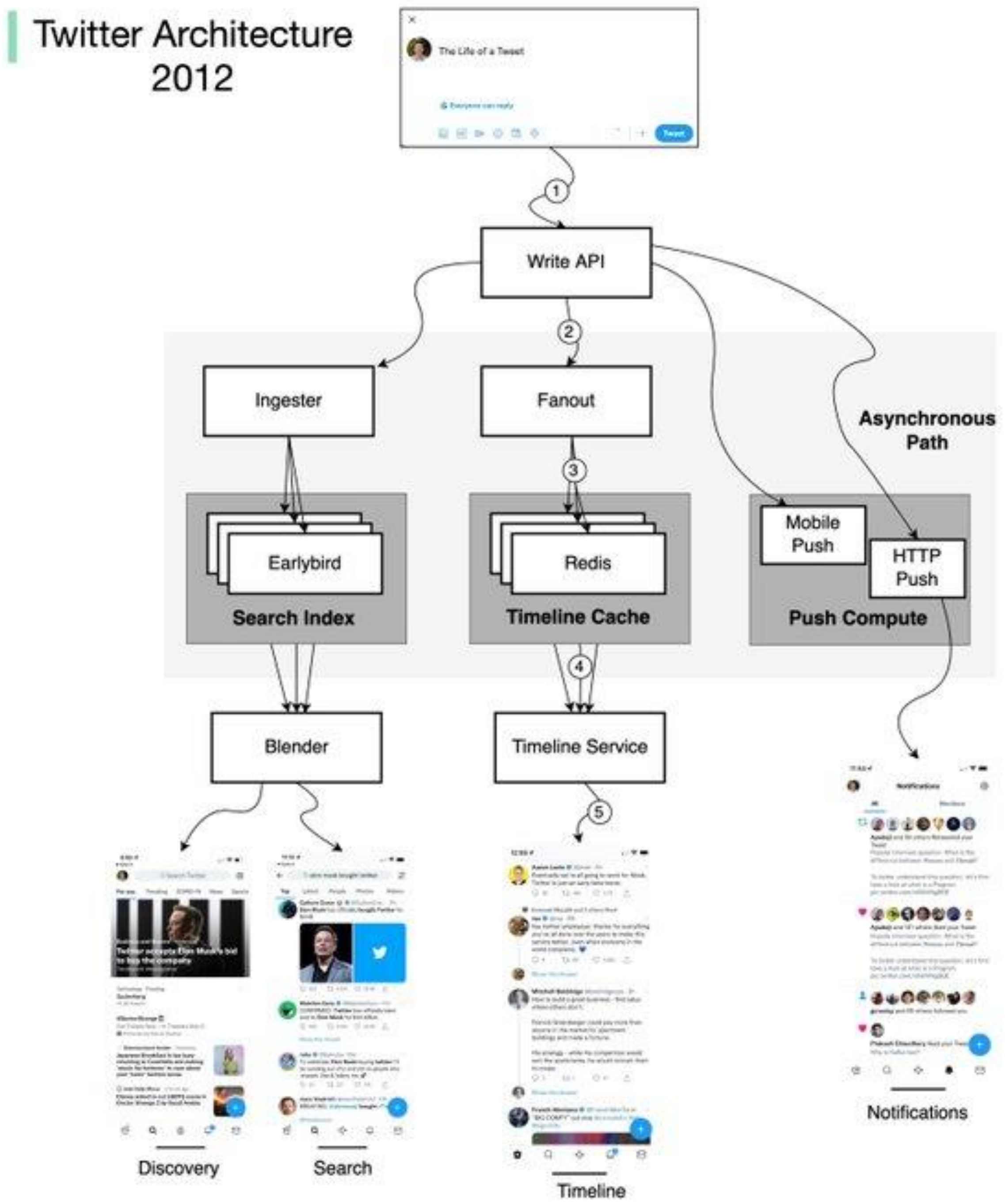
# Transaction systems are complex.

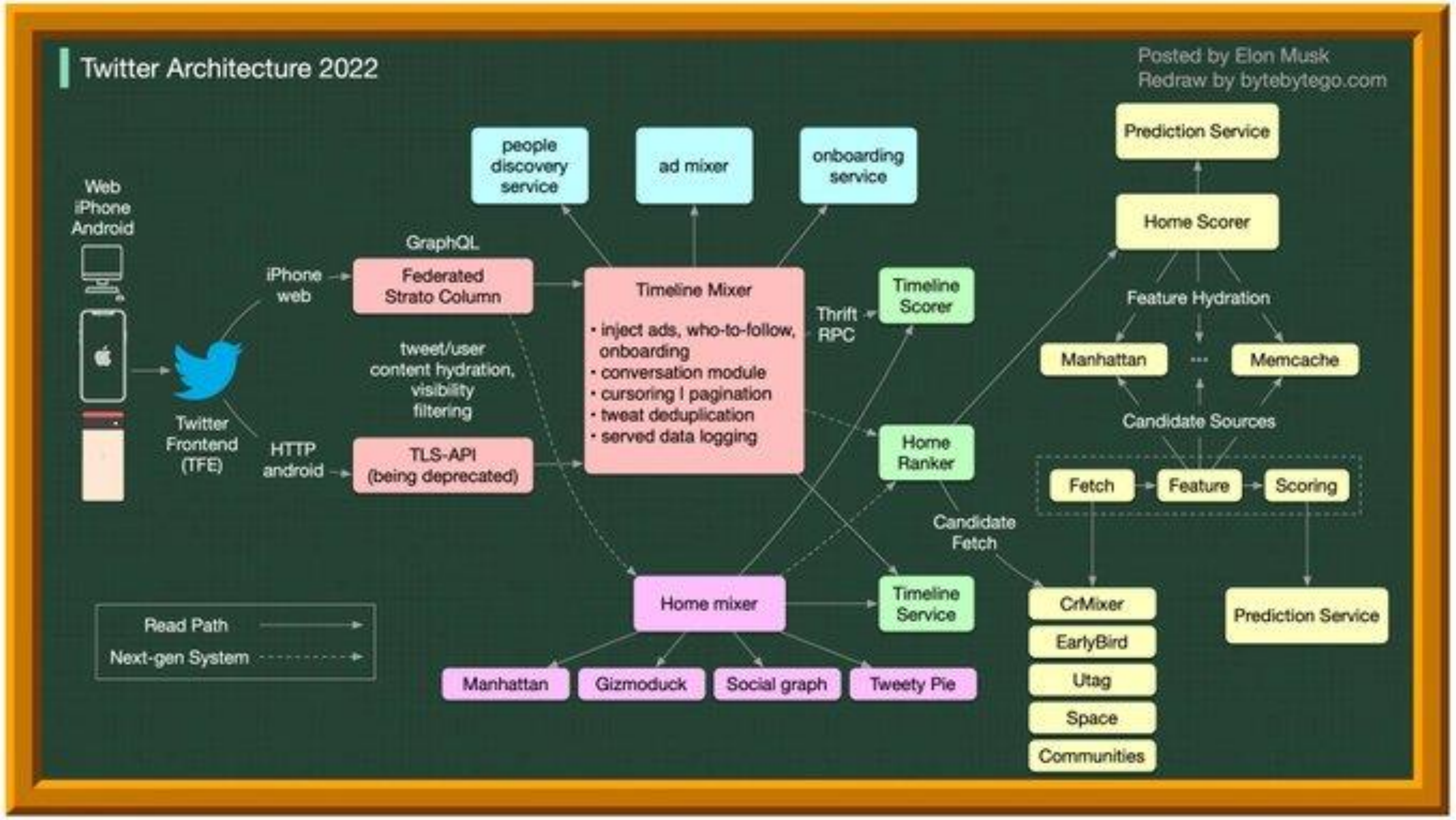



Elon Musk's Twitter System Design Diagram Explained
https://www.youtube.com/watch?v=_Y5aGCOkymQ

# Transaction systems need to be highly available.



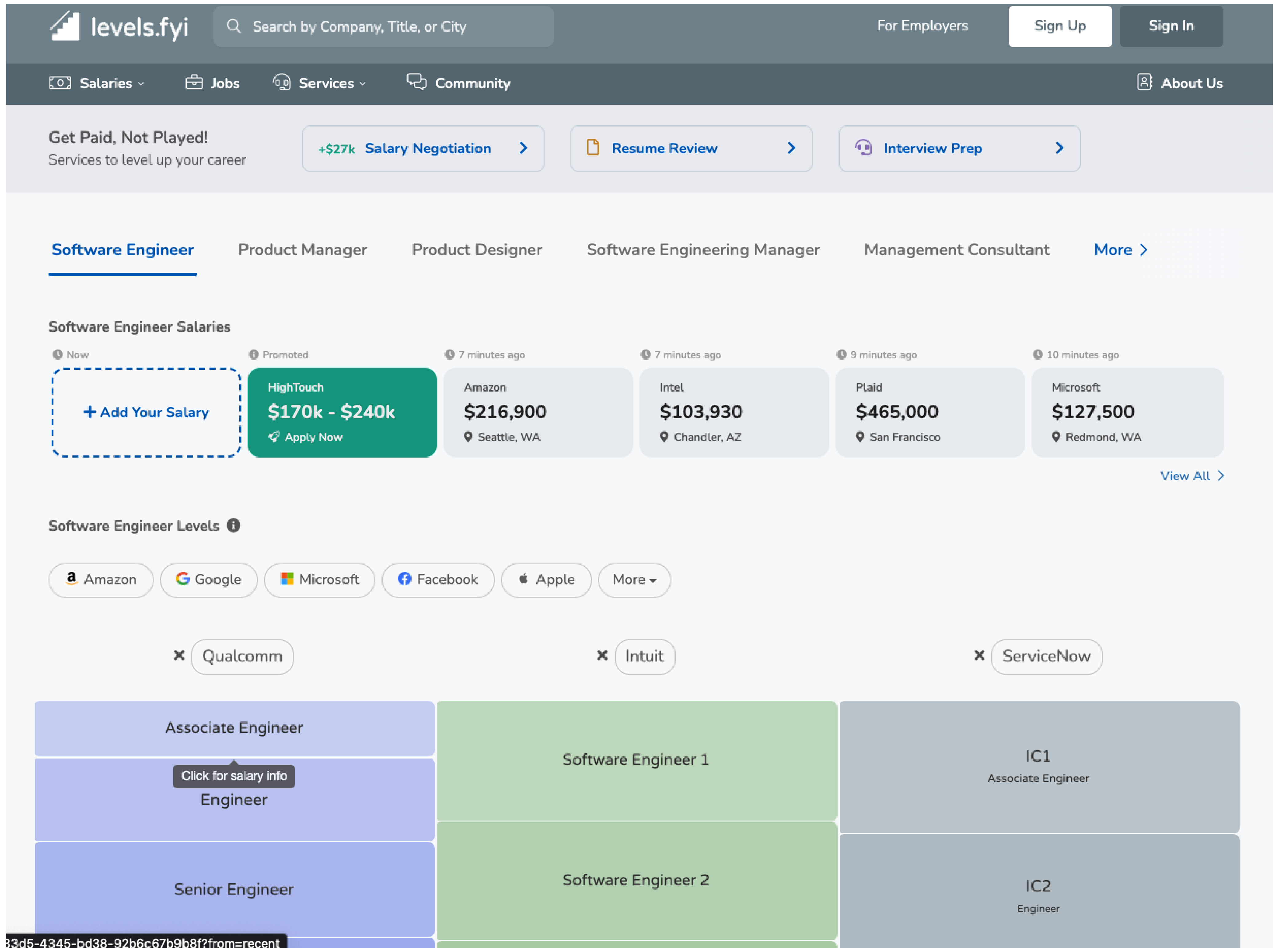

- Low latency.
- Highly available.
- Ad hoc analytic queries are expensive.

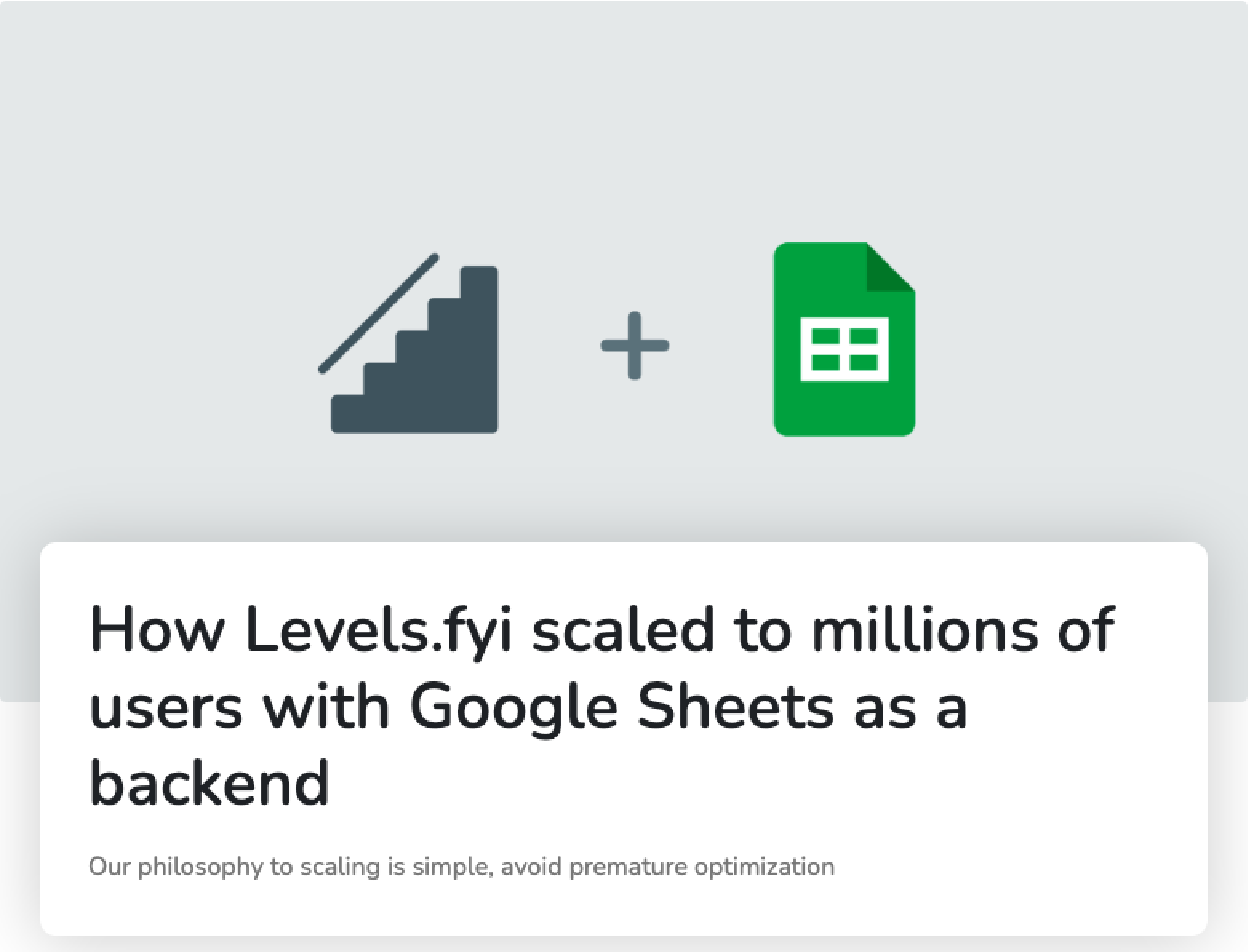
https://twitter.com/alexxubyte/status/1594008281340530688

# Data warehouse

- A separate database that analysts can query to their hearts' content, without affecting OLTP operations.
- Maintain a read-only copy for analytic purposes.
- Only exist in almost all large enterprises.

# Small companies?

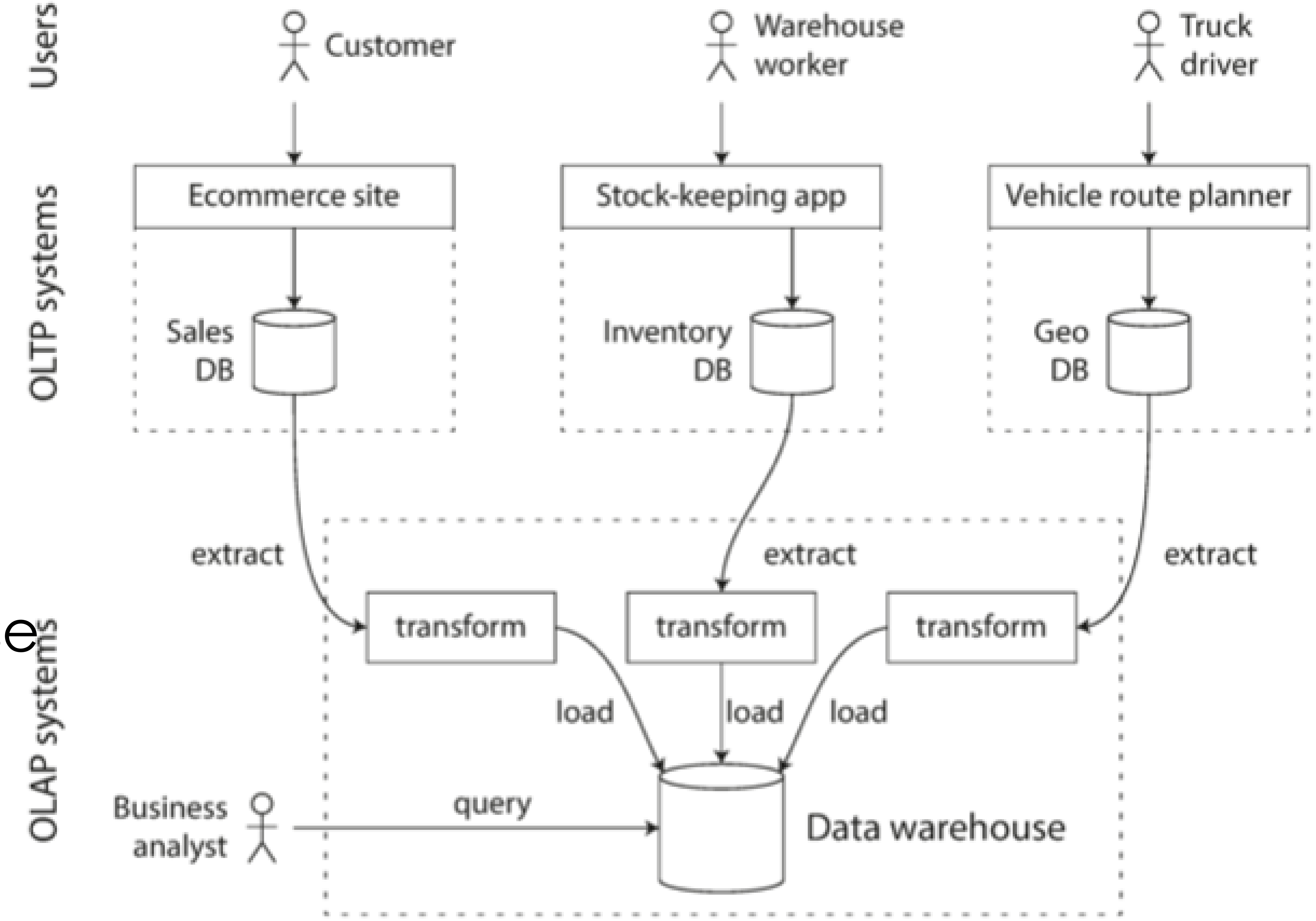



https://www.levels.fyi/blog/scaling-to-millions-with-google-sheets.html

# Extract-Transform-Load (ETL)

- Extract
  - Periodica data dump
  - Continuous streaming
- Transform
  - Analysis-friendly schema
  - Data cleaning
- Load into a data warehouse



# Why data warehouse?

- Separation of concerns
  - Performance (reliability, latency)
  - Expertise requirement, management
- The classic indexes (e.g., SSTable, B-tree) are good for reading and writing a single record.
  - But are not good at answering analytic queries.

# How do you interact with OLAP & OLTP

- SQL query interface
  - *Select * from*
  - "A database system can be considered mature when it has an SQL query interface".
  - Both OLAP and OLTP
- OLAP:
  - More and more codeless user interfaces.
  - Text2SQL
  - Note: This is a big market of innovations

# Sumary

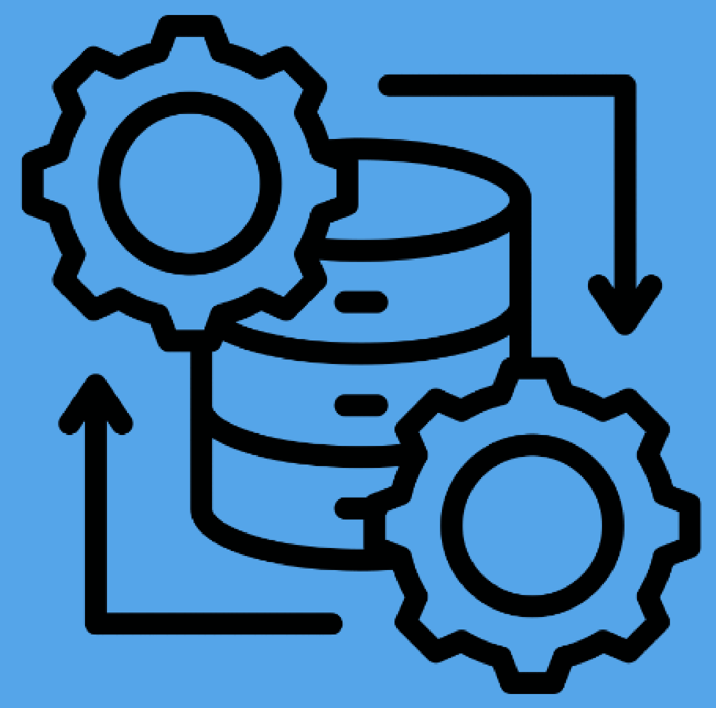
## OLAP

- Analytical
- Show queries
- Denormalised
- Historical Data

**BUSINESS DATA WAREHOUSE**

## OLTP

- Transactional
- Fast Processing
- Normalised
- Current Data



**BUSINESS PROCESS**

# More Stories?

TRIFACTA

400M

Streamlit

800M / ~30 persons

databricks

100B

snowflake®

83B

# Where We Are

Motivations, Economics, Ecosystems, Trends

Cloud

Networking

~~Datacenter networking~~

~~Collective communication~~

Storage

(Distributed) File Systems / Database

Cloud storage

**Part3: Compute**

Distributed Computing

Big data processing

# Where We Are

| | |
|---|---|
| Machine Learning Systems | |
| Big Data | 2010 - Now |
| Cloud | 2000 - 2016 |
| Foundations of Data Systems | 1980 - 2000 |

# Distributed Computing and Big Data

- Parallelism Basics
- Data Replication and partitioning
- Batched Processing
- Streaming Processing

# Today's topic: Parallelism

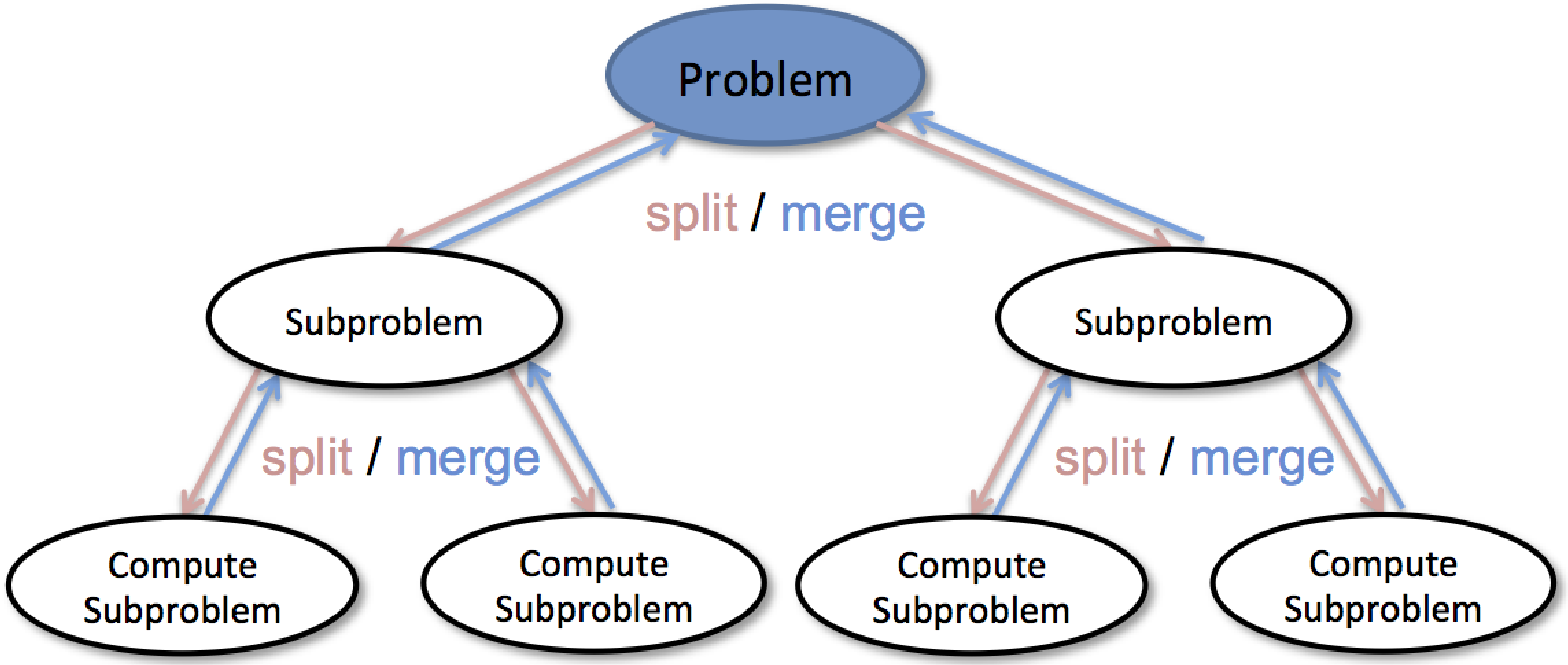
- Express data processing in abstraction
- Parallelisms

# Parallel Data Processing

**Central Issue:** Workload takes too long for one processor!

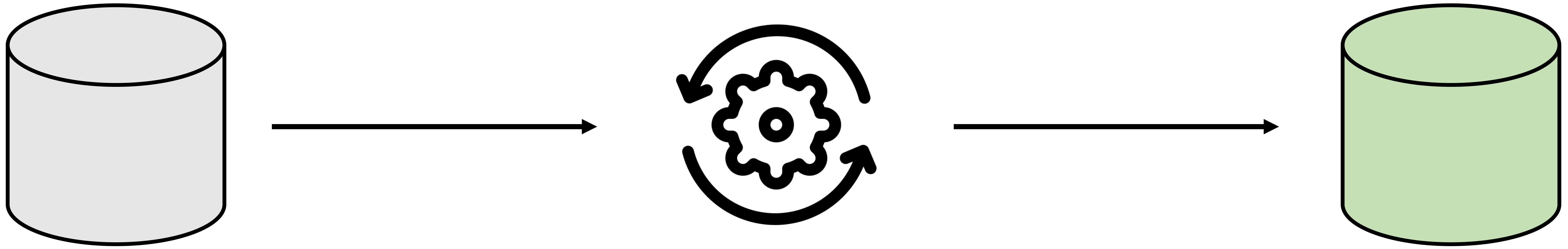
**Basic Idea:** Split up workload across processors and perhaps also across machines/workers (aka "Divide and Conquer")

Remind you of PA1 (hope you've enjoyed it)




https://medium.com/cracking-the-data-science-interview/divide-and-conquer-algorithms-b135681d08fc

# Data Processing: Abstraction



Original data

Processing functions

- sum, mean
- Page rank
- Supervised Learning
- Clustering
- Model inference

Result data

- data
- ML models

Q: How to represent various processing functions?

# How to Express Arbitrarily Complex Processing Functions?

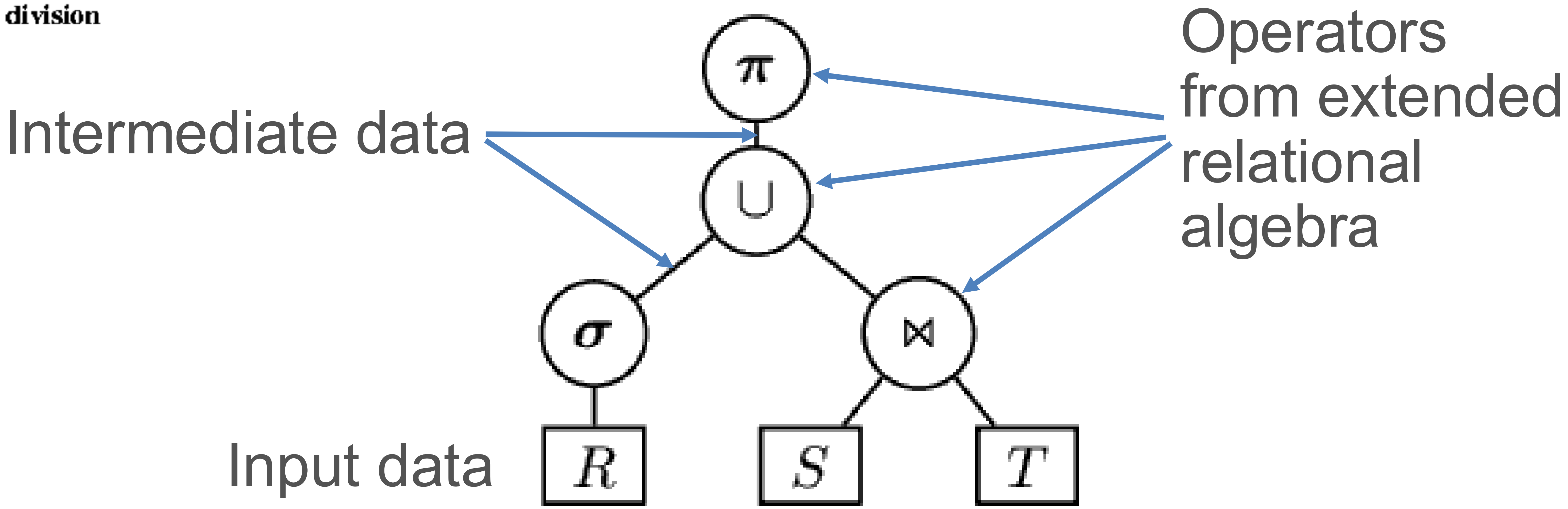
***Dataflow Graph*:** common in parallel data processing

- A ***directed*** graph representation of a program
  - **Vertices:** abstract operations from a restricted set of computational primitives:
  - **Edges:** data flowing directions (hence data dependency)
- Examples
  - Relational dataflows: RDBMS, Pandas, Modin
  - Matrix/tensor dataflows: NumPy, PyTorch, TensorFlow
- Enables us to reason about data-intensive programs at a higher level

# Example: Relational Dataflow Graph

| | | | |
|---|---|---|---|
| $\cup$ | set union | $\sigma$ | selection |
| $\cap$ | set intersection | $\pi$ | projection |
| $-$ | set difference | $\bowtie$ | join |
| $\times$ | cartesian product | $\div$ | set division |

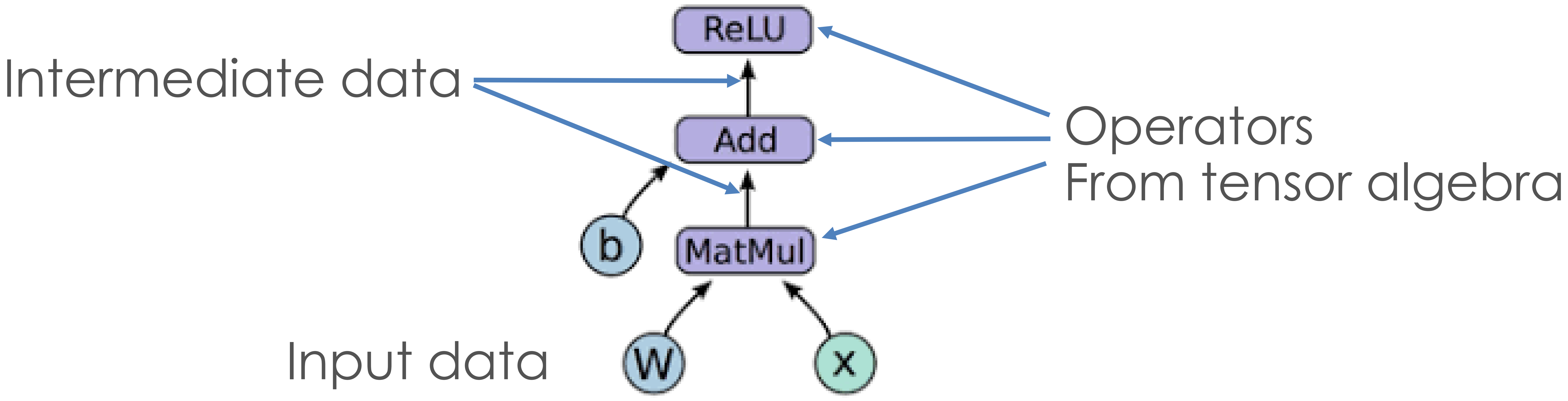
$$\pi(\sigma(R) \cup S \bowtie T)$$




Aka **Logical Query Plan** in the DB systems world

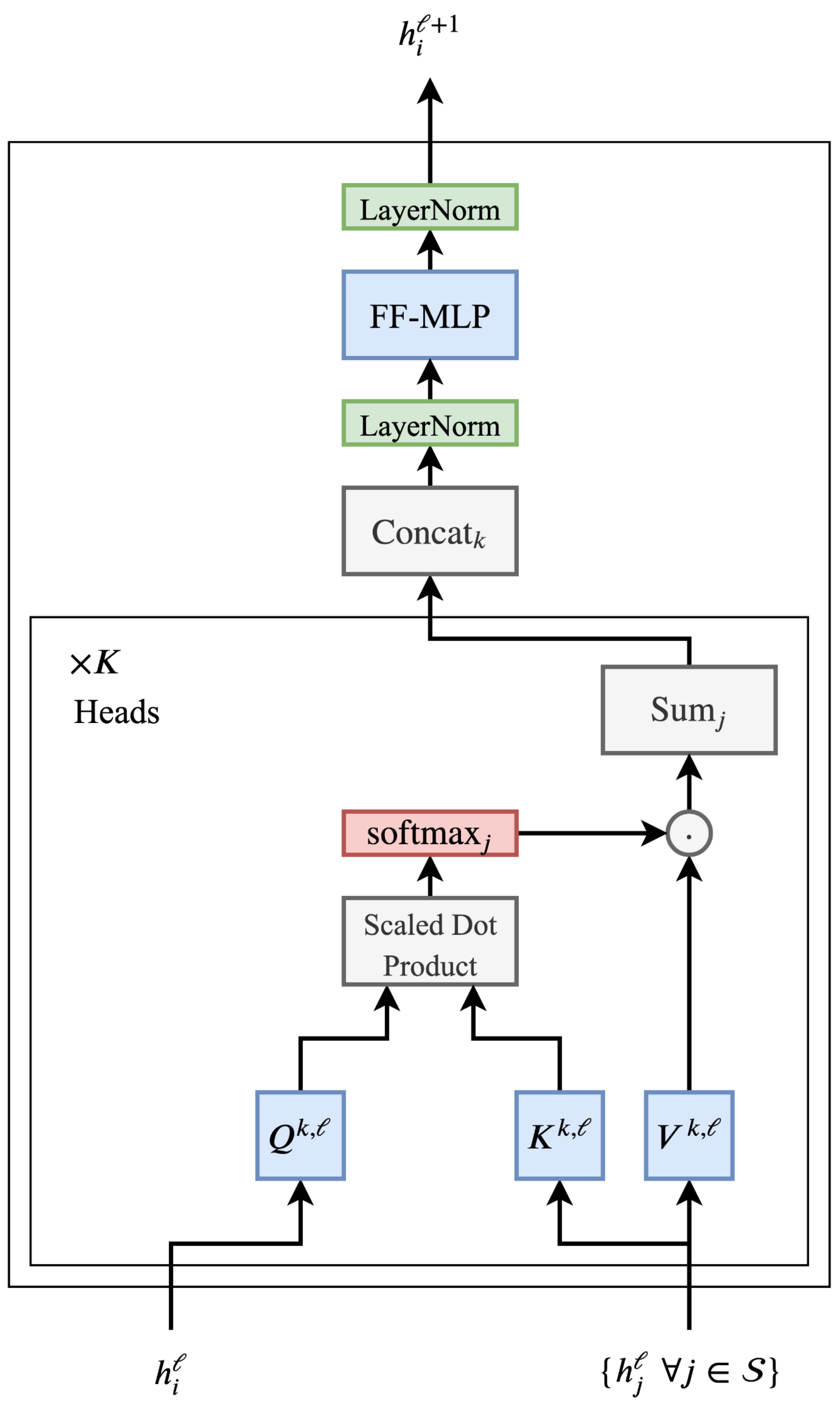
# Example: Machine Learning Dataflow Graph

$$ReLU(WX + b)$$




Aka **Neural network computational graph** in ML systems

# What is ChatGPT's dataflow graph Looking like?



# Parallelism

**Central Issue:** Workload takes too long for one processor!

**Basic Idea:** Split up workload across processors and perhaps also across machines/workers (aka "Divide and Conquer")

**Key parallelism paradigms in data systems**

- **assuming there will be coordination:**

| func \ data | **Shared** | **Replicated** | **Partitioned** |
|---|---|---|---|
| **Replicated** | N/A (rare cases) | N/A (rare cases) | Data parallelism |
| **Partitioned** | Task parallelism | Task parallelism | Hybrid parallelism |