SeedTree: A Dynamically Optimal And Local Self-Adjusting Tree

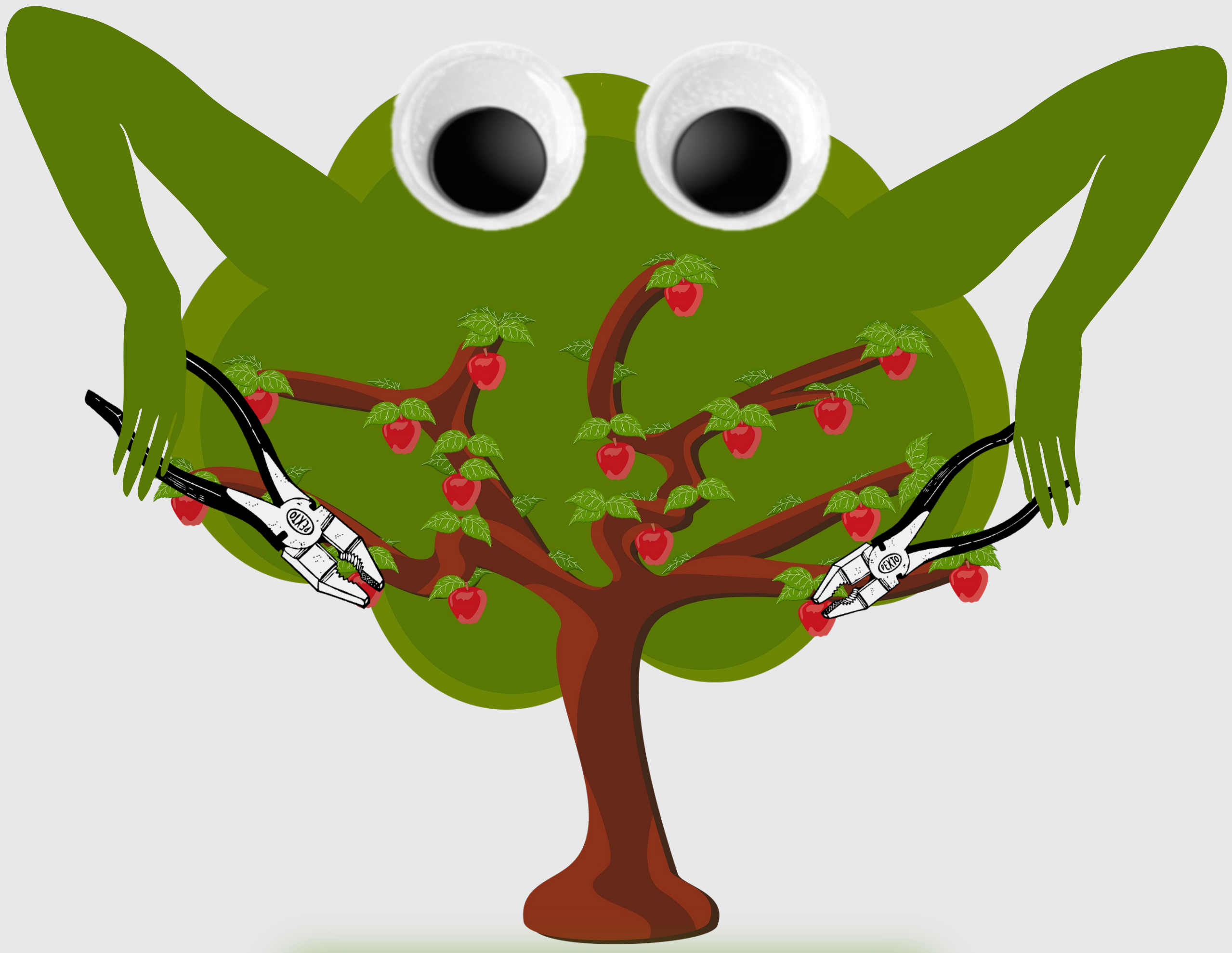
Arash Pourdamghani

Joint work with Chen Avin, Robert Sama, Stefan Schmid

Dutch Optimization Seminar

29 June 2023

An adopted version of INFOCOM’23 Talk



SeedTree at a glance

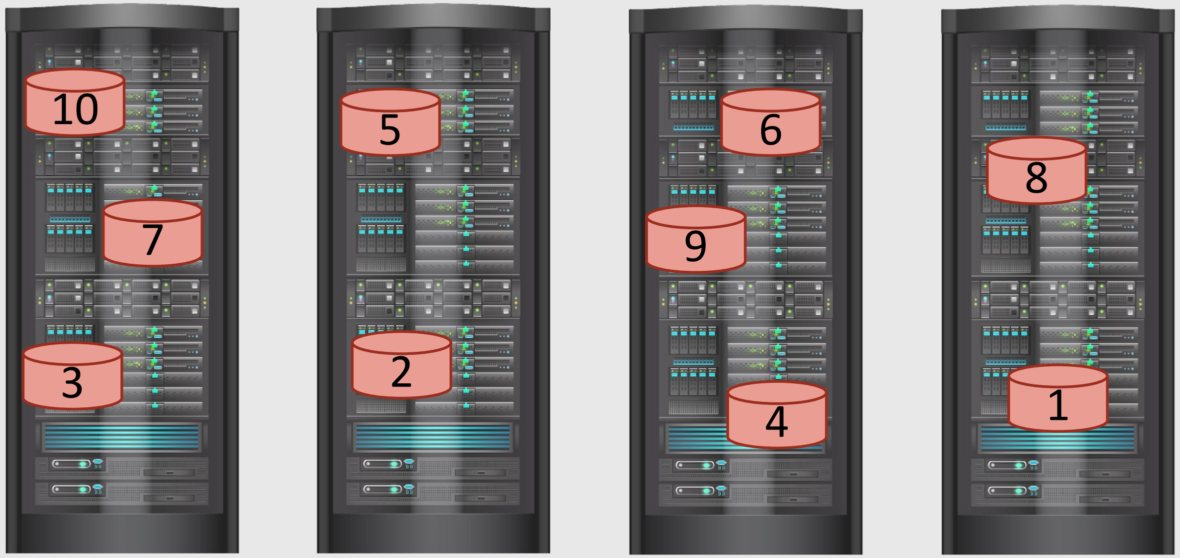
SeedTree

1) Why? 2) What? 3) How?

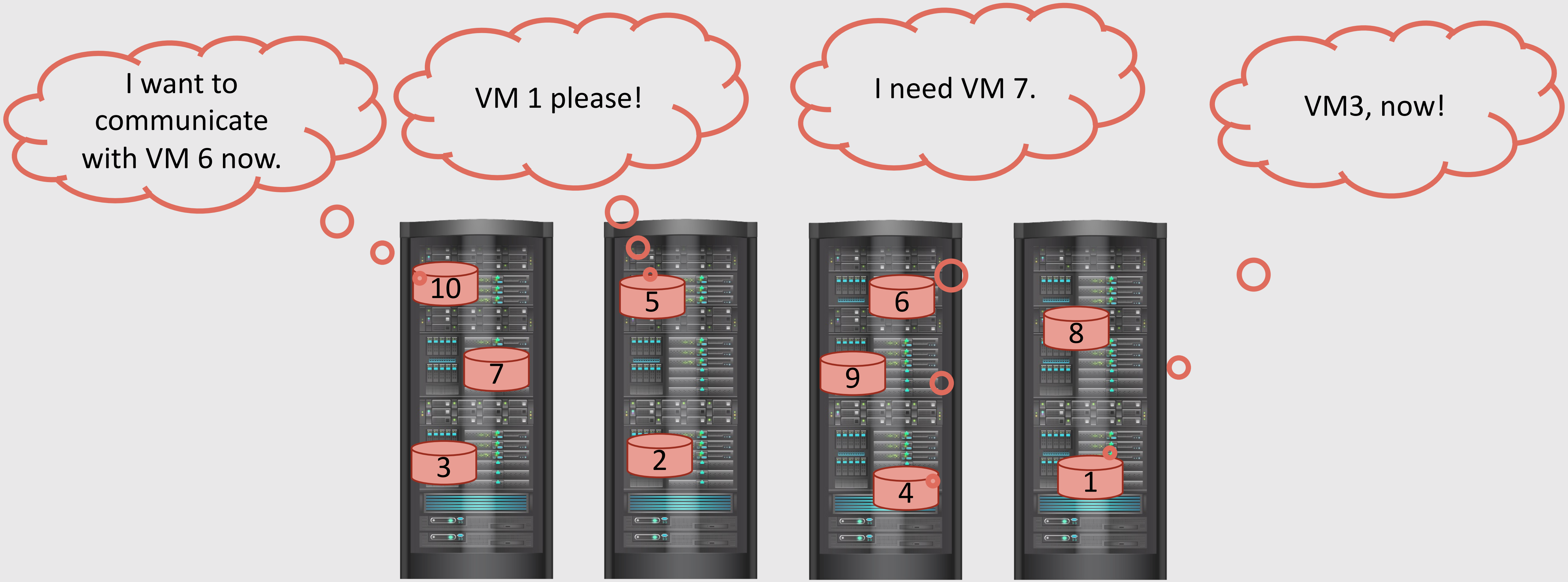
SeedTree

1) Why? 2) What? 3) How?

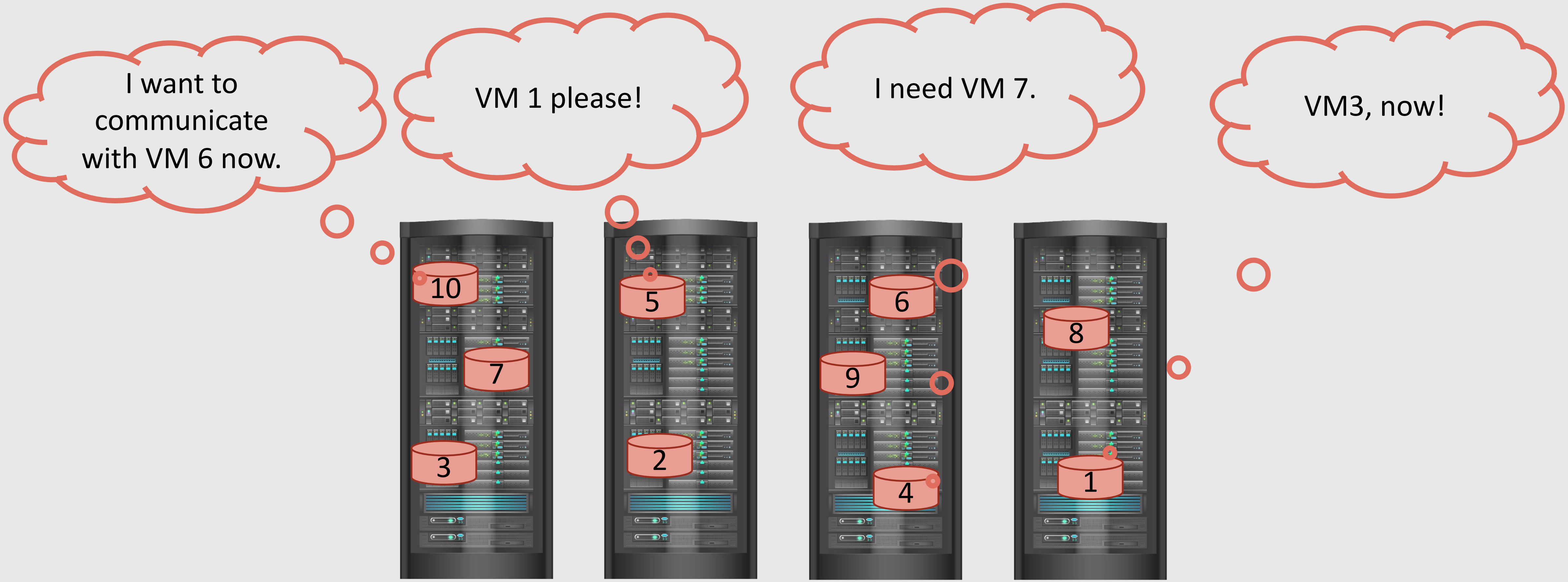
Servers and VMs



Online Request Sequence

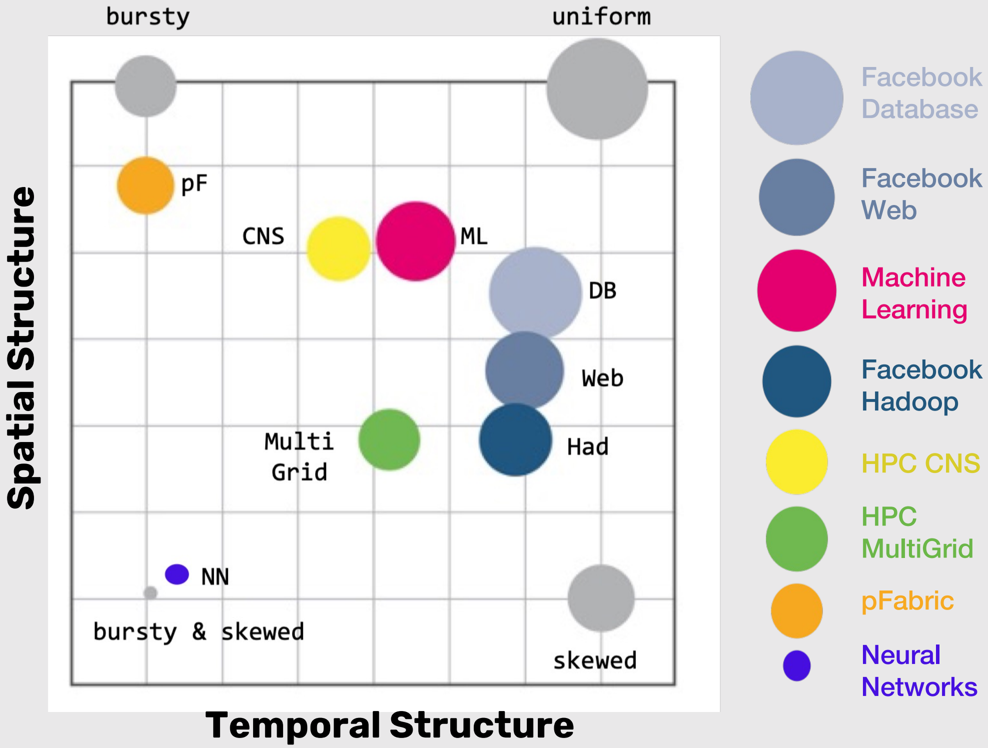


Online Request Sequence



Do we have any structure in the demand?

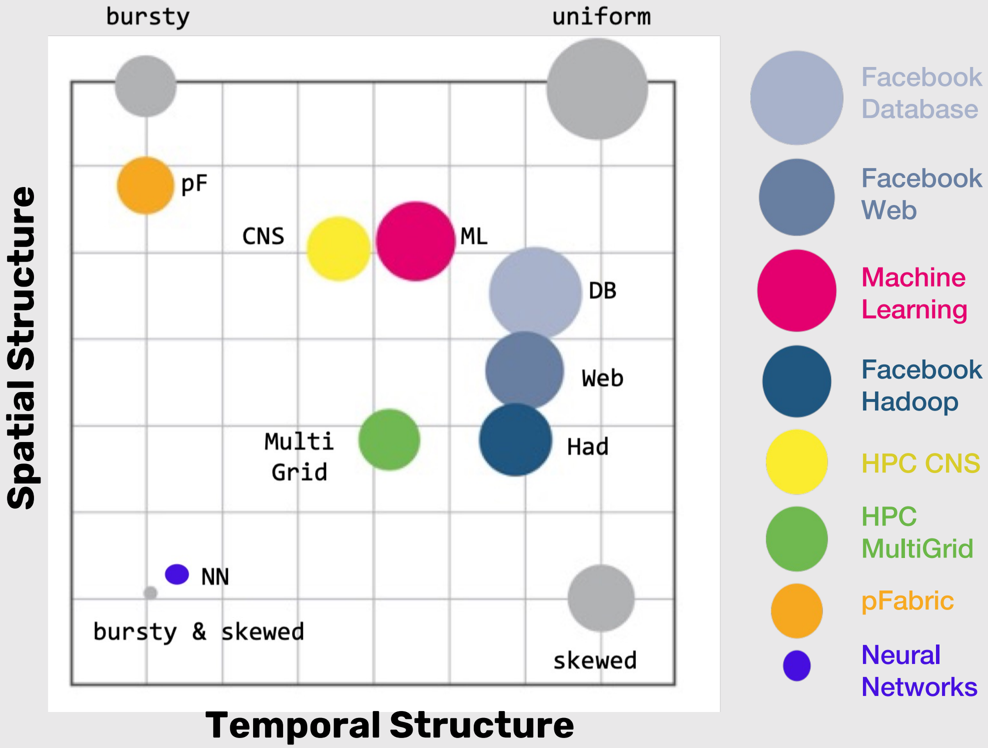
Structure in The Demand



On the Complexity of Traffic Traces and Implications
Avin et al., ACM SIGMETRICS 2020

Structure in The Demand

Can we design a self-adjusting network that utilizes demand?

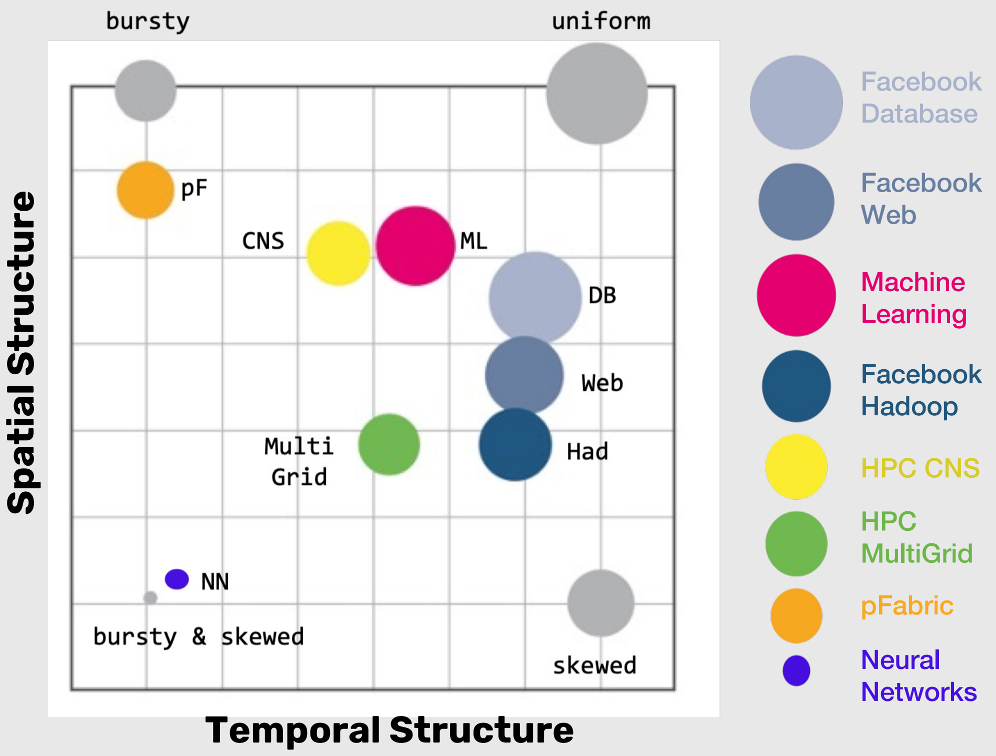


On the Complexity of Traffic Traces and Implications
Avin et al., ACM SIGMETRICS 2020

Structure in The Demand

Can we design a self-adjusting network that utilizes demand?

Let us start by a dynamically optimal self-adjusting tree!



On the Complexity of Traffic Traces and Implications
Avin et al., ACM SIGMETRICS 2020

SeedTree

1) Why? 2) What? 3) How?

Requests From A Single Source

Using Local Routing (i.e., Without A Routing Table)!

Considering A Binary Tree Structure on Servers

Allowing Item Movements on Edges

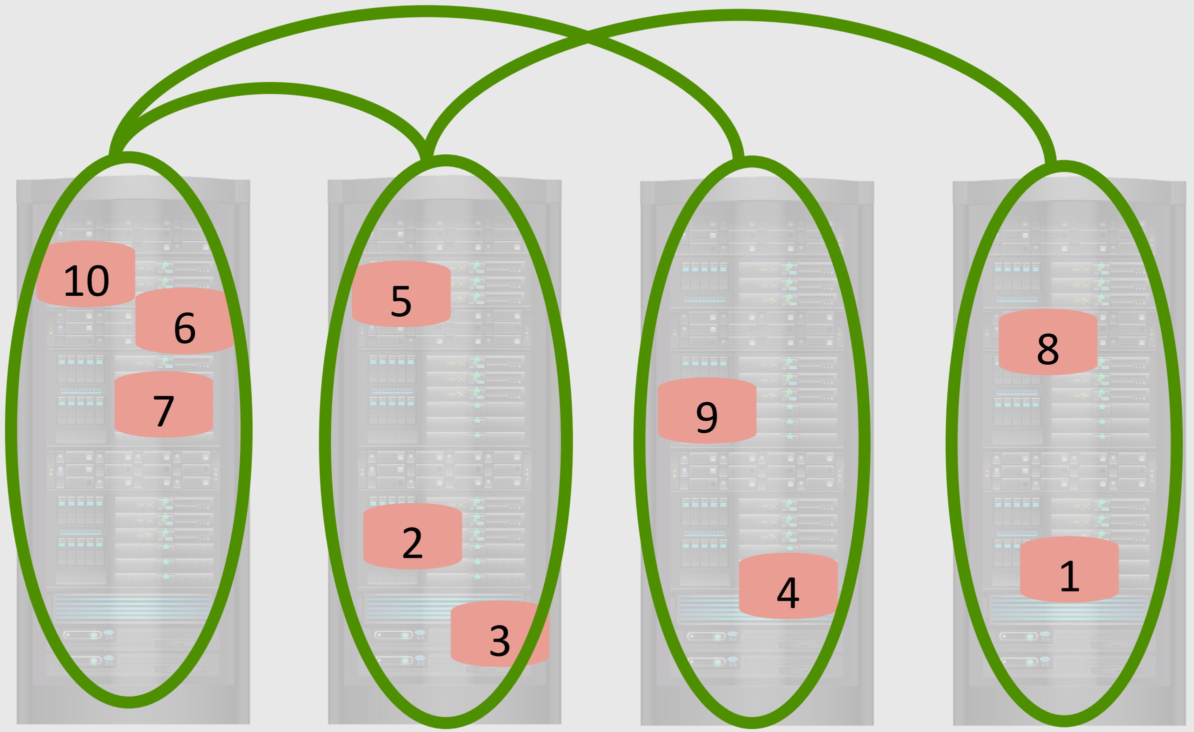
Allowing Item Movements on Edges

I do not have capacity!

Allowing Item Movements on Edges

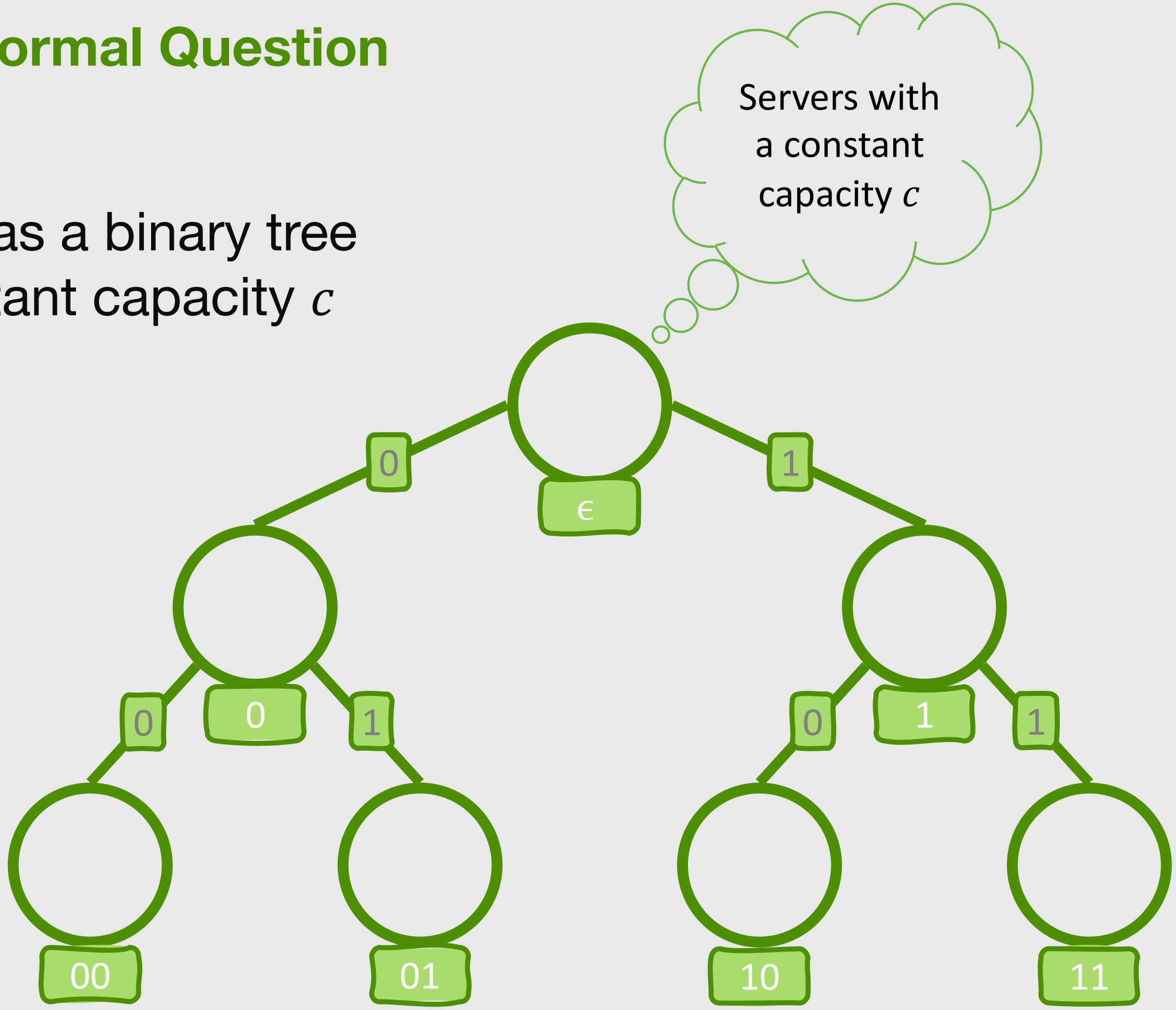
Allowing Item Movements on Edges

An Abstraction



Formal Question

- Given:
  - A set of servers connected as a binary tree
    - Each server with a constant capacity c

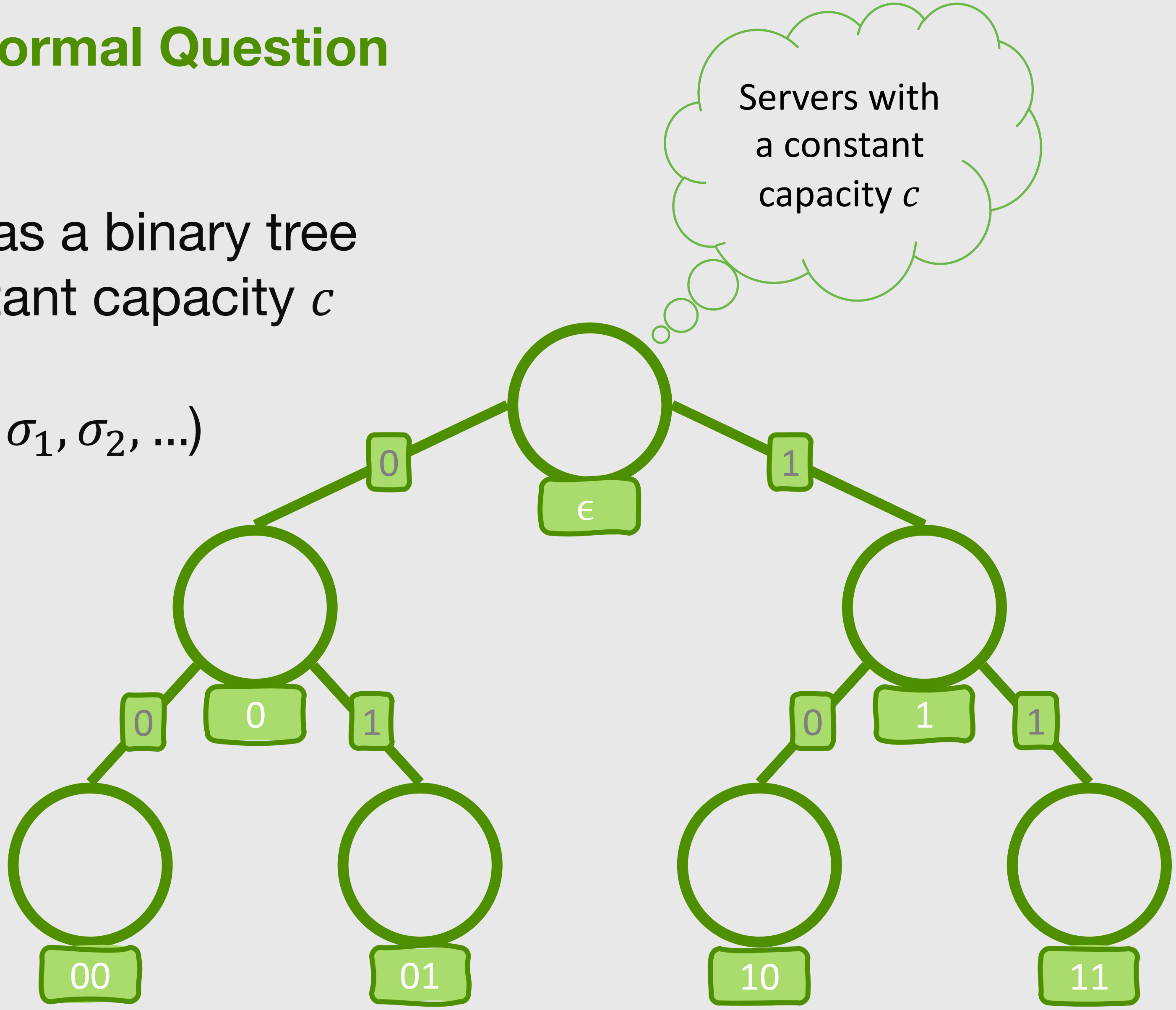


Tree

Formal Question

- Given:
  - A set of servers connected as a binary tree
    - Each server with a constant capacity c
  - A set of items
    - Revealed over time ($\sigma = \sigma_1, \sigma_2, \ldots$)

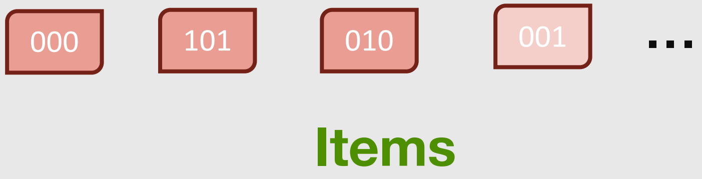
Items

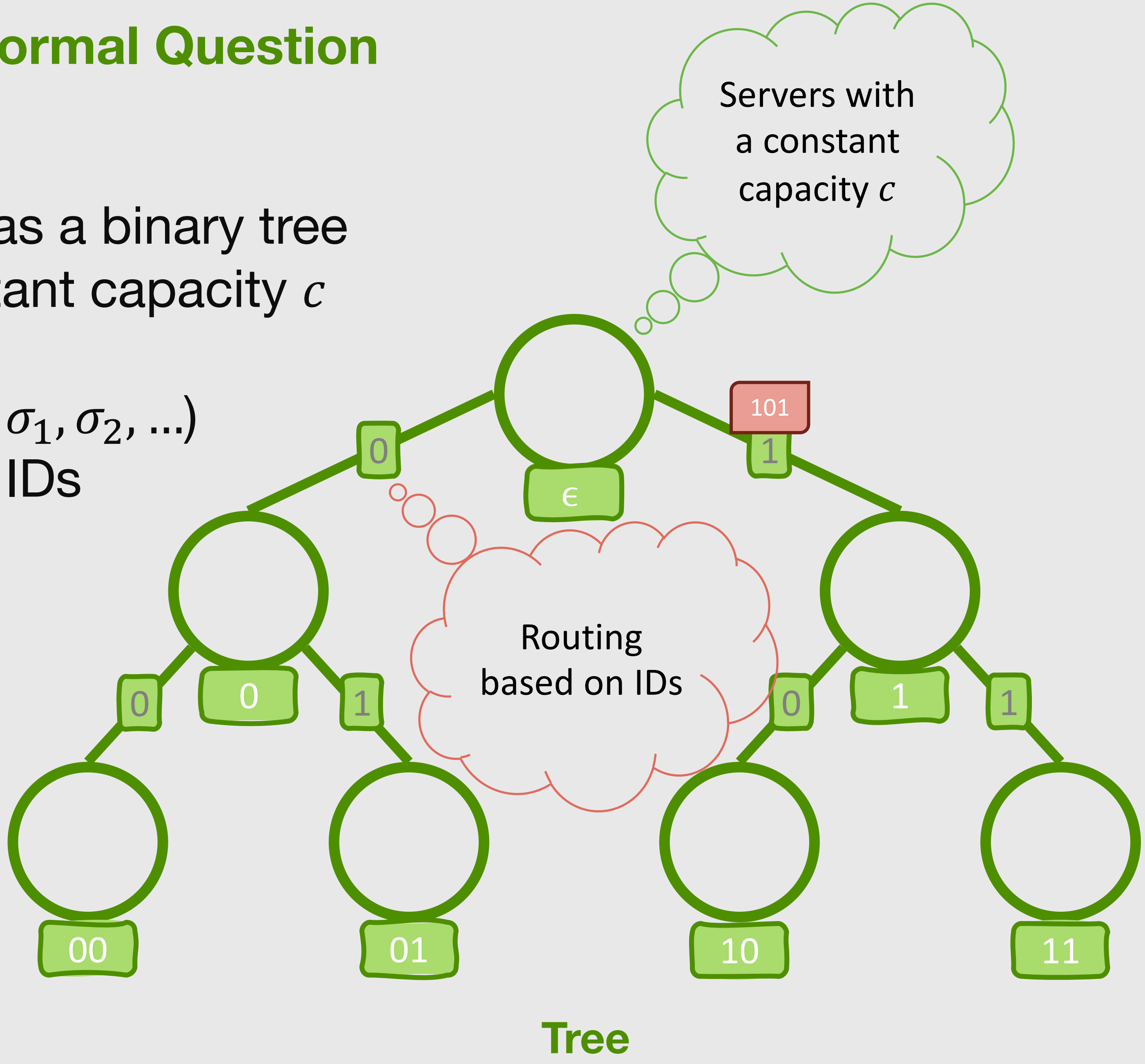


Tree

Formal Question

- Given:
  - A set of servers connected as a binary tree
    - Each server with a constant capacity c
  - A set of items
    - Revealed over time ($\sigma = \sigma_1, \sigma_2, \ldots$)
  - Only local routing based on IDs

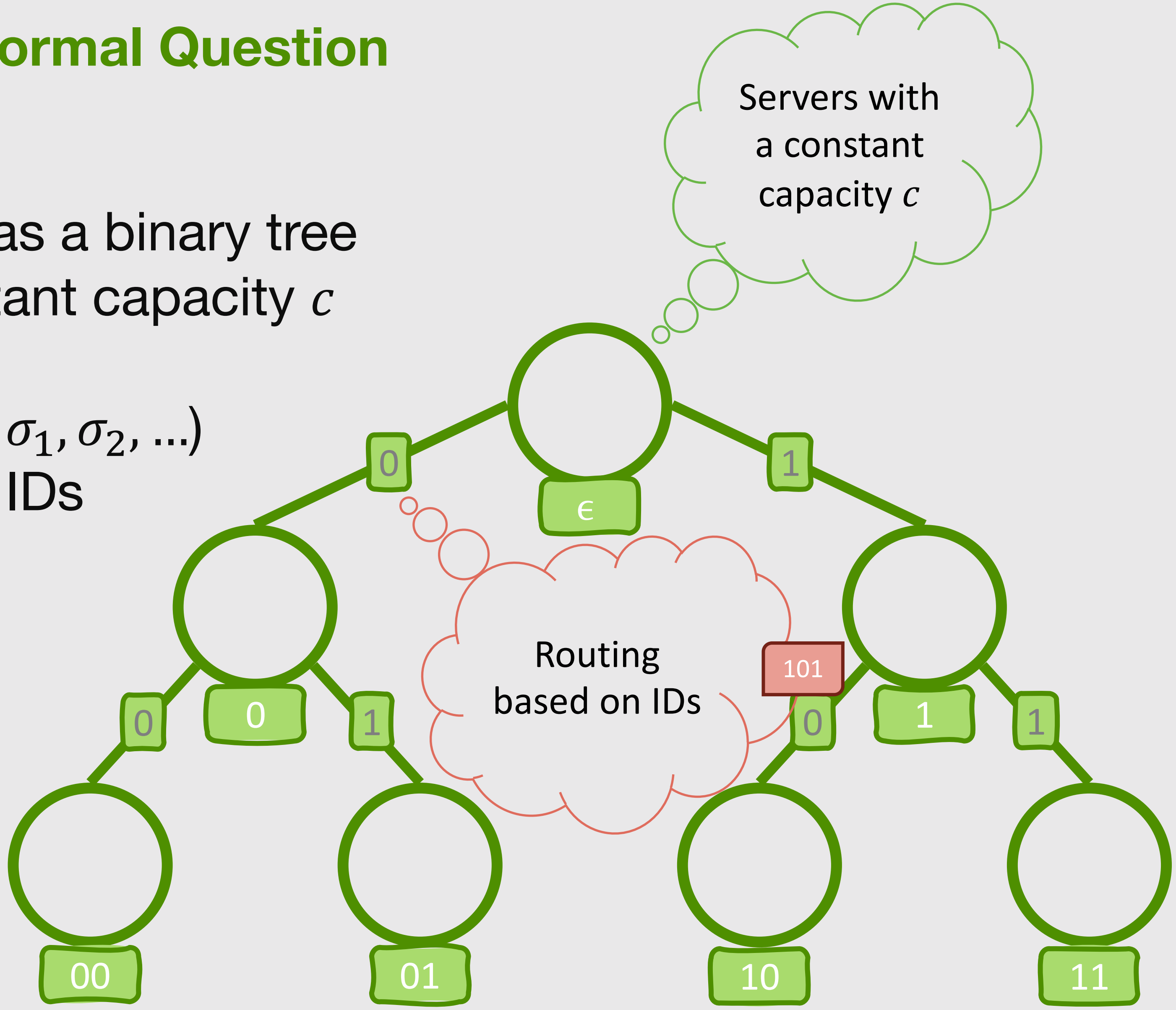




Formal Question

- Given:
  - A set of servers connected as a binary tree
    - Each server with a constant capacity c
  - A set of items
    - Revealed over time ($\sigma = \sigma_1, \sigma_2, \ldots$)
  - Only local routing based on IDs

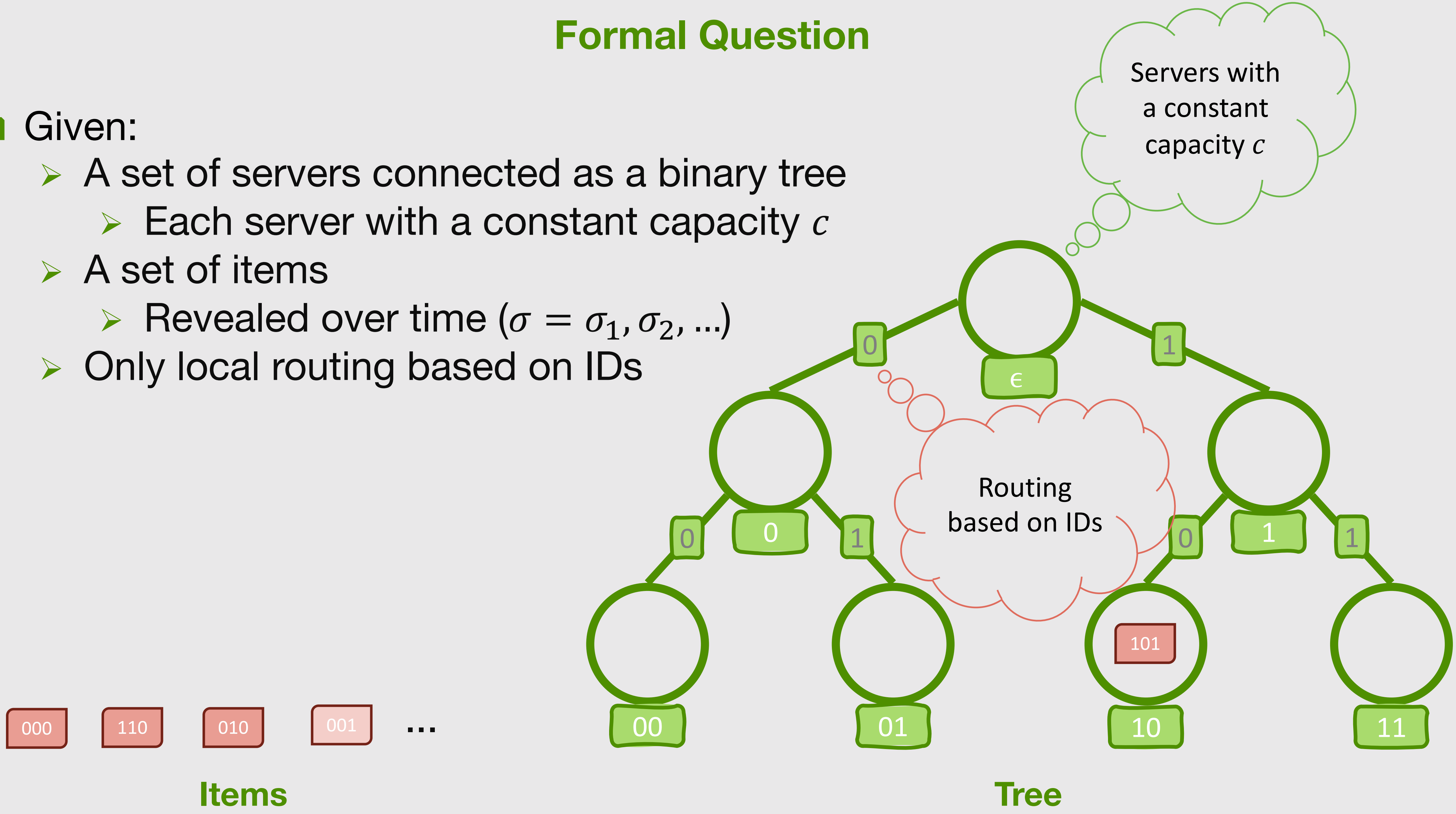
Items



Tree

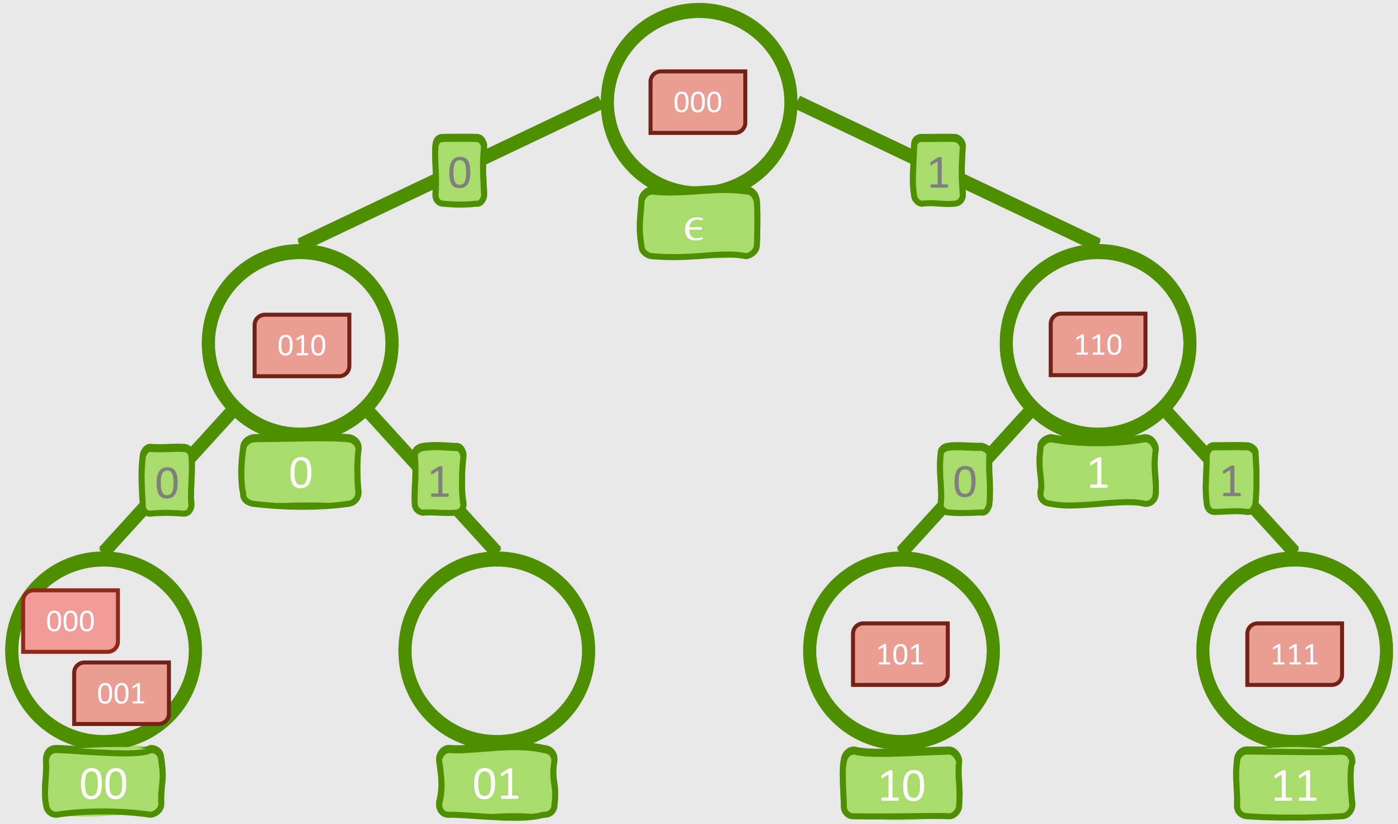
Formal Question

- Given:
  - A set of servers connected as a binary tree
    - Each server with a constant capacity c
  - A set of items
    - Revealed over time ($\sigma = \sigma_1, \sigma_2, \ldots$)
  - Only local routing based on IDs



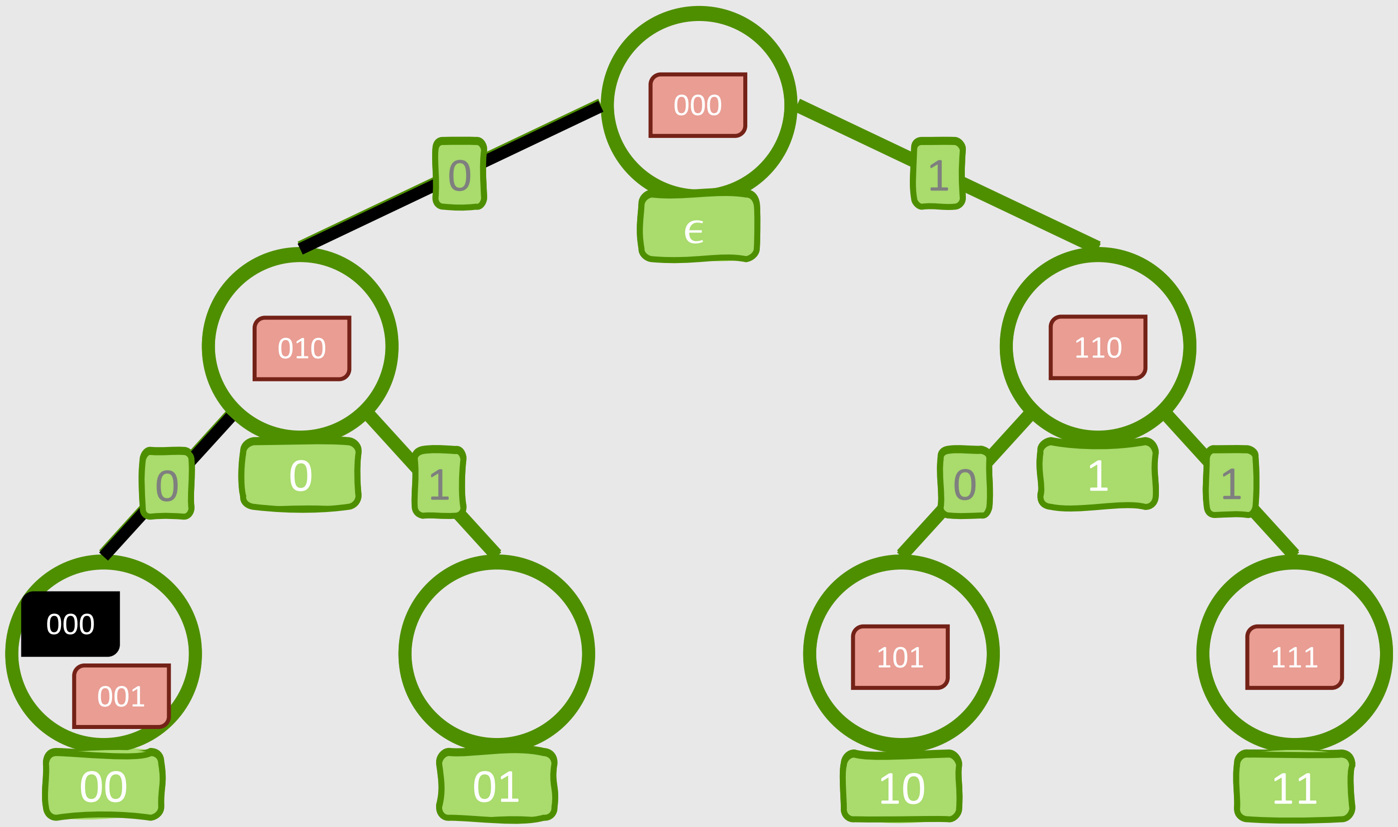
Formal Question

- Actions (for a prefilled tree*):



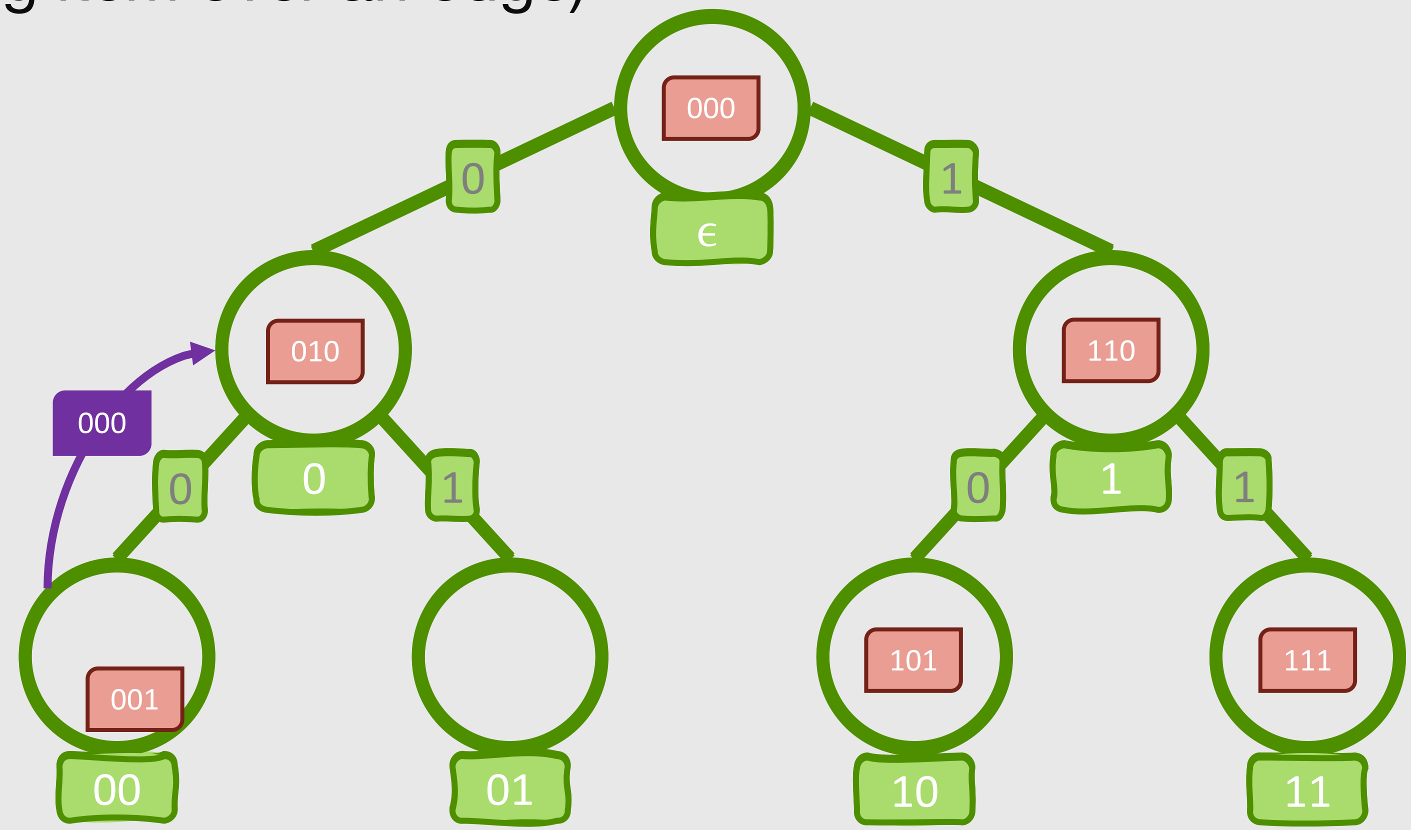
Formal Question

- Actions (for a prefilled tree*):
  - Access an element (depth)



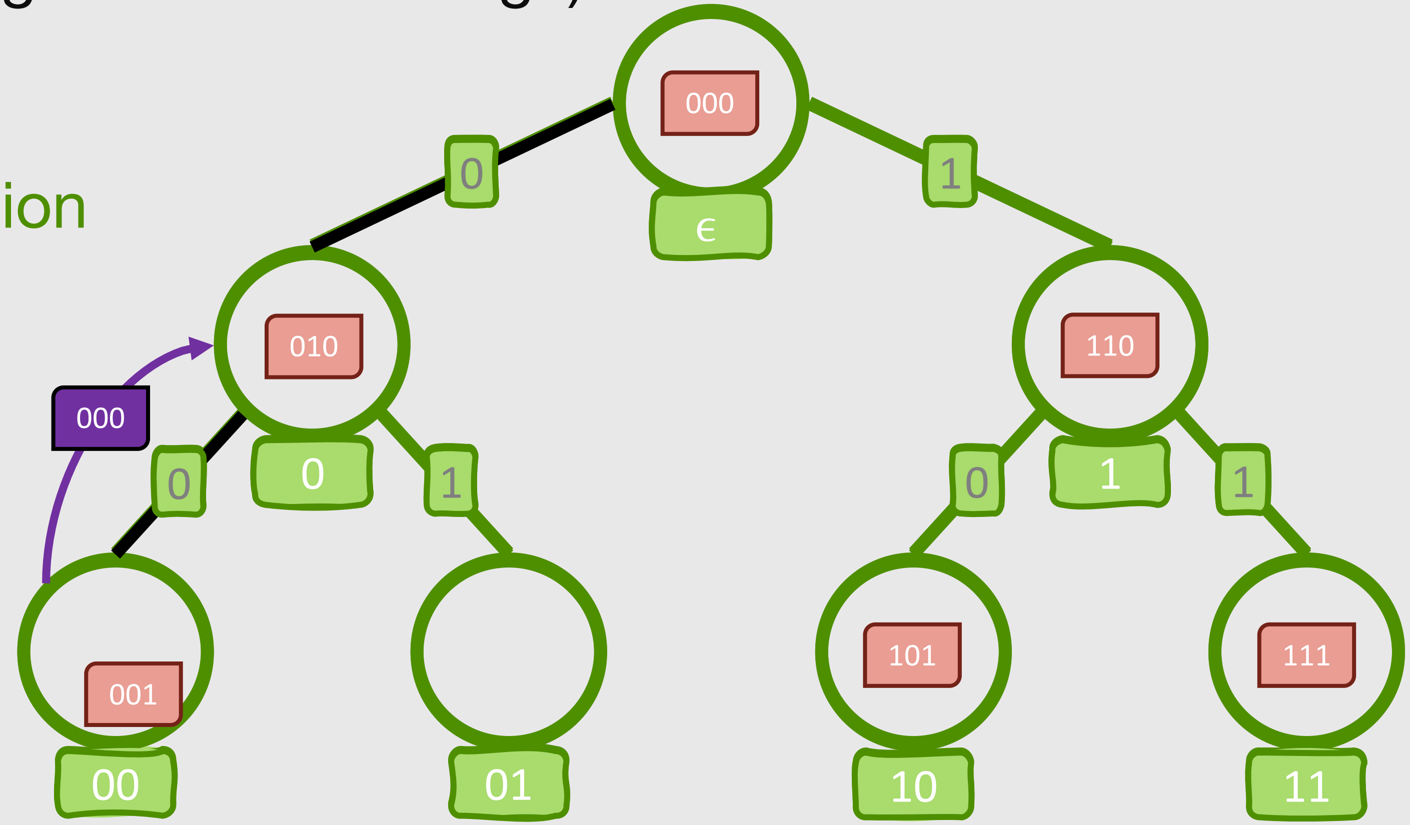
Formal Question

- Actions (for a prefilled tree*):
  - Access an element (depth)
  - Reconfigure the tree (moving item over an edge)



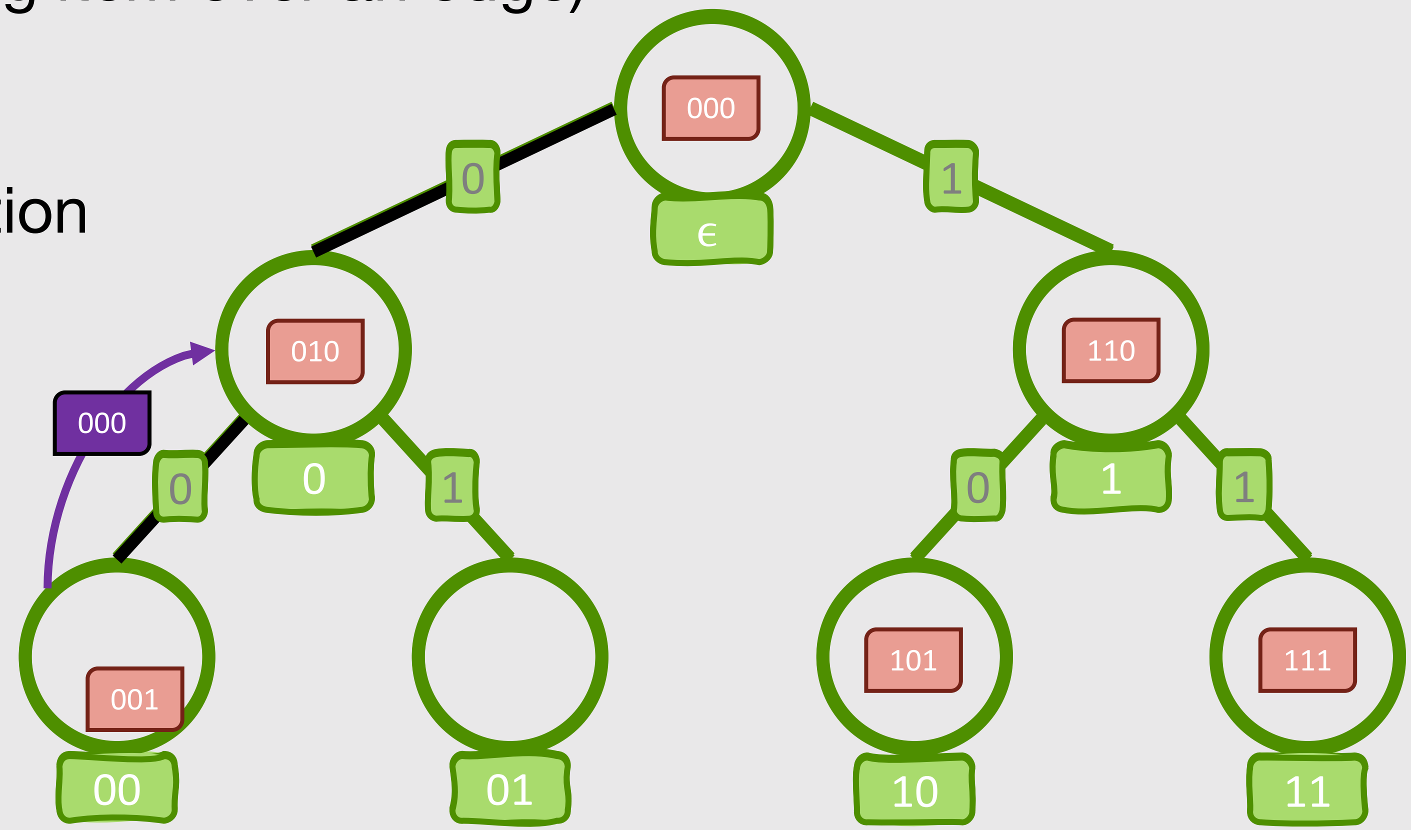
Formal Question

- Actions (for a prefilled tree*):
  - Access an element (depth)
  - Reconfigure the tree (moving item over an edge)
- Objective
  - Minimize the total cost
  - Total: access + reconfiguration



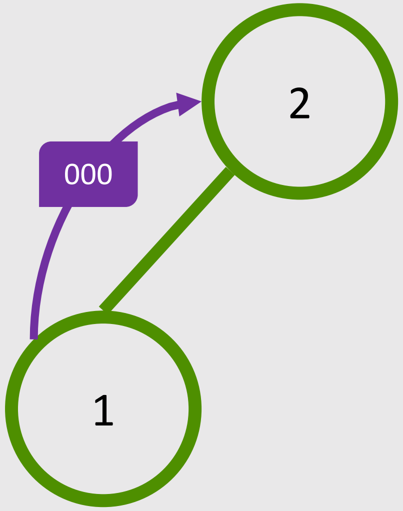
Formal Question

- Actions (for a prefilled tree*):
  - Access an element (depth)
  - Reconfigure the tree (moving item over an edge)
- Objective
  - Minimize the total cost
  - Total: access + reconfiguration
- Dynamically optimal
  - i.e., constant competitive
  - $Cost_{ALG} \leq \alpha \cdot Cost_{OPT}$



Previous work

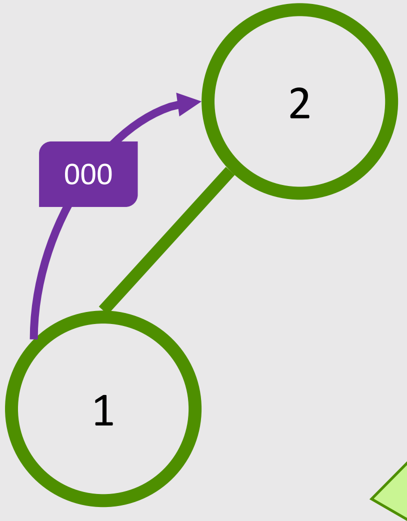
Data structure	Operation	Dynamically Optimal	Local Routing?
SeedTree [Pouradmghani et al., INFOCOM'23]	Item Movement	✅	✅



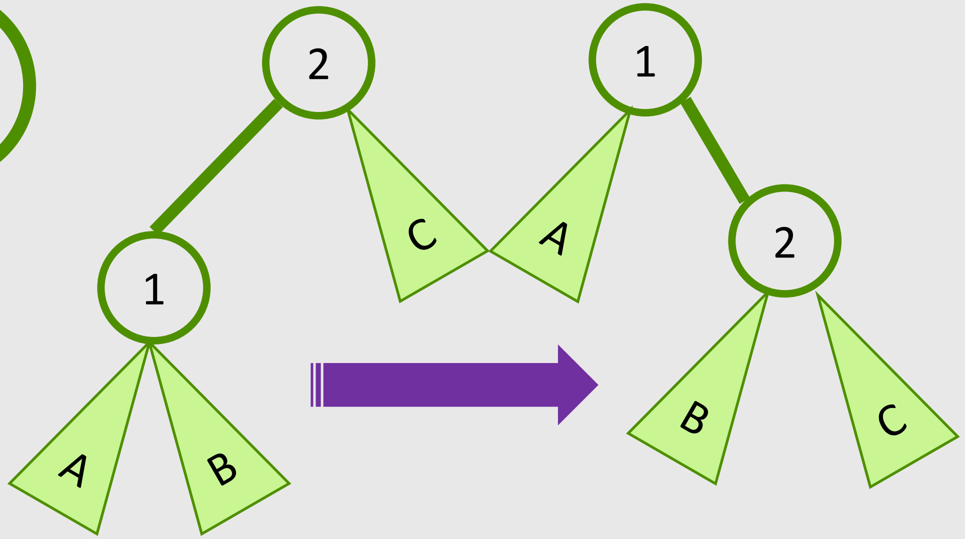
Moving
Items

Previous work

Data structure	Operation	Dynamically Optimal	Local Routing?
SeedTree [Pouradmghani et al., INFOCOM'23]	Item Movement	✅	✅
Splay Tree [Sleator & Tarjan J. ACM'85]	Rotation	?	✅
Tango Tree [Demaine et al. FOCS'04, J. Comput07]	Rotation	❌	✅
MultiSplay [Wamg et al. SODA'06]	Rotation	?	✅



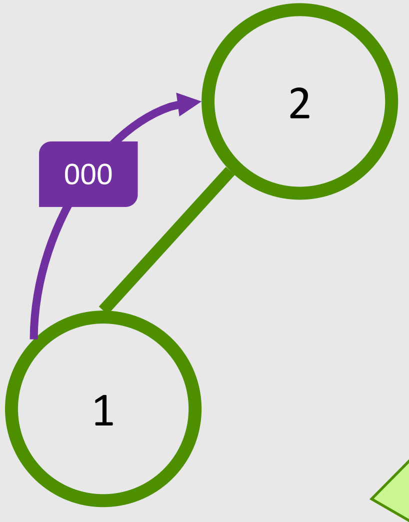
Moving Items



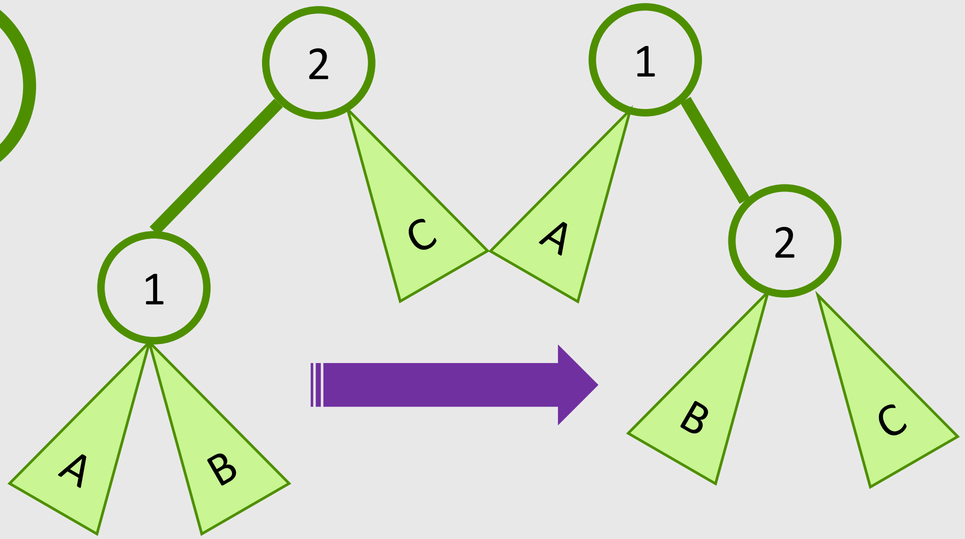
Rotation

Previous work

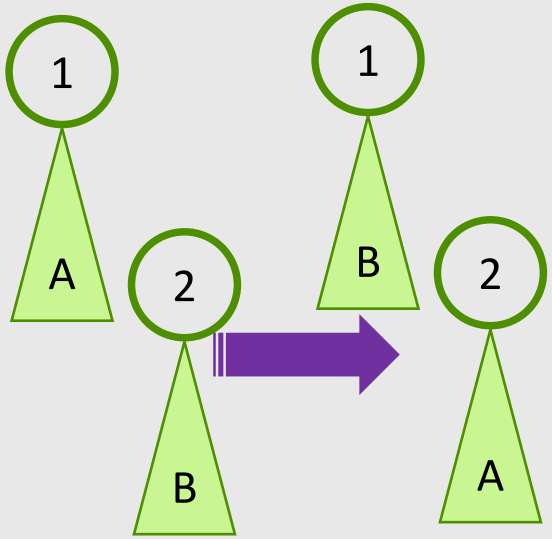
Data structure	Operation	Dynamically Optimal	Local Routing?
SeedTree [Pouradmghani et al., INFOCOM’23]	Item Movement	✅	✅
Splay Tree [Sleator & Tarjan J. ACM’85]	Rotation	?	✅
Tango Tree [Demaine et al. FOCS’04, J. Comput07]	Rotation	❌	✅
MultiSplay [Wamg et al. SODA’06]	Rotation	?	✅
Adaptive Huffman [Vitter J.ACM’87, FGK J.ACM’85]	Subtree Swap	✅	❌



Moving Items



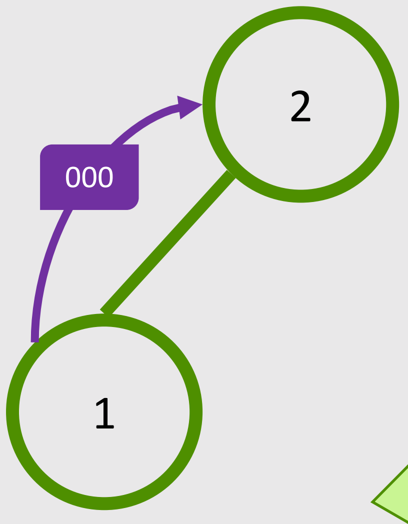
Rotation



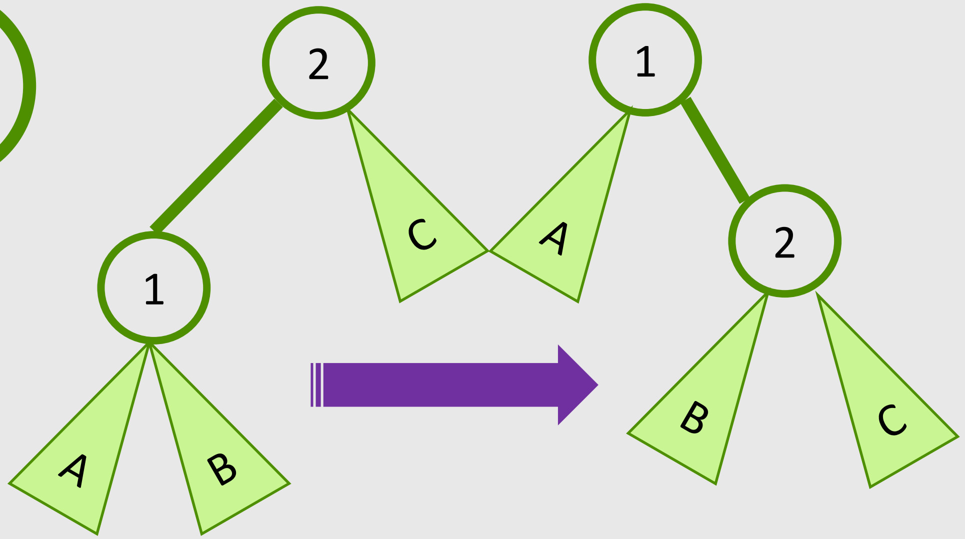
Subtree Swap

Previous work

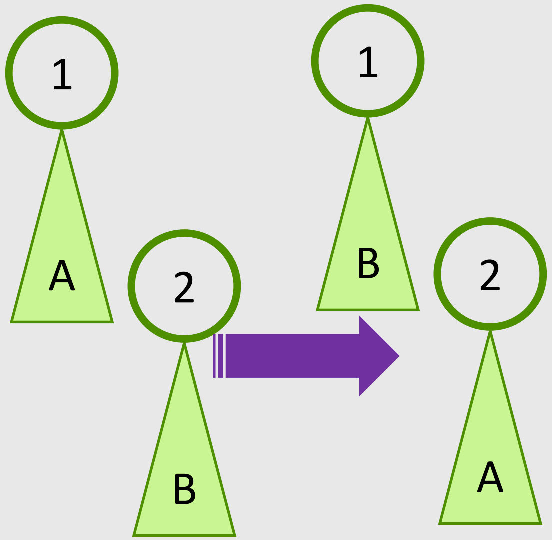
Data structure	Operation	Dynamically Optimal	Local Routing?
SeedTree [Pouradmghani et al., INFOCOM'23]	Item Movement	✅	✅
Splay Tree [Sleator & Tarjan J. ACM'85]	Rotation	?	✅
Tango Tree [Demaine et al. FOCS'04, J. Comput07]	Rotation	❌	✅
MultiSplay [Wamg et al. SODA'06]	Rotation	?	✅
Adaptive Huffman [Vitter J.ACM'87, FGK J.ACM'85]	Subtree Swap	✅	❌
Push-down-Tree [Avin et al. LATIN'20, TON'22]	Item Swap	✅	❌



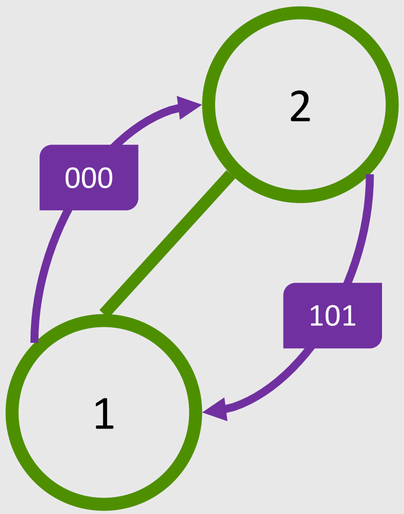
Moving Items



Rotation



Subtree Swap

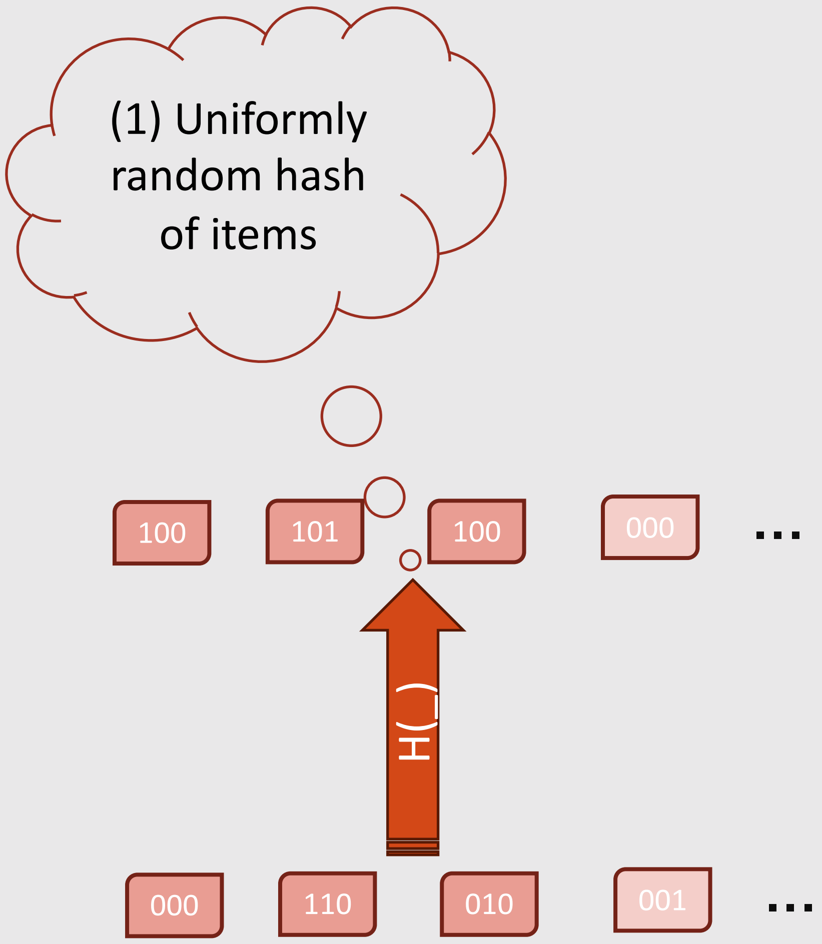


Item Swap

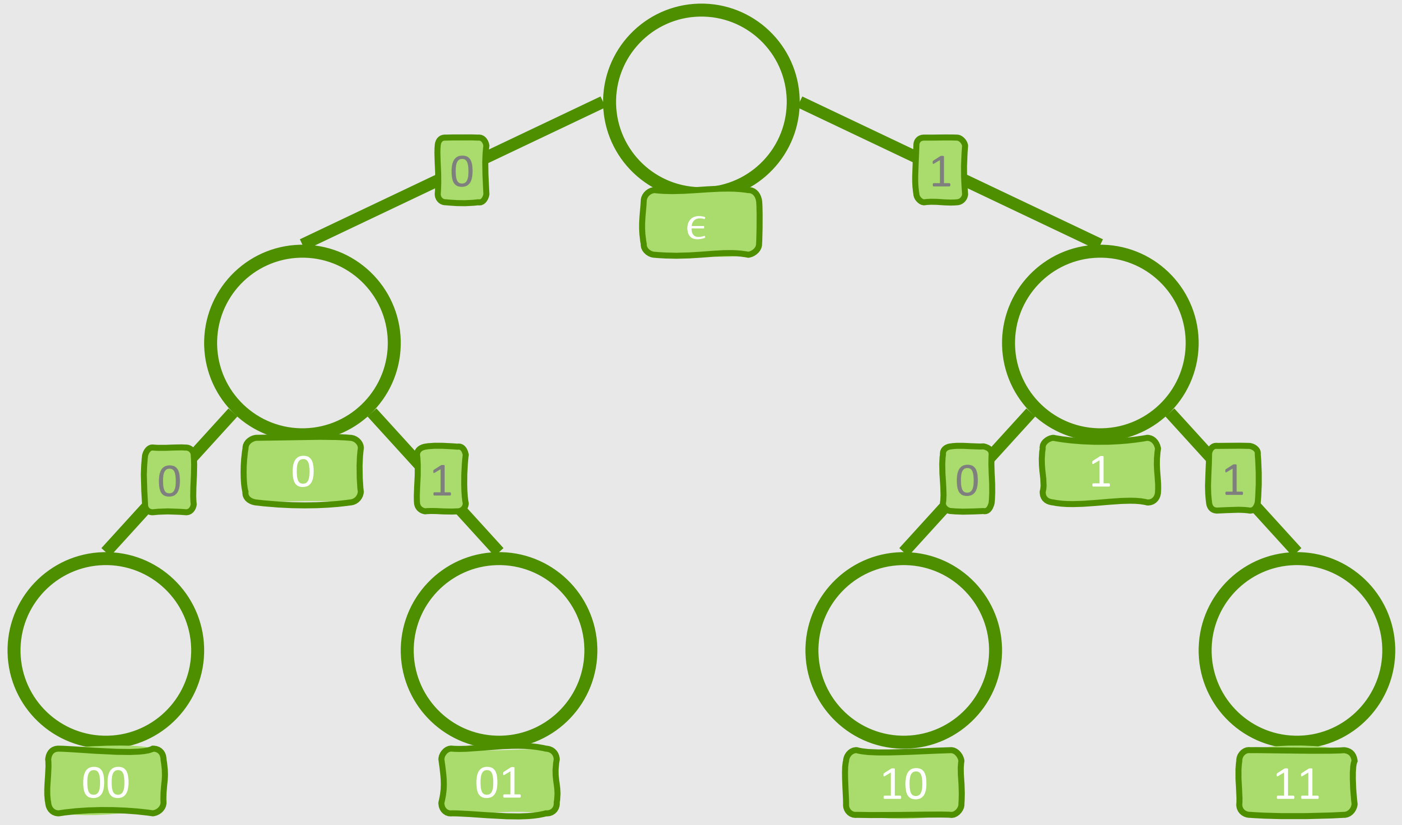
SeedTree

1) Why? 2) What? 3) How?

Tools & Techniques

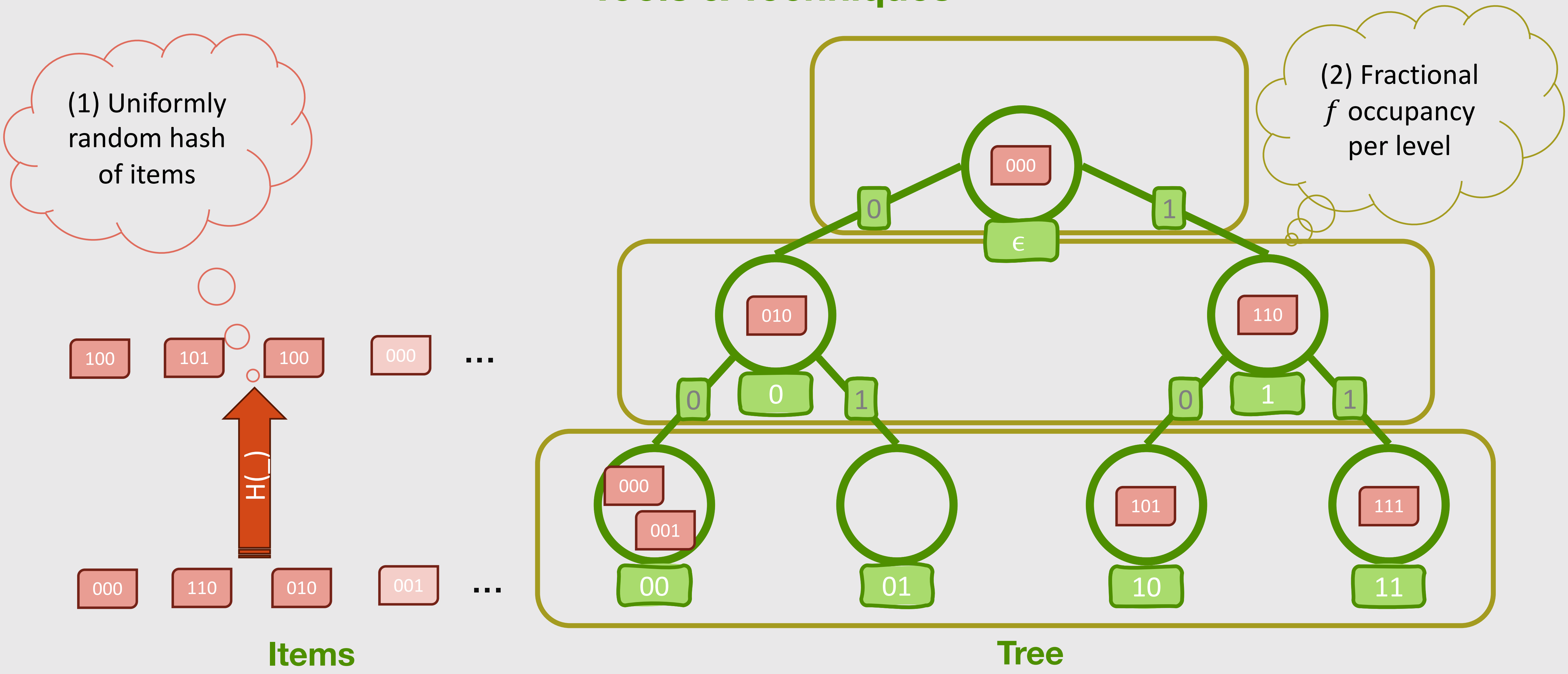


Items

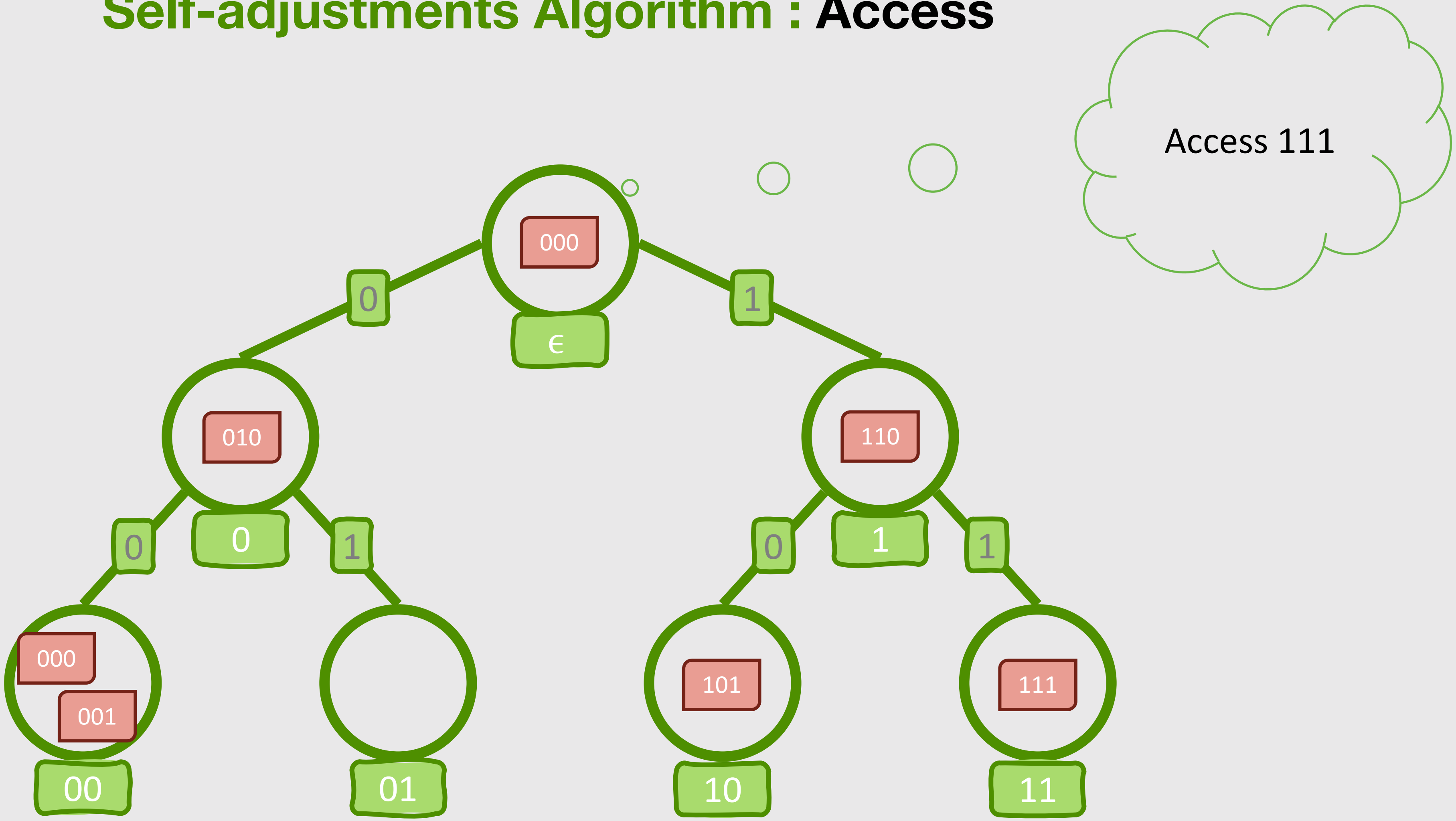


Tree

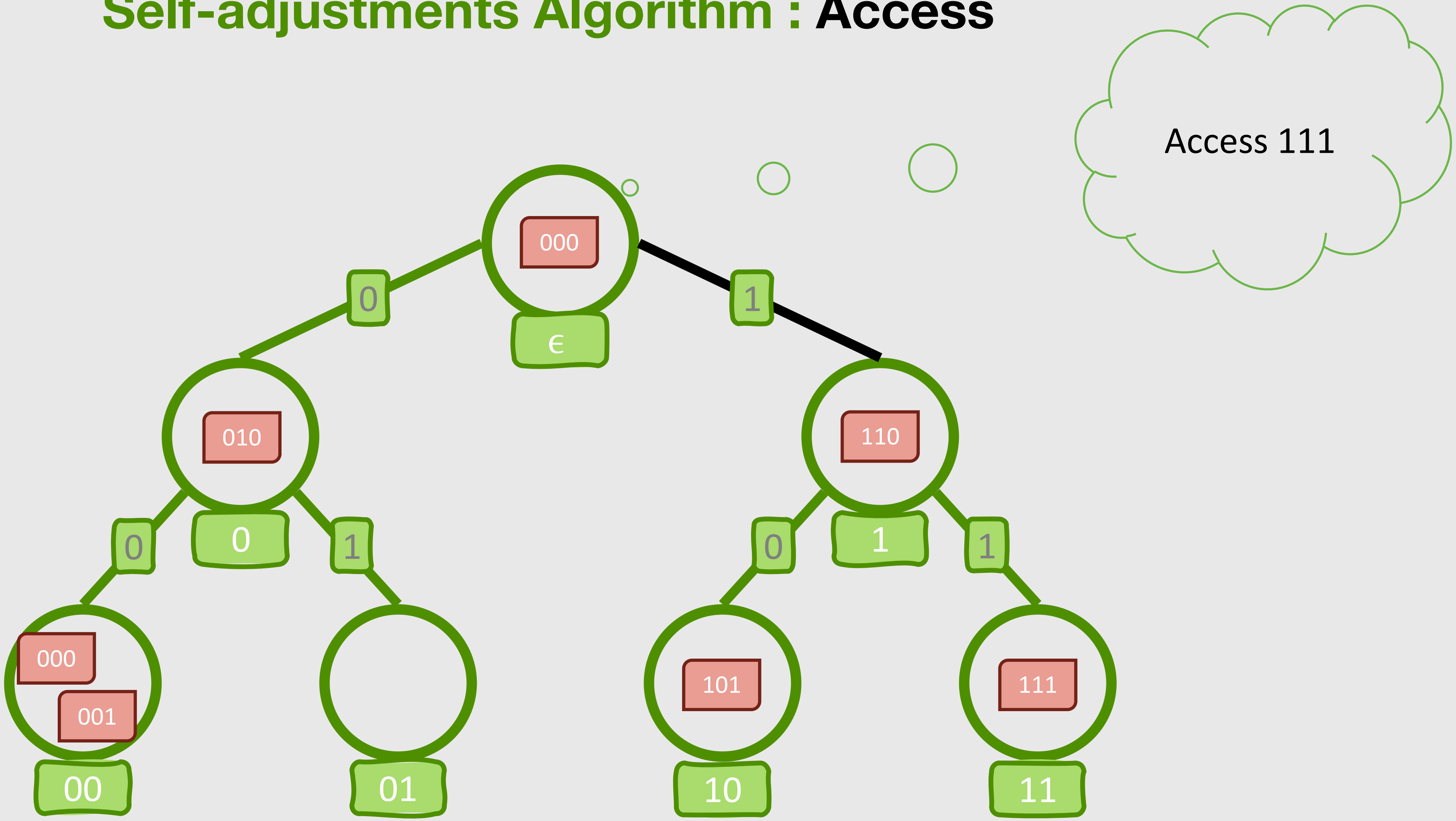
Tools & Techniques



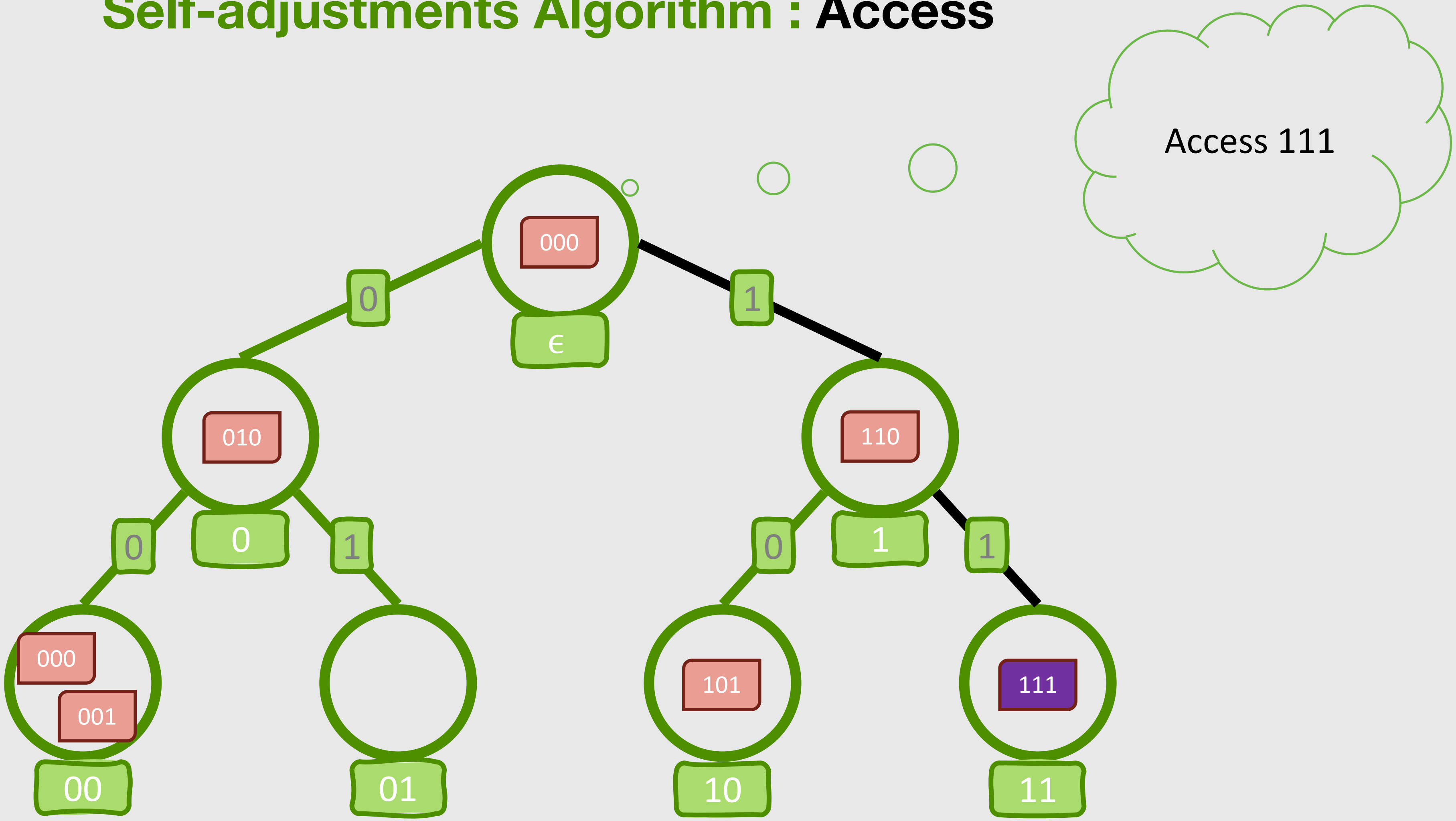
Self-adjustments Algorithm : Access



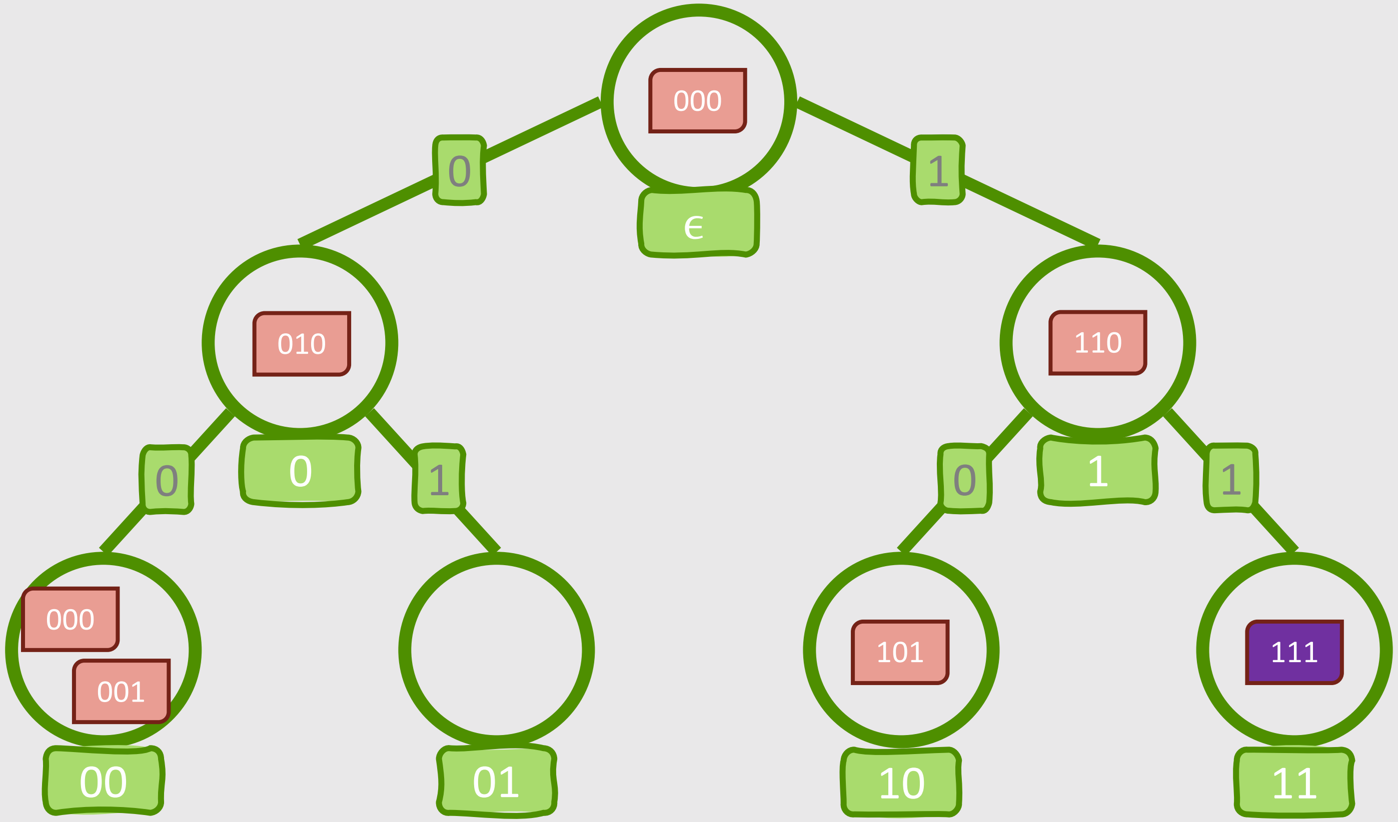
Self-adjustments Algorithm : Access



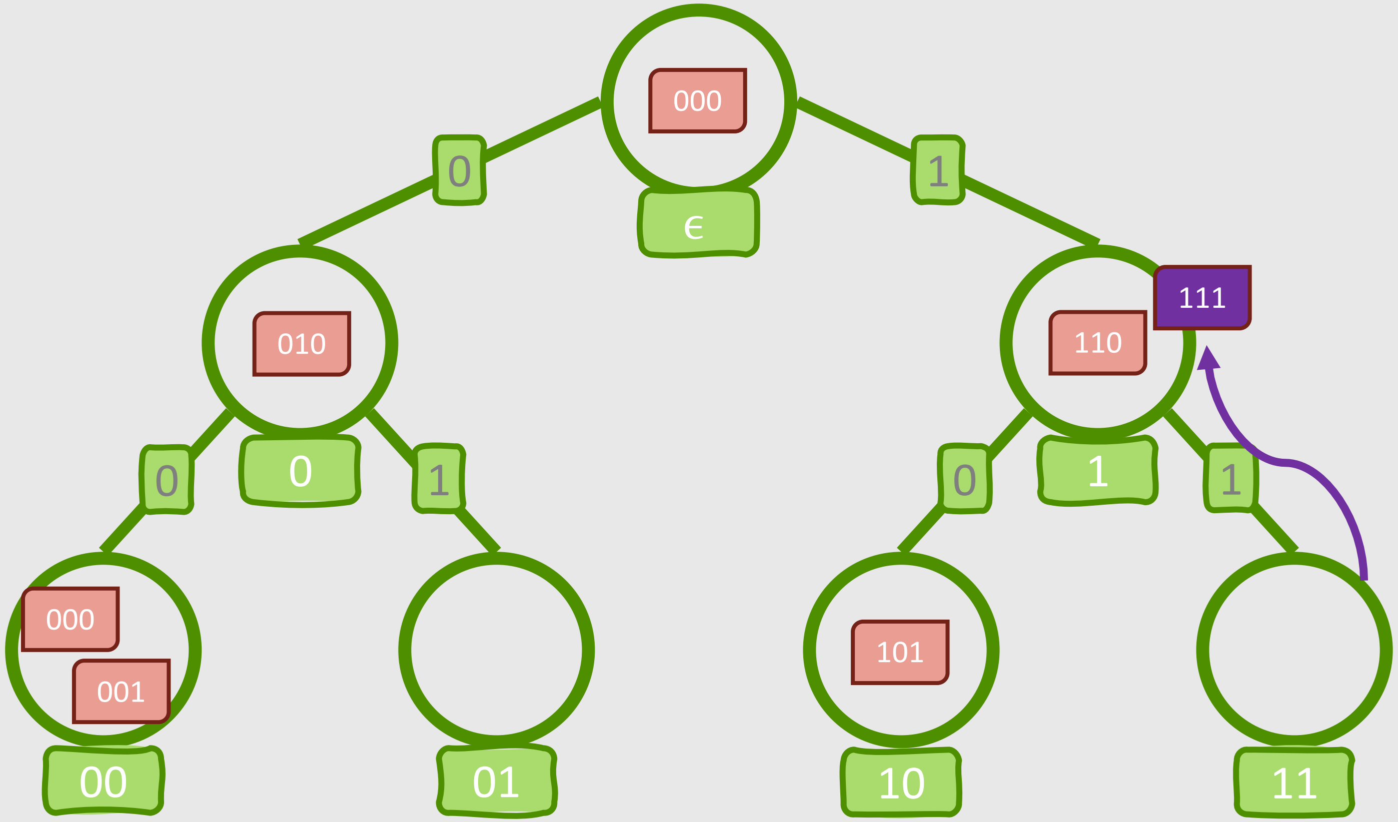
Self-adjustments Algorithm : Access



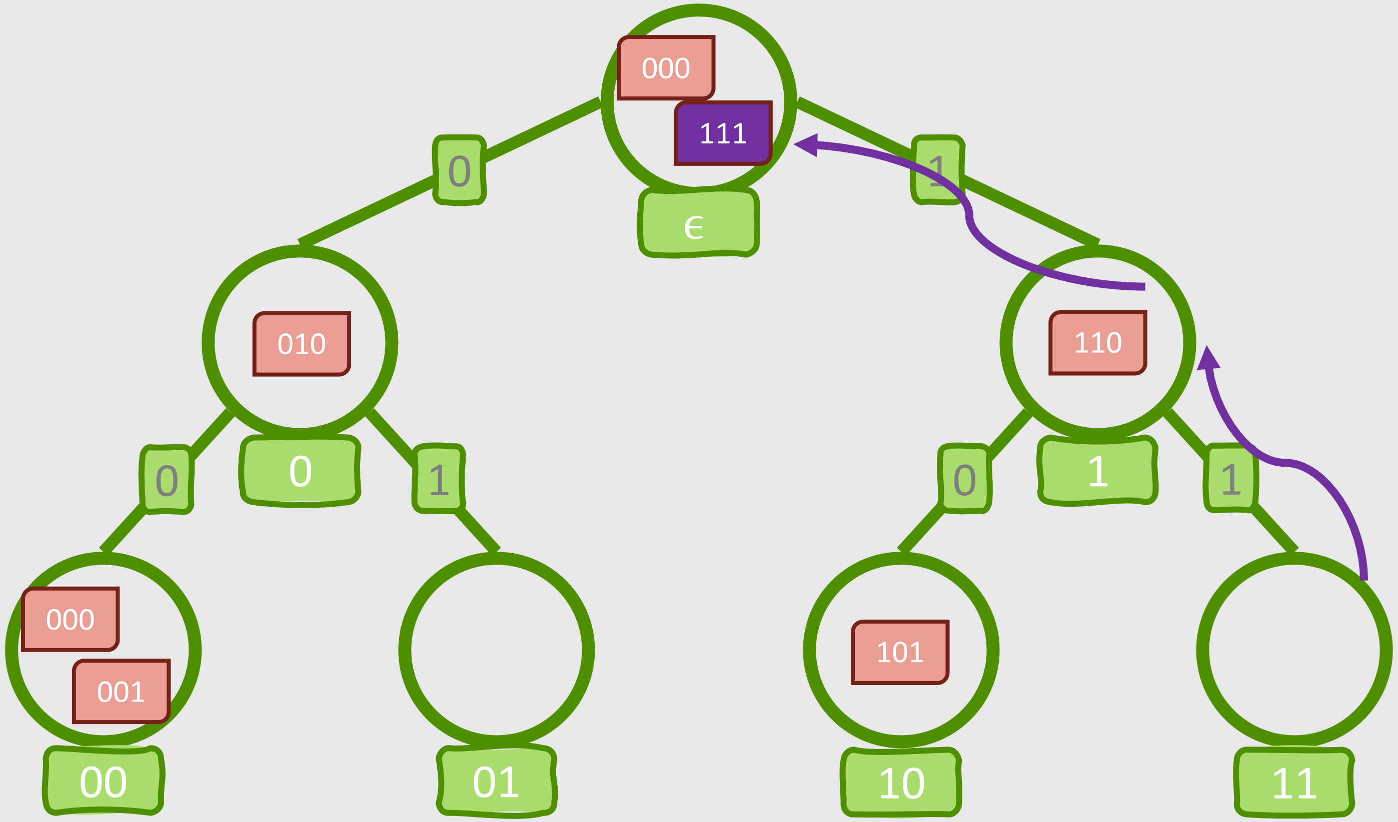
Self-adjustments Algorithm : **Pull-up**



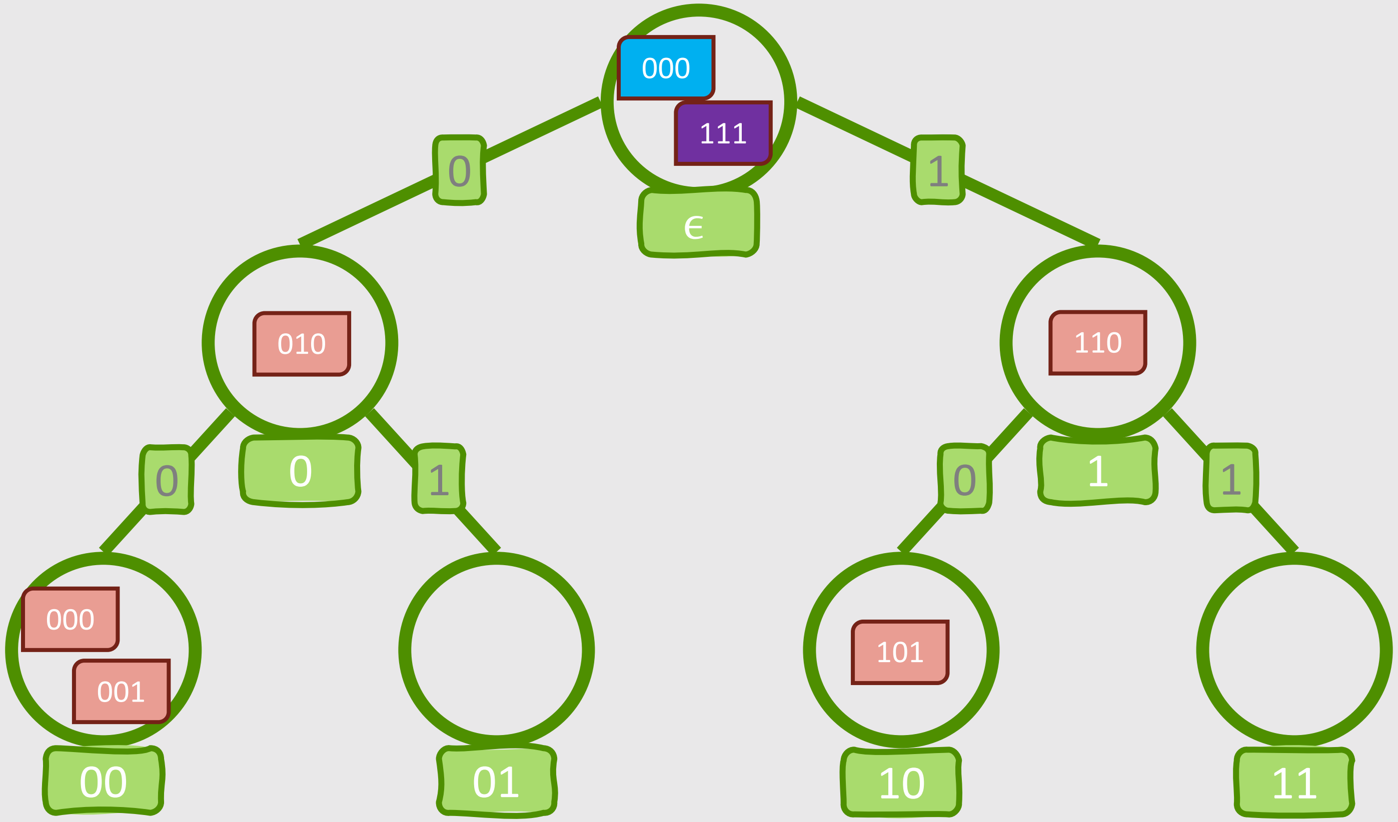
Self-adjustments Algorithm: **Pull-up**



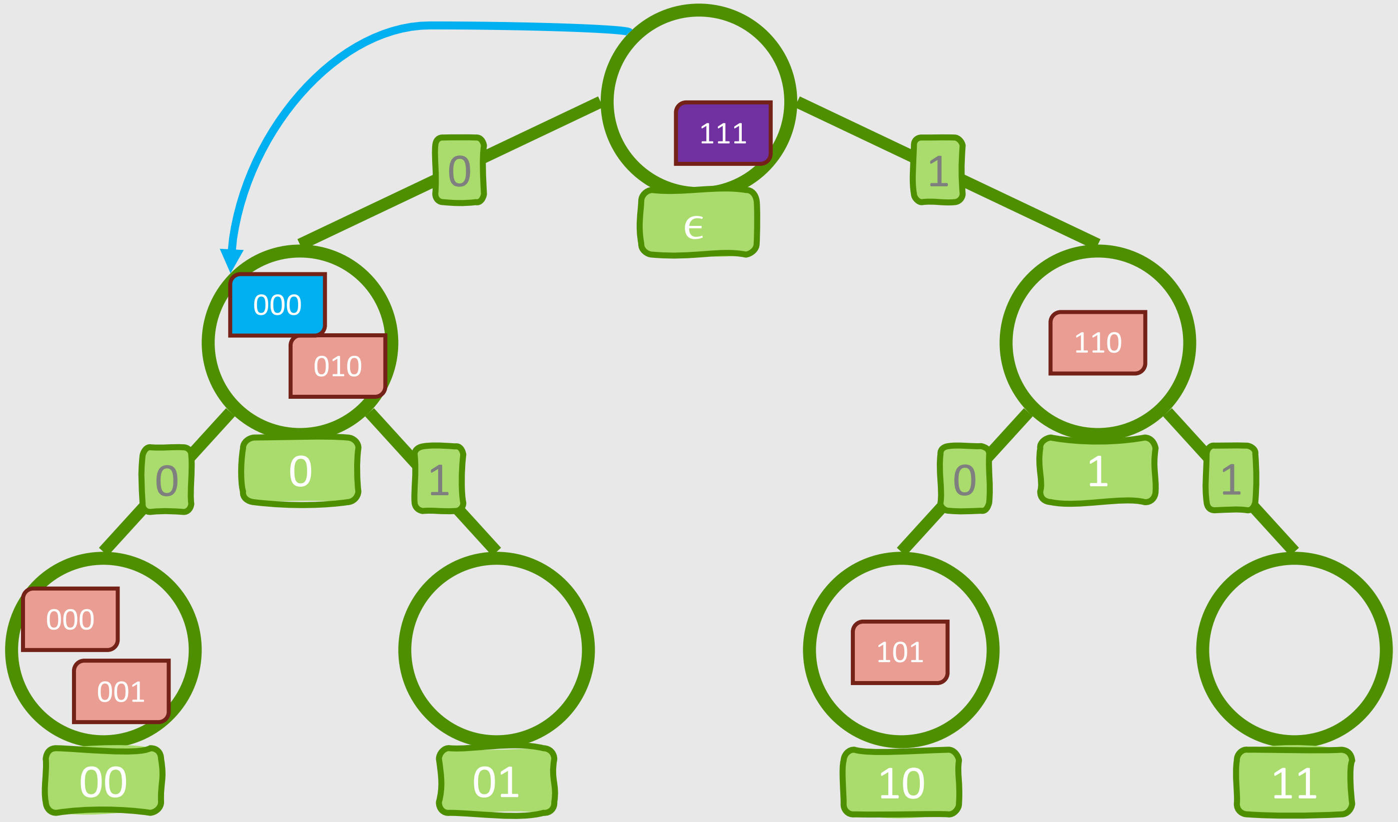
Self-adjustments Algorithm: **Pull-up**



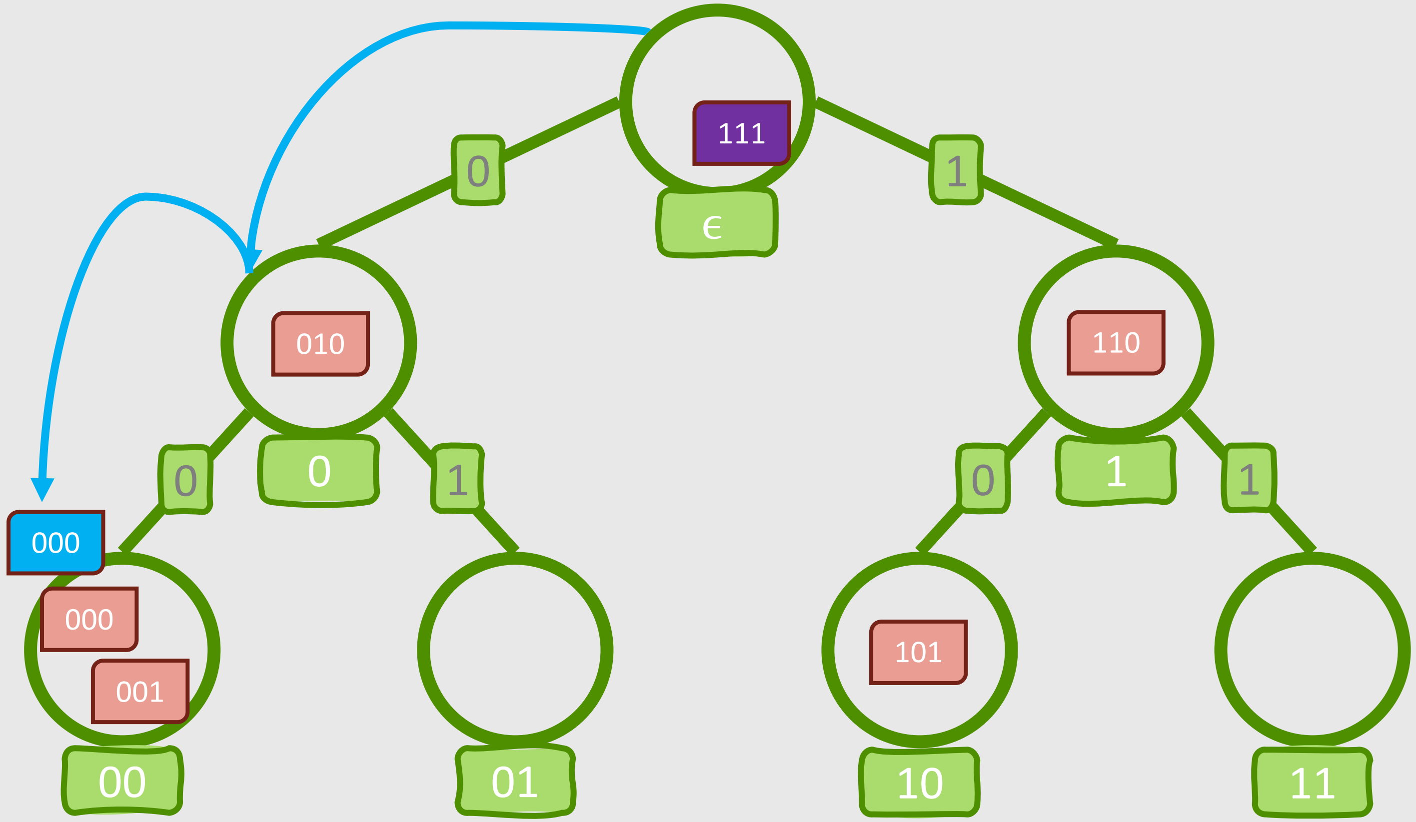
Self-adjustments Algorithm: Push-down



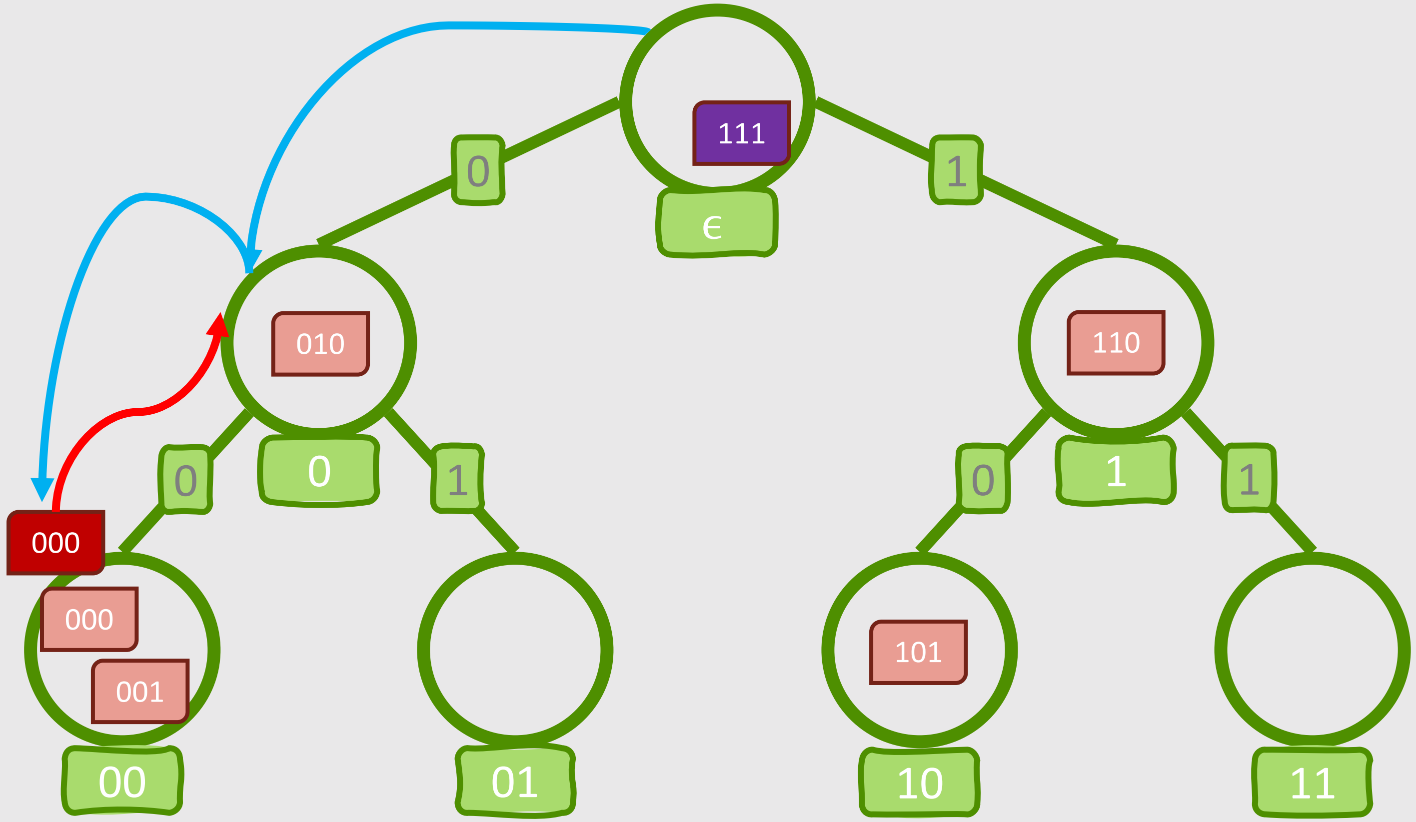
Self-adjustments Algorithm: Push-down



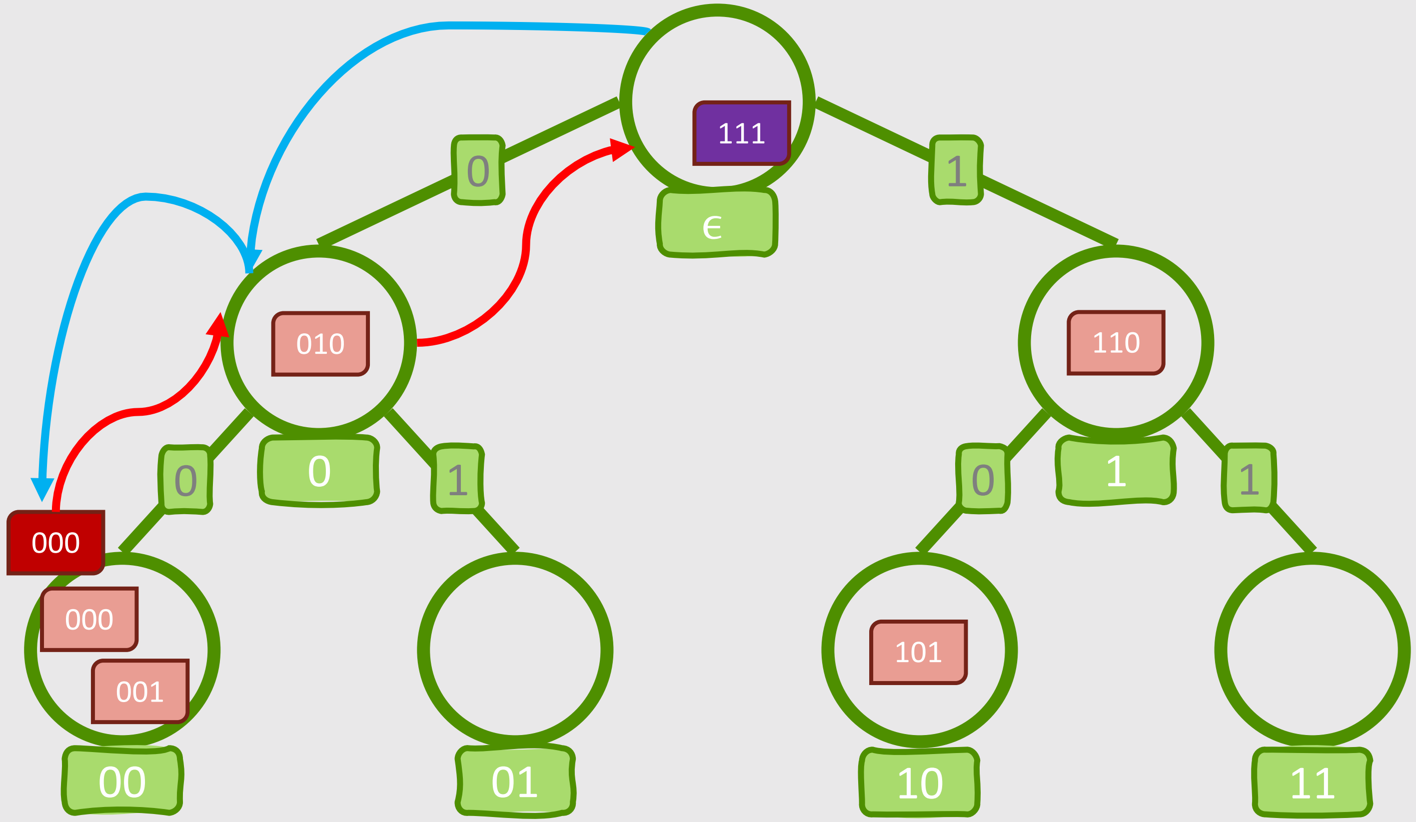
Self-adjustments Algorithm: Push-down



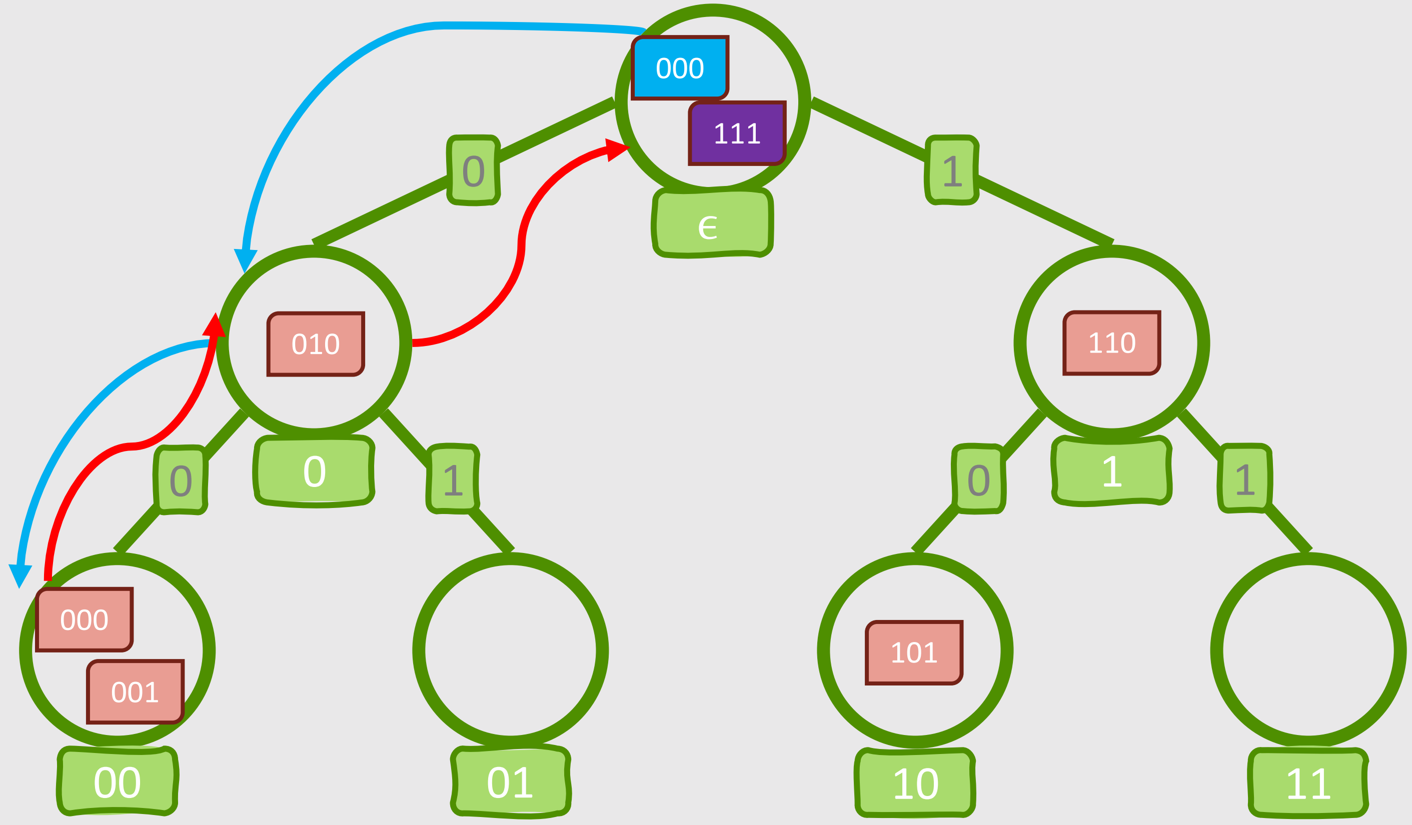
Self-adjustments Algorithm: Push-down



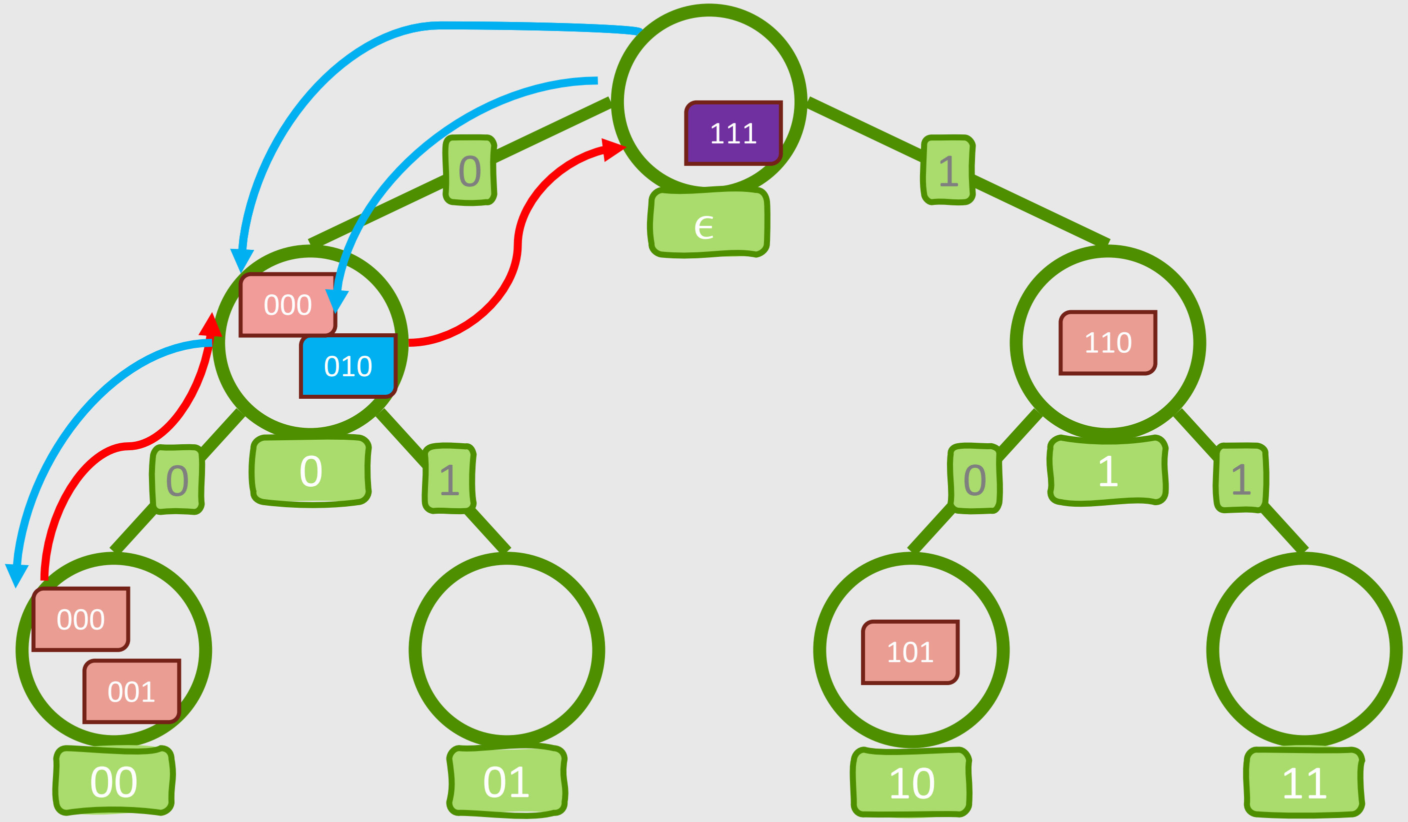
Self-adjustments Algorithm: **Push-down**



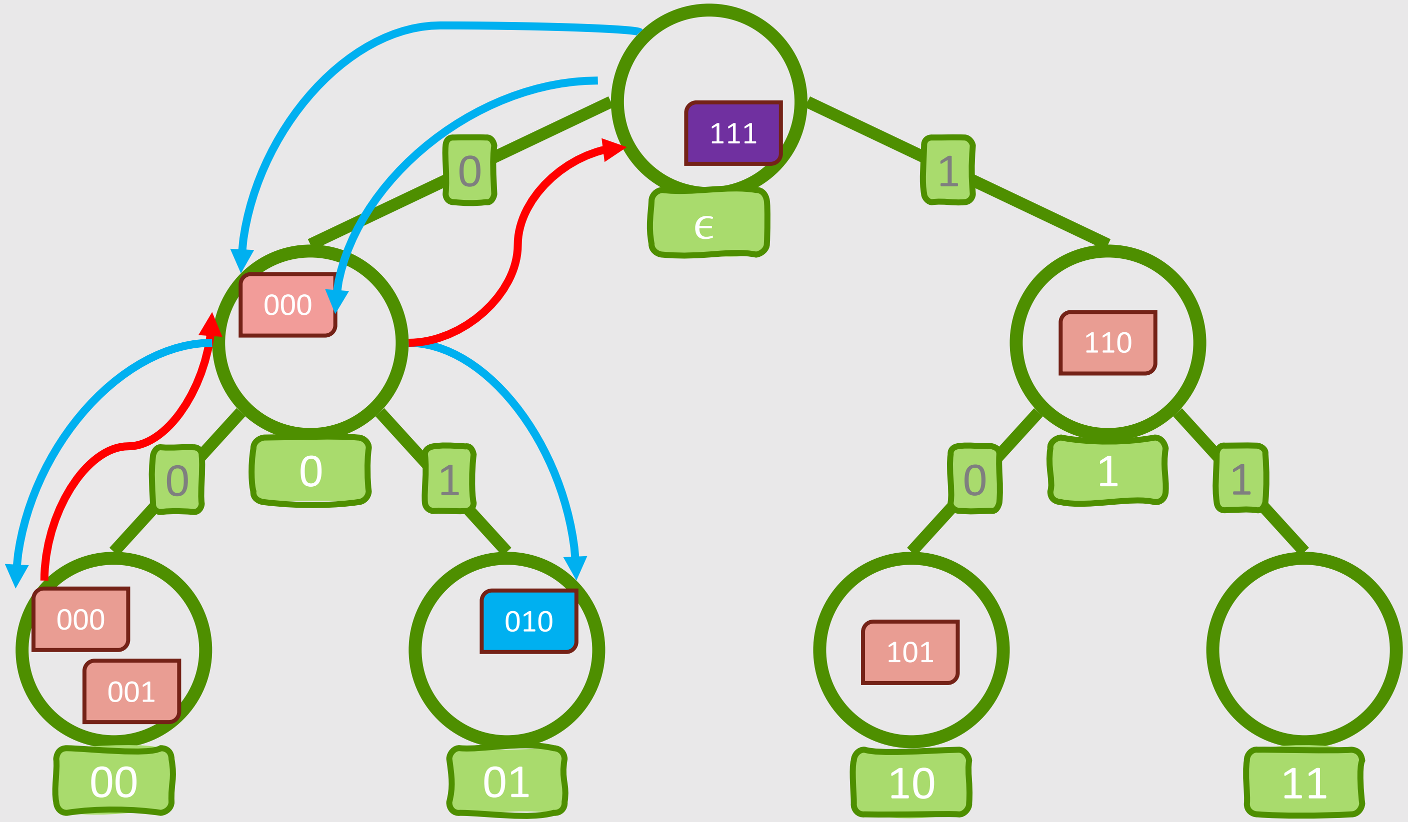
Self-adjustments Algorithm: Push-down



Self-adjustments Algorithm: Push-down



Self-adjustments Algorithm: Push-down



SeedTree Analysis

SeedTree Analysis

- Objective (over time and in expectation):

  SeedTree is dynamically optimal.

SeedTree Analysis

- Objective (over time and in expectation):

SeedTree is dynamically optimal.

- Property 1:

Reconfiguration cost of SeedTree $\leq 2 \cdot \left(\left\lceil \frac{1}{1-f} \right\rceil + 1\right) \cdot$Access cost of SeedTree.

SeedTree Analysis

- **Objective** (over time and in expectation):

  SeedTree is dynamically optimal.

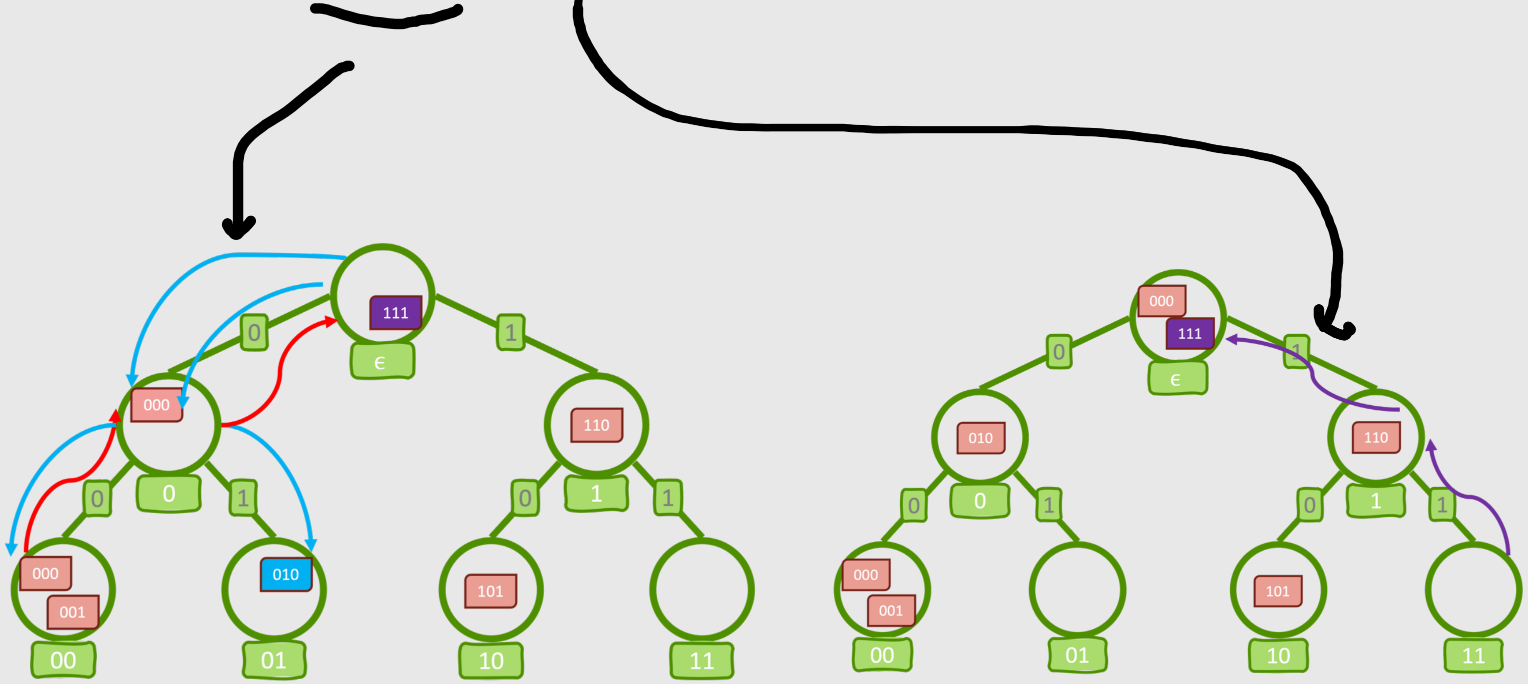
- **Property 1:**

  Reconfiguration cost of SeedTree $\leq 2 \cdot \left(\left\lceil \frac{1}{1-f} \right\rceil + 1 \right) \cdot$ Access cost of SeedTree.

Proof sketch:

- 1 for pull-up
- Fractional occupancy ensures success

  after $\left\lceil \frac{1}{1-f} \right\rceil$ tries, in expectation



SeedTree Analysis

- **Objective** (over time and in expectation):

  SeedTree is dynamically optimal.

- **Property 1:**

  Reconfiguration cost of SeedTree $\leq 2 \cdot \left(\left\lceil \frac{1}{1-f} \right\rceil + 1\right) \cdot$Access cost of SeedTree.

- **Property 2:**

  Access cost of SeedTree $\leq 2 - \log(f) \cdot$Access cost in MRU Tree.

- **Most Recently Used (MRU) Tree:**

  More recently accessed items ⇒ Lower level in the tree

SeedTree Analysis

- Objective (over time and in expectation):

SeedTree is dynamically optimal.

- Property 1:

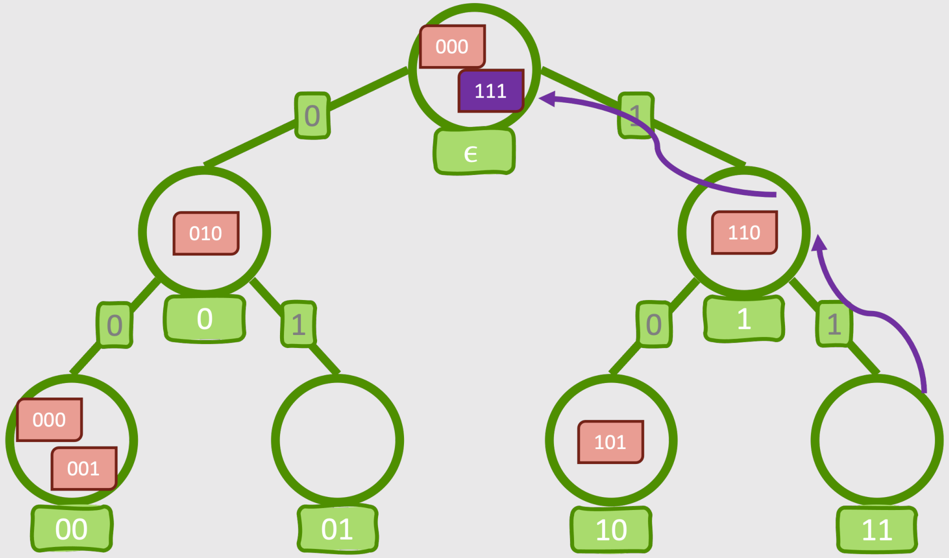
Reconfiguration cost of SeedTree $\leq 2 \cdot \left(\left\lceil \frac{1}{1-f} \right\rceil + 1\right) \cdot$Access cost of SeedTree.

- Property 2:

Access cost of SeedTree $\leq 2 - \log(f) \cdot$Access cost in MRU Tree.

Proof sketch:

- Recent ones go to the root
- Probability of going a level down decreases exponentially per level



SeedTree Analysis

- Objective (over time and in expectation):

SeedTree is dynamically optimal.

- Property 1:

Reconfiguration cost of SeedTree $\leq 2 \cdot \left(\left\lceil \frac{1}{1-f} \right\rceil + 1\right)$ ·Access cost of SeedTree.

- Property 2:

Access cost of SeedTree $\leq 2 - \log(f)$ ·Access cost in MRU Tree.

- Property 3:

Access cost in MRU Tree $\leq (1 + e)$ ·Access cost of OPT.

SeedTree Analysis

- Objective (over time and in expectation):

SeedTree is dynamically optimal.

- Property 1:

Reconfiguration cost of SeedTree $\leq 2 \cdot \left(\left\lceil \frac{1}{1-f} \right\rceil + 1 \right) \cdot$ Access cost of SeedTree.

- Property 2:

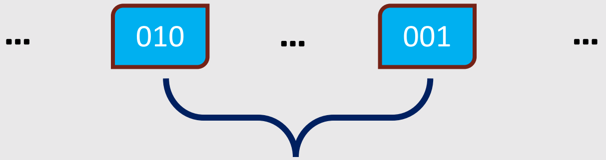
Access cost of SeedTree $\leq 2 - \log(f) \cdot$ Access cost in MRU Tree.

- Property 3:

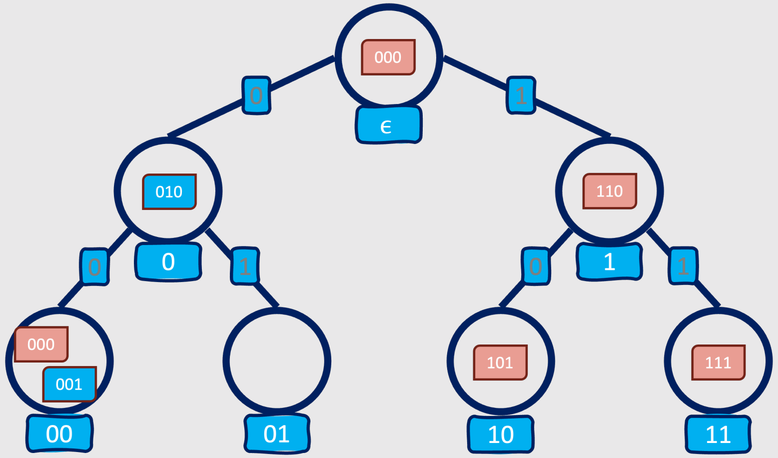
Access cost in MRU Tree $\leq (1 + e) \cdot$ Access cost of OPT.

Proof sketch:

- Potential function analysis based on inversions
- Inversion: item with lower level but accessed earlier

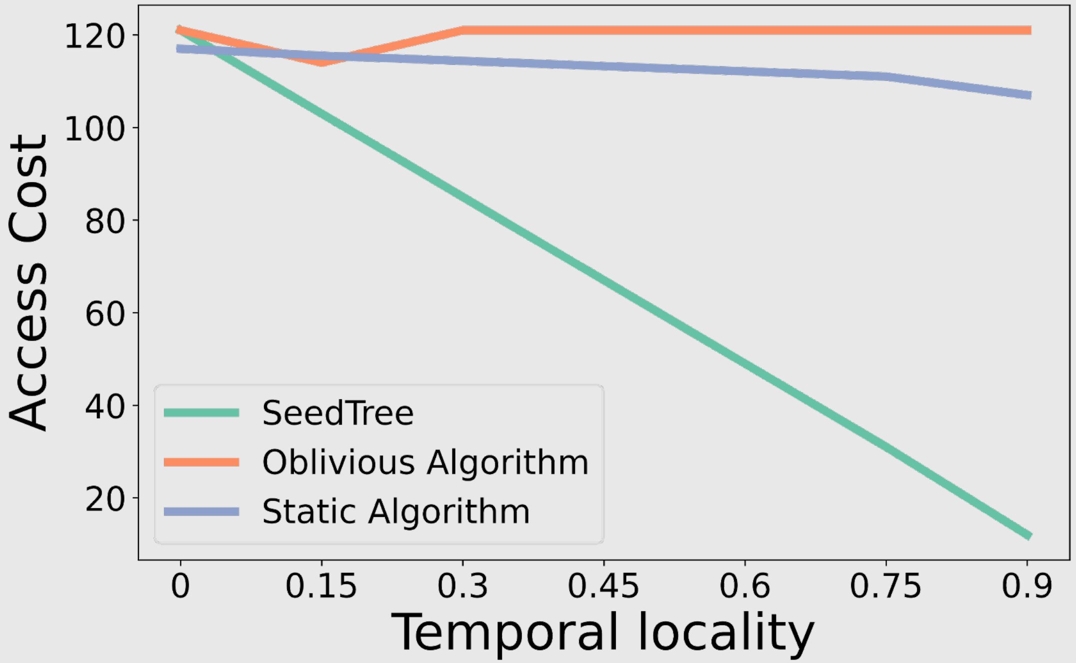


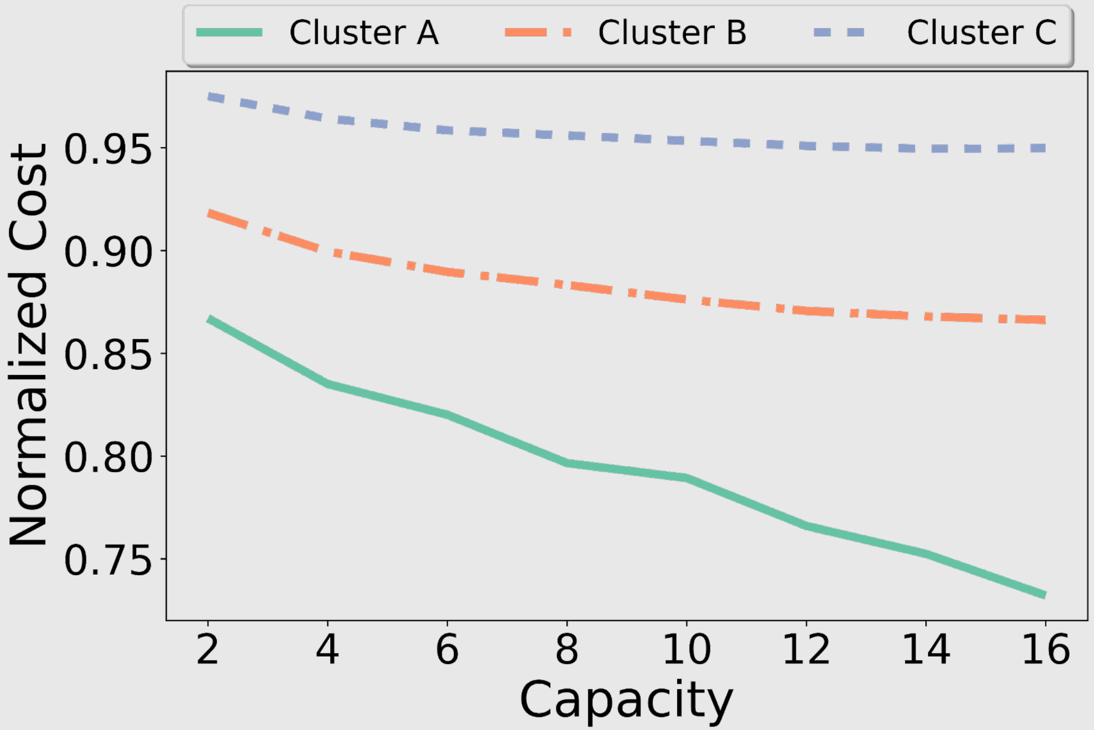
Their last accesses.



OPT's tree

Performance

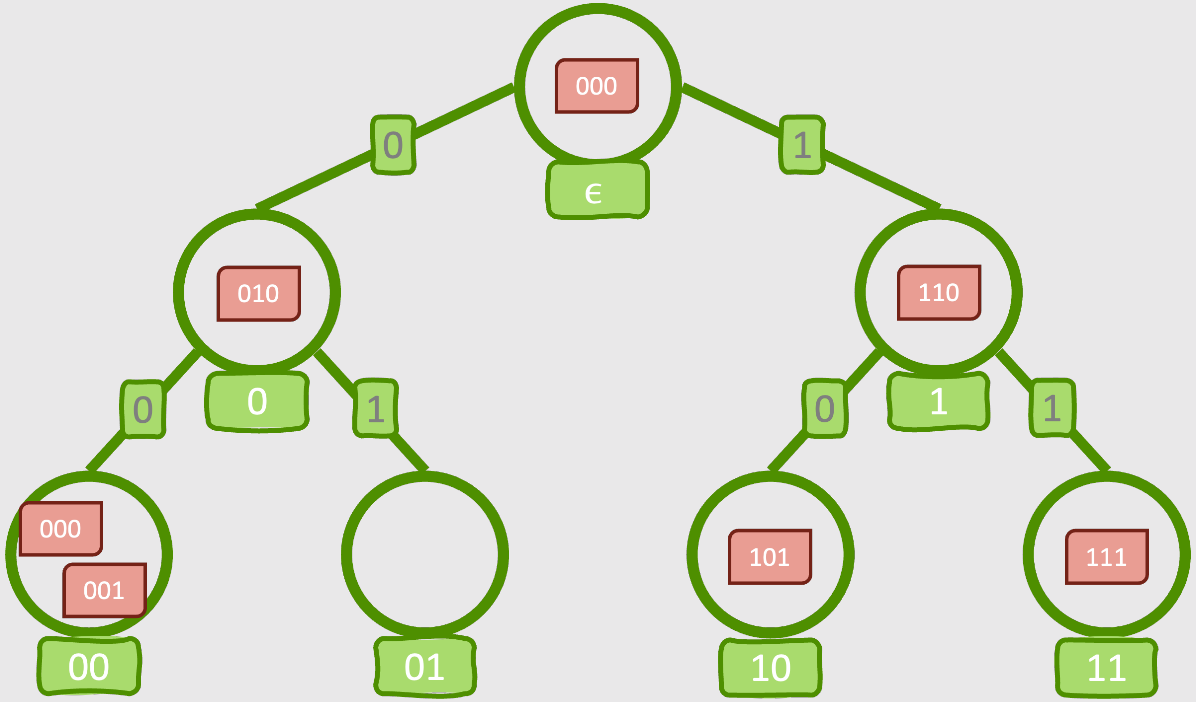




github.com/inet-tub/SeedTree

Conclusion

- Designing a constant competitive algorithm, utilizing randomization in each step.
- Introducing the notion of capacity and item movement for self-adjusting trees.
- Showing significant improvements in the algorithm given inputs with high temporal locality.



Thank You!

Full paper:
arxiv.org/pdf/2301.03074.pdf

Our group's website:
tu.berlin/en/eninet

My website:
pourdamghani.net