Towards Optimal-pass Semi-streaming Matchings and Beyond

Anish Mukherjee

Based on joint work with Slobodan Mitrović, Wen-Horng Sheu (UC Davis), and Piotr Sankowski (Warsaw)

Maximum matching problem

- Let $G = (V, E)$ be an unweighted graph
- A *matching* is a set of edges that do not share an endpoint
- **Goal:** Find the largest matching

Prior work

- The problem is extensively studied in different settings:

- **Polynomial time:** [Berge ‘57] [Edmonds ‘65] [Hopcroft, Karp ‘73] [Micali, Vazirani ‘80] [Gabow ‘90] [Kalantari, Shokoufandeh ‘95] …
- **Dynamic:** [Bernstein, Stein ‘16] [Solomon ‘16] [Bhattacharya, Kulkarni ‘19] [Behnezhad, Łącki, Mirrokni ‘19] [Behnezhad, Khanna ‘22] [Assadi, Khanna, Kiss ‘25] …
- **Semi-streaming:** [McGregor ‘05] [Ahn, Guha, '11] [Eggert, Kliemann, Munstermann, Srivastav, '12] [Ahn, Guha, '13] [Kapralov, '13] [Ahn, Guha, '18] [Tirodkar, '18] [Gamlath, Kale, Mitrović, Svensson, '19] [Assadi, Liu, Tarjan, '21] [Assadi, Jambulapati, Jin, Sidford, Tian, '22] [Fischer, Mitrović, Uitto, '22] [Huang, Su, '23] [Assadi, ‘24] …
- **Random Order Streams, Estimating Size, Vertex arrival:** [Kapralov, Khanna, Sudan ‘14] [Assadi, Khanna, Li ‘17] [Karp, Vazirani, Vazirani ‘90] …
- **Sublinear, Distributed, Parallel:** …

Semi-streaming setting

- No random access to the input graph
- Edges are presented as a stream, arriving in arbitrary order
- Reading the stream once is called a pass

✓ The algorithm can use $O(n \cdot polylog\, n)$ memory
✓ Allowed to make multiple passes over the stream

- Exact maximum matching "requires" near-quadratic space

This talk:

- **Problem:** find a $(1+\varepsilon)$-approximate maximum matching (on general graphs)
- **Goal:** minimize the number of passes

Prior work

- Constant number of passes is achievable
- [McGregor '05]: $(1/\varepsilon)^{O(1/\varepsilon)}$ passes

- Dependence on ε has been improved since then

- Two classes of graphs: **bipartite** and **general**

- Two families of studies:
  - **constant-pass:** complexity depends only on $1/\varepsilon$ (our focus)
  - **ε-efficient:** complexity depends on $\log n$ and $1/\varepsilon$

Prior work (bipartite)

- poly$(1/\varepsilon)$ is known since 2009 [Eggert, Kliemann, Munstermann, Srivastav]

Source	Pass	Weighted?
[McG, 2005]	$(1/\varepsilon)^{O(1/\varepsilon)}$	
[EKS, 2009]	$1/\varepsilon^8$	
[EKMS, 2012]	$1/\varepsilon^5$	
[AG, 2013]	$1/\varepsilon^5 \cdot \log(1/\varepsilon)$	Yes
[Kap, 2013]	$1/\varepsilon^2$ (vertex arrival)	
[AG, 2018]	$\log(n)/\varepsilon$	Yes
[ALT, 2021]	$\mathbf{1/\varepsilon^2}$	
[AJJST, 2022]	$\log(n)/\varepsilon \cdot \log(1/\varepsilon)$	
[Assadi 2024]	$\log(n)/\varepsilon$	Yes

Prior work (general)

- $poly(1/\varepsilon)$ is only known recently
  [Fischer, Mitrović, Uitto, 2022]
- Huge gap between bipartite and general graphs
- Bipartite graphs: $1/\varepsilon^2$ passes
  [Assadi, Liu, Tarjan, 2021]
- General graphs: $1/\varepsilon^{19}$ passes
  [Fischer, Mitrović, Uitto, 2022]

Source	Pass	Weight?
[McG, 2005]	$(1/\varepsilon)^{O(1/\varepsilon)}$	
[AG, 2011]	$\log(n)/\varepsilon^7 \cdot \log(1/\varepsilon)$	
[AG, 2013]	$\log(n)/\varepsilon^4$	Yes
[AG, 2018]	$\log(n)/\varepsilon$	Yes
[Tir, 2018]	$\exp(1/\varepsilon)$	
[GKMS, 2019]	$\exp(1/\varepsilon^2)$	Yes
[FMU, 2022]	$\mathbf{1/\varepsilon^{19}}$	
[HS, 2023]	poly(1/ε) but > $1/\varepsilon^{19}$	Yes
[Assadi 2024]	$\log(n)/\varepsilon$	Yes

Our result

- $1/\varepsilon^6$ - pass algorithm
- Bridging the gap between bipartite and general graphs
- Simpler approach
- Simpler analysis

Source	Pass	Weight?
[McG05]	$(1/\varepsilon)^{O(1/\varepsilon)}$	
[AG11]	$\log(n)/\varepsilon^7 \cdot \log(1/\varepsilon)$	
[AG13]	$\log(n)/\varepsilon^4$	Yes
[AG18]	$\log(n)/\varepsilon$	Yes
[AG18]	$\log(n)/\varepsilon$	Yes
[Tir18]	$\exp(1/\varepsilon)$	
[GKMS19]	$\exp(1/\varepsilon^2)$	Yes
[FMU22]	$1/\varepsilon^{19}$	
[HS23]	more than $1/\varepsilon^{19}$	Yes
[Assadi24]	$\log(n)/\varepsilon$	Yes
[this talk]	$\mathbf{1/\varepsilon^6}$	

Remark: other models

Our algorithm can be simulated in other computational models [FMU22]

- Improve round complexity in MPC and CONGEST models by $1/\varepsilon^{13}$ factor
- Mitrović and Sheu ’25 brings it down to $1/\varepsilon^{7}$ and $1/\varepsilon^{10}$, respectively

Warm-up: *Bipartite graphs*

Based on [Eggert, Kliemann, Munstermann, Srivastav ’12]

Definition

- *Free node*: unmatched vertex
- *Alternating path*: path alternates between matched and unmatched edges
- *Augmenting path*: alternating path from a free node to another

Starting point - short augmenting paths

Claim

Let M be a matching and let Y be a maximal collection of vertex disjoint augmenting paths of length at most $2/\varepsilon$. If $|Y| < \varepsilon^2|M|/6$, then M is a $(1+\varepsilon)$-approximate maximum matching.

[Kalantari, Shokoufandeh ‘95] [McGregor ‘05] [Eggert, Kliemann, Munstermann, Srivastav ‘12]

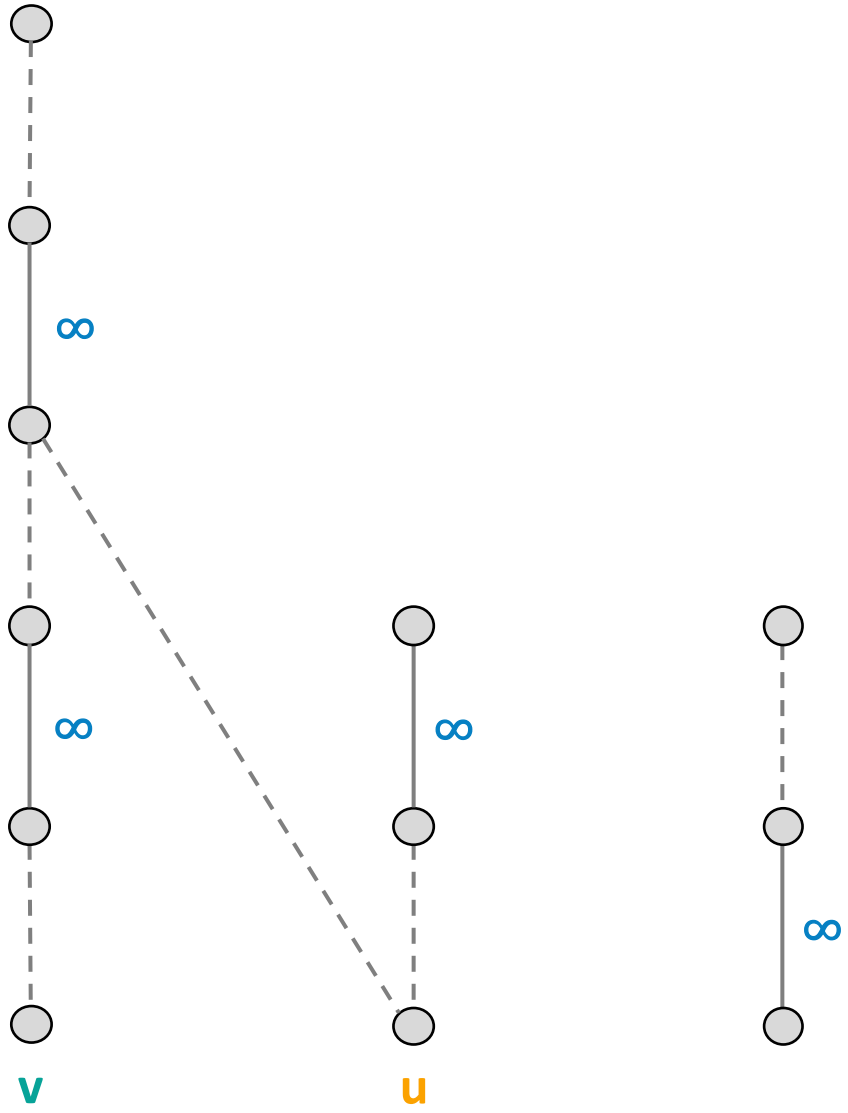
Idea: Execute truncated DFS from *free* nodes.

Idea: Execute truncated DFS from *free* nodes.

Matched ——————

Unmatched - - - - -

Each **matched** edge has a **distance label**.

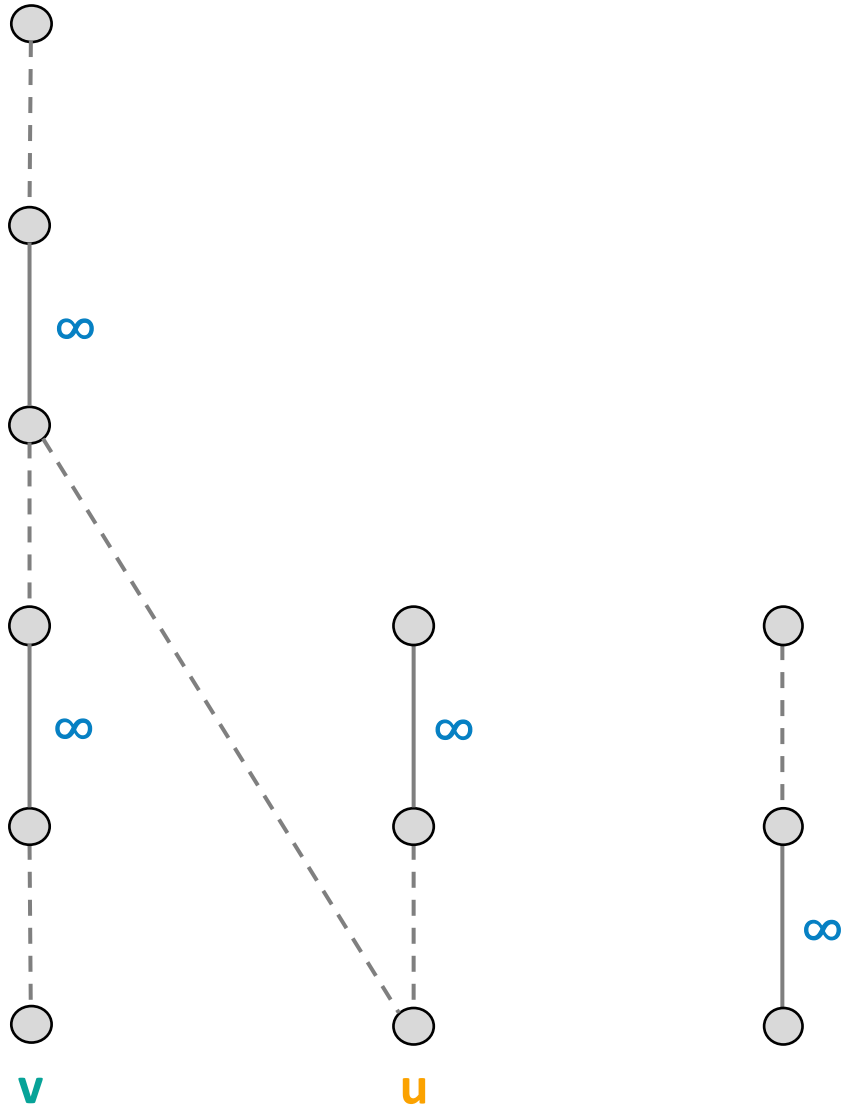


Idea: Execute truncated DFS from *free* nodes.

Matched ——————

Unmatched - - - - -

Each **matched** edge
has a **distance label**.



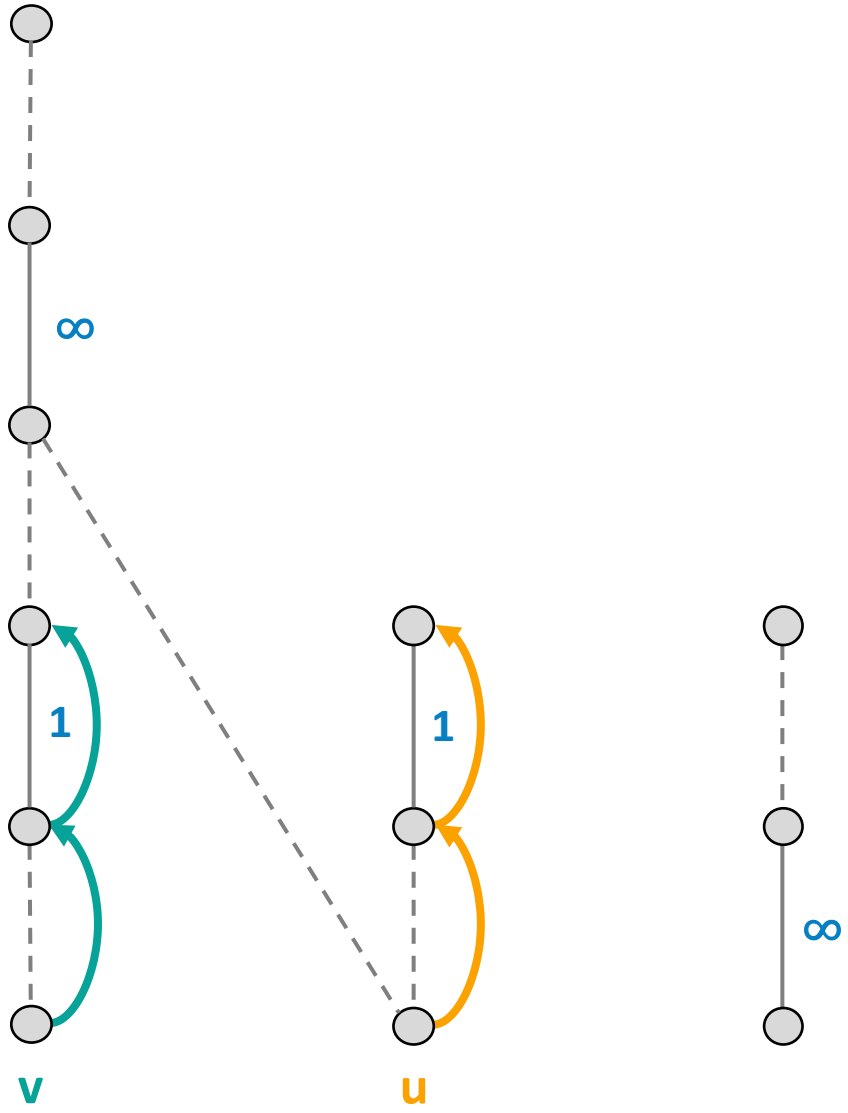
- Meaning of label: current shortest distance
- Each free node maintains an **active path**
  (DFS search path)

Idea: Execute truncated DFS from *free* nodes.

Matched ——————

Unmatched - - - - -

Each **matched** edge has a distance label.



- Meaning of label: current shortest distance
- Each free node maintains an active path (DFS search path)

Each pass: Extend by length-2 paths

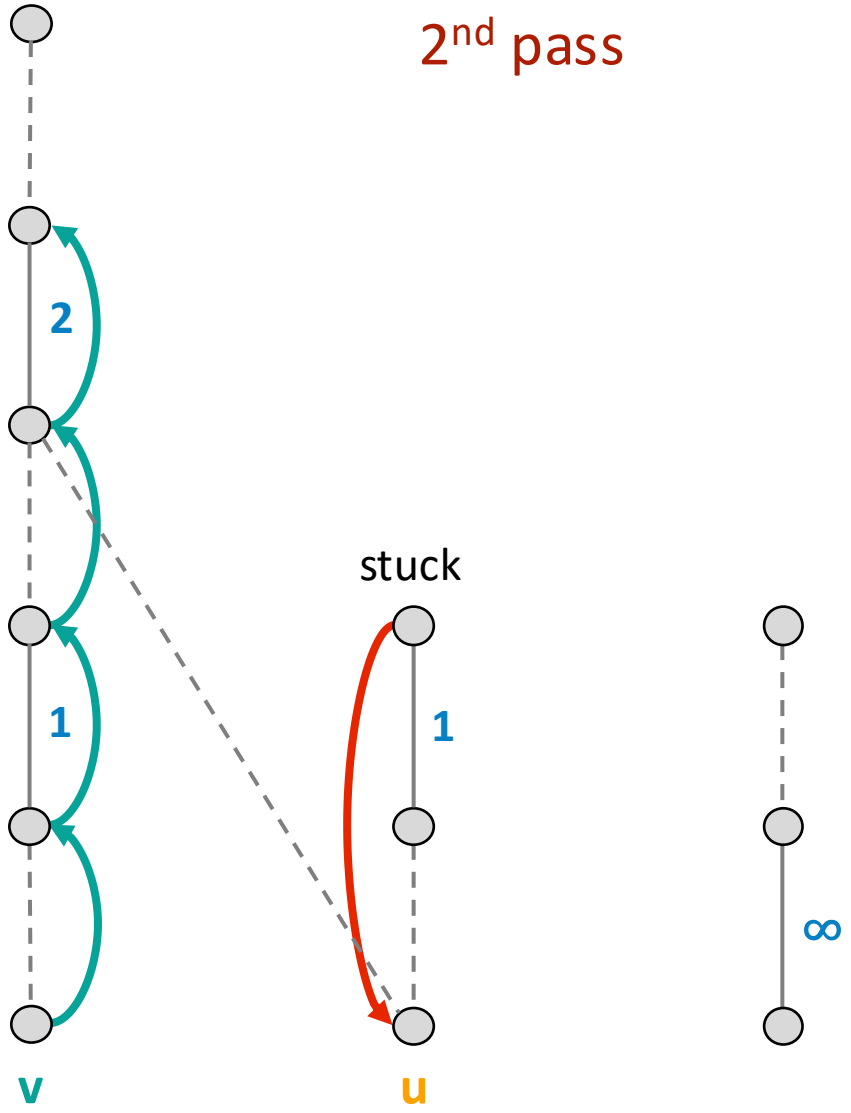
- Scan unmatched edges
- Extend when distance label can be reduced

Idea: Execute truncated DFS from *free* nodes.

Matched ——————

Unmatched - - - - -

Each **matched** edge has a distance label.



- Meaning of label: current shortest distance
- Each free node maintains an active path (alternating path)

Each pass: Extend by length-2 paths

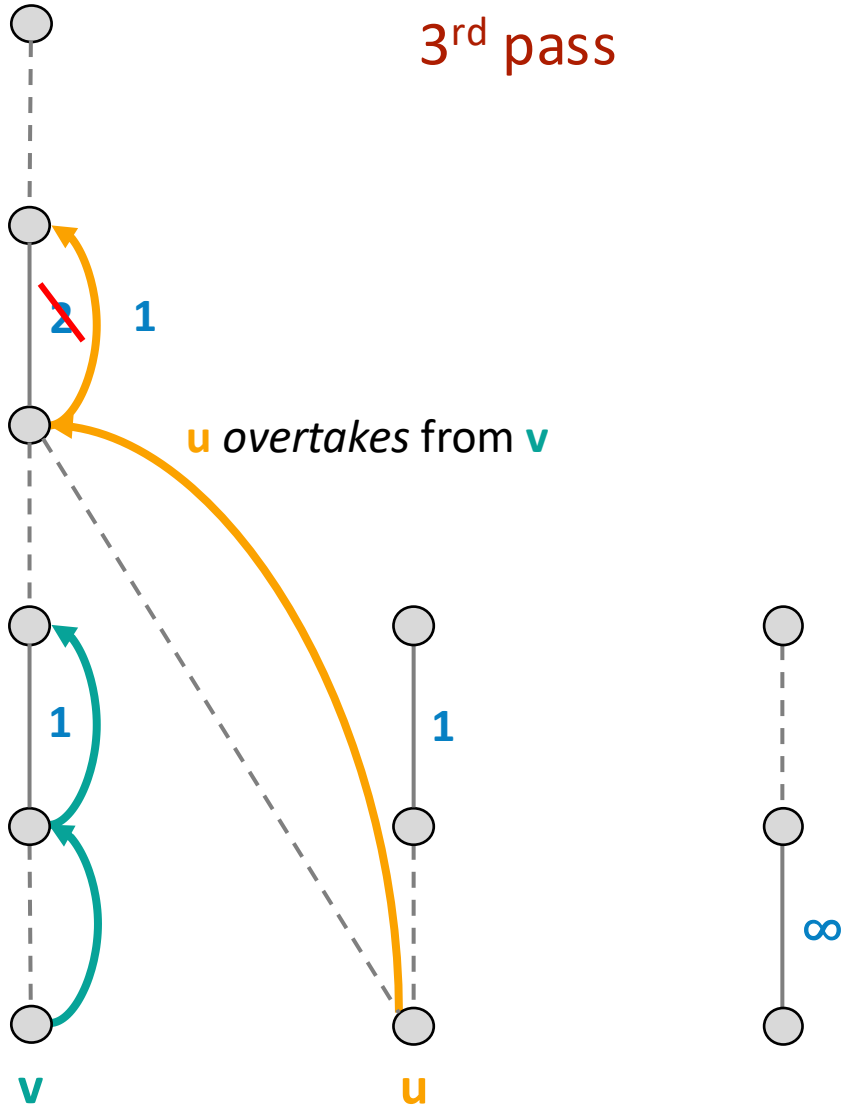
- Scan unmatched edges
- Extend when distance label can be reduced
- Backtrack if stuck

Idea: Execute truncated DFS from *free* nodes.

Matched ——————

Unmatched - - - - -

Each **matched** edge has a distance label.



- Meaning of label: current shortest distance
- Each free node maintains an active path (alternating path)

Each pass: Extend by length-2 paths

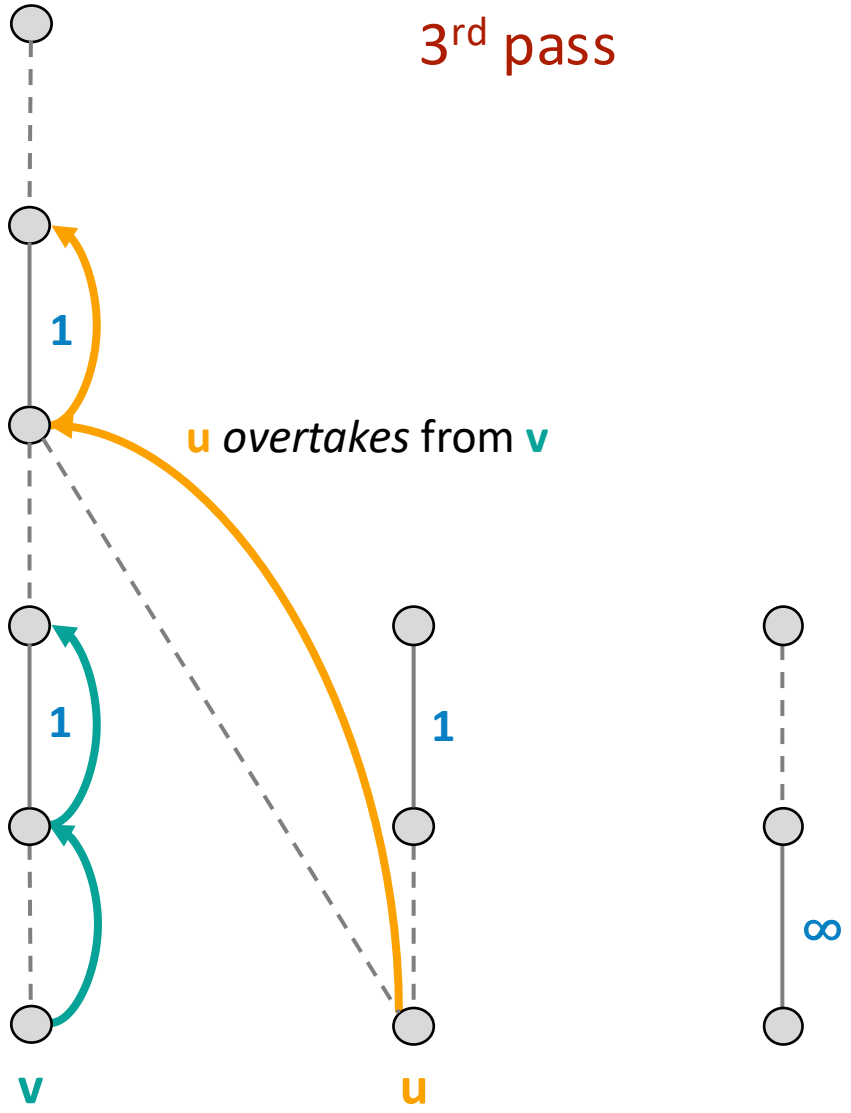
- Scan unmatched edges
- Extend when distance label can be reduced
- Backtrack if stuck
- Can overtake another path to reduce label

Idea: Execute truncated DFS from *free* nodes.

Matched ——————

Unmatched - - - - -

Each **matched** edge has a distance label.



- Meaning of label: current shortest distance
- Each free node maintains an active path (alternating path)

Each pass: Extend by length-2 paths

- Scan unmatched edges
- Extend when distance label can be reduced
- Backtrack if stuck
- Can overtake another path to reduce label

Idea: Execute truncated DFS from *free* nodes.

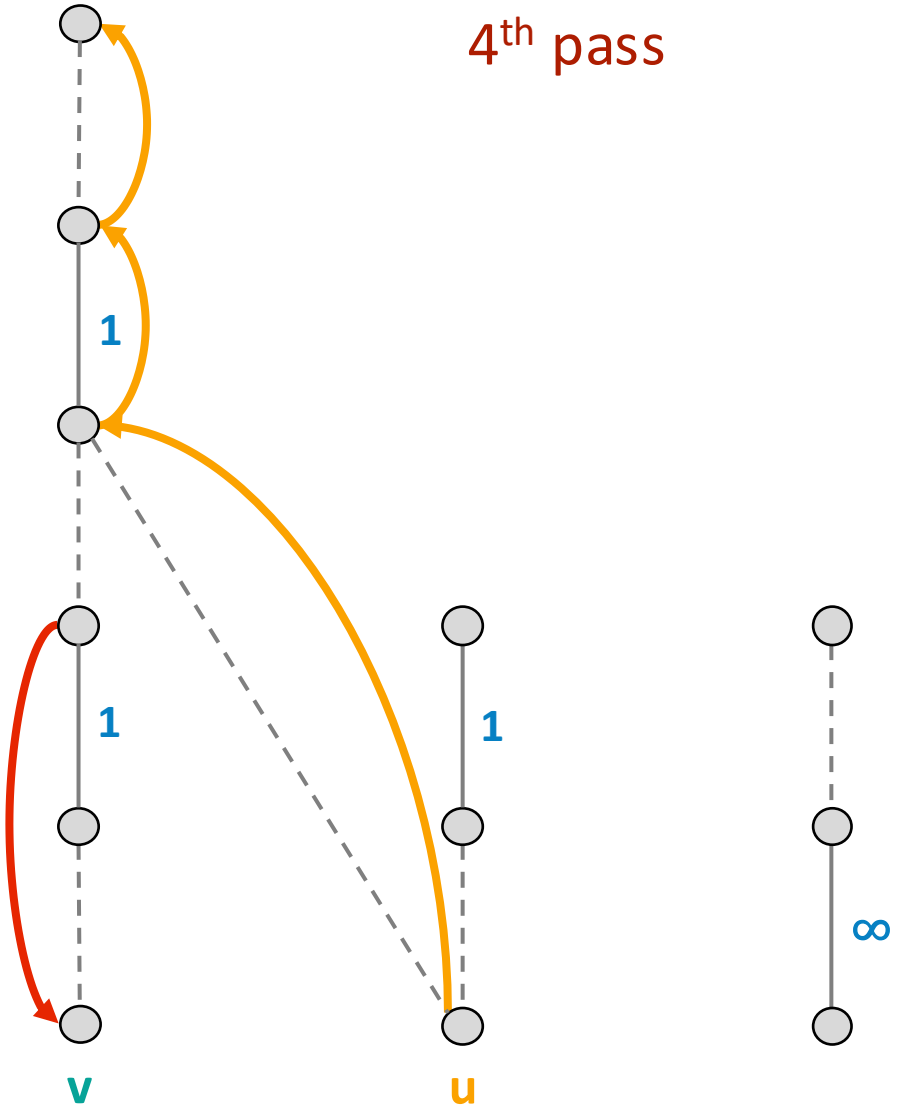
Matched ——————

Unmatched - - - - -

Each **matched** edge has a distance label.

Augmentation

4th pass



- Meaning of label: current shortest distance
- Each free node maintains an active path (alternating path)

Each pass: Extend by length-2 paths

- Scan unmatched edges
- Extend when distance label can be reduced
- Backtrack if stuck
- Can overtake another path to reduce label

Idea: Execute truncated DFS from *free* nodes.

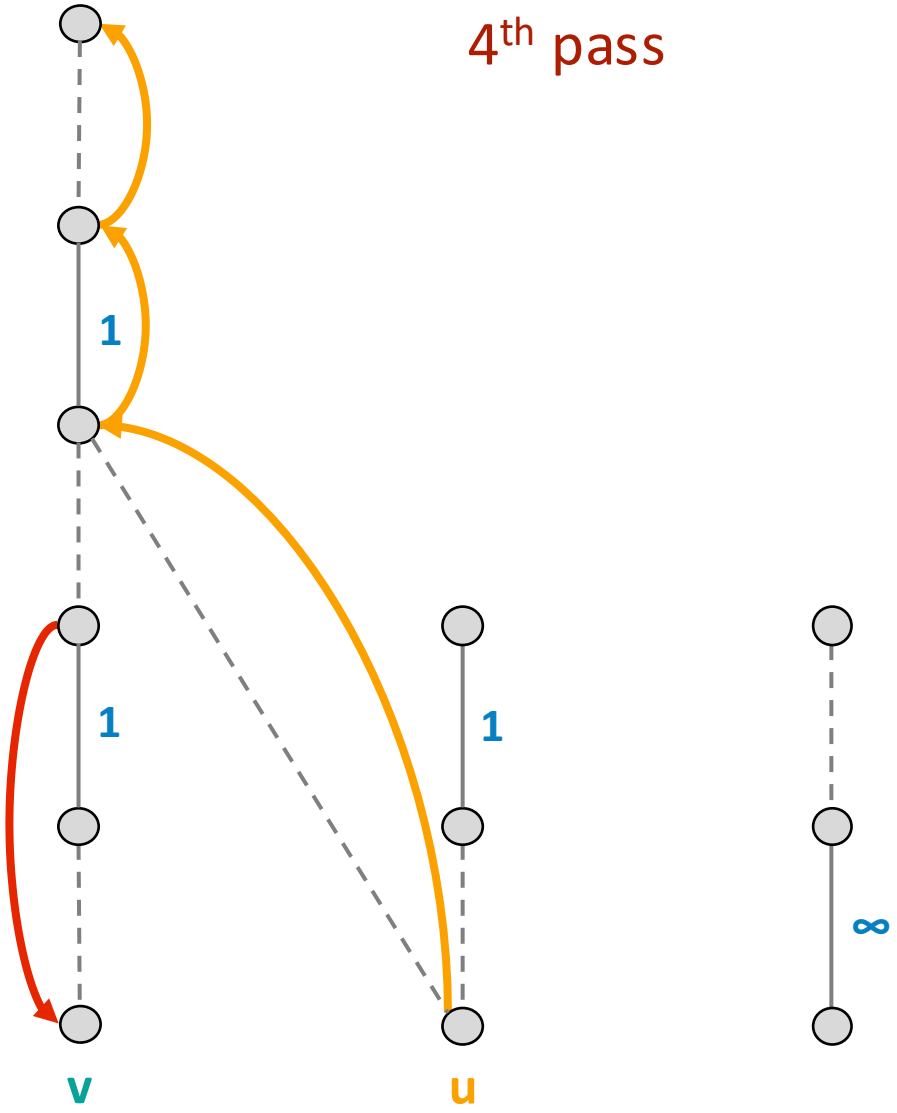
Matched ——————

Unmatched - - - - -

Each **matched** edge has a **label**.

Augmentation

4th pass



Analysis

- Runs in poly$(1/\varepsilon)$ passes
- Finds an "almost" maximal set of short augmenting paths

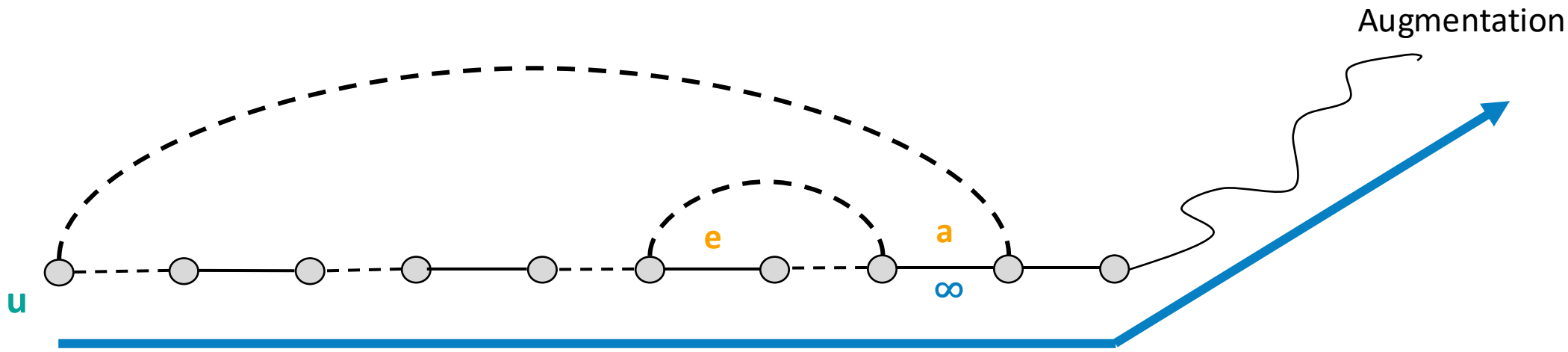
General graphs

Free node can block itself due to odd cycles

General graphs: *tricky example*

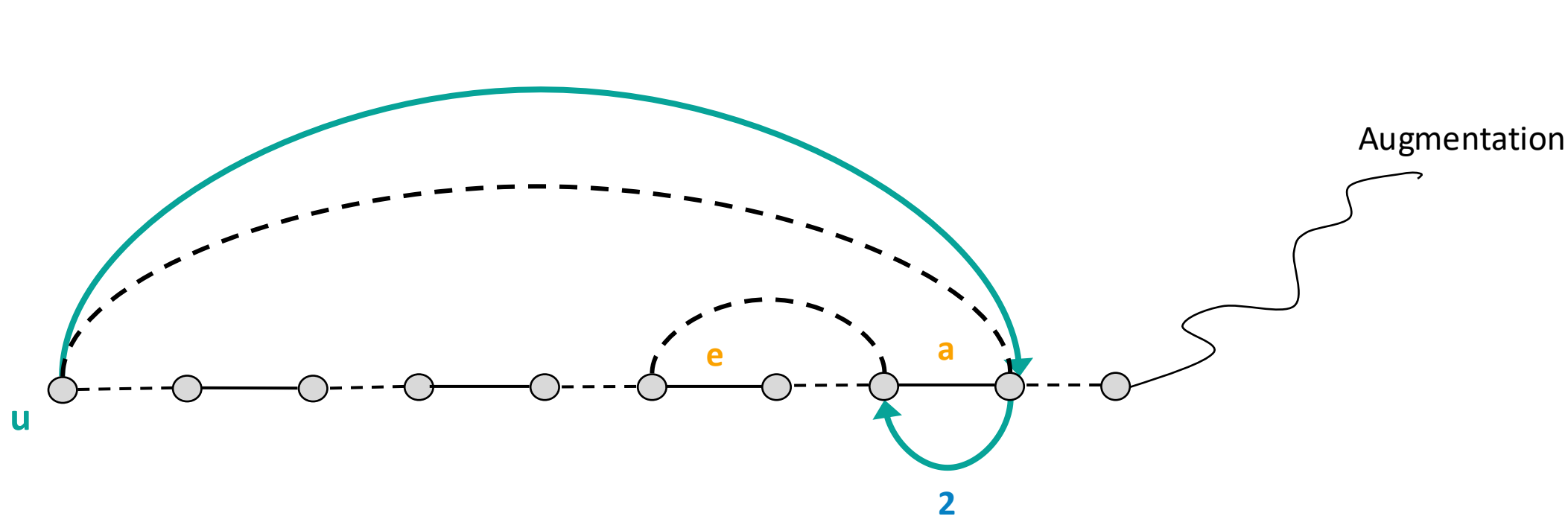
Matched

Unmatched

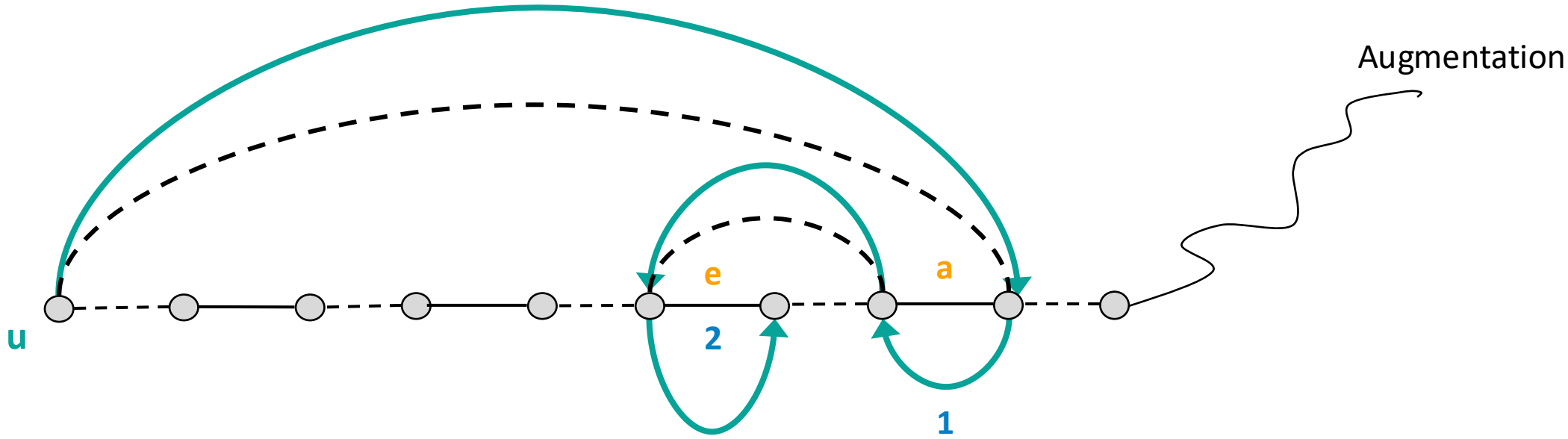


Goal: find this augmentation

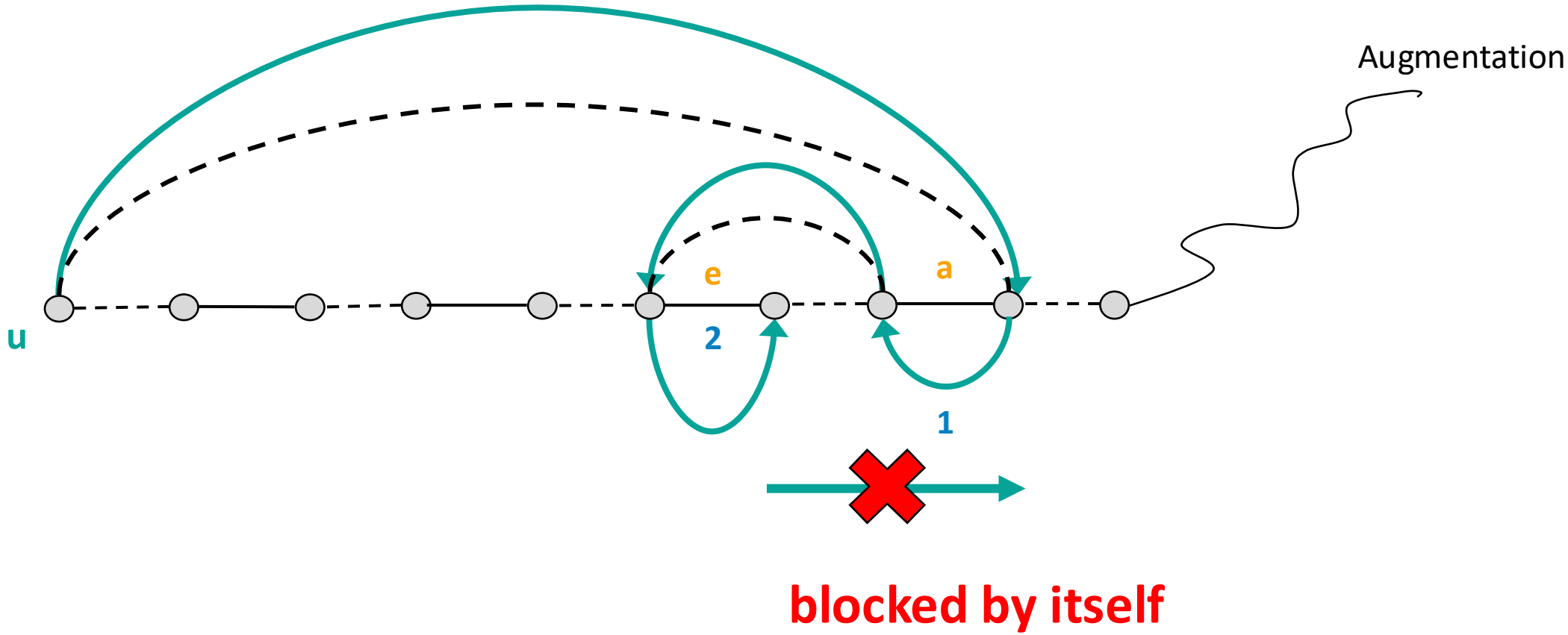
General graphs: *tricky example*



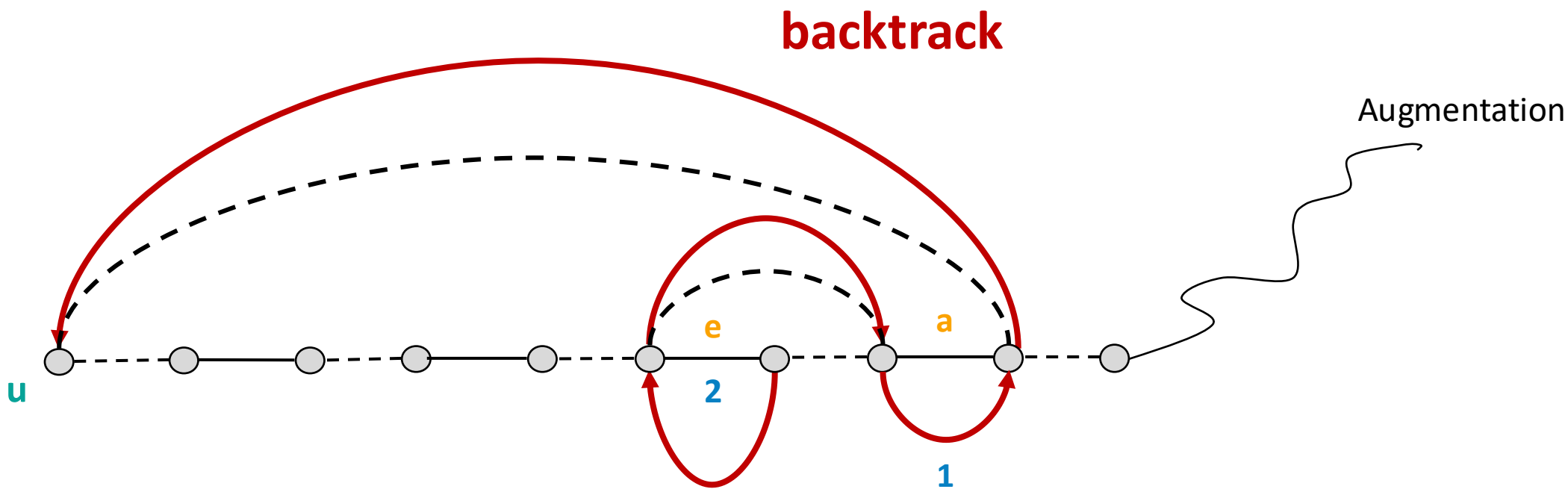
General graphs: *tricky example*



General graphs: *tricky example*



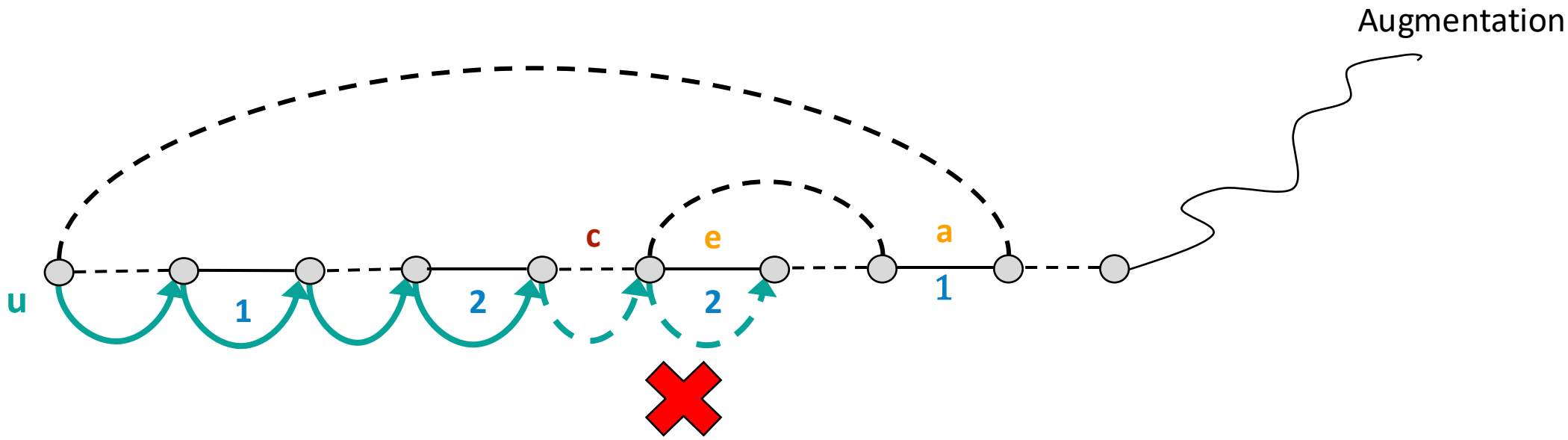
General graphs: *tricky example*



General graphs: *tricky example*

Matched

Unmatched

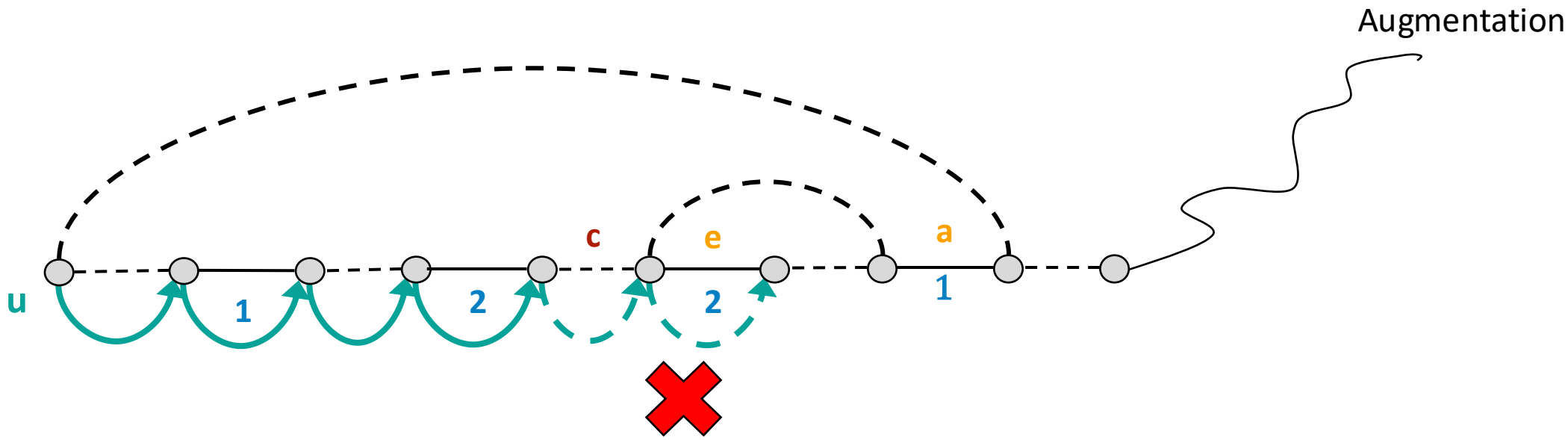


Cannot extend due to small label

General graphs: *tricky example*

Matched

Unmatched



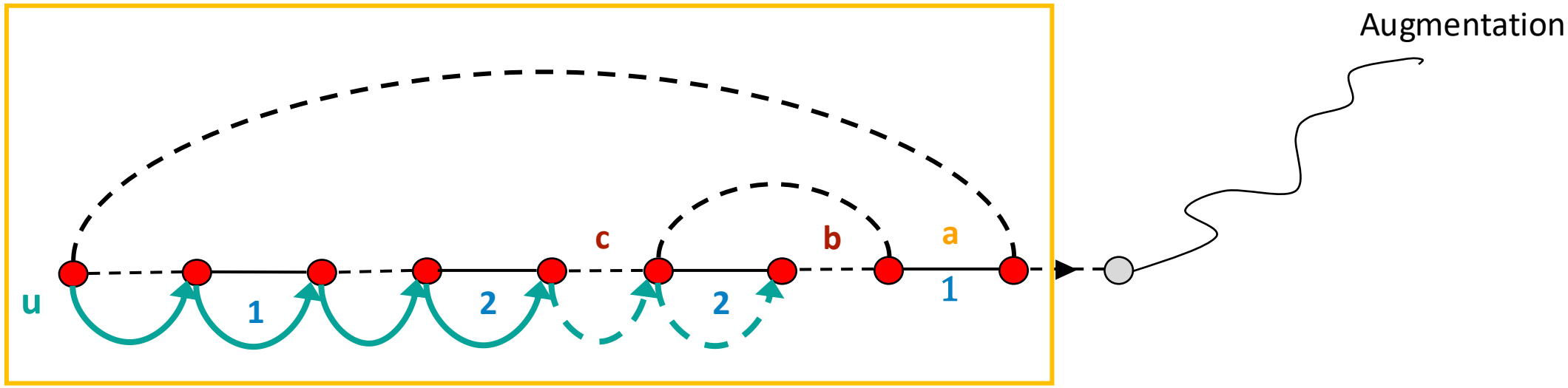
Cannot extend due to small label

→ augmentation is never found

General graphs: *trickier example*

Matched

Unmatched

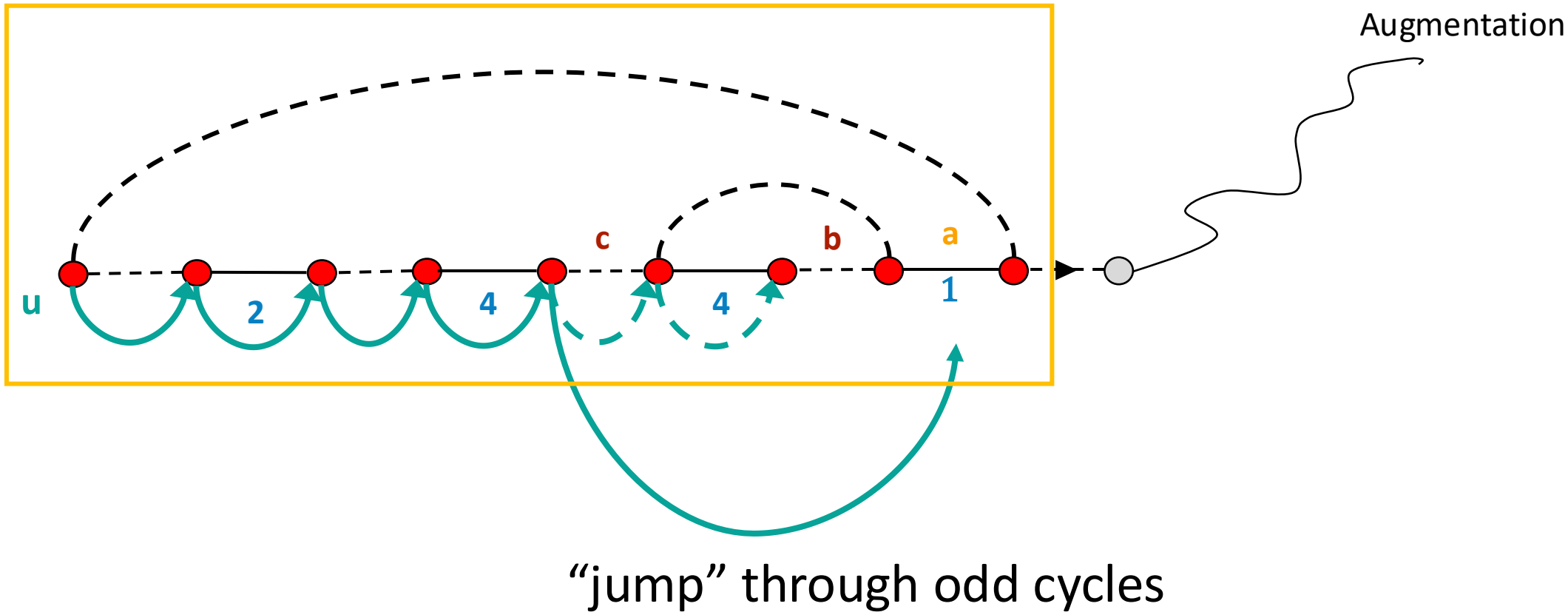


[FMU]'s approach:
Store all visited vertices and edges
to detect odd cycles

General graphs: *trickier example*

Matched

Unmatched

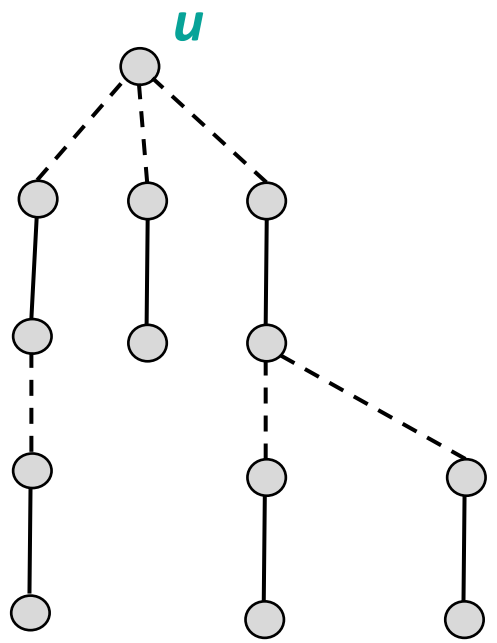


"jump" through odd cycles

General graphs: our approach

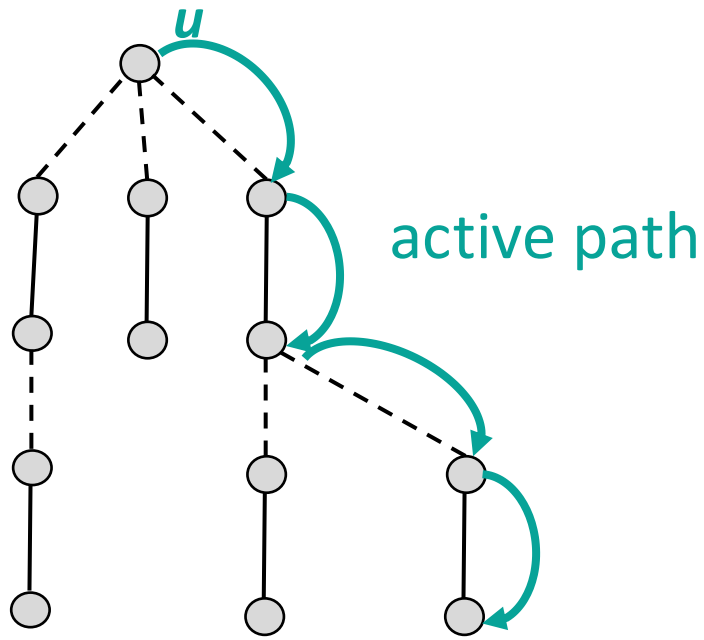
Idea: Maintain alternating trees

- Each free node grows alternating trees
- Trees are vertex-disjoint



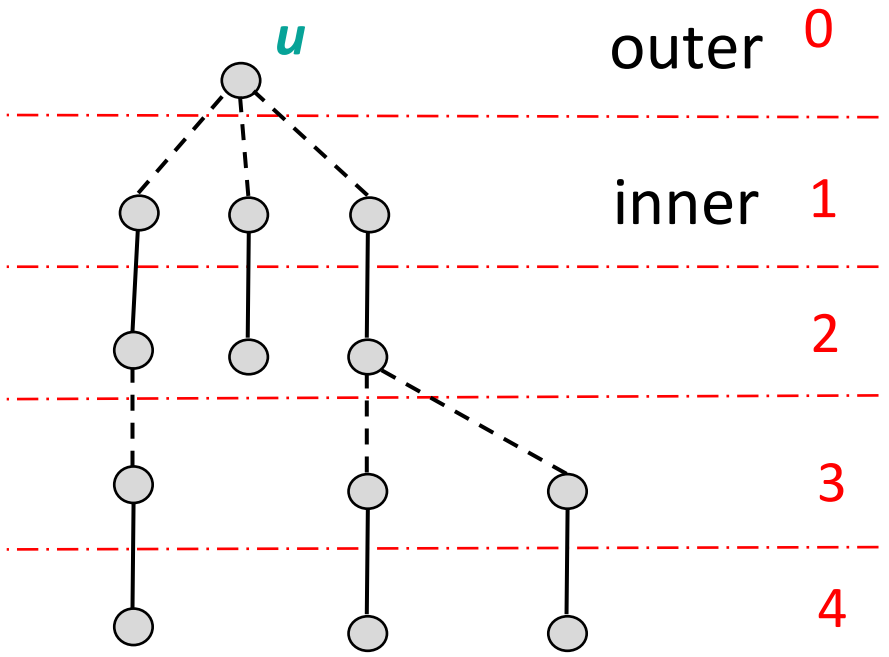
Idea: Maintain alternating trees

- Each free node grows alternating trees
- Trees are vertex-disjoint
- Each tree has an active path



Idea: Maintain alternating trees

- Each free node grows alternating trees
- Trees are vertex-disjoint
- Each tree has an active path

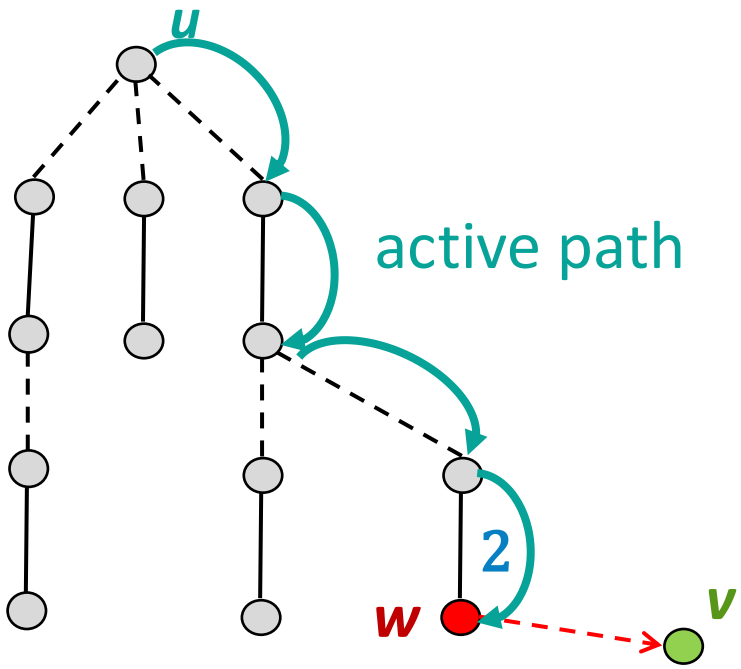


- Even layers: outer vertices
- Odd layers: inner vertices

- Root: outer vertex
- Each inner vertex has one child

Idea: Maintain alternating trees

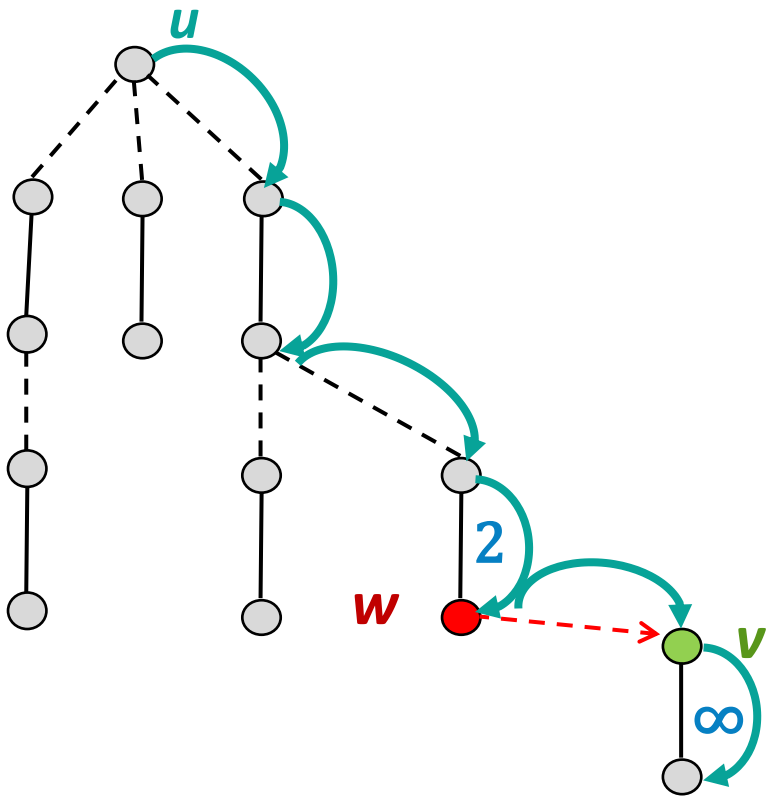
- Read edges (***w***, ***v***) from stream
- Focus on edges from an active path



Idea: Maintain alternating trees

- Read edges (***w***, ***v***) from stream
- Focus on edges from an active path

Case 1: ***v*** is not in any tree → **Extend**

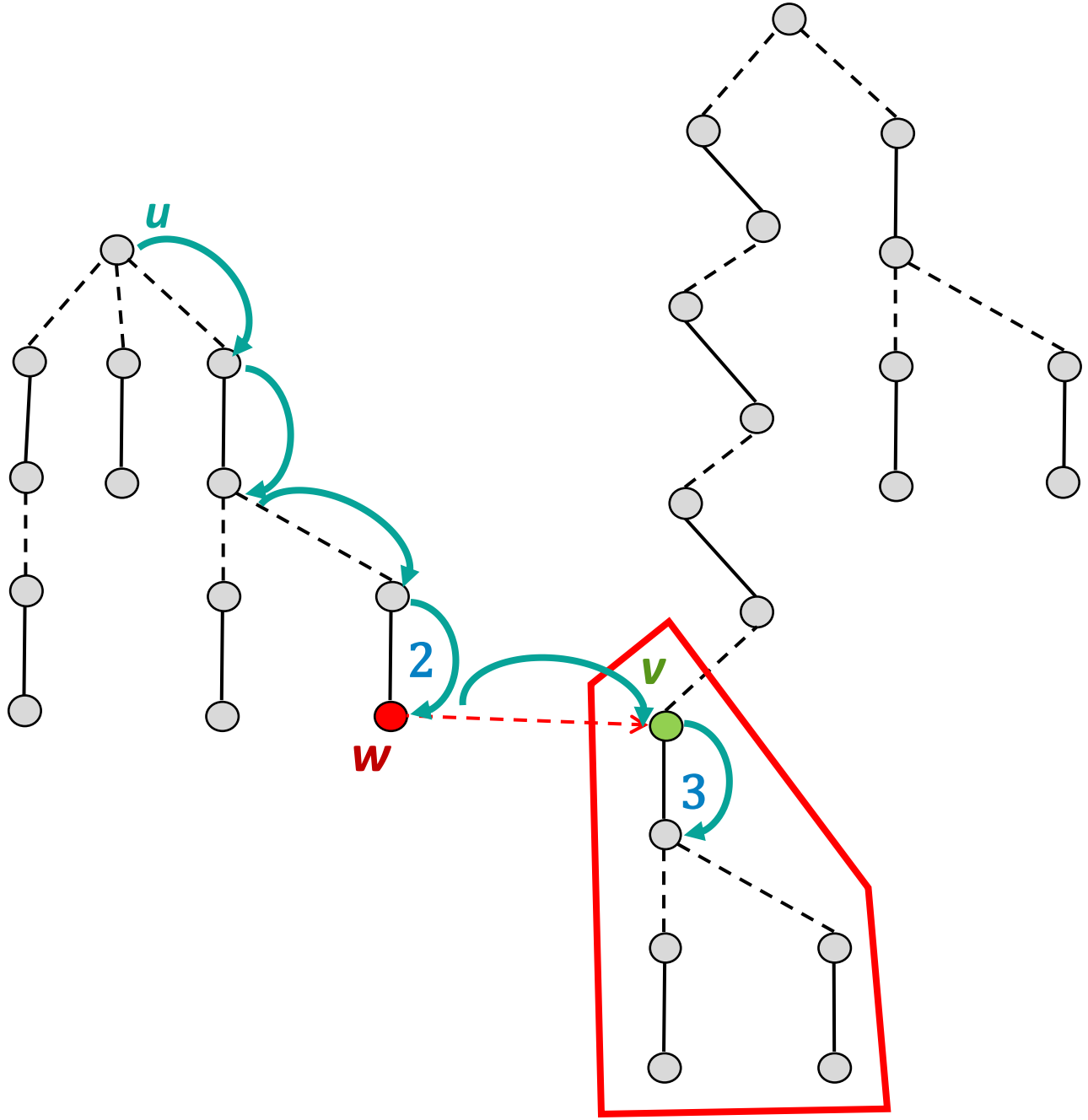


Idea: Maintain alternating trees

- Read edges (***w***, ***v***) from stream
- Focus on edges from an active path

Case 1: ***v*** is not in any tree → **Extend**

Case 2: ***v*** is an inner vertex → **Overtake**
(Also take the subtree of ***v***)

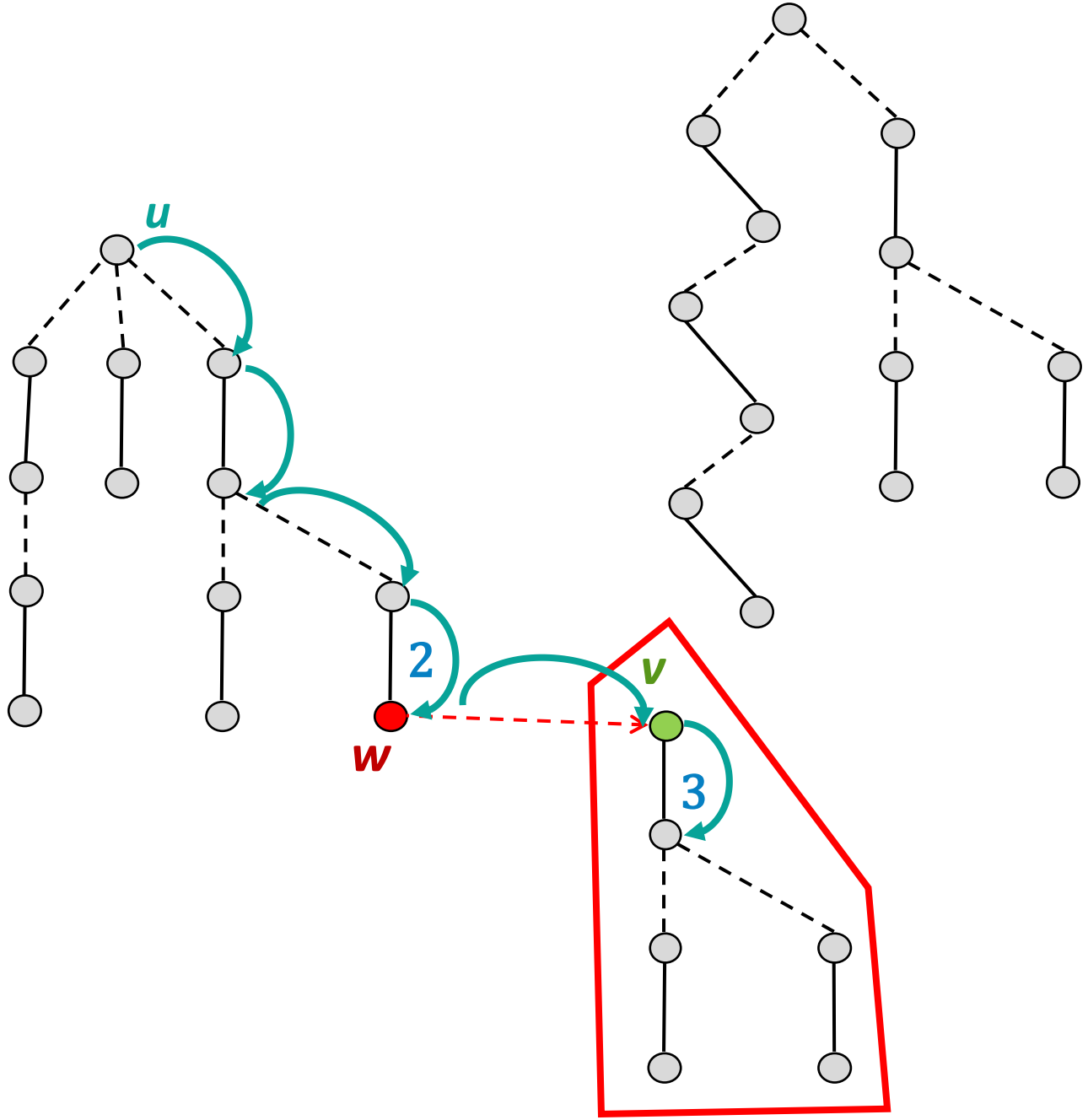


Idea: Maintain alternating trees

- Read edges (***w***, ***v***) from stream
- Focus on edges from an active path

Case 1: ***v*** is not in any tree → **Extend**

Case 2: ***v*** is an inner vertex → **Overtake**
(Also take the subtree of ***v***)



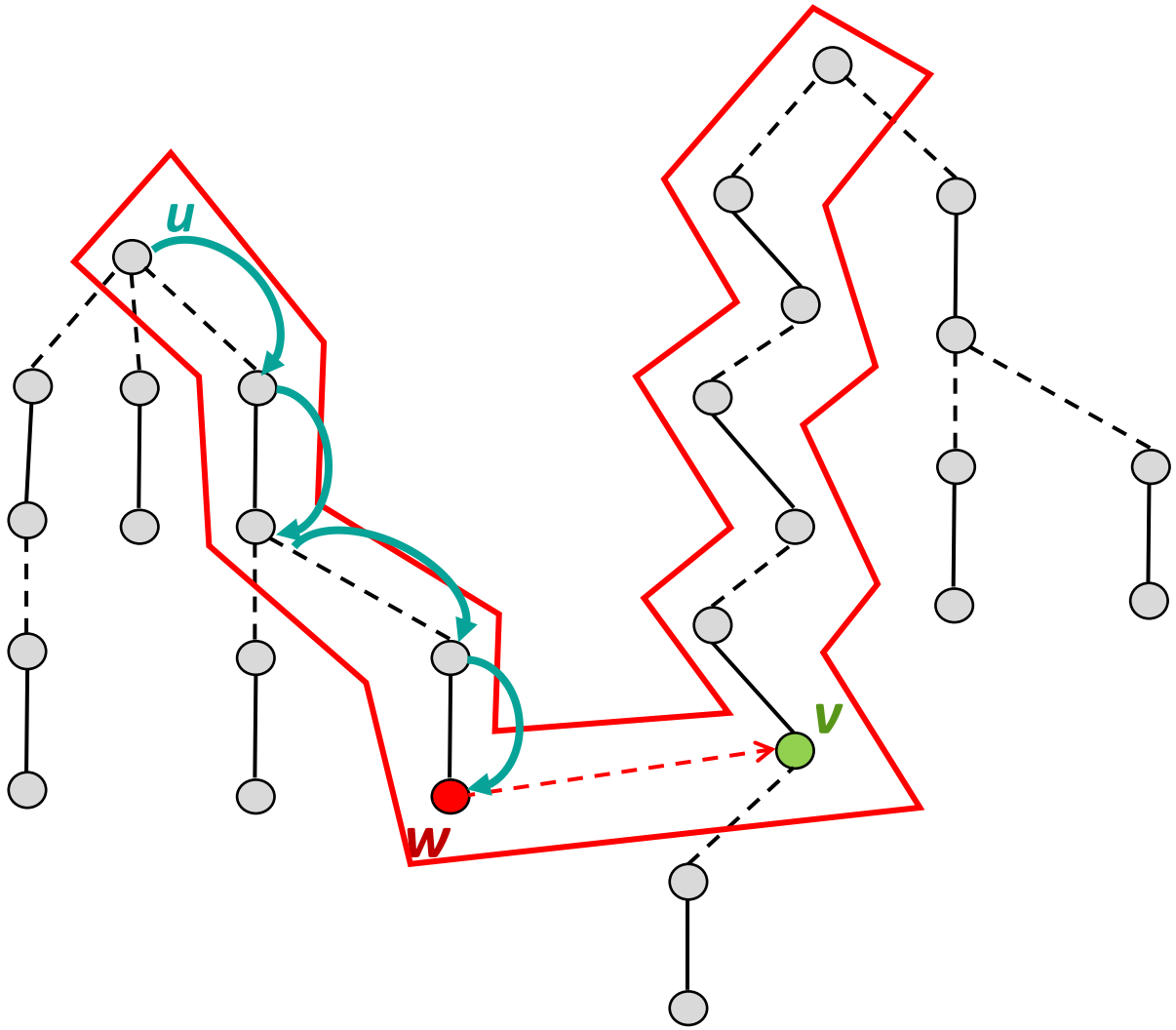
Idea: Maintain alternating trees

- Read edges (***w***, ***v***) from stream
- Focus on edges from an active path

Case 1: ***v*** is not in any tree → **Extend**

Case 2: ***v*** is an inner vertex → **Overtake**
(Also take the subtree of ***v***)

Case 3: ***v*** is an outer vertex of another tree
→ **Augmentation found**
(remove both trees)



Idea: Maintain alternating trees

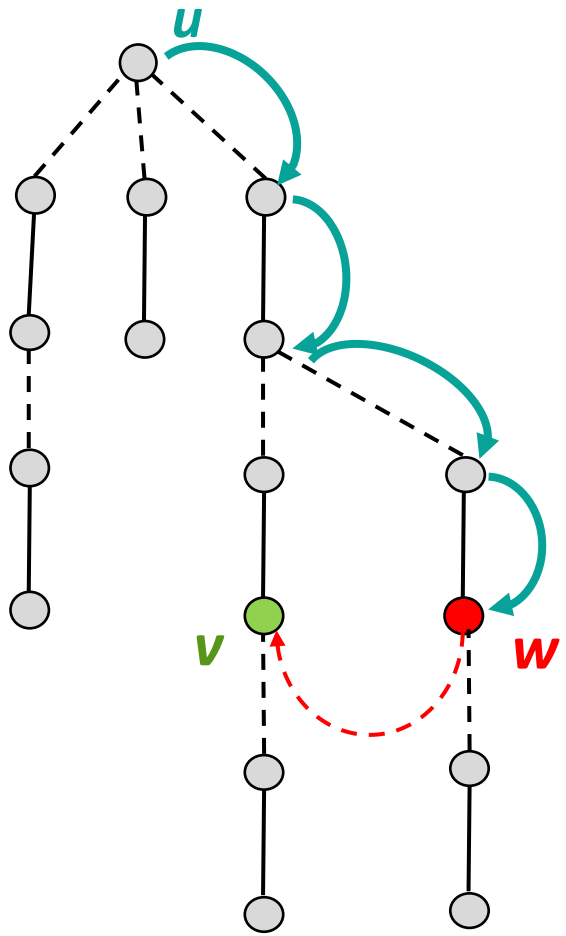
- Read edges (***w***, ***v***) from stream
- Focus on edges from an active path

Case 1: ***v*** is not in any tree → **Extend**

Case 2: ***v*** is an inner vertex → **Overtake**
(Also take the subtree of ***v***)

Case 3: ***v*** is an outer vertex of another tree
→ **Augmentation found**
(remove both trees)

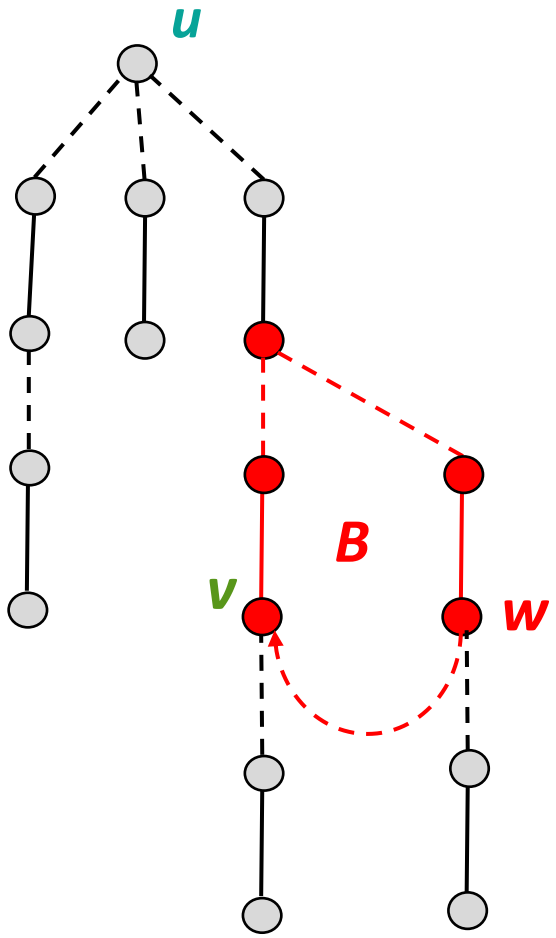
Case 4: ***v*** is an outer vertex of the same tree??
(odd cycle!)



Idea: Blossom contraction

Claim (part 1) [Edmonds, 1965]

Let T be an alternating tree. An edge connecting two outer vertices of T forms a blossom.

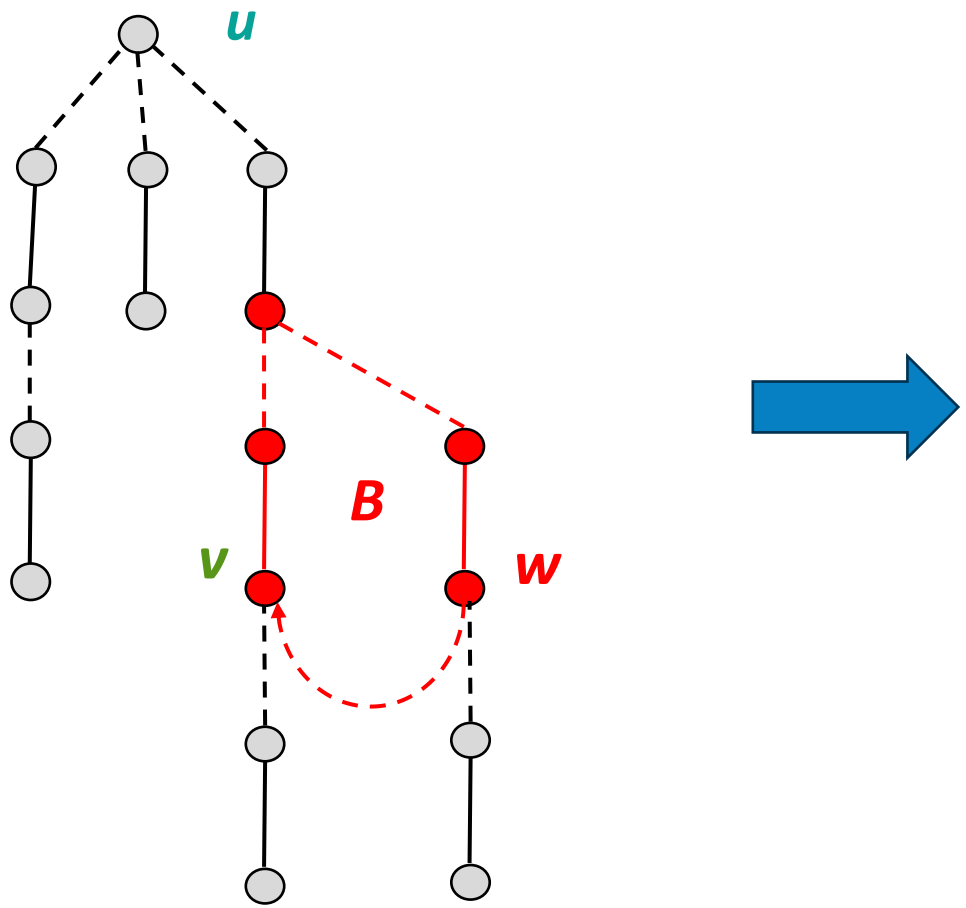


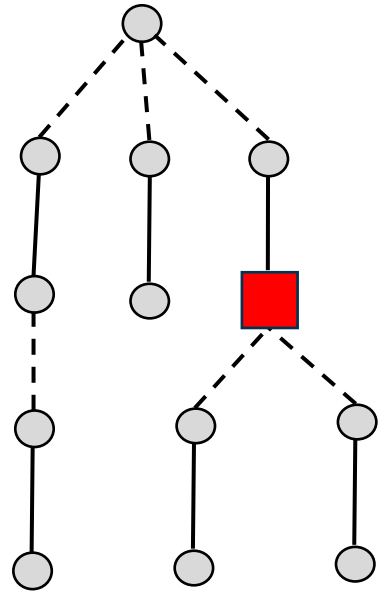
Definition

A blossom is a subgraph that forms an odd cycle with exactly one unmatched vertex.

Claim (part 2) [Edmonds, 1965]

By contracting such a blossom, T remains an alternating tree.





Analysis

Find "almost" maximal set of short augmenting paths

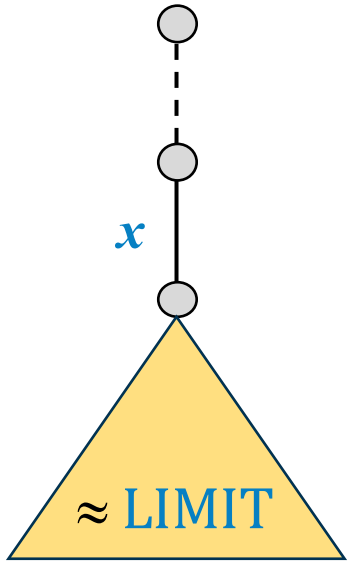
A tree cannot “grow” beyond poly(1/ε)

(Recall that a tree is **removed** after an augmentation.)

A tree cannot “grow” beyond poly(1/ε)

Idea: A tree gets frozen when its size reaches $\mathrm{LIMIT} = 1/\varepsilon^2$.

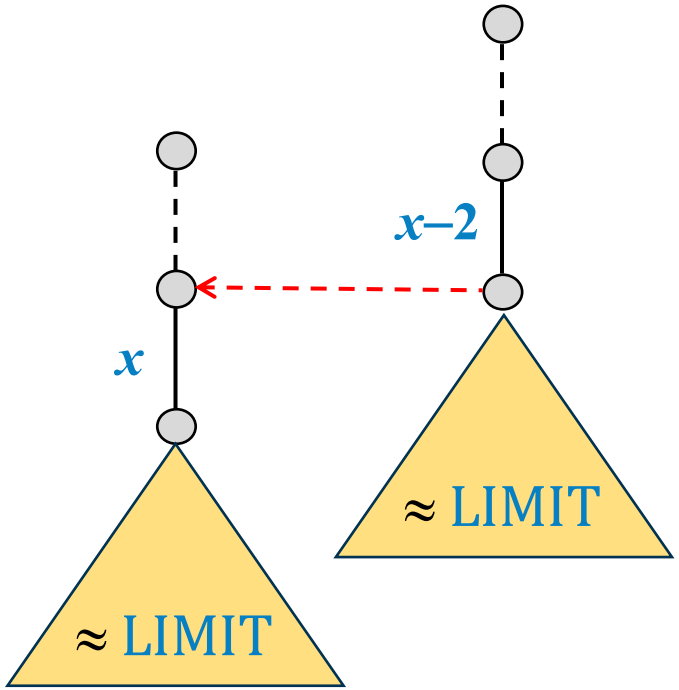
So, no tree goes beyond LIMIT ?



A tree cannot "grow" beyond poly(1/ε)

Idea: A tree gets frozen when its size reaches $\mathrm{LIMIT} = 1/\varepsilon^2$.

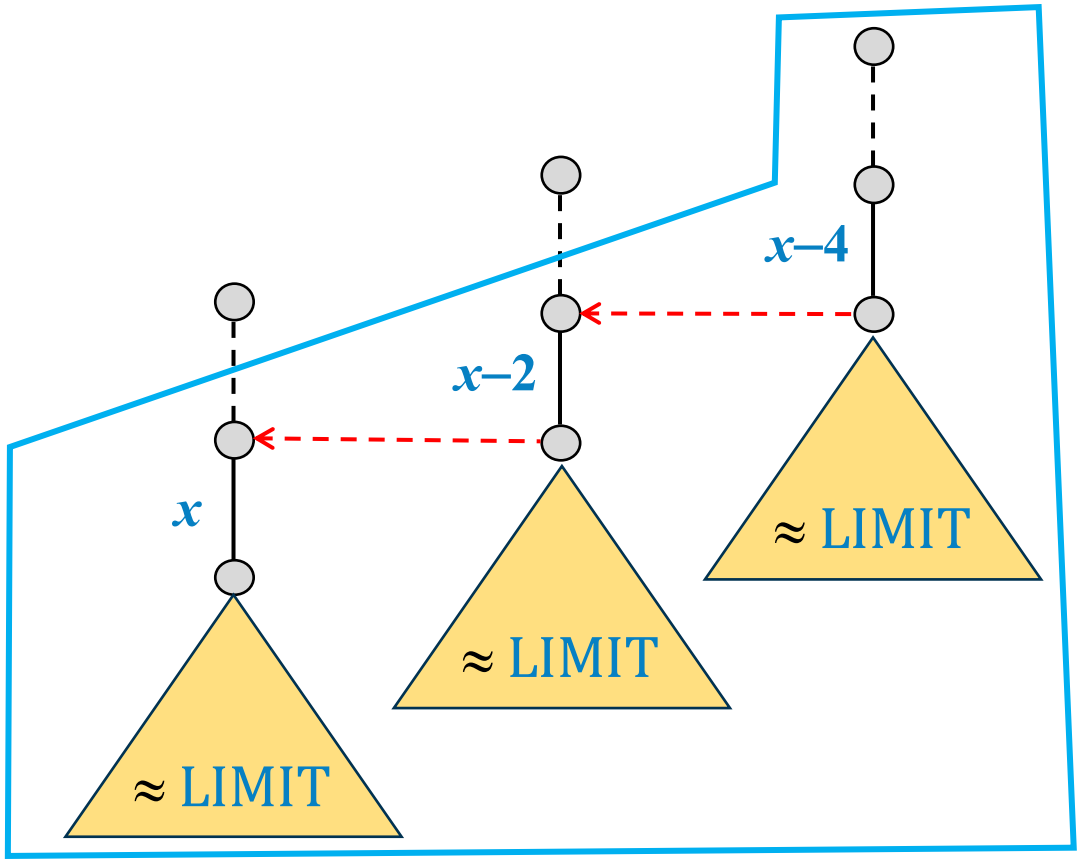
So, no tree goes beyond LIMIT ?



A tree cannot "grow" beyond poly(1/ε)

Idea: A tree gets frozen when its size reaches $\mathrm{LIMIT} = 1/\varepsilon^2$.

So, no tree goes beyond LIMIT ?

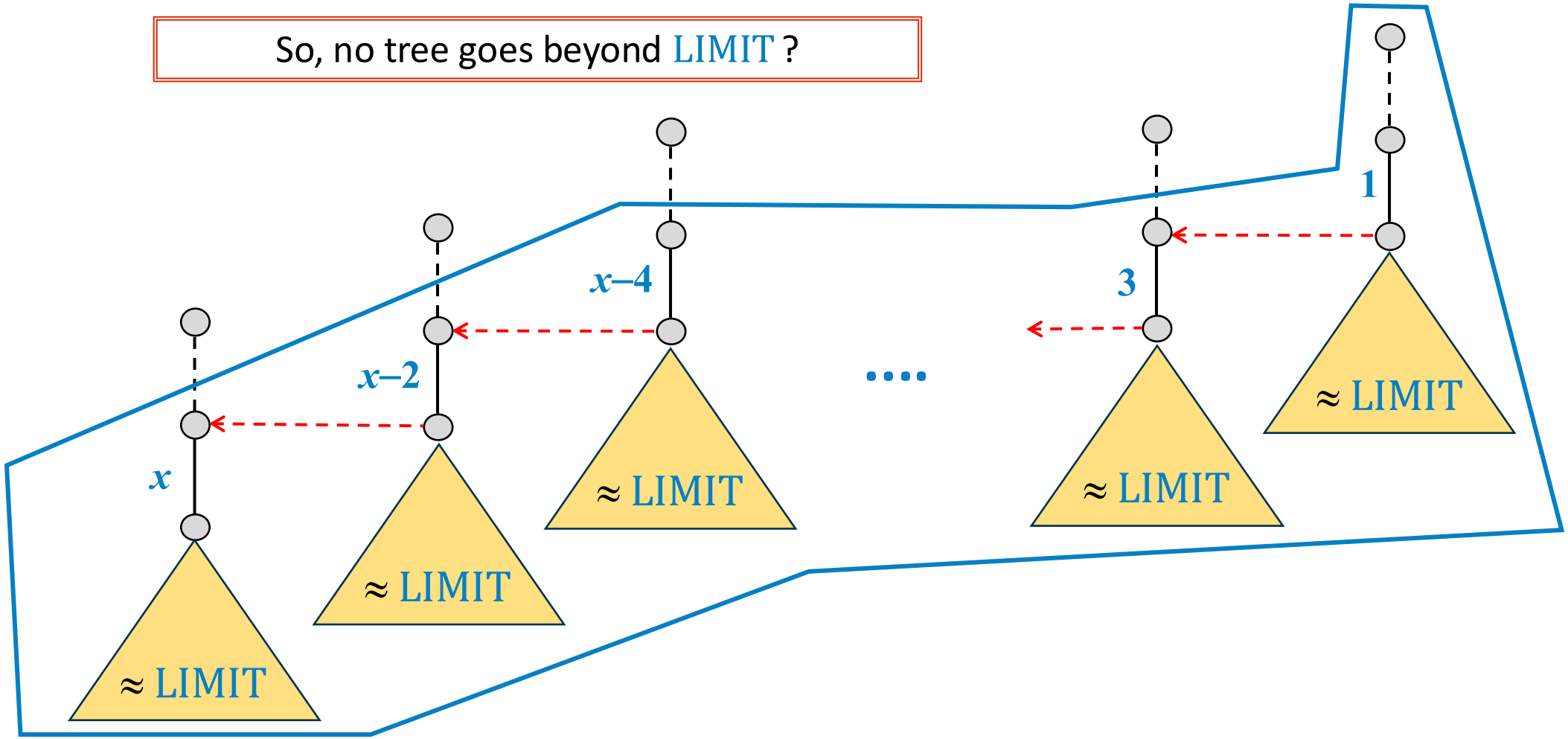


$\approx$ 3*LIMIT

A tree cannot "grow" beyond poly(1/ε)

Idea: A tree gets frozen when its size reaches $\text{LIMIT} = 1/\varepsilon^2$.

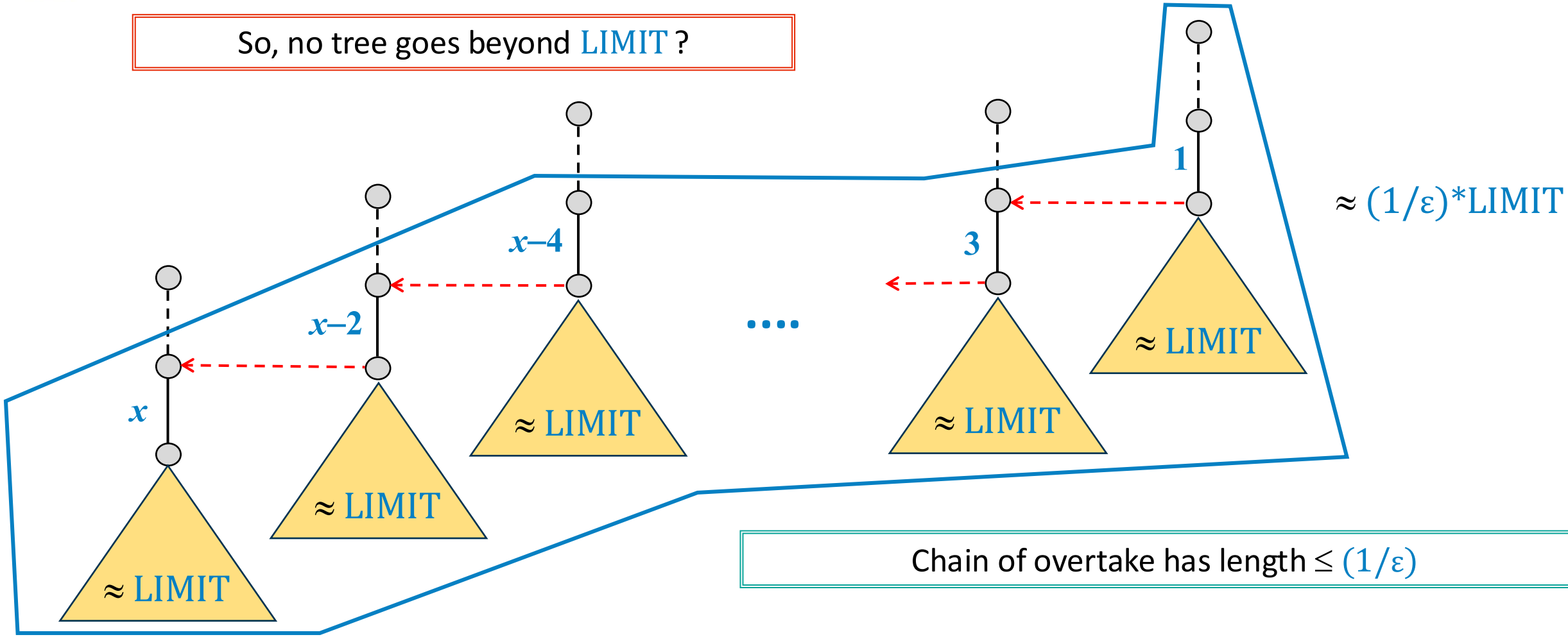
So, no tree goes beyond LIMIT ?



A tree cannot "grow" beyond poly(1/ε)

Idea: A tree gets frozen when its size reaches $\text{LIMIT} = 1/\varepsilon^2$.

So, no tree goes beyond LIMIT ?



Chain of overtake has length ≤ $(1/\varepsilon)$

Open questions

Improving the poly-dependence on $1/\varepsilon$

Further simplification of the current approach

b-matching in $\mathrm{poly}(1/\varepsilon)$ passes in general graphs?

Thank You!