Fast Algorithms for Finding a Maximum Matching
—Centralized and Distributed—

Yutaro Yamaguchi (Osaka University)

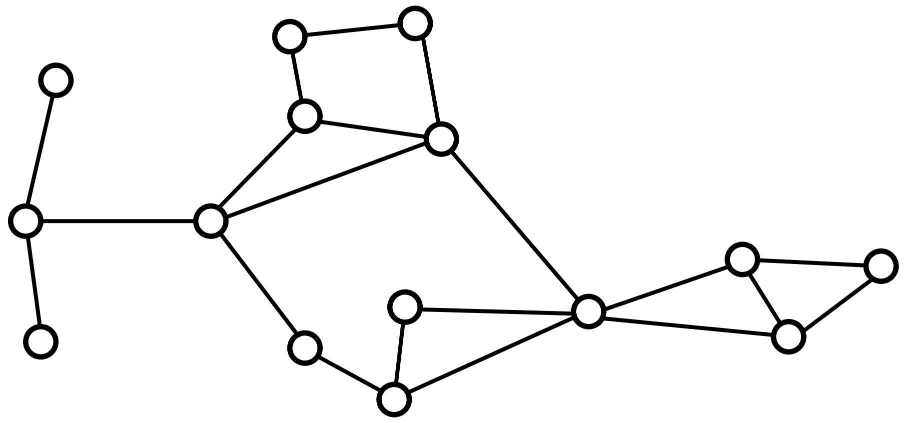
WEPA 2024 2024/10/22

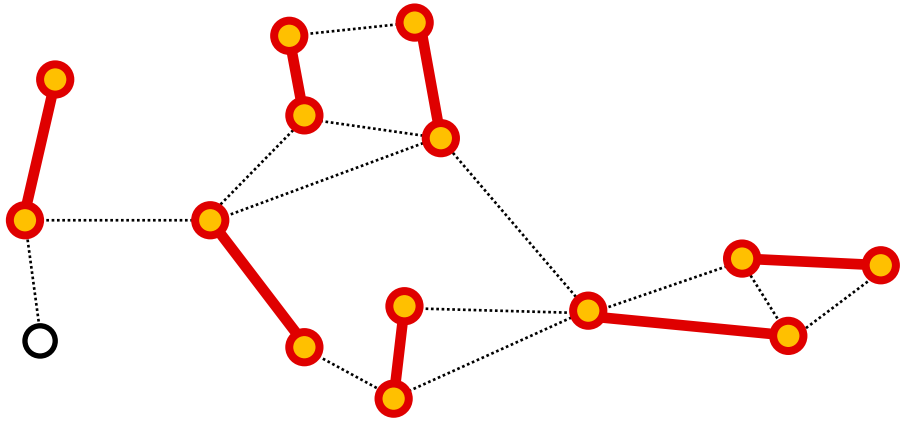
Maximum Matching Problem

Input: $G = (V, E)$: Undirected Graph

Goal: Find a **Matching** $M \subseteq E$ of maximum cardinality

A set of vertex-disjoint edges





Outline

- Basics: Augmenting Paths and Algorithm Framework [Kőnig 1931; Edmonds 1965]
- $O(\sqrt{n}m)$-time Algorithm (Centralized)
  - Update with Maximal Disjoint Shortest Augmenting Paths [Hopcroft–Karp 1973]
  - BFS-honesty of Shortest Alternating Paths: Bipartite vs. General
  - Overview of $O(\sqrt{n}m)$-time Algorithm in General [Micali–Vazirani 1980; Vazirani 2024]
- $O(n \log n)$-round Algorithm under CONGEST Model (Distributed)
  - $O(n)$-round Matching Verification Algorithm [Ahmadi–Kuhn 2020]
  - $O(n^{1.5})$-round Algorithm (Augmenting Path in $O(n)$ rounds) [Kitamura–Izumi 2022]
  - $O(n \log n)$-round Algorithm (Augmenting Path of Length ℓ in $O(\ell)$ rounds) [Izumi–Kitamura–Y. 2024]

Outline

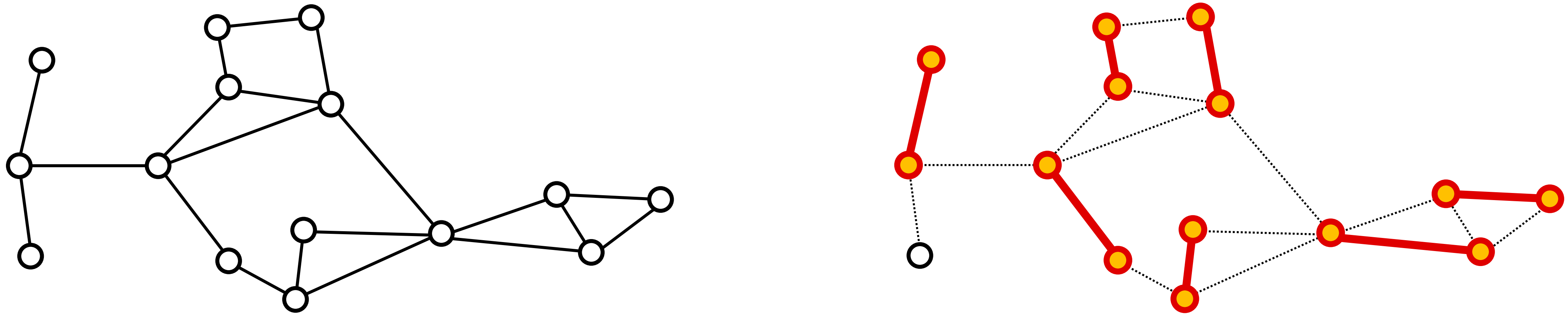
- Basics: Augmenting Paths and Algorithm Framework [Kőnig 1931; Edmonds 1965]
- $\mathrm{O}(\sqrt{n}m)$-time Algorithm (Centralized)
  - Update with Maximal Disjoint Shortest Augmenting Paths [Hopcroft–Karp 1973]
  - BFS-honesty of Shortest Alternating Paths: Bipartite vs. General
  - Overview of $\mathrm{O}(\sqrt{n}m)$-time Algorithm in General [Micali–Vazirani 1980; Vazirani 2024]
- $\mathrm{O}(n \log n)$-round Algorithm under CONGEST Model (Distributed)
  - $\mathrm{O}(n)$-round Matching Verification Algorithm [Ahmadi–Kuhn 2020]
  - $\mathrm{O}(n^{1.5})$-round Algorithm (Augmenting Path in $\mathrm{O}(n)$ rounds) [Kitamura–Izumi 2022]
  - $\mathrm{O}(n \log n)$-round Algorithm (Augmenting Path of Length ℓ in $\mathrm{O}(\ell)$ rounds) [Izumi–Kitamura–Y. 2024]

Maximum Matching Problem

Input: $G = (V, E)$: Undirected Graph

Goal: Find a **Matching** $M \subseteq E$ of maximum cardinality

A set of vertex-disjoint edges

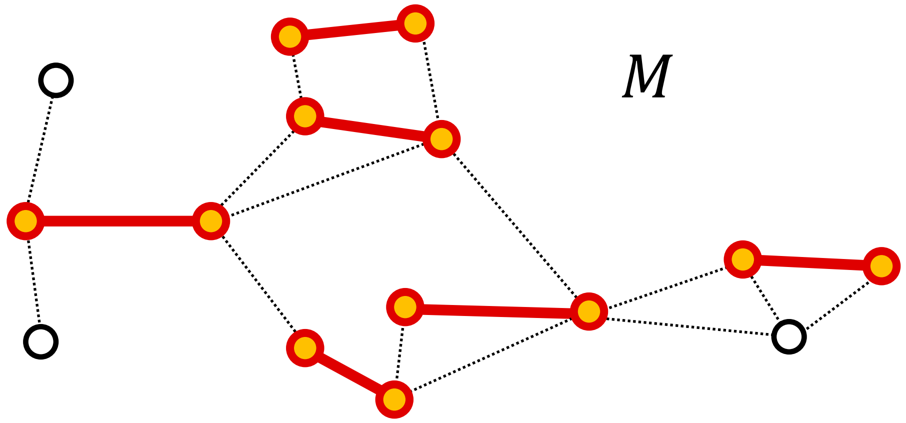


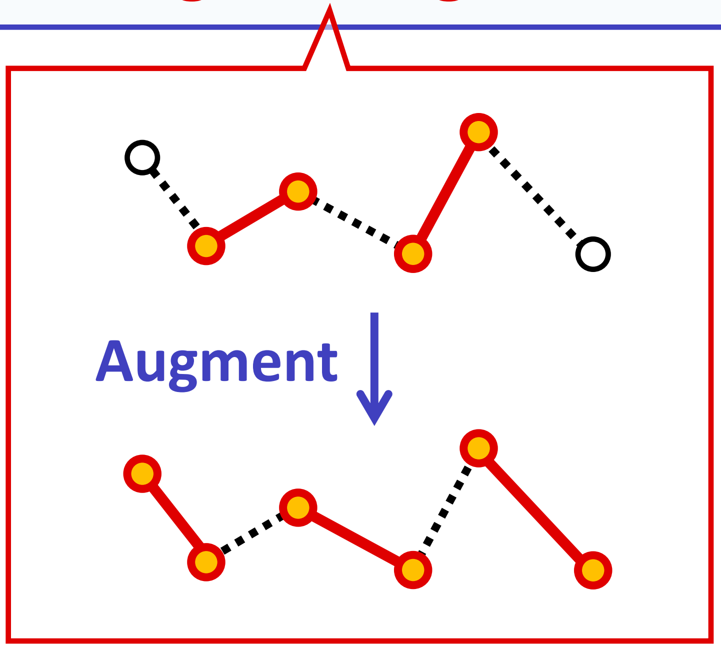
Augmenting Paths

Input: $G = (V, E)$: Undirected Graph

Goal: Find a **Matching** $M \subseteq E$ of maximum cardinality

<u>**Lem.**</u> A matching M is not maximum $\Longleftrightarrow$ $\exists P$: <u>**Augmenting Path**</u> w.r.t. M



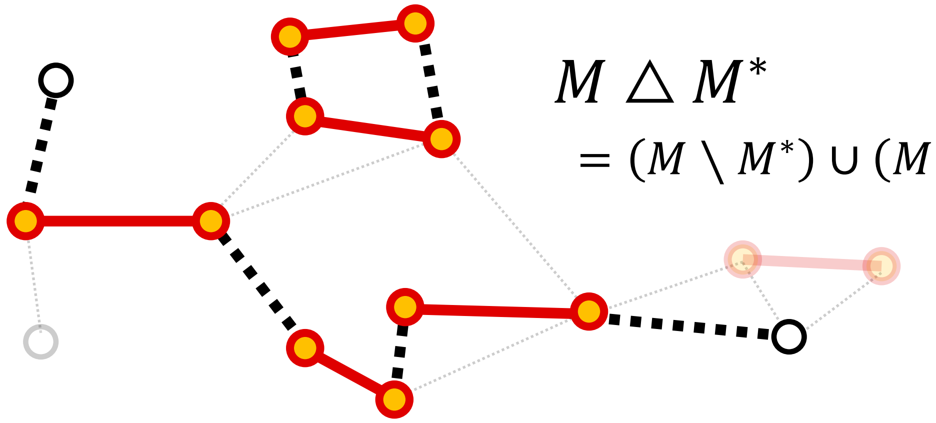


Augmenting Paths

Input: $G = (V, E)$: Undirected Graph

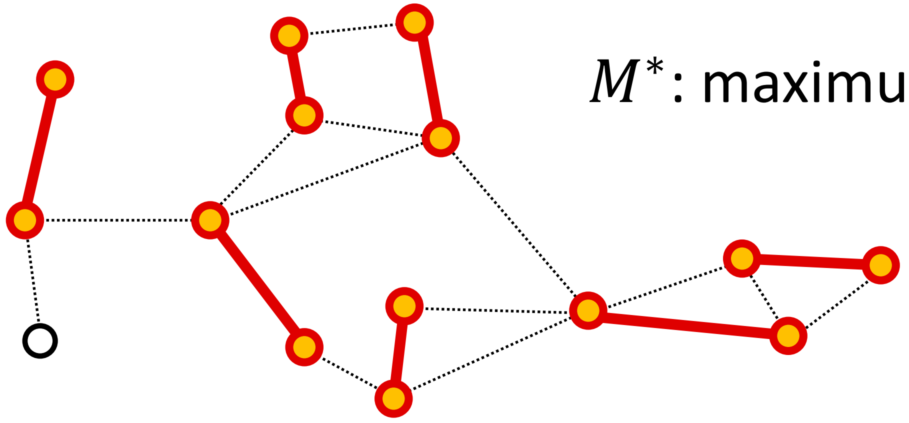
Goal: Find a **Matching** $M \subseteq E$ of maximum cardinality

<u>Lem.</u> A matching M is not maximum $\iff$ $\exists P$: **Augmenting Path** w.r.t. M



$M \triangle M^*$

$= (M \setminus M^*) \cup (M^* \setminus M)$



M^*: maximum

Augmenting Paths

Input: $G = (V, E)$: Undirected Graph
Goal: Find a **Matching** $M \subseteq E$ of maximum cardinality

$n = |V|,\ m = |E|$
μ: optimal value

<u>**Lem.**</u> A matching M is not maximum $\Longleftrightarrow$ $\exists P$: **Augmenting Path** w.r.t. M

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}\big(\mu \cdot \mathrm{FP}(n, m)\big)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)

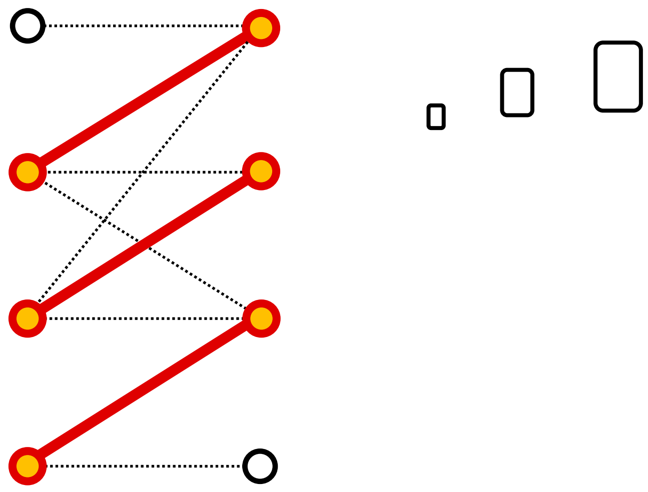
Augmenting Path Algorithm for Bipartite Matching

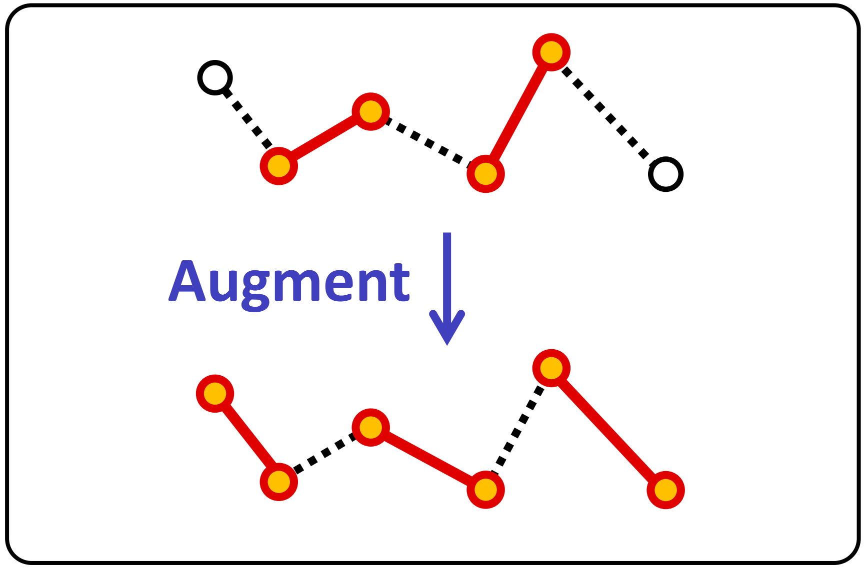
[Kőnig 1931]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}\big(\mu \cdot \mathrm{FP}(n, m)\big)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)





Augmenting Path Algorithm for Bipartite Matching

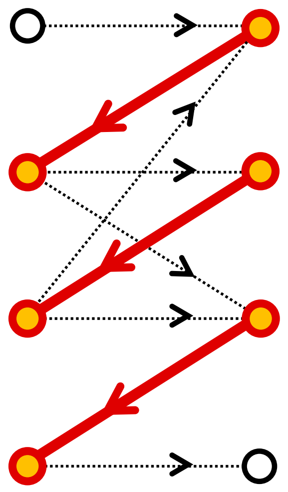
[Kőnig 1931]

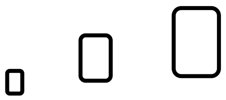
[Naive Algorithm]

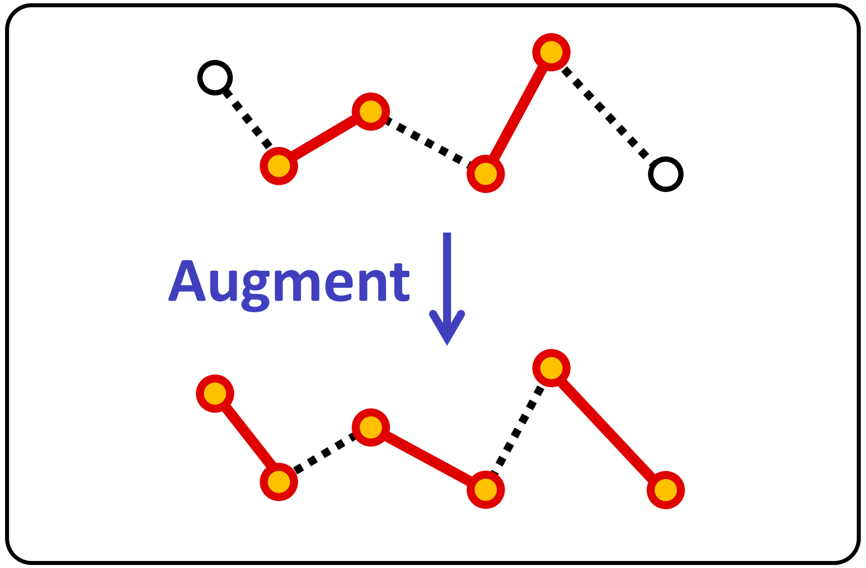
1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}(\mu \cdot \mathrm{FP}(n, m))$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)

Orientation







Augmenting Path Algorithm for Bipartite Matching

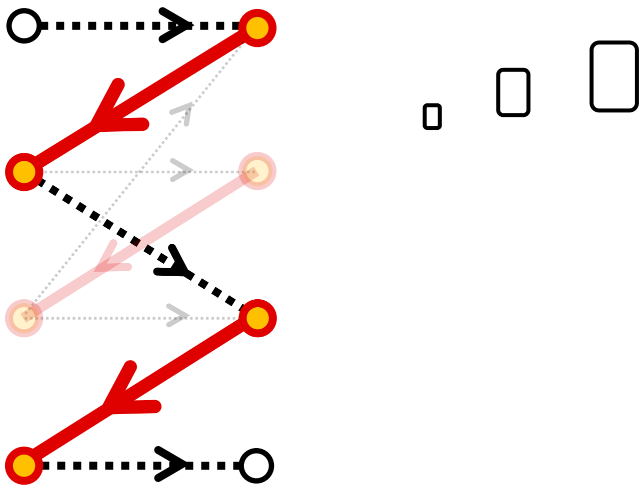
[Kőnig 1931]

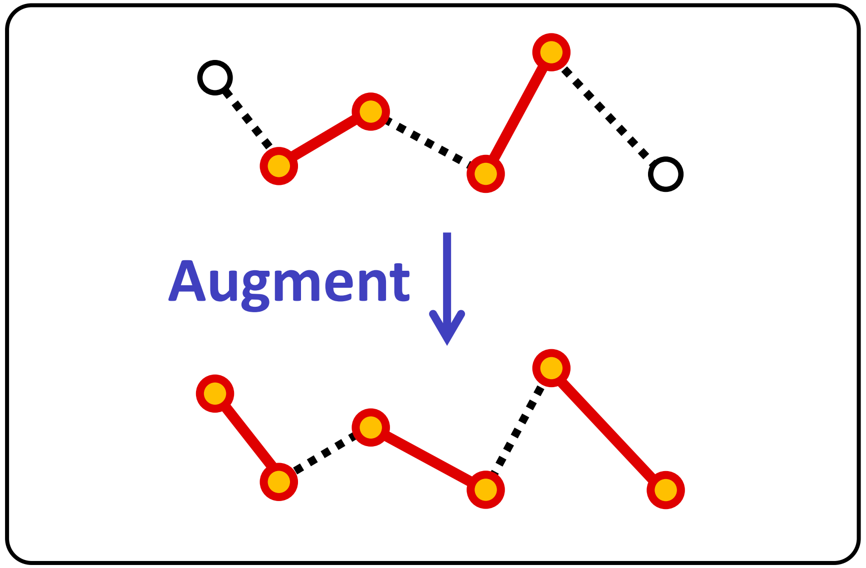
[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}\big(\mu \cdot \mathrm{FP}(n, m)\big)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)

Reachability





Augmenting Path Algorithm for Bipartite Matching

[Kőnig 1931]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}\big(\mu \cdot \mathrm{FP}(n, m)\big)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)

Orientation + BFS (DFS) is enough to find an augmenting path.

$\mathrm{FP}(n, m) = \mathrm{O}(m) \;\; \rightarrow \;\; \mathrm{O}(nm)$ time in total

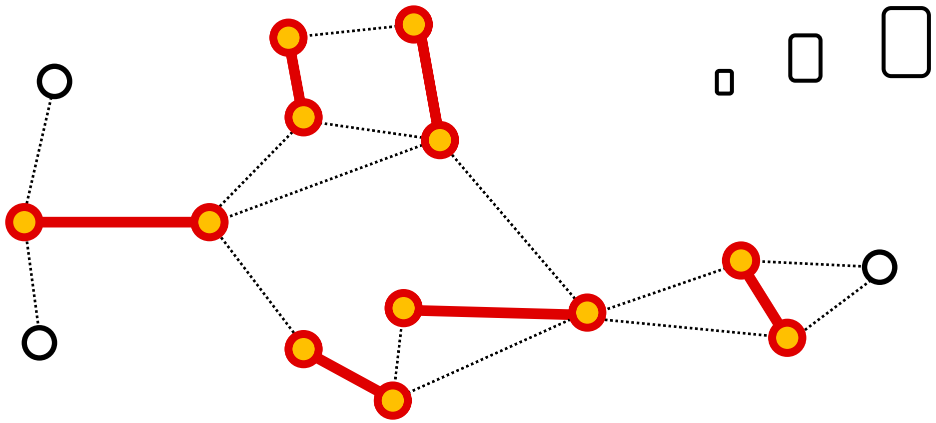
Edmonds' Blossom Algorithm for General Matching

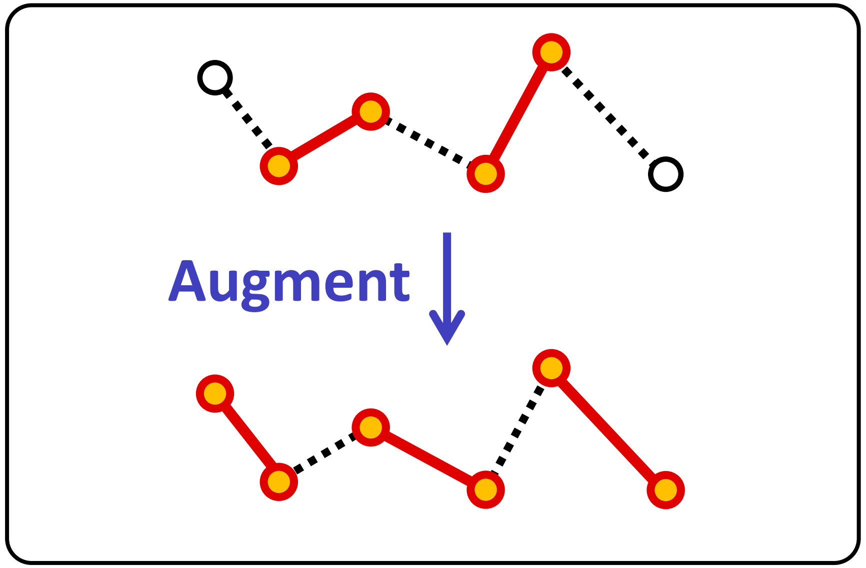
[Edmonds 1965]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}(\mu \cdot \mathrm{FP}(n,m))$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)





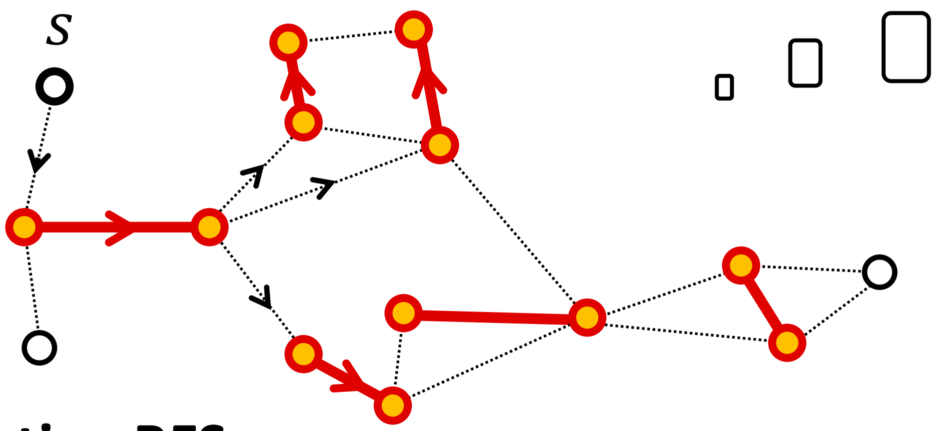
Edmonds' Blossom Algorithm for General Matching

[Edmonds 1965]

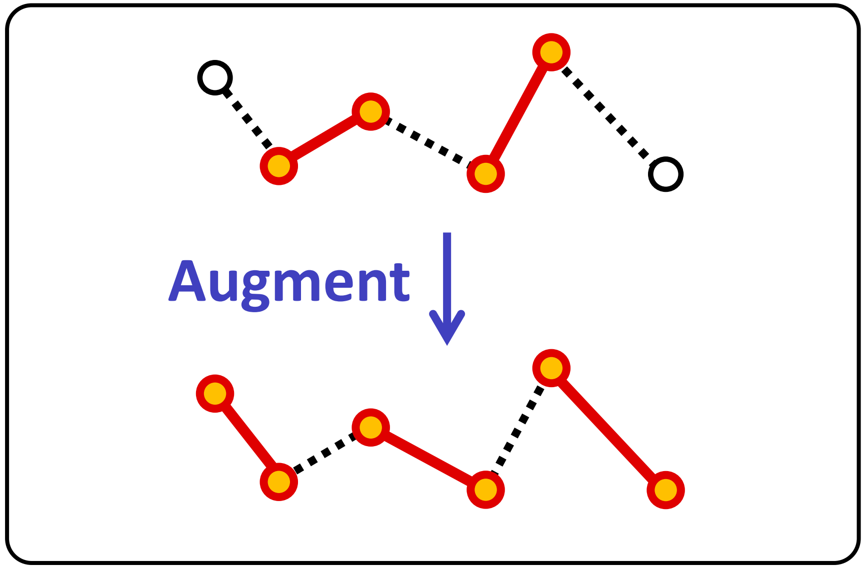
[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}\big(\mu \cdot \mathrm{FP}(n, m)\big)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)



Alternating BFS



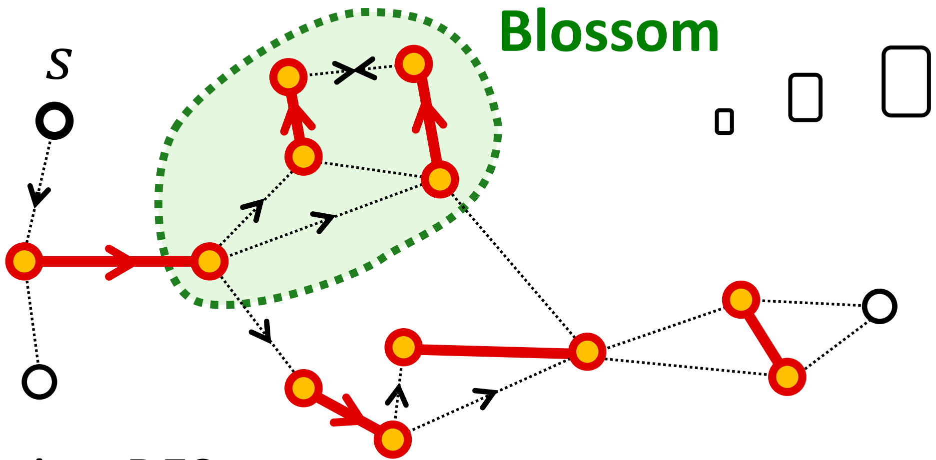
Edmonds' Blossom Algorithm for General Matching

[Edmonds 1965]

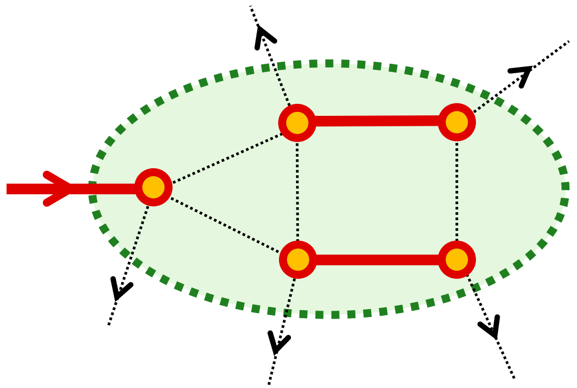
[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}\big(\mu \cdot \mathrm{FP}(n, m)\big)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)



Alternating BFS



We can leave from any vertex

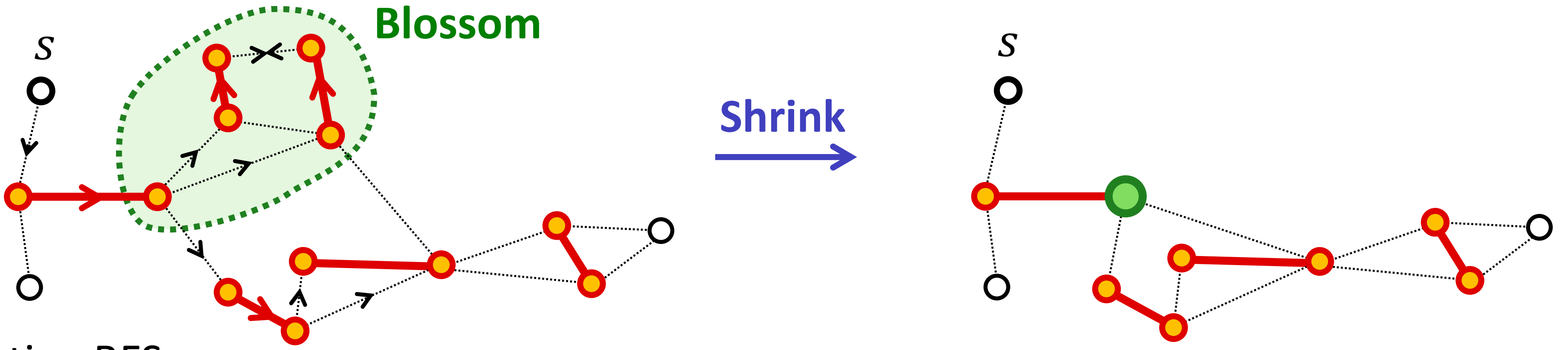
Edmonds' Blossom Algorithm for General Matching

[Edmonds 1965]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}\big(\mu \cdot \mathrm{FP}(n, m)\big)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)



Alternating BFS

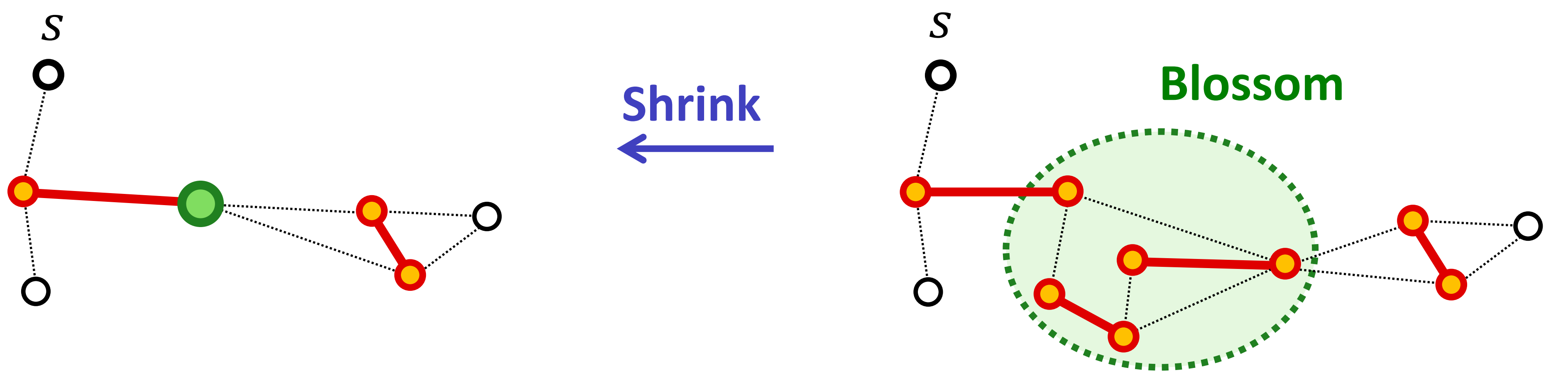
Edmonds' Blossom Algorithm for General Matching

[Edmonds 1965]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}\big(\mu \cdot \mathrm{FP}(n, m)\big)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)



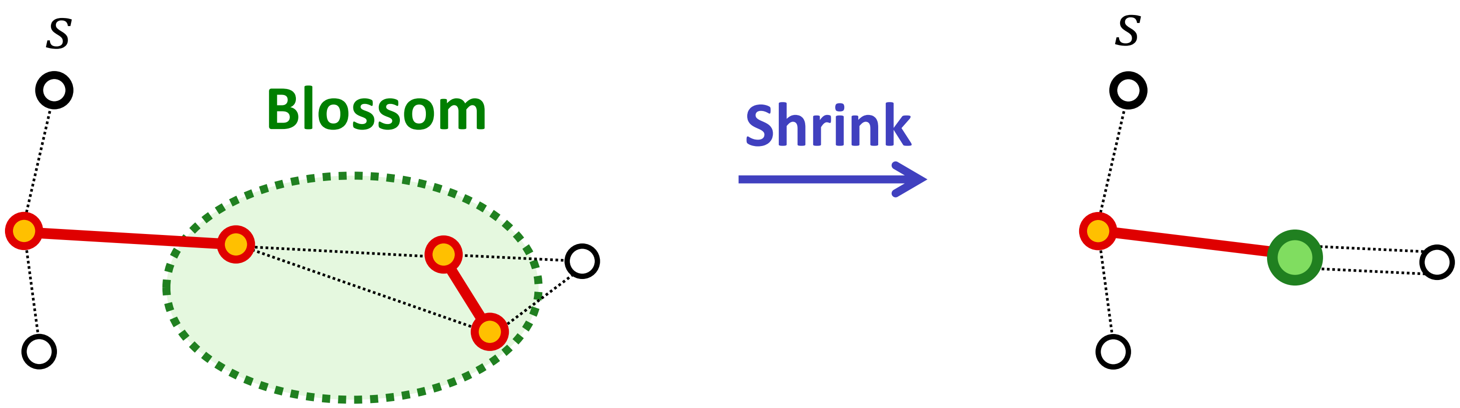
Edmonds' Blossom Algorithm for General Matching

[Edmonds 1965]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}\big(\mu \cdot \mathrm{FP}(n, m)\big)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)



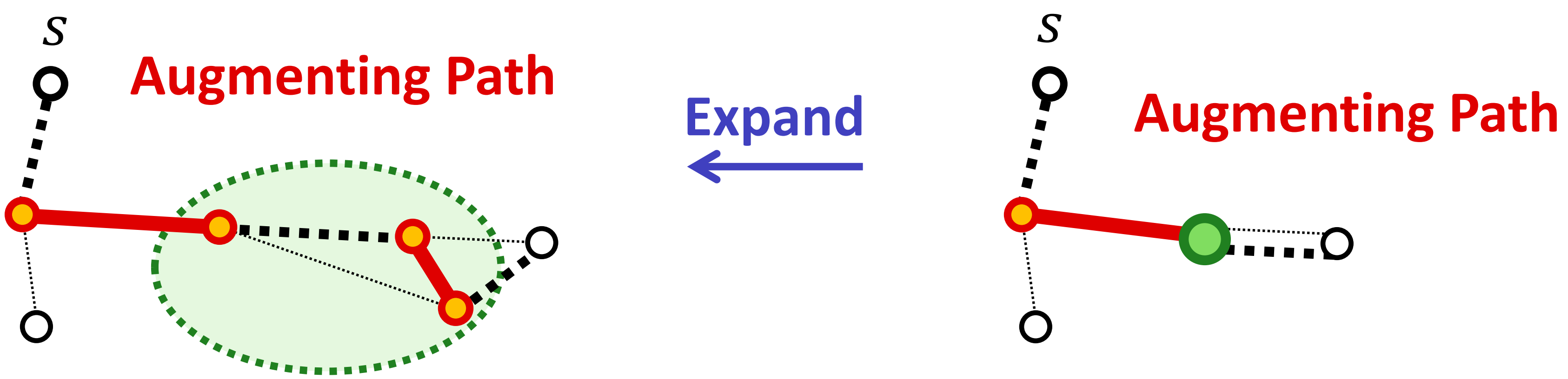
Edmonds' Blossom Algorithm for General Matching

[Edmonds 1965]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}\big(\mu \cdot \mathrm{FP}(n, m)\big)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)



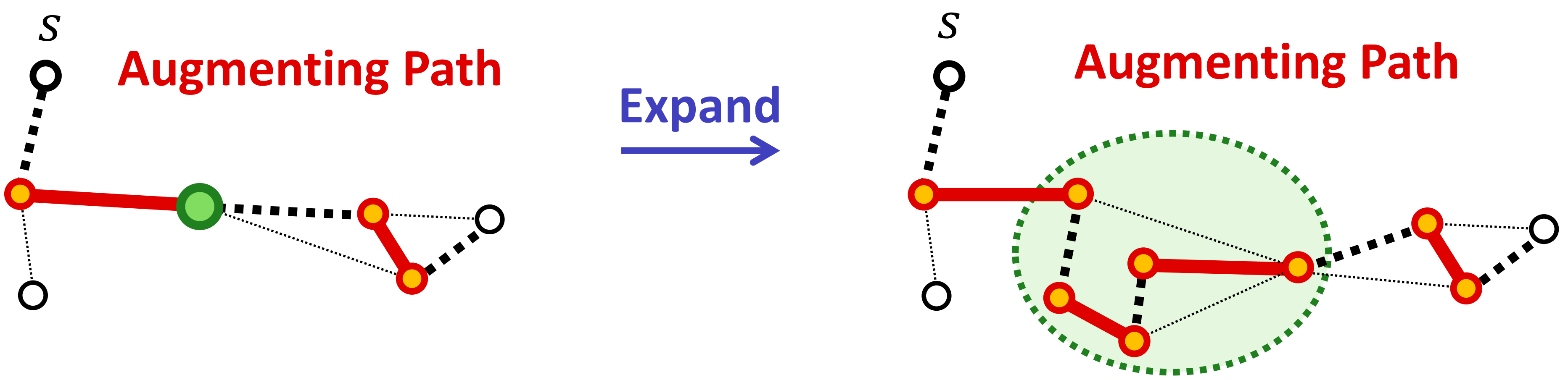
Edmonds' Blossom Algorithm for General Matching

[Edmonds 1965]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$O\big(\mu \cdot \mathrm{FP}(n, m)\big)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)



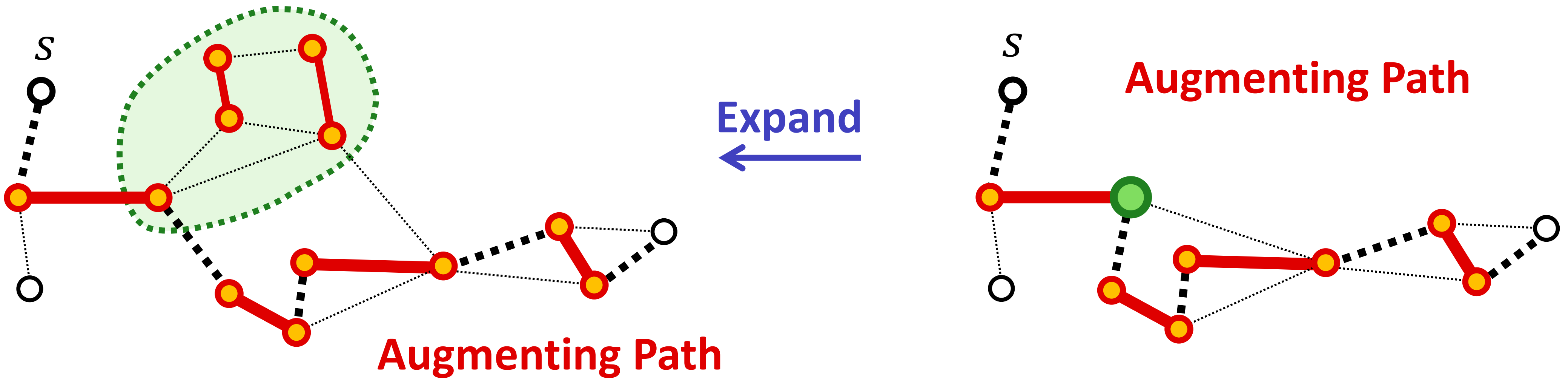
Edmonds' Blossom Algorithm for General Matching

[Edmonds 1965]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}(\mu \cdot \mathrm{FP}(n, m))$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)



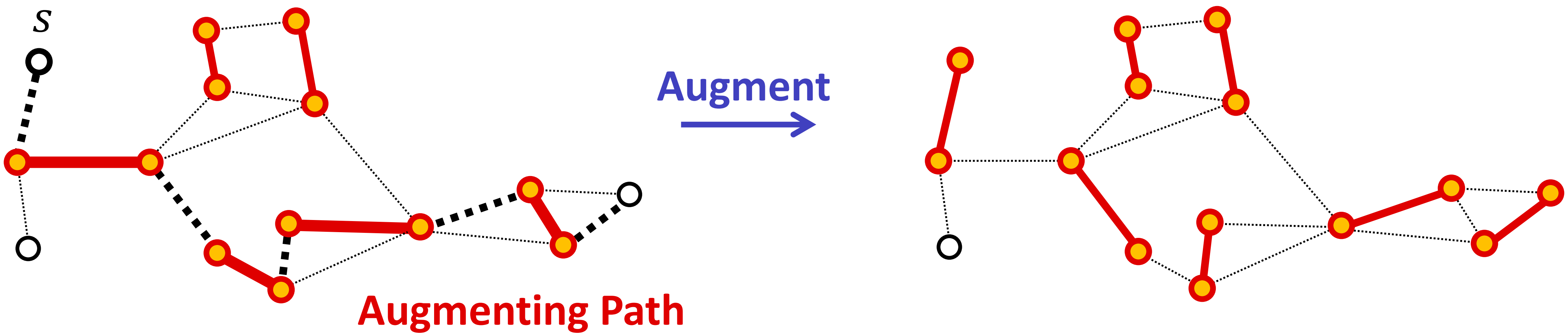
Edmonds' Blossom Algorithm for General Matching

[Edmonds 1965]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}\big(\mu \cdot \mathrm{FP}(n, m)\big)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)



Edmonds' Blossom Algorithm for General Matching

[Edmonds 1965]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}\big(\mu \cdot \mathrm{FP}(n, m)\big)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)

- $\#(\text{Shrink per Augment}) \leq \frac{n}{2}$
- Shrink, Expand, and BFS are done in $\mathrm{O}(m)$ time

$\mathrm{FP}(n, m) = \mathrm{O}(nm) \ \rightarrow \ \mathrm{O}(n^2 m)$ time in total

Outline

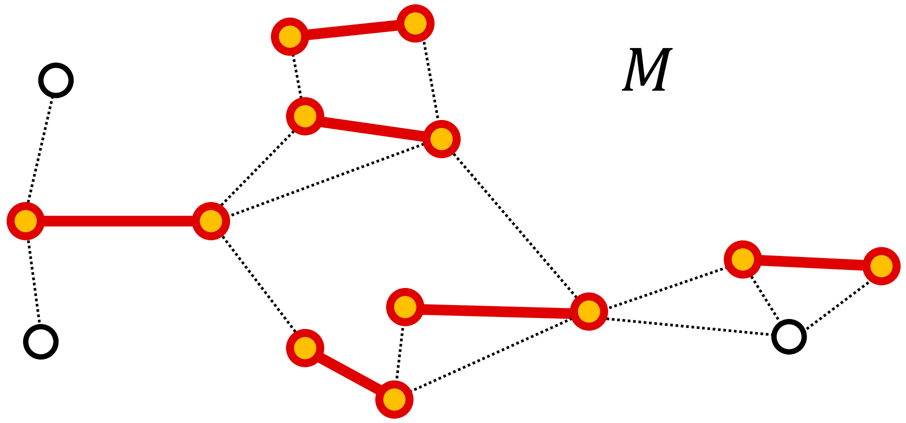
- Basics: Augmenting Paths and Algorithm Framework [Kőnig 1931; Edmonds 1965]
- $O(\sqrt{n}m)$-time Algorithm (Centralized)
  - Update with Maximal Disjoint Shortest Augmenting Paths [Hopcroft–Karp 1973]
  - BFS-honesty of Shortest Alternating Paths: Bipartite vs. General
  - Overview of $O(\sqrt{n}m)$-time Algorithm in General [Micali–Vazirani 1980; Vazirani 2024]
- $O(n \log n)$-round Algorithm under CONGEST Model (Distributed)
  - $O(n)$-round Matching Verification Algorithm [Ahmadi–Kuhn 2020]
  - $O(n^{1.5})$-round Algorithm (Augmenting Path in $O(n)$ rounds) [Kitamura–Izumi 2022]
  - $O(n \log n)$-round Algorithm (Augmenting Path of Length ℓ in $O(\ell)$ rounds) [Izumi–Kitamura–Y. 2024]

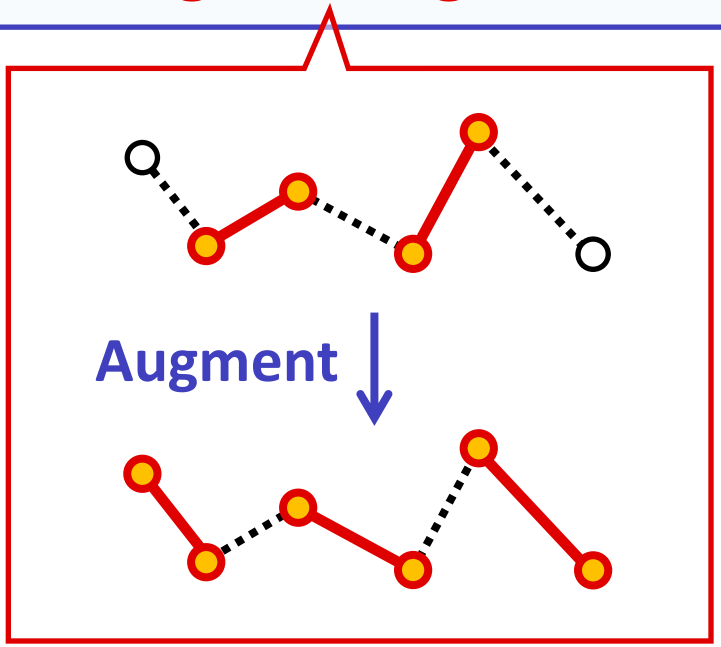
Augmenting Paths

Input: $G = (V, E)$: Undirected Graph

Goal: Find a **Matching** $M \subseteq E$ of maximum cardinality

Lem. A matching M is not maximum $\iff$ $\exists P$: **Augmenting Path** w.r.t. M





Length of Shortest Augmenting Paths

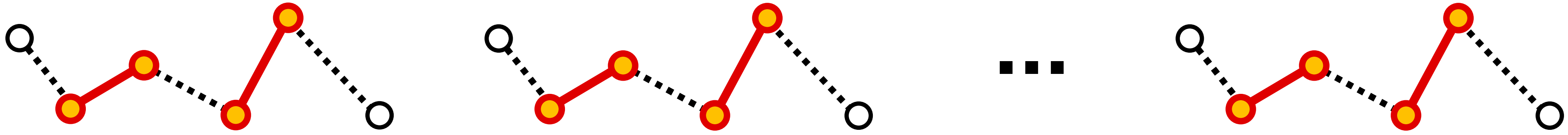
Input: $G = (V, E)$: Undirected Graph
Goal: Find a **Matching** $M \subseteq E$ of maximum cardinality

$n = |V|,\ m = |E|$
μ: optimal value

Lem. A matching M is of cardinality $\mu - k\ \ (1 \leq k \leq \mu)$
$\Longrightarrow\ \exists P$: augmenting path w.r.t. M of length less than $\frac{2\mu}{k}$

[Hopcroft–Karp 1973]

In the worst case, $M \triangle M^*$ forms k disjoint augmenting paths of the same length.

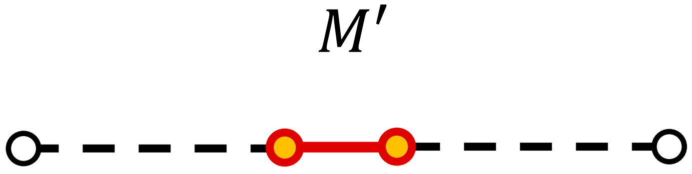


Update with Maximal Shortest Augmenting Paths

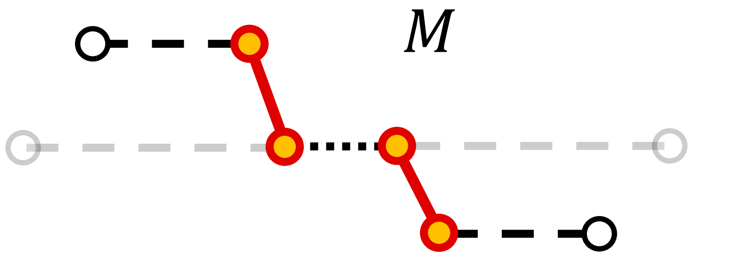
<u>Lem.</u> M: matching, ℓ: length of a **shortest** augmenting path w.r.t. M
$M \triangle M'$ forms **maximal** disjoint augmenting paths of length ℓ
$\Longrightarrow$ M' has no augmenting path of length at most ℓ

[Hopcroft–Karp 1973]

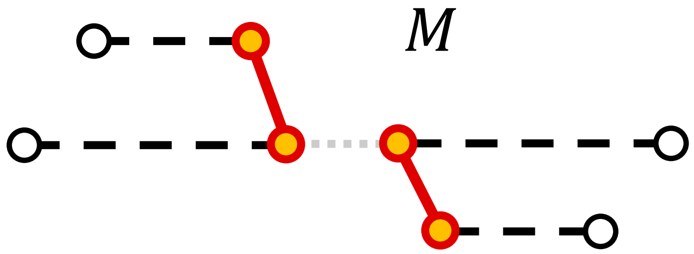
If some remains, a contradiction is obtained, e.g., as follows. (Informal)



at most ℓ



$\exists$augmenting path in $M \triangle M'$
of length exactly ℓ



either is shorter than ℓ

Fast Algorithm for Maximum Matching Problem

[Fast Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M,
   find maximal disjoint shortest ones and update M with them

Fast Algorithm for Maximum Matching Problem

[Fast Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M,
   find maximal disjoint shortest ones and update M with them

- The length of a shortest augmenting path monotonically increases

Lem. M: matching, ℓ: length of a **shortest** augmenting path w.r.t. M
$M \triangle M'$ forms **maximal** disjoint augmenting paths of length ℓ
$\Longrightarrow$ M' has no augmenting path of length at most ℓ

[Hopcroft–Karp 1973]

Fast Algorithm for Maximum Matching Problem

[Fast Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M,
   find maximal disjoint shortest ones and update M with them

- The length of a shortest augmenting path monotonically increases

  $\rightarrow$ $\#\left(\text{iterations with } |M| \leq \mu - \sqrt{\mu}\right) \leq \sqrt{\mu}$

Lem. A matching M is of cardinality $\mu - k \ (1 \leq k \leq \mu)$

$\Longrightarrow$ $\exists P$: augmenting path w.r.t. M of length less than $\frac{2\mu}{k}$

[Hopcroft–Karp 1973]

Fast Algorithm for Maximum Matching Problem

[Fast Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M,
   find maximal disjoint shortest ones and update M with them

- The length of a shortest augmenting path monotonically increases
  $\rightarrow$ $\#\left(\text{iterations with } |M| \leq \mu - \sqrt{\mu}\right) \leq \sqrt{\mu}$
- Clearly, $\#\left(\text{iterations with } |M| \geq \mu - \sqrt{\mu}\right) \leq \sqrt{\mu}$

$\mathrm{O}\left(\sqrt{\mu} \cdot \mathrm{FMDSP}(n, m)\right)$ time in total [Hopcroft–Karp 1973]

($\mathrm{FMDSP}(\cdot)$: time to find maximal disjoint shortest augmenting paths)

Maximum Matching in $\mathrm{O}(\sqrt{n}m)$ Time

$\mathrm{O}\left(\sqrt{\mu} \cdot \mathrm{FMDSP}(n, m)\right)$ via maximal disjoint shortest augmenting paths

($\mathrm{FMDSP}(\cdot)$: time to find maximal disjoint shortest augmenting paths)

$\rightarrow$ $\mathrm{FMDSP}(n, m) = \mathrm{O}(m)$ is sufficient

- When G is bipartite, it is easy (Orientation + DFS on the DAG after BFS)
  [Hopcroft–Karp 1973]
- When G is not bipartite, it is not so easy but possible
  [Micali–Vazirani 1980; Vazirani 2024]

Q. What is the essential difference?

A. **BFS-honesty** of **Shortest Alternating Paths**
(Intuitively, any prefix of a shortest path should be a shortest path.)

Maximum Matching in $\mathrm{O}(\sqrt{n}m)$ Time

$\mathrm{O}\left(\sqrt{\mu} \cdot \mathrm{FMDSP}(n, m)\right)$ via maximal disjoint shortest augmenting paths

($\mathrm{FMDSP}(\cdot)$: time to find maximal disjoint shortest augmenting paths)

$\rightarrow$ $\mathrm{FMDSP}(n, m) = \mathrm{O}(m)$ is sufficient

- When G is bipartite, it is easy (Orientation + DFS on the DAG after BFS)
  [Hopcroft–Karp 1973]
- When G is not bipartite, it is not so easy but possible
  [Micali–Vazirani 1980; Vazirani 2024]

Q. What is the essential difference?

A. **BFS-honesty** of **Shortest Alternating Paths**

(Intuitively, any prefix of a shortest path should be a shortest path.)

Maximum Matching in $\mathrm{O}(\sqrt{n}m)$ Time

$\mathrm{O}\left(\sqrt{\mu}\cdot \mathrm{FMDSP}(n,m)\right)$ via maximal disjoint shortest augmenting paths

($\mathrm{FMDSP}(\cdot)$: time to find maximal disjoint shortest augmenting paths)

$\rightarrow$ $\mathrm{FMDSP}(n,m) = \mathrm{O}(m)$ is sufficient

- When G is bipartite, it is easy (Orientation + DFS on the DAG after BFS)
  [Hopcroft–Karp 1973]
- When G is not bipartite, it is not so easy but possible
  [Micali–Vazirani 1980; Vazirani 2024]

Q. What is the essential difference?

A. **BFS-honesty** of **Shortest Alternating Paths**
(Intuitively, any prefix of a shortest path should be a shortest path.)

Outline

- Basics: Augmenting Paths and Algorithm Framework [Kőnig 1931; Edmonds 1965]
- $O(\sqrt{n}m)$-time Algorithm (Centralized)
  - Update with Maximal Disjoint Shortest Augmenting Paths [Hopcroft–Karp 1973]
  - BFS-honesty of Shortest Alternating Paths: Bipartite vs. General
  - Overview of $O(\sqrt{n}m)$-time Algorithm in General [Micali–Vazirani 1980; Vazirani 2024]
- $O(n \log n)$-round Algorithm under CONGEST Model (Distributed)
  - $O(n)$-round Matching Verification Algorithm [Ahmadi–Kuhn 2020]
  - $O(n^{1.5})$-round Algorithm (Augmenting Path in $O(n)$ rounds) [Kitamura–Izumi 2022]
  - $O(n \log n)$-round Algorithm (Augmenting Path of Length ℓ in $O(\ell)$ rounds) [Izumi–Kitamura–Y. 2024]

BFS-honesty of Shortest Alternating Paths

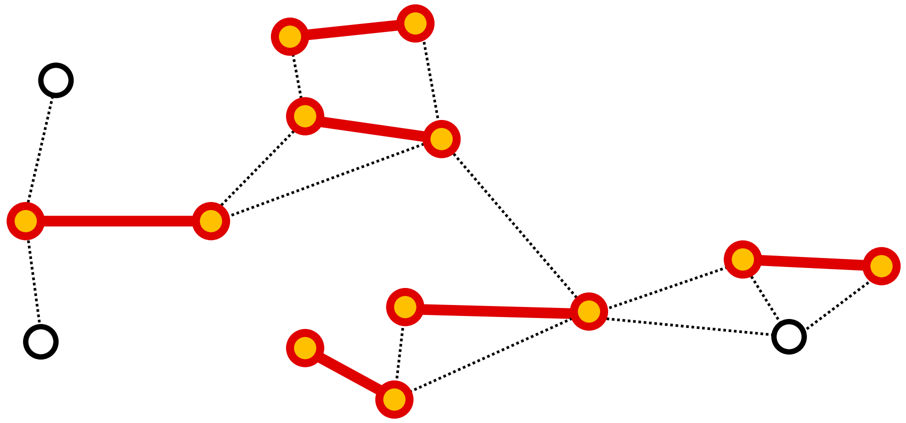
For each vertex $v \in V$, define the followings.

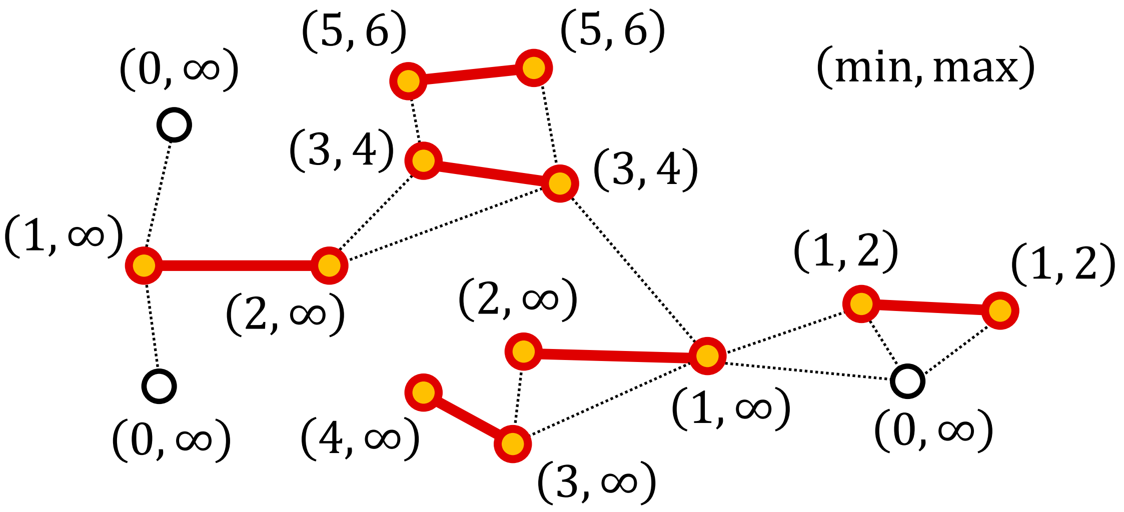
$\mathrm{oddlevel}(v)$: the length of a shortest **odd** alternating path from an unmatched vertex

$\mathrm{evenlevel}(v)$: the length of a shortest **even** alternating path from an unmatched vertex

$\mathrm{minlevel}(v) = \min\{\mathrm{oddlevel}(v), \mathrm{evenlevel}(v)\}$

$\mathrm{maxlevel}(v) = \max\{\mathrm{oddlevel}(v), \mathrm{evenlevel}(v)\}$





BFS-honesty of Shortest Alternating Paths

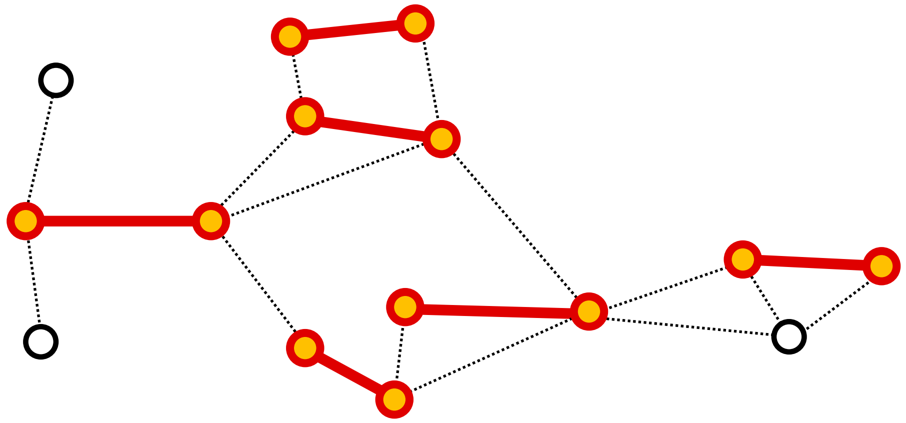
For each vertex $v \in V$, define the followings.

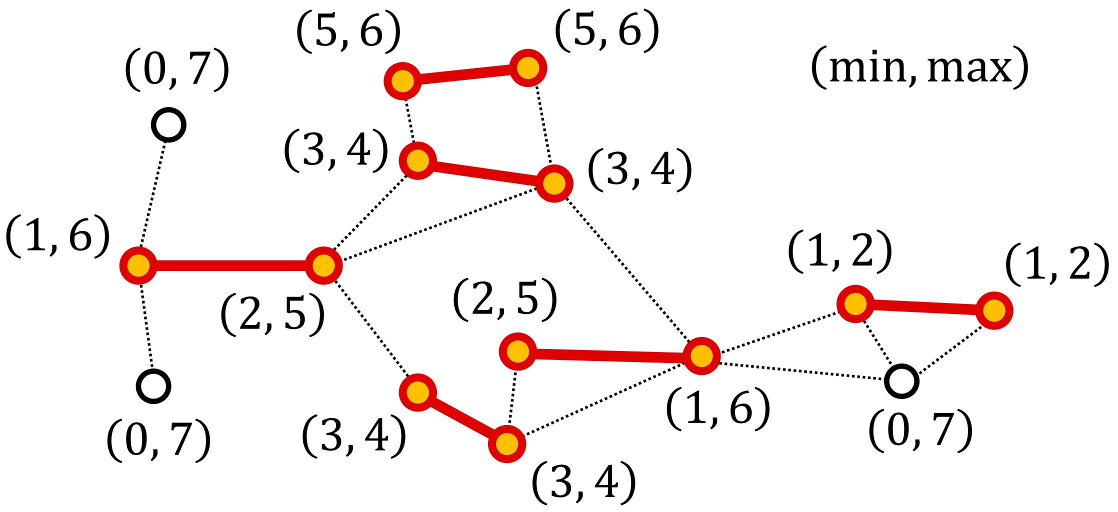
$\mathrm{oddlevel}(v)$: the length of a shortest **odd** alternating path from an unmatched vertex

$\mathrm{evenlevel}(v)$: the length of a shortest **even** alternating path from an unmatched vertex

$\mathrm{minlevel}(v) = \min\{\mathrm{oddlevel}(v), \mathrm{evenlevel}(v)\}$

$\mathrm{maxlevel}(v) = \max\{\mathrm{oddlevel}(v), \mathrm{evenlevel}(v)\}$





BFS-honesty of Shortest Alternating Paths

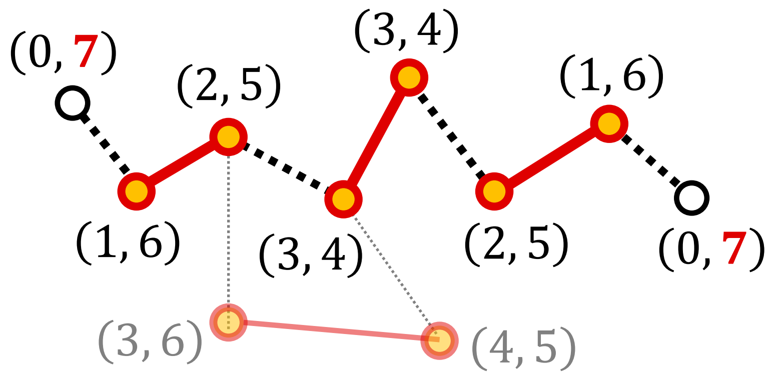
For each vertex $v \in V$, define the followings.

$\mathrm{oddlevel}(v)$: the length of a shortest **odd** alternating path from an unmatched vertex

$\mathrm{evenlevel}(v)$: the length of a shortest **even** alternating path from an unmatched vertex

$\mathrm{minlevel}(v) = \min\{\mathrm{oddlevel}(v), \mathrm{evenlevel}(v)\}$

$\mathrm{maxlevel}(v) = \max\{\mathrm{oddlevel}(v), \mathrm{evenlevel}(v)\}$



- $\ell = \min\{\,\mathrm{oddlevel}(v) \mid v \text{ is unmatched}\,\}$
- G is bipartite

  $\Longrightarrow$ any prefix of any odd/evenlevel path attains odd/evenlevel (BFS-honest)
- G is not bipartite

  $\Longrightarrow$ a prefix of an odd/evenlevel path may not attain odd/evenlevel

BFS-honesty of Shortest Alternating Paths

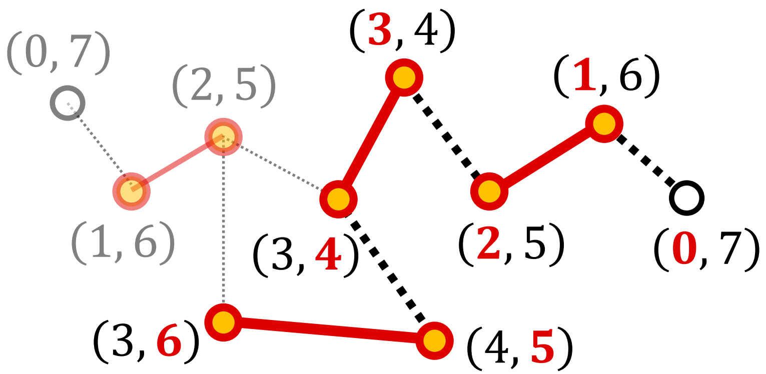
For each vertex $v \in V$, define the followings.

$\mathrm{oddlevel}(v)$: the length of a shortest **odd** alternating path from an unmatched vertex

$\mathrm{evenlevel}(v)$: the length of a shortest **even** alternating path from an unmatched vertex

$\mathrm{minlevel}(v) = \min\{\mathrm{oddlevel}(v), \mathrm{evenlevel}(v)\}$

$\mathrm{maxlevel}(v) = \max\{\mathrm{oddlevel}(v), \mathrm{evenlevel}(v)\}$



- $\ell = \min\{\,\mathrm{oddlevel}(v) \mid v \text{ is unmatched}\,\}$
- G is bipartite
  $\Longrightarrow$ any prefix of any odd/evenlevel path attains odd/evenlevel (BFS-honest)
- G is not bipartite
  $\Longrightarrow$ a prefix of an odd/evenlevel path may not attain odd/evenlevel

BFS-honesty of Shortest Alternating Paths

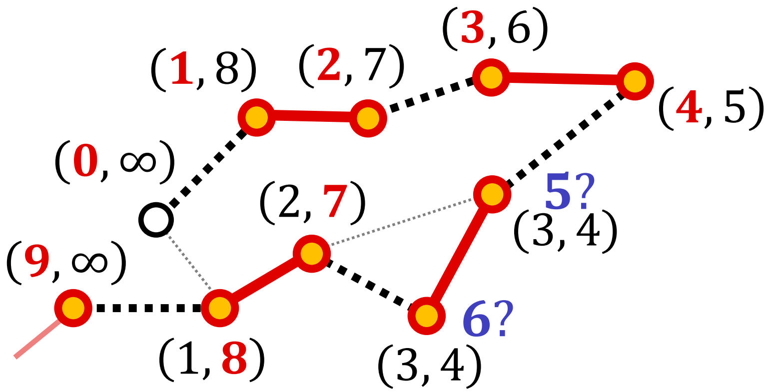
For each vertex $v \in V$, define the followings.

$\mathrm{oddlevel}(v)$: the length of a shortest **odd** alternating path from an unmatched vertex

$\mathrm{evenlevel}(v)$: the length of a shortest **even** alternating path from an unmatched vertex

$\mathrm{minlevel}(v) = \min\{\mathrm{oddlevel}(v), \mathrm{evenlevel}(v)\}$

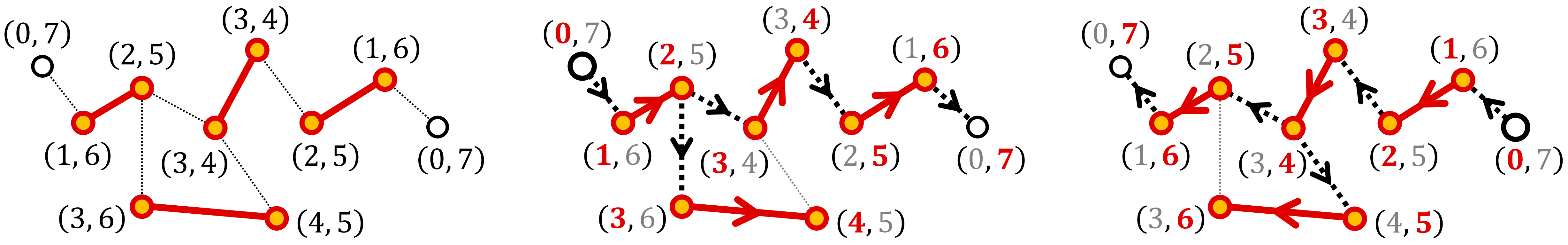
$\mathrm{maxlevel}(v) = \max\{\mathrm{oddlevel}(v), \mathrm{evenlevel}(v)\}$



- $\ell = \min\{\, \mathrm{oddlevel}(v) \mid v \text{ is unmatched} \,\}$
- G is bipartite
  $\Longrightarrow$ any prefix of any odd/evenlevel path attains odd/evenlevel (BFS-honest)
- G is not bipartite
  $\Longrightarrow$ a prefix of an odd/evenlevel path may not attain odd/evenlevel

Computing Odd/Evenlevels

- G is bipartite
  $\Longrightarrow$ any prefix of any odd/evenlevel path attains odd/evenlevel (BFS-honest)
  - All odd/evenlevels are assigned by their neighbors via BFS
  - The parity is determined by the roots of BFS (in which side of bipartition)
  - Easy to find maximal disjoint shortest augmenting paths (by DFS on the DAG after BFS)



Computing Odd/Evenlevels

- G is bipartite
  $\Longrightarrow$ any prefix of any odd/evenlevel path attains odd/evenlevel (BFS-honest)
  - All odd/evenlevels are assigned by their neighbors via BFS
  - The parity is determined by the roots of BFS (in which side of bipartition)
  - Easy to find maximal disjoint shortest augmenting paths (by DFS on the DAG after BFS)

- G is not bipartite
  $\Longrightarrow$ a prefix of an odd/evenlevel path may not attain odd/evenlevel
  - Still, all minlevels are assigned by their neighbors in a BFS-like manner
  - Who does assign maxlevels? $\rightarrow$ **Bridge!**

Computing Odd/Evenlevels

- G is bipartite
  $\Longrightarrow$ any prefix of any odd/evenlevel path attains odd/evenlevel (BFS-honest)
  - All odd/evenlevels are assigned by their neighbors via BFS
  - The parity is determined by the roots of BFS (in which side of bipartition)
  - Easy to find maximal disjoint shortest augmenting paths (by DFS on the DAG after BFS)

- G is not bipartite
  $\Longrightarrow$ a prefix of an odd/evenlevel path may not attain odd/evenlevel
  - Still, all minlevels are assigned by their neighbors in a BFS-like manner
  - Who does assign maxlevels? $\rightarrow$ **Bridge!**

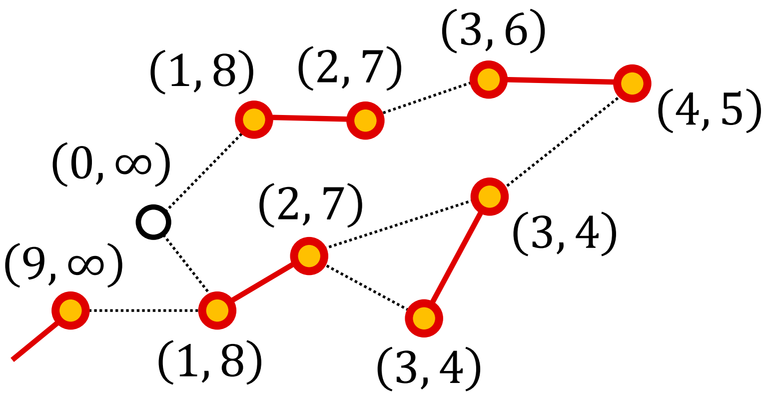
Outline

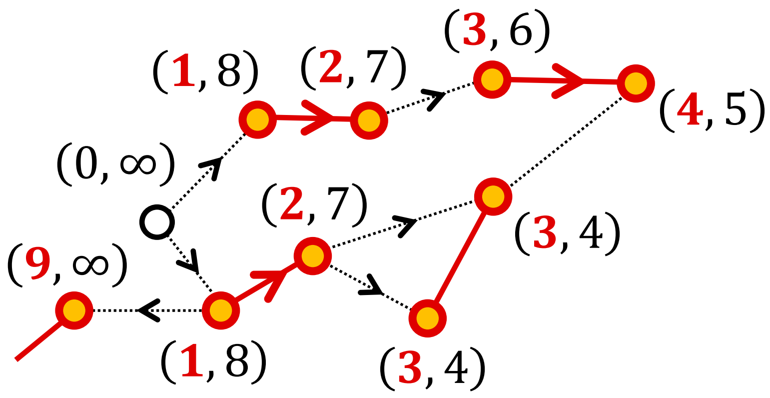
- Basics: Augmenting Paths and Algorithm Framework [Kőnig 1931; Edmonds 1965]
- $\mathrm{O}(\sqrt{n}m)$-time Algorithm (Centralized)
  - Update with Maximal Disjoint Shortest Augmenting Paths [Hopcroft–Karp 1973]
  - BFS-honesty of Shortest Alternating Paths: Bipartite vs. General
  - Overview of $\mathrm{O}(\sqrt{n}m)$-time Algorithm in General [Micali–Vazirani 1980; Vazirani 2024]
- $\mathrm{O}(n \log n)$-round Algorithm under CONGEST Model (Distributed)
  - $\mathrm{O}(n)$-round Matching Verification Algorithm [Ahmadi–Kuhn 2020]
  - $\mathrm{O}(n^{1.5})$-round Algorithm (Augmenting Path in $\mathrm{O}(n)$ rounds) [Kitamura–Izumi 2022]
  - $\mathrm{O}(n \log n)$-round Algorithm (Augmenting Path of Length ℓ in $\mathrm{O}(\ell)$ rounds) [Izumi–Kitamura–Y. 2024]

Bridges and Tenacity

An edge $e = uw$ is a **Bridge** $\overset{\text{def}}{\Longleftrightarrow}$ e does not assign the minlevel of either vertex u or w

- When $e \in M$,
  - $\mathrm{evenlevel}(u) = \mathrm{maxlevel}(u)$
  - $\mathrm{evenlevel}(w) = \mathrm{maxlevel}(w)$
- When $e \in E \setminus M$,
  - $\mathrm{oddlevel}(u) = \mathrm{maxlevel}(u)$ or $\mathrm{oddlevel}(u) < \mathrm{evenlevel}(w) + 1$
  - $\mathrm{oddlevel}(w) = \mathrm{maxlevel}(w)$ or $\mathrm{oddlevel}(w) < \mathrm{evenlevel}(u) + 1$





Bridges and Tenacity

An edge $e = uw$ is a **Bridge** $\overset{\text{def}}{\iff}$ e does not assign the minlevel of either vertex u or w

- When $e \in M$,
  - $\text{evenlevel}(u) = \text{maxlevel}(u)$
  - $\text{evenlevel}(w) = \text{maxlevel}(w)$
- When $e \in E \setminus M$,
  - $\text{oddlevel}(u) = \text{maxlevel}(u)$ or $\text{oddlevel}(u) < \text{evenlevel}(w) + 1$
  - $\text{oddlevel}(w) = \text{maxlevel}(w)$ or $\text{oddlevel}(w) < \text{evenlevel}(u) + 1$

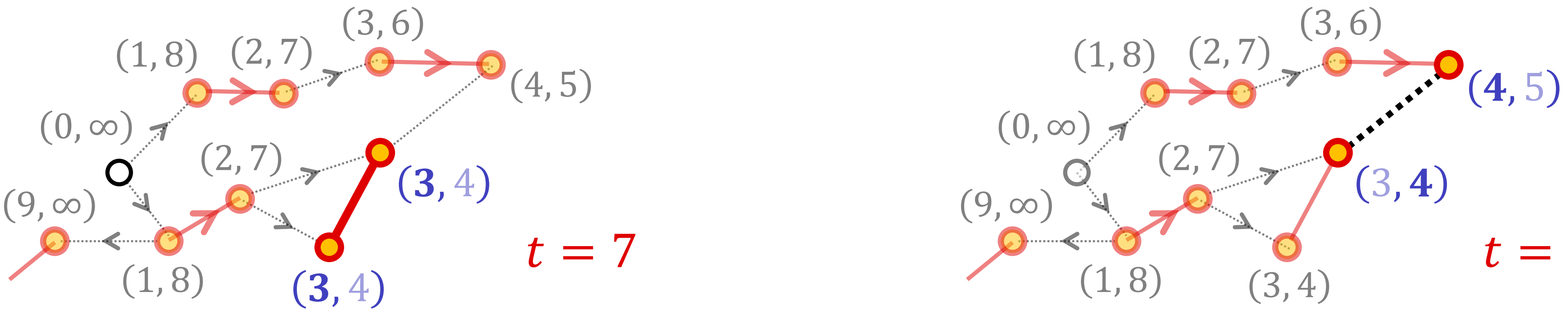
Bridges and Tenacity

An edge $e = uw$ is a **Bridge** $\overset{\text{def}}{\iff}$ e does not assign the minlevel of either vertex u or w

Tenacity of a bridge $e = uw$ or a vertex $v \in V$ is defined as follows:

- $\text{tenacity}(e) \coloneqq \begin{cases} \text{oddlevel}(u) + \text{oddlevel}(w) + 1 & (e \in M) \\ \text{evenlevel}(u) + \text{evenlevel}(w) + 1 & (e \in E \setminus M) \end{cases}$
- $\text{tenacity}(v) \coloneqq \text{oddlevel}(v) + \text{evenlevel}(v) = \text{minlevel}(v) + \text{maxlevel}(v) \quad (v \in V)$

Maxlevel of a vertex of tenacity t is assigned by a bridge of tenacity t



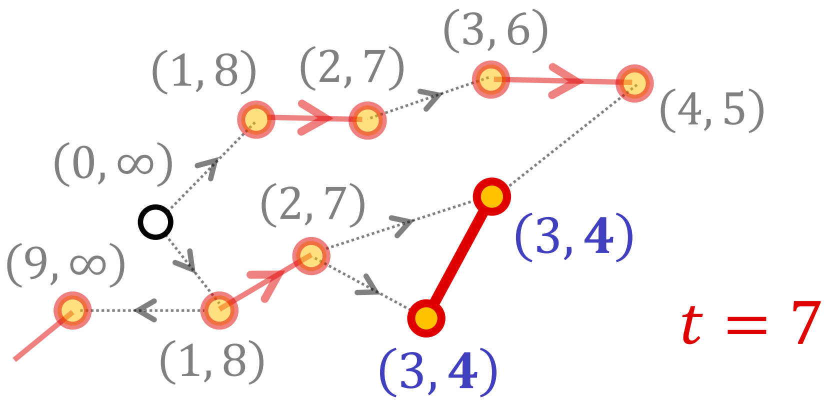
Bridges and Tenacity

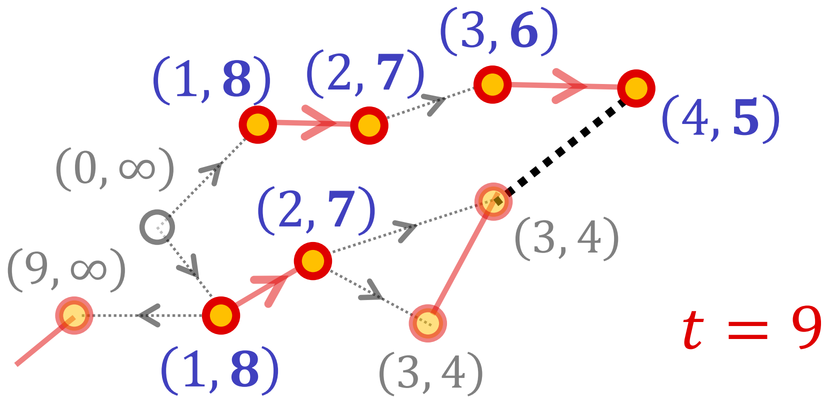
An edge $e = uw$ is a **Bridge** $\overset{\text{def}}{\iff}$ e does not assign the minlevel of either vertex u or w

<u>Tenacity</u> of a bridge $e = uw$ or a vertex $v \in V$ is defined as follows:

- $\text{tenacity}(e) \coloneqq \begin{cases} \text{oddlevel}(u) + \text{oddlevel}(w) + 1 & (e \in M) \\ \text{evenlevel}(u) + \text{evenlevel}(w) + 1 & (e \in E \setminus M) \end{cases}$
- $\text{tenacity}(v) \coloneqq \text{oddlevel}(v) + \text{evenlevel}(v) = \text{minlevel}(v) + \text{maxlevel}(v) \quad (v \in V)$

Maxlevel of a vertex of tenacity t is assigned by a bridge of tenacity t





$O(\sqrt{n}m)$-time Algorithm for General Matching

[Micali–Vazirani 1980; Vazirani 2024]

Do the following in $O(m)$ time in each phase:

<u>Procedure MIN</u>: Find minlevels of all vertices of minlevel $i = 1, 2, \ldots, \frac{\ell-1}{2}$ in this order.

<u>Procedure MAX</u>: Find maxlevels of all vertices of tenacity $t = 3, 5, \ldots, \ell$ in this order.

These are synchronized in ascending order of i and $\frac{t}{2}$.

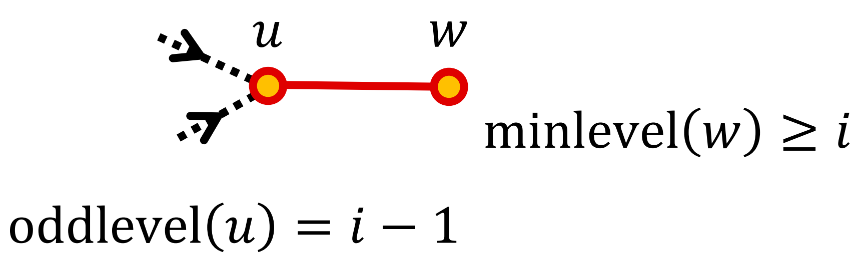
$O(\sqrt{nm})$-time Algorithm for General Matching

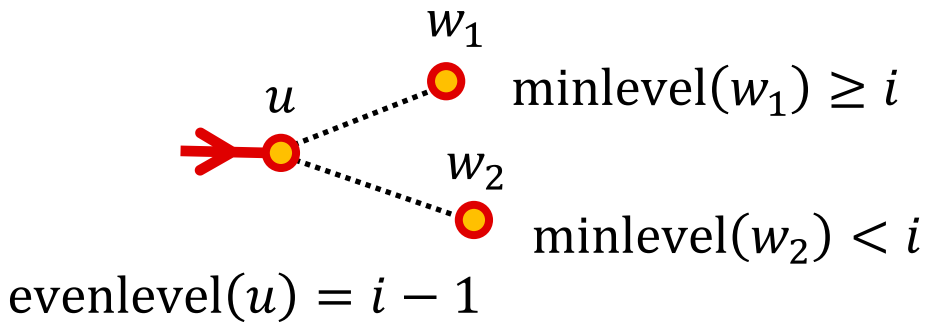
[Micali–Vazirani 1980; Vazirani 2024]

Do the following in $O(m)$ time in each phase:

<u>**Procedure MIN**</u>**:** Find minlevels of all vertices of minlevel $i = 1, 2, \ldots, \frac{\ell-1}{2}$ in this order.

For each $u \in V$ with $\mathrm{odd/evenlevel}(u) = i - 1$ and each **consistent** neighbor w of u, if $\mathrm{minlevel}(w) \geq i$, then update $\mathrm{minlevel}(w) \leftarrow i$ and declare u as a **predecessor** of w.





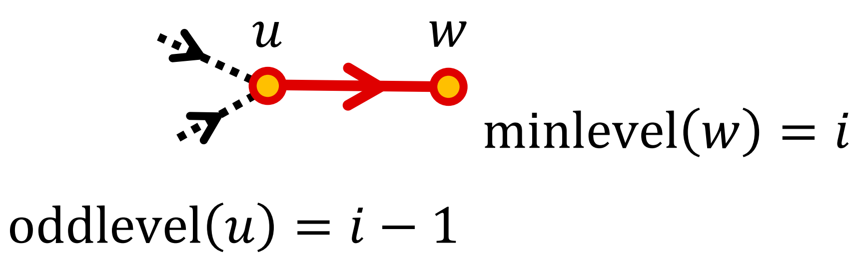
$O(\sqrt{n}m)$-time Algorithm for General Matching

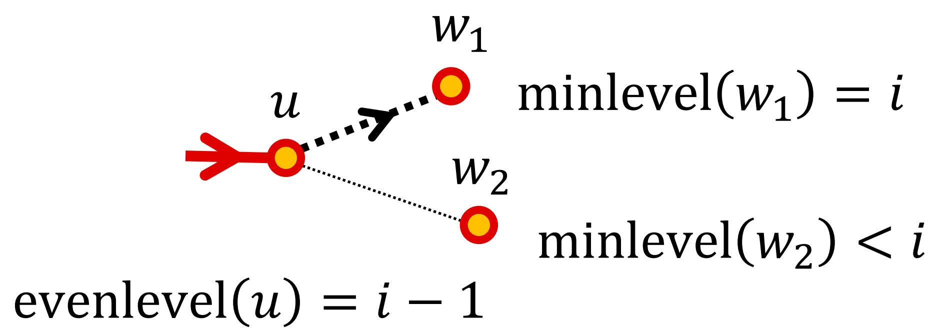
[Micali–Vazirani 1980; Vazirani 2024]

Do the following in $O(m)$ time in each phase:

Procedure MIN: Find minlevels of all vertices of minlevel $i = 1, 2, \dots, \frac{\ell-1}{2}$ in this order.

For each $u \in V$ with $\mathrm{odd/evenlevel}(u) = i - 1$ and each **consistent** neighbor w of u, if $\mathrm{minlevel}(w) \geq i$, then update $\mathrm{minlevel}(w) \leftarrow i$ and declare u as a **predecessor** of w.





$O(\sqrt{n}m)$-time Algorithm for General Matching

[Micali–Vazirani 1980; Vazirani 2024]

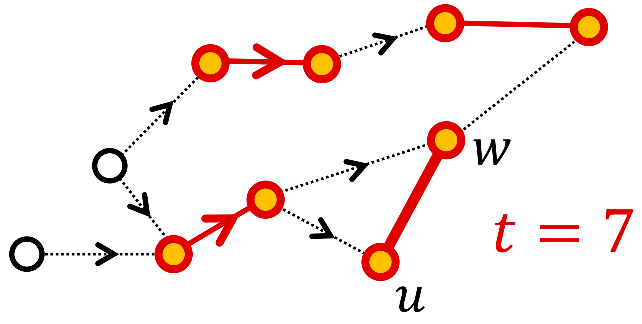
Do the following in $O(m)$ time in each phase:

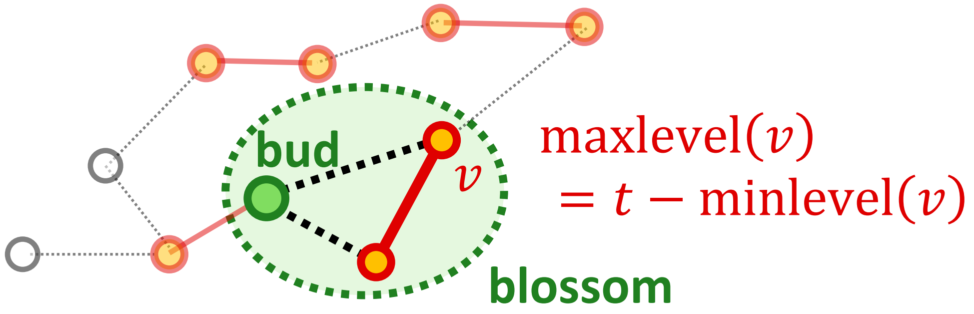
Procedure MIN: Find minlevels of all vertices of minlevel $i = 1, 2, \dots, \frac{\ell-1}{2}$ in this order.

Procedure MAX: Find maxlevels of all vertices of tenacity $t = 3, 5, \dots, \ell$ in this order.

For each bridge $e = uw$ with $\mathrm{tenacity}(e) = t$, by **Double Depth First Search (DDFS)**, find the blossom with its bud, or a **shortest augmenting path** when $t = \ell$.

If such a path is found, recursively remove the vertices that can no longer be used.





$O(\sqrt{n}m)$-time Algorithm for General Matching

[Micali–Vazirani 1980; Vazirani 2024]

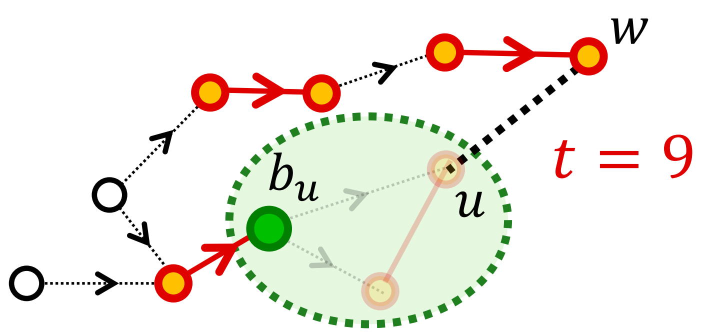
Do the following in $O(m)$ time in each phase:

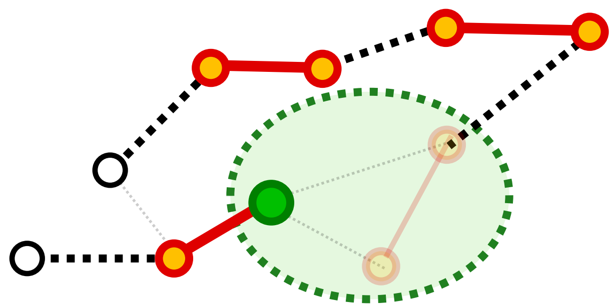
Procedure MIN: Find minlevels of all vertices of minlevel $i = 1, 2, \dots, \frac{\ell-1}{2}$ in this order.

Procedure MAX: Find maxlevels of all vertices of tenacity $t = 3, 5, \dots, \ell$ in this order.

For each bridge $e = uw$ with tenacity$(e) = t$, by **Double Depth First Search (DDFS)**, find the **blossom** with its **bud**, or a shortest augmenting path when $t = \ell$.

If such a path is found, recursively remove the vertices that can no longer be used.





$O(\sqrt{n}m)$-time Algorithm for General Matching

[Micali–Vazirani 1980; Vazirani 2024]

Do the following in $O(m)$ time in each phase:

Procedure MIN: Find minlevels of all vertices of minlevel $i = 1, 2, \dots, \frac{\ell-1}{2}$ in this order.

Procedure MAX: Find maxlevels of all vertices of tenacity $t = 3, 5, \dots, \ell$ in this order.

For each bridge $e = uw$ with $\mathrm{tenacity}(e) = t$, by **Double Depth First Search (DDFS)**, find the **blossom** with its **bud**, or a shortest augmenting path when $t = \ell$.

If such a path is found, recursively remove the vertices that can no longer be used.

$O(\sqrt{n}m)$-time Algorithm for General Matching

[Micali–Vazirani 1980; Vazirani 2024]

Do the following in $O(m)$ time in each phase:

Procedure MIN: Find minlevels of all vertices of minlevel $i = 1, 2, \ldots, \frac{\ell - 1}{2}$ in this order.

Procedure MAX: Find maxlevels of all vertices of tenacity $t = 3, 5, \ldots, \ell$ in this order.

These are **synchronized** in ascending order of i and $\frac{t}{2}$.

[Remark]

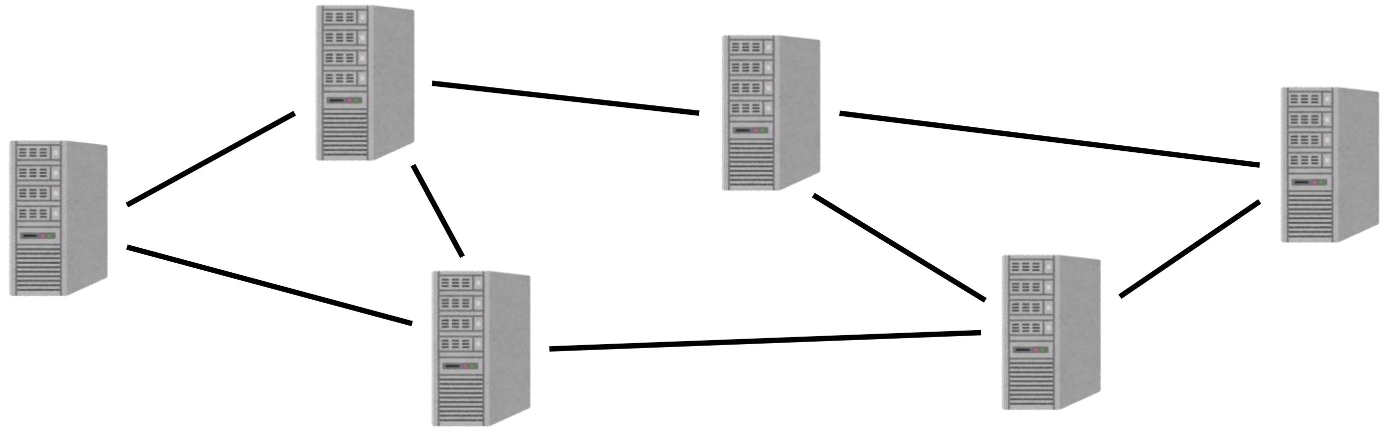
- Synchronization of the two procedures is essential;
  intuitively, it finds minlevels and processes bridges (blossoms) **in a BFS-like manner**.
- Odd/evenlevel paths (also in blossoms) are recursively constructed in linear time;
  this part is also nontrivial due to **nested blossoms**, which are shown to be **well-structured**.

Outline

- Basics: Augmenting Paths and Algorithm Framework [Kőnig 1931; Edmonds 1965]
- $O(\sqrt{n}m)$-time Algorithm (Centralized)
  - Update with Maximal Disjoint Shortest Augmenting Paths [Hopcroft–Karp 1973]
  - BFS-honesty of Shortest Alternating Paths: Bipartite vs. General
  - Overview of $O(\sqrt{n}m)$-time Algorithm in General [Micali–Vazirani 1980; Vazirani 2024]
- $O(n \log n)$-round Algorithm under CONGEST Model (Distributed)
  - $O(n)$-round Matching Verification Algorithm [Ahmadi–Kuhn 2020]
  - $O(n^{1.5})$-round Algorithm (Augmenting Path in $O(n)$ rounds) [Kitamura–Izumi 2022]
  - $O(n \log n)$-round Algorithm (Augmenting Path of Length ℓ in $O(\ell)$ rounds) [Izumi–Kitamura–Y. 2024]

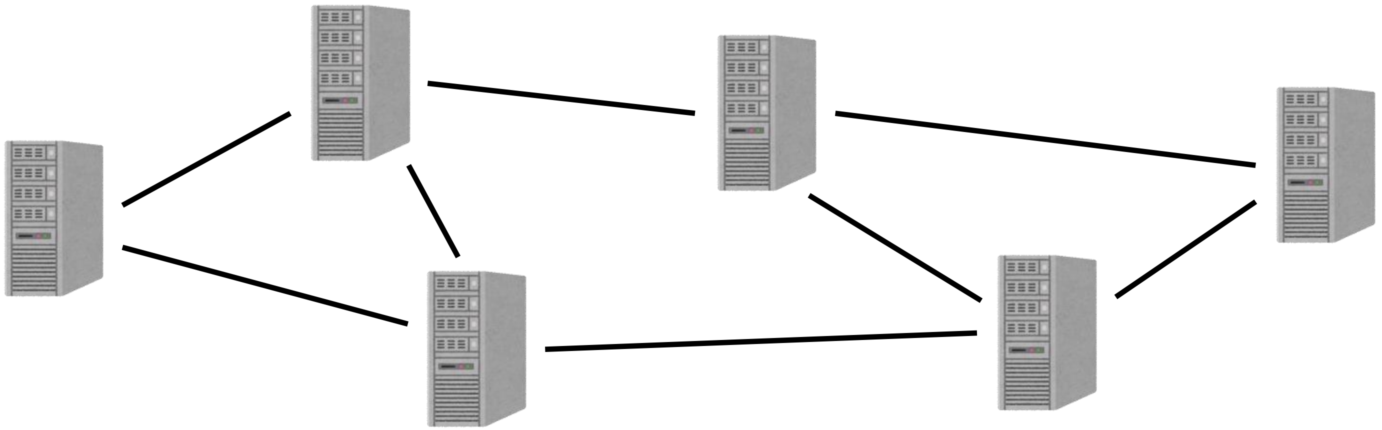
Distributed Situation

- Computers form a communication network (graph)
- Each vertex has sufficient computational power
- Each vertex only knows the local information, itself and its neighbors
- Each vertex can send and receive a message
  through each of its incident edges



CONGEST Model

- Computers form a communication network (graph)
- Each vertex has sufficient computational power
- Each vertex only knows the local information, itself and its neighbors
- Each vertex can send and receive a message **of $O(\log n)$ bits** through each of its incident edges **in each synchronous round**



Question and Trivial Upper Bound

- Computers form a communication network (graph)
- Each vertex has sufficient computational power
- Each vertex only knows the local information, itself and its neighbors
- Each vertex can send and receive a message of $O(\log n)$ bits through each of its incident edges in each **synchronous round**

Q. **How many rounds** are sufficient to solve a problem on the graph?

A. By deciding a leader vertex and gathering all information to it, most problems are solved in $O(m) = O(n^2)$ rounds.

Q. **How faster can it be?**

Question and Trivial Upper Bound

- Computers form a communication network (graph)
- Each vertex has sufficient computational power
- Each vertex only knows the local information, itself and its neighbors
- Each vertex can send and receive a message of $\mathrm{O}(\log n)$ bits through each of its incident edges in each **synchronous round**

Q. **How many rounds** are sufficient to solve a problem on the graph?

A. By deciding a leader vertex and gathering all information to it, most problems are solved in $\mathrm{O}(m) = \mathrm{O}(n^2)$ rounds.

Q. How faster can it be?

$\mu = \mathrm{O}(n)$: opt. value

Exact Algorithms for Maximum Matching

Q. How faster can we find a maximum matching than $\Theta(n^2)$ rounds?

- $\mathrm{O}(\mu^2)$-round deterministic algorithm [Ben-Basat–Kawarabayashi–Schwartzman 2019]
- $\mathrm{O}(\mu \log \mu)$-round deterministic algorithm for bipartite graphs [Ahmadi–Kuhn–Oshman 2018]
- $\mathrm{O}(\mu)$-round randomized algorithm to verify maximality of a matching [Ahmadi–Kuhn 2020]
- $\mathrm{O}(\mu^{1.5})$-round randomized algorithm [Kitamura–Izumi 2022]
- $\mathrm{O}(\mu \log \mu)$-round randomized algorithm [Izumi–Kitamura–Y. 2024]

$\mu = \mathrm{O}(n)$: opt. value

Exact Algorithms for Maximum Matching

Q. How faster can we find a maximum matching than $\Theta(n^2)$ rounds?

- $\mathrm{O}(\mu^2)$-round deterministic algorithm [Ben-Basat–Kawarabayashi–Schwartzman 2019]
- $\mathrm{O}(\mu \log \mu)$-round deterministic algorithm for bipartite graphs [Ahmadi–Kuhn–Oshman 2018]
- $\mathrm{O}(\mu)$-round randomized algorithm to verify maximality of a matching [Ahmadi–Kuhn 2020]
- $\mathrm{O}(\mu^{1.5})$-round randomized algorithm [Kitamura–Izumi 2022]
- $\mathrm{O}(\mu \log \mu)$-round randomized algorithm [Izumi–Kitamura–Y. 2024]

$\mu = \mathrm{O}(n)$: opt. value

Exact Algorithms for Maximum Matching

Q. How faster can we find a maximum matching than $\Theta(n^2)$ rounds?

- $\mathrm{O}(\mu^2)$-round deterministic algorithm [Ben-Basat–Kawarabayashi–Schwartzman 2019]
- $\mathrm{O}(\mu \log \mu)$-round deterministic algorithm for bipartite graphs [Ahmadi–Kuhn–Oshman 2018]
- $\mathrm{O}(\mu)$-round randomized algorithm to verify maximality of a matching [Ahmadi–Kuhn 2020]
- $\mathrm{O}(\mu^{1.5})$-round randomized algorithm [Kitamura–Izumi 2022]
- $\mathrm{O}(\mu \log \mu)$-round randomized algorithm [Izumi–Kitamura–Y. 2024]

Road to $O(\mu \log \mu)$-round Algorithm [Izumi–Kitamura–Y. 2024]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$O(\mu \cdot \mathrm{FP}(n, m))$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)

- It suffices to upper-bound $\mathrm{FP}(n, m)$ by $O(\log \mu)$ in amortized sense.
- $\mathrm{FP}(n, m) = O(\ell)$ is sufficient with the aid of **Hopcroft–Karp analysis**.
- We can restrict a situation (with end vertices and odd/evenlevels known) with the aid of **Ahmadi–Kuhn Matching Verification algorithm**.

Road to $\mathrm{O}(\mu \log \mu)$-round Algorithm [Izumi–Kitamura–Y. 2024]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}(\mu \cdot \mathrm{FP}(n, m))$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)

- It suffices to upper-bound $\mathrm{FP}(n, m)$ by $\mathrm{O}(\log \mu)$ in amortized sense.
- $\mathrm{FP}(n, m) = \mathrm{O}(\ell)$ is sufficient with the aid of **Hopcroft–Karp analysis**.

Lem. A matching M is of cardinality $\mu - k$ $(1 \leq k \leq \mu)$

$\Longrightarrow$ $\exists P$: augmenting path w.r.t. M of length less than $\frac{2\mu}{k}$

Road to $O(\mu \log \mu)$-round Algorithm [Izumi–Kitamura–Y. 2024]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$O(\mu \cdot \mathrm{FP}(n, m))$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)

- It suffices to upper-bound $\mathrm{FP}(n, m)$ by $O(\log \mu)$ in amortized sense.
- $\mathrm{FP}(n, m) = O(\ell)$ is sufficient with the aid of **Hopcroft–Karp analysis**.
- We can restrict a situation (with end vertices and odd/evenlevels known) with the aid of **Ahmadi–Kuhn Matching Verification algorithm**.

Outline

- Basics: Augmenting Paths and Algorithm Framework [Kőnig 1931; Edmonds 1965]
- $O(\sqrt{n}m)$-time Algorithm (Centralized)
  - Update with Maximal Disjoint Shortest Augmenting Paths [Hopcroft–Karp 1973]
  - BFS-honesty of Shortest Alternating Paths: Bipartite vs. General
  - Overview of $O(\sqrt{n}m)$-time Algorithm in General [Micali–Vazirani 1980; Vazirani 2024]
- $O(n \log n)$-round Algorithm under CONGEST Model (Distributed)
  - $O(n)$-round Matching Verification Algorithm [Ahmadi–Kuhn 2020]
  - $O(n^{1.5})$-round Algorithm (Augmenting Path in $O(n)$ rounds) [Kitamura–Izumi 2022]
  - $O(n \log n)$-round Algorithm (Augmenting Path of Length ℓ in $O(\ell)$ rounds) [Izumi–Kitamura–Y. 2024]

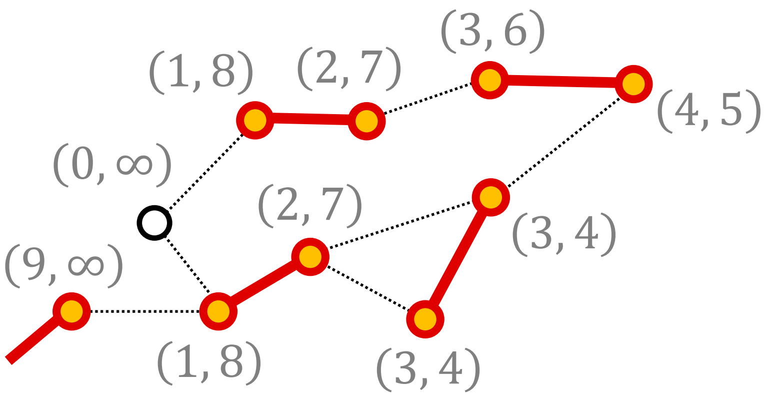
Matching Verification in $O(\min\{\mu, \ell\})$ Rounds

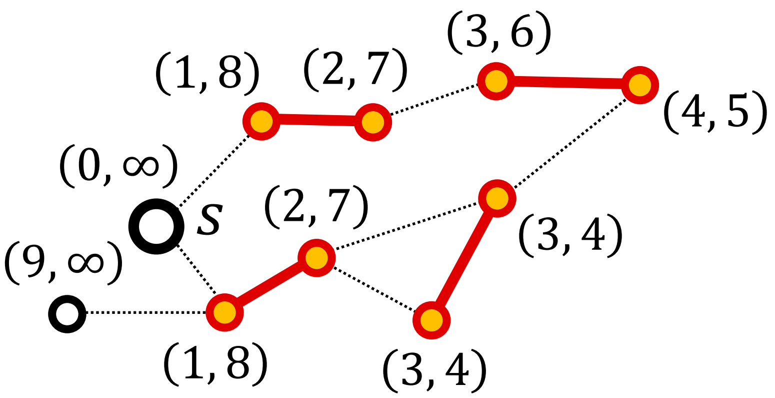
[Ahmadi–Kuhn 2020]

Input: $G = (V, E)$: Undirected Graph, $M \subseteq E$: Matching

Goal: Do the correct one of the following two candidates:

- Determine that M is maximum.
- Find **a pair of end vertices** of a shortest augmenting path w.r.t. M, compute **odd/evenlevels** from one of the end vertices up to ℓ.





Matching Verification in $\mathrm{O}(\min\{\mu, \ell\})$ Rounds

[Ahmadi–Kuhn 2020]

Input: $G = (V, E)$: Undirected Graph, $M \subseteq E$: Matching

Goal: Do the correct one of the following two candidates:

- Determine that M is maximum.
- Find **a pair of end vertices** of a shortest augmenting path w.r.t. M, compute **odd/evenlevels** from one of the end vertices up to ℓ.

Thm. This problem is solved by a randomized CONGEST algorithm that terminates in $\mathrm{O}(\mu)$ rounds (former) and in $\mathrm{O}(\ell)$ rounds (latter)

"We hope that our algorithm constitutes a significant step towards developing a CONGEST algorithm to *compute* a maximum matching in time $\tilde{O}(s^*)$, where s^* is the size of a maximum matching."

Outline

- Basics: Augmenting Paths and Algorithm Framework [Kőnig 1931; Edmonds 1965]
- $O(\sqrt{n}m)$-time Algorithm (Centralized)
  - Update with Maximal Disjoint Shortest Augmenting Paths [Hopcroft–Karp 1973]
  - BFS-honesty of Shortest Alternating Paths: Bipartite vs. General
  - Overview of $O(\sqrt{n}m)$-time Algorithm in General [Micali–Vazirani 1980; Vazirani 2024]
- $O(n \log n)$-round Algorithm under CONGEST Model (Distributed)
  - $O(n)$-round Matching Verification Algorithm [Ahmadi–Kuhn 2020]
  - $O(n^{1.5})$-round Algorithm (Augmenting Path in $O(n)$ rounds) [Kitamura–Izumi 2022]
  - $O(n \log n)$-round Algorithm (Augmenting Path of Length ℓ in $O(\ell)$ rounds) [Izumi–Kitamura–Y. 2024]

Road to $\mathrm{O}(\mu \log \mu)$-round Algorithm [Izumi–Kitamura–Y. 2024]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}(\mu \cdot \mathrm{FP}(n, m))$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)

- It suffices to upper-bound $\mathrm{FP}(n, m)$ by $\mathrm{O}(\log \mu)$ in amortized sense.
- $\mathrm{FP}(n, m) = \mathrm{O}(\ell)$ is sufficient with the aid of **Hopcroft–Karp analysis**.
- We can restrict a situation (with end vertices and odd/evenlevels known) with the aid of **Ahmadi–Kuhn Matching Verification algorithm**.

$\mathrm{O}(\mu^{1.5})$-round Algorithm [Kitamura–Izumi 2022]

[Naive Algorithm]

1. $M \leftarrow \emptyset$
2. While $\exists P$: augmenting path w.r.t. M, find it and update $M \leftarrow M \triangle P$

$\mathrm{O}\left(\mu \cdot \mathrm{FP}(n, m)\right)$ time ($\mathrm{FP}(\cdot)$: time to find an augmenting path)

- It suffices to upper-bound $\mathrm{FP}(n, m)$ by $\mathrm{O}\left(\sqrt{\mu}\right)$ in amortized sense.
- $\mathrm{FP}(n, m) = \mathrm{O}(\min\{\ell^2, \mu\})$ is sufficient with the aid of **HK analysis**.
- We can restrict a situation (with end vertices and odd/evenlevels known) with the aid of **Ahmadi–Kuhn Matching Verification algorithm**.

$\mathrm{O}(\mu^{1.5})$-round Algorithm [Kitamura–Izumi 2022]

$\mathrm{FP}(n, m) = \mathrm{O}(\min\{\ell^2, \mu\})$ is sufficient with the aid of **Hopcroft–Karp analysis**

Lem. A matching M is of cardinality $\mu - k \ (1 \le k \le \mu)$

$\Longrightarrow \ \exists P$: augmenting path w.r.t. M of length less than $\frac{2\mu}{k}$

- In the first $\mu - \sqrt{\mu}$ augmentations, use an $\mathrm{O}(\ell^2)$-round algorithm.

$$\sum_{k=0}^{\mu-\sqrt{\mu}} \left(\frac{2\mu}{\mu - k}\right)^2 = 4\mu^2 \sum_{j=\sqrt{\mu}}^{\mu} \left(\frac{1}{j}\right)^2 \approx 4\mu^2 \int_{\sqrt{\mu}}^{\mu} x^{-2} \mathrm{d}x \approx 4\mu^2 \cdot \frac{1}{2\sqrt{\mu}} = 2\mu^{1.5}$$

- In the last $\sqrt{\mu}$ augmentations, use an $\mathrm{O}(\mu)$-round algorithm.

Find Augmenting Path in $\mathrm{O}(\min\{\ell^2, \mu\})$ Rounds

[Kitamura–Izumi 2022]

Both algorithms are based on **Ahmadi–Kuhn Matching Verification**

- $\mathrm{O}(\ell^2)$-round is straightforward:

  A path is constructed from one end vertex by finding a predecessor one-by-one.

- $\mathrm{O}(\mu)$-round is achieved by Construction of **Sparse Subgraph**:
  - It consists of $\mathrm{O}(\mu)$ edges.
  - It preserves at least one **odd/even alternating paths** from one end vertex.
  - In particular, it contains an **augmenting path**.

  → Gathering all information to a leader and distributing the result in the subgraph.

Such a subgraph exists because of the correctness of Edmonds' blossom algorithm!

Find Augmenting Path in $O(\min\{\ell^2, \mu\})$ Rounds

[Kitamura–Izumi 2022]

Both algorithms are based on **Ahmadi–Kuhn Matching Verification**

- $O(\ell^2)$-round is straightforward:

  A path is constructed from one end vertex by finding a predecessor one-by-one.

- $O(\mu)$-round is achieved by Construction of **Sparse Subgraph**:
  - It consists of $O(\mu)$ edges.
  - It preserves at least one **odd/even alternating paths** from one end vertex.
  - In particular, it contains an **augmenting path**.

  → Gathering all information to a leader and distributing the result in the subgraph.

  Such a subgraph exists because of the correctness of Edmonds' blossom algorithm!

How to Reduce to $\mathrm{O}(\ell)$ Rounds [Izumi–Kitamura–Y. 2024]

Both algorithms are based on **Ahmadi–Kuhn Matching Verification**

- $\mathrm{O}(\ell^2)$-round is straightforward:

  A path is constructed from one end vertex by finding a predecessor one-by-one.

- $\mathrm{O}(\mu)$-round is achieved by Construction of **Sparse Subgraph**:
  - It consists of $\mathrm{O}(\mu)$ edges.
  - It preserves at least one **odd/even alternating paths** from one end vertex.
  - In particular, it contains an **augmenting path** **(which can be arbitrarily long!)**.

  → Gathering all information to a leader and distributing the result in the subgraph.

Such a subgraph exists because of the correctness of Edmonds' blossom algorithm!

How to Reduce to $O(\ell)$ Rounds [Izumi–Kitamura–Y. 2024]

Both algorithms are based on **Ahmadi–Kuhn Matching Verification**

- $O(\ell^2)$-round is straightforward:

  A path is constructed from one end vertex by finding a predecessor one-by-one.

- $O(\ell)$-round is achieved by Construction of **Sparse Subgraph**:
  - All the vertices are of tenacity (= oddlevel + evenlevel) at most ℓ.
  - It preserves at least one **shortest odd/even alternating paths between necessary pairs**.
  - In particular, it contains (and is enough to reconstruct) a **shortest augmenting path**.

  → Gathering all information to a leader and distributing the result in the subgraph.

Such a subgraph is constructed by getting inspiration from Micali–Vazirani algorithm!

Outline

- Basics: Augmenting Paths and Algorithm Framework [Kőnig 1931; Edmonds 1965]
- $\mathrm{O}(\sqrt{n}m)$-time Algorithm (Centralized)
  - Update with Maximal Disjoint Shortest Augmenting Paths [Hopcroft–Karp 1973]
  - BFS-honesty of Shortest Alternating Paths: Bipartite vs. General
  - Overview of $\mathrm{O}(\sqrt{n}m)$-time Algorithm in General [Micali–Vazirani 1980; Vazirani 2024]
- $\mathrm{O}(n \log n)$-round Algorithm under CONGEST Model (Distributed)
  - $\mathrm{O}(n)$-round Matching Verification Algorithm [Ahmadi–Kuhn 2020]
  - $\mathrm{O}(n^{1.5})$-round Algorithm (Augmenting Path in $\mathrm{O}(n)$ rounds) [Kitamura–Izumi 2022]
  - $\mathrm{O}(n \log n)$-round Algorithm (Augmenting Path of Length ℓ in $\mathrm{O}(\ell)$ rounds) [Izumi–Kitamura–Y. 2024]