DAG パス幅に基づく種々のアルゴリズムの設計 及び DAG 木幅への拡張

2025/02/05

伊豆 真哉，川原 純
京都大学大学院情報学研究科 湊研究室

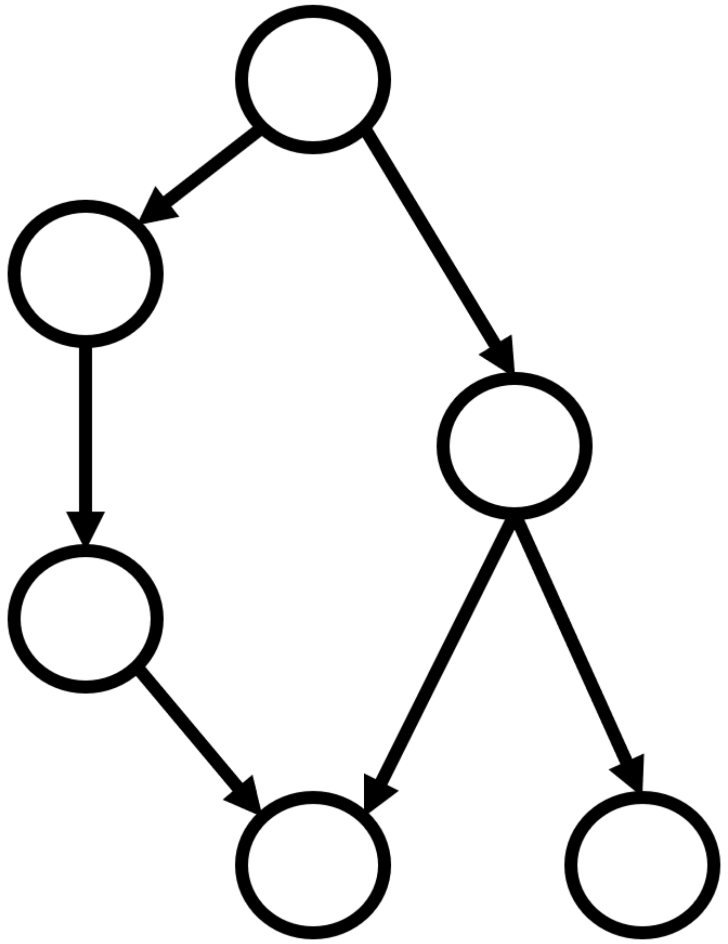
発表の流れ

1. 研究背景
2. DAGパス分解とDAGパス幅の定義
3. 本研究の成果
4. まとめ

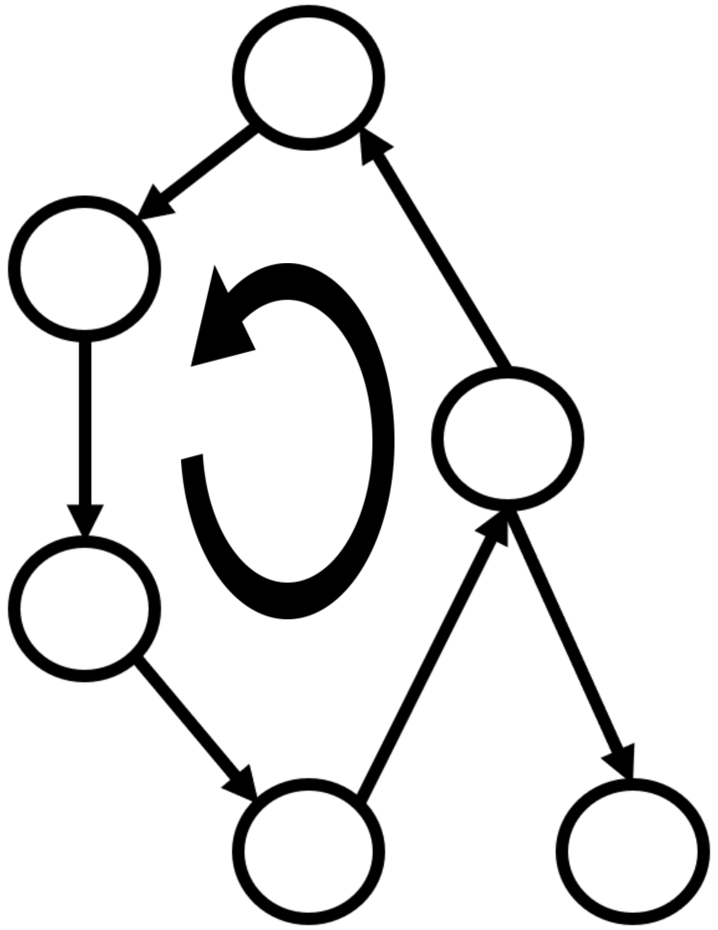
1. 研究背景

今回考えるグラフ構造【DAG】

- **DAG** (Directed Acyclic Graph)：
  有向非巡回グラフ．サイクルのない有向グラフ
- 今回は入力グラフがDAGの場合の組合せ問題を考える

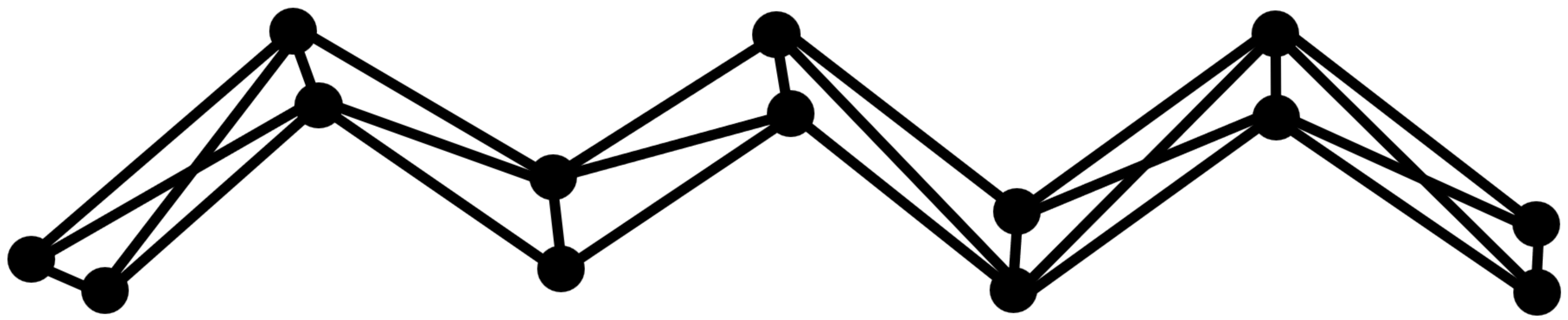


DAG



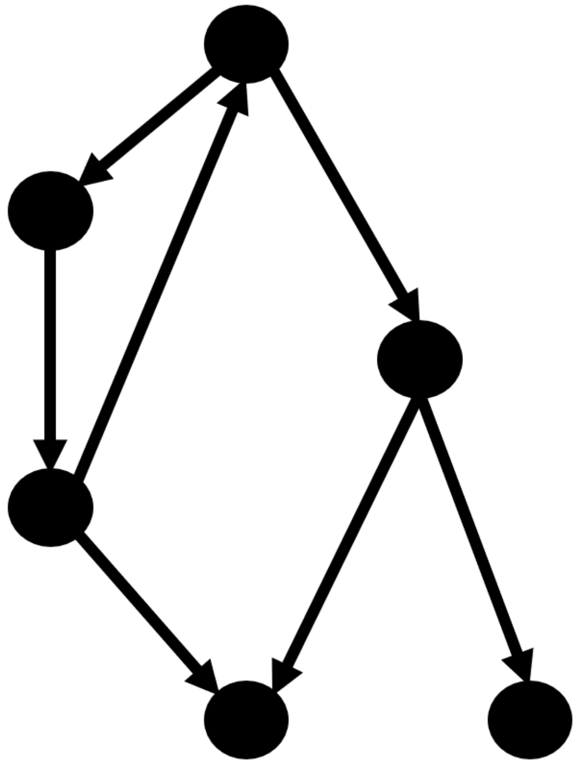
DAGでない

パス幅とは



- パス幅[Robert et al. 83]：
  無向グラフが**どれだけパスに近いか**

パスに近い
→ パス幅 <u>小</u>

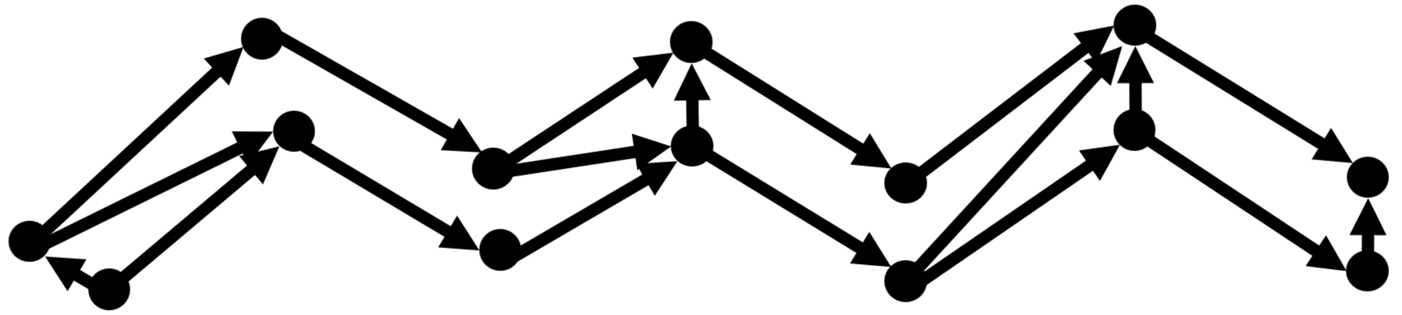


- 有向パス幅[Johnson et al. 01]：
  有向グラフが**どれだけDAGに近いか**

DAGに近い
→ 有向パス幅 <u>小</u>

- 有用性：**パラメータ化アルゴリズム**で利用
  - 最大独立集合問題　：$2^{O(w)}n$ 時間 (n: 頂点数, w: パス幅) [Lim et al. 18]
  - 有向ハミルトン閉路問題：$n^{O(w)}$ 時間 (w: 有向パス幅) [Johnson et al. 01]

有向パス幅はDAGに対して有用でない



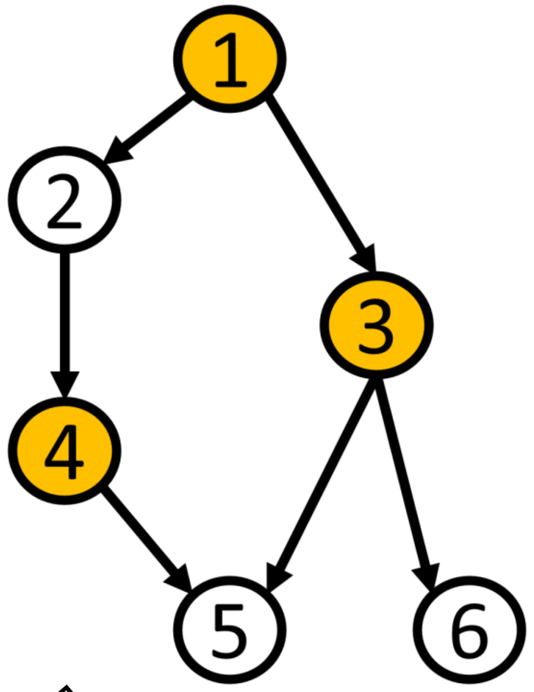
有向パスに近い → DAGパス幅 小

- 有向パス幅の欠点：
  入力グラフがDAGだと常に0
  (DAGはDAGへの近さ0)

DAGにも
対応させたい…

- **DAGパス幅** [Kasahara et al. 23] ：
  有向グラフがどれだけ有向パスに近いか
  →DAGの複雑さも表現可能

有向支配集合問題
(入力がDAGでもNP-hard)



有向パス幅
→解けない

DAGパス幅: w
→$O(2^w wn)$

本研究の成果【DAGパス幅に関する種々のアルゴリズムを設計】

①4つのパラメータ化アルゴリズム

有向支配集合問題→ $O(2^w wn)$

最大葉分岐数問題→ $O(2^w wn)$

***k*-有向点素パス問題**→ $O((k+1)^w (w^2+k)\, n)$

***k*-有向シュタイナー木問題**→ $O(2^w (k+w)n)$

n: 頂点数, w: DAGパス幅

③nによらない幅を求めるアルゴリズム

今回説明

幅が高々$\boldsymbol{O(ld^k)}$のDAGパス分解

→グラフの埋め込みを利用

※l: 根数, d: 最大出次数, k: 入力整数

②DAGパス幅を求める近似アルゴリズム

$\boldsymbol{O(\log^2 n)}$**-近似**

→セパレータを利用

$\boldsymbol{O(\log^{3/2} n)}$**-近似**

→Pebbling gameを利用

④DAG木幅への拡張

DAG木幅→ 有向木への近さを表現

→ DAGパス幅より小

有向支配集合問題→ $O(2^w w^2 n)$

パス幅の計算は難しい

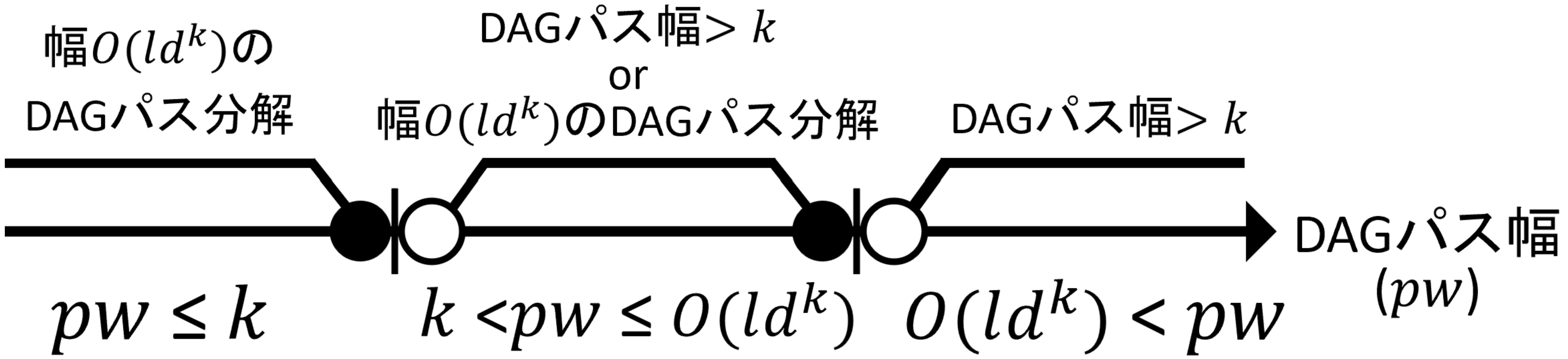
- 入力整数kに対し，Gのパス幅はk以下か？→ **NP-complete**
- ではどうするか…→ 以下のアルゴリズムを利用する
- DAGパス幅のアルゴリズムは知られていなかった

入力：頂点数$\boldsymbol{n}$のグラフ$\boldsymbol{G}$, 整数$\boldsymbol{k}$
出力：「パス幅 > $\boldsymbol{k}$の事実と証拠」or「幅が高々$\boldsymbol{w(k)}$のパス分解」
時間：$\boldsymbol{f(n,k)}$ (nに対し多項式時間)

パス幅の種類	G	$w(k)$	$f(n,k)$	参照
パス幅	無向グラフ	$O(2^k)$	$O(2^k n)$	[Kevin et al. 96]
		k	$2^{O(k^3)}n$	[Bodlaender 96]
有向パス幅	有向グラフ	k	$O(mn^{k+1})$ (m=$\|E(G)\|$)	[Tamaki 2011]
DAGパス幅	***open***			

本研究の成果③

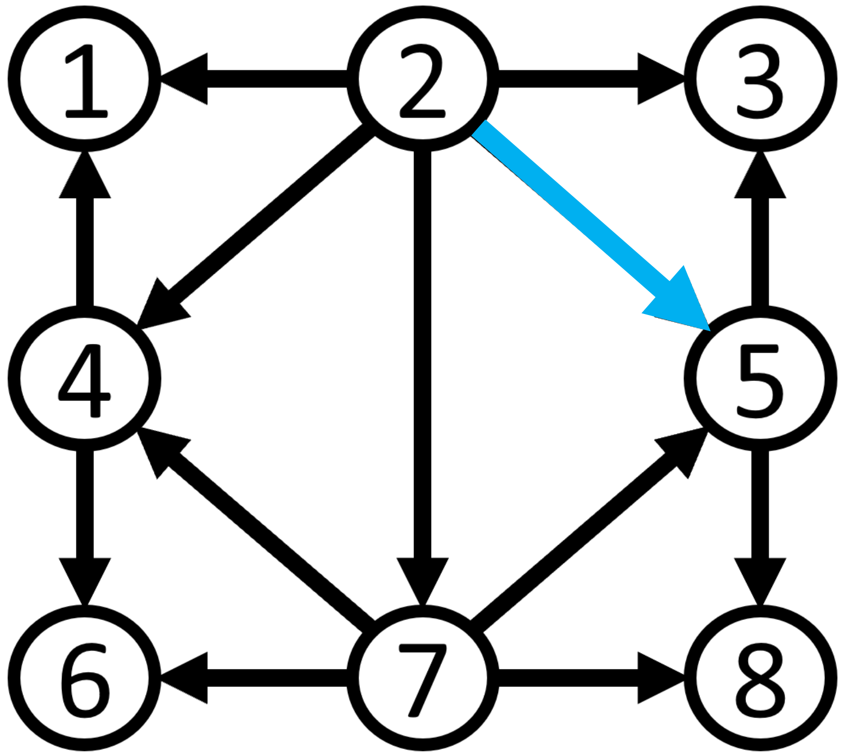
- 「DAGパス幅 > kの事実と証拠」or「幅が高々$O(ld^k)$のDAGパス分解」を出力するアルゴリズムを初めて構築　(l: Gの根数, d: Gの最大出次数)



パス幅の種類	G	$w(k)$	$f(k)$	参照
パス幅	無向グラフ	$O(2^k)$	$O(2^k n)$	[Kevin et al. 96]
		k	$2^{O(k^3)}n$	[Bodlaender 96]
有向パス幅	有向グラフ	k	$O(mn^{k+1})$ ($m=\|E(G)\|$)	[Tamaki 2011]
DAGパス幅	DAG	$\boldsymbol{O(ld^k)}$	$\boldsymbol{O(d^k n^2)}$	

2. DAGパス分解とDAGパス幅の定義

DAGパス分解・DAGパス幅 [Kasahara et al. 23]



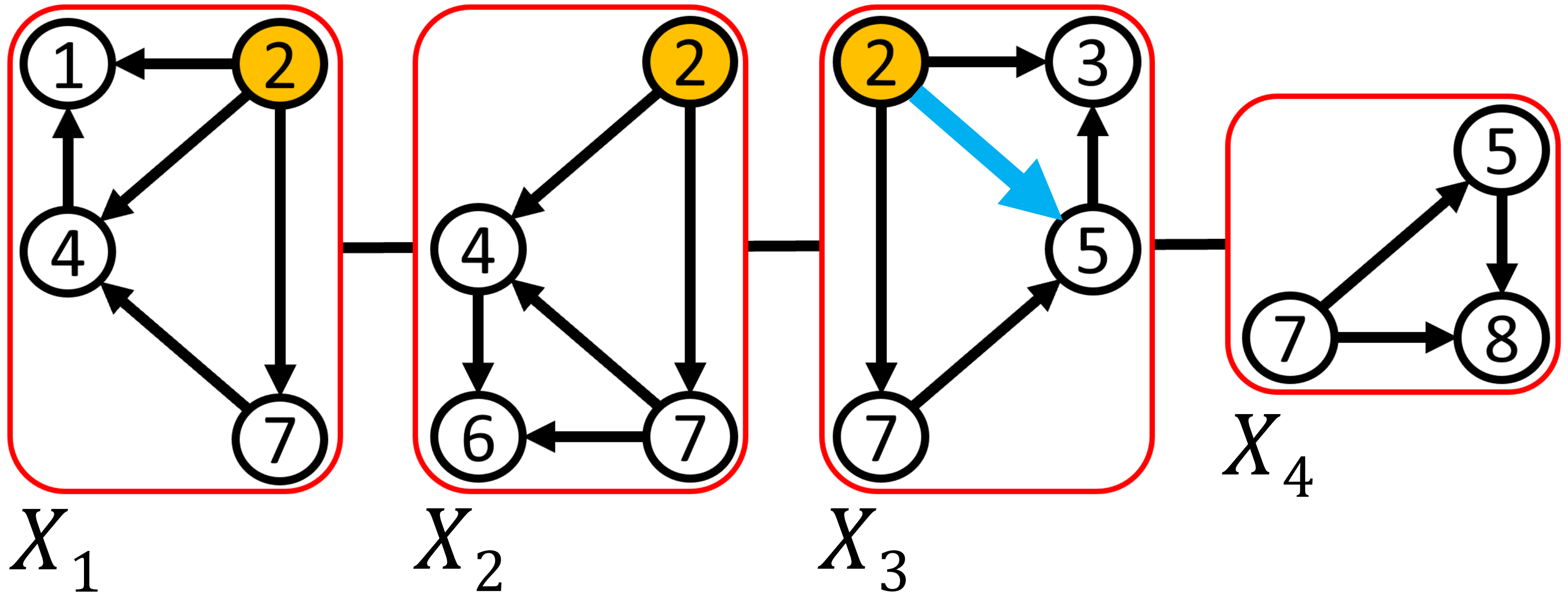
■ **DAGパス分解**：有向グラフ$G = (V, E)$に対し，以下を満たすパス$X = (X_1, X_2, \ldots, X_s)$.

1. $X_1 \cup X_2 \cup \cdots \cup X_s = V$
2. 各$(u, v) \in E$に対し，あるiがあり，$u, v \in X_i$, $\boldsymbol{v \notin X_{i-1}}$
3. 各$v \in V$に対し，vを含むバッグは連結なパスを構成する

パス分解との違い

DAGパス分解

幅=DAGパス幅=3



■ 幅：$\max_{1 \le i \le s} |X_i|$ -1

■ **DAGパス幅**：最小の幅を与えるDAGパス分解の幅. 値が小さいほど**有向パスに近い**グラフ

3. 今回紹介するアルゴリズム

参考にしたアルゴリズム

- **[Kevin et al. 96]**：多項式時間で「パス幅 > kの事実とその証拠」or「幅が高々$O(2^k)$のパス分解」を出力
- 入力グラフに二分木を**埋め込み**することを行っている

入力：頂点数$\boldsymbol{n}$のグラフ$\boldsymbol{G}$, 整数$\boldsymbol{k}$
出力：「パス幅 > $\boldsymbol{k}$の事実と証拠」or「幅が高々$\boldsymbol{w(k)}$のパス分解」
時間：$\boldsymbol{f(n,k)}$　(nに対し多項式時間)

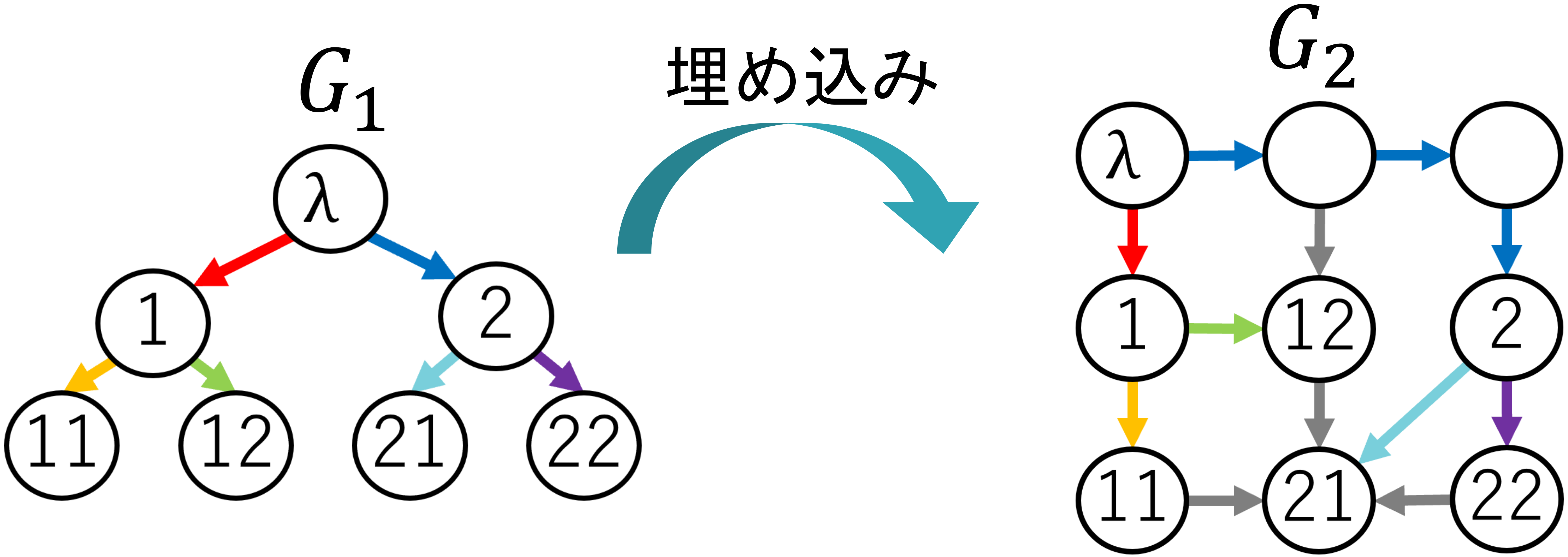
パス幅の種類	G	$w(k)$	$f(n,k)$	参照
パス幅	無向グラフ	$O(2^k)$	$O(2^k n)$	**[Kevin et al. 96]**
		k	$2^{O(k^3)}n$	[Bodlaender 96]
有向パス幅	有向グラフ	k	$O(mn^{k+1})$ ($m=\|E(G)\|$)	[Tamaki 2011]
DAGパス幅	DAG	$\boldsymbol{O(ld^k)}$	$\boldsymbol{O(d^k n^2)}$	

【準備】DAGの埋め込み

■**埋め込み**：

DAG $G_1 = (V_1, E_1)$からDAG $G_2 = (V_2, E_2)$への埋め込みとは，以下を満たすV_1からV_2への写像$f : V_1 \to V_2$．

- fは単射
- E_1の各枝はG_2において互いに素なパスに対応



【準備】完全有向d分技のDAGパス幅

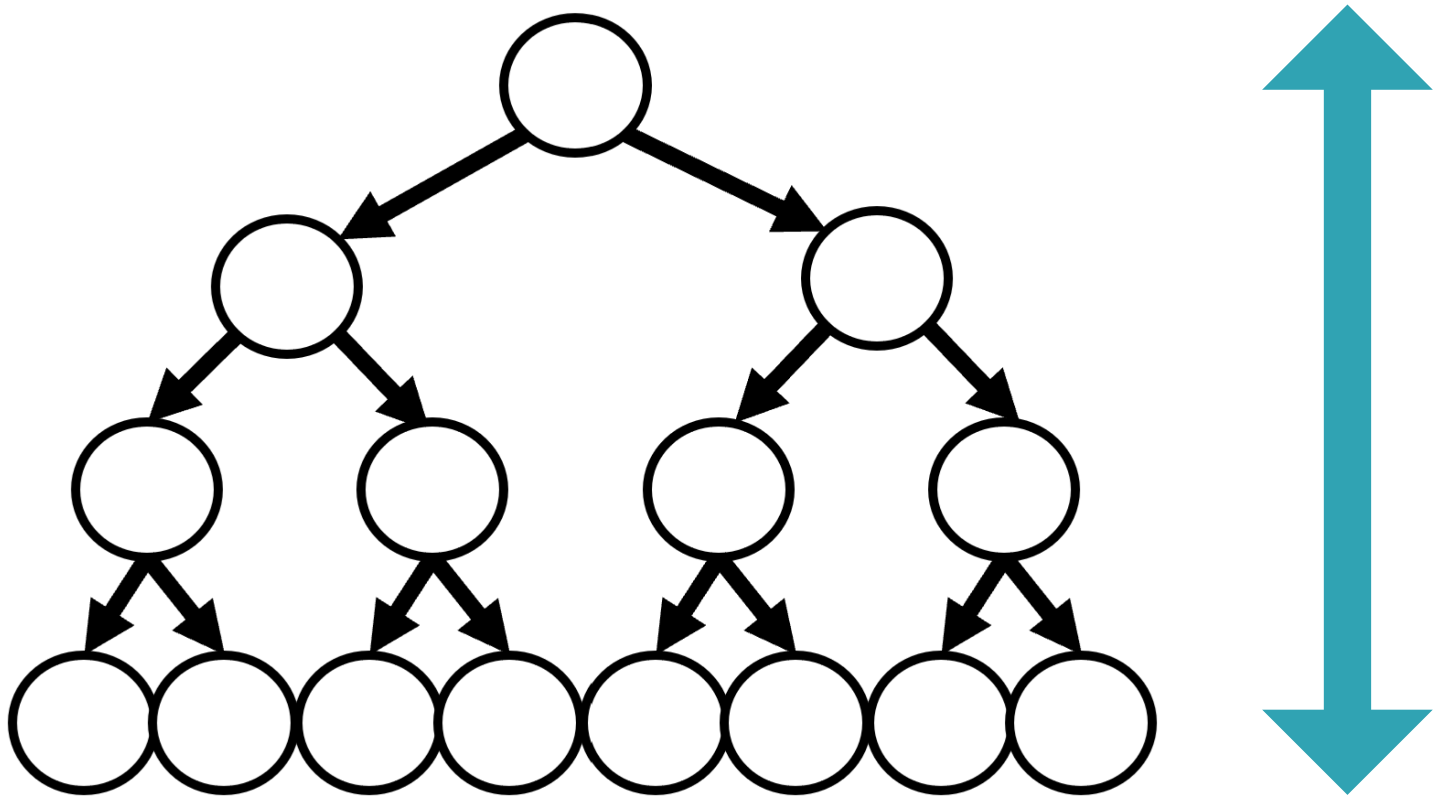
■完全有向d分木：

・ただ一つの根をもつ有向木

・葉以外の各頂点が$d(\geq 2)$個の子をもつ

・全ての葉が同じ高さにある

■高さhの完全有向d分木のDAGパス幅：$\boldsymbol{h-1}$

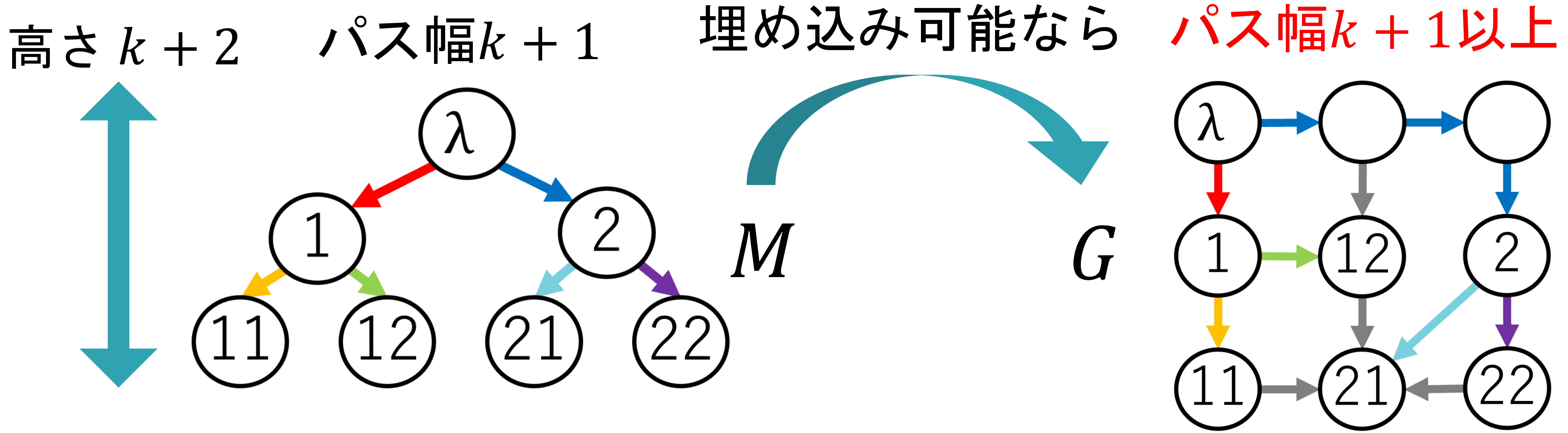


$h = 4$の完全有向2分木

→DAGパス幅は3

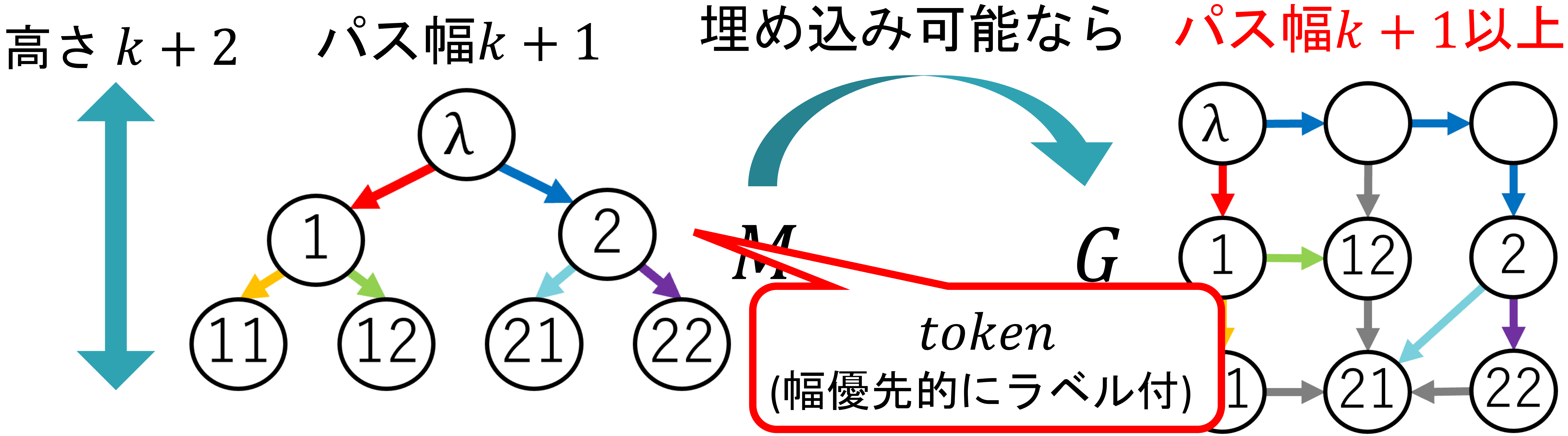
アルゴリズムのアイデア

- input：最大出次数$\boldsymbol{d}$のDAG $\boldsymbol{G}$, 非負整数$\boldsymbol{k}$
- 用意するもの：高さ$\boldsymbol{k+2}$の完全有向d分木$\boldsymbol{M}$ (←DAGパス幅$k+1$)
- $\boldsymbol{M}$が$\boldsymbol{G}$へ埋め込み可能なら… →**$\boldsymbol{G}$のDAGパス幅は$k+1$以上**



アルゴリズムのアイデア

- input：最大出次数$\boldsymbol{d}$のDAG $\boldsymbol{G}$, 非負整数$\boldsymbol{k}$
- 用意するもの：高さ$\boldsymbol{k+2}$の完全有向d分木$\boldsymbol{M}$ (←DAGパス幅$k+1$)
- $\boldsymbol{M}$が$\boldsymbol{G}$へ埋め込み可能なら… →**$\boldsymbol{G}$のDAGパス幅は$\boldsymbol{k+1}$以上**

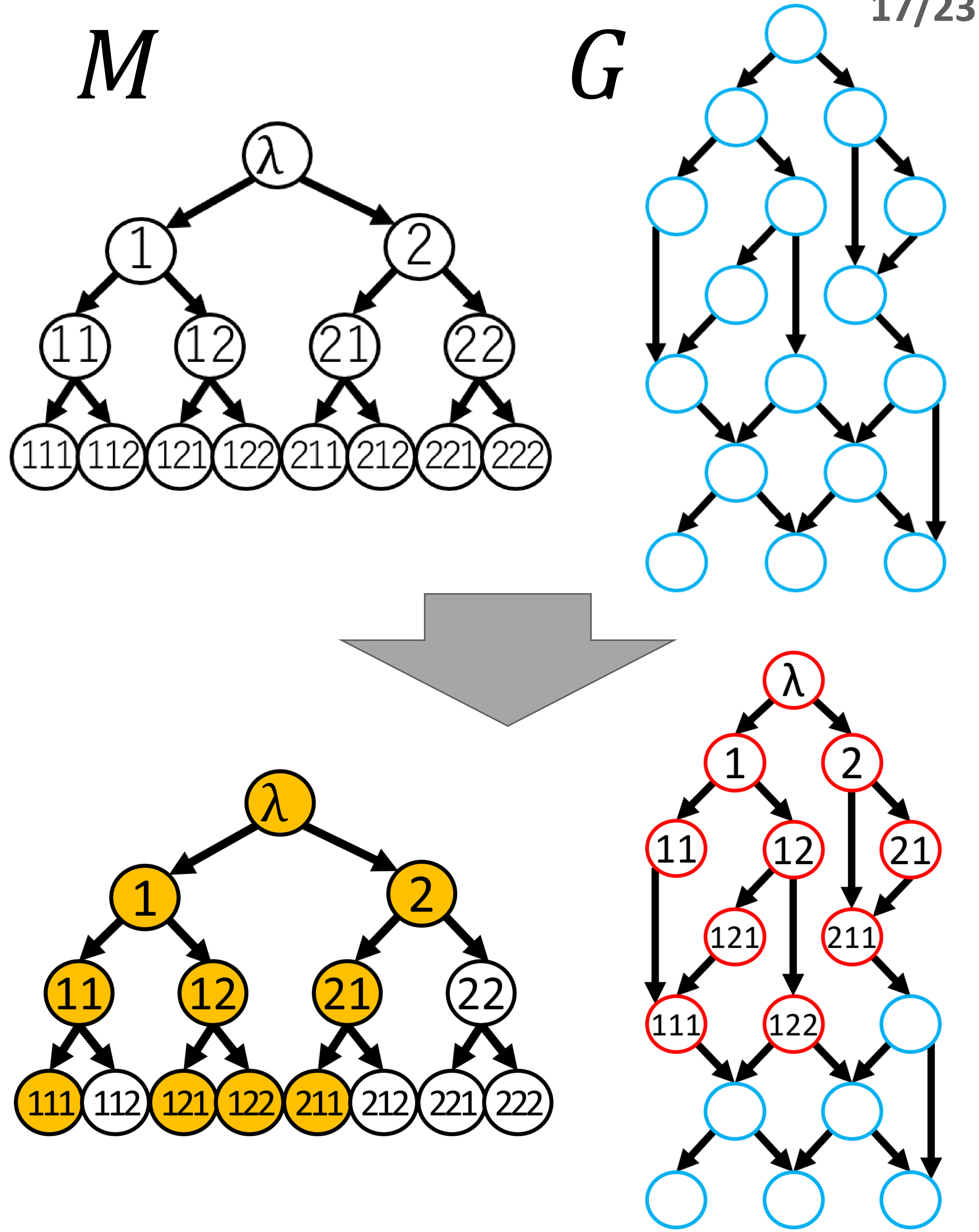


アルゴリズム1：*GrowTokenTree*

■初めにGの全頂点はblue
→ *token*が置かれたらredに
→ *token*が除かれてもredのまま

GrowTokenTree

- Mの木構造を保ったままgreedyにGに*token*を(可能な限り)置いていく
- 親全てに*token*が置かれている場合のみ自身に*token*を置ける
- 最後に*token*を置いた頂点集合のみ出力



アルゴリズム2：*FindEmbedding*

1. Gの根に$token\ \lambda$を置く
2. $X_1 \leftarrow GrowTokenTree$. これをDAGパス分解の**<u>最初のバッグ</u>**
3. until 【Gの全頂点がred】or【$|X_i| = |V[M]|$】:

   if {ある$token\ T$ ($on\ v \in V[G]$)があり, $suc(v)$が全てred,
   かつTの配置済の子$token$が高々1個}:

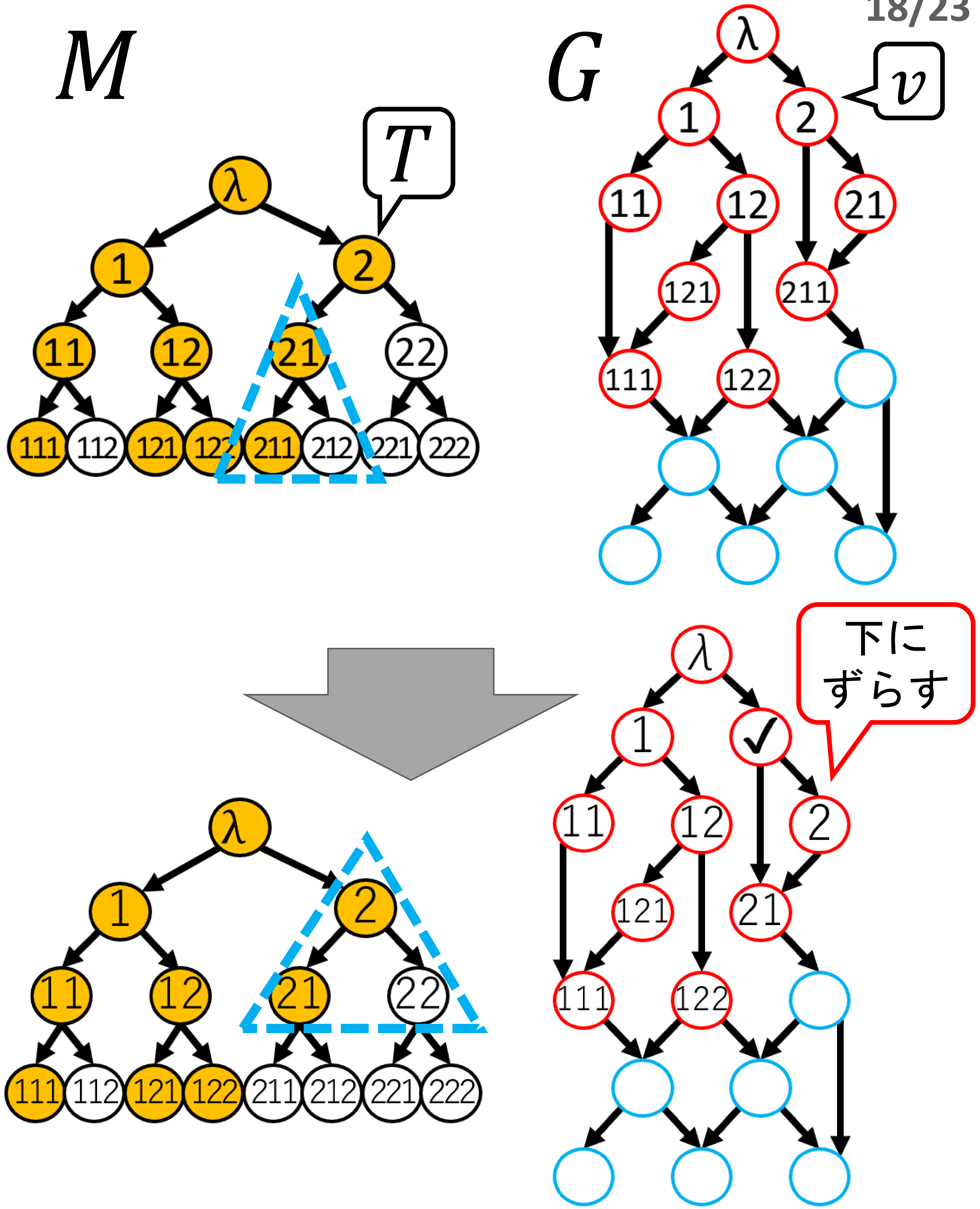
   - Tをvから取り除く
   - Tが配置済の子$T \cdot b$をもつ: $T \cdot b \cdot S \leftrightarrow T \cdot S$

   else:

   - return X_i

   $X_{i+1} \leftarrow GrowTokenTree$. これを**<u>$i$+1番目のバッグ</u>**

$token\ T$を選んで
"下にずらしていく"



- *GrowTokenTree*の出力がDAGパス分解の各バッグに対応

①②③のいずれかで終了

1. Gの根に$token\ \lambda$を置く
2. $X_1 \leftarrow GrowTokenTree$. これをDAGパス分解の最初のバッグ
3. until 【Gの全頂点がred】 or 【$|X_i| = |V[M]|$】 :

   1

   if {ある$token\ T\ (on\ v \in\ V[G])$があり, $suc(v)$が全てred, かつTの配置済の子$token$が高々1個}:
   - Tをvから取り除く
   - Tが配置済の子$T \cdot b$をもつ: $T \cdot b \cdot S \leftrightarrow T \cdot S$

   else:
   - return X_i

   $X_{i+1} \leftarrow GrowTokenTree$. これを$i$+1番目のバッグ

2

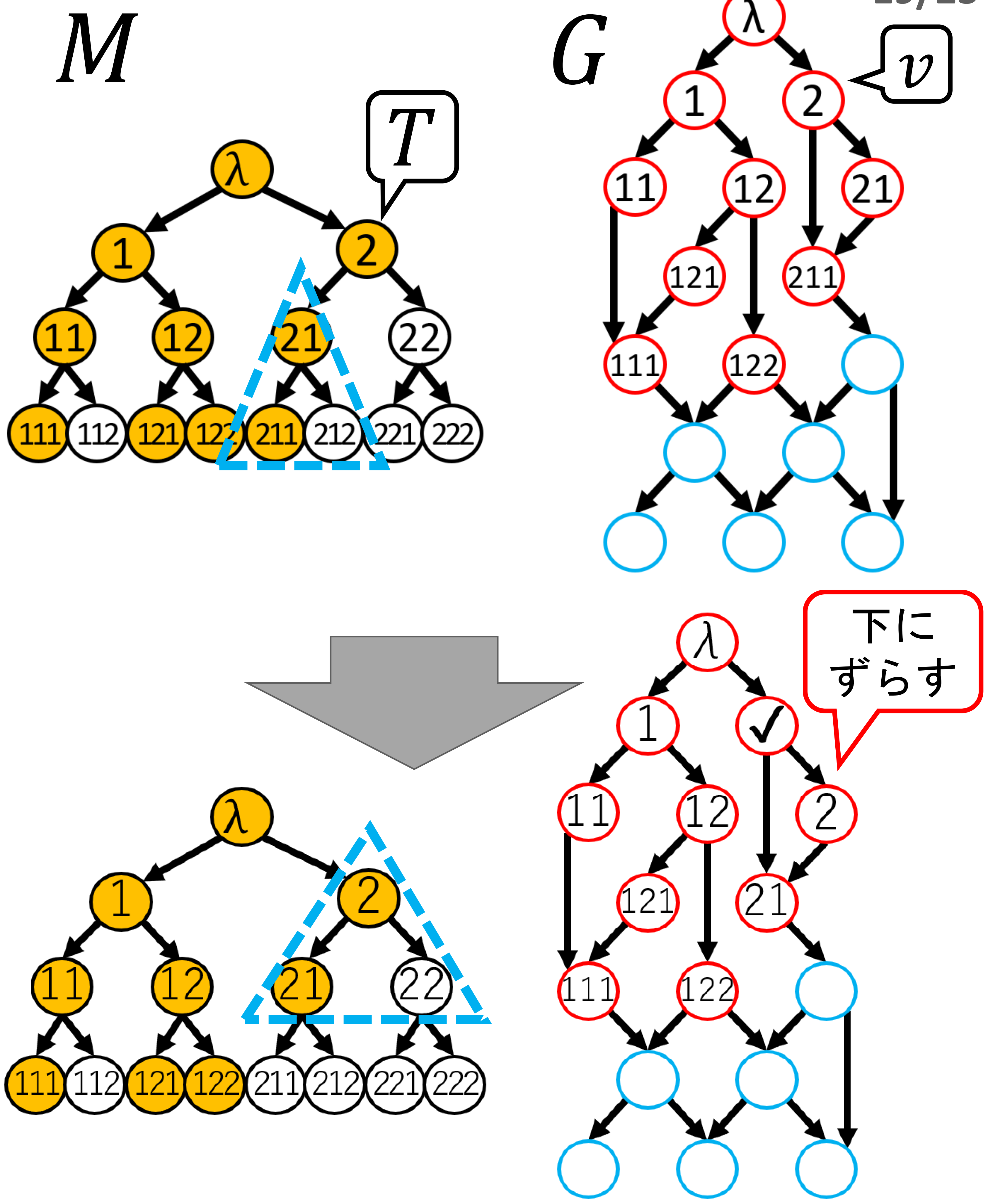
3

■ $i = s$で終了したとき

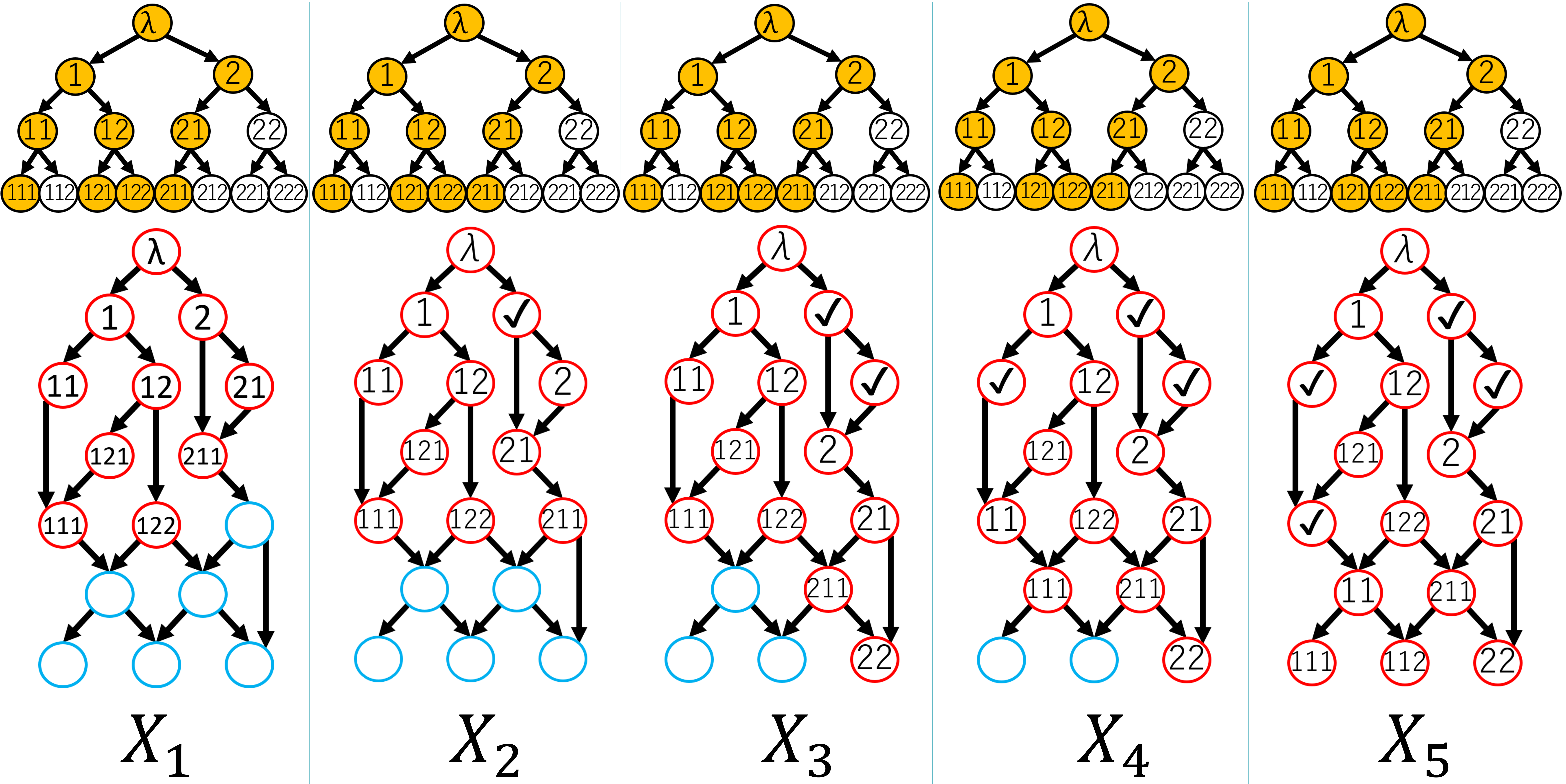
① → $(X_1, X_2, \ldots, X_s)$が幅$\boldsymbol{O(ld^k)}$のDAGパス分解

② → DAGパス幅 $> \boldsymbol{k}$

③ → DAGパス幅 $> \boldsymbol{k}$



アルゴリズムの動作例 (①幅$O(ld^k)$のDAGパス分解を出力)



先行研究[Kevin et al. 96]との違い

[Kevin et al. 96]

1. Gの任意の頂点に$token\ \lambda$を置く
2. $i=1$とし，X_i ←call $GrowTokenTree$
3. until 【Gの全頂点がred】or【$|X_i|=|V[M]|$】:
   - $token\ T\ (on\ v\in V[G'],$ 配置済の子$token$が高々1個$)$を選択
   - Tをvから取り除く
   - Tが配置済の子$T\cdot b$をもつ: $T\cdot b\cdot S$↔$T\cdot S$
   - $i=i+1$とし, X_i ←call $GrowTokenTree$

このようなTが**必ず**存在

提案アルゴリズム

1. Gの根に$token\ \lambda$を置く
2. $i=1$とし，X_i ←call $GrowTokenTree$
3. until 【Gの全頂点がred】or【$|X_i|=|V[M]|$】:
   if {ある$token\ T\ (on\ v\in V[G'])$があり, $suc(v)$が全てred, かつTの配置済の子$token$が高々1個}:
   - Tをvから取り除く
   - Tが配置済の子$T\cdot b$をもつ: $T\cdot b\cdot S$↔$T\cdot S$

   else:
   - return X_i ③

   $i=i+1$とし，X_i ←call $GrowTokenTree$

Tが存在する**とは限らない**

- 有向グラフの埋め込みは**無向グラフよりも難しい**
- $token\ T$が見つからず③で終了しても「***G*のDAGパス幅 > *k***」を示した

証明の方針

①：幅$O(ld^k)$のDAGパス分解を出力

→ X_iの列がDAGパス分解の3つのルールを満たし，かつ高々$|V[M]|$個の$token$のみ使う

②：GのDAGパス幅$> k$

→ X_sがMからGへの埋め込みになっている

③：GのDAGパス幅$> k$

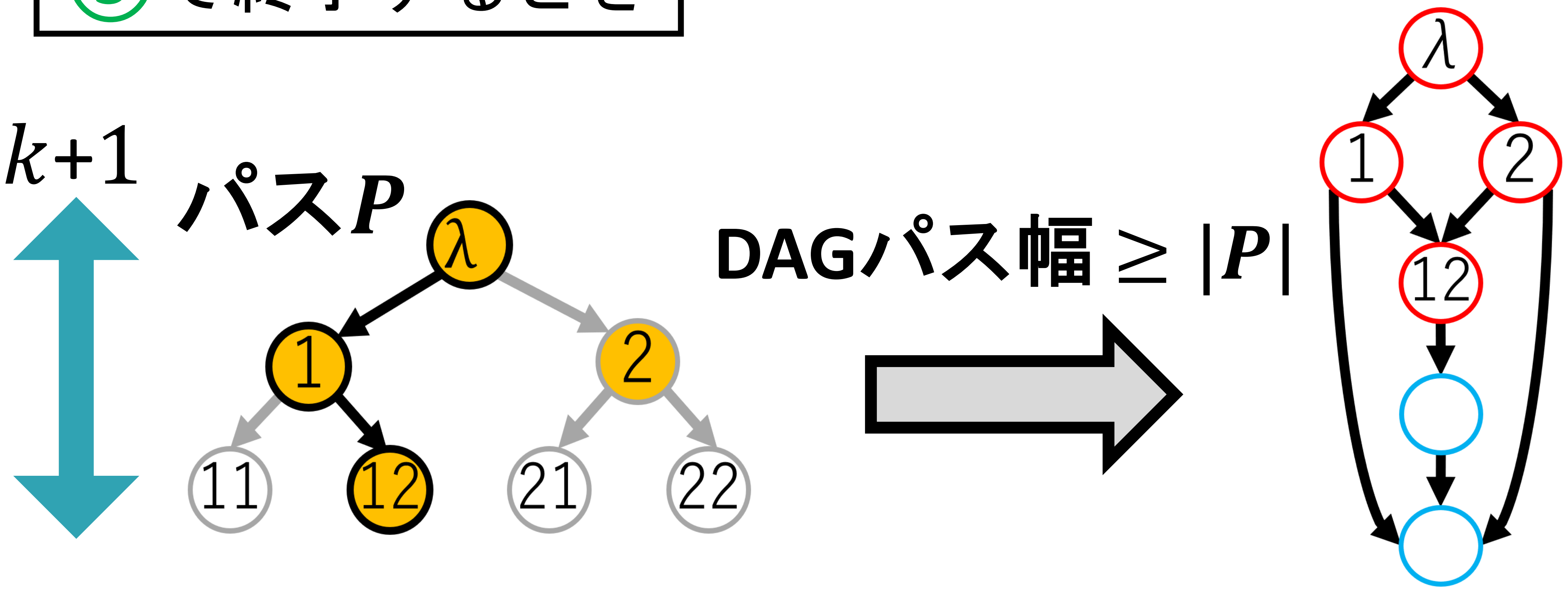
→ M上で根λから葉までの配置済の$token$のみからなる**パスP ($|P|>k+1$)**が存在し，任意のDAGパス分解は必ずあるバッグ$X' \subseteq X_s$ ($|X'| \geq |P| > k+1$)をもつ

1. Gの根に$token\ \lambda$を置く
2. $X_1 \leftarrow GrowTokenTree$. これをDAGパス分解の最初のバッグ
3. until 【Gの全頂点がred】 or 【$|X_i| = |V[M]|$】: 2
   1 if {ある$token\ T$ ($on\ v \in V[G]$)があり, $suc(v)$が全てred, かつTの配置済の子$token$が高々1個}:
      - Tをvから取り除く
      - Tが配置済の子$T \cdot b$をもつ: $T \cdot b \cdot S \leftrightarrow T \cdot S$

   else:
      - return X_i 3

   $X_{i+1} \leftarrow GrowTokenTree$. これを$i$+1番目のバッグ

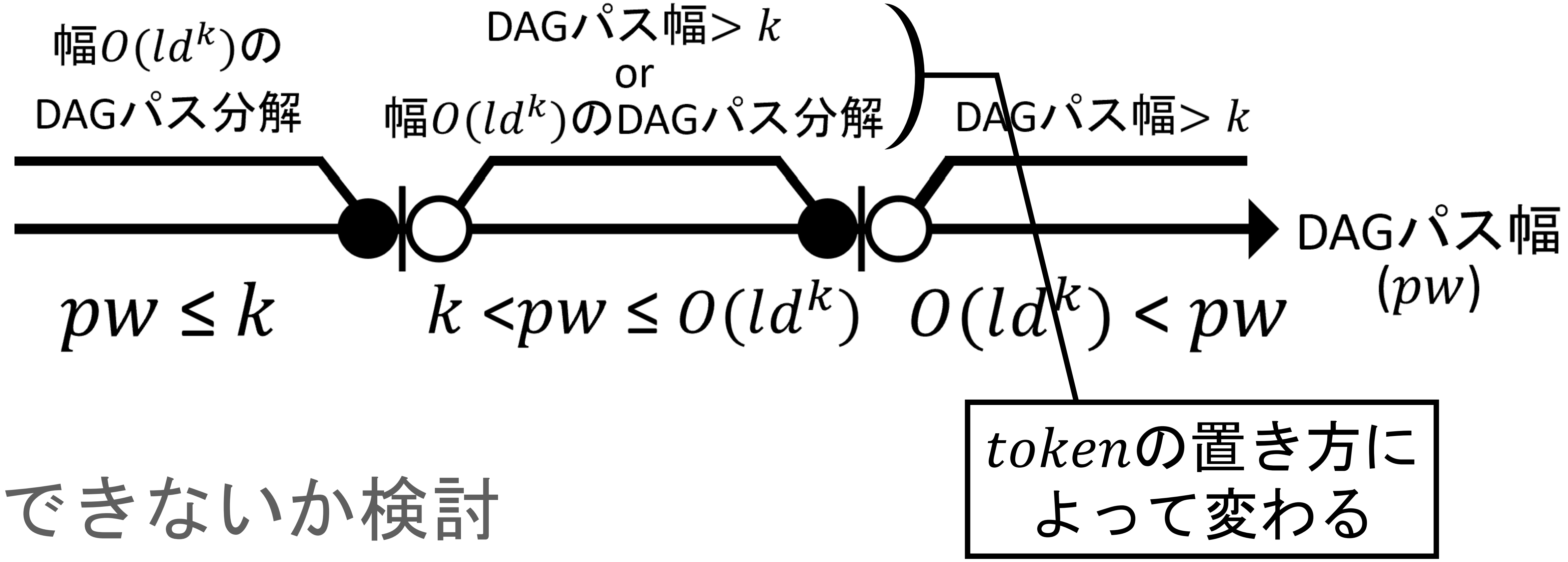
③で終了するとき



4. まとめ

ある幅のDAGパス分解を求める多項式時間アルゴリズムを初めて構築

パス幅の種類	G	$w(k)$	$f(n,k)$	参照
パス幅	無向グラフ	$O(2^k)$	$O(2^k n)$	[Kevin et al. 96]
		k	$2^{O(k^3)}n$	[Bodlaender 96]
有向パス幅	有向グラフ	k	$O(mn^{k+1})$ $(m=\lvert E(G)\rvert)$	[Tamaki 2011]
DAGパス幅	DAG	$\boldsymbol{O(ld^k)}$	$\boldsymbol{O(d^k n^2)}$	



■ 今後の課題：

・幅$O(ld^k)$のdを定数に改善できないか検討

コメント用

■コメント