

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

京都大学大学院情報学研究科 湊研究室

発表の流れ

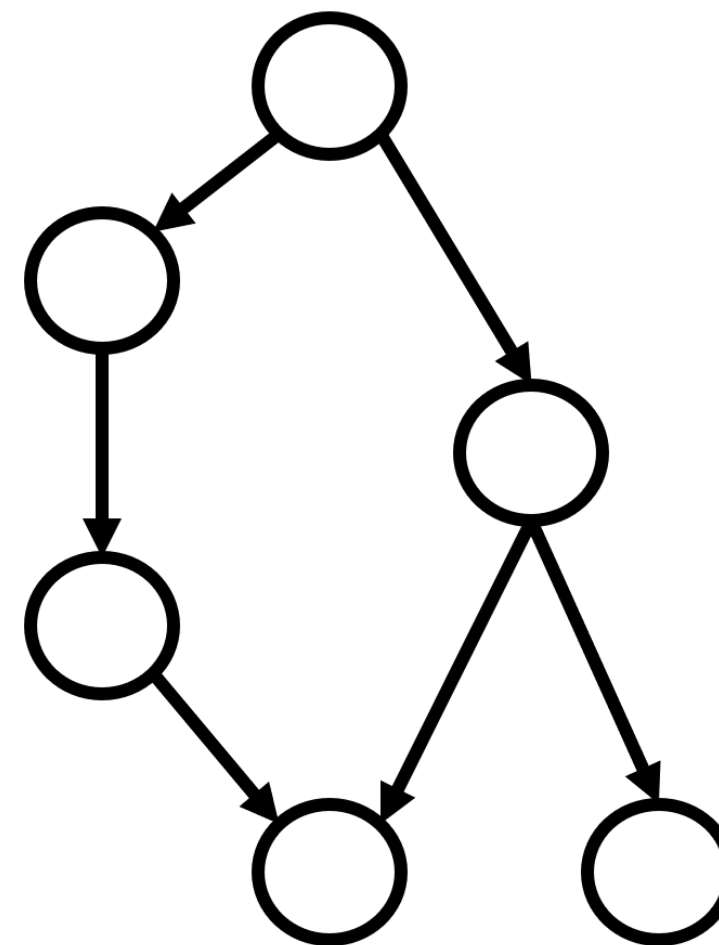
1. 研究背景
2. DAGパス分解とDAGパス幅の定義
3. 有向シュタイナー木問題への応用
4. 幅の小さなDAGパス分解を求める
アルゴリズム
5. まとめ

1. 研究背景

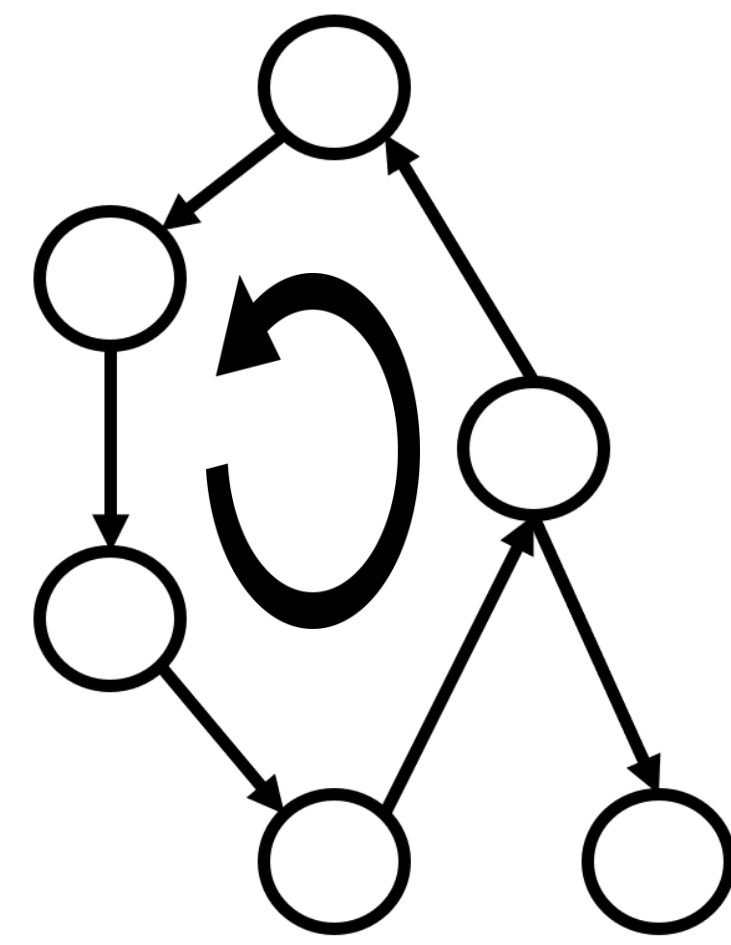
今回考えるグラフ構造【DAG】

■ **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



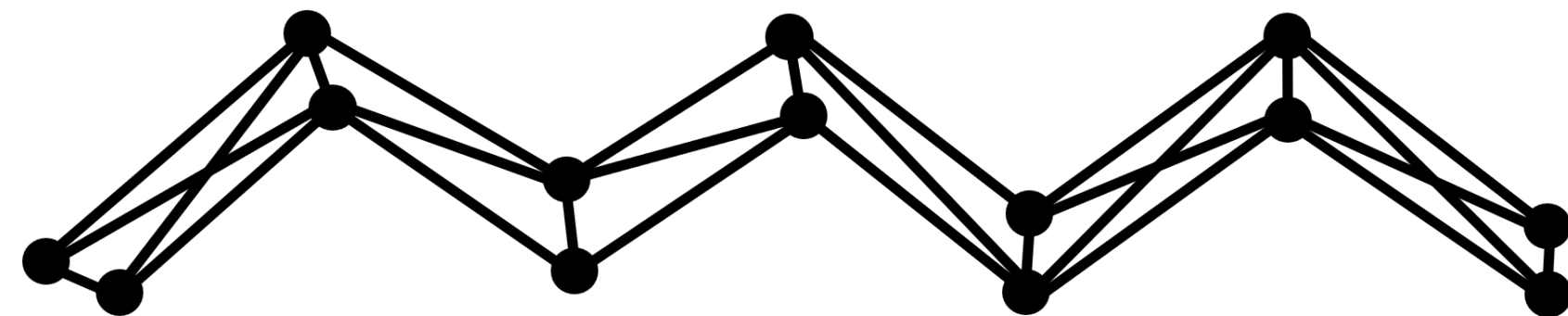
DAG



DAGでない

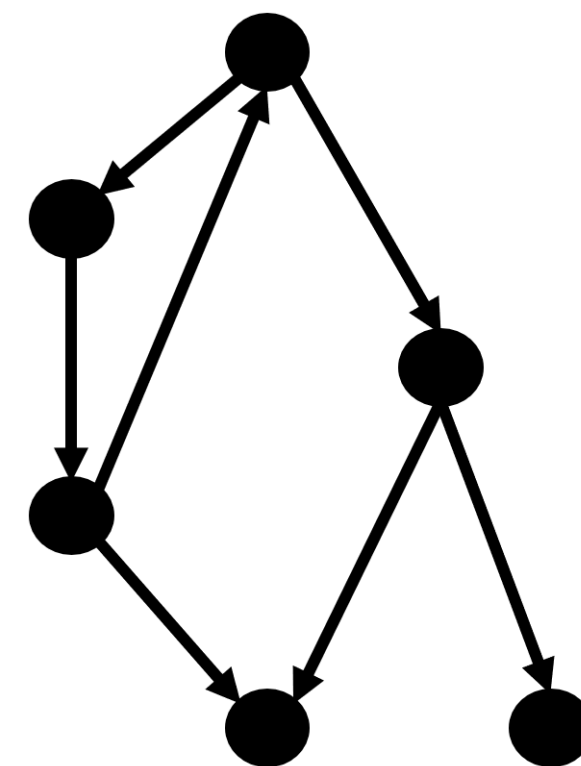
パス幅とは

- 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 小

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



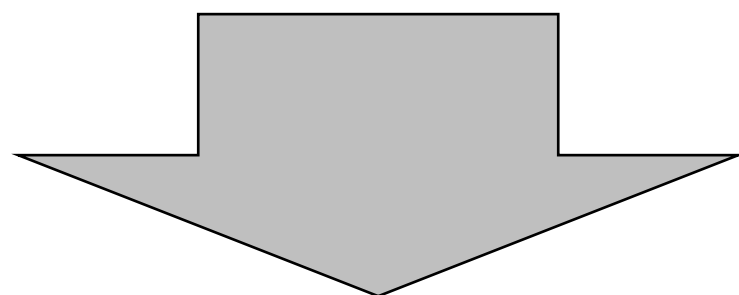
DAGに近い
→ 有向パス幅 小

- 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]

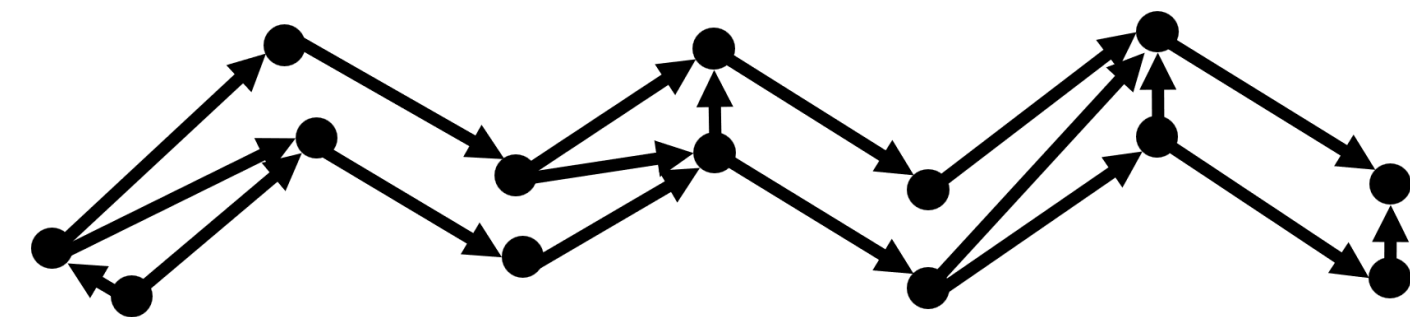
有向パス幅はDAGに対して有用でない

- 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)



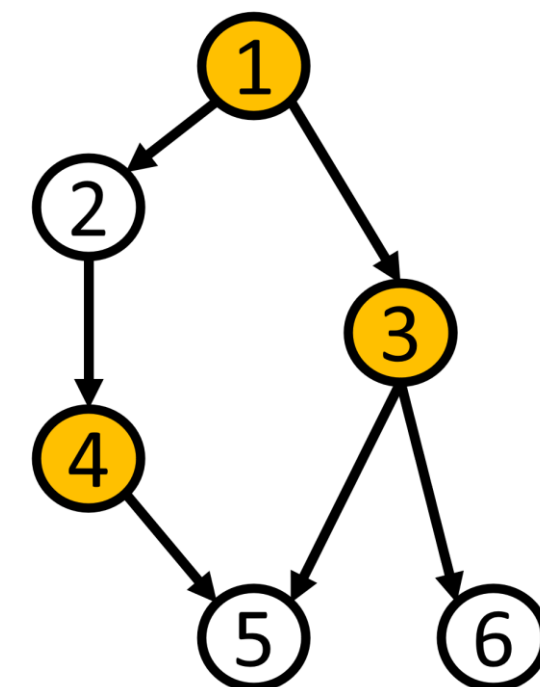
DAGにも
対応させたい...

- **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**



有向パスに近い → DAGパス幅 **小**

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】

① DAGパス幅に基づくアルゴリズム

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

③ n によらない幅を求めるアルゴリズム

幅が高々 $O(ld^k)$ 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

② DAGパス幅を求める近似アルゴリズム

$O(\log^2 n)$ -近似

→ セパレータを利用

$O(\log^{3/2} n)$ -近似

→ Pebbling gameを利用

④ DAG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 → $O(2^w w^2 n)$

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】

① DAGパス幅に基づくアルゴリズム

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 → $O((k+1)2^w n)$

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

今回説明

③ n によらない幅を求めるアルゴリズム

今回説明

幅が高々 $O(ld^k)$ 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

② DAGパス幅を求める近似アルゴリズム

$O(\log^2 n)$ -近似

→ セパレータを利用

$O(\log^{3/2} n)$ -近似

→ Pebbling gameを利用

④ DAG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

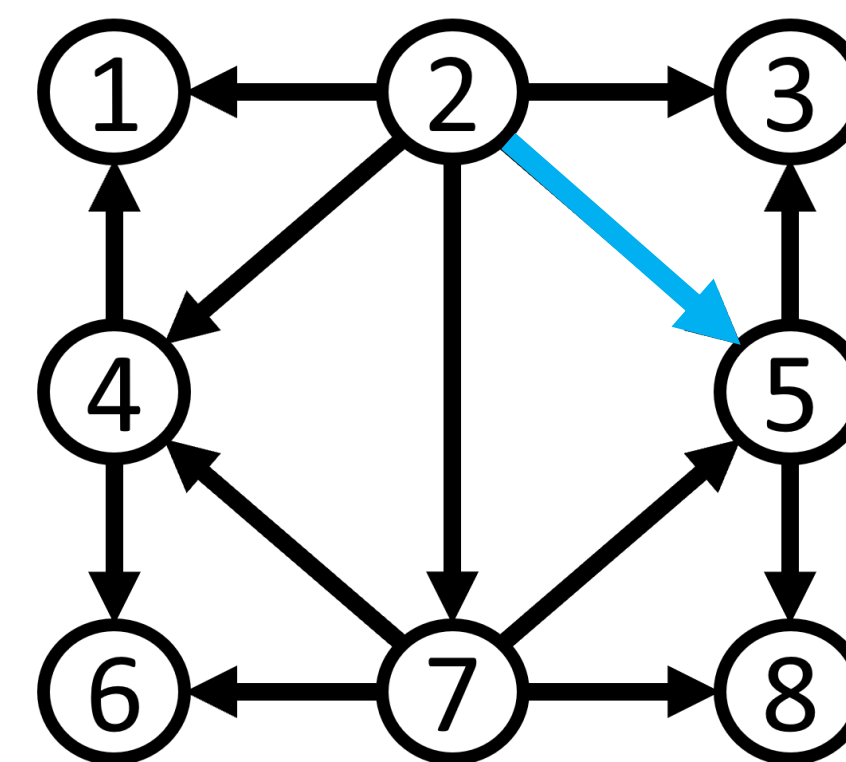
有向支配集合問題 → $O(2^w w^2 n)$

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 $X = (X_1, X_2, \dots, X_s)$.

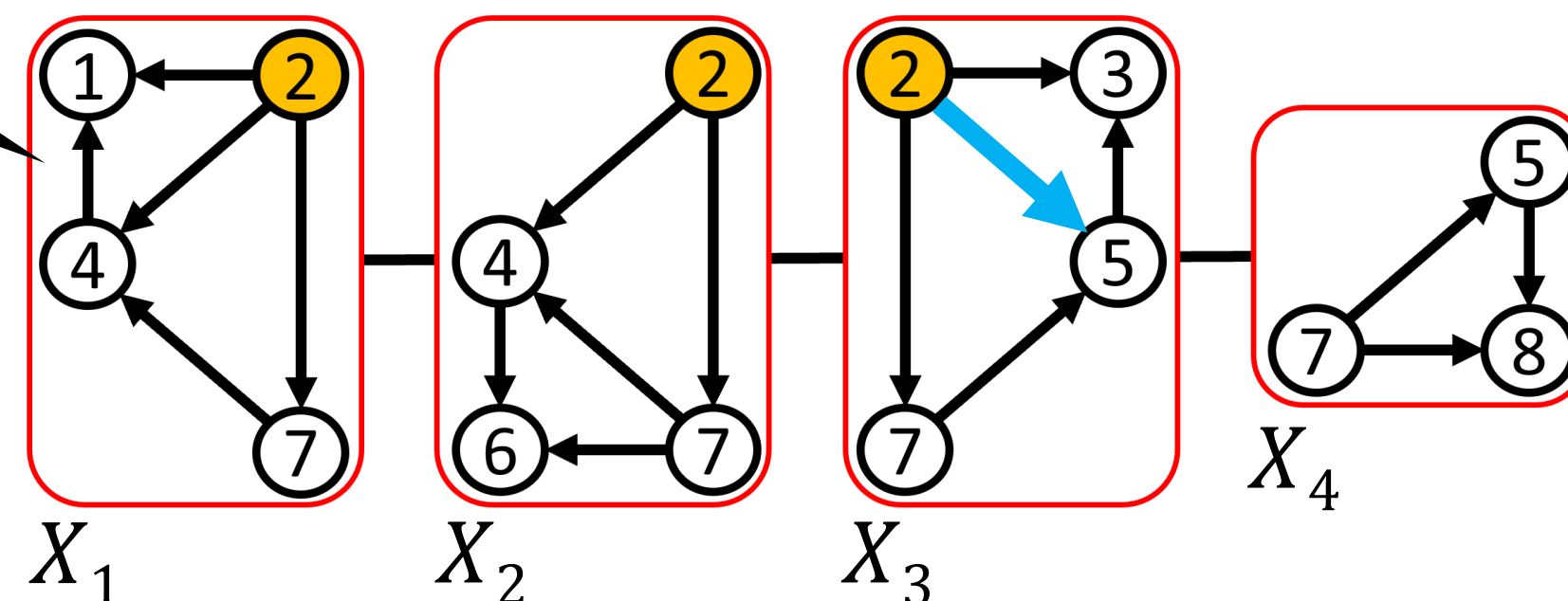
1. $X_1 \cup X_2 \cup \dots \cup X_s = V$
2. 各 $(u, v) \in E$ に対し, ある i があり, $u, v \in X_i, v \notin X_{i-1}$
3. 各 $v \in V$ に対し, v を含むバッグは連結なパスを構成する



DAGパス分解

幅=DAGパス幅=3

バッグ



■ 幅 : $\max_{1 \leq i \leq s} |X_i| - 1$

■ **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

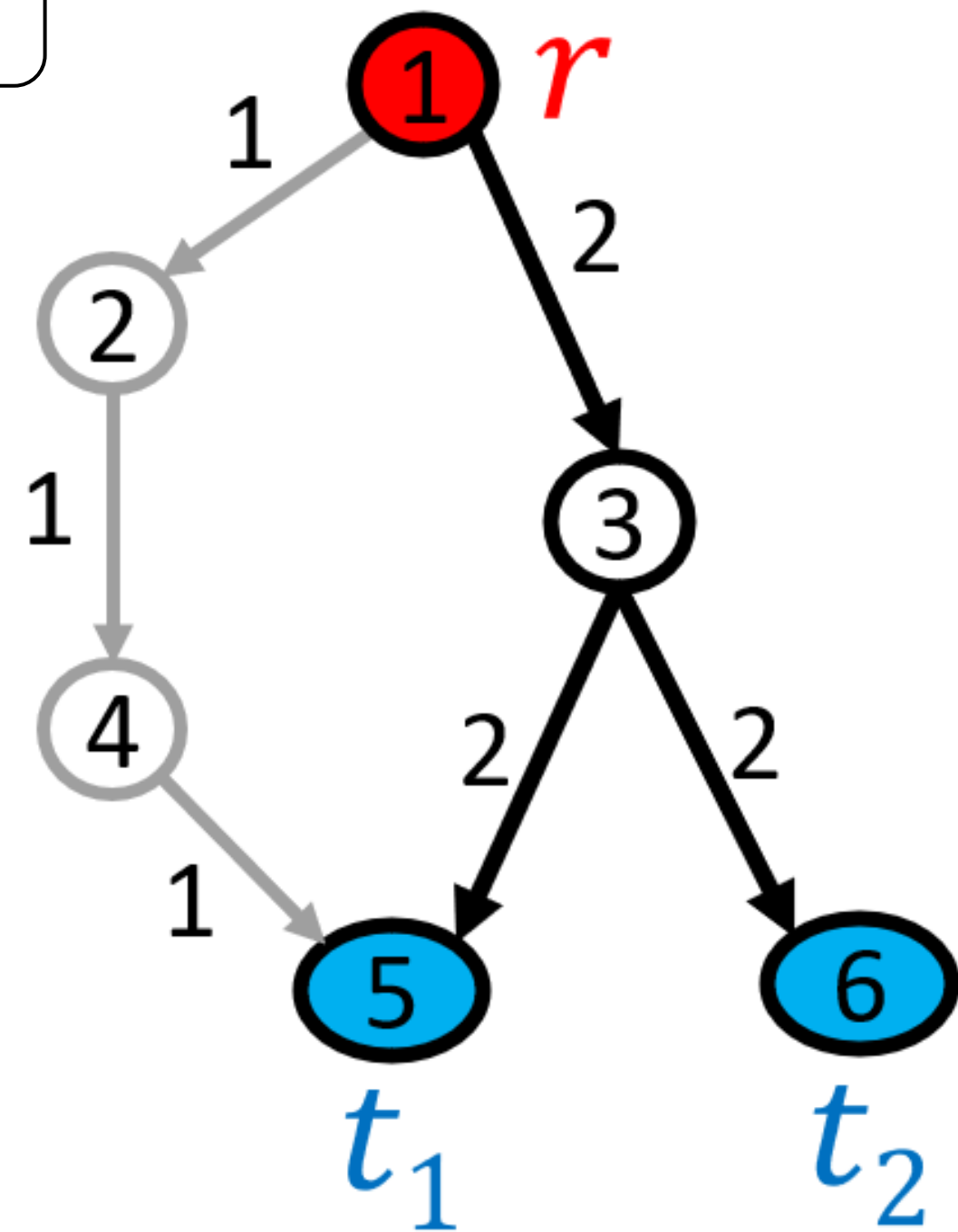
成果①

3. 有向シュタイナー木問題への応用

有向シュタイナー木問題

今回はDAG上

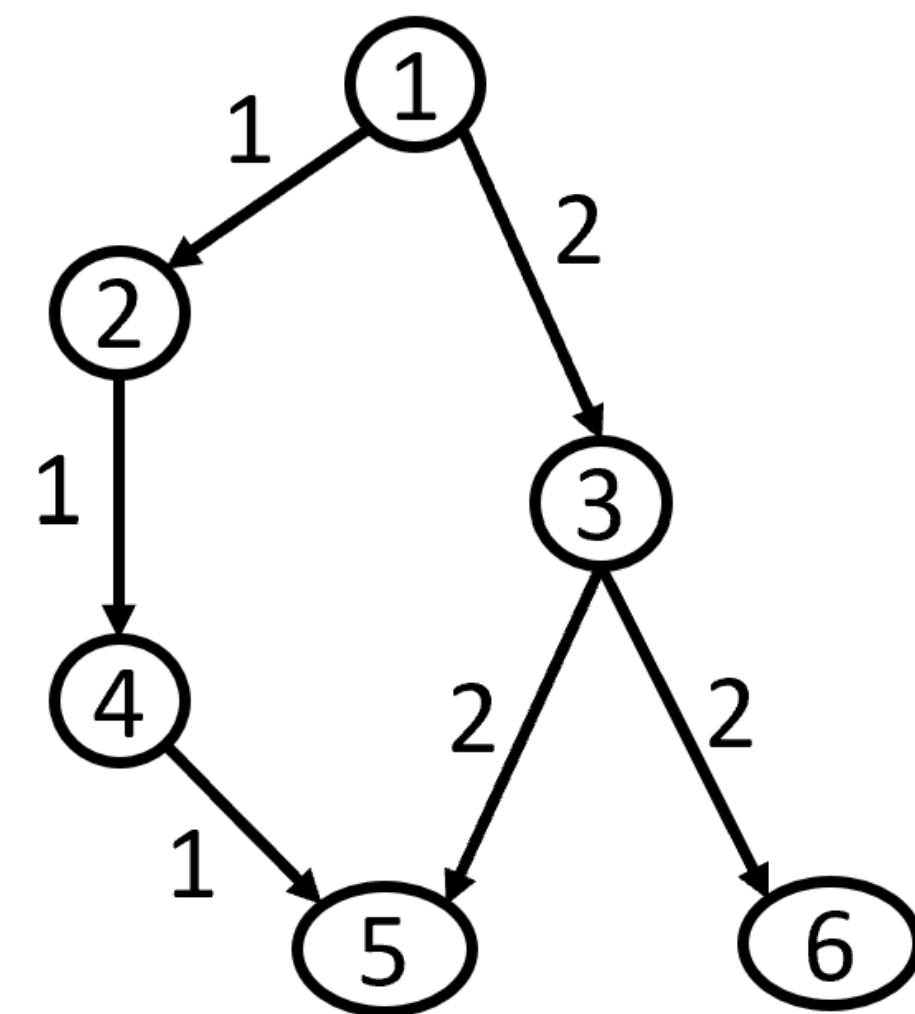
- input : 枝重み付き有向グラフ $G = (V, E)$,
 $r \in V, R = \{t_1, t_2, \dots, t_k\} \subseteq V$
- objective : r を根とし, R を含む有向木の総枝重みの最小値
- DAG上でもNP-hard [Ganian et.al 14]
- 無向パス幅を使ったFPTの先行研究は(調べた限り)行われていない



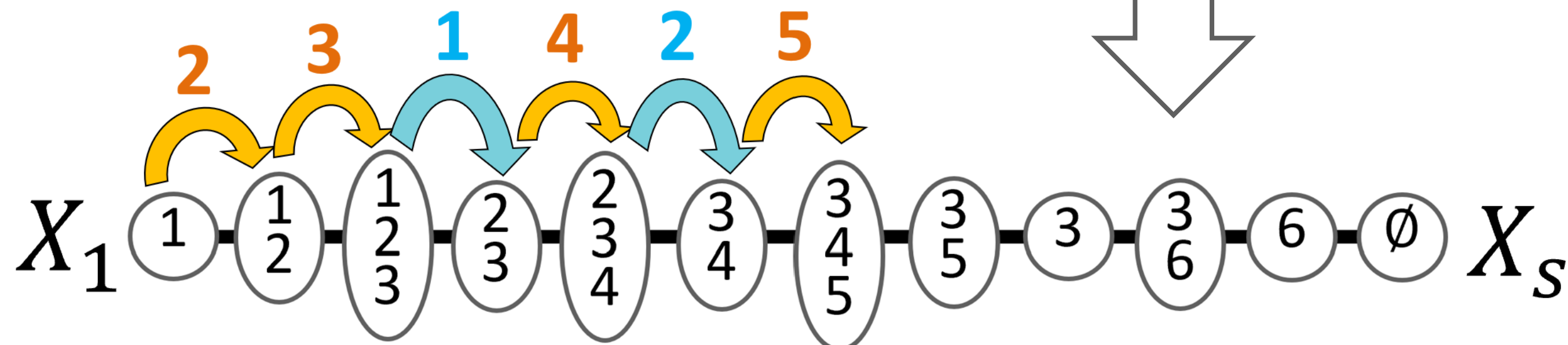
総枝重み最小のDSTの例
(総枝重み6)

アルゴリズムの大まかな動作

- 入力グラフの**DAGパス分解**が与えられているとする
- **動的計画法**を用いてバッグを左側(X_1)から順に見る
 - ・ 頂点 v が追加 $\rightarrow v$ を解に含めるか否かで**場合分け**
 - ・ 頂点 v が削除 $\rightarrow v$ を解に含めるか否かを**確定**
- 右端のバッグ(X_s)まで到達したら解が得られている



DAGパス分解



動的計画法の具体的な計算式

- $ST(i; A_i, B_i)$: 現在のバッグ X_i での最適解
- $pred(v)$: v の先行頂点集合

■ A_i (解となる頂点集合), B_i (解とならない頂点集合) を用意

■ 頂点 v が追加 $\rightarrow v$ を解 (A_i) に含めるか否か (B_i) で場合分け

頂点 v が削除 $\rightarrow v$ を解 (A_i) に含めるか否か (B_i) を確定

頂点 v が追加

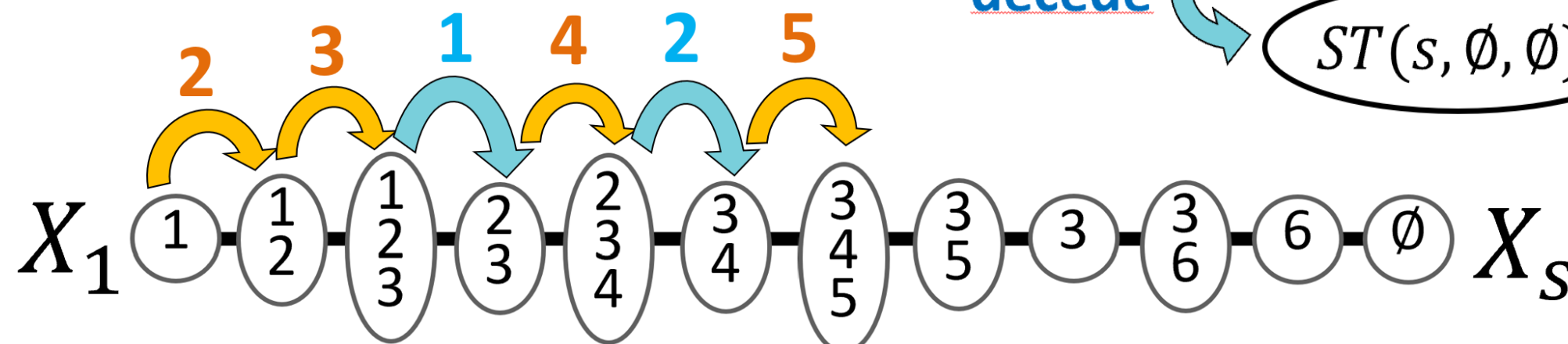
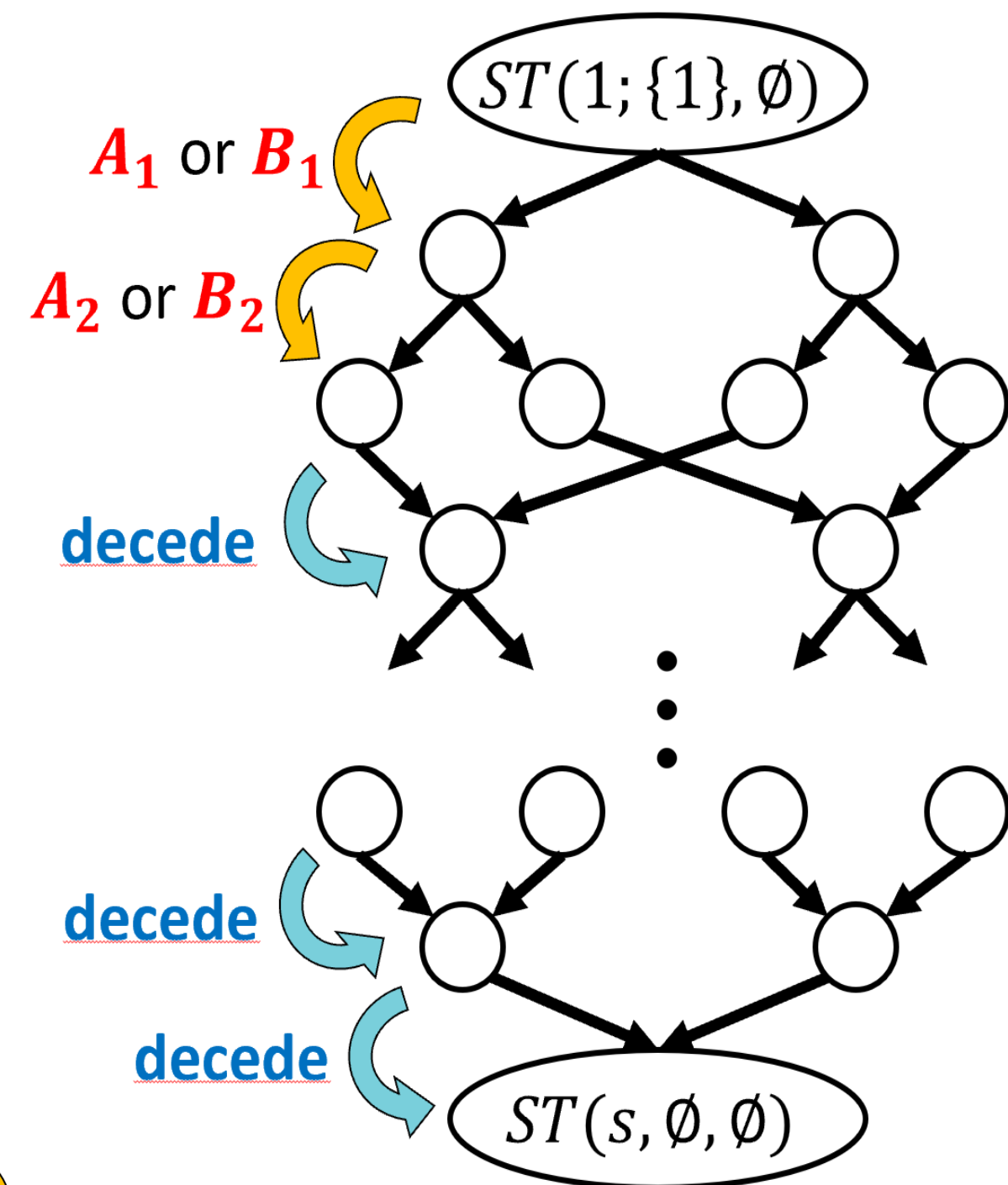
$$ST(i; A_i, B_i) = \begin{cases} ST(i-1; A_i \setminus \{v\}, B_i) + \min_{u \in pred(v) \cap A_i} d(u, v) & (v \in A_i \text{ かつ } pred(v) \cap A_i \neq \emptyset) \\ ST(i-1; A_i, B_i \setminus \{v\}) & (v \in B_i \text{ かつ } v \notin R \cup \{r\}) \\ \infty & (\text{otherwise}) \end{cases}$$

頂点 v が削除

$$ST(i; A_i, B_i) = \min \{ST(i-1; A_i \cup \{v\}, B_i), ST(i-1; A_i, B_i \cup \{v\})\}$$

■ 計算量 (w : DAG パス幅)

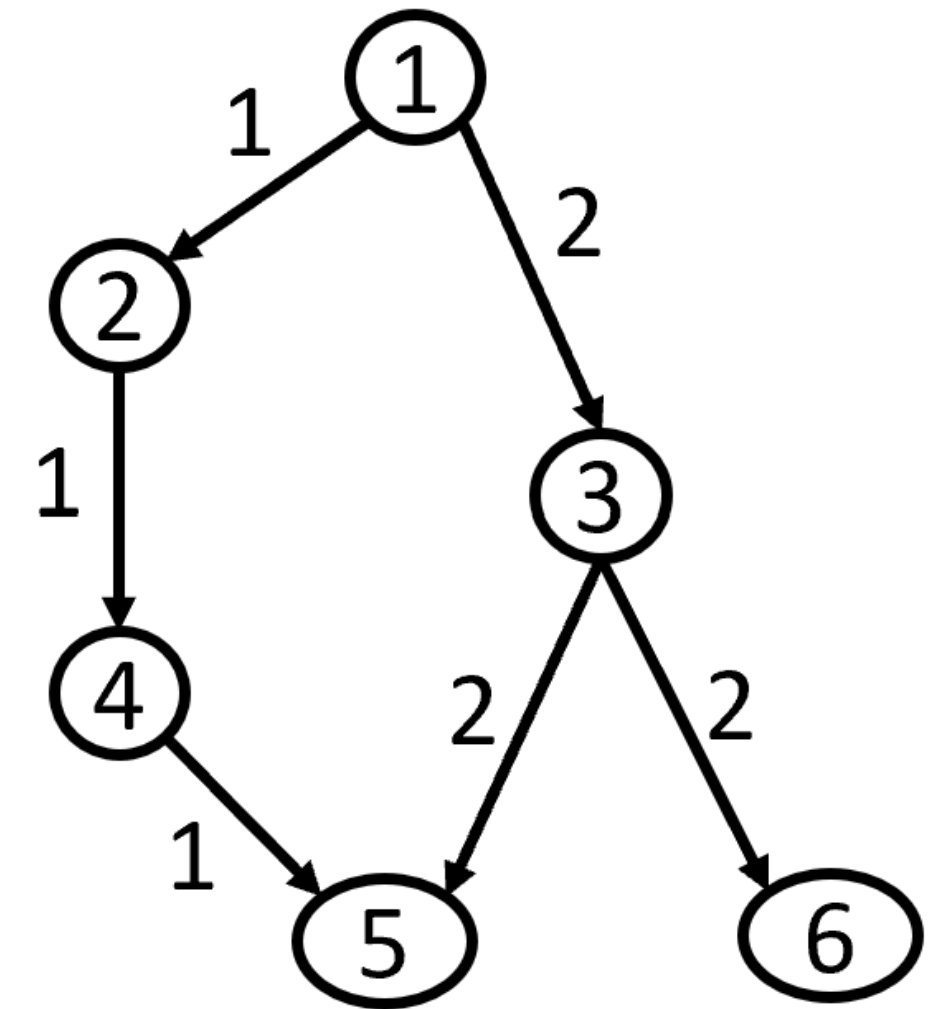
$$\rightarrow O(2^w(w + |R|)n + n^2)$$



本アルゴリズムの利点と問題点

■ 利点 :

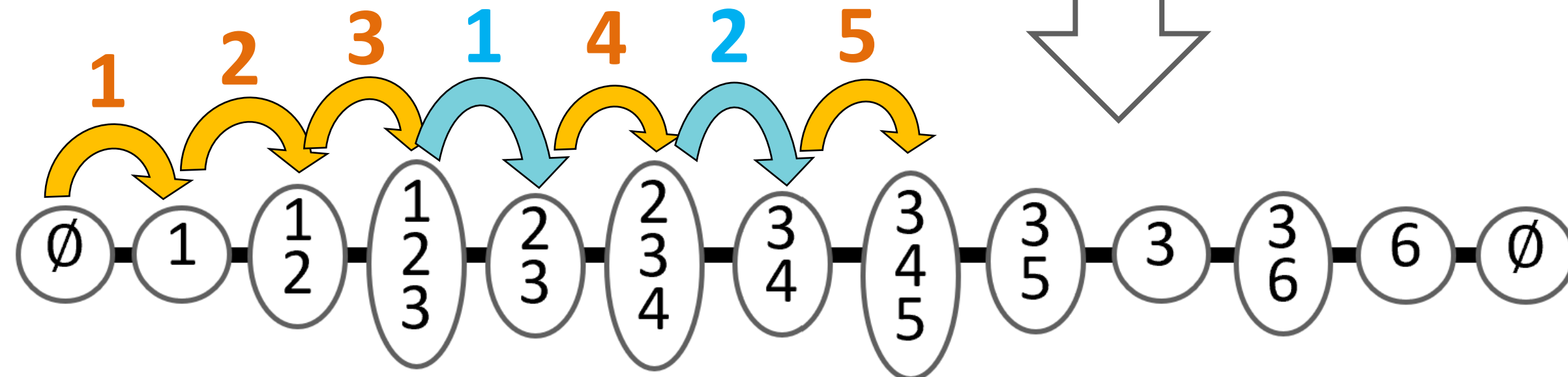
DAGパス分解が矢印の向きに沿って構成
→根から葉に向かって有向木を構築しやすい
→**アルゴリズムが単純に**



■ 問題点 :

あらかじめ幅の小さなDAGパス分解が必要
→実際は構築が難しい
→**成果③で構築**

DAGパス分解

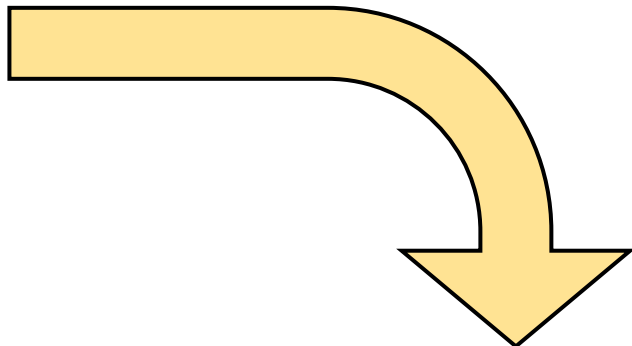


成果③

4. 幅の小さなDAGパス分解を求める アルゴリズム

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**

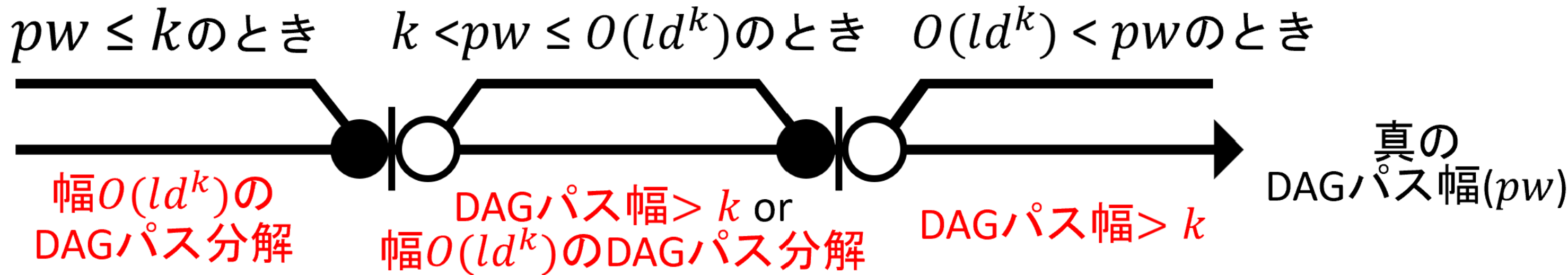


入力：頂点数 n のグラフ G , 整数 k
 出力：「パス幅 $> k$ の事実と証拠」 or 「幅が高々 $w(k)$ のパス分解」
 時間： $f(n, k)$ (n に対し多項式時間)

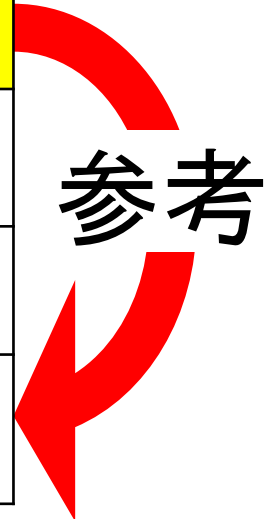
パス幅の種類	G	$w(k)$	$f(n, k)$	参照
パス幅	無向グラフ	$O(2^k)$	$O(2^k n)$	[Kevin et al. 96]
		k	$2^{O(k^3)} n$	[Bodlaender 96]
有向パス幅	有向グラフ	k	$O(mn^{k+1})$ ($m= E(G) $)	[Tamaki 2011]
DAGパス幅	<i>open</i>			

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」
のいずれか1つ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	$O(ld^k)$	$O(d^k n^2)$	



参考

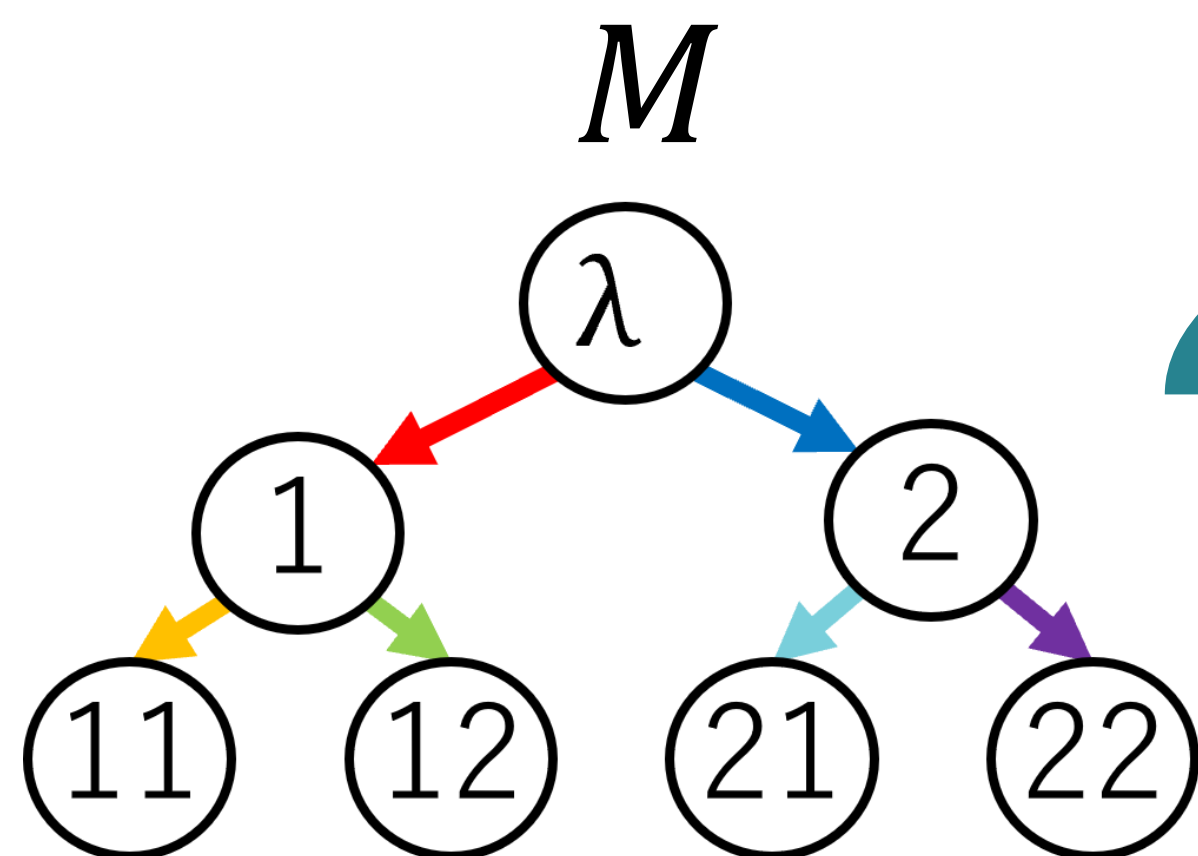
アルゴリズム構築の準備

■埋め込み (DAG $M \rightarrow$ DAG G) :

- ・ M の頂点をグラフ構造を保ったまま G の各頂点に対応付ける

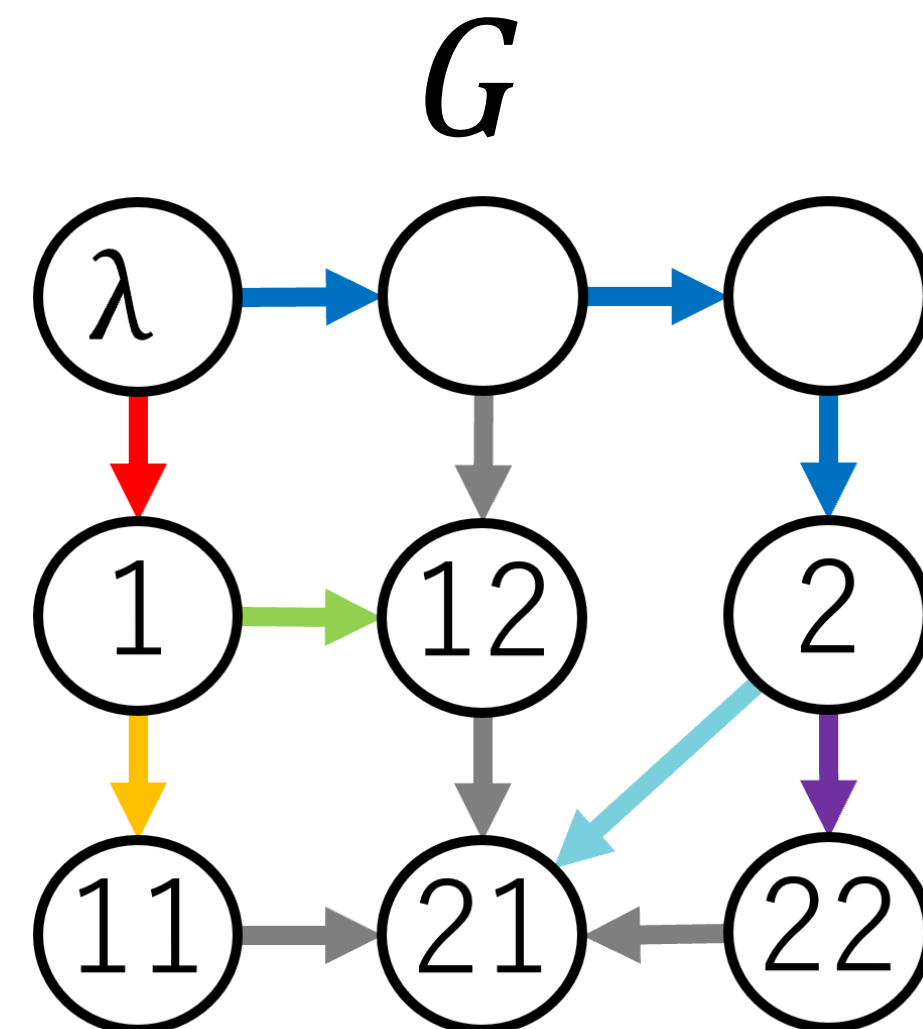
■埋め込み可能ならば...

- ・ G のDAGパス幅 $\geq M$ のDAGパス幅



埋め込み

グラフ構造を保ったまま
頂点に対応付け



アルゴリズムのアイデア

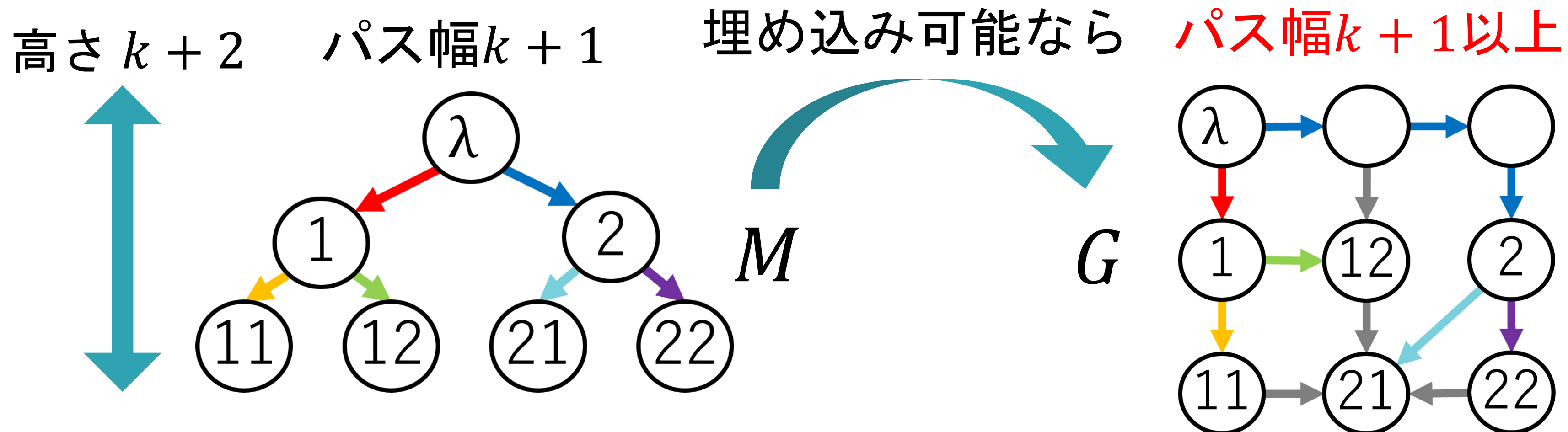
■ 入力：整数 k , DAG G (最大出次数 d)

■ M ：出次数 d , 高さ $k + 2$ の完全有向木

DAG パス幅 = $k + 1$

■ $(M \rightarrow G)$ に埋め込み可能ならば...

- G の DAG パス幅 $\geq M$ の DAG パス幅 ($= k + 1$)



アルゴリズムのアイデア

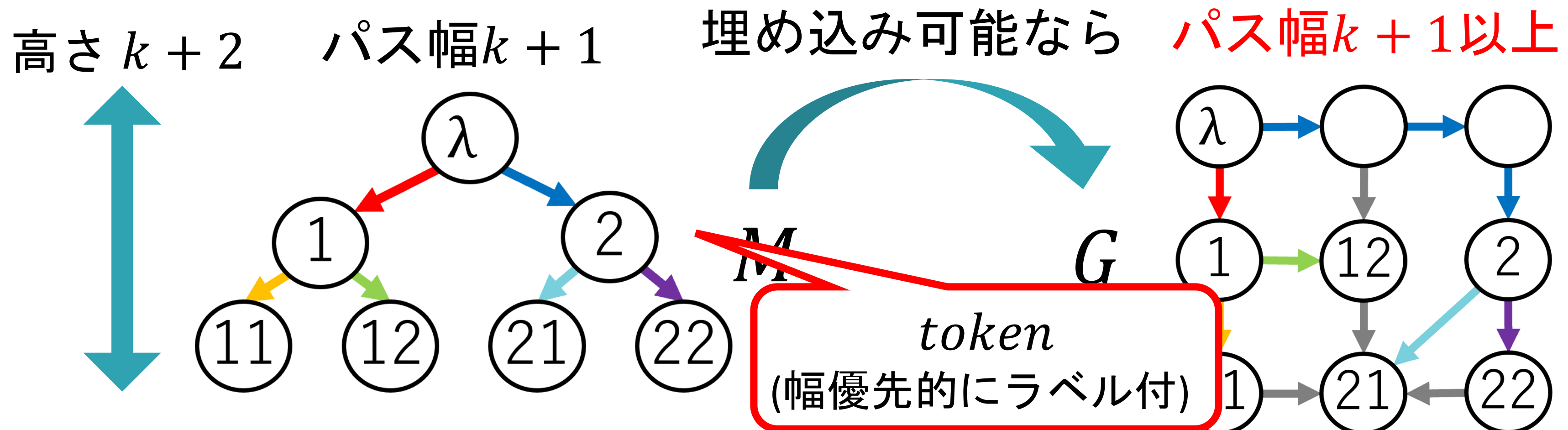
■ 入力：整数 k , DAG G (最大出次数 d)

■ M ：出次数 d , 高さ $k + 2$ の完全有向木

DAG パス幅 = $k + 1$

■ $(M \rightarrow G)$ に埋め込み可能ならば...

- G の DAG パス幅 $\geq M$ の DAG パス幅 ($= k + 1$)

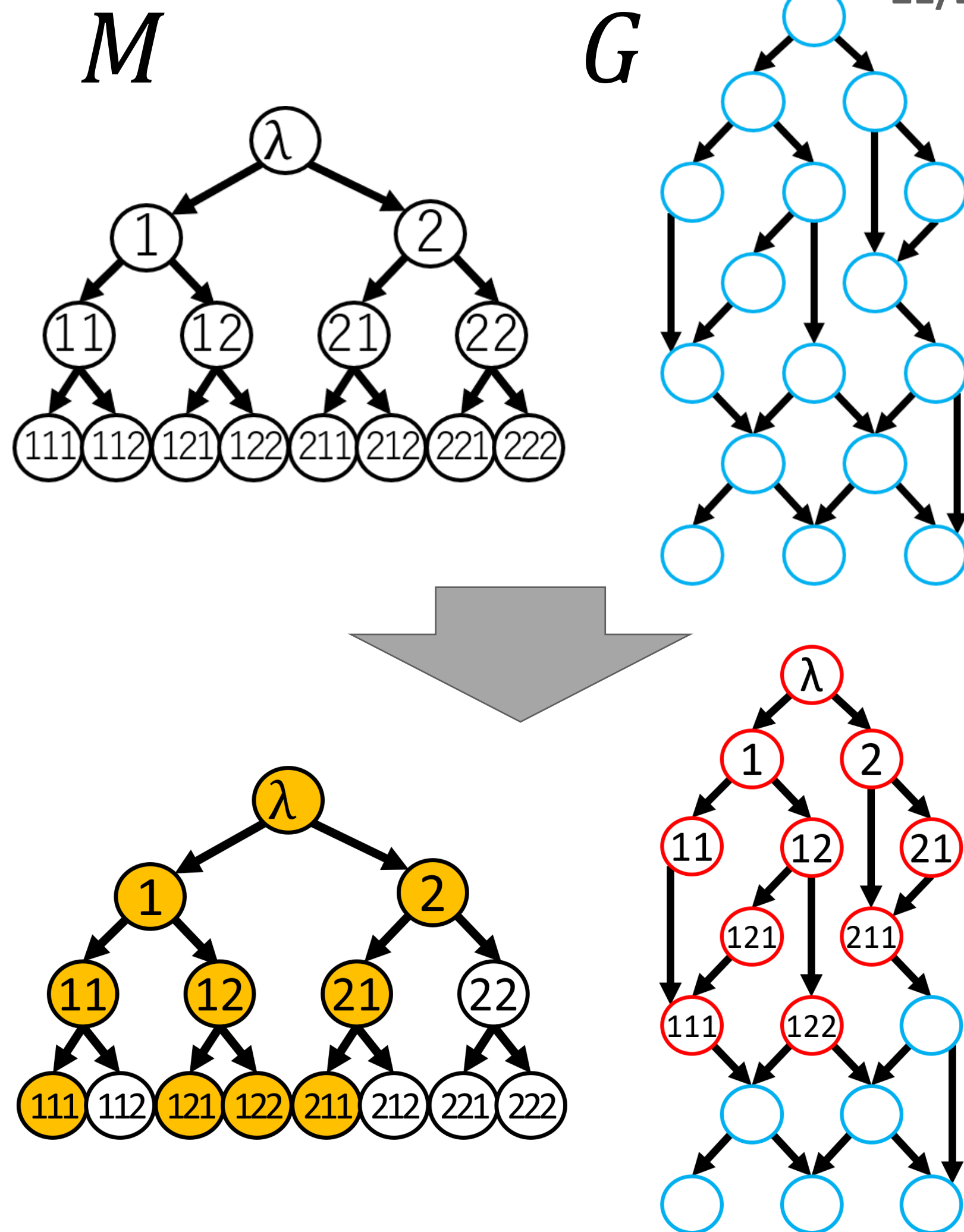


アルゴリズム1 : *GrowTokenTree*

- 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

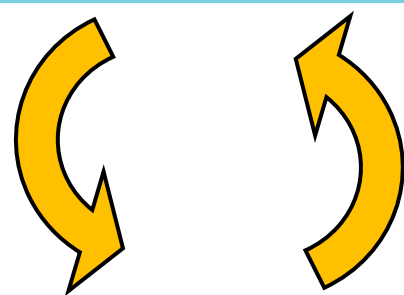
- M の木構造を保ったまま G に *token* を配置する
- *token* を配置できる頂点は, 親全てに *token* が置かれているもののみ
- 最後に *token* を配置した頂点集合を出力



アルゴリズム2 : *FindEmbedding*

FindEmbedding

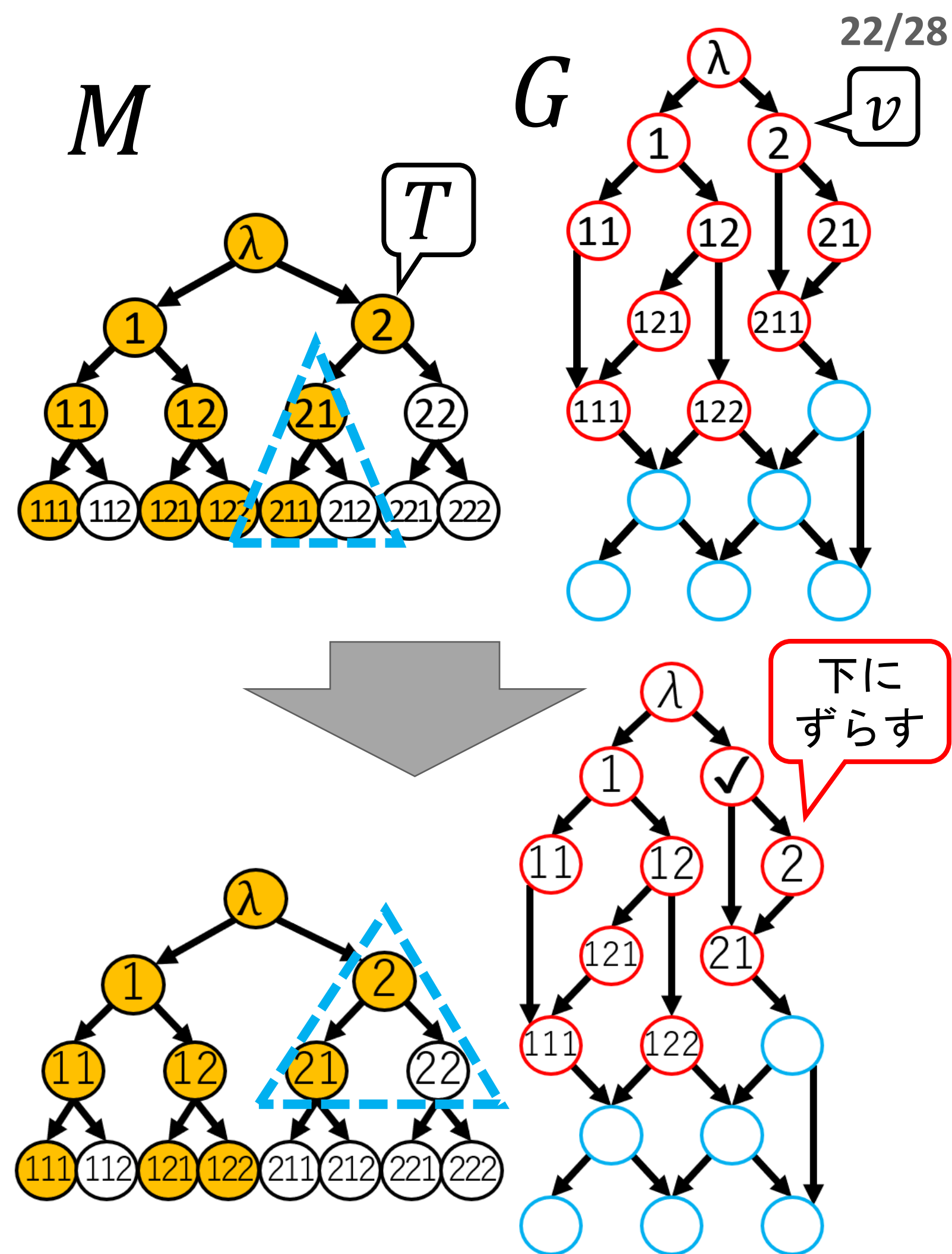
- *GrowTokenTree* を出力



- ある条件(*)を満たす頂点 v , token T を選択
- T とその子孫を v から1つずつ下にずらす

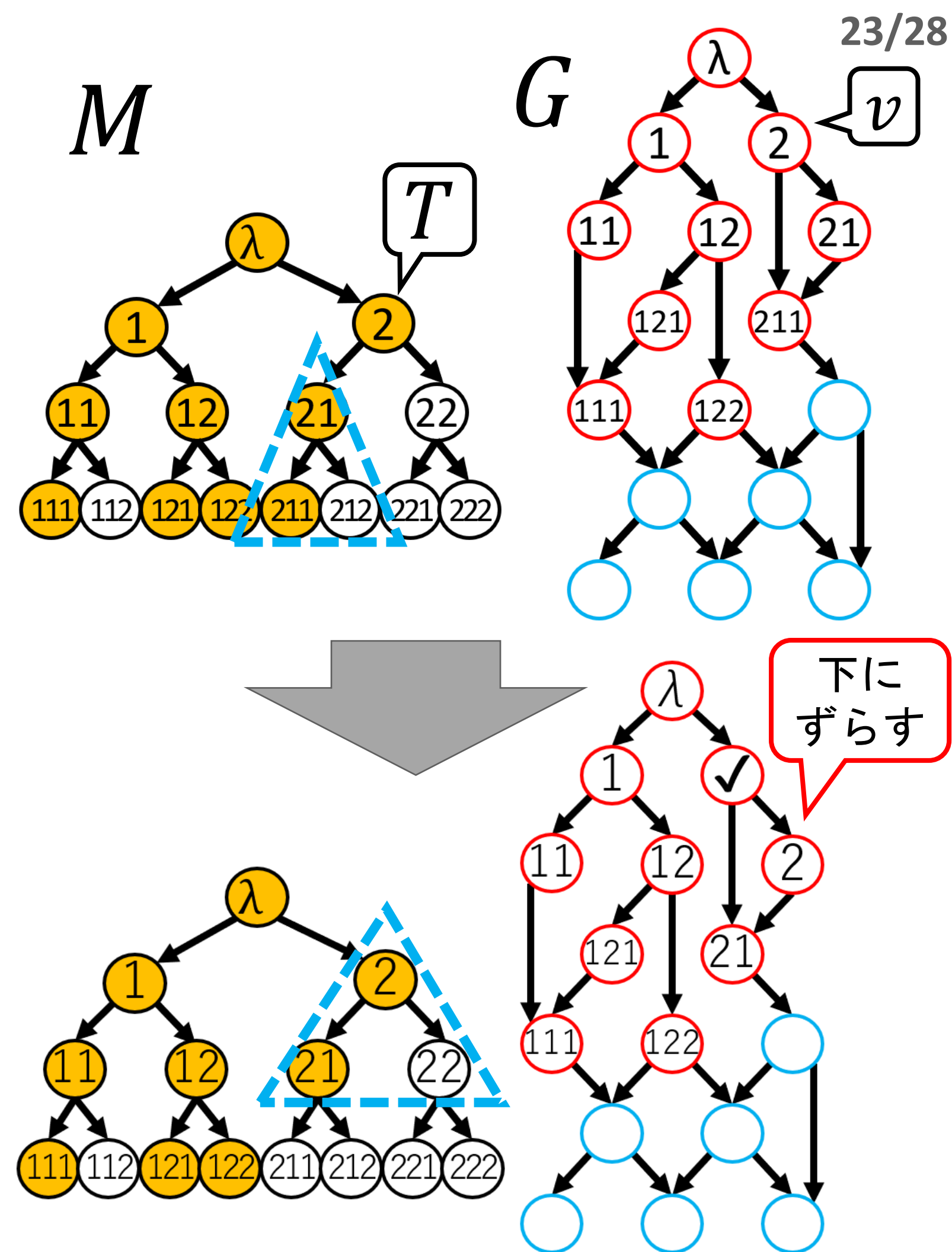
(*) v の子がすべて red & T の配置済みの子が高々1つ

- *GrowTokenTree* の出力を順に
(X_1, X_2, \dots) とすると, **G の DAG パス分解**

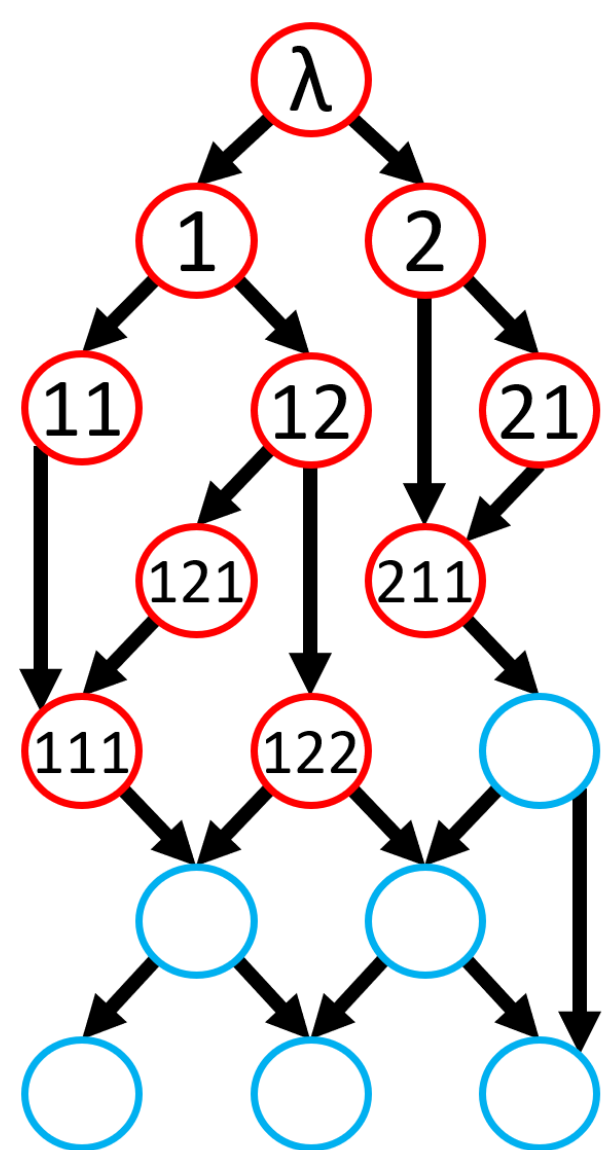
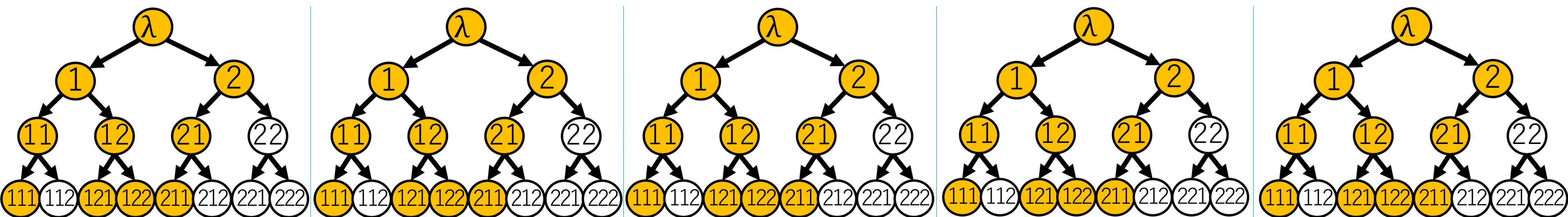
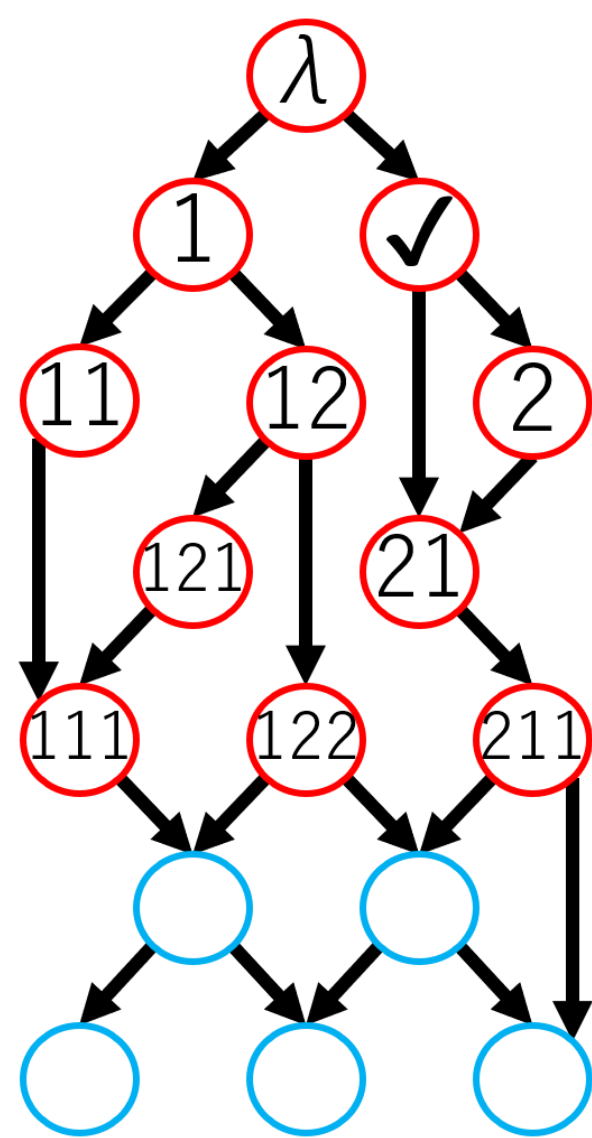
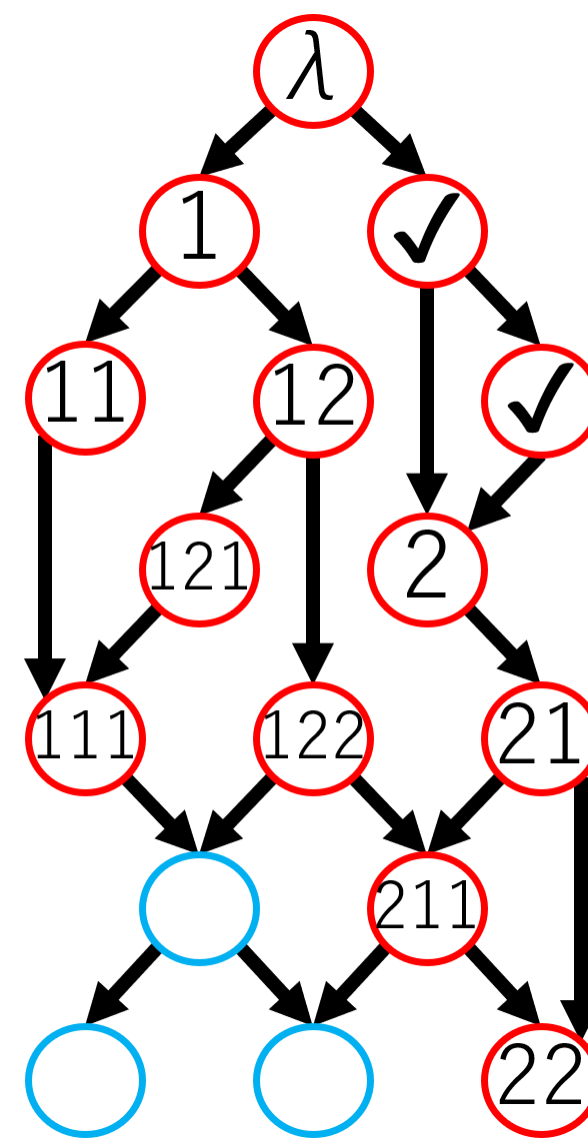
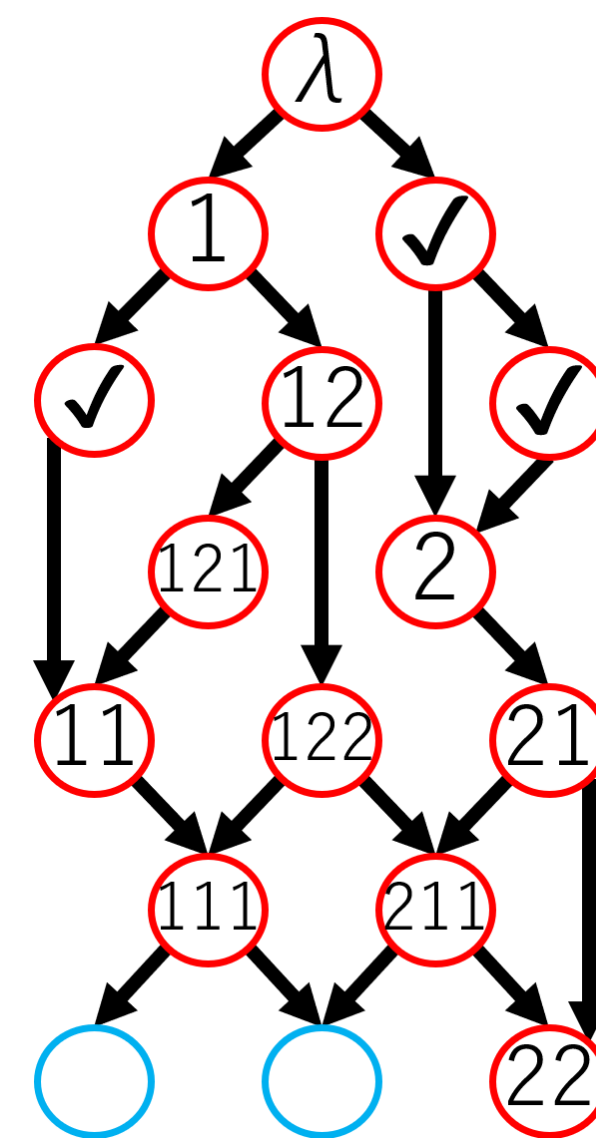
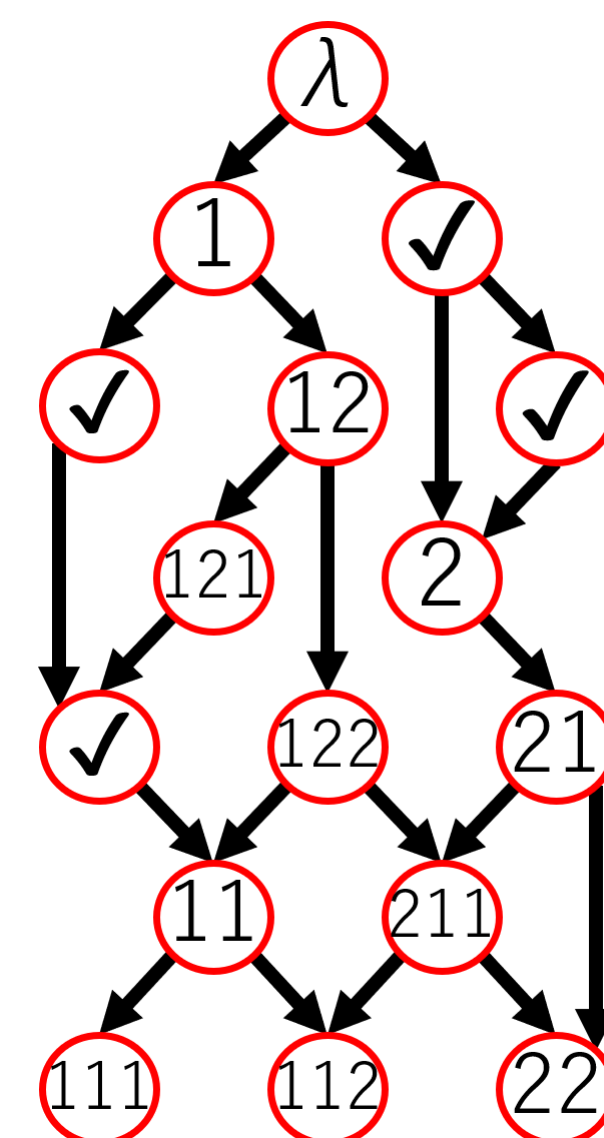


3つの終了条件とその結果

- ①途中で G がすべてred
→ (X_1, X_2, \dots) が幅 $O(ld^k)$ のDAGパス分解
- ②途中で M の全 $token$ を使い切る
→ G のDAGパス幅 $> k$
- ③途中で条件(*)を満たす T がなくなる
→ G のDAGパス幅 $> k$



アルゴリズムの動作例 (①幅 $O(ld^k)$ のDAGパス分解を出力)


 X_1

 X_2

 X_3

 X_4

 X_5

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 λ を置く
2. $i = 1$ とし, $X_i \leftarrow \text{call } \text{GrowTokenTree}$
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:

$$T \cdot b \cdot S \leftrightarrow T \cdot S$$
 - $i = i + 1$ とし, $X_i \leftarrow \text{call } \text{GrowTokenTree}$

このような T
が必ず存在

T が存在する
とは限らない

提案アルゴリズム

1. G の根にtoken λ を置く
2. $i = 1$ とし, $X_i \leftarrow \text{call } \text{GrowTokenTree}$
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, $\text{suc}(v)$ 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
 - $i = i + 1$ とし, $X_i \leftarrow \text{call } \text{GrowTokenTree}$

■ 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 $> k$

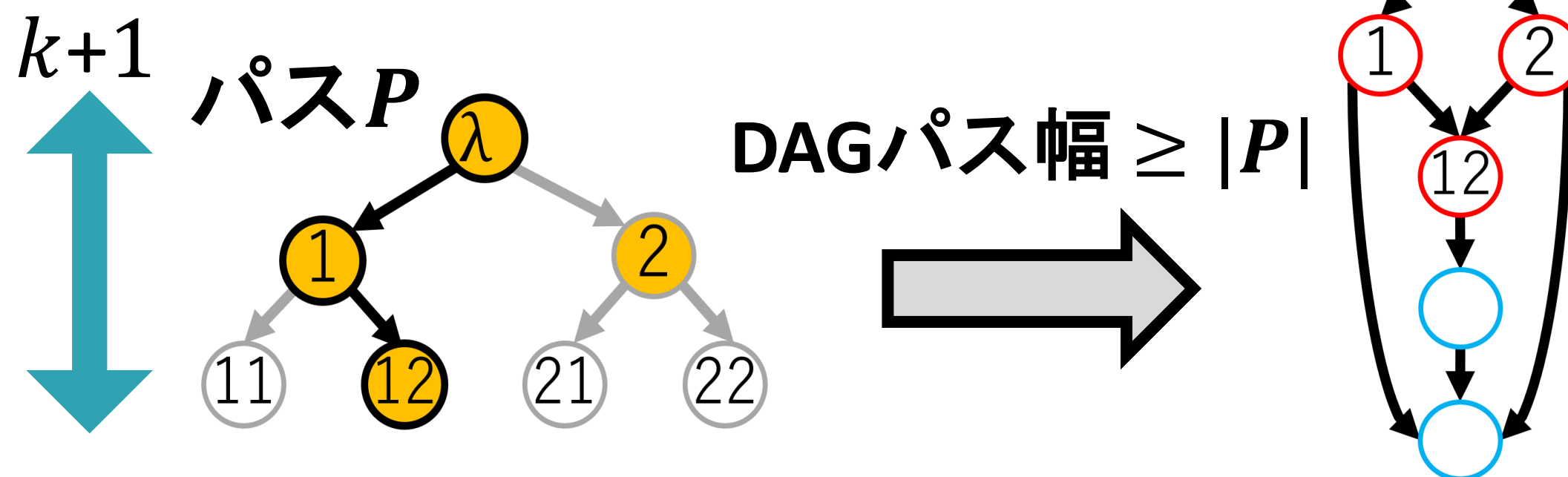
→ X_s が M から G へ の 埋め込み になっている

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* λ を 置く
 2. $X_1 \leftarrow \text{GrowTokenTree}$. 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, $\text{suc}(v)$ 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:
 - ③ return X_i
- $X_{i+1} \leftarrow \text{GrowTokenTree}$. これ を $i+1$ 番 目 の バッグ

③ で 終 了 す る と き



4. まとめ

DAGパス幅に関する種々のアルゴリズムを設計した

① DAGパス幅に基づくアルゴリズム

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

③ n によらない幅を求めるアルゴリズム

幅が高々 $O(ld^k)$ 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

② DAGパス幅を求める近似アルゴリズム

$O(\log^2 n)$ -近似

→ **セパレータ**を利用

$O(\log^{3/2} n)$ -近似

→ **Pebbling game**を利用

④ DAG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 → $O(2^w w^2 n)$

■ 今後の課題

- ・ 幅 $O(ld^k)$ の d を定数に改善できないか検討
- ・ DAG木幅の利用が適した問題の特徴づけ

コメント用

■コメント