離散最適化基礎論

ジョブ・スケジューリングのアルゴリズム

第 9 回

先行制約：他の半順序

岡本 吉央　(電気通信大学)

okamotoy@uec.ac.jp

2024 年 12 月 17 日

最終更新：2024 年 12 月 16 日　20:43

スケジュール (前半)

1. スケジューリング問題の分類 (10/1)
 * 休み (出張) (10/8)
 * 休み (体育祭) (10/15)
2. 整列による解法 (10/22)
3. 動的計画法 (10/29)
4. NP 困難性と計算量の分類 (11/5)
5. 計算複雑性による問題の分類 (11/12)
6. リスト・スケジューリング (11/19)

スケジュール (後半)

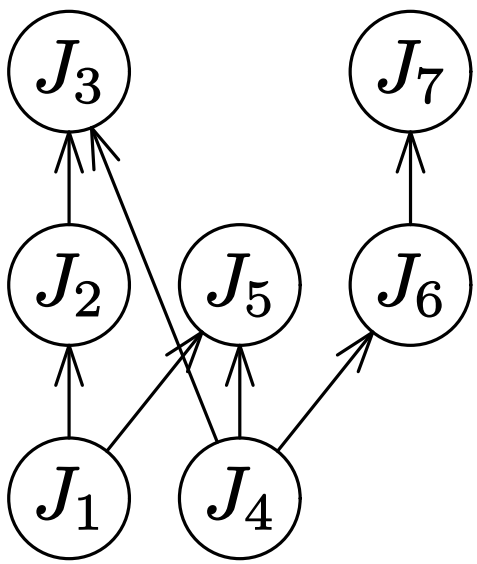
7.	先行制約：基礎	(11/26)
＊	休み (秋ターム試験)	(12/3)
8.	先行制約：多機械	(12/10)
9.	**先行制約：他の半順序**	(12/17)
10.	ショップ・スケジューリング：基礎	(12/24)
＊	休み (冬季休業)	(12/31)
11.	ショップ・スケジューリング：機械数が定数	(1/7)
12.	ショップ・スケジューリング：機械数が可変	(1/14)
13.	近似可能性と近似不可能性	(1/21)
14.	多項式時間近似スキーム	(1/28)
＊	なし	(2/4)

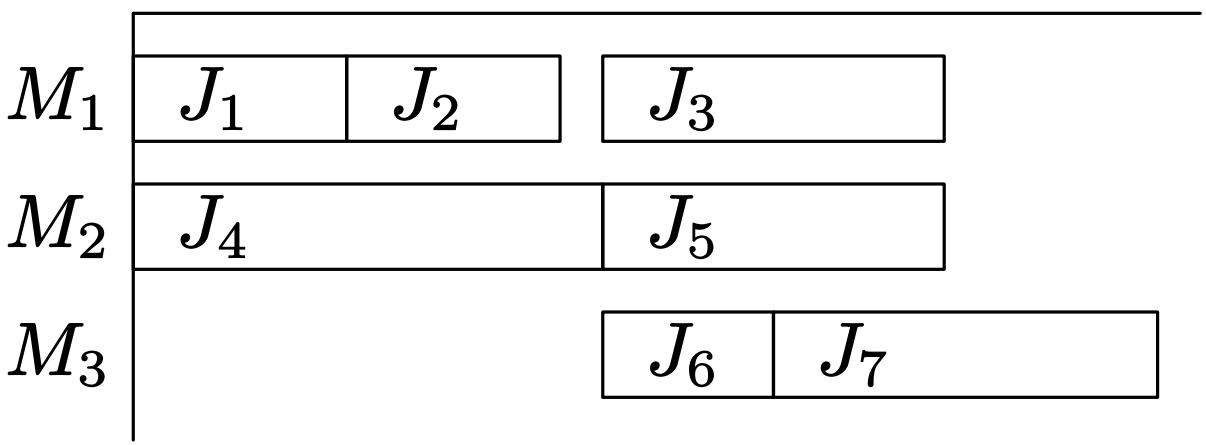
復習：先行制約

ジョブの特性の１つ：先行制約 (precedence constraint)

ジョブの集合 $\mathcal{J}$ 上の半順序 $\rightarrow$ を使って
J_j の処理が完了しないと $J_{j'}$ の処理を開始できない
ことを $J_j \rightarrow J_{j'}$ で表す

先行制約があるとき，β に「prec」と書く　　(precedence)

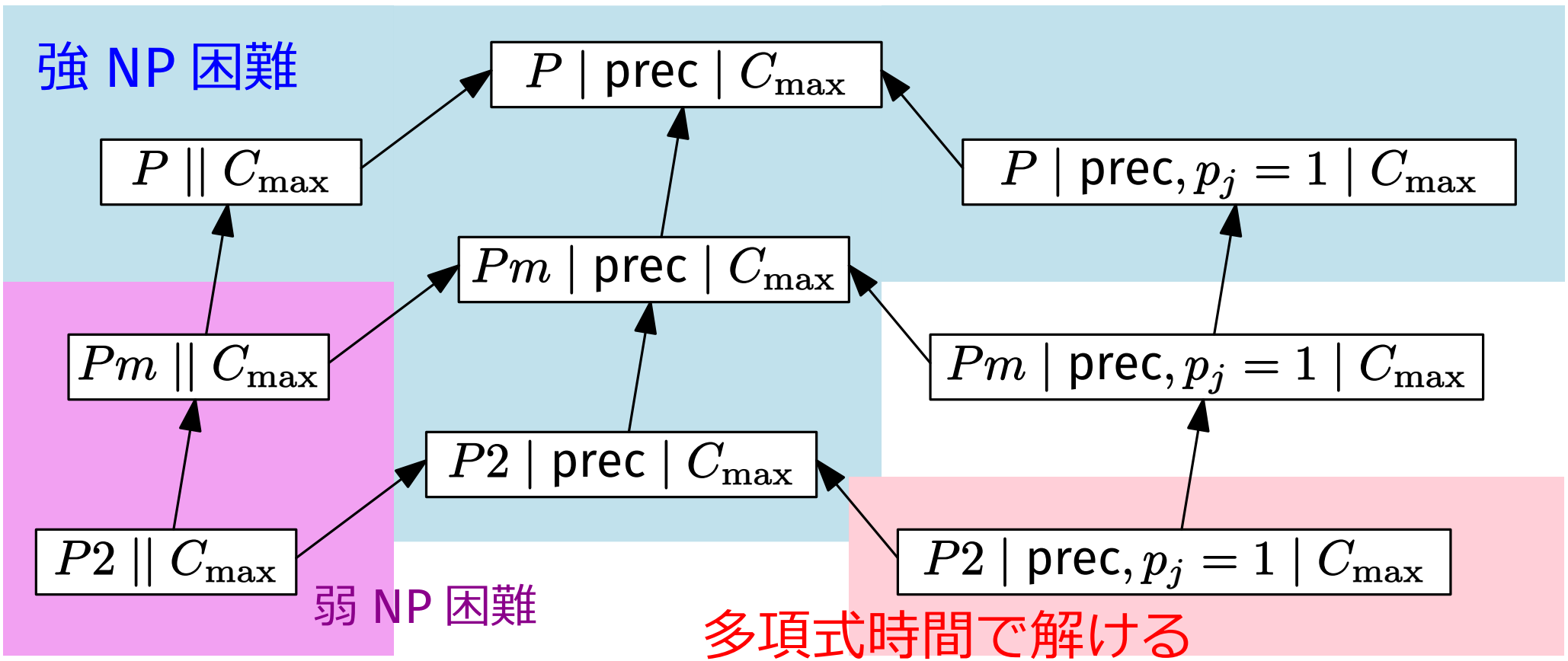




復習：$Pm \mid \text{prec} \mid C_{\max}$ の難しさ

定理 (今日の内容)

- 問題 $P2 \mid \text{prec} \mid C_{\max}$ は強 NP 困難
- 問題 $P \mid \text{prec}, p_j = 1 \mid C_{\max}$ は (強) NP 困難
- 問題 $P2 \mid \text{prec}, p_j = 1 \mid C_{\max}$ は多項式時間で解ける



復習：$Pm \mid \text{prec} \mid C_{\max}$ の難しさ

定理 (今日の内容)

- 問題 $P2 \mid \text{prec} \mid C_{\max}$ は強 NP 困難
- 問題 $P \mid \text{prec}, p_j = 1 \mid C_{\max}$ は (強) NP 困難
- 問題 $P2 \mid \text{prec}, p_j = 1 \mid C_{\max}$ は多項式時間で解ける

未解決問題 (Garey, Johnson '79)

各整数 $m \geq 3$ に対して,
$Pm \mid \text{prec}, p_j = 1 \mid C_{\max}$ は 多項式時間で解けるか？

疑問：先行制約の「かたち」

疑問

現実世界で現れる先行制約の「かたち」は
もっと「行儀がよい」のでは？

↓ 数学的な言い換え (のひとつ)

疑問 (数学的な言い方)

先行制約 (閉路を持たない有向グラフ) として,
もっと限定したものを考えれば, 解きやすくなるのでは？

視点 (問題解決一般において)

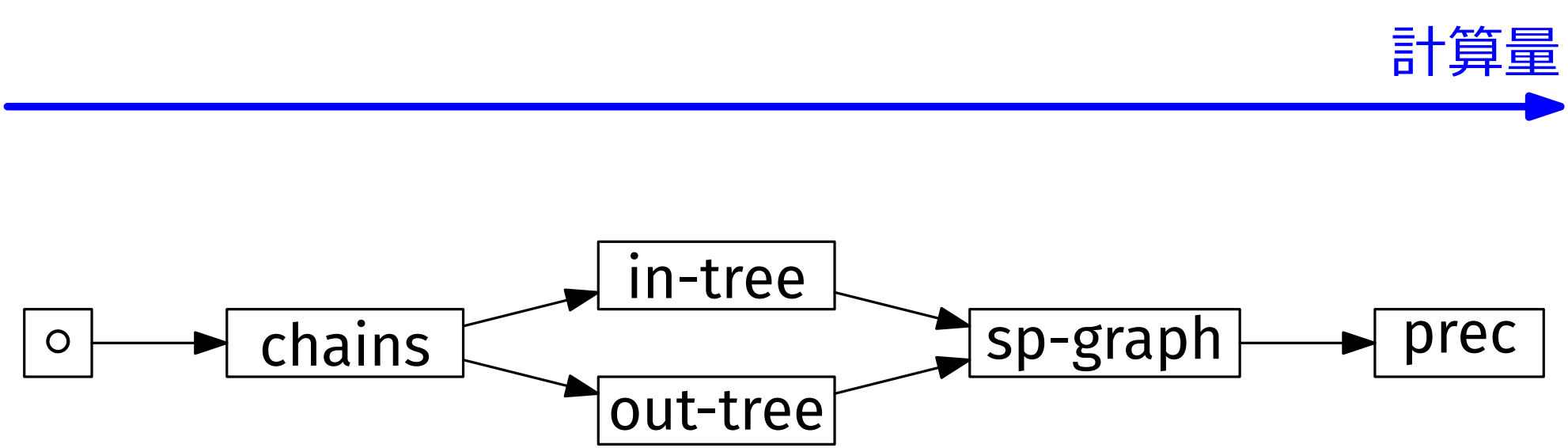
現実世界の問題が持つ **特殊な構造** を活用する

今日の内容

今日の内容

特殊な構造を持つ先行制約の取り扱い方

- 鎖の集まり
- 内向木，外向木
- 直並列グラフ



本日の内容

1. **いろいろな半順序**
2. $1 \mid \text{sp-graph} \mid \sum w_j C_j$ のアルゴリズム
3. $P \mid \text{in-tree}, p_j = 1 \mid C_{\max}$ のアルゴリズム

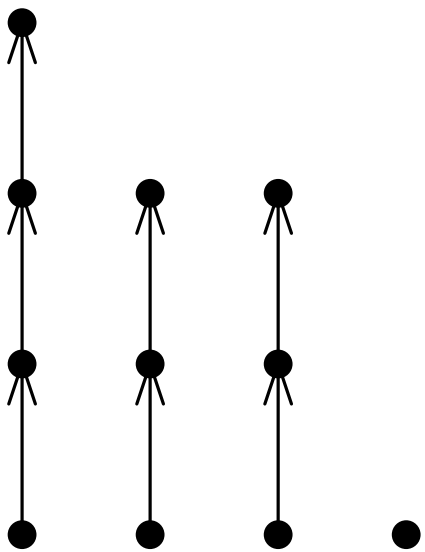
- D. Prot, O. Bellenguez-Morineau, A survey on how the structure of precedence constraints may change the complexity class of scheduling problems. *Journal of Scheduling* 21 (2018) pp. 3–16.
- J. Valdes, R. E. Tarjan, E. L. Lawler, The recognition of series parallel digraphs. *SIAM Journal on Computing* 11 (1982) pp. 298–313.

鎖の集まり (鎖の集合)

$\mathcal{G}$：先行制約を表す有向グラフ (閉路なし，推移辺なし)

定義：鎖の集まり

$\mathcal{G}$ が **鎖の集まり** (a set of chains) であるとは，
すべての頂点の出次数と入次数が 1 以下であること



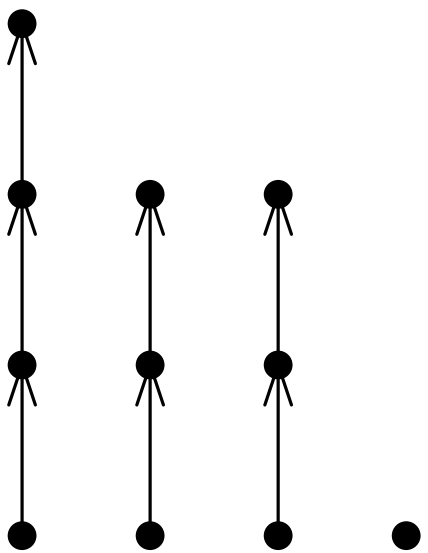
先行制約が鎖の集まりであるとき，β に chains と書く

鎖の集まり (鎖の集合)

$\mathcal{G}$：先行制約を表す有向グラフ (閉路なし，推移辺なし)

定義：鎖の集まり

$\mathcal{G}$ が **鎖の集まり** (a set of chains) であるとは，
すべての頂点の出次数と入次数が 1 以下であること



先行制約がない $\Rightarrow$

すべての頂点の出次数と入次数が 0

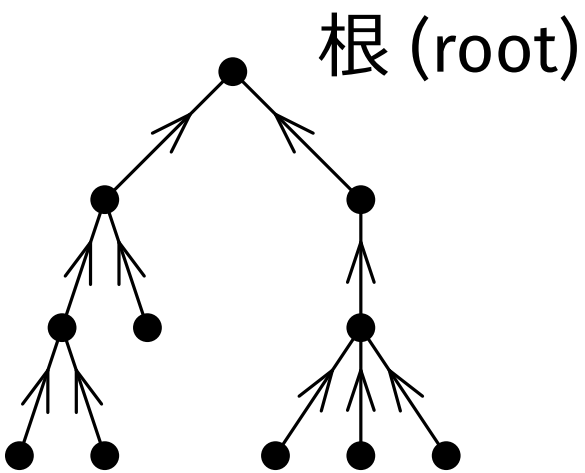
先行制約が鎖の集まりであるとき，β に chains と書く

内向木

$\mathcal{G}$：先行制約を表す有向グラフ (閉路なし，推移辺なし)

定義：内向木

$\mathcal{G}$ が **内向木** (in-tree) であるとは，
1 つの頂点の出次数が 0 で
他のすべての頂点の出次数が 1 であること



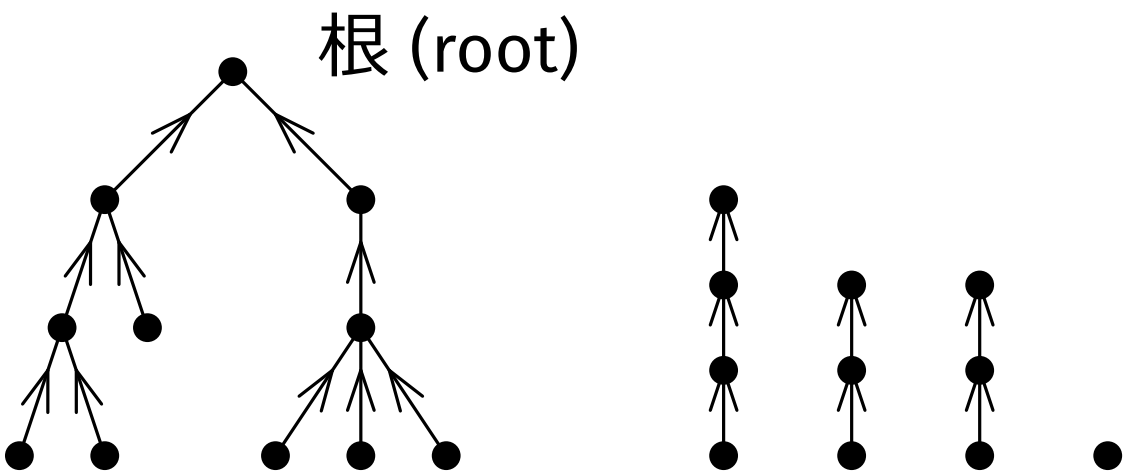
先行制約が内向木であるとき，β に in-tree と書く

内向木

$\mathcal{G}$：先行制約を表す有向グラフ (閉路なし，推移辺なし)

定義：内向木

$\mathcal{G}$ が **内向木** (in-tree) であるとは，
1 つの頂点の出次数が 0 で
他のすべての頂点の出次数が 1 であること



鎖の集まり は

内向木 と見なせる

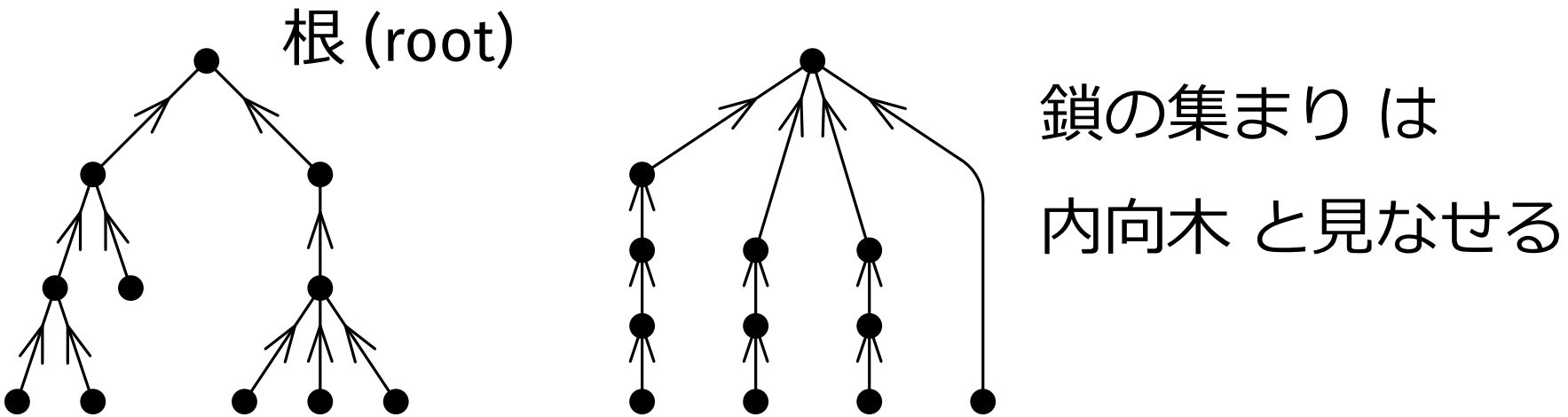
先行制約が内向木であるとき，β に in-tree と書く

内向木

$\mathcal{G}$：先行制約を表す有向グラフ (閉路なし，推移辺なし)

定義：内向木

$\mathcal{G}$ が **内向木** (in-tree) であるとは，
1 つの頂点の出次数が 0 で
他のすべての頂点の出次数が 1 であること



鎖の集まり は

内向木 と見なせる

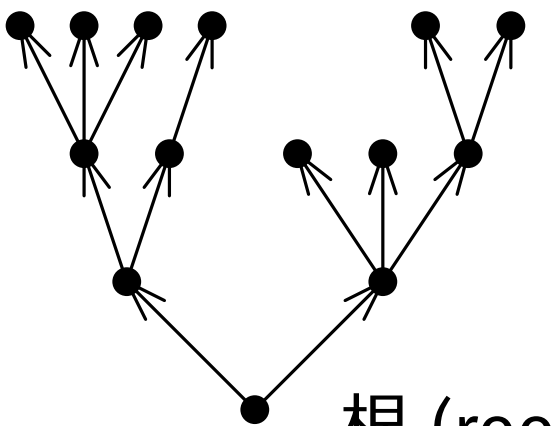
先行制約が内向木であるとき，β に in-tree と書く

外向木

$\mathcal{G}$：先行制約を表す有向グラフ (閉路なし，推移辺なし)

定義：外向木

$\mathcal{G}$ が **外向木** (out-tree) であるとは，
1 つの頂点の入次数が 0 で
他のすべての頂点の入次数が 1 であること



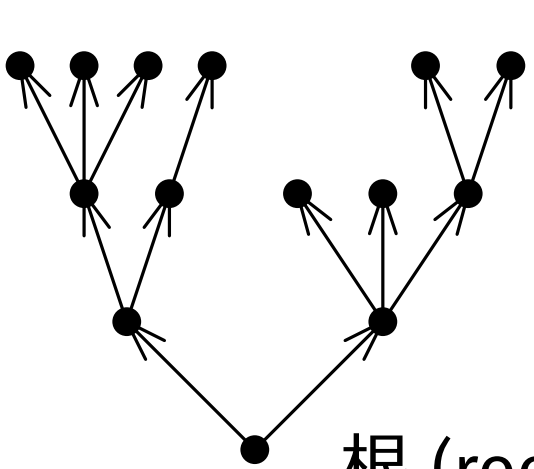
先行制約が外向木であるとき，β に out-tree と書く

外向木

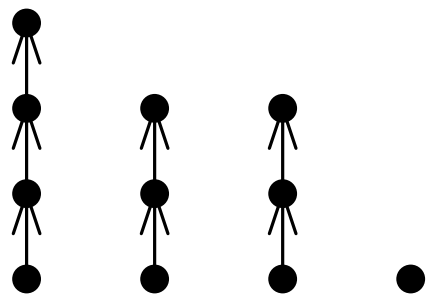
$\mathcal{G}$：先行制約を表す有向グラフ (閉路なし，推移辺なし)

定義：外向木

$\mathcal{G}$ が **外向木** (out-tree) であるとは，
1 つの頂点の入次数が 0 で
他のすべての頂点の入次数が 1 であること



根 (root)



鎖の集まり は
外向木 と見なせる

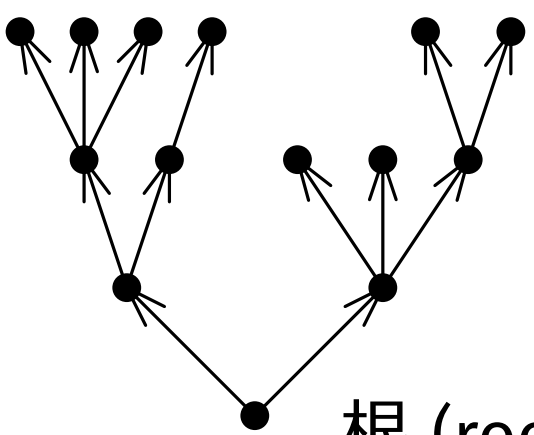
先行制約が外向木であるとき，β に out-tree と書く

外向木

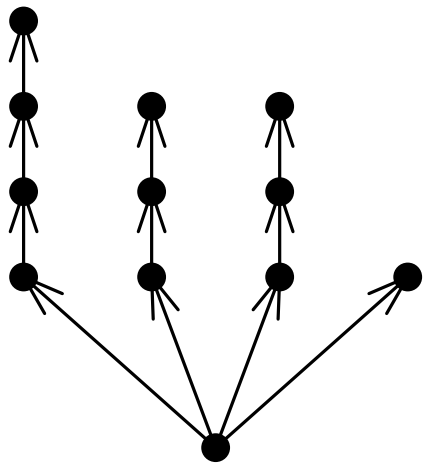
$\mathcal{G}$：先行制約を表す有向グラフ (閉路なし，推移辺なし)

定義：外向木

$\mathcal{G}$ が **外向木** (out-tree) であるとは，
1 つの頂点の入次数が 0 で
他のすべての頂点の入次数が 1 であること



根 (root)

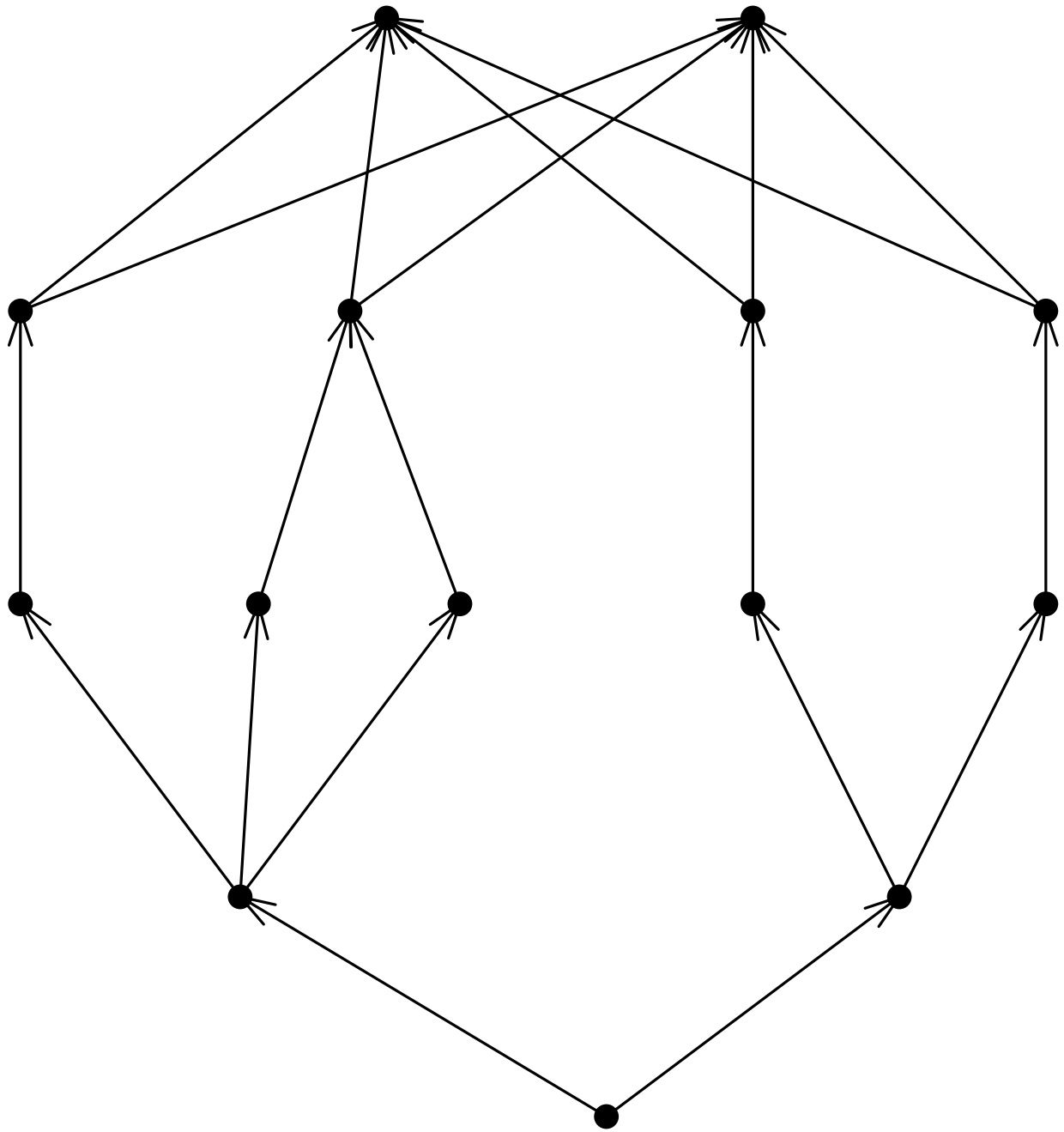


鎖の集まり は

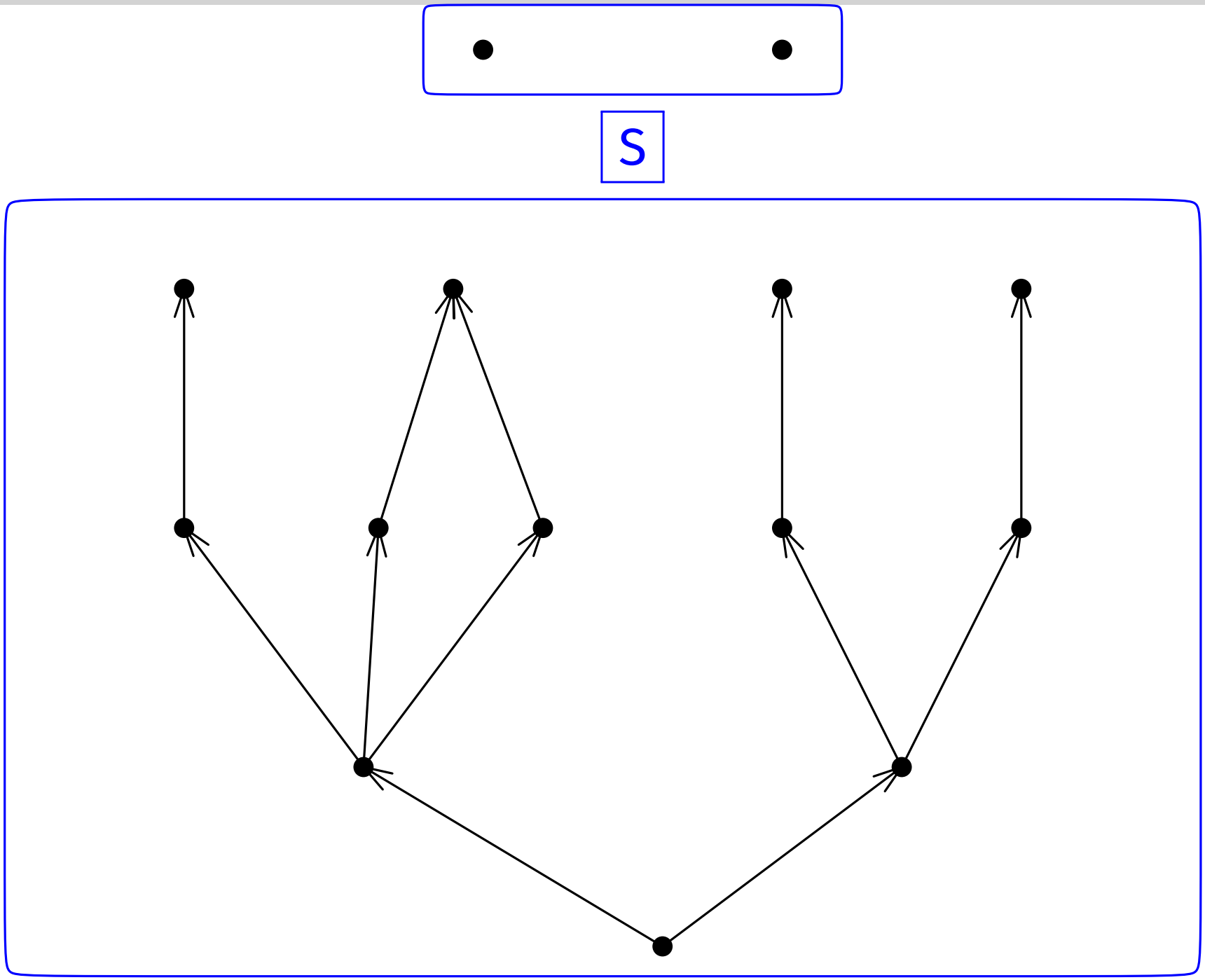
外向木 と見なせる

先行制約が外向木であるとき，β に out-tree と書く

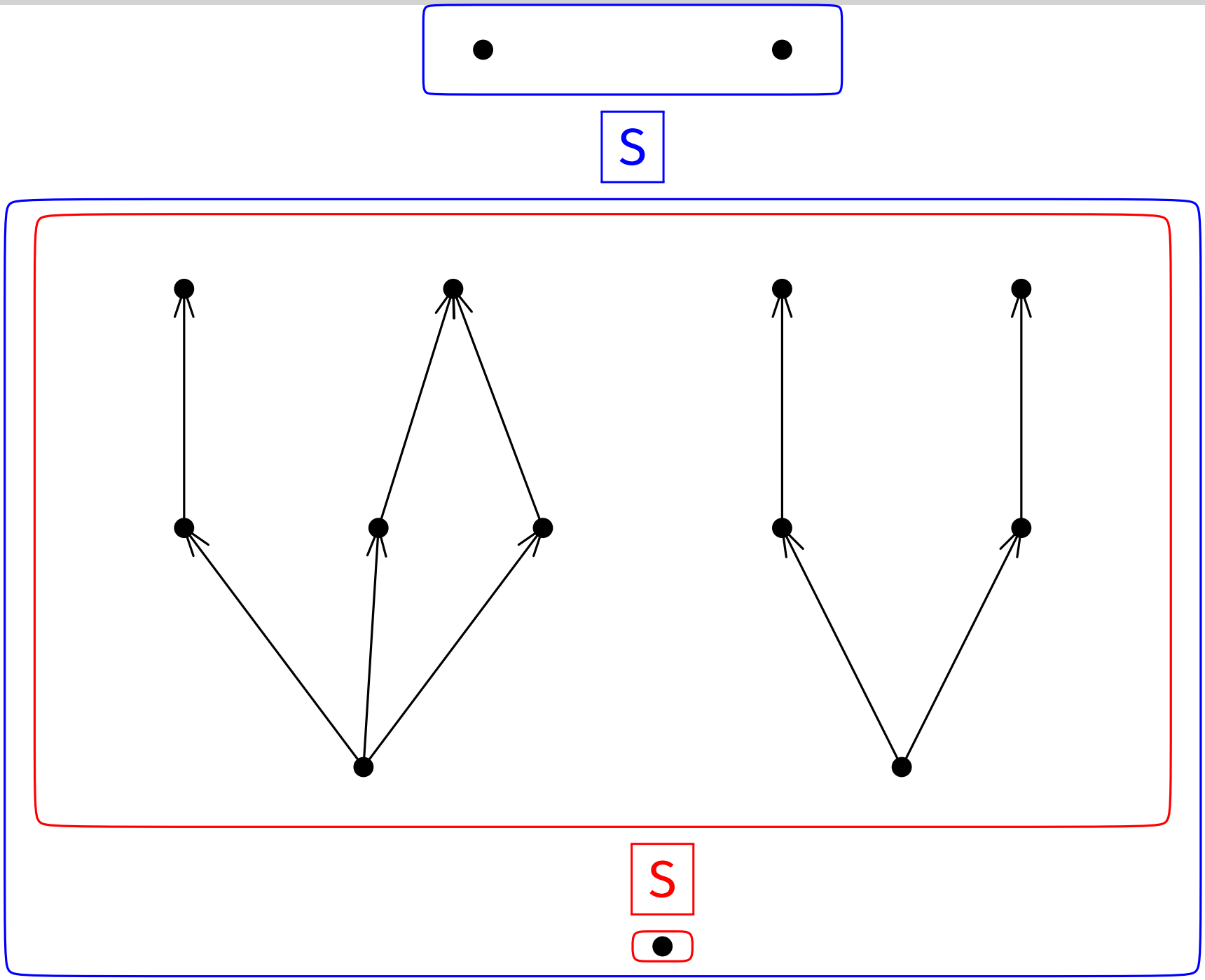
直並列グラフ：例



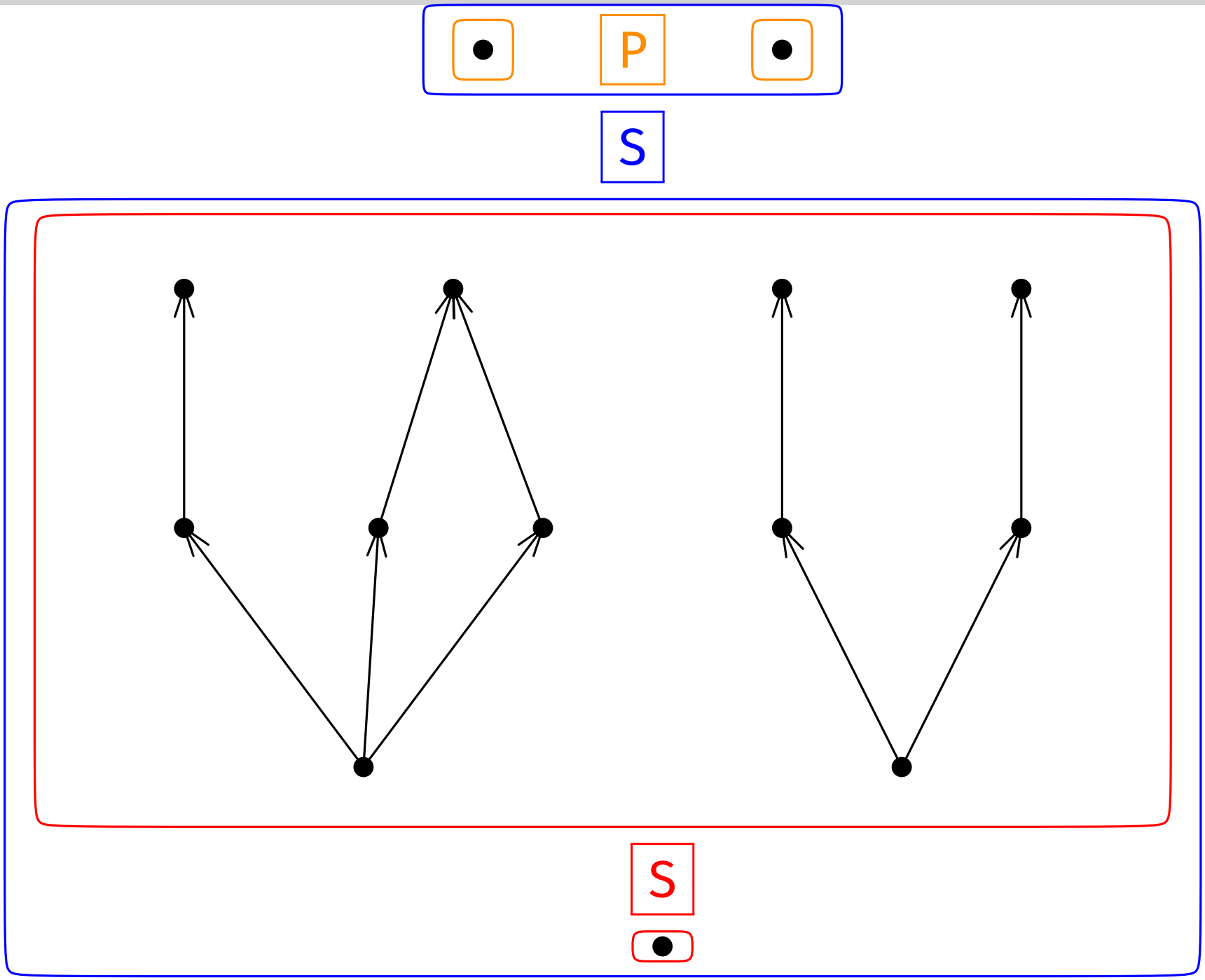
直並列グラフ：例



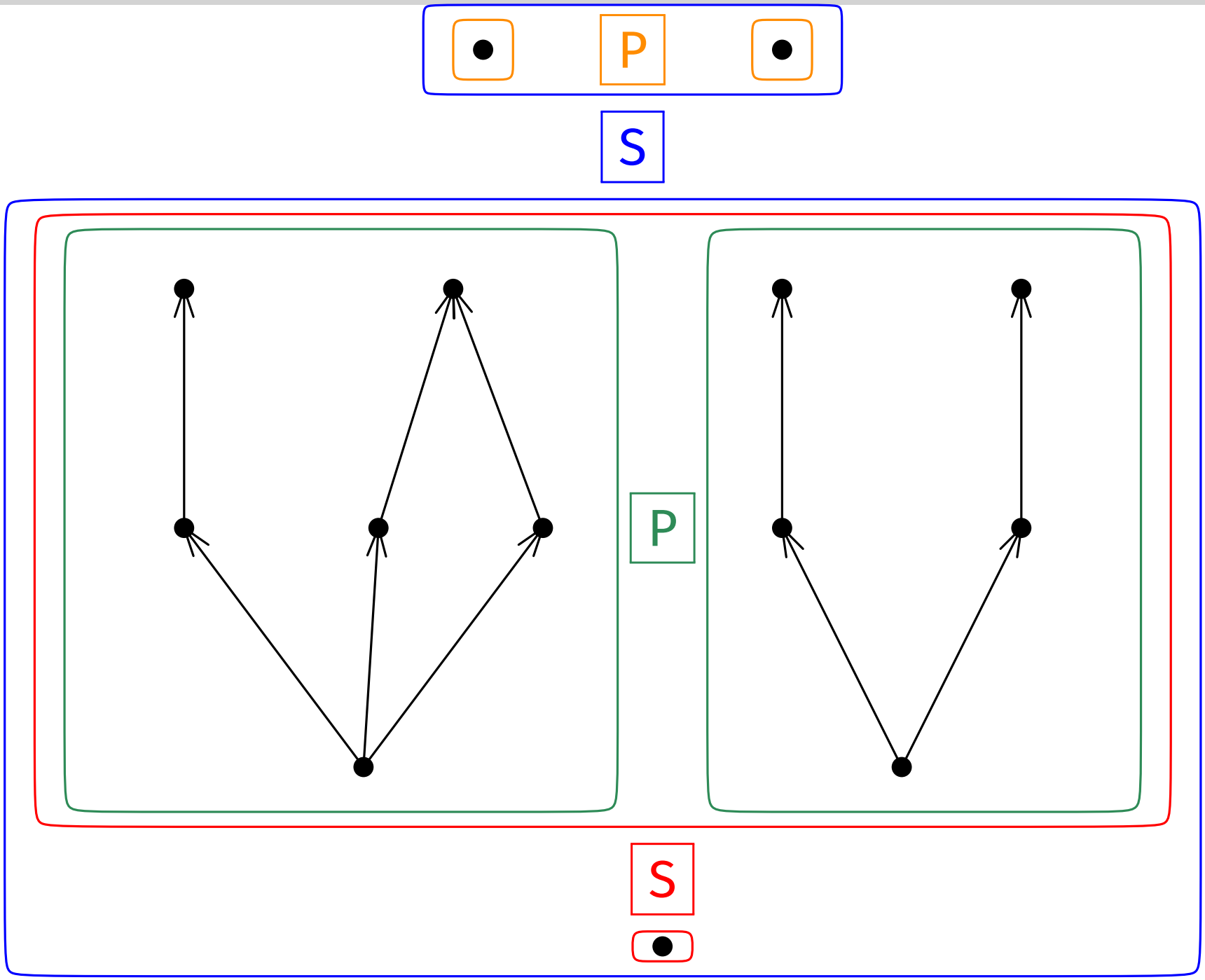
直並列グラフ：例



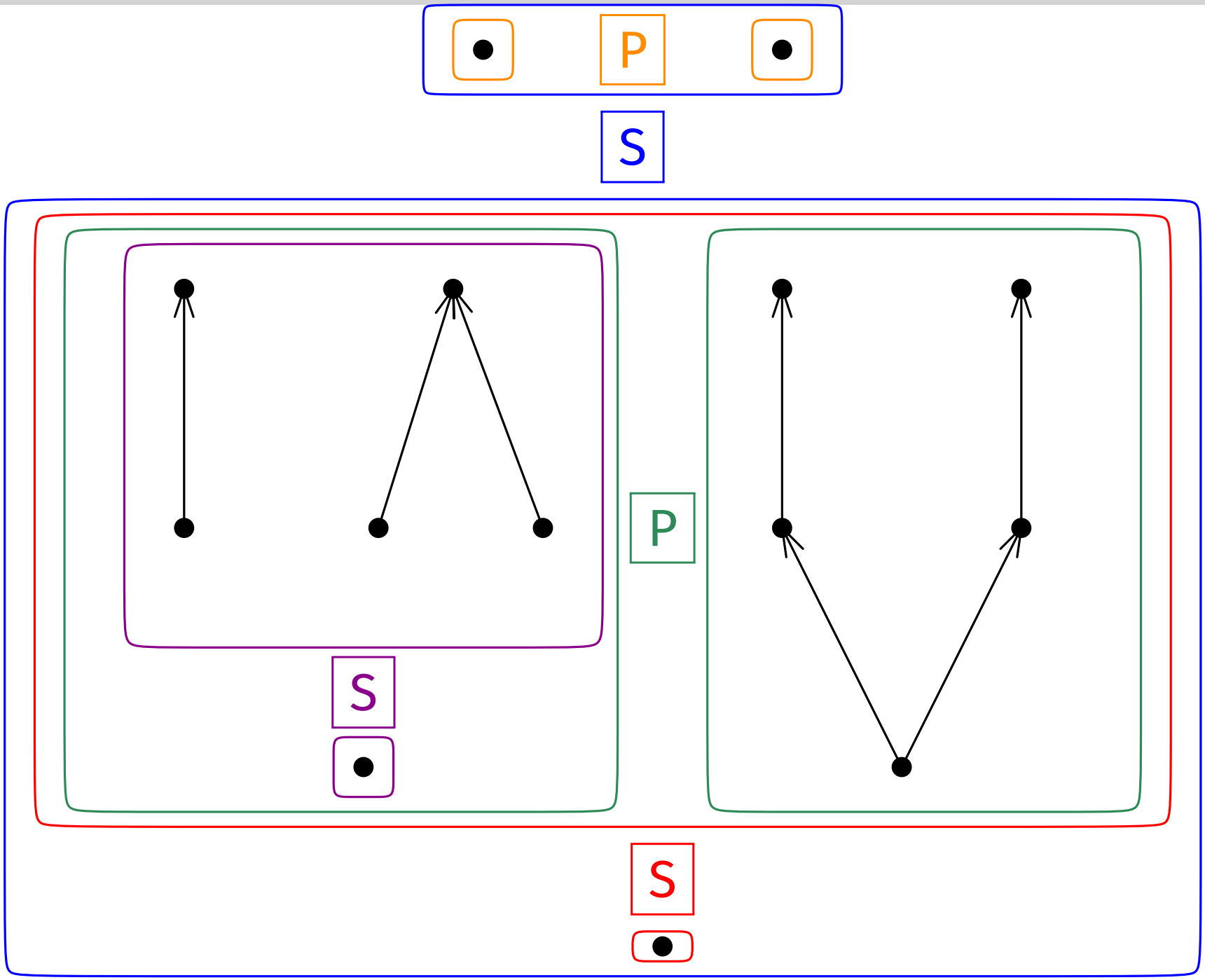
直並列グラフ：例



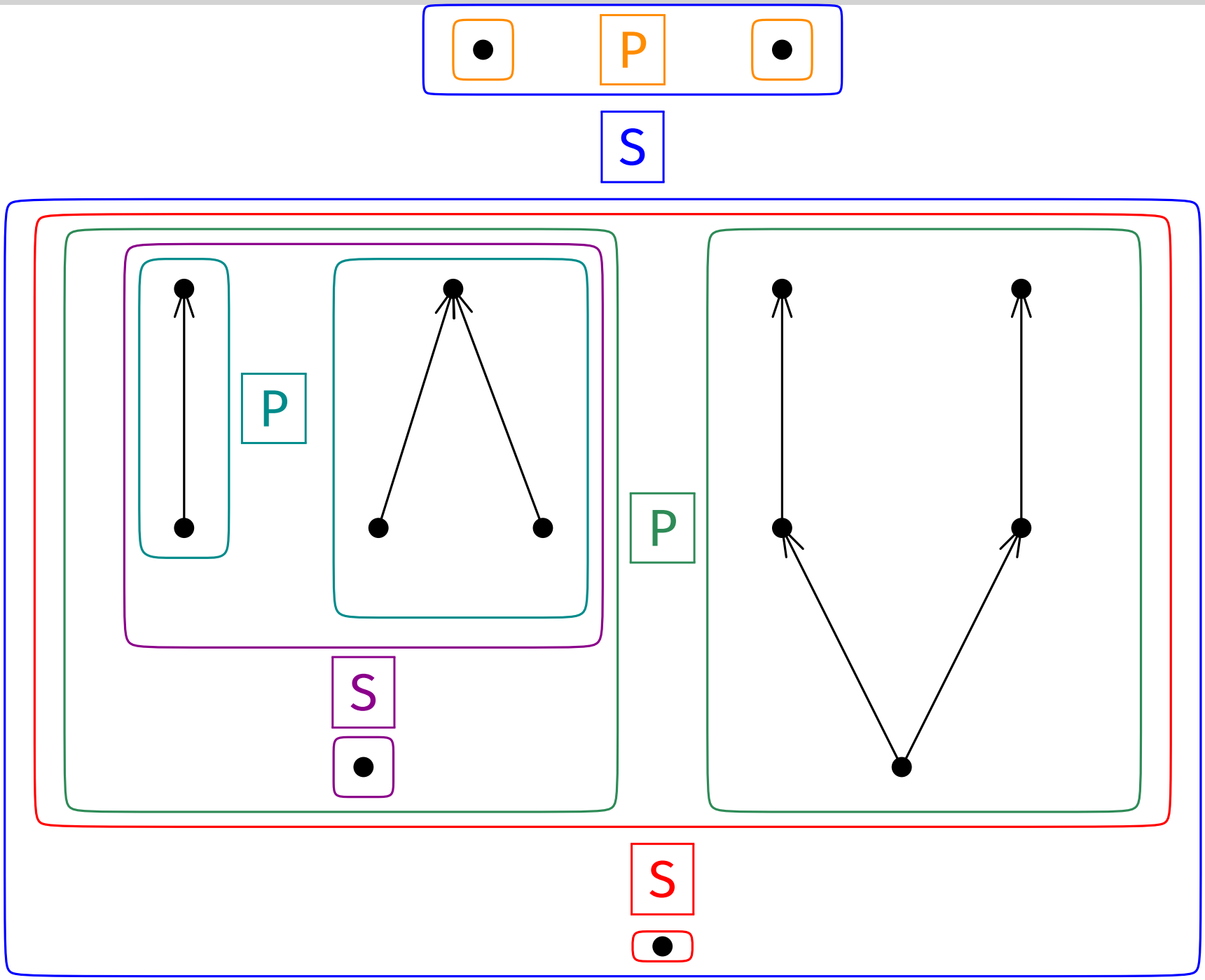
直並列グラフ：例



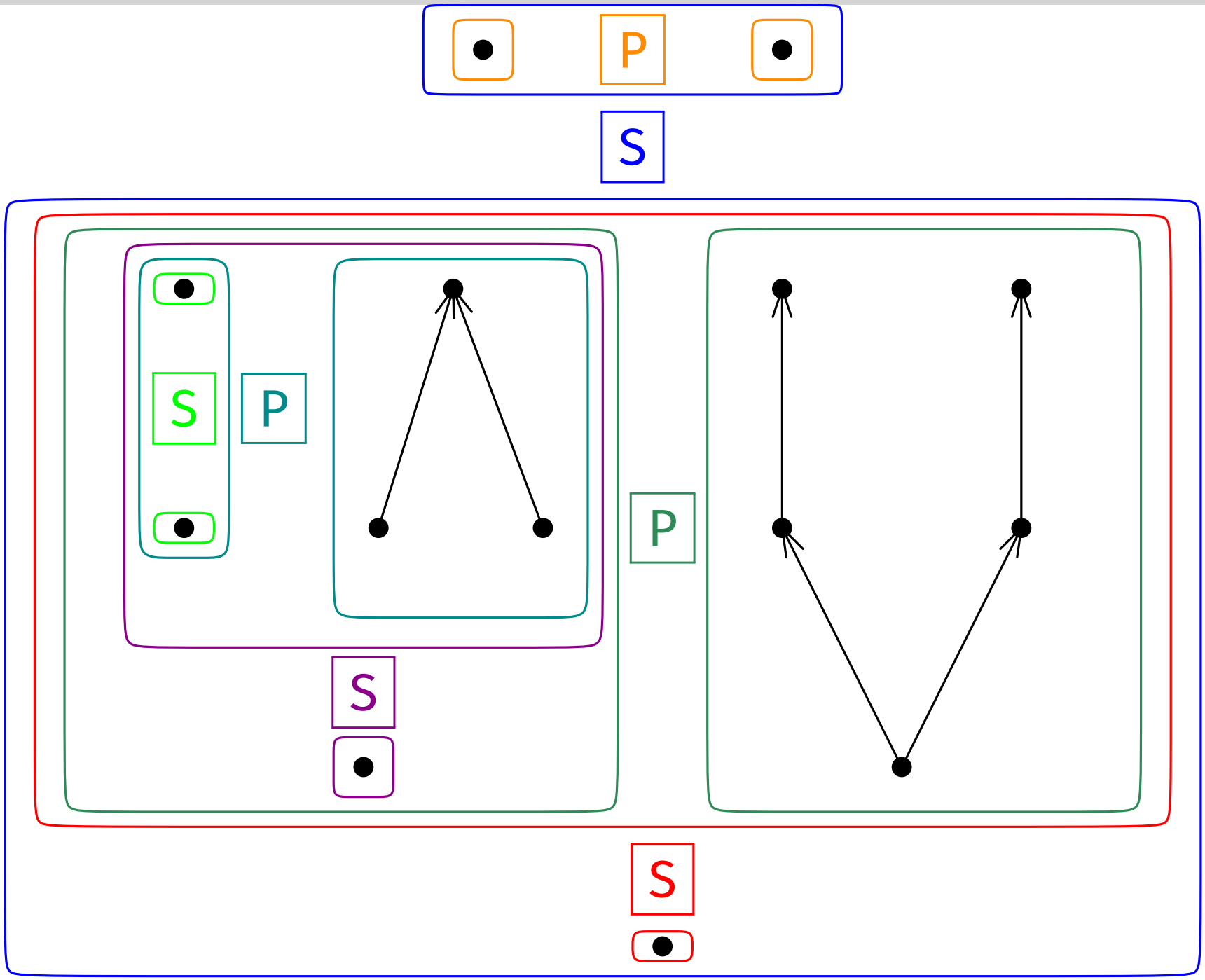
直並列グラフ：例



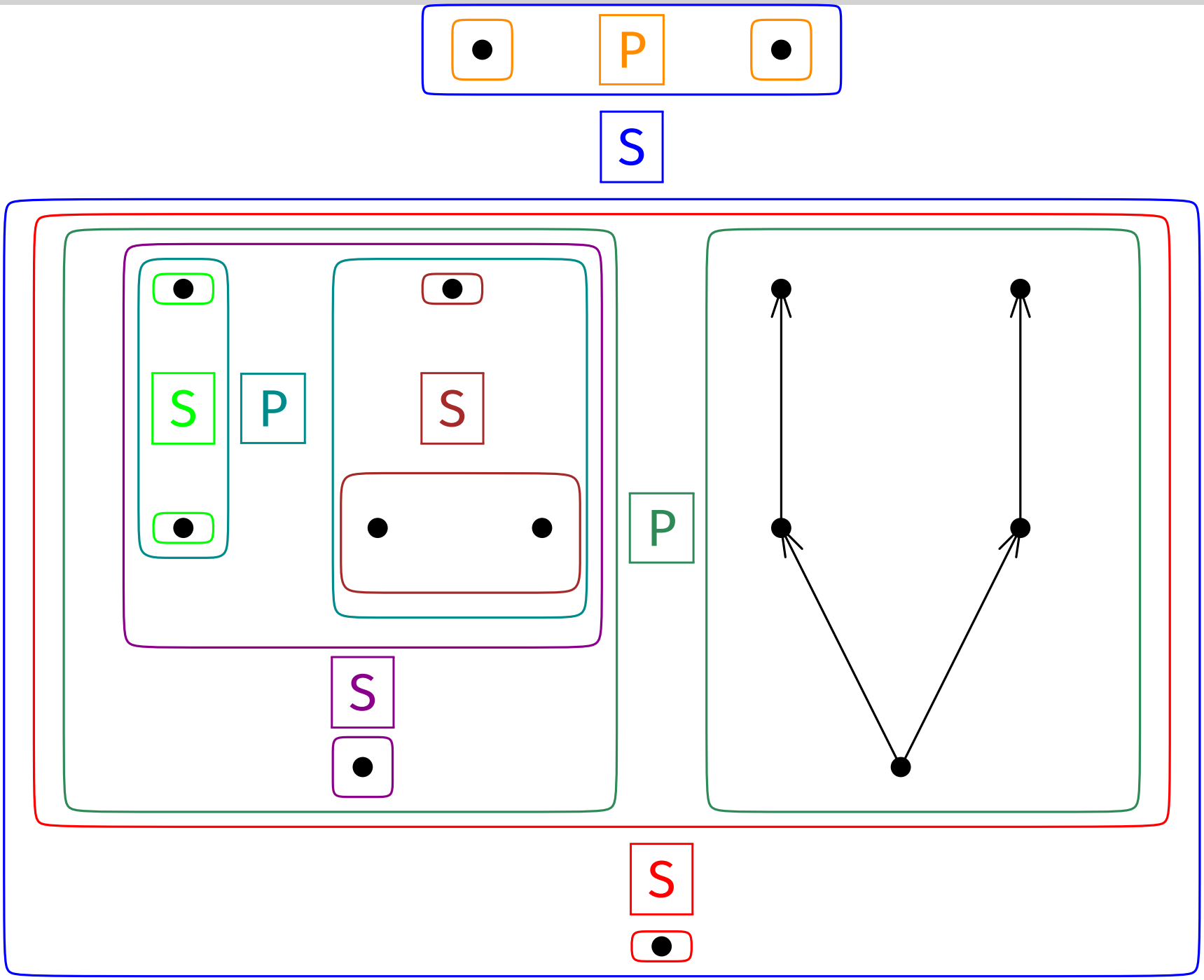
直並列グラフ：例



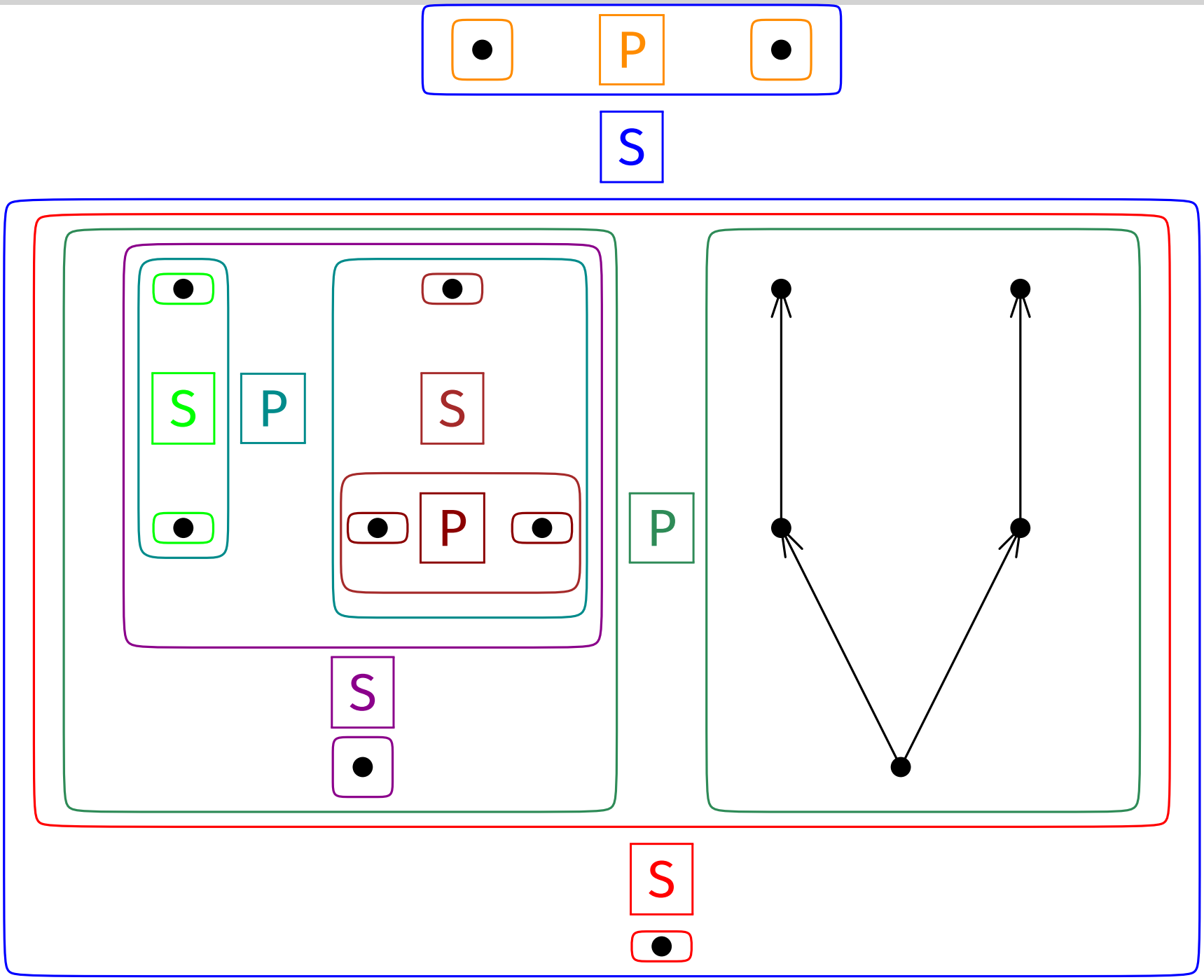
直並列グラフ：例



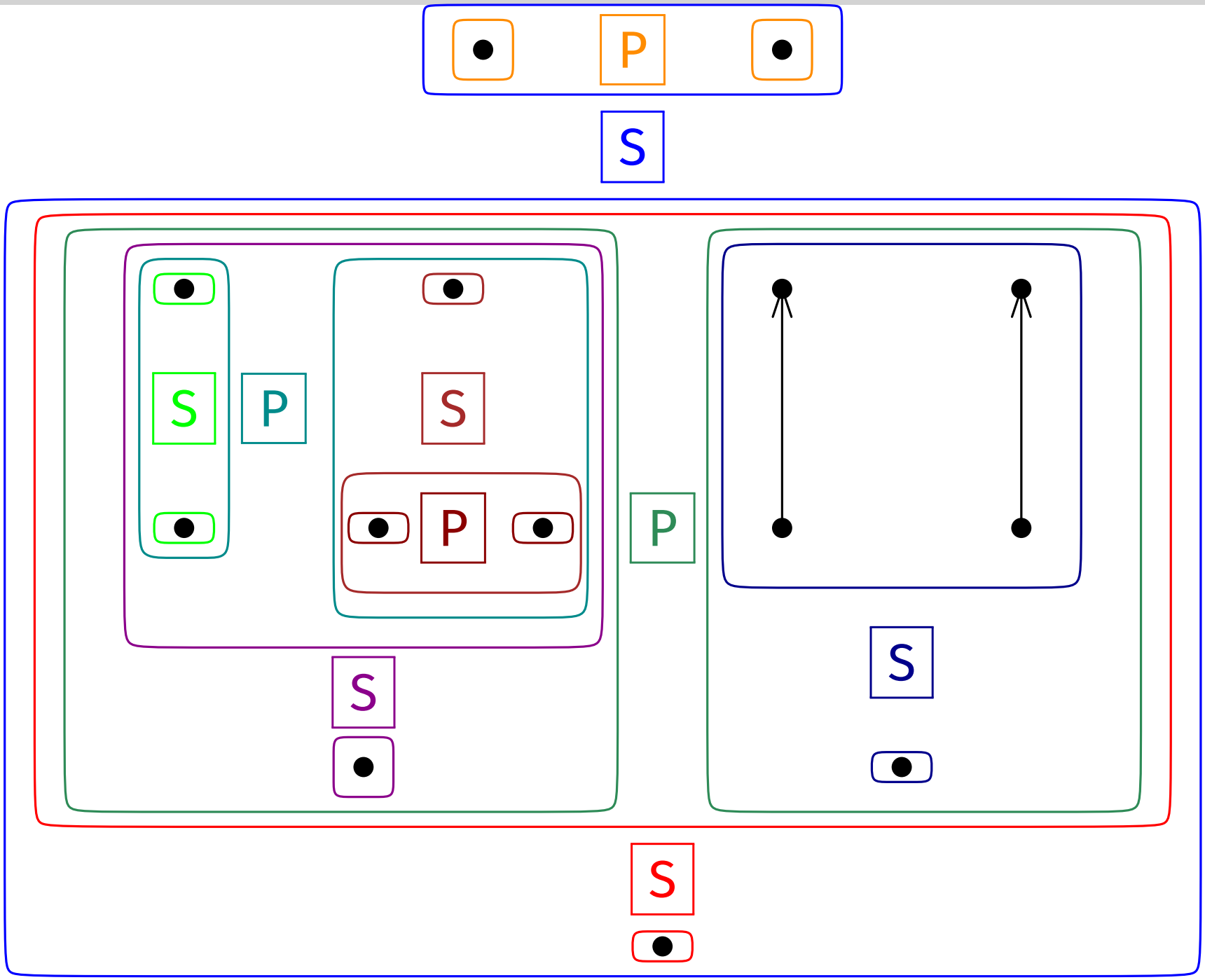
直並列グラフ：例



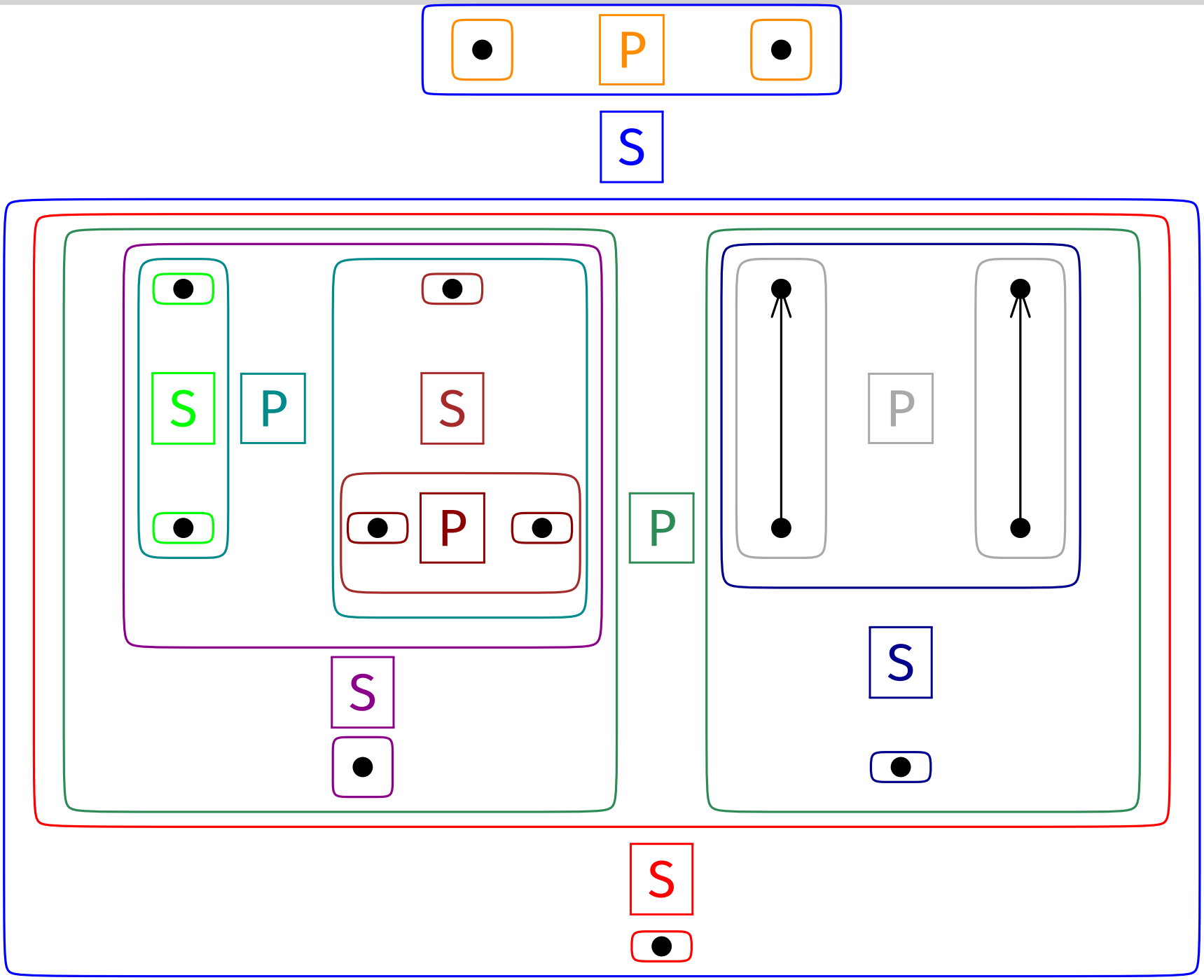
直並列グラフ：例



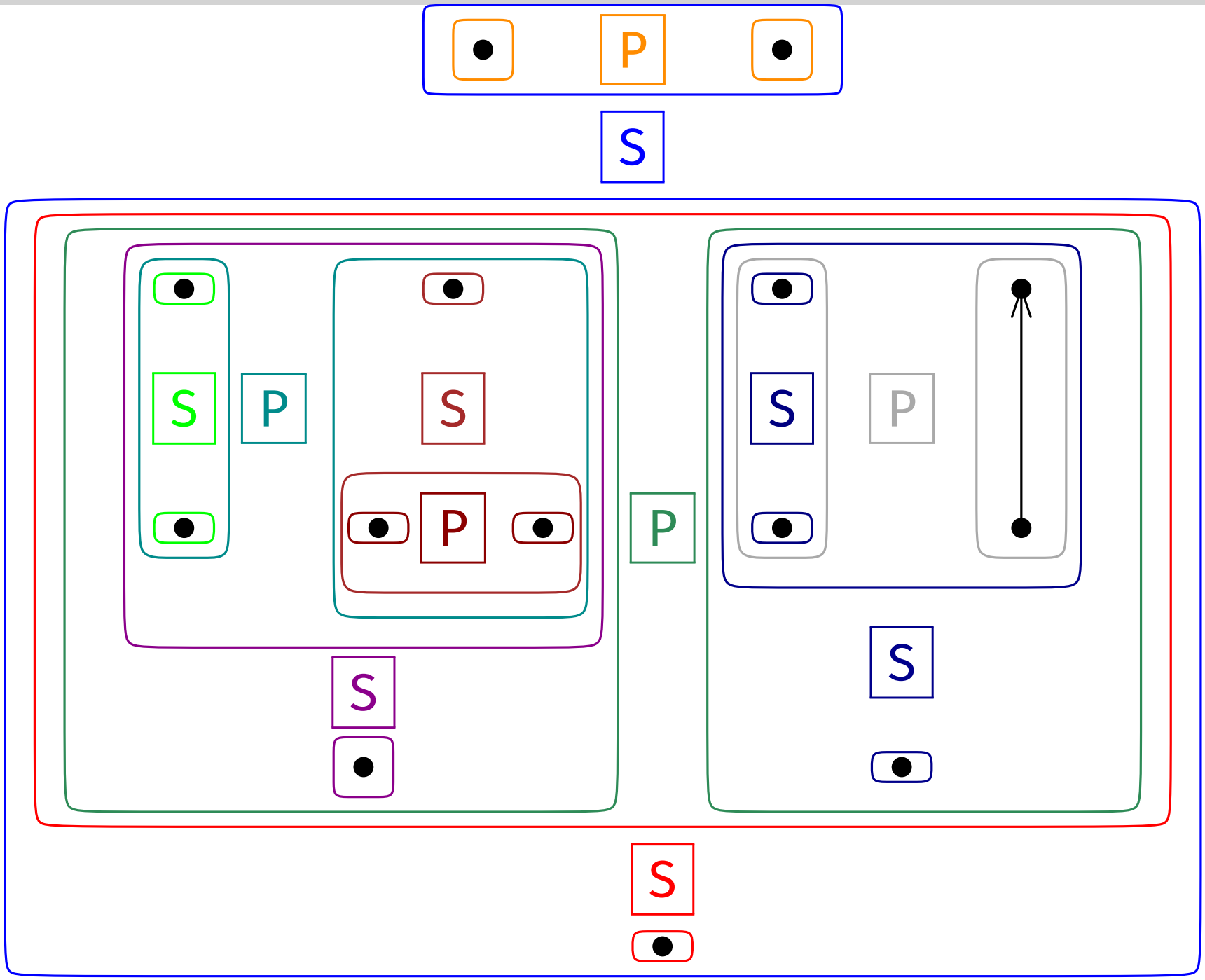
直並列グラフ：例



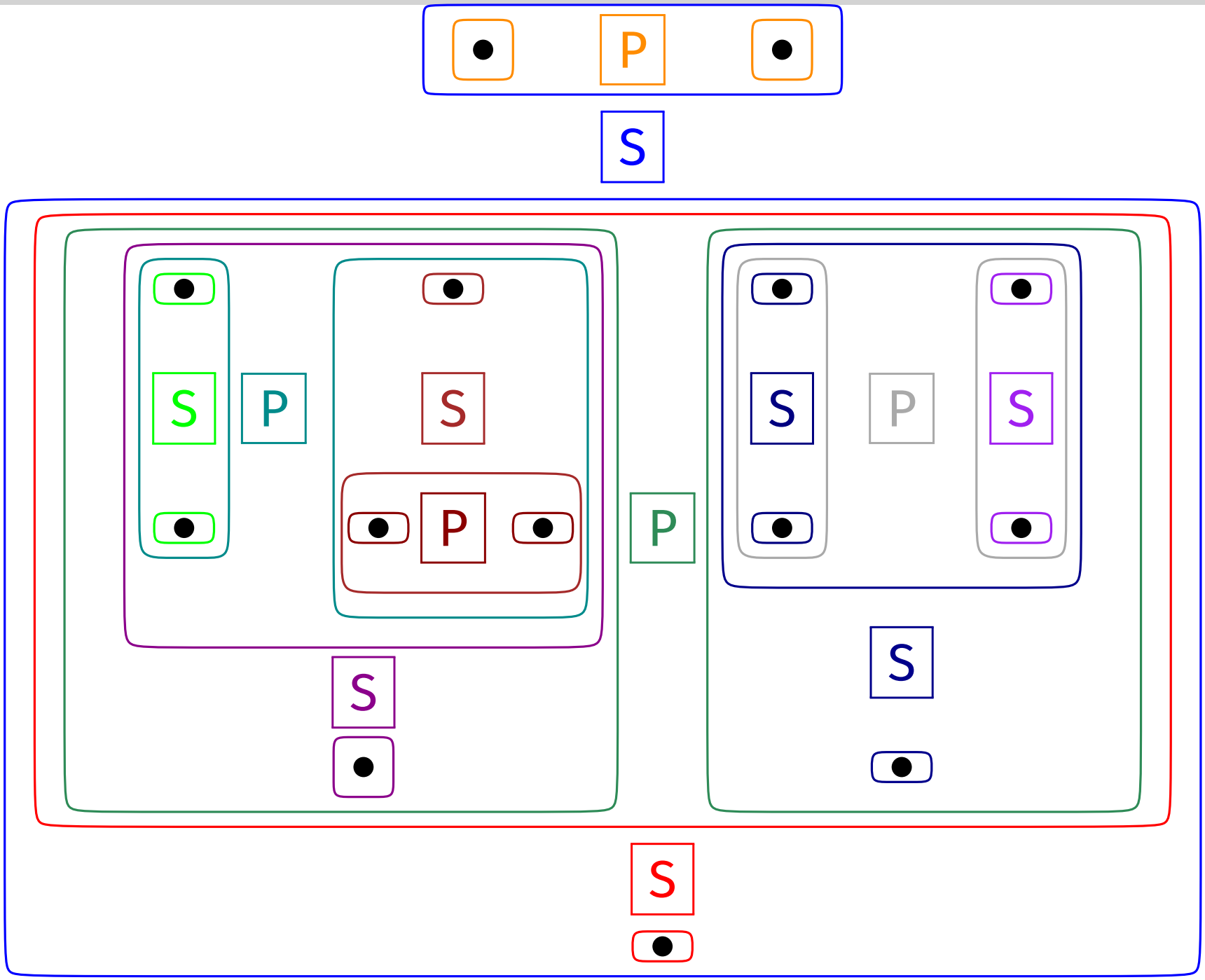
直並列グラフ：例



直並列グラフ：例



直並列グラフ：例



直並列グラフ

$\mathcal{G}$：先行制約を表す有向グラフ (閉路なし，推移辺なし)

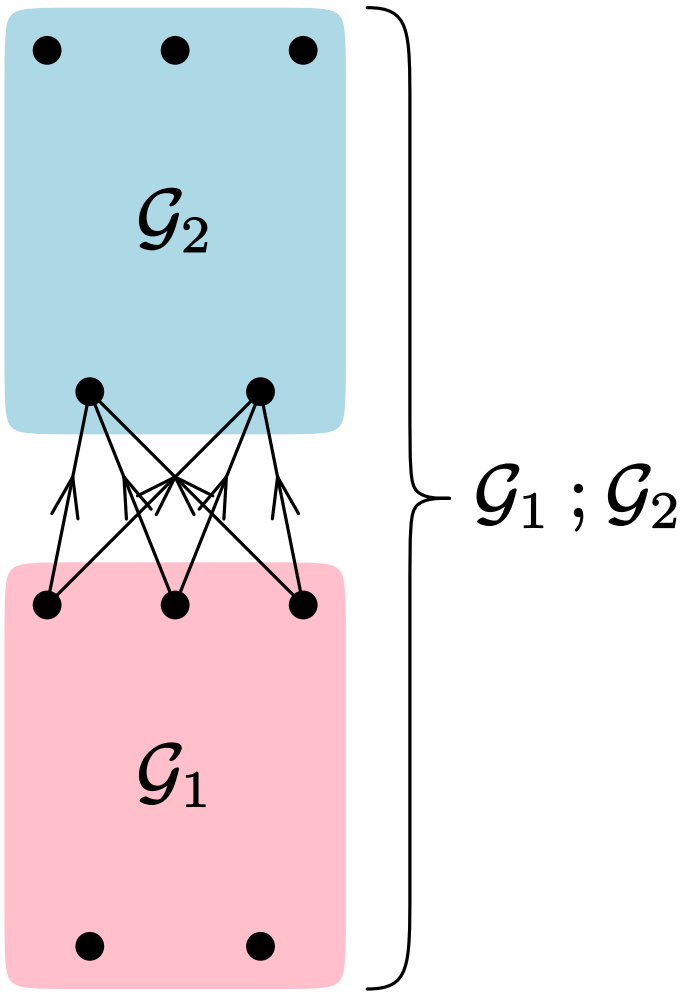
定義：直並列グラフ

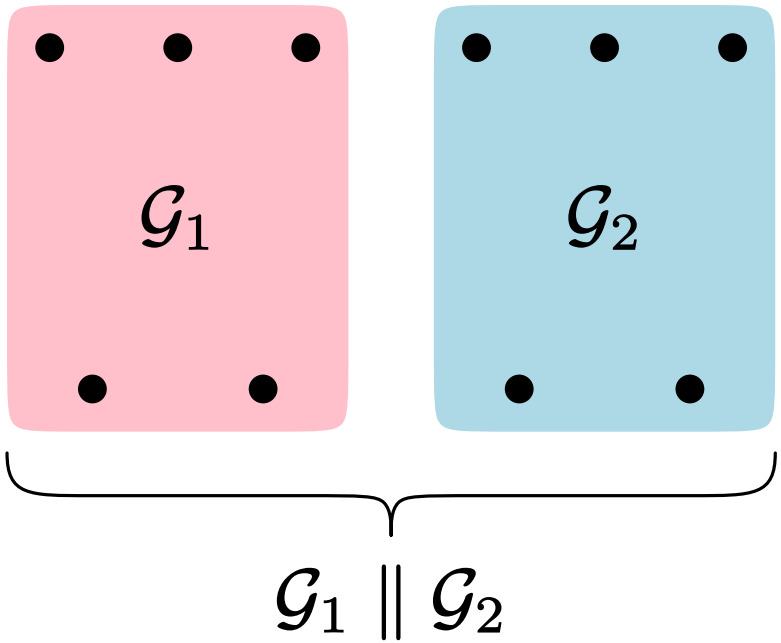
$\mathcal{G}$ が**直並列**グラフ (series-parallel graph) であるとは，
次の条件のどれかを満たすこと

- $\mathcal{G}$ はひとつの頂点から成るグラフである
- ある直並列グラフ $\mathcal{G}_1, \mathcal{G}_2$ に対して，
  $\mathcal{G}$ が $\mathcal{G}_1$ と $\mathcal{G}_2$ の **直列接続** $\mathcal{G}_1 ; \mathcal{G}_2$ である
- ある直並列グラフ $\mathcal{G}_1, \mathcal{G}_2$ に対して，
  $\mathcal{G}$ が $\mathcal{G}_1$ と $\mathcal{G}_2$ の **並列接続** $\mathcal{G}_1 \parallel \mathcal{G}_2$ である

注：直並列グラフの定義として他のものが文献にあるが，
概念としては違うもの

直列接続 と 並列接続

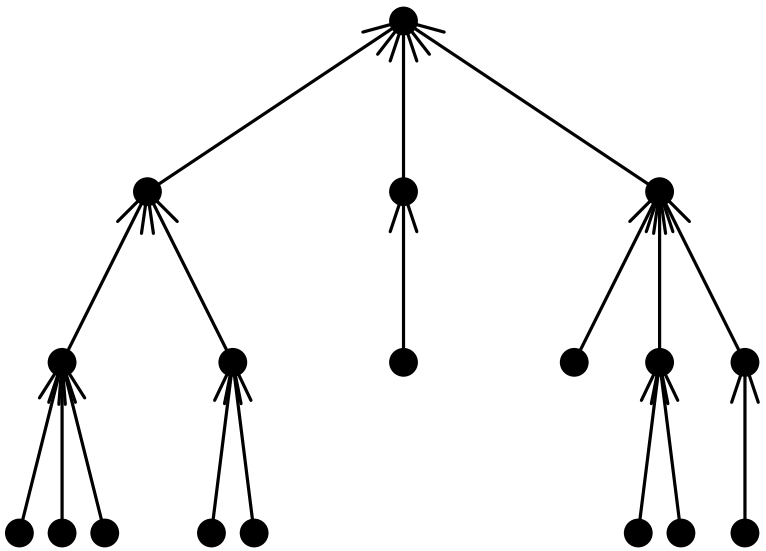


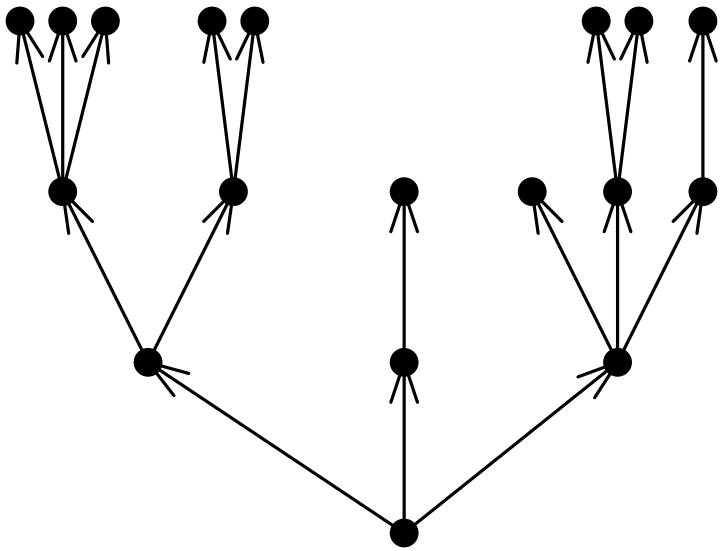


一般に，$\mathcal{G}_1 : \mathcal{G}_2 \neq \mathcal{G}_2 ; \mathcal{G}_1$，　$(\mathcal{G}_1 ; \mathcal{G}_2) ; \mathcal{G}_3 = \mathcal{G}_1 ; (\mathcal{G}_2 ; \mathcal{G}_3)$

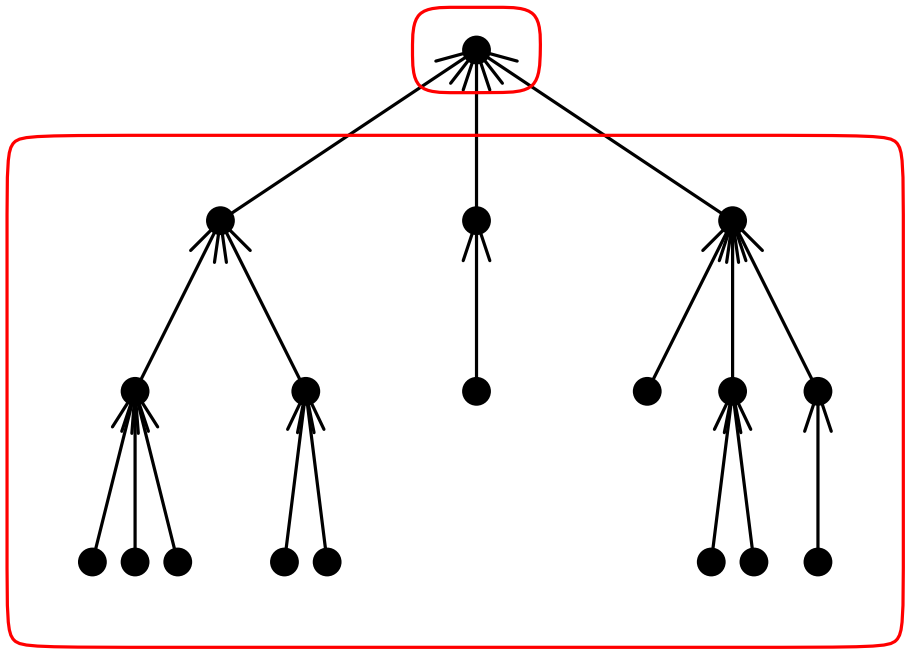
$\mathcal{G}_1 \parallel \mathcal{G}_2 = \mathcal{G}_2 \parallel \mathcal{G}_1$，　$(\mathcal{G}_1 \parallel \mathcal{G}_2) \parallel \mathcal{G}_3 = \mathcal{G}_1 \parallel (\mathcal{G}_2 \parallel \mathcal{G}_3)$

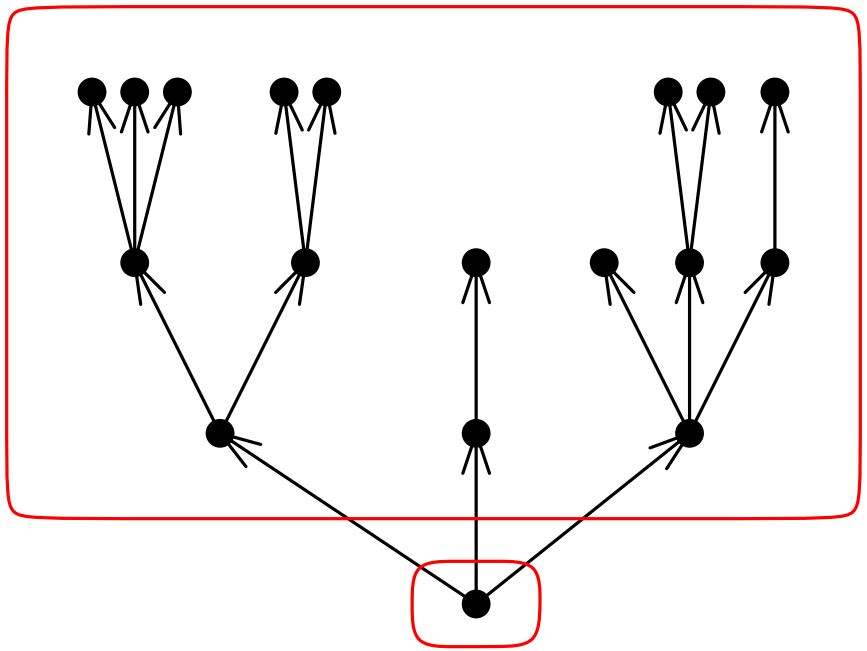
内向木と外向木は直並列グラフ



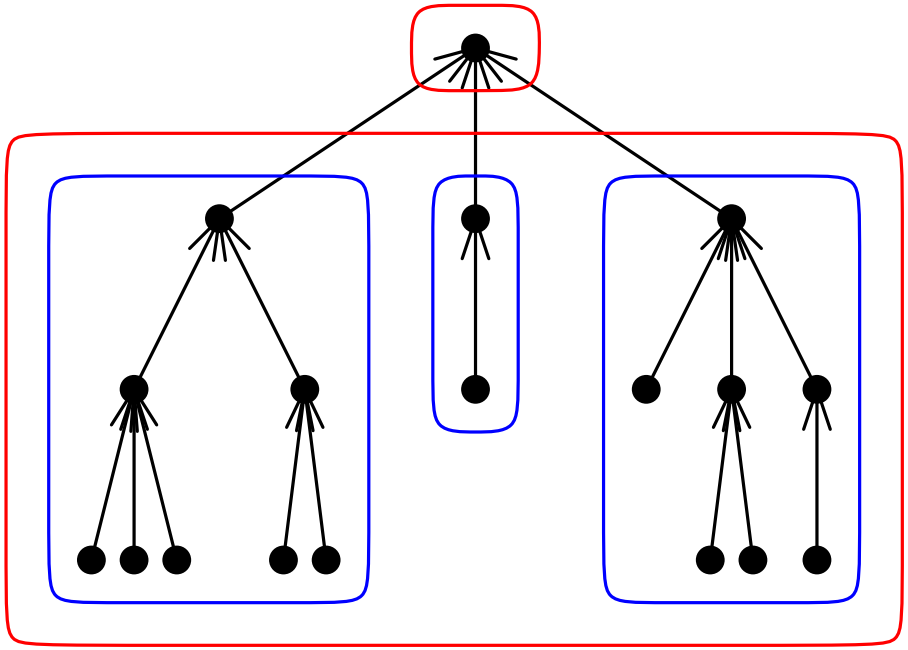


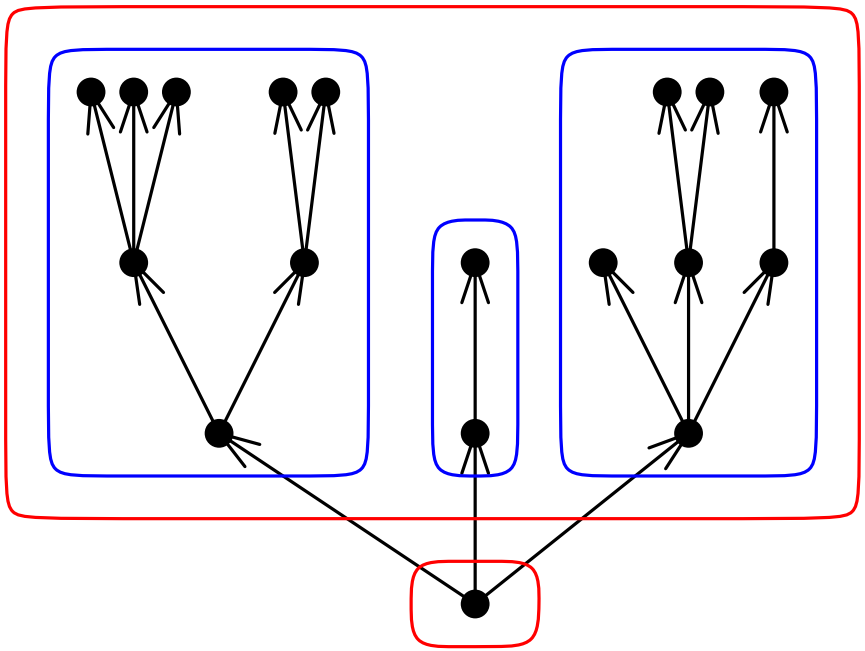
内向木と外向木は直並列グラフ



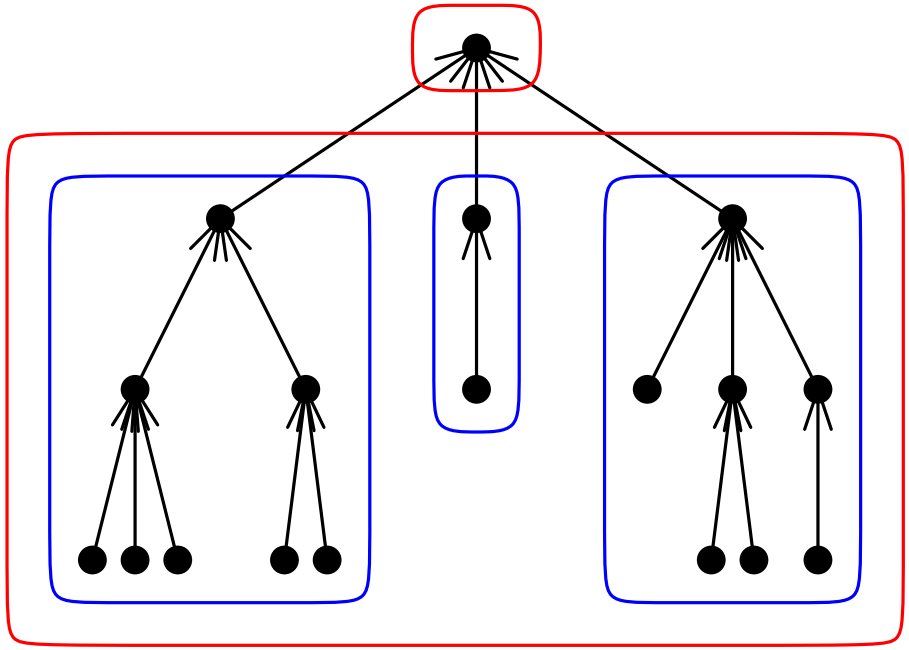


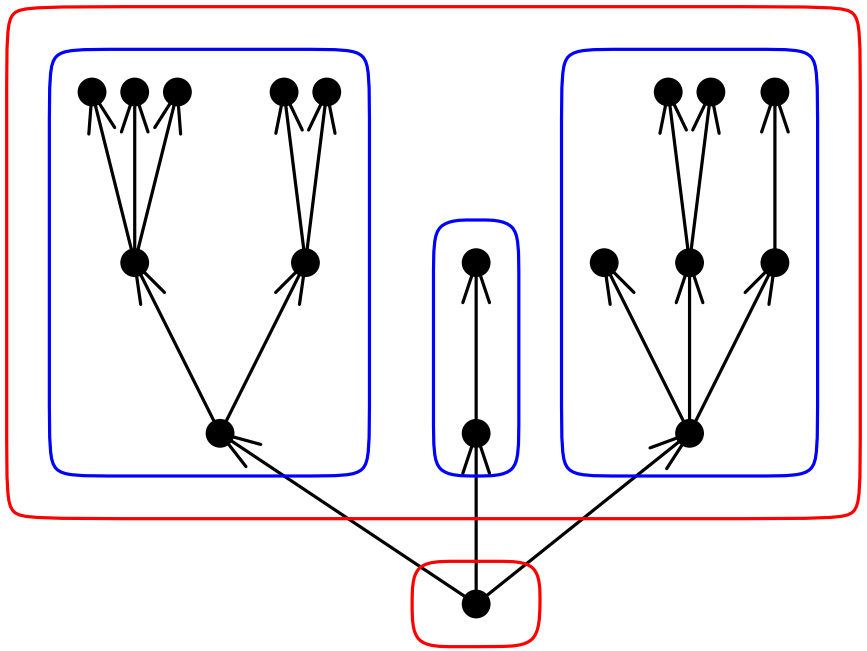
内向木と外向木は直並列グラフ

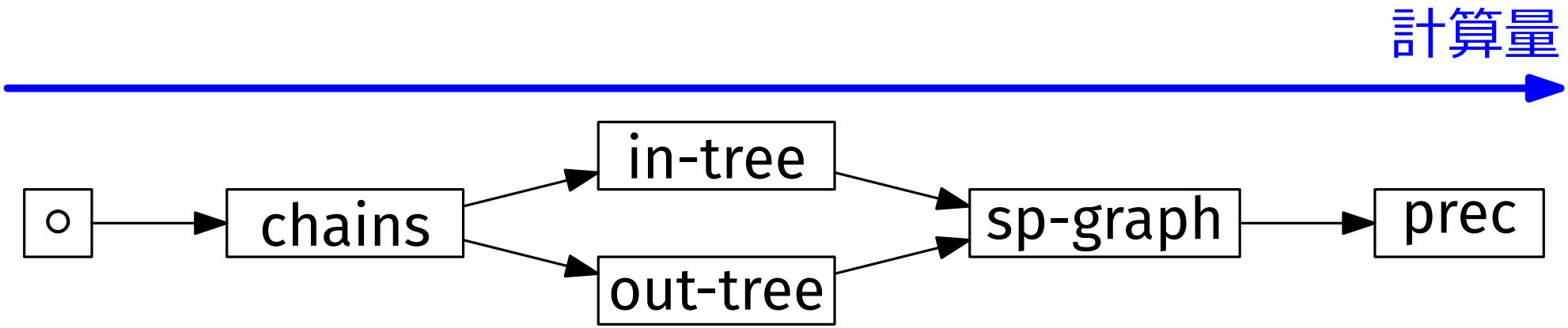




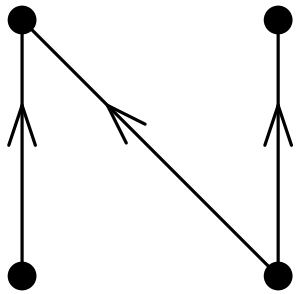
内向木と外向木は直並列グラフ







直並列グラフではない有向グラフ



半順序集合の文献では N と呼ばれる

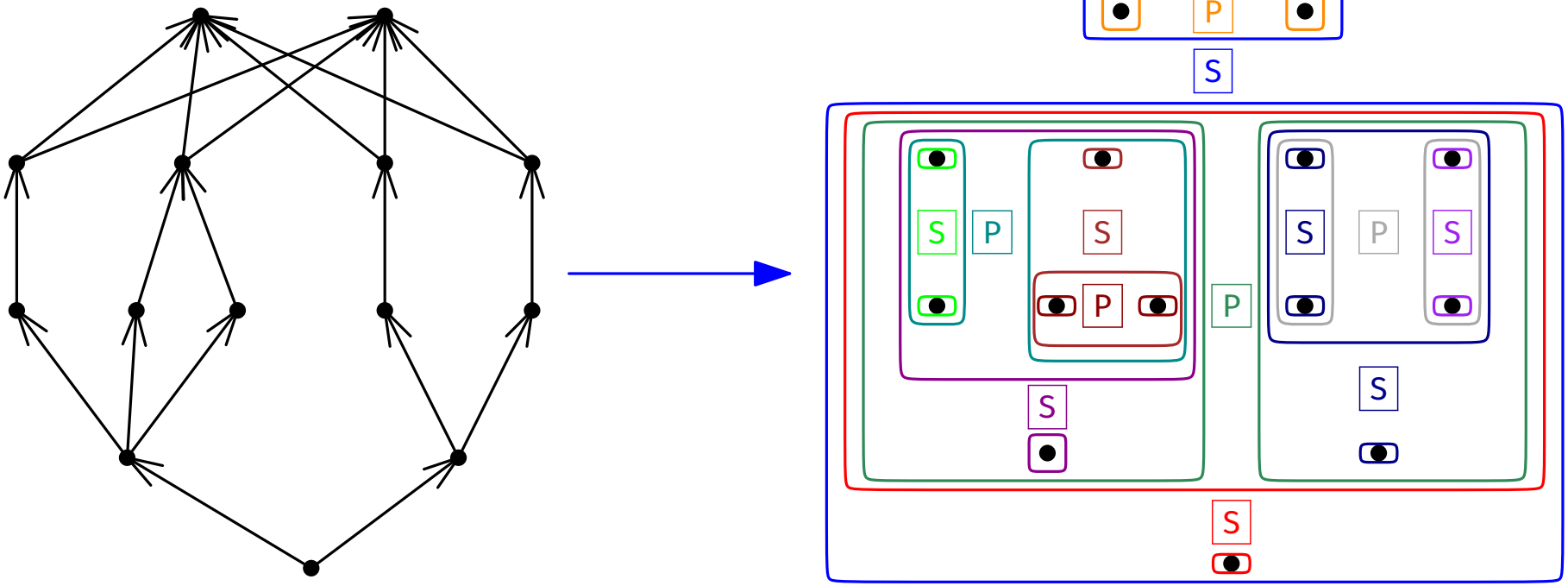
性質 (Valdes, Tarjan, Lawler '82)

閉路と推移辺を持たない有向グラフ $\mathcal{G}$ に対して,

$\mathcal{G}$ が直並列グラフ $\Leftrightarrow$

$\mathcal{G}$ (の推移的閉包) が N を誘導部分グラフとして含まない

直並列分解 (series-parallel decomposition)



性質 (Valdes, Tarjan, Lawler '82)

直並列グラフ $\mathcal{G}$ の直並列分解は $O(n+h)$ 時間で構成できる

$n = \mathcal{G}$ の頂点数，$h = \mathcal{G}$ の辺数

証明はせず，この性質を使っていく

本日の内容

1. いろいろな半順序
2. $1 \mid$ **sp-graph** $\mid \sum w_j C_j$ **のアルゴリズム**
3. $P \mid$ in-tree, $p_j = 1 \mid C_{\max}$ のアルゴリズム

- E. L. Lawler, Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics* 2 (1978) 75–90.
- M. X. Goemans, D. P. Williamson, Two-dimensional Gantt charts and a scheduling algorithm of Lawler. *SIAM Journal on Discrete Mathematics* 13 (2000) pp. 281–294.

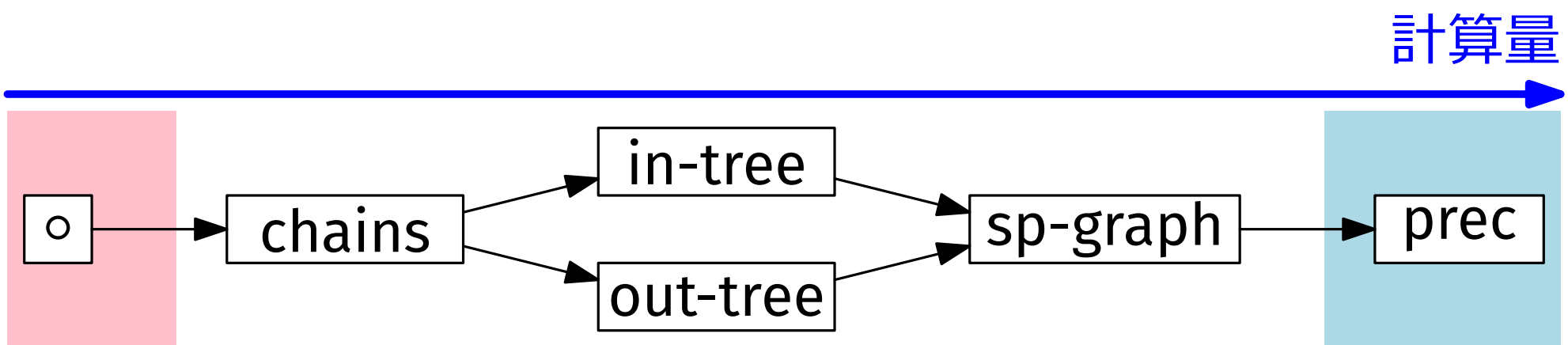
一機械，先行制約，重み付き総完了時刻

定理 (Lawler '78)

問題 $1 \mid \text{sp-graph} \mid \sum w_j C_j$ は強多項式時間で解ける

復習

- $1 \mid\mid \sum w_j C_j$ P (Smith '56)
- $1 \mid \text{prec} \mid \sum w_j C_j$ 強 NP 困難 (Lawler '78)



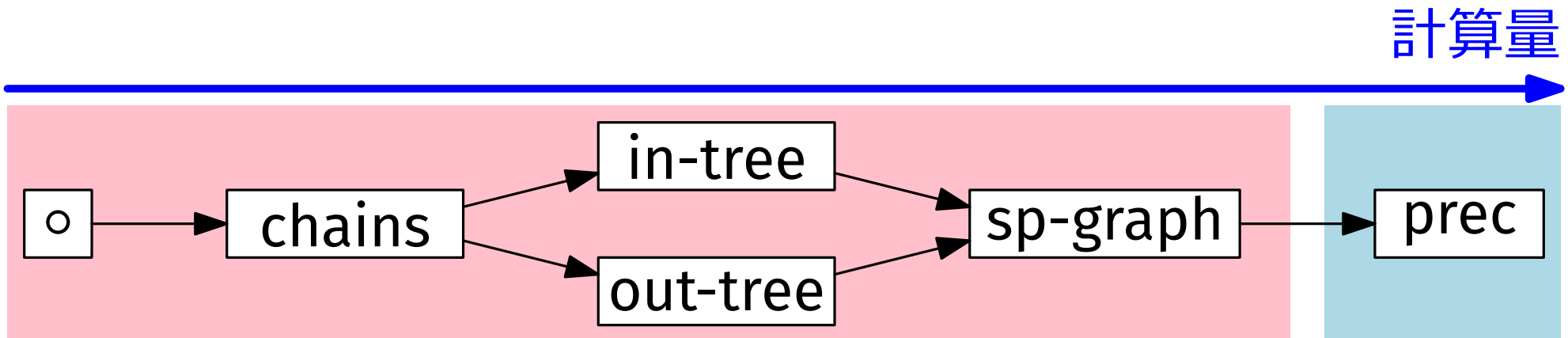
一機械，先行制約，重み付き総完了時刻

定理 (Lawler '78)

問題 $1 \mid \text{sp-graph} \mid \sum w_j C_j$ は強多項式時間で解ける

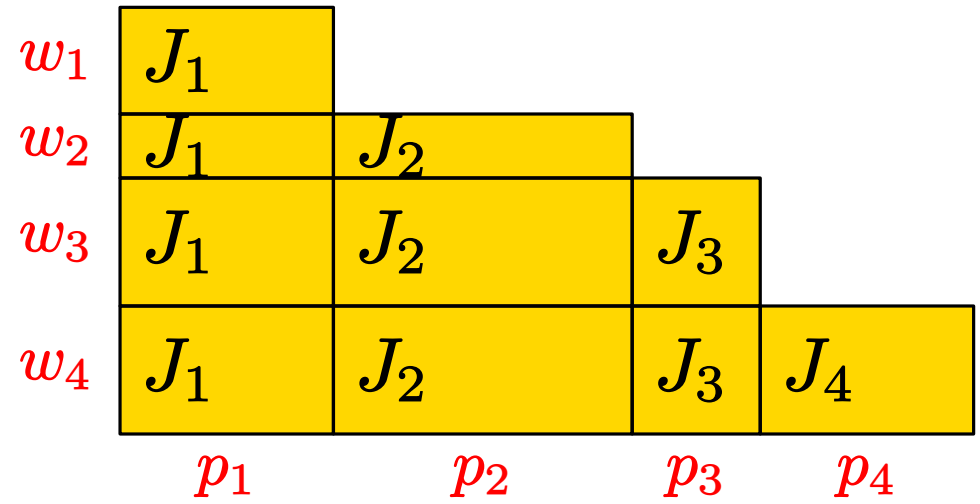
復習

- $1 \mid\mid \sum w_j C_j$ P (Smith '56)
- $1 \mid \text{prec} \mid \sum w_j C_j$ 強 NP 困難 (Lawler '78)

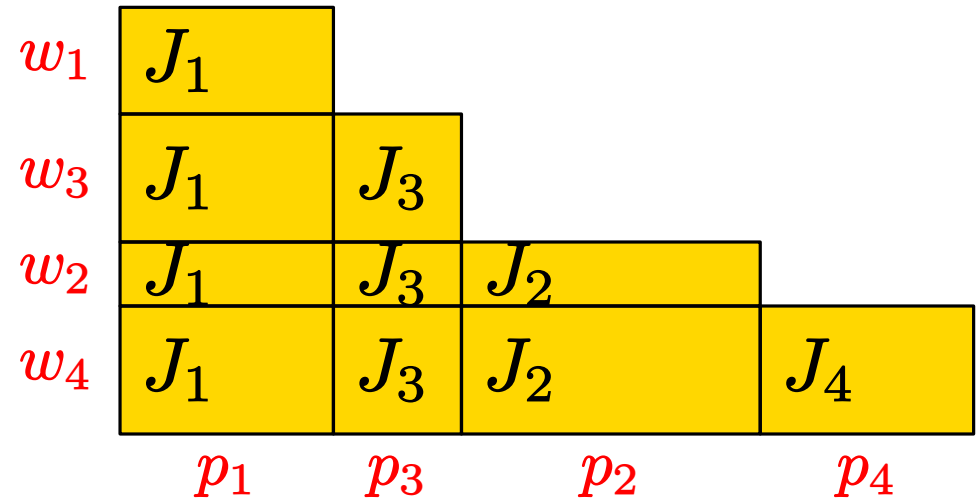


[復習] $1 \mid\mid \sum w_j C_j$：考え方

J_1	J_2	J_3	J_4

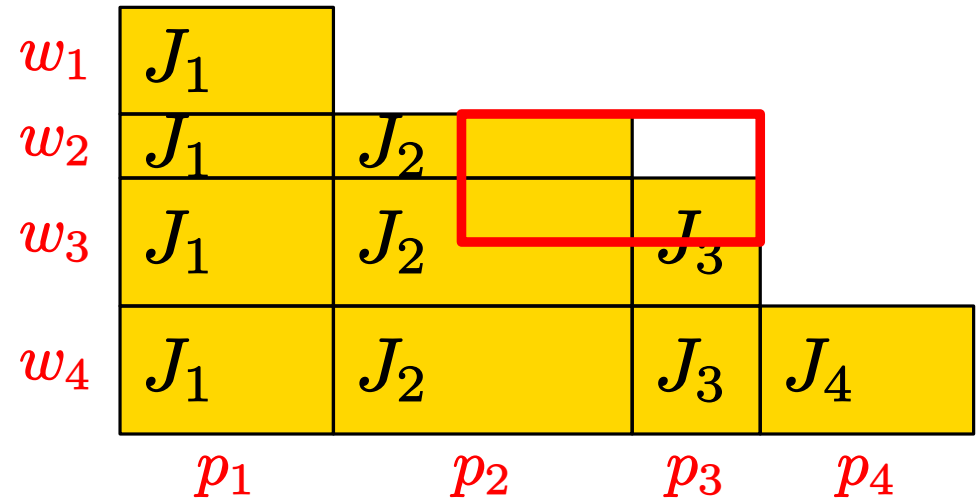


J_1	J_3	J_2	J_4

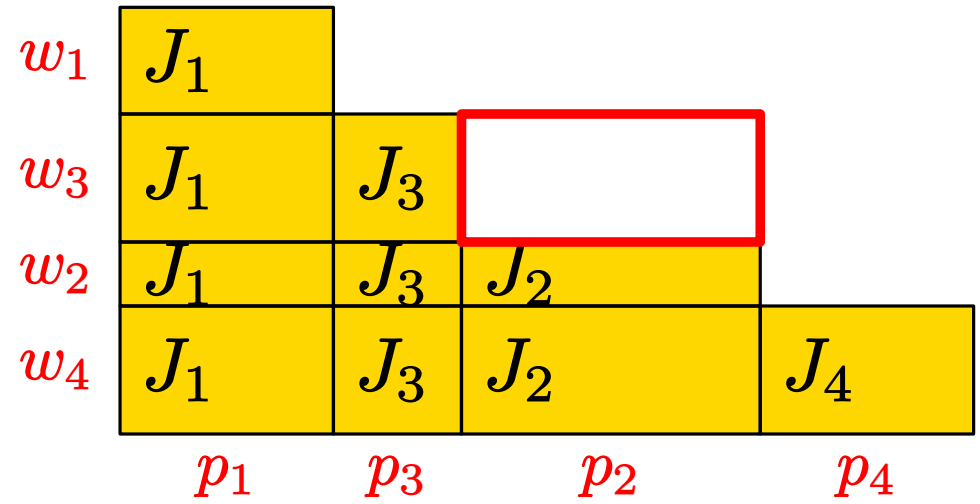


[復習] $1 \mid\mid \sum w_j C_j$：考え方

J_1	J_2	J_3	J_4

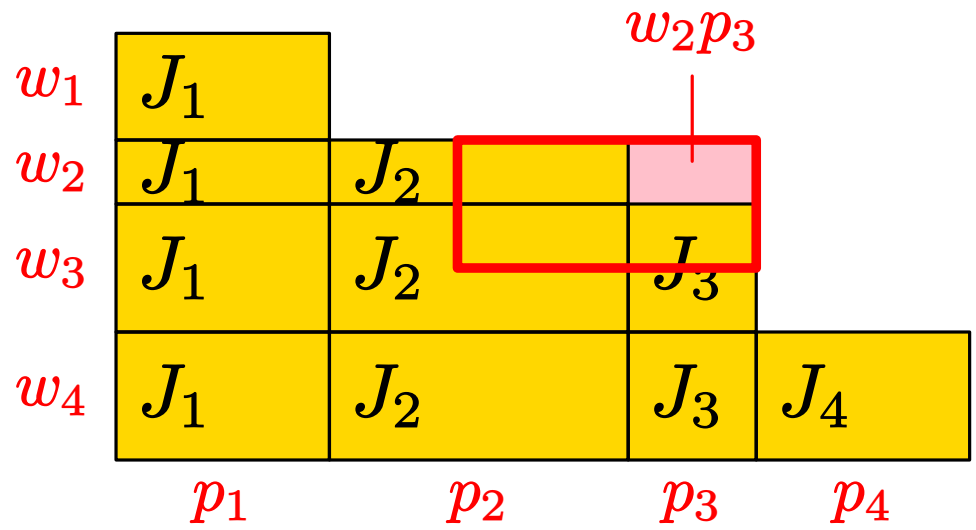


J_1	J_3	J_2	J_4

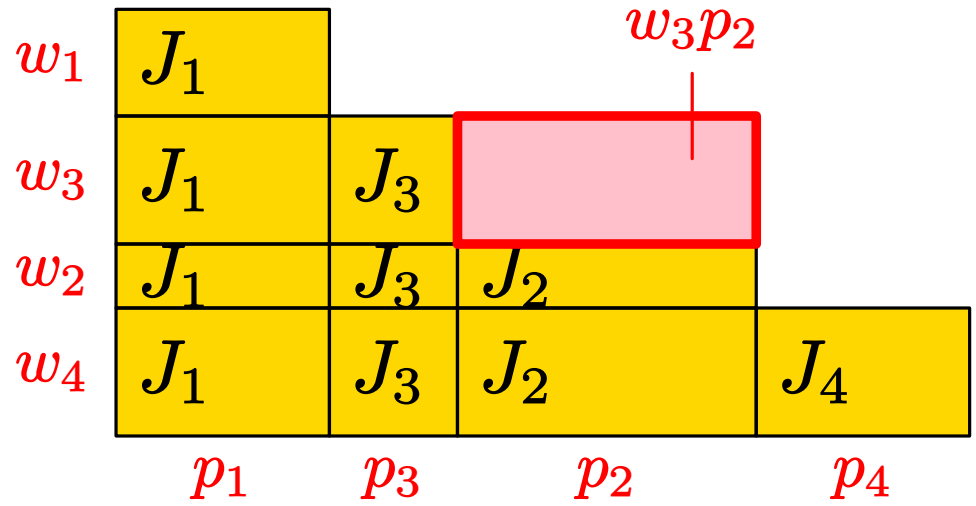


[復習] $1 \mid\mid \sum w_j C_j$：考え方

J_1	J_2	J_3	J_4

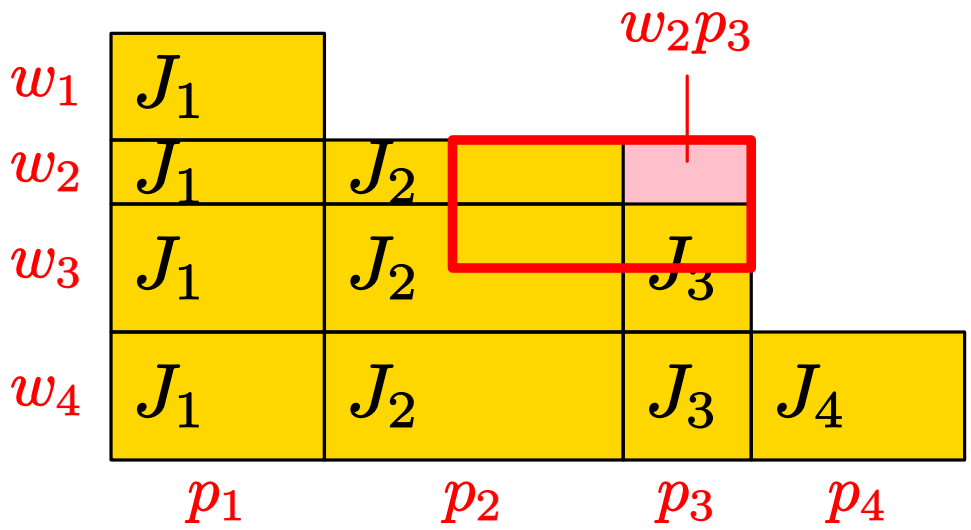


J_1	J_3	J_2	J_4

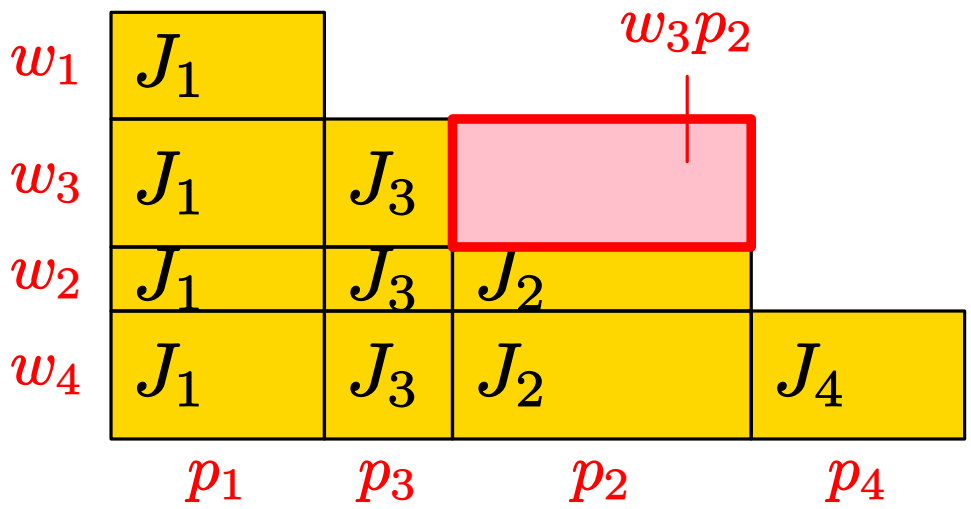


[復習] $1 \mid\mid \sum w_j C_j$：考え方

J_1	J_2	J_3	J_4



J_1	J_3	J_2	J_4

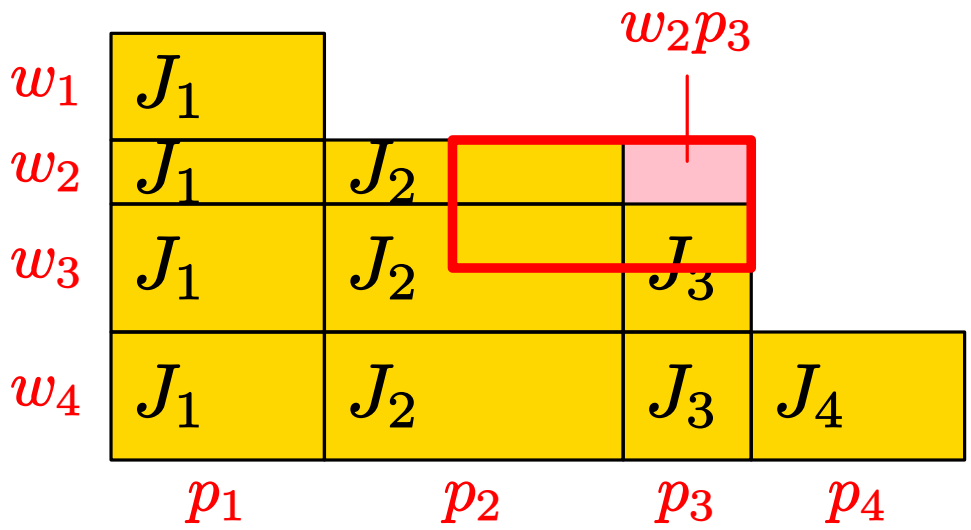


$w_3p_2 - w_2p_3 > 0 \Rightarrow$

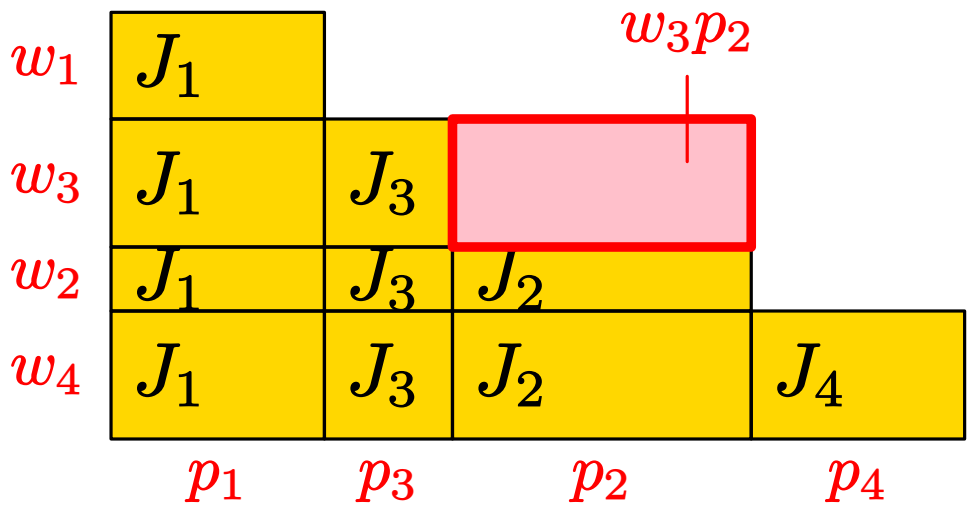
J_3 を J_2 の前に処理したほうがよい

[復習] $1 \mid\mid \sum w_j C_j$：考え方

J_1	J_2	J_3	J_4



J_1	J_3	J_2	J_4



$$\left(\frac{p_3}{w_3} < \frac{p_2}{w_2}\right)$$

$$w_3 p_2 - w_2 p_3 > 0 \Rightarrow$$

J_3 を J_2 の前に処理したほうがよい

[復習] $1 \mid\mid \sum w_j C_j$：アルゴリズム

定理：$1 \mid\mid \sum w_j C_j$ のアルゴリズム (Smith '56)

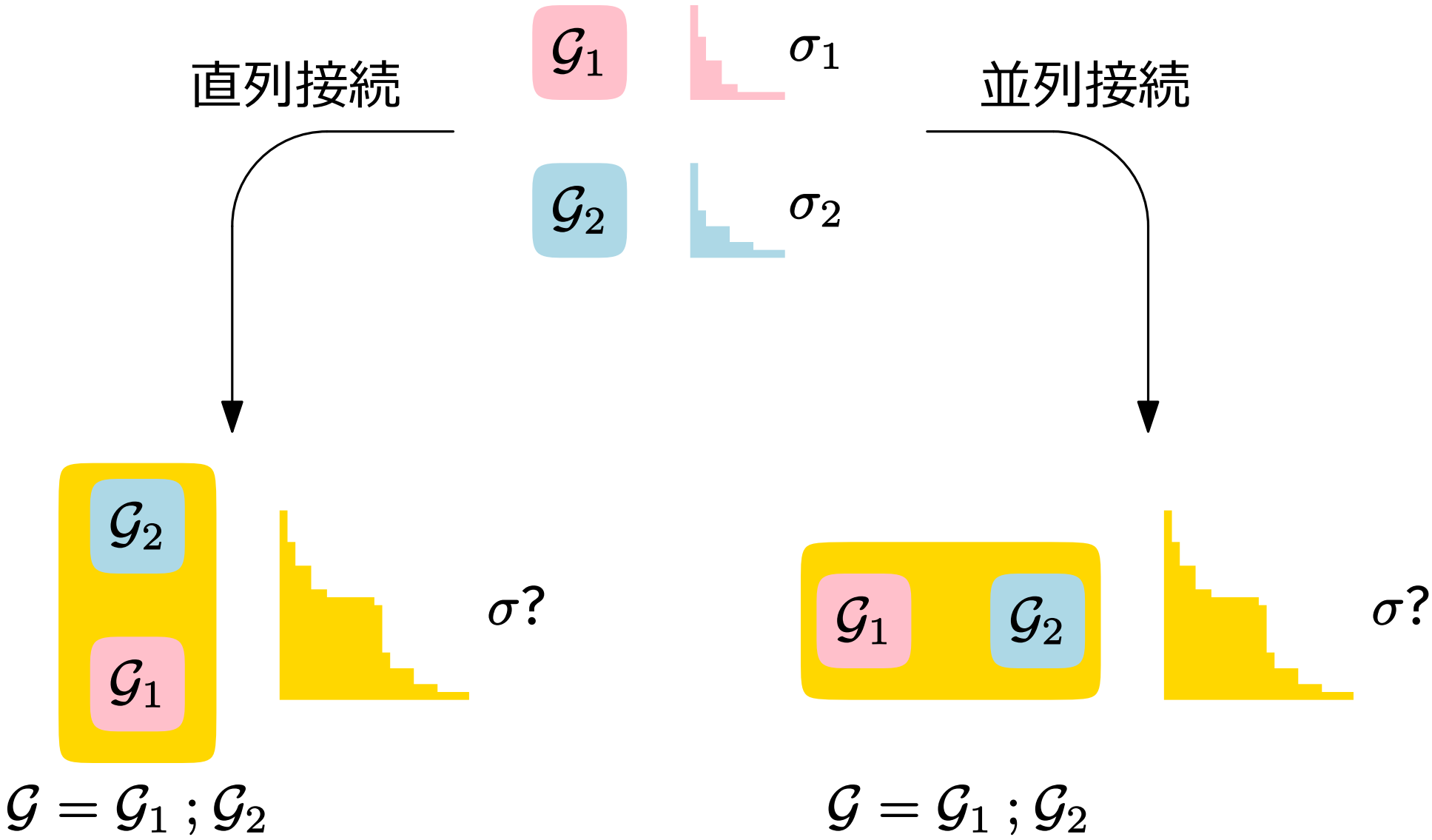
$1 \mid\mid \sum w_j C_j$ に対して，次のアルゴリズムは最適解を与える

アルゴリズム：重み付き最短処理時間優先規則 (WSPT)

1. $\dfrac{p_j}{w_j}$ が小さい順にジョブを並べる
2. その順に従ってジョブを処理する

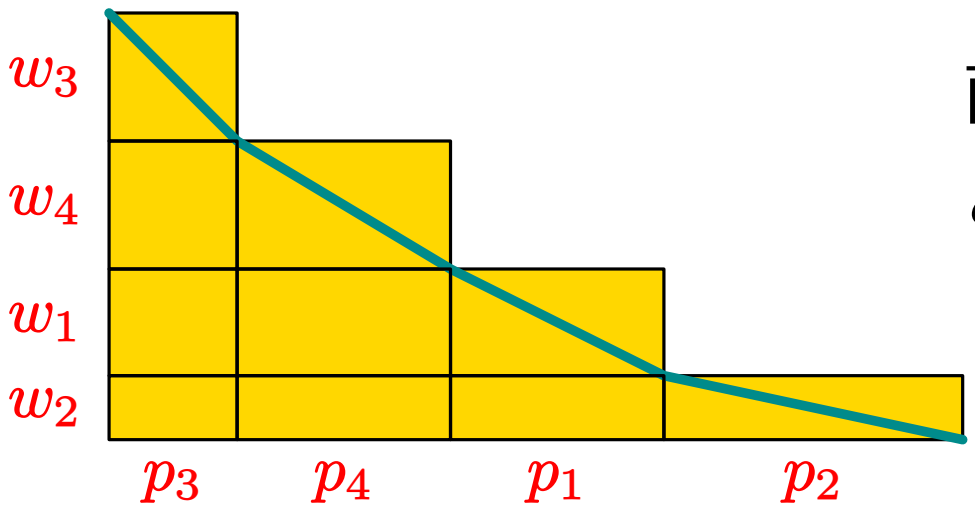
WSPT = Weighted Shortest Processing Time

直列接続，並列接続で 最適スケジュールを作り上げていく



WSPT：$\frac{p_j}{w_j}$ が小さい順に並べる

傾きが単調減少 ⇒ 最適解

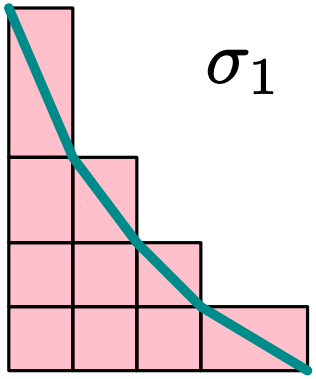


直列接続・並列接続で
この形を保ちたい

(Goemans, Williamson '00)

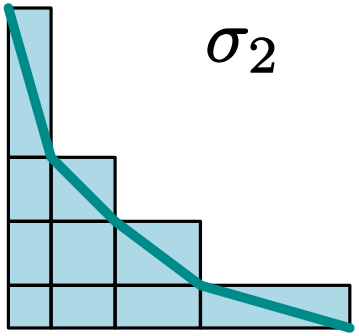
並列接続

$\mathcal{G}_1$



σ_1

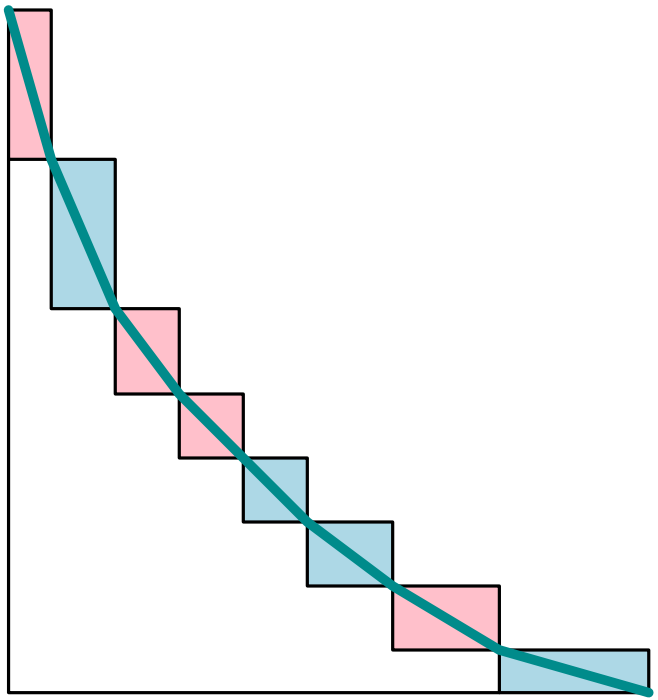
$\mathcal{G}_2$



σ_2

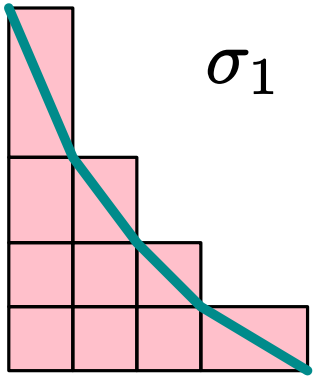
$\mathcal{G}_1$ $\mathcal{G}_2$

$\mathcal{G} = \mathcal{G}_1 \,;\, \mathcal{G}_2$



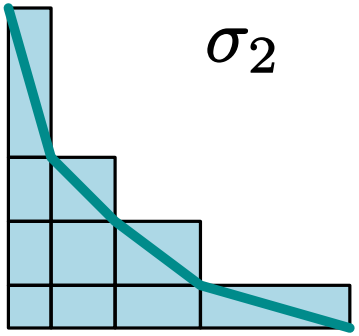
直列接続

$\mathcal{G}_1$



σ_1

$\mathcal{G}_2$

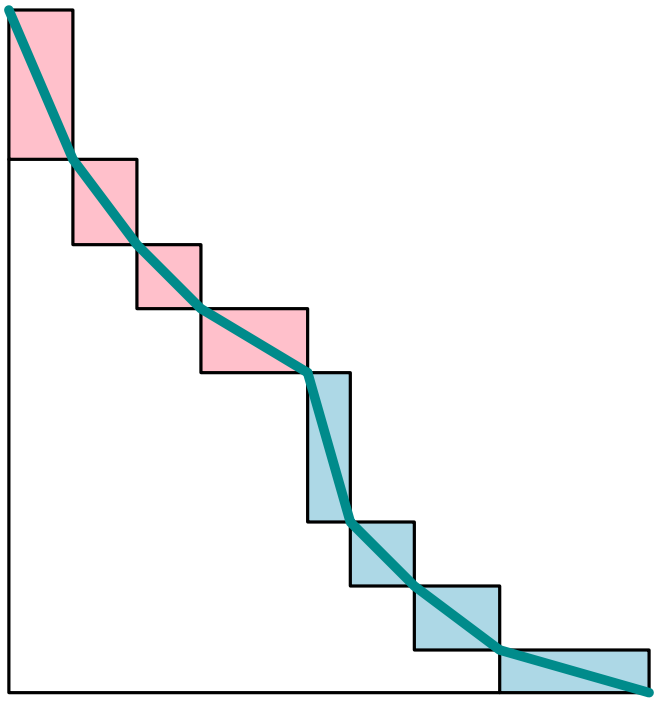


σ_2

$\mathcal{G}_2$

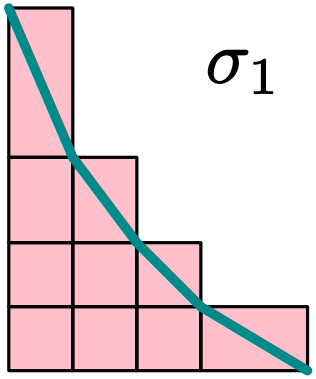
$\mathcal{G}_1$

$\mathcal{G} = \mathcal{G}_1 ; \mathcal{G}_2$



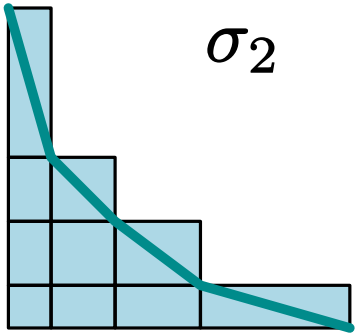
直列接続

$\mathcal{G}_1$



σ_1

$\mathcal{G}_2$

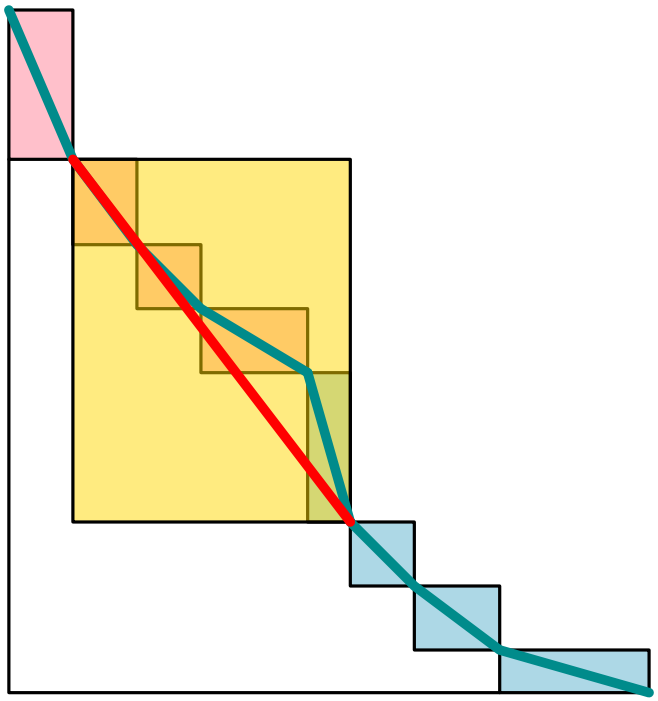


σ_2

$\mathcal{G}_2$

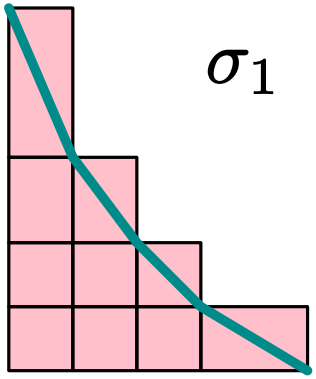
$\mathcal{G}_1$

$\mathcal{G} = \mathcal{G}_1 ; \mathcal{G}_2$



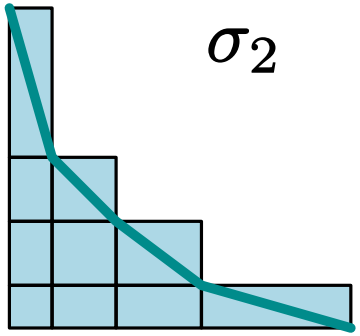
直列接続

$\mathcal{G}_1$



σ_1

$\mathcal{G}_2$

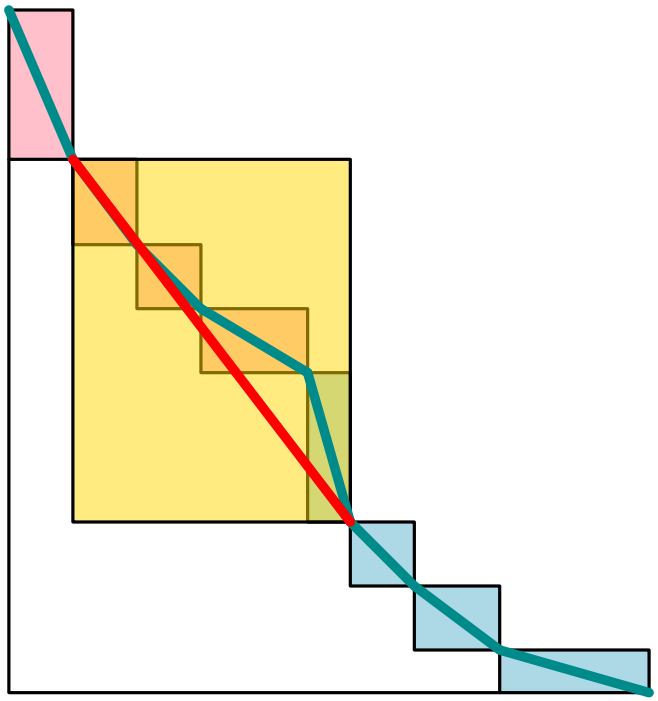


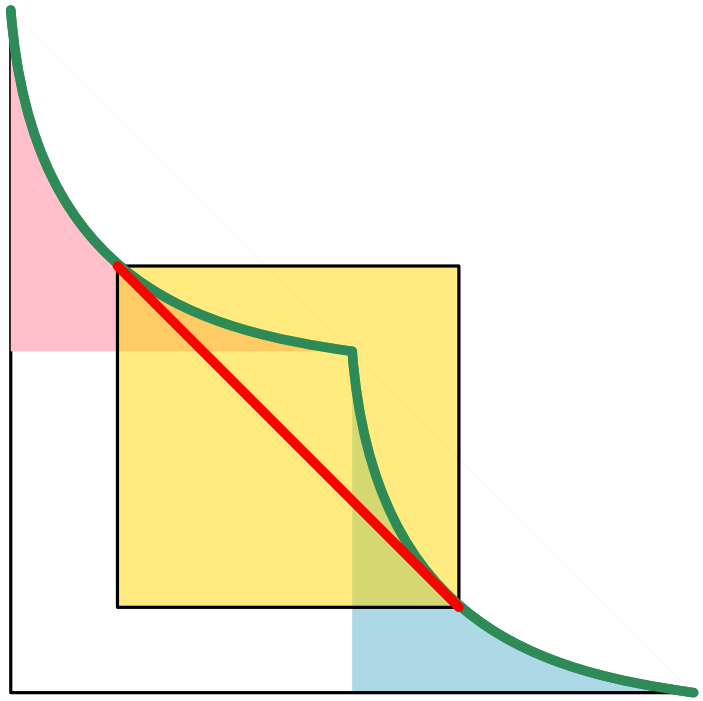
σ_2

$\mathcal{G}_2$

$\mathcal{G}_1$

$\mathcal{G} = \mathcal{G}_1 ; \mathcal{G}_2$



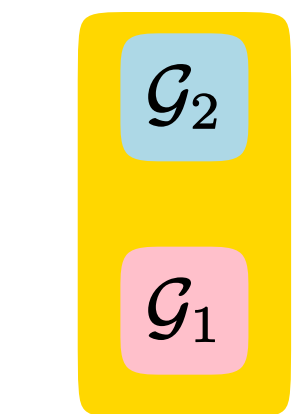


直列接続：補題

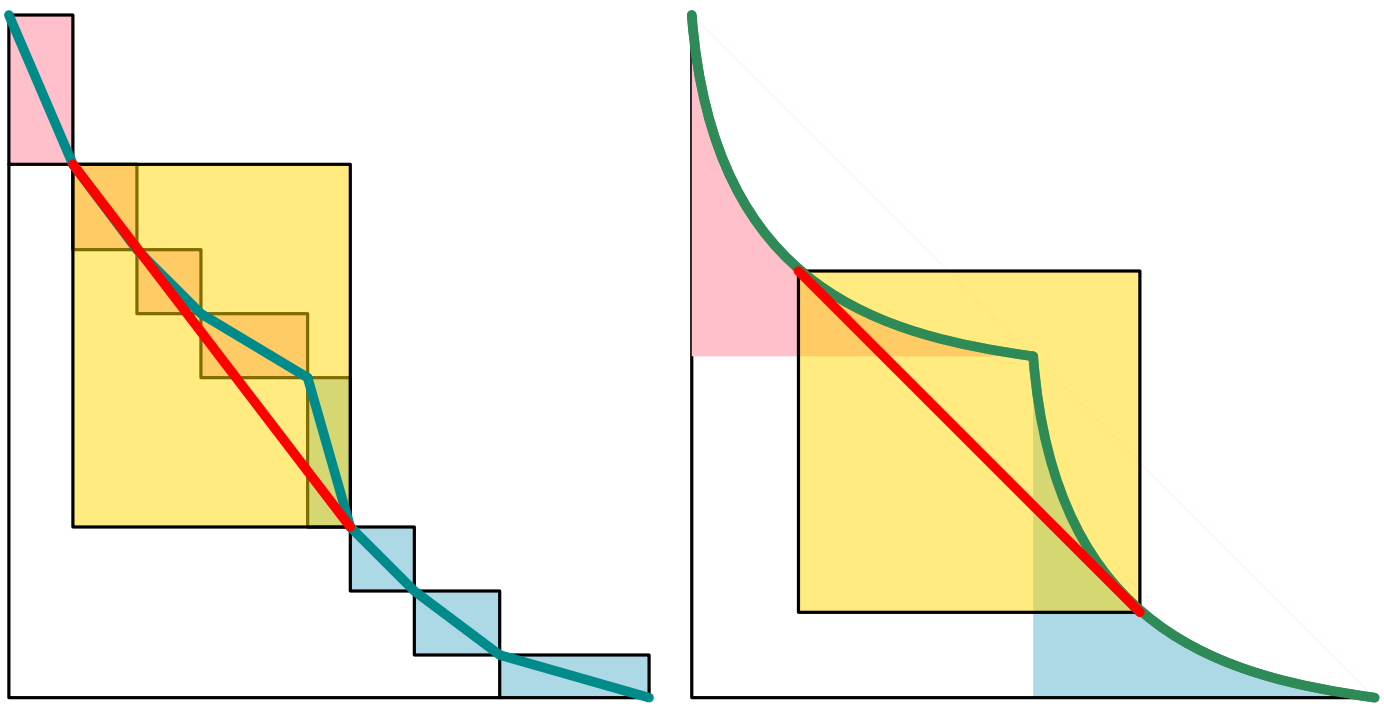
補題

直列接続で，前ページのように「まとめた」ジョブを連続で処理する最適解が存在する

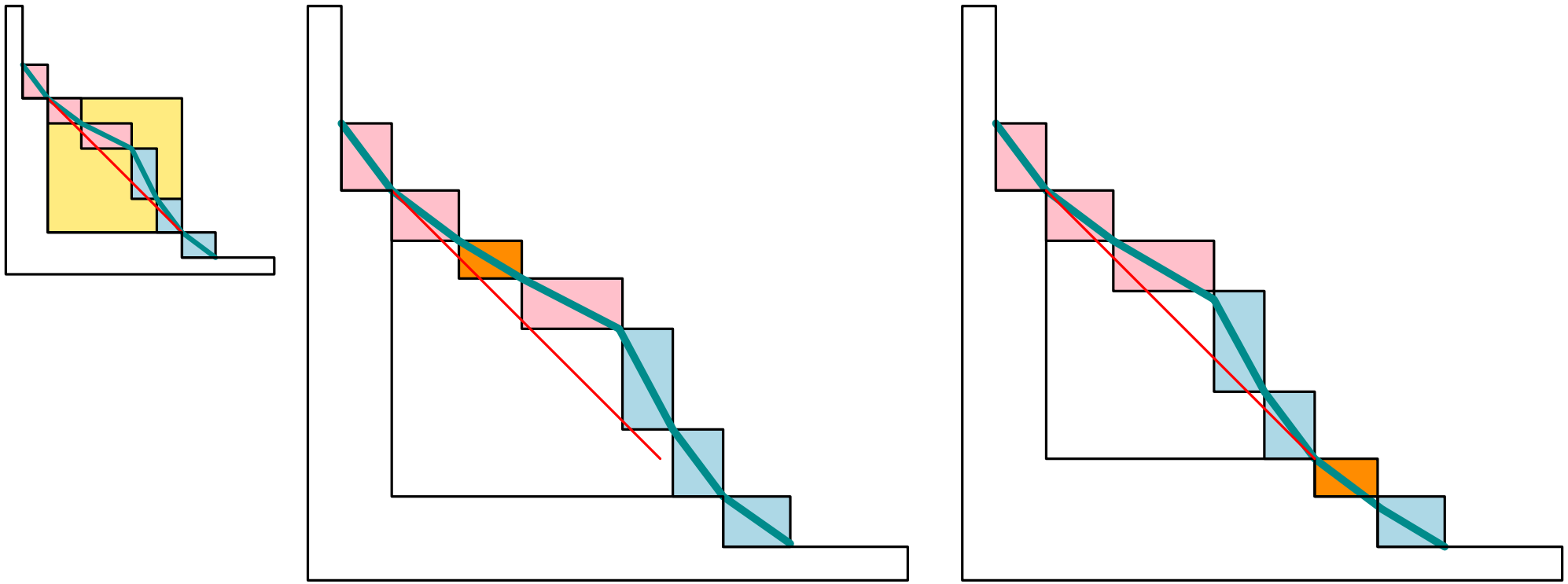
これを証明すれば，アルゴリズムの正しさが分かる



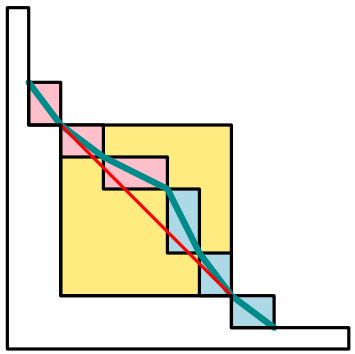
$\mathcal{G} = \mathcal{G}_1 ; \mathcal{G}_2$

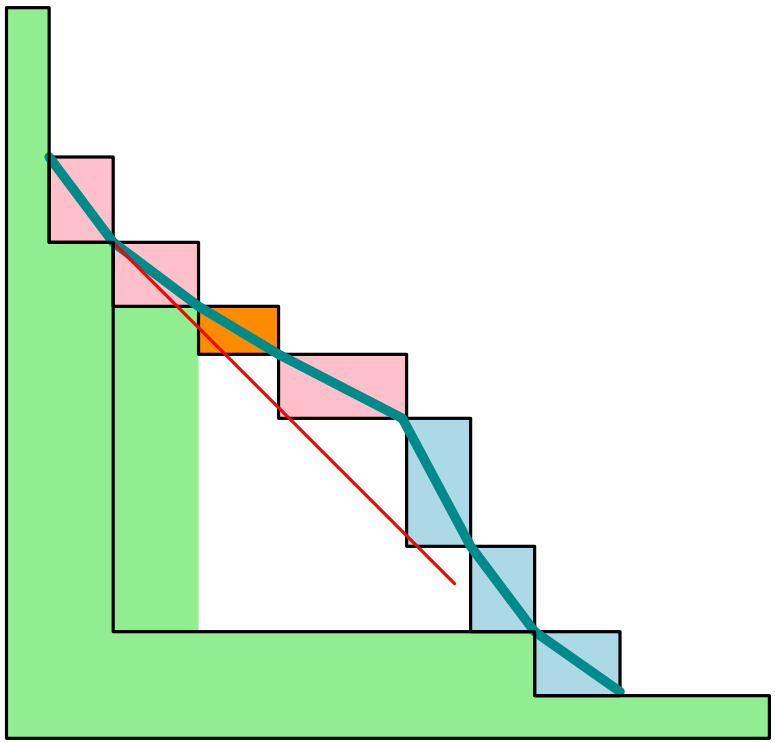


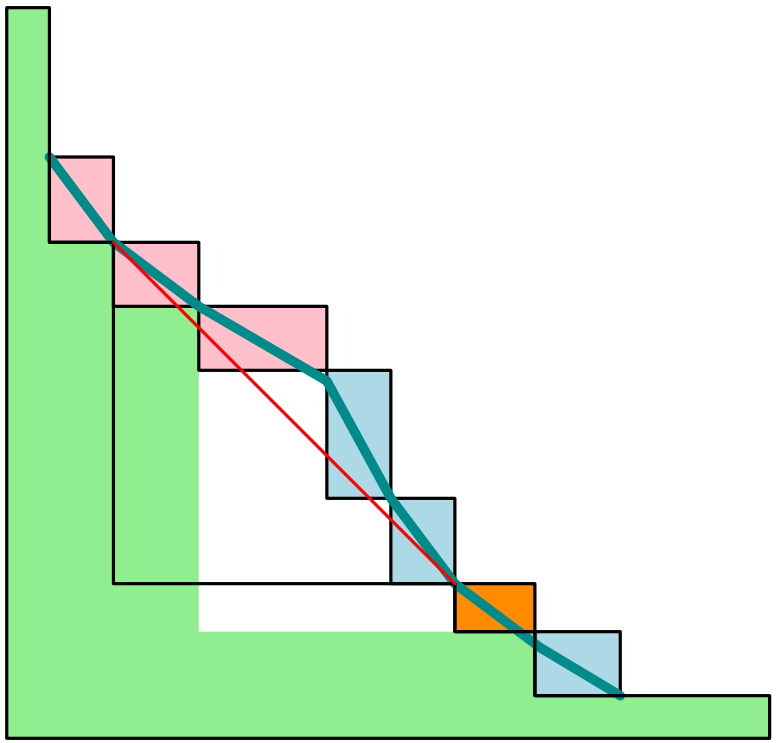
直列接続：補題の証明 (1)



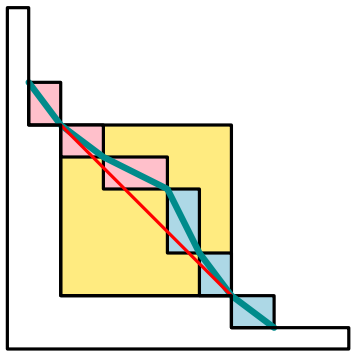
直列接続：補題の証明 (1)

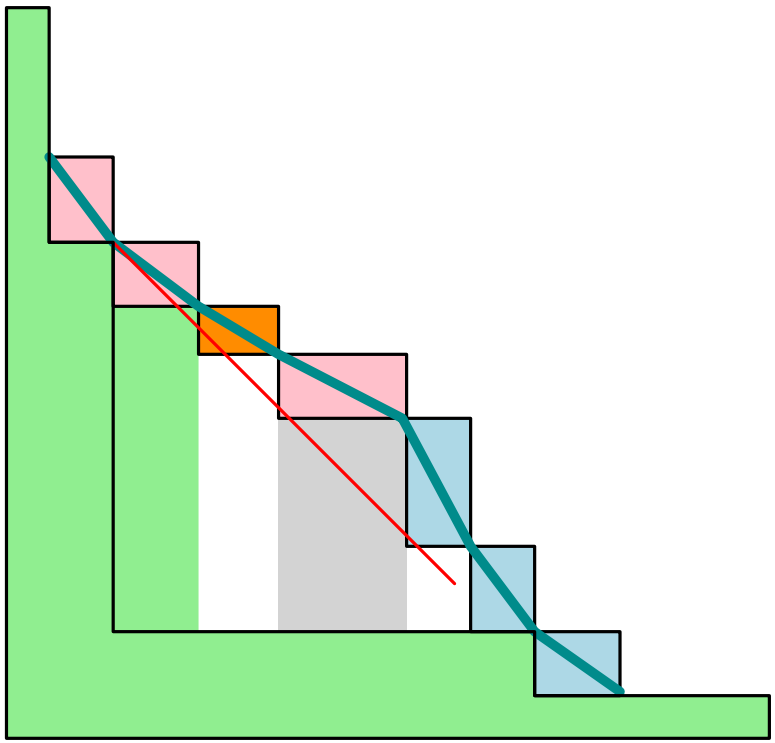


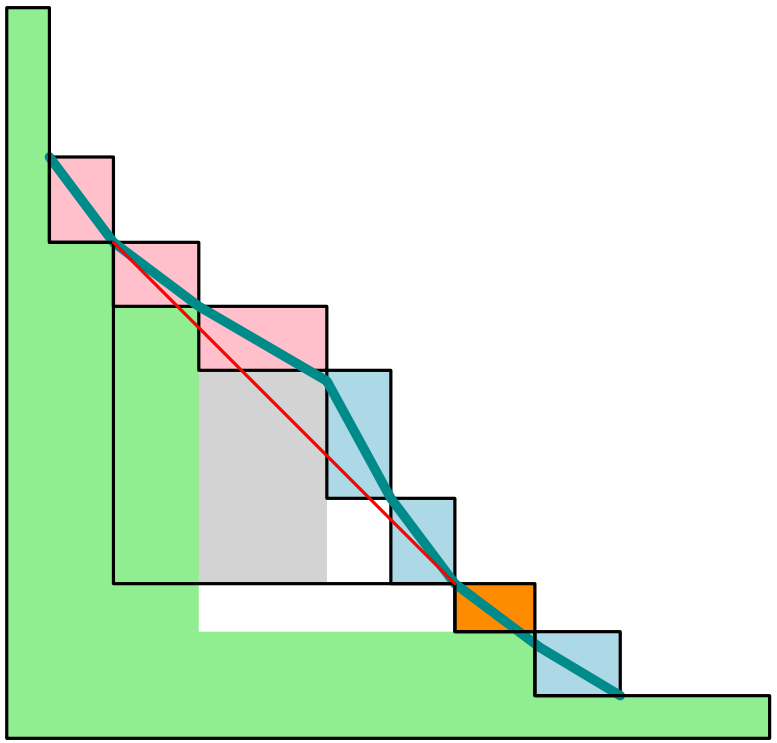




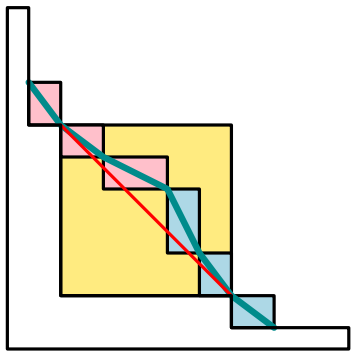
直列接続：補題の証明 (1)

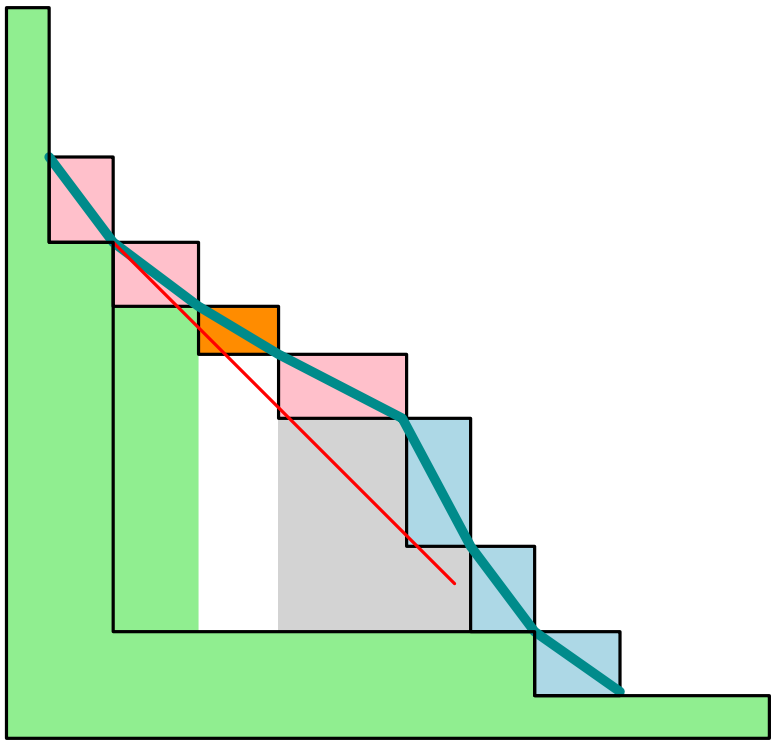


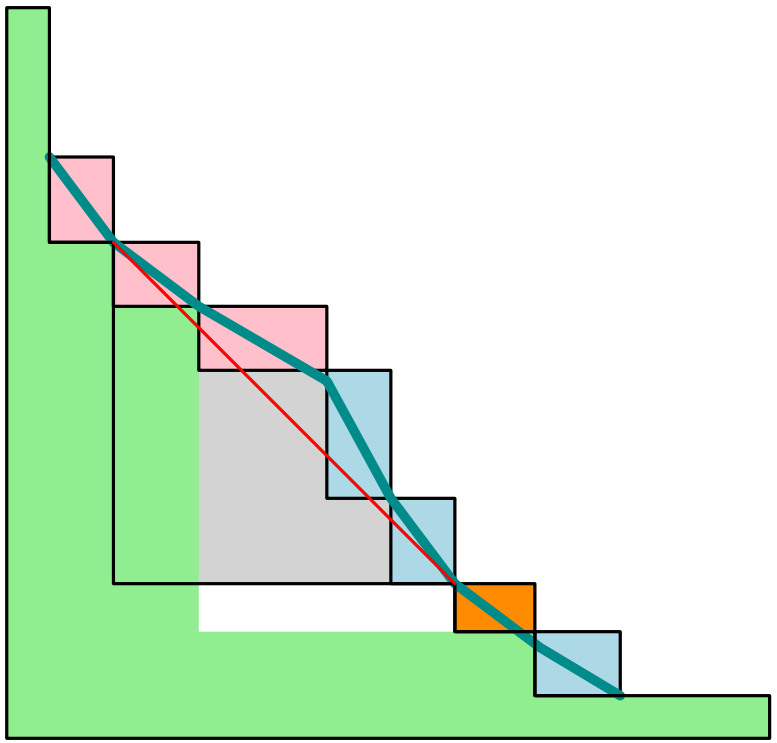




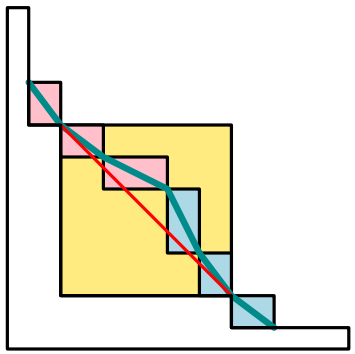
直列接続：補題の証明 (1)

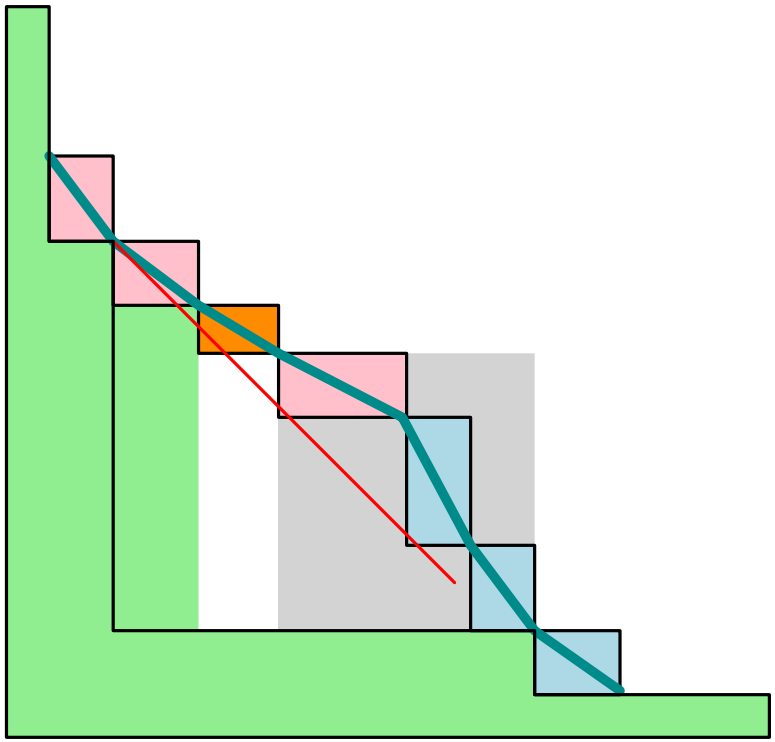


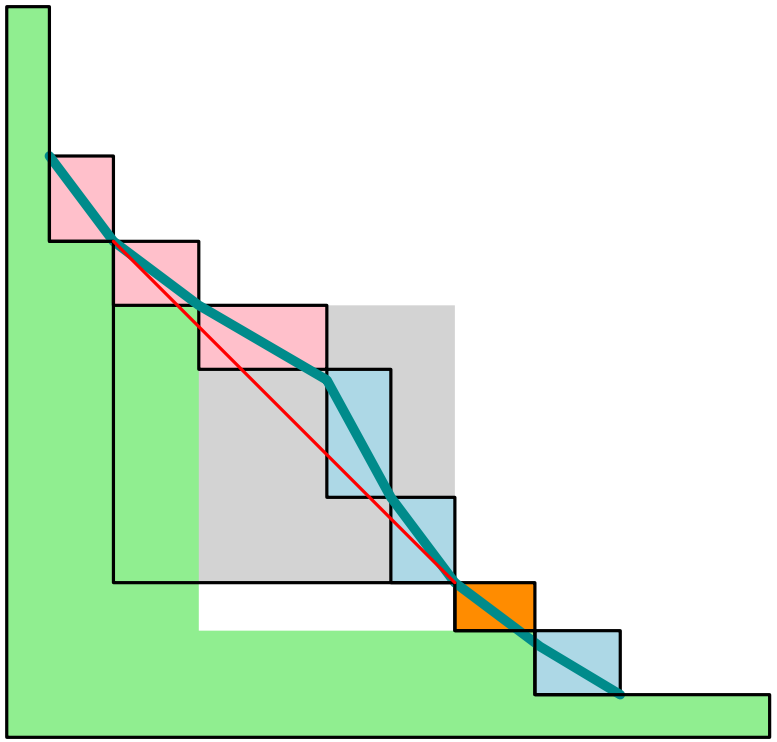




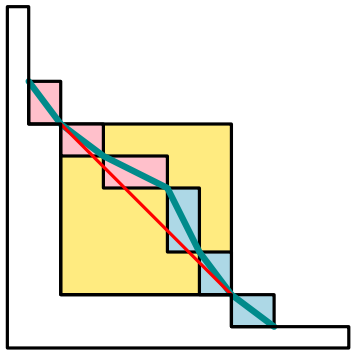
直列接続：補題の証明 (1)

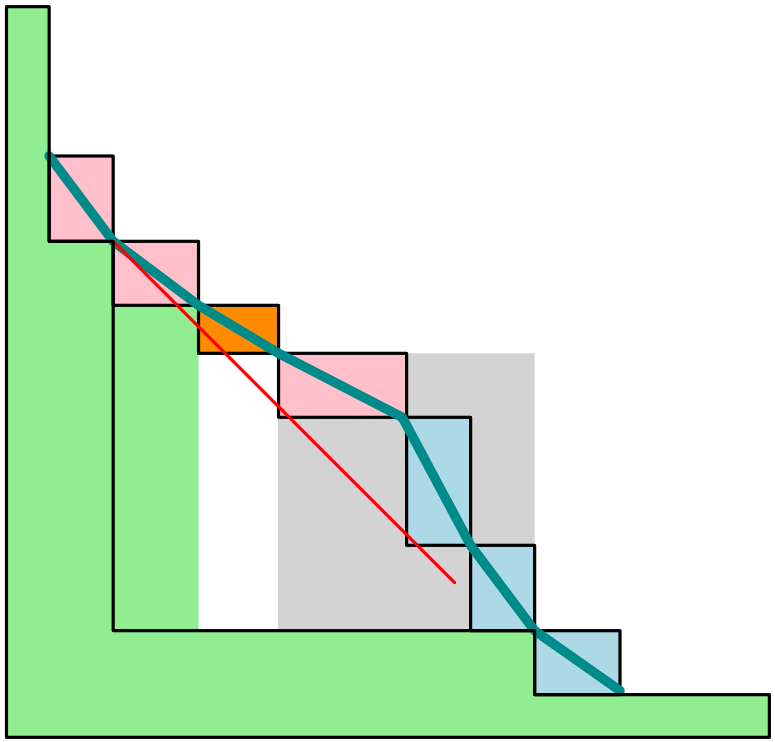


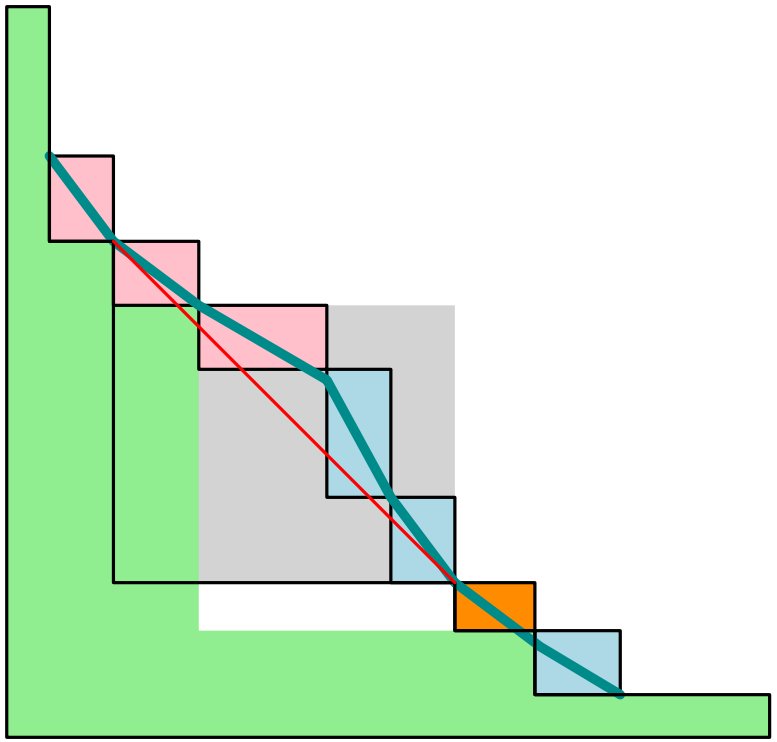


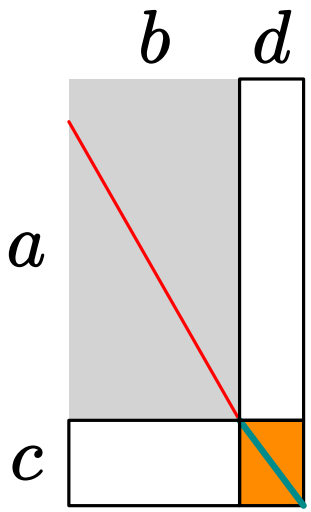


直列接続：補題の証明 (1)





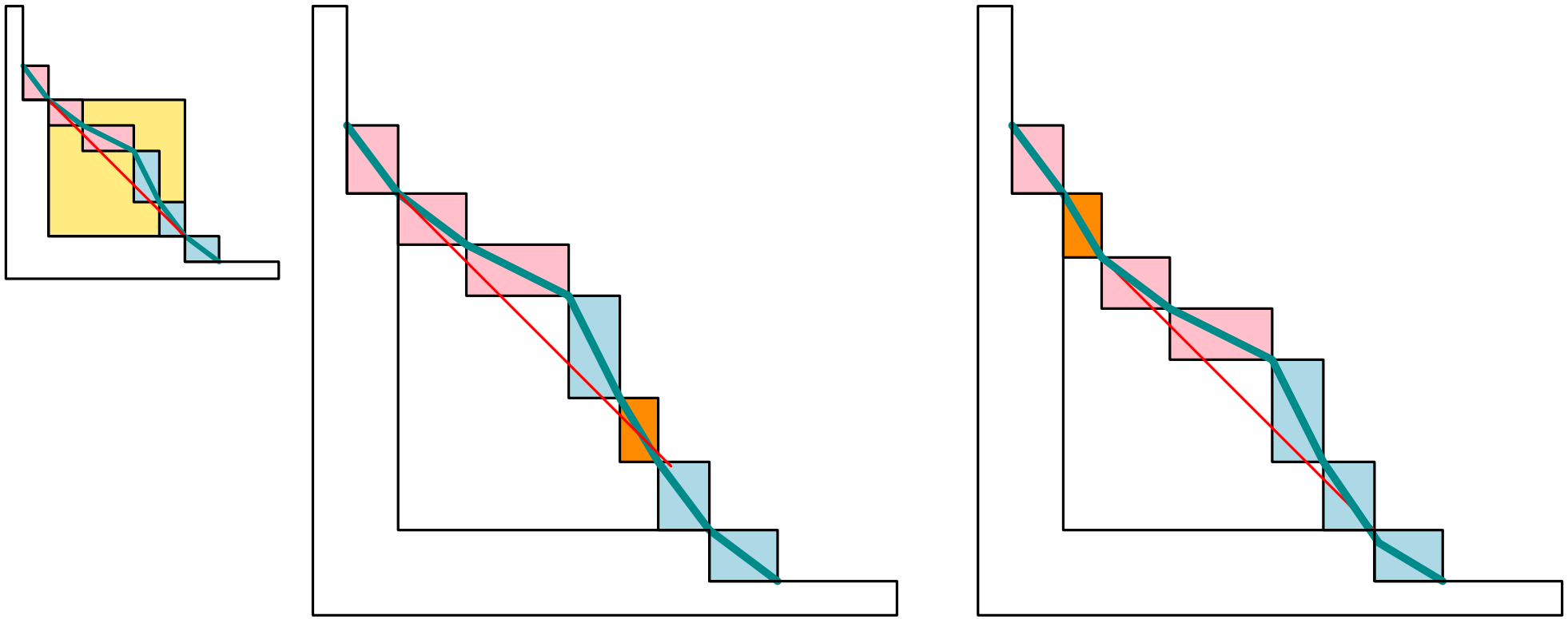




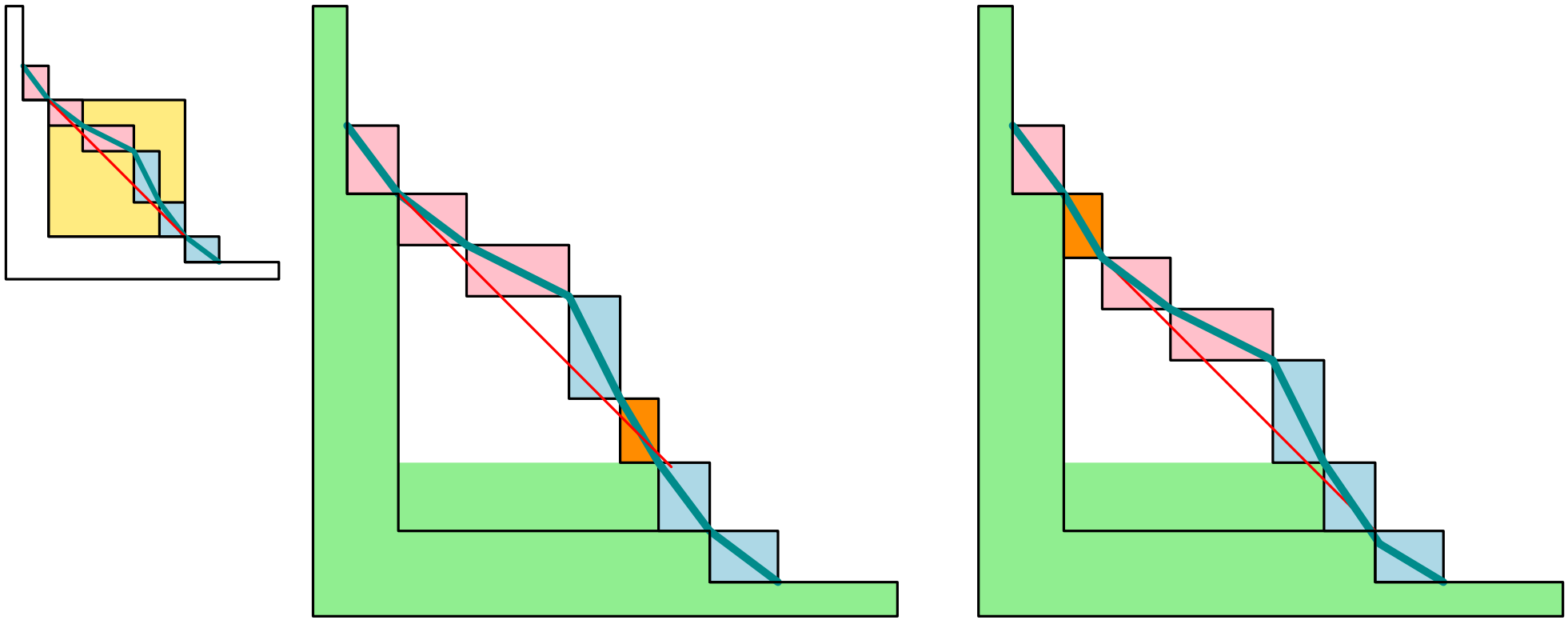
傾きの大小 $\frac{a}{b} > \frac{c}{d}$

$\therefore$ 面積の大小 $ad > cb$

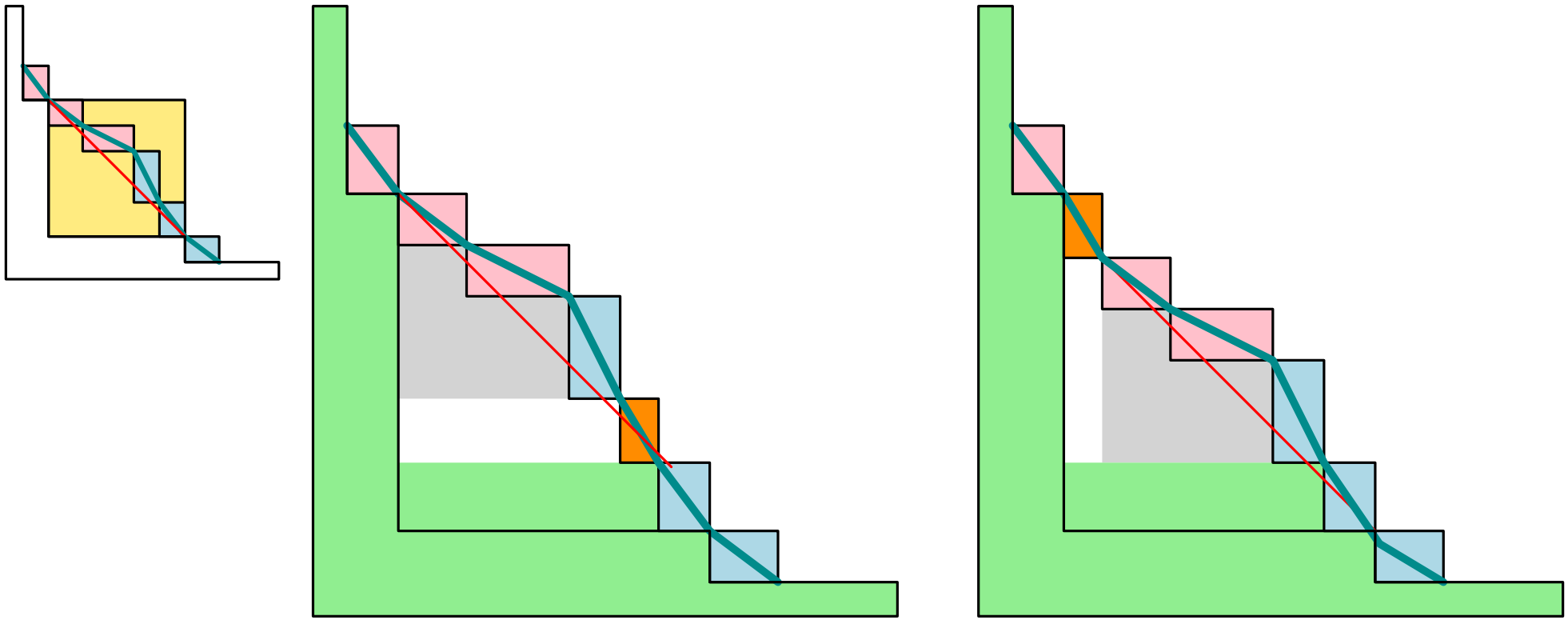
直列接続：補題の証明 (2)



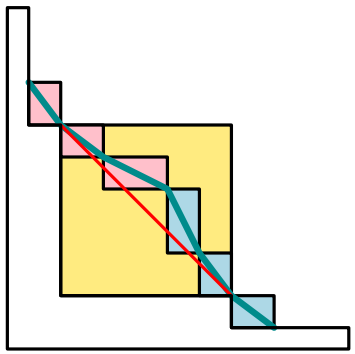
直列接続：補題の証明 (2)

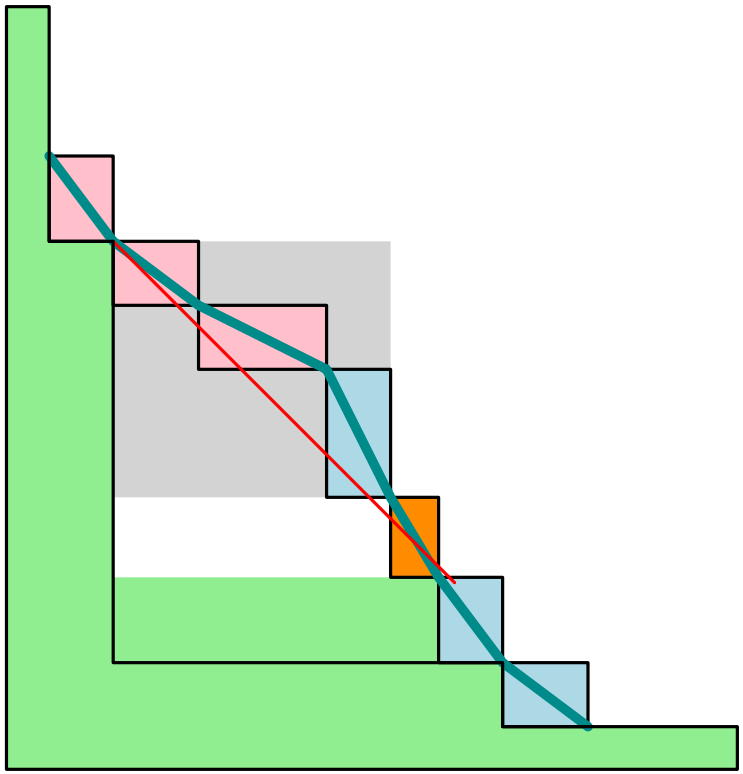


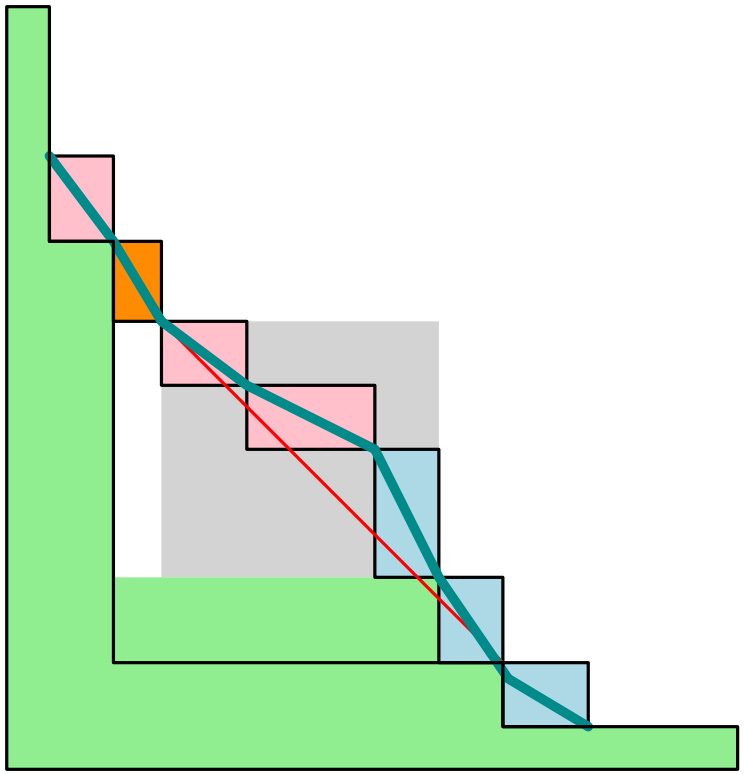
直列接続：補題の証明 (2)



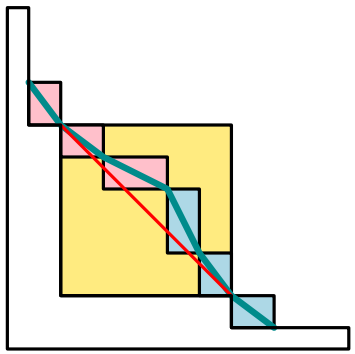
直列接続：補題の証明 (2)

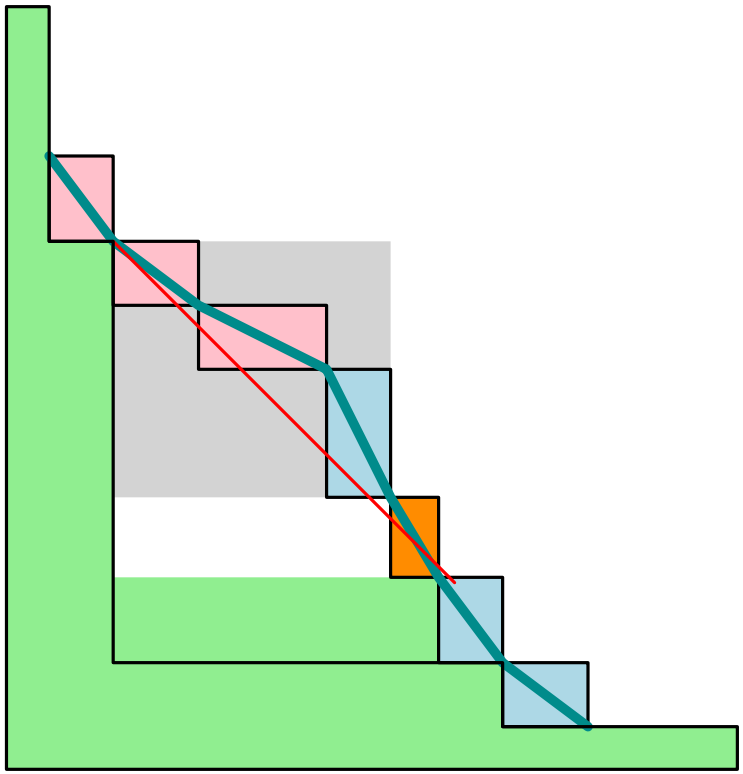


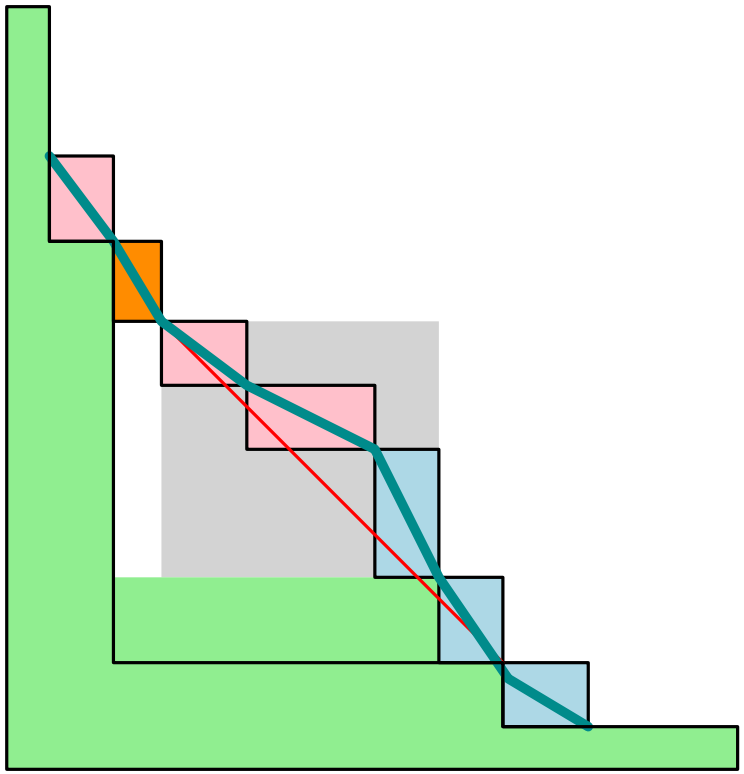


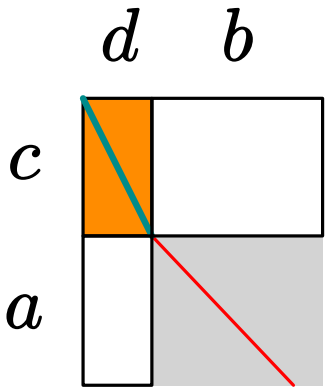


直列接続：補題の証明 (2)









傾きの大小 $\frac{a}{b} < \frac{c}{d}$

$\therefore$ 面積の大小 $ad < cb$

□

本日の内容

1. いろいろな半順序
2. 1 | sp-graph | $\sum w_j C_j$ のアルゴリズム
3. P | **in-tree**, $p_j = 1$ | $C_{\max}$ **のアルゴリズム**

- T. C. Hu, Parallel sequencing and assembly line problems. *Operations Research* 9 (1961) pp. 841–848.
- M. R. Garey, D. S. Johnson, R. E. Tarjan, M. Yannakakis, Scheduling opposing forests. *SIAM Journal on Algebraic Discrete Methods* 4 (1983) pp. 72–93.

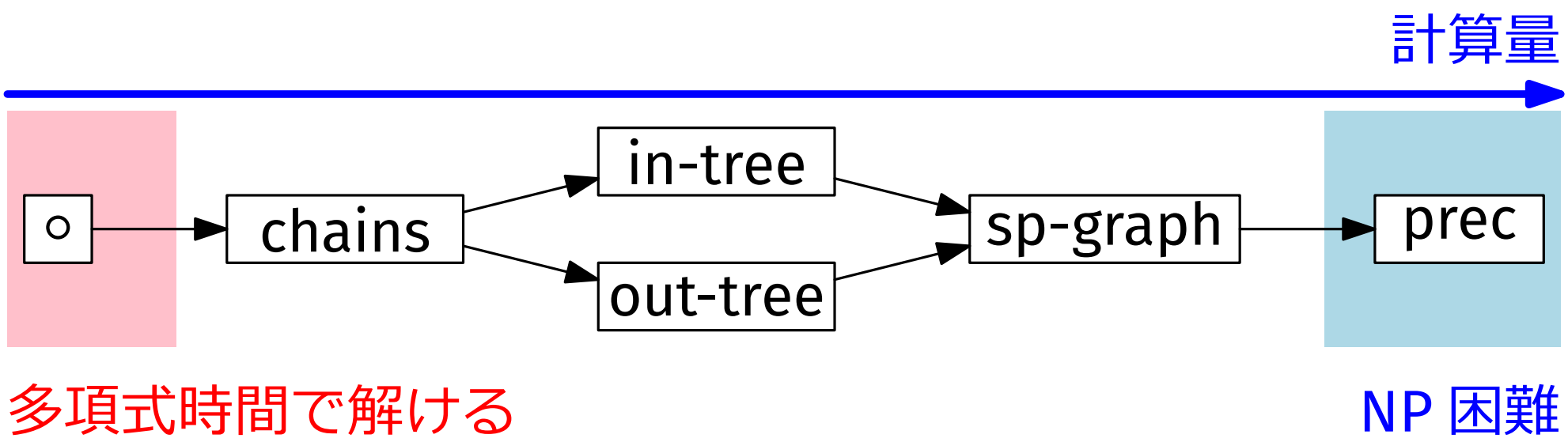
同一並列機械，同一処理時間，$C_{\max}$

定理 (Hu '61)

問題 $P \mid \text{in-tree}, p_j = 1 \mid C_{\max}$ は $O(n)$ で解ける

復習

- $P \mid p_j = 1 \mid C_{\max}$ P (簡単)
- $P \mid \text{prec}, p_j = 1 \mid C_{\max}$ NP 困難 (Lenstra, Rinnooy Kan '78)



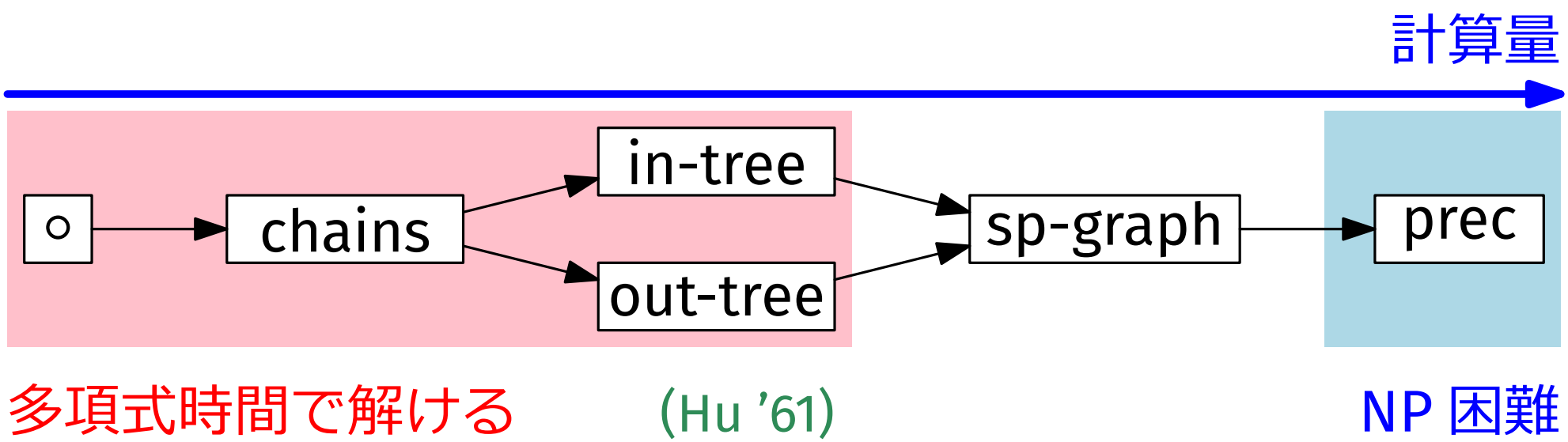
同一並列機械，同一処理時間，$C_{\max}$

定理 (Hu '61)

問題 $P \mid \text{in-tree}, p_j = 1 \mid C_{\max}$ は $O(n)$ で解ける

復習

- $P \mid p_j = 1 \mid C_{\max}$ P (簡単)
- $P \mid \text{prec}, p_j = 1 \mid C_{\max}$ NP 困難 (Lenstra, Rinnooy Kan '78)



同一並列機械，同一処理時間，$C_{\max}$

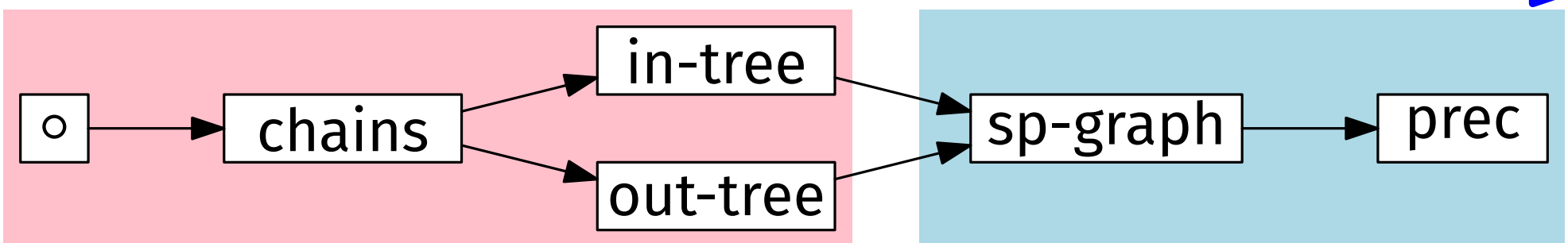
定理 (Hu '61)

問題 $P \mid \text{in-tree}, p_j = 1 \mid C_{\max}$ は $O(n)$ で解ける

復習

- $P \mid p_j = 1 \mid C_{\max}$ P (簡単)
- $P \mid \text{prec}, p_j = 1 \mid C_{\max}$ NP 困難 (Lenstra, Rinnooy Kan '78)

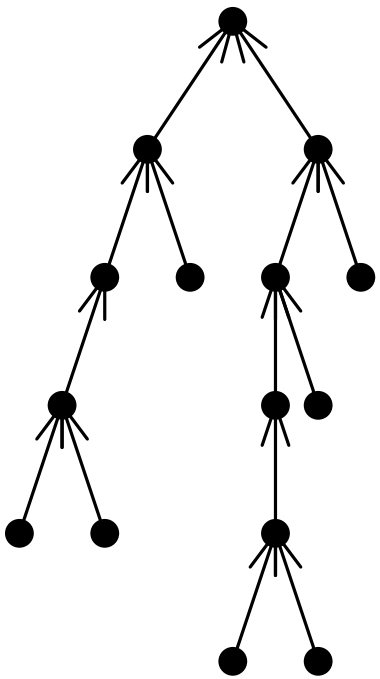
(Garey, Johnson, Tarjan, Yannakakis '83) 計算量



多項式時間で解ける (Hu '61) NP 困難

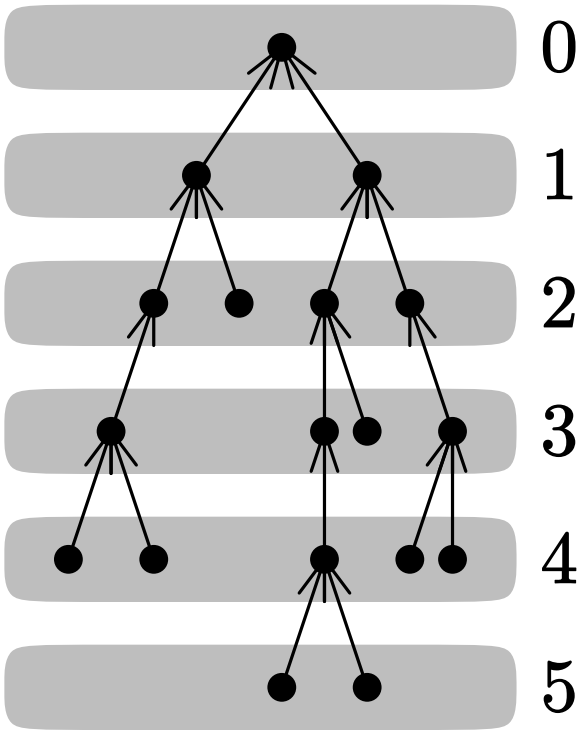
Hu のアルゴリズム

解く問題：$P \mid \text{in-tree}, p_j = 1 \mid C_{\max}$



Hu のアルゴリズム

解く問題：$P \mid \text{in-tree}, p_j = 1 \mid C_{\max}$

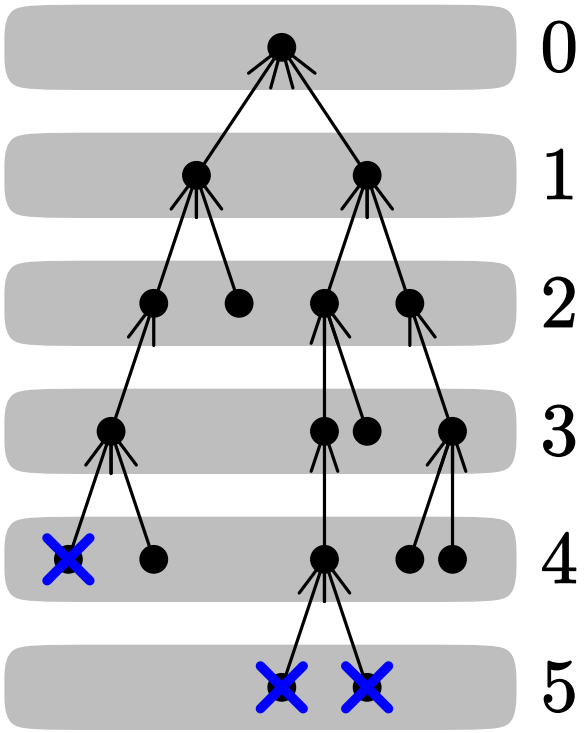


アルゴリズム

深いジョブを優先するリスト・スケジューリング

Hu のアルゴリズム

解く問題：$P \mid \text{in-tree}, p_j = 1 \mid C_{\max}$

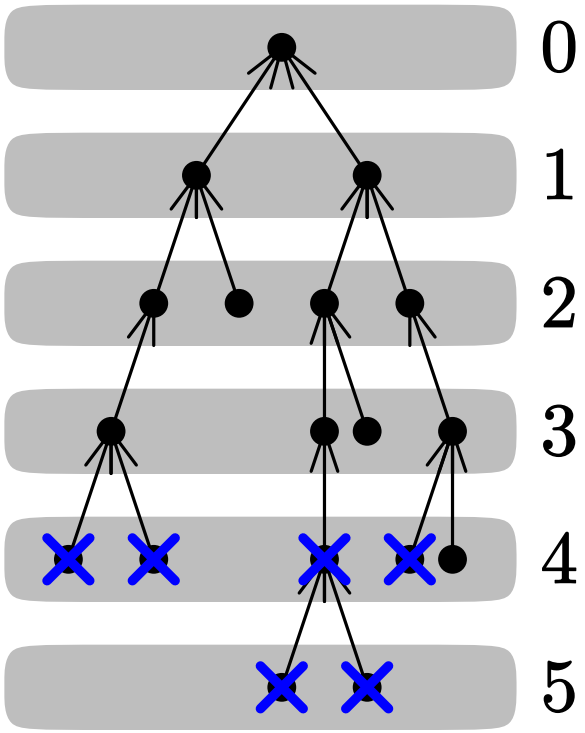


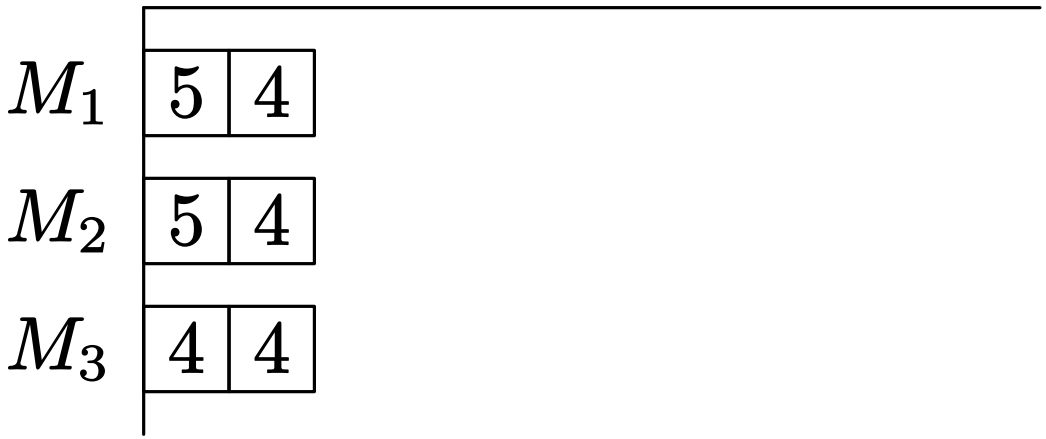
アルゴリズム

深いジョブを優先するリスト・スケジューリング

Hu のアルゴリズム

解く問題：$P \mid \text{in-tree}, p_j = 1 \mid C_{\max}$



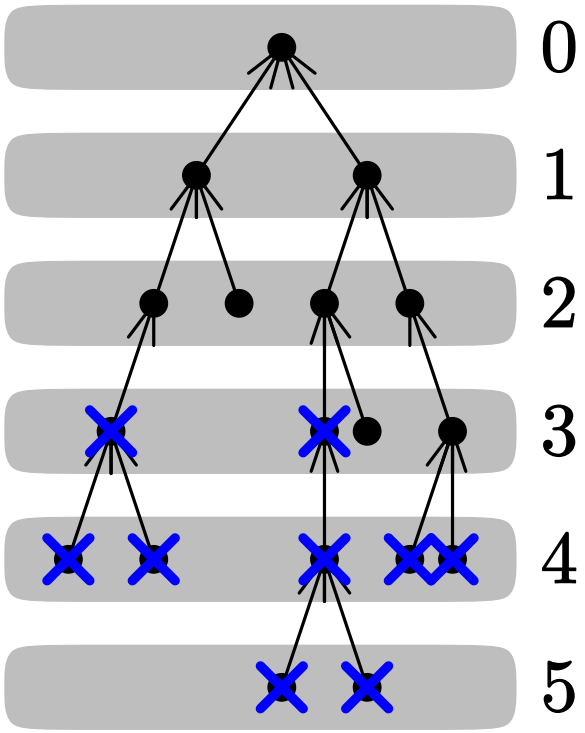


アルゴリズム

深いジョブを優先するリスト・スケジューリング

Hu のアルゴリズム

解く問題：$P \mid \text{in-tree}, p_j = 1 \mid C_{\max}$



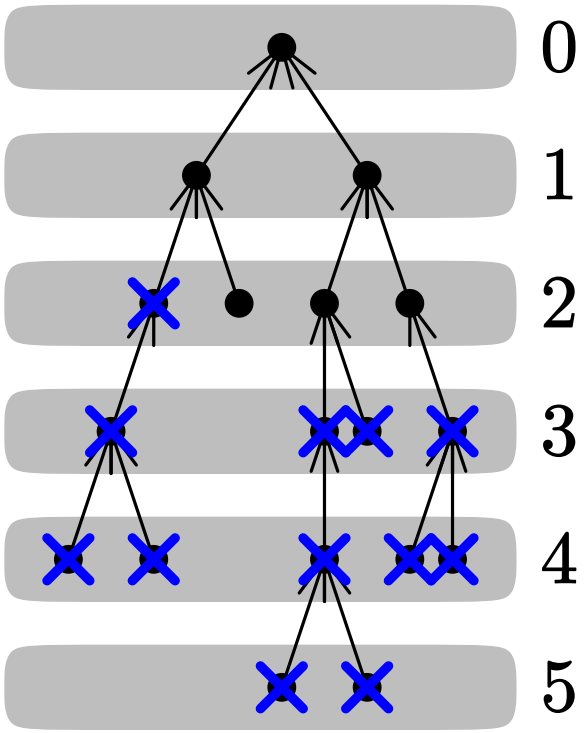
M_1	5	4	4
M_2	5	4	3
M_3	4	4	3

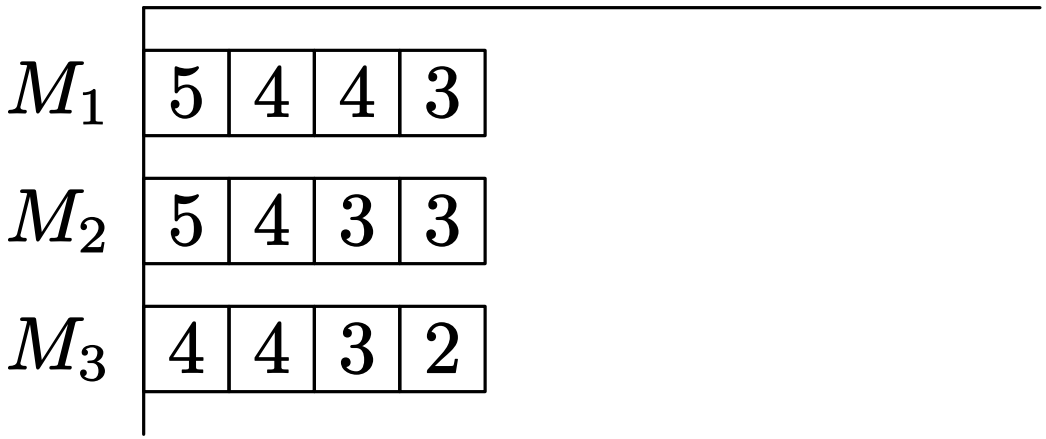
アルゴリズム

深いジョブを優先するリスト・スケジューリング

Hu のアルゴリズム

解く問題：$P \mid \text{in-tree}, p_j = 1 \mid C_{\max}$



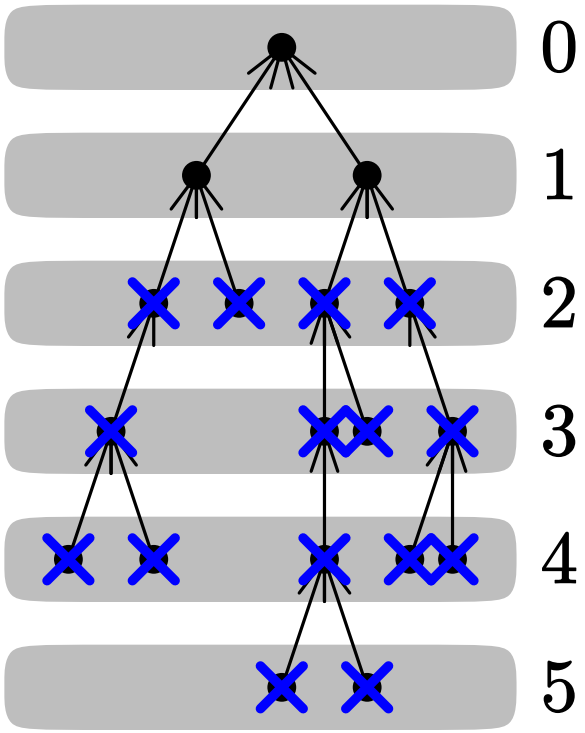


アルゴリズム

深いジョブを優先するリスト・スケジューリング

Hu のアルゴリズム

解く問題：$P \mid \text{in-tree}, p_j = 1 \mid C_{\max}$



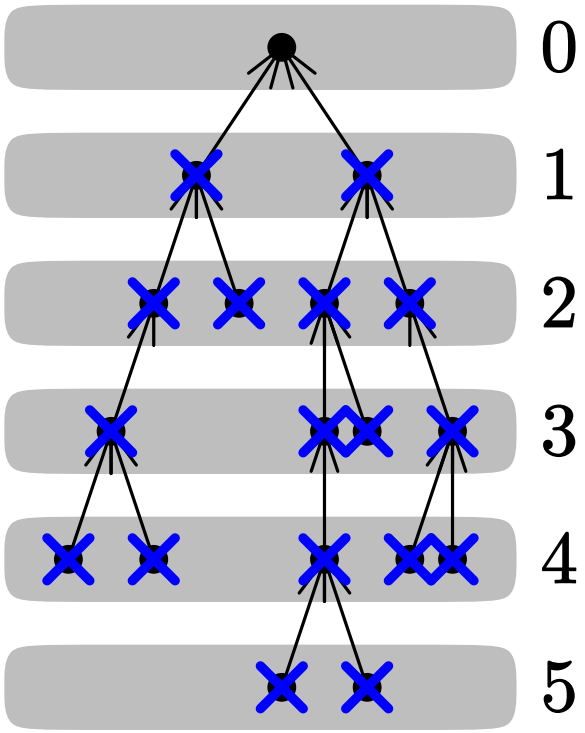
M_1	5	4	4	3	2
M_2	5	4	3	3	2
M_3	4	4	3	2	2

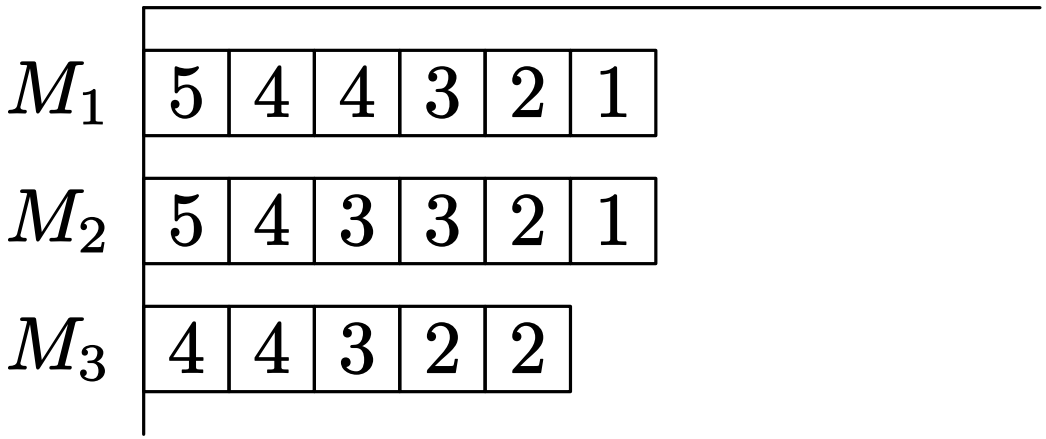
アルゴリズム

深いジョブを優先するリスト・スケジューリング

Hu のアルゴリズム

解く問題：$P \mid \text{in-tree}, p_j = 1 \mid C_{\max}$



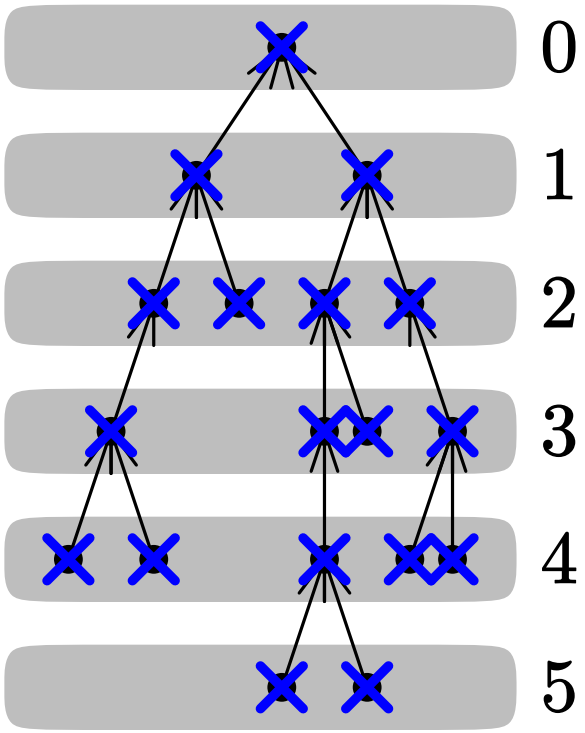


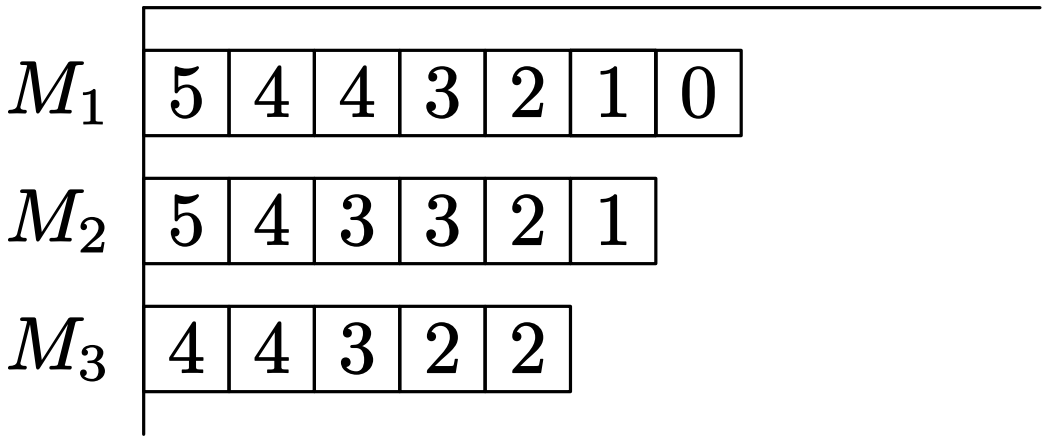
アルゴリズム

深いジョブを優先するリスト・スケジューリング

Hu のアルゴリズム

解く問題：$P \mid \text{in-tree}, p_j = 1 \mid C_{\max}$

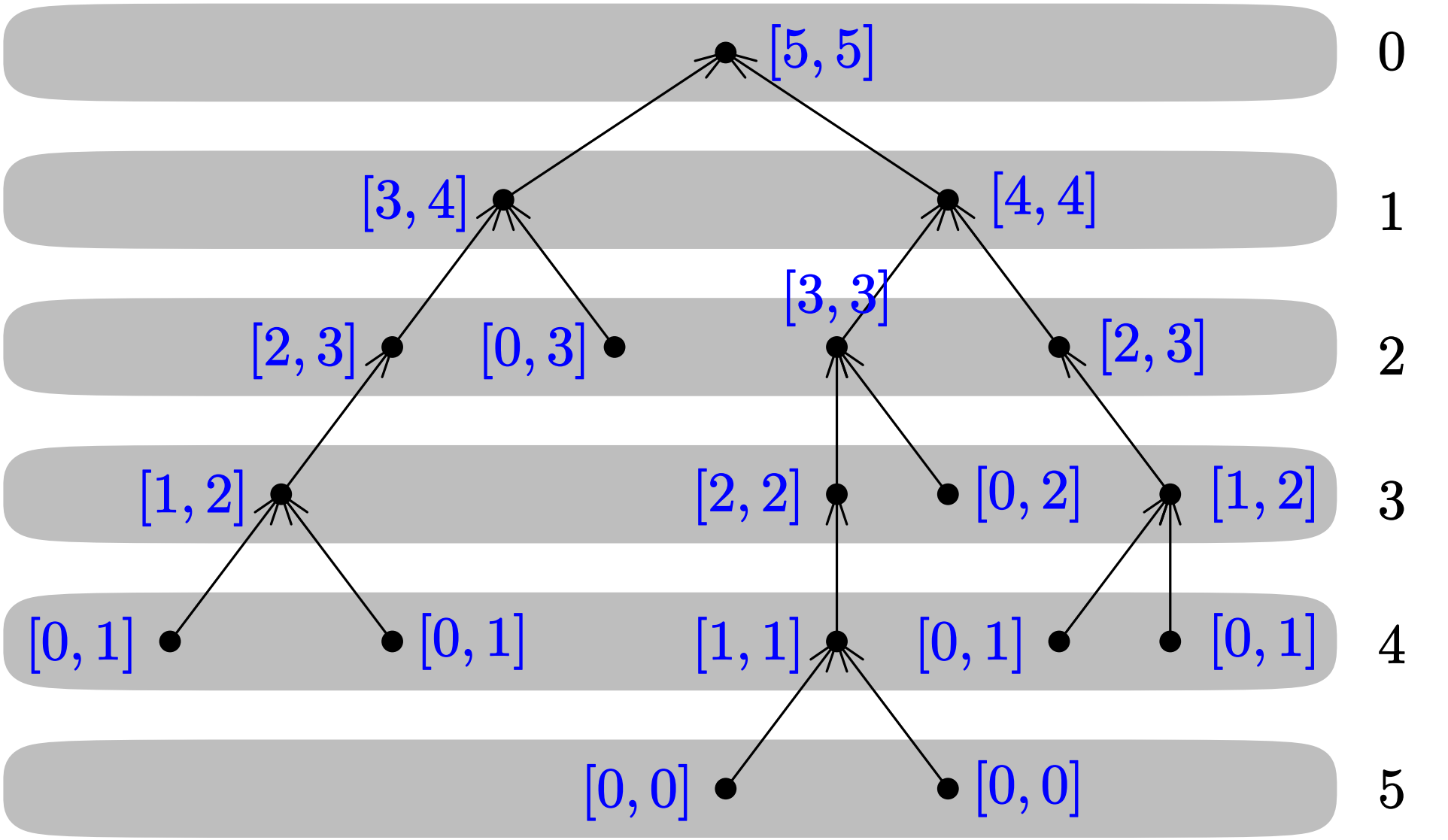




アルゴリズム

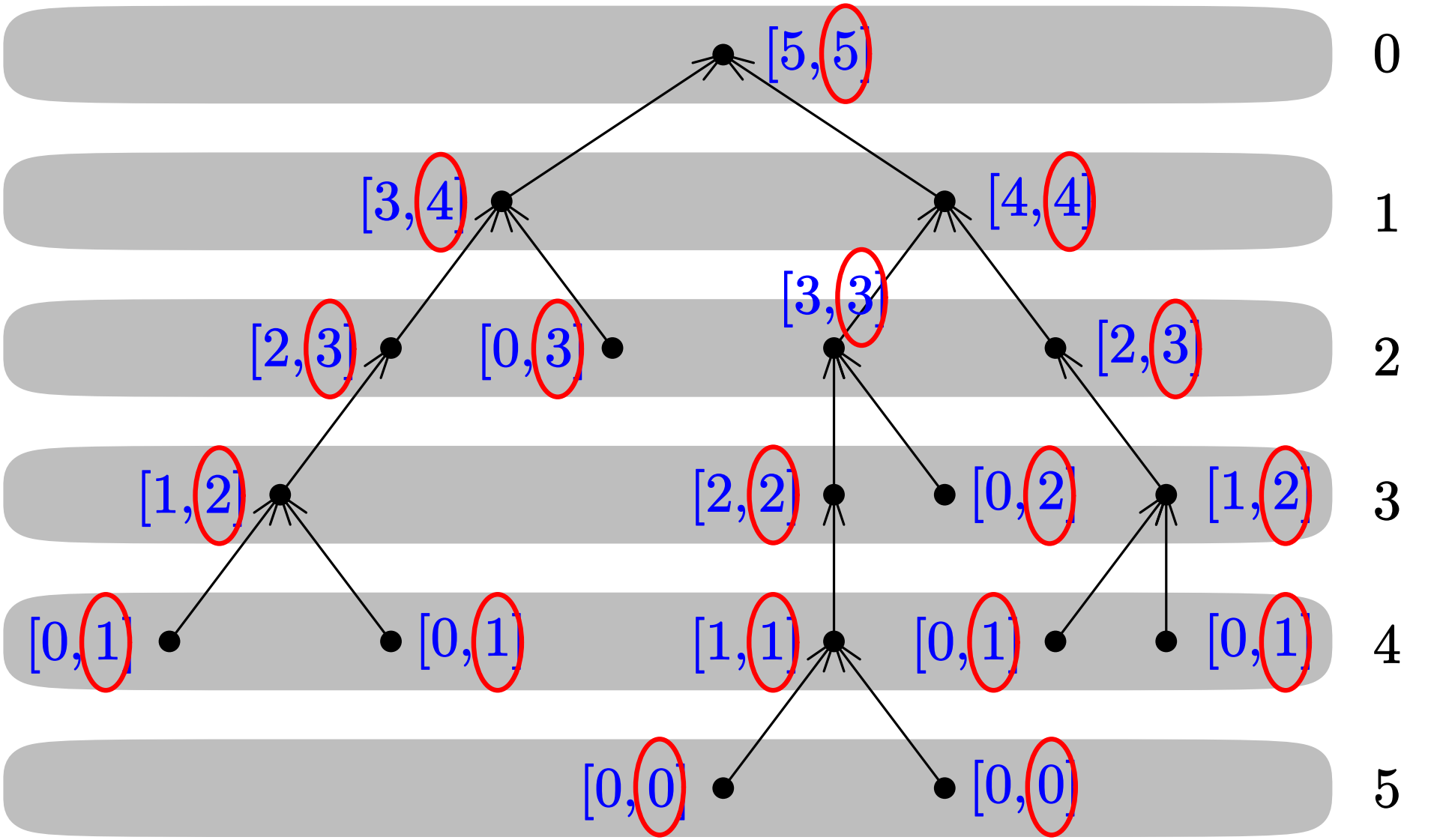
深いジョブを優先するリスト・スケジューリング

Hu のアルゴリズム：正当性 (1)



[最早開始時刻, 最遅開始時刻]

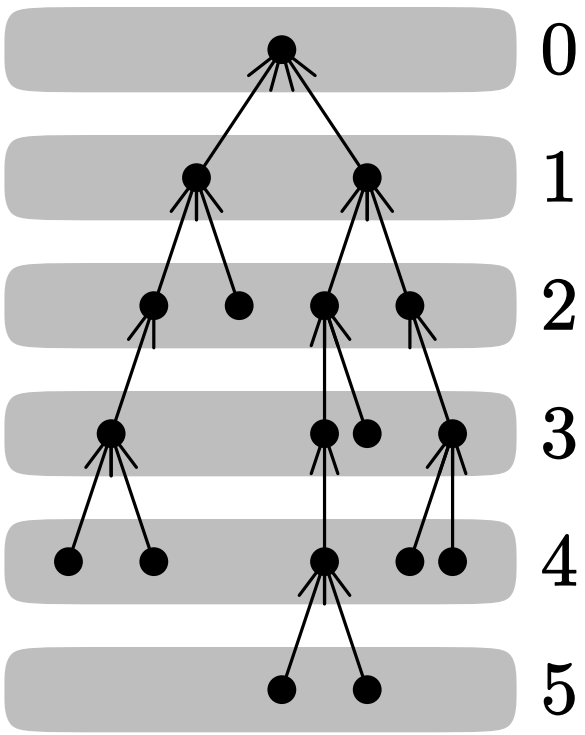
Hu のアルゴリズム：正当性 (1)



[最早開始時刻, 最遅開始時刻]

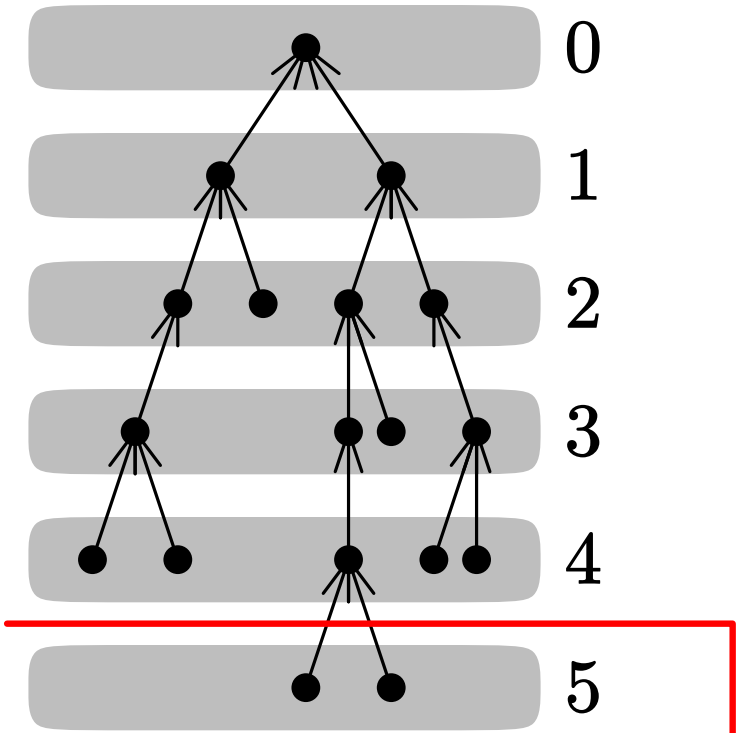
Hu のアルゴリズム：正当性 (2)

$C_{\max} = 6$ となるスケジュールはあるか？　(機械数 $m = 3$)



Hu のアルゴリズム：正当性 (2)

$C_{\max} = 6$ となるスケジュールはあるか？　(機械数 $m = 3$)



処理したいジョブ数　深さ　2

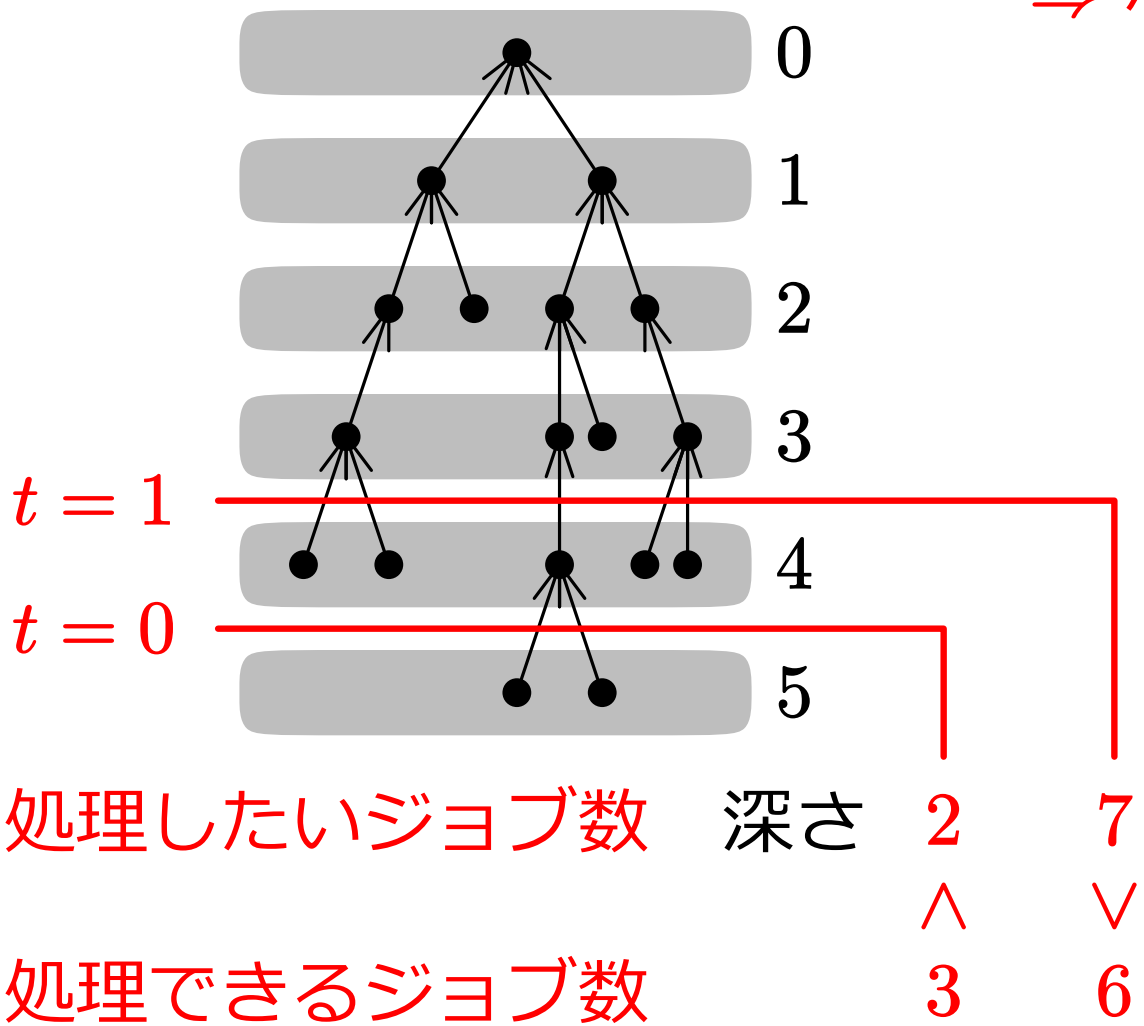
∧

処理できるジョブ数　　3

Hu のアルゴリズム：正当性 (2)

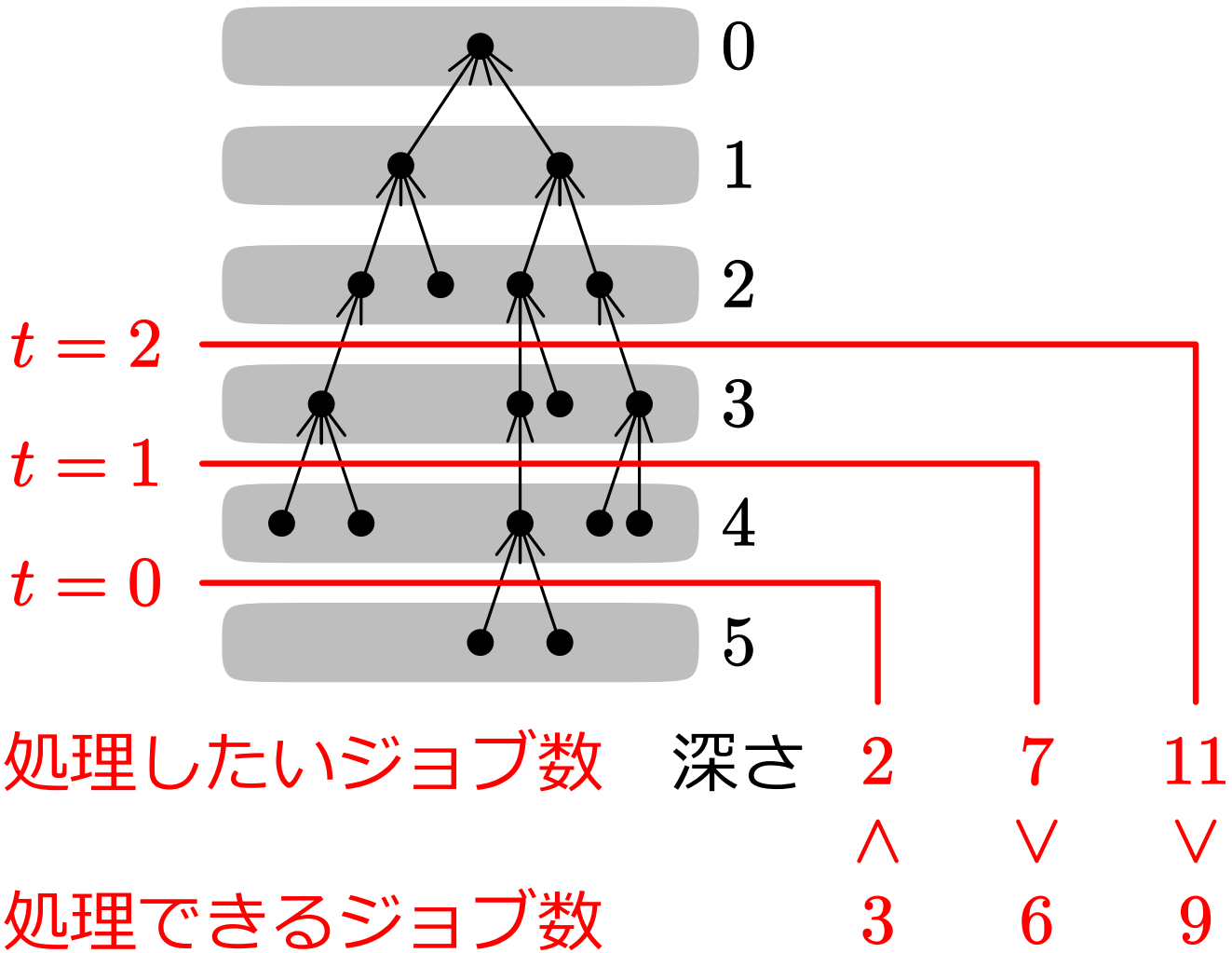
$C_{\max} = 6$ となるスケジュールはあるか？　(機械数 $m = 3$)

$\Rightarrow$ ない！



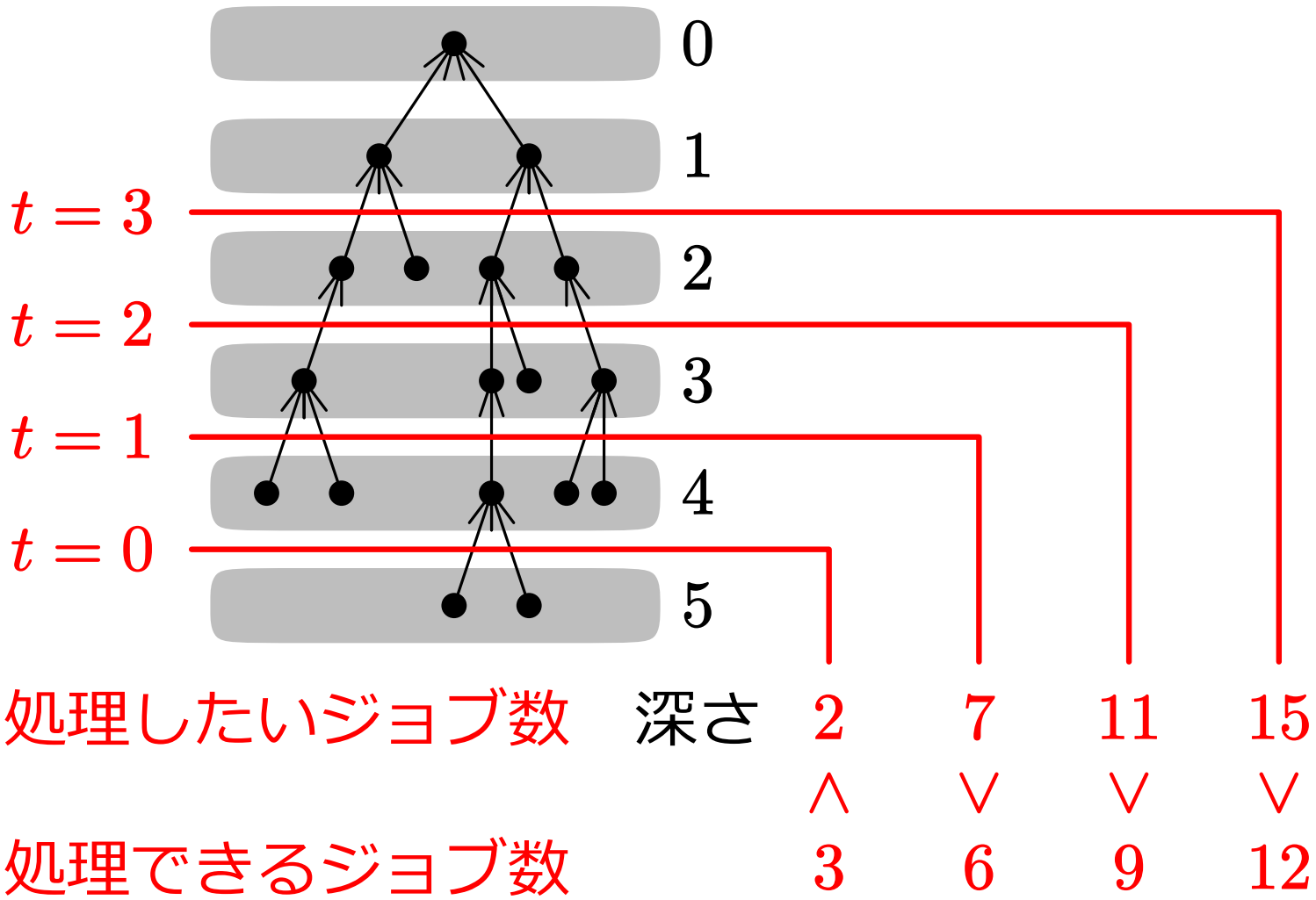
Hu のアルゴリズム：正当性 (2)

$C_{\max} = 6$ となるスケジュールはあるか？　(機械数 $m = 3$)



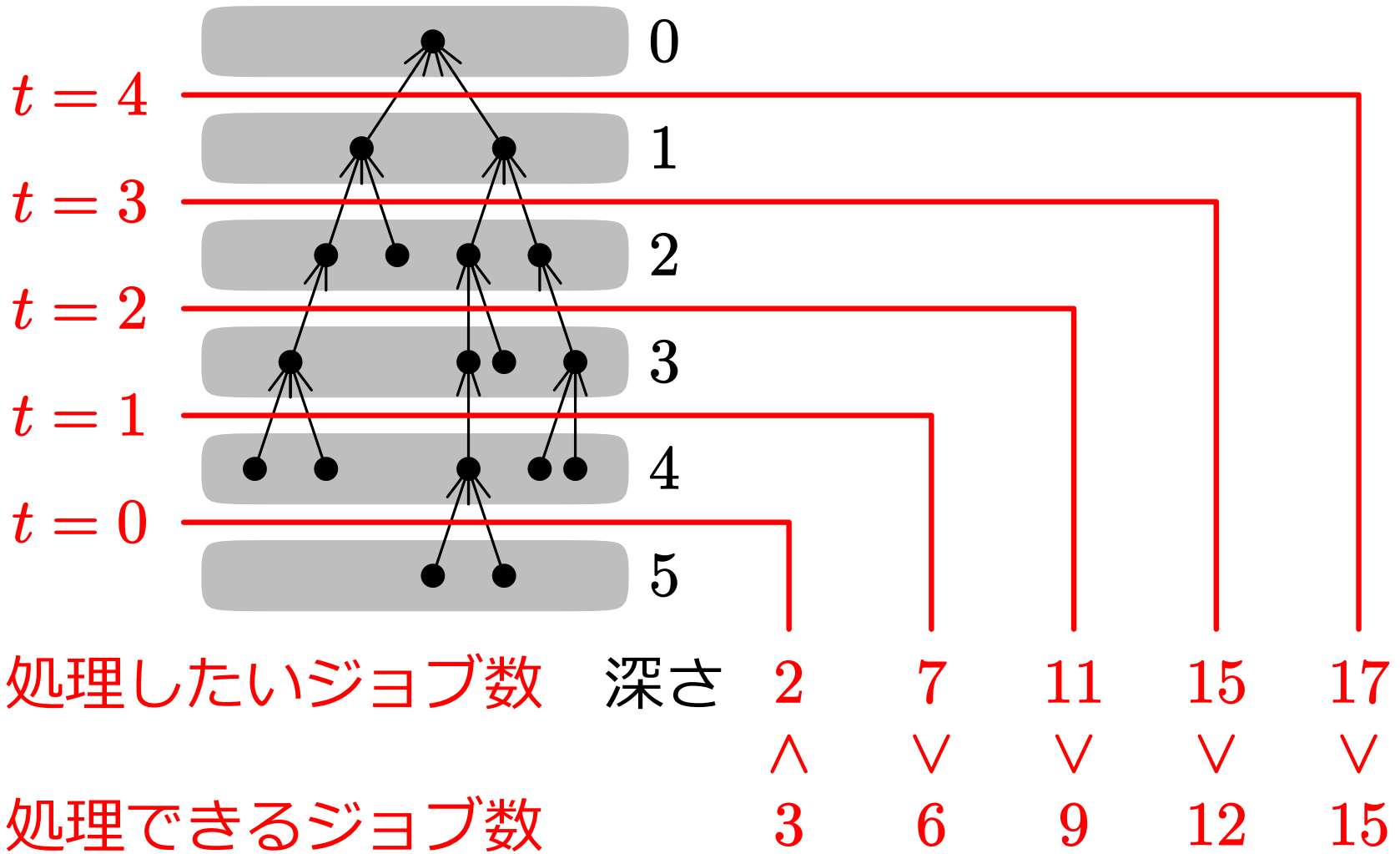
Hu のアルゴリズム：正当性 (2)

$C_{\max} = 6$ となるスケジュールはあるか？　(機械数 $m = 3$)



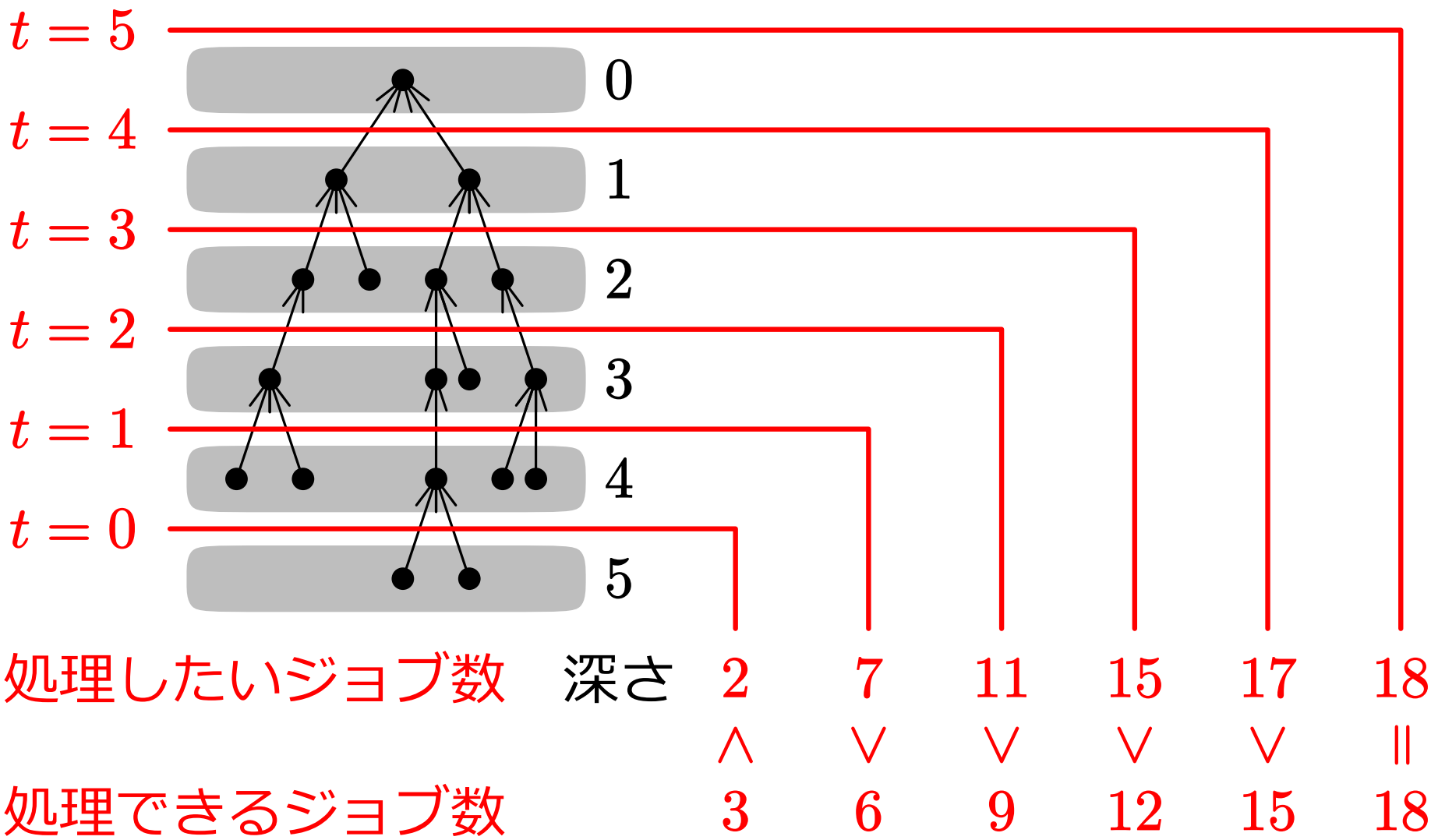
Hu のアルゴリズム：正当性 (2)

$C_{\max} = 6$ となるスケジュールはあるか？　(機械数 $m = 3$)



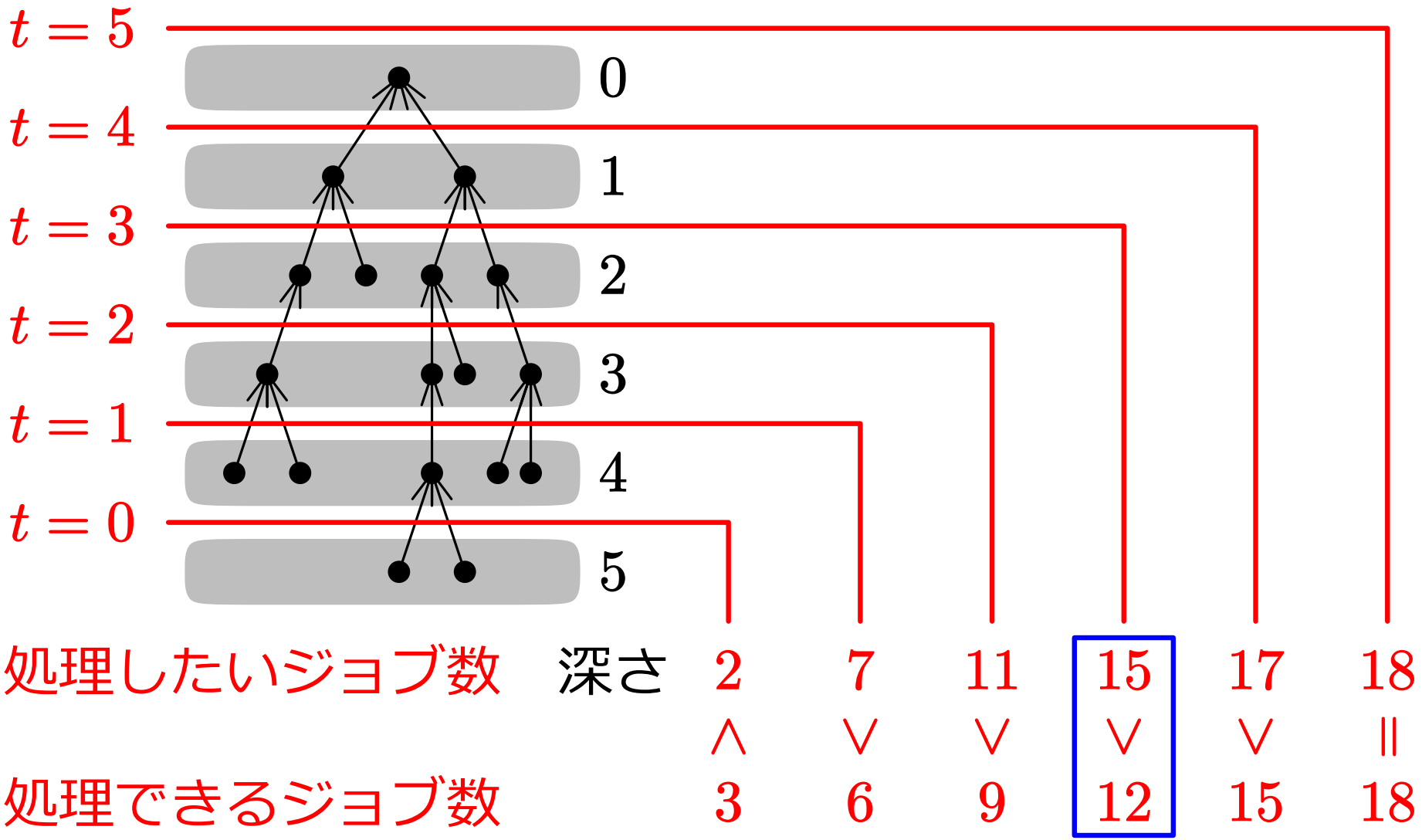
Hu のアルゴリズム：正当性 (2)

$C_{\max} = 6$ となるスケジュールはあるか？　(機械数 $m = 3$)



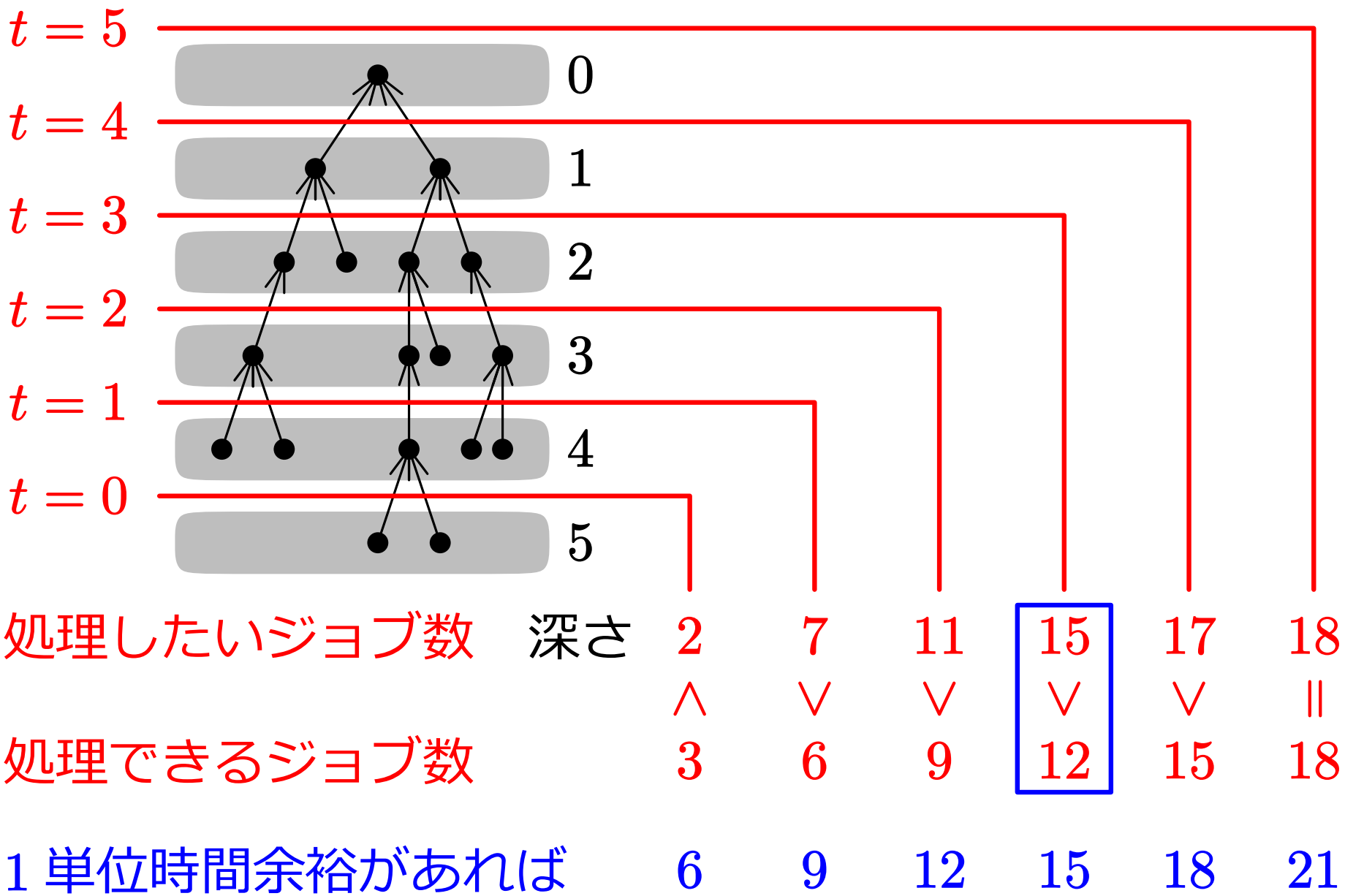
Hu のアルゴリズム：正当性 (2)

$C_{\max} = 6$ となるスケジュールはあるか？　(機械数 $m = 3$)



Hu のアルゴリズム：正当性 (2)

$C_{\max} = 6$ となるスケジュールはあるか？　(機械数 $m = 3$)



Hu のアルゴリズム：正当性 (3)

余裕の導出法

- 深さの最大値を $d_{\max}$ とする
- 次を満たす最小の c を見つける

$$\underbrace{\text{深さが } d_{\max}-t \text{ 以上のジョブ数}} \leq m(t+1+c) \quad \forall\, t \in \{0,\ldots,d_{\max}\}$$

- その最小値を c^* とする

このとき，最適値 $\geq \underbrace{\text{クリティカル・パス上のジョブ数}}_{=\,d_{\max}+1} + c^*$

Hu のアルゴリズム：正当性 (3)

余裕の導出法

- 深さの最大値を $d_{\max}$ とする
- 次を満たす最小の c を見つける

$$\underbrace{\text{深さが } d_{\max}-t \text{ 以上のジョブ数}} \leq m(t+1+c) \quad \forall\, t \in \{0, \ldots, d_{\max}\}$$

- その最小値を c^* とする

このとき，最適値 $\geq$ クリティカル・パス上のジョブ数 $+ c^*$

$$= d_{\max} + 1$$

証明したい目標

Hu のアルゴリズムの出力値 $= d_{\max} + 1 + c^*$

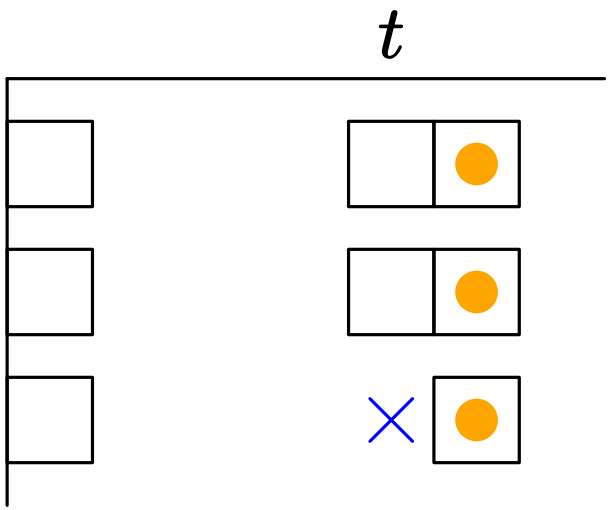
Hu のアルゴリズム：正当性 (4)

補題

Hu のアルゴリズムの出力 σ では
任意の時刻 $t \in [0, C_{\max}(\sigma) - 1]$ に対して
t に開始するジョブ数 $\geq t + 1$ に開始するジョブ数

証明 (背理法)：そうでないと仮定する

- $t + 1$ に開始した任意のジョブ J に対して，
  $J' \to J$ となるジョブ J' で，t に開始するものが存在
  ($\because$ Hu のアルゴリズムの動作)
- これら J' は互いに異なる ($\because$ 内向木)



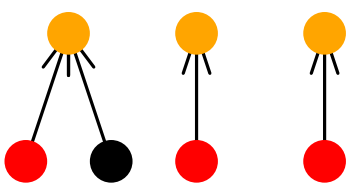
Hu のアルゴリズム：正当性 (4)

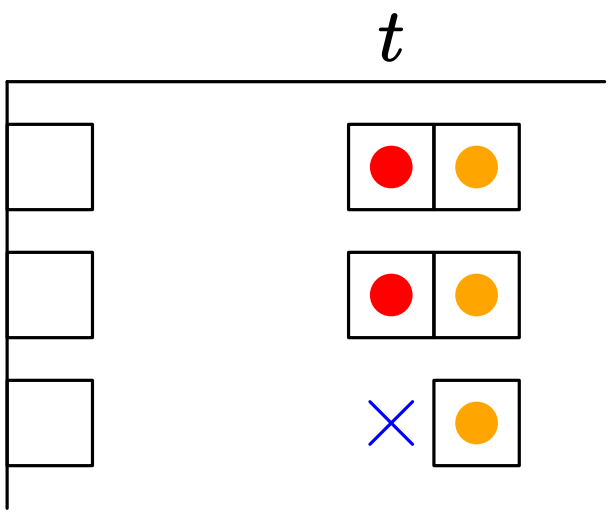
補題

Hu のアルゴリズムの出力 σ では
任意の時刻 $t \in [0, C_{\max}(\sigma) - 1]$ に対して
t に開始するジョブ数 $\geq t + 1$ に開始するジョブ数

証明 (背理法)：そうでないと仮定する

- $t + 1$ に開始した任意のジョブ J に対して，
  $J' \to J$ となるジョブ J' で，t に開始するものが存在
  ($\because$ Hu のアルゴリズムの動作)
- これら J' は互いに異なる ($\because$ 内向木)

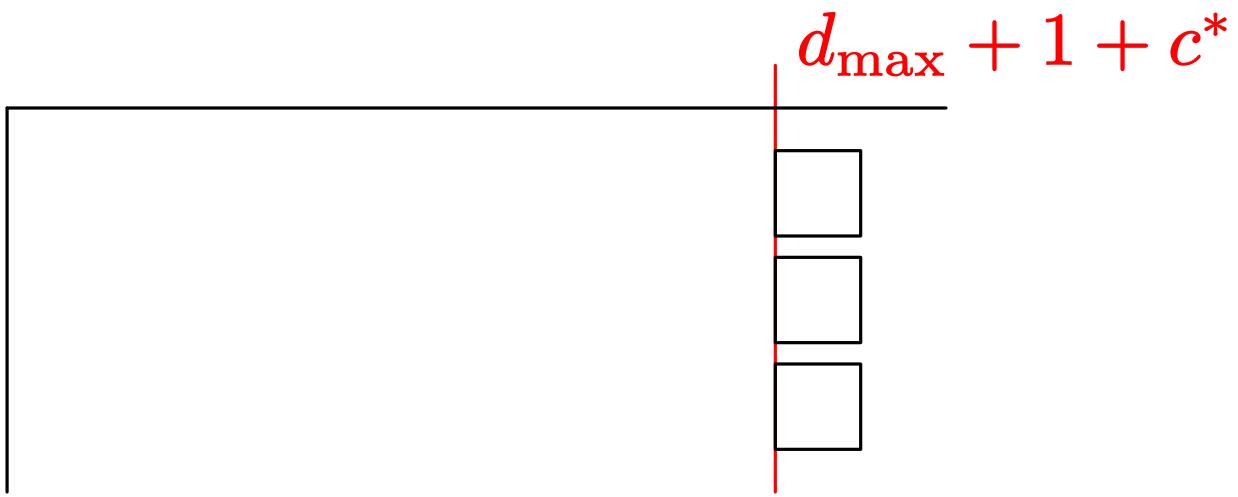




- 仮定に矛盾　□

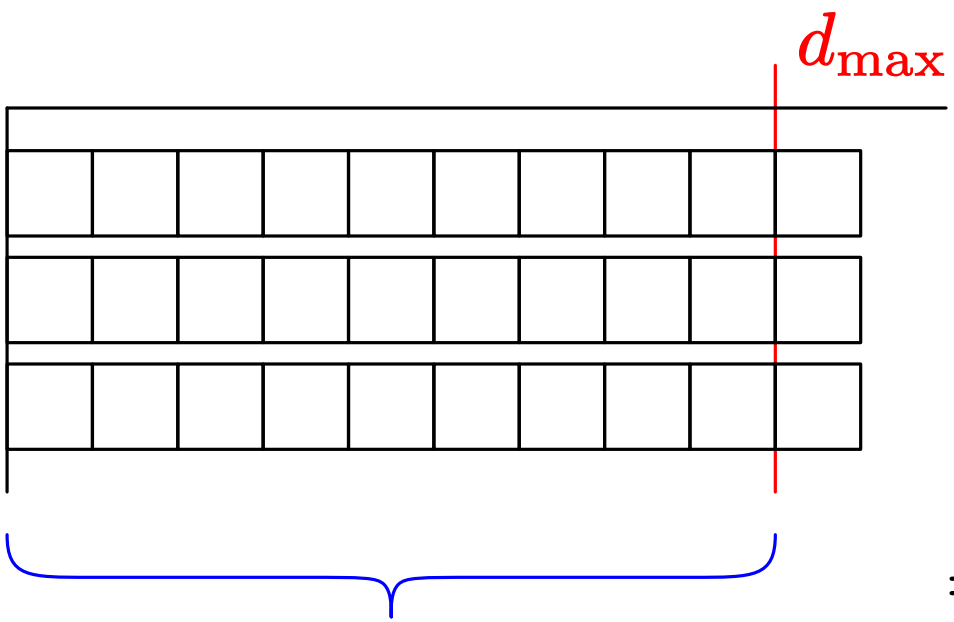
Hu のアルゴリズム：正当性 (5)

証明 (背理法)：Hu のアルゴリズムで，$d_{\max} + 1 + c^*$ までに完了しないとする



Hu のアルゴリズム：正当性 (5)

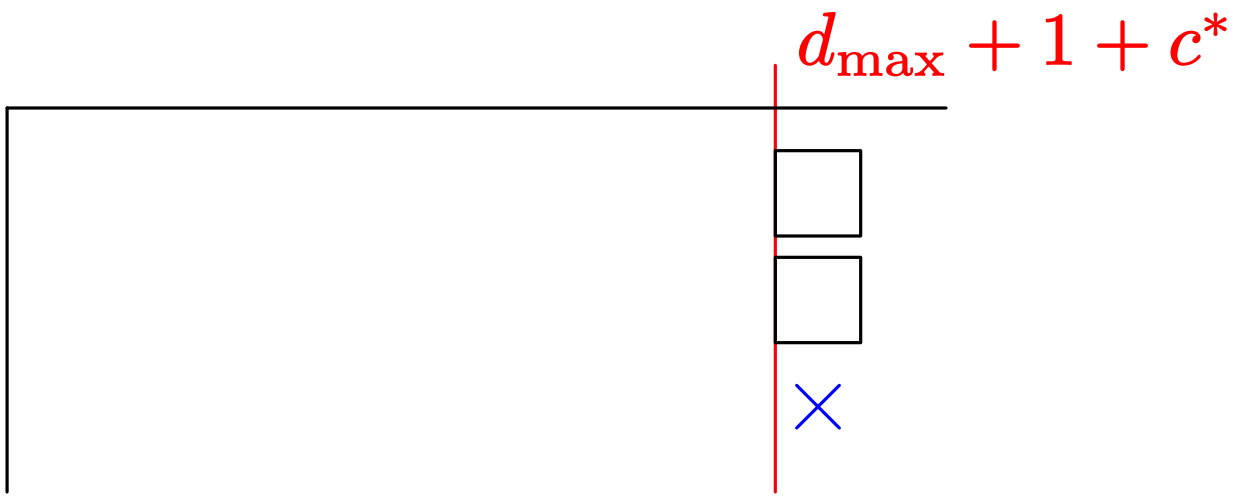
証明 (背理法)：Hu のアルゴリズムで，$d_{\max} + 1 + c^*$ までに完了しないとする



$\Rightarrow c^*$ の決め方に矛盾

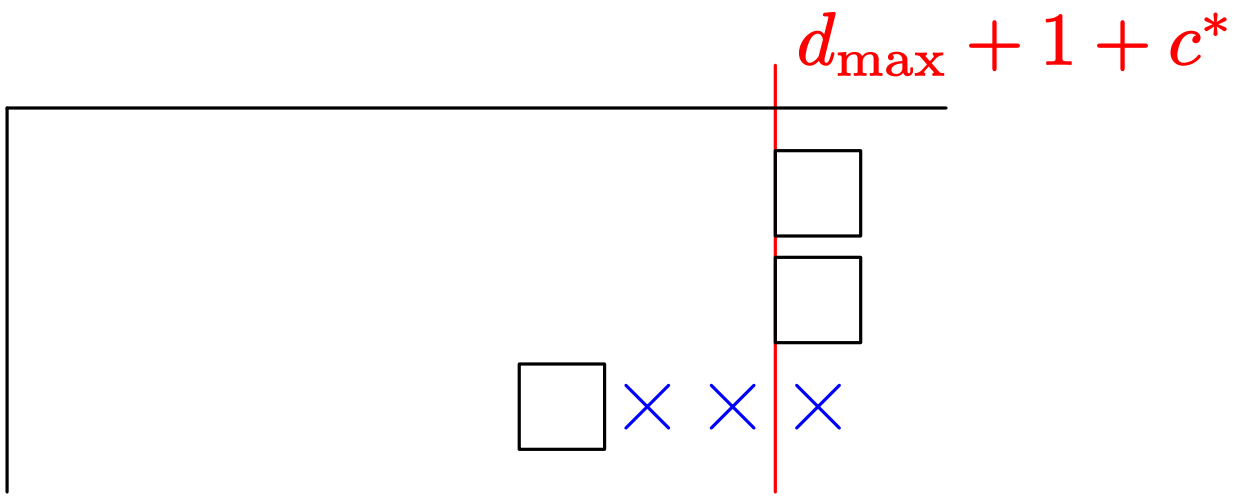
Hu のアルゴリズム：正当性 (5)

証明 (背理法)：Hu のアルゴリズムで，$d_{\max} + 1 + c^*$ までに完了しないとする



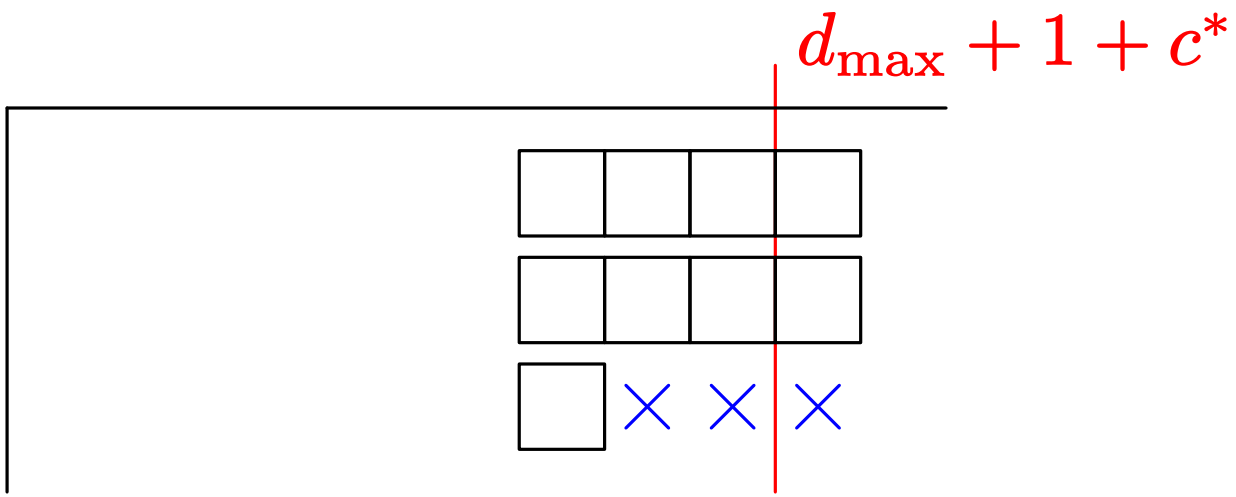
Hu のアルゴリズム：正当性 (5)

証明 (背理法)：Hu のアルゴリズムで，$d_{\max} + 1 + c^*$ までに完了しないとする



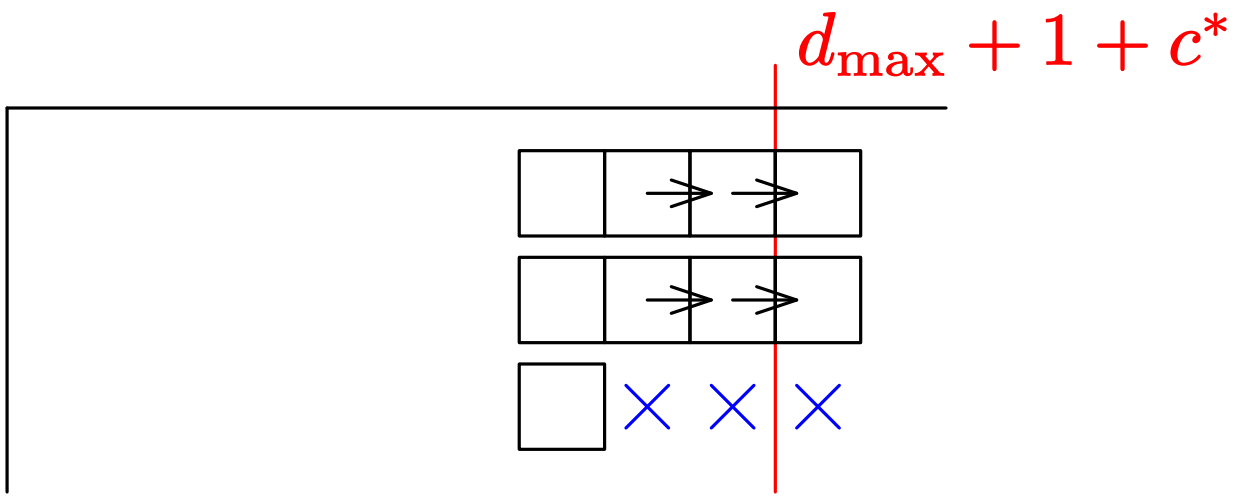
Hu のアルゴリズム：正当性 (5)

証明 (背理法)：Hu のアルゴリズムで，$d_{\max}+1+c^*$ までに完了しないとする



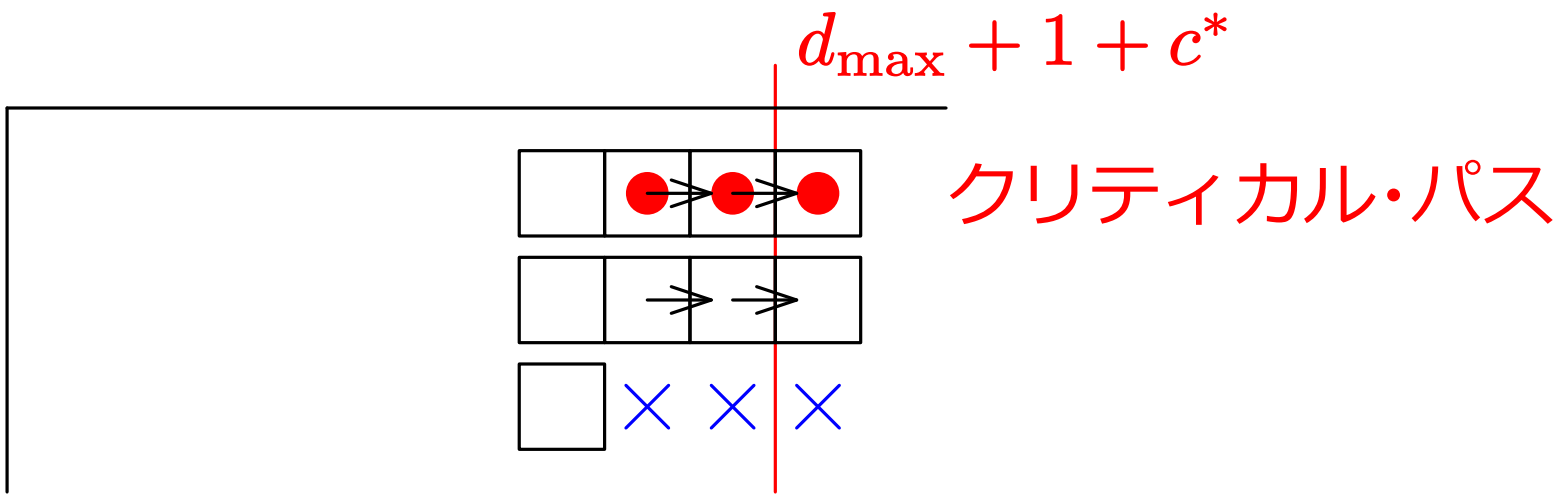
Hu のアルゴリズム：正当性 (5)

証明 (背理法)：Hu のアルゴリズムで，$d_{\max} + 1 + c^*$ までに完了しないとする



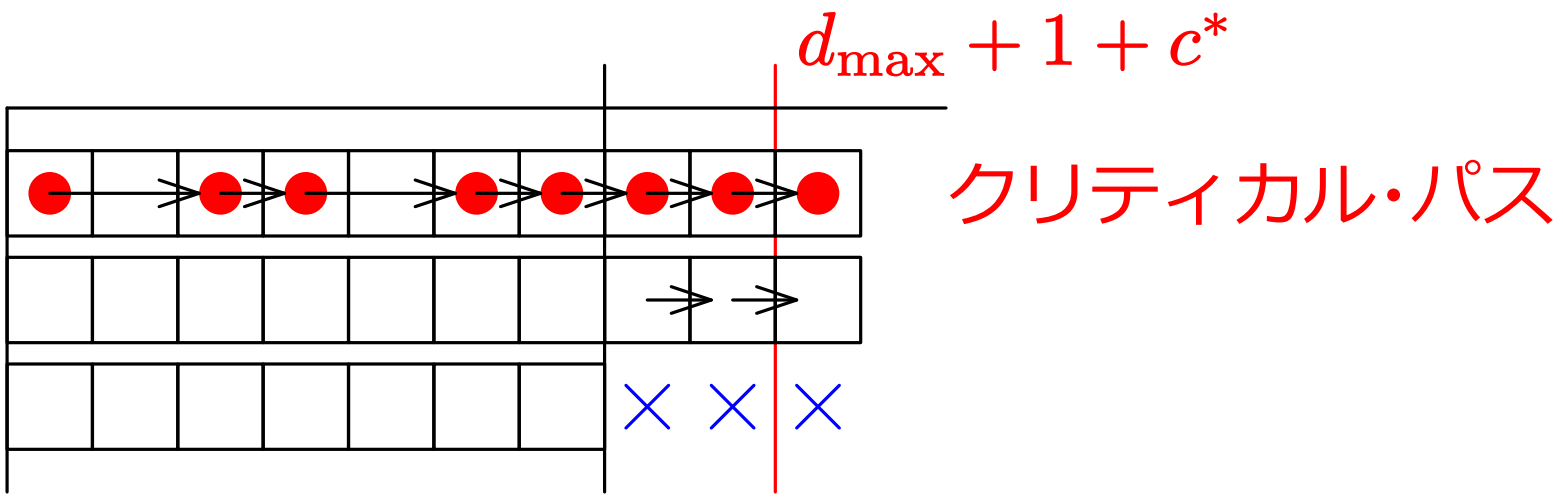
Hu のアルゴリズム：正当性 (5)

証明 (背理法)：Hu のアルゴリズムで，$d_{\max} + 1 + c^*$ までに完了しないとする



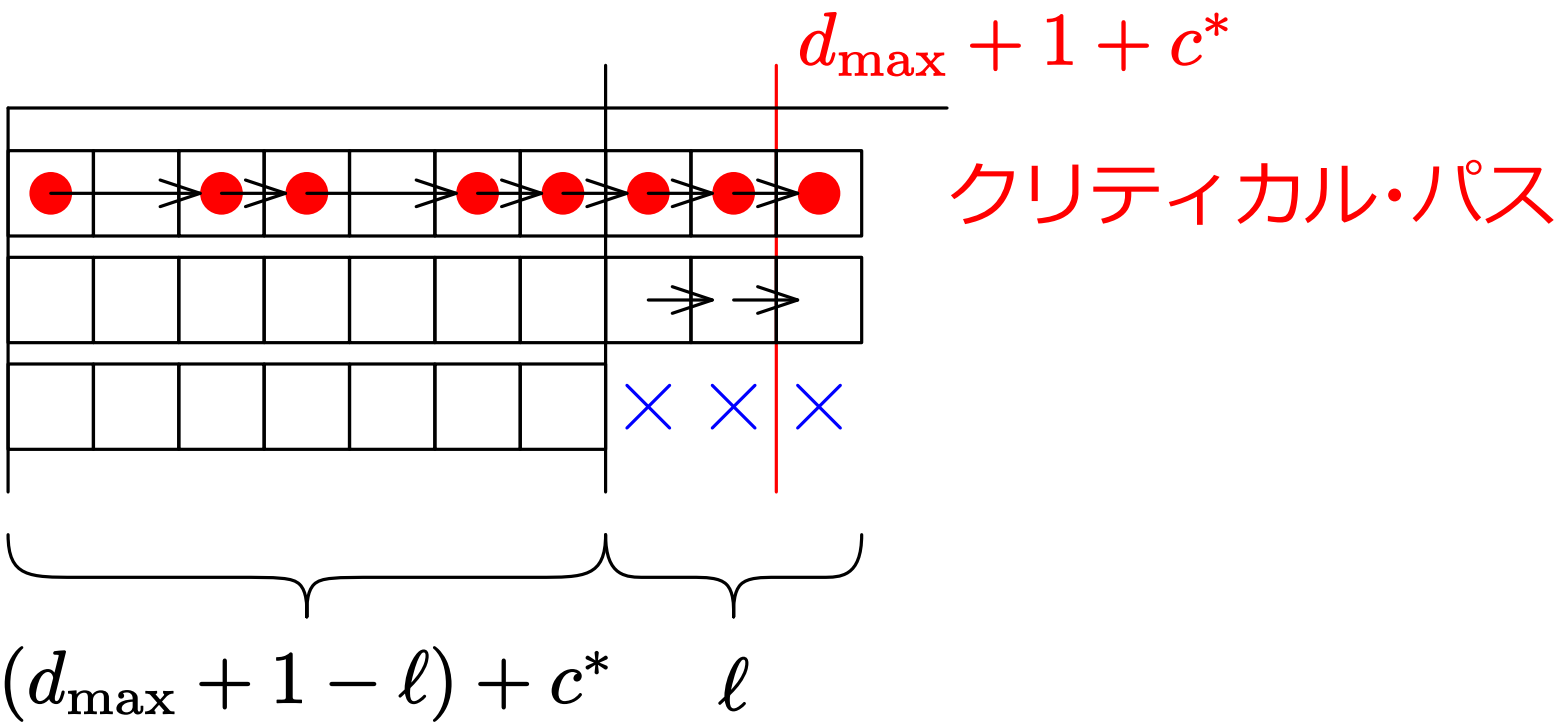
Hu のアルゴリズム：正当性 (5)

証明 (背理法)：Hu のアルゴリズムで，$d_{\max} + 1 + c^*$ までに完了しないとする



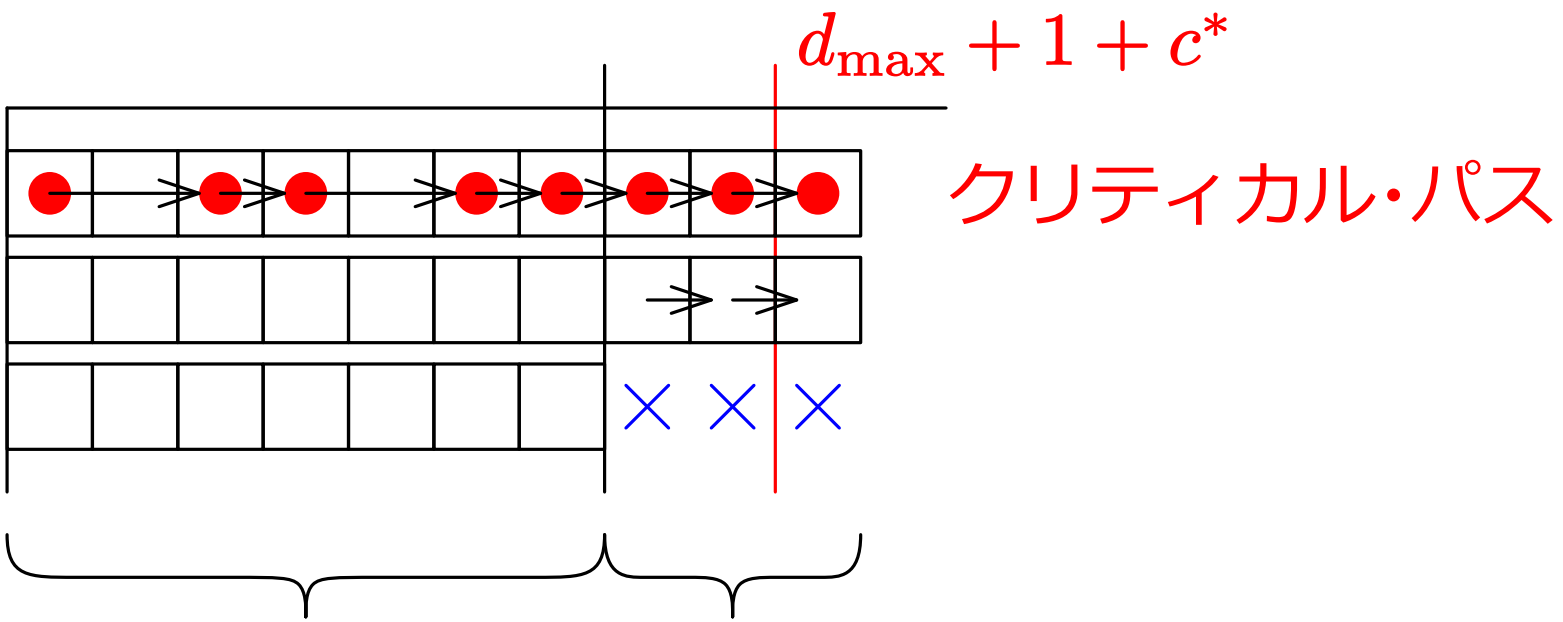
Hu のアルゴリズム：正当性 (5)

証明 (背理法)：Hu のアルゴリズムで，$d_{\max}+1+c^*$ までに完了しないとする



Hu のアルゴリズム：正当性 (5)

証明 (背理法)：Hu のアルゴリズムで，$d_{\max}+1+c^*$ までに完了しないとする



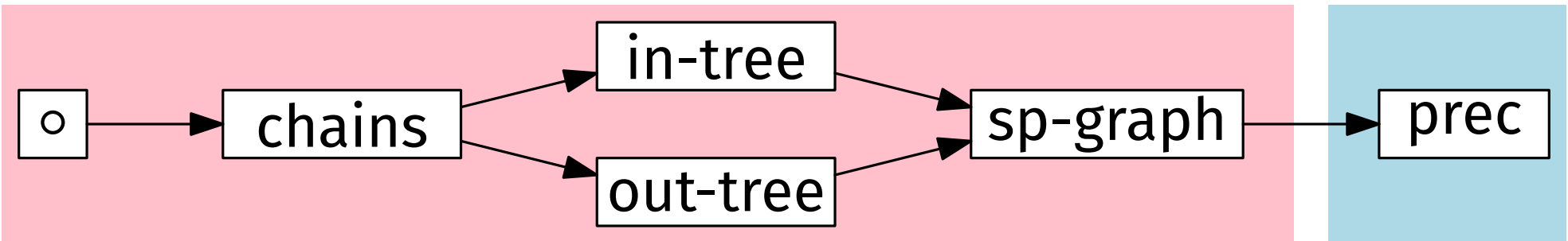
$$(d_{\max}+1-\ell)+c^*+\ell = d_{\max}+1+c^*$$

$\Rightarrow$ 矛盾 $\square$

まとめ

計算量

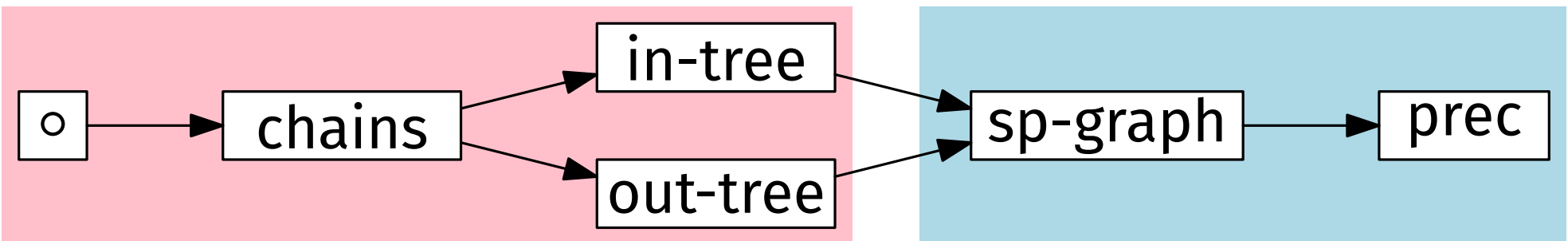
問題 $1 \mid \text{prec} \mid \sum w_j C_j$



多項式時間で解ける

強 NP 困難

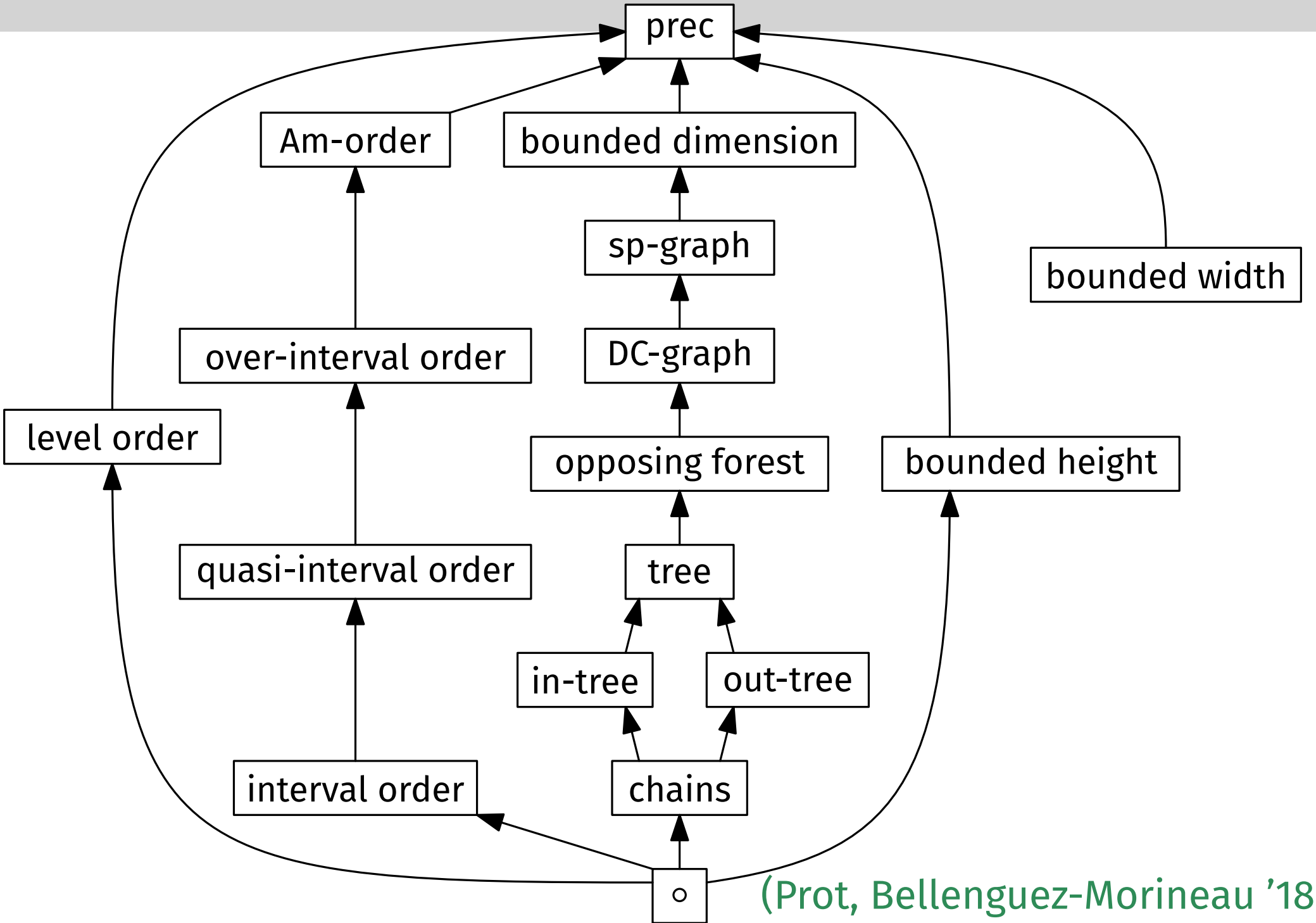
問題 $P \mid \text{prec}, p_j = 1 \mid C_{\max}$



多項式時間で解ける

NP 困難

半順序集合の階層



(Prot, Bellenguez-Morineau '18)

次回の予告

次回から 3 回の予告

ショップ・スケジューリング (shop scheduling)
= 一つのジョブを複数の機械で処理したい設定

- フローショップ・スケジューリング
- ジョブショップ・スケジューリング
- オープンショップ・スケジューリング