# 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 8 回

## 先行制約：多機械

岡本 吉央　(電気通信大学)

okamotoy@uec.ac.jp

2024 年 12 月 10 日

最終更新：2024 年 12 月 11 日　14:01

# 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) |

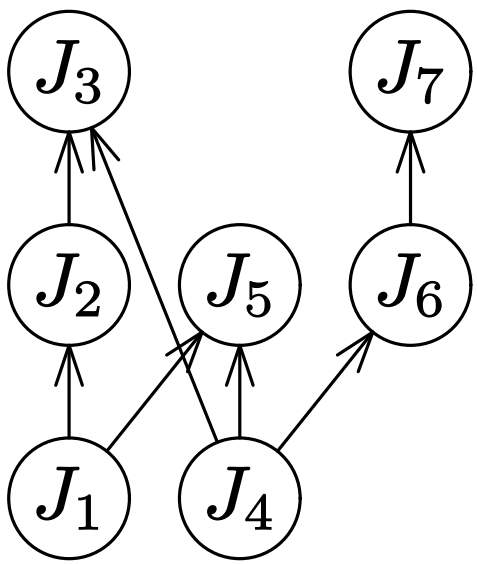
# 復習：先行制約

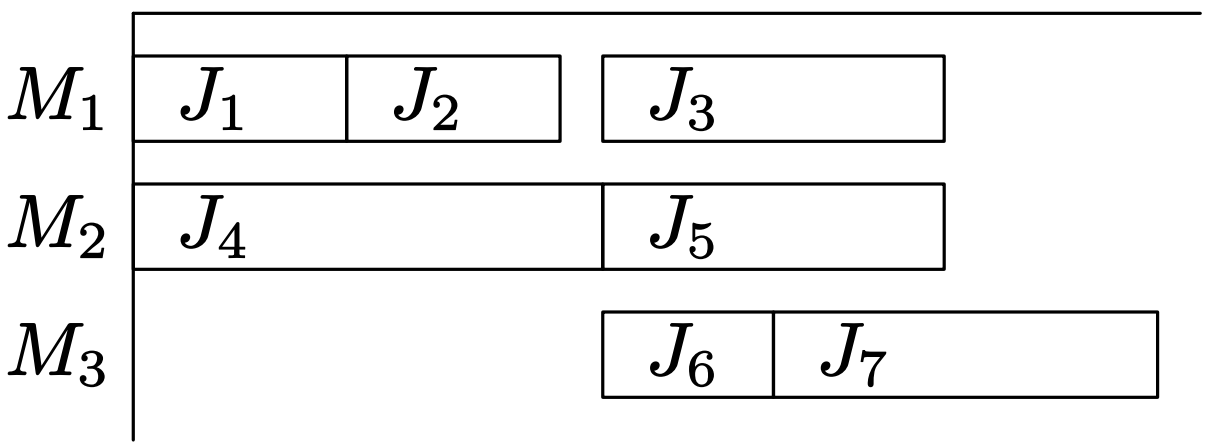


## ジョブの特性の１つ：先行制約 (precedence constraint)

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，$\beta$ に「prec」と書く　　(precedence)



# 復習：$P\infty \mid \text{prec} \mid C_{\max}$



**定理：$P\infty \mid \text{prec} \mid C_{\max}$**

問題 $P\infty \mid \text{prec} \mid C_{\max}$ は $O(n+h)$ 時間で解ける

$n =$ ジョブ数，$h =$ 先行制約を表すグラフの辺数

**クリティカル・パス法** を紹介した

注：クリティカル・パス法で，$P\infty \mid \text{prec} \mid \sum C_j$ も解ける

# 復習：$P\infty \mid \mathrm{prec} \mid C_{\max}$



**定理：$P\infty \mid \mathrm{prec} \mid C_{\max}$**

問題 $P\infty \mid \mathrm{prec} \mid C_{\max}$ は $O(n+h)$ 時間で解ける

$n =$ ジョブ数，$h =$ 先行制約を表すグラフの辺数

**クリティカル・パス法** を紹介した

注：クリティカル・パス法で，$P\infty \mid \mathrm{prec} \mid \sum C_j$ も解ける

**今日 考える問題**

- $Pm \mid \mathrm{prec} \mid C_{\max}$
- $Pm \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$

# 復習：$Pm \mid\mid C_{\max}$ の難しさ

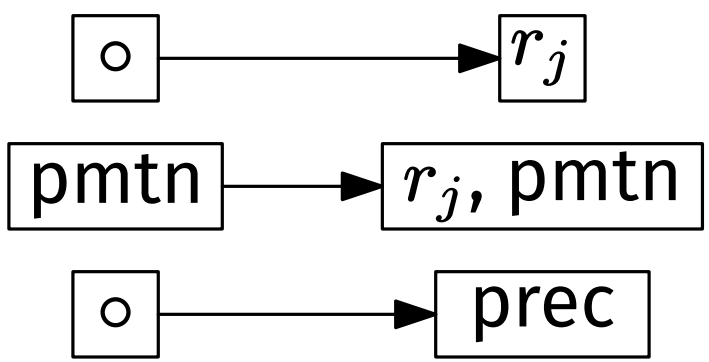


**定理 (復習)**

- 任意の整数 $m \geq 1$ に対して，
  問題 $Pm \mid\mid C_{\max}$ は弱 NP 困難
- 問題 $P \mid\mid C_{\max}$ は強 NP 困難

計算量

$\beta$　ジョブの特性

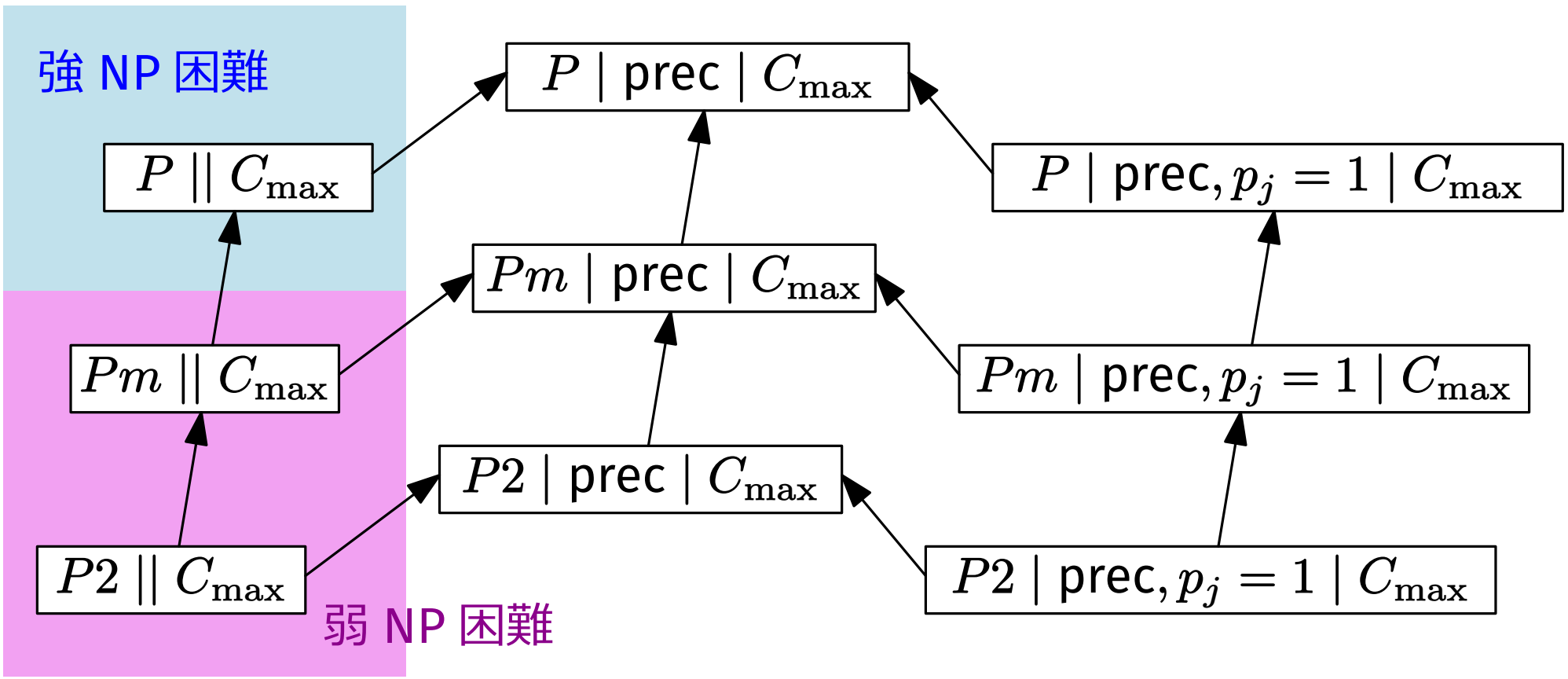


# 今日の内容：$Pm \mid \text{prec} \mid C_{\max}$ の難しさ



## 定理 (今日の内容)

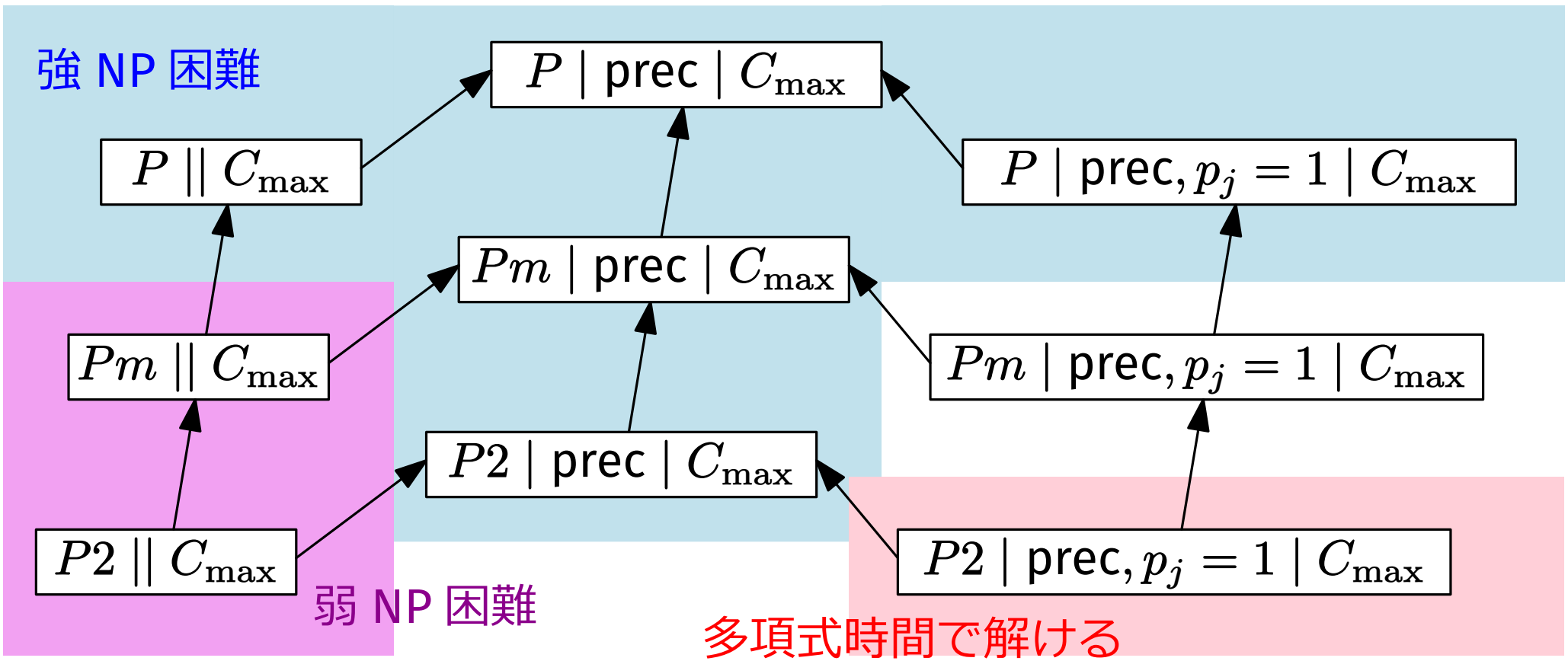
- 問題 $P2 \mid \text{prec} \mid C_{\max}$ は強 NP 困難
- 問題 $P \mid \text{prec}, p_j = 1 \mid C_{\max}$ は (強) NP 困難
- 問題 $P2 \mid \text{prec}, p_j = 1 \mid C_{\max}$ は多項式時間で解ける



# 今日の内容：$Pm \mid \mathrm{prec} \mid C_{\max}$ の難しさ 

## 定理 (今日の内容)

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



# 今日の内容：$Pm \mid \text{prec} \mid C_{\max}$ 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)

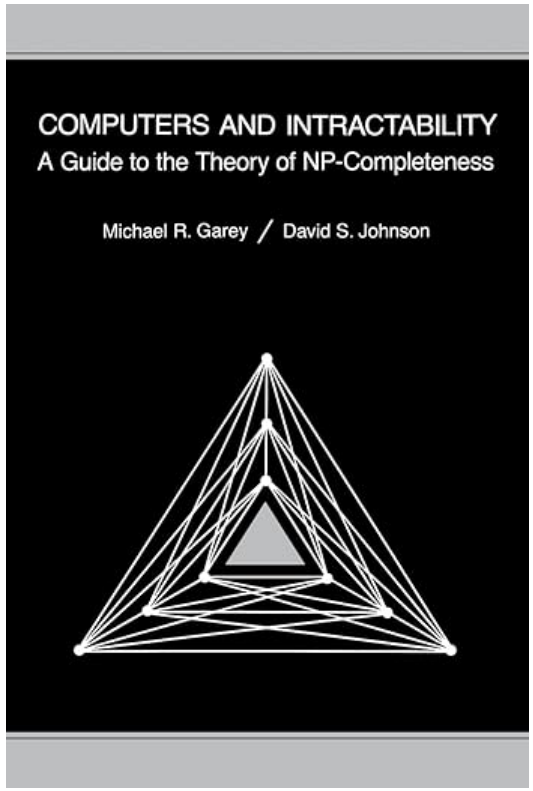
各整数 $m \geq 3$ に対して,
$Pm \mid \text{prec}, p_j = 1 \mid C_{\max}$ は 多項式時間で解けるか？

# Garey–Johnson の未解決問題 ('79)



Garey と Johnson の本 "Computers and Intractability" ('79) には，多項式時間で解けるか NP 困難か，未解決である問題が 11 個挙げてある

1. グラフ同型性
2. 部分グラフ同相性
3. グラフの種数
4. 弦グラフ補完
5. 全域木パリティ
6. 半順序の次元
7. $P3 \mid \text{prec}, p_j = 1 \mid C_{\max}$
8. 線形計画法
9. 完全ユニモジュラ性
10. 合成数
11. 最小三角形分割



# Garey–Johnson の未解決問題 ('79)



Garey と Johnson の本 "Computers and Intractability" ('79) には，多項式時間で解けるか NP 困難か，未解決である問題が 11 個挙げてある

| | | | |
|---|---|---|---|
| 1. | グラフ同型性 | 未解決 | |
| 2. | 部分グラフ同相性 | P | (Robertson, Seymour 95) |
| 3. | グラフの種数 | NP 困難 | (Thomassen '89) |
| 4. | 弦グラフ補完 | NP 困難 | (Yannakakis '81) |
| 5. | 全域木パリティ | P | (Lovász '80) |
| 6. | 半順序の次元 | NP 困難 | (Yannakakis '82) |
| 7. | $P3 \mid \text{prec}, p_j = 1 \mid C_{\max}$ | 未解決 | |
| 8. | 線形計画法 | P | (Khachiyan '79) |
| 9. | 完全ユニモジュラ性 | P | (Seymour '80) |
| 10. | 合成数 | P | (Agrawal, Kayal, Saxena '04) |
| 11. | 最小三角形分割 | NP 困難 | (Mulzer, Rote '08) |

# 本日の内容



1. $Pm \mid$ **prec** $\mid C_{\max}$ **の困難性**
2. $P2 \mid \text{prec}, p_j = 1 \mid C_{\max}$ のアルゴリズム
3. $P \mid \text{prec} \mid C_{\max}$ の近似アルゴリズム

---


- J. D. Ullman, *NP*-complete scheduling problems. *Journal of Computer and System Sciences* 10 (1975) pp. 384–393.
- J. K. Lenstra, A. H. G. Rinnooy Kan, Complexity of scheduling under precedence constraints. *Operations Research* 26 (1978) pp. 22–35.

# $P2 \mid \mathrm{prec} \mid C_{\max}$ の強 NP 困難性

**定理** (Ullman '75)

$P2 \mid \mathrm{prec} \mid C_{\max}$ は強 NP 困難

補足：Ullman ('75) はジョブの処理時間が $1$ か $2$ という値しか
とらない場合でも，強 NP 困難であることを証明している

今から紹介する証明は，
Ullman ('75) や Lenstra, Rinnooy Kan ('78) とは違うもの

# 3 分割問題



## 定義：3 分割問題 (3-partition problem)

次を満たす正整数 $a_1, a_2, \ldots, a_{3n}$ が与えられる

$$\frac{1}{4}T < a_i < \frac{1}{2}T \quad \text{ただし，} \ T = \frac{1}{n}\sum_{i=1}^{3n} a_i$$

それらを同じ和の $n$ 個のグループに分けられるか？

例：$119, 120, 122, 123, 125, 127, 130, 130, 130, 140, 145, 149$

- $120 + 125 + 145 = 390$
- $123 + 127 + 140 = 390$
- $119 + 122 + 149 = 390$
- $130 + 130 + 130 = 390$

できる

## 事実 (Garey, Johnson '75)

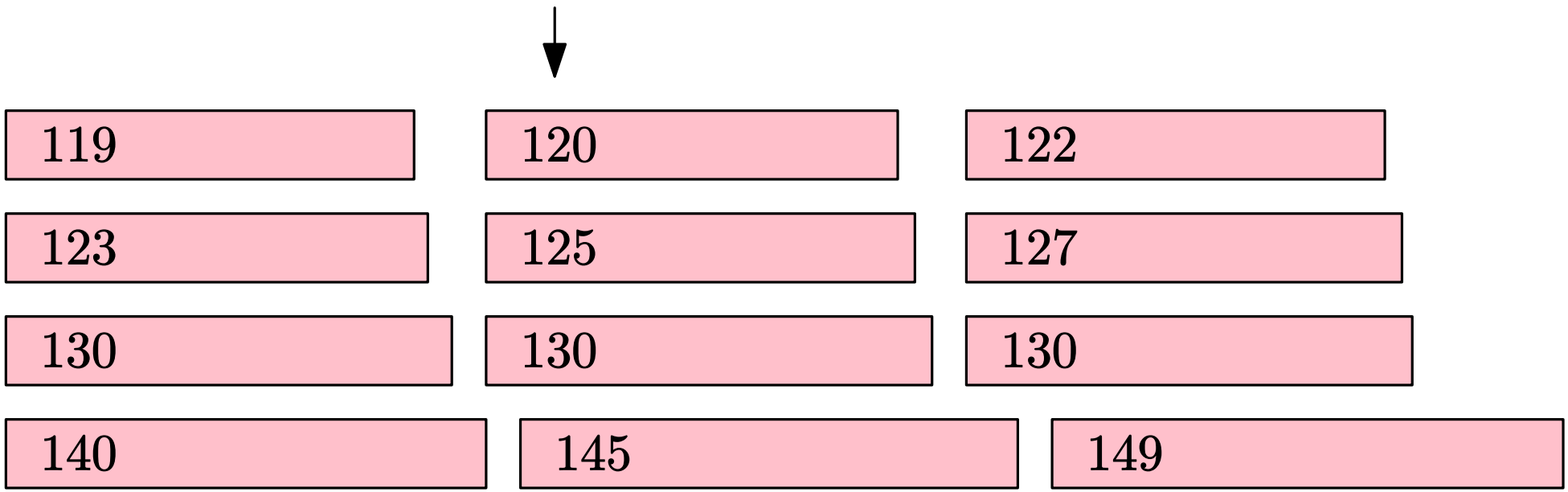
3 分割問題 は強 NP 困難

# $P2 \mid \mathrm{prec} \mid C_{\max}$：証明 (1)



証明：3 分割問題を帰着する

**入力の変換**：$119, 120, 122, 123, 125, 127, 130, 130, 130, 140, 145, 149$




3 分割問題を解くアルゴリズム

# $P2 \mid \mathrm{prec} \mid C_{\max}$：証明 (2)

証明：3 分割問題を帰着する

**入力の変換**：$119, 120, 122, 123, 125, 127, 130, 130, 130, 140, 145, 149$

$\boxed{1} \quad \times (nT + 2n - 2)$

$$T = \frac{1}{n} \sum_{i=1}^{3n} a_i$$

注意：入力の長さ

3 分割問題を解くアルゴリズム

# $P2 \mid \mathrm{prec} \mid C_{\max}$：証明 (2)

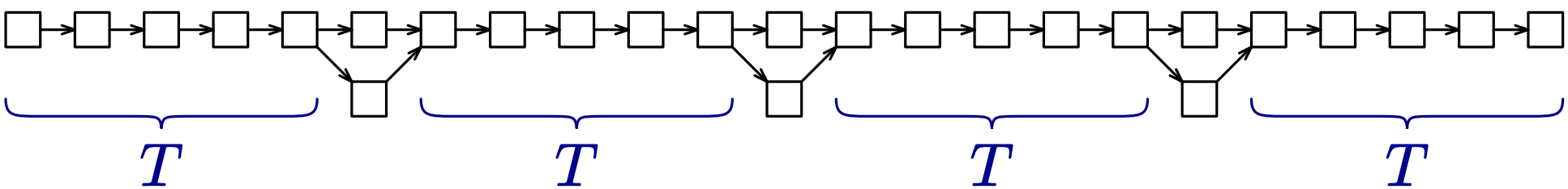
証明：3 分割問題を帰着する

**入力の変換**：$119, 120, 122, 123, 125, 127, 130, 130, 130, 140, 145, 149$

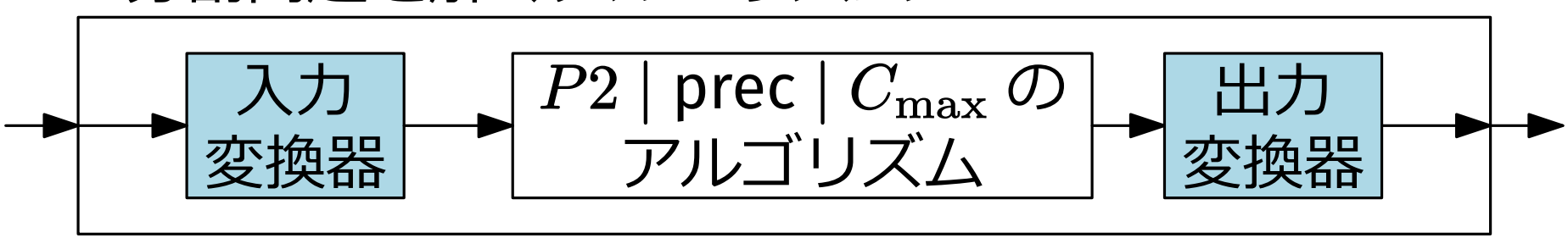
$\boxed{1} \quad \times (nT + 2n - 2)$

$$T = \frac{1}{n} \sum_{i=1}^{3n} a_i$$

先行制約



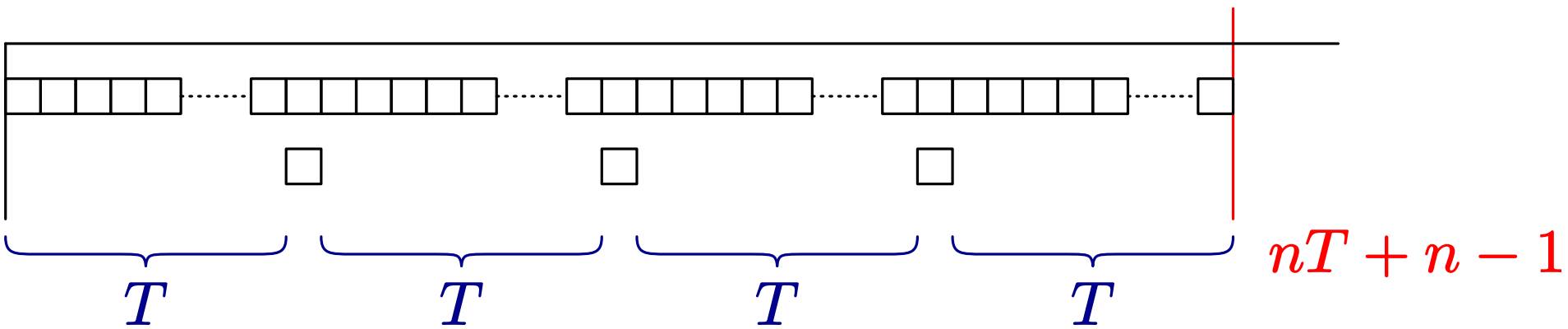

3 分割問題を解くアルゴリズム



# $P2 \mid \mathrm{prec} \mid C_{\max}$：証明 (3)

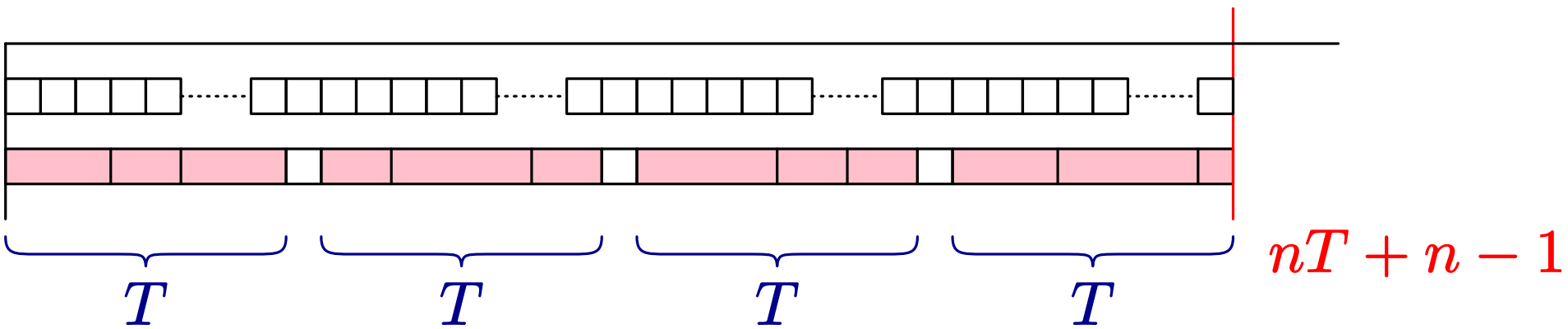
証明：3 分割問題を帰着する

**入力の変換**：



証明：3 分割問題を帰着する

**入力の変換**：

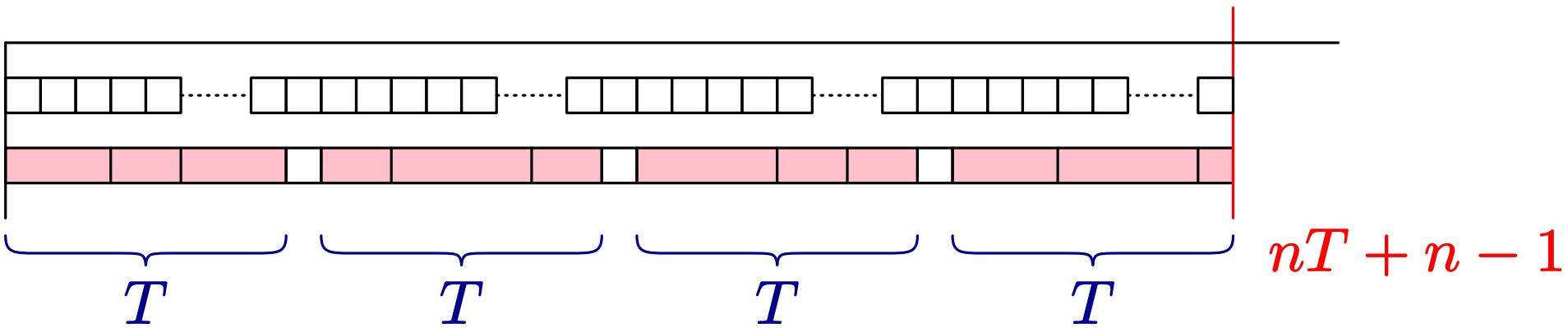


# $P2 \mid \mathrm{prec} \mid C_{\max}$：証明 (3)



証明：3 分割問題を帰着する

**入力の変換**：




**出力の変換**：

最適値 $\leq nT + n - 1 \Rightarrow$ 3 分割問題の答えは「できる」

最適値 $> nT + n - 1 \Rightarrow$ 3 分割問題の答えは「できない」

□

# $P \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$ の強 NP 困難性



**定理** (Lenstra, Rinnooy Kan '78)

$P \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$ は強 NP 困難

すべてのジョブ $J_j$ に対して，$p_j = 1$

(処理時間は入力に含めない)

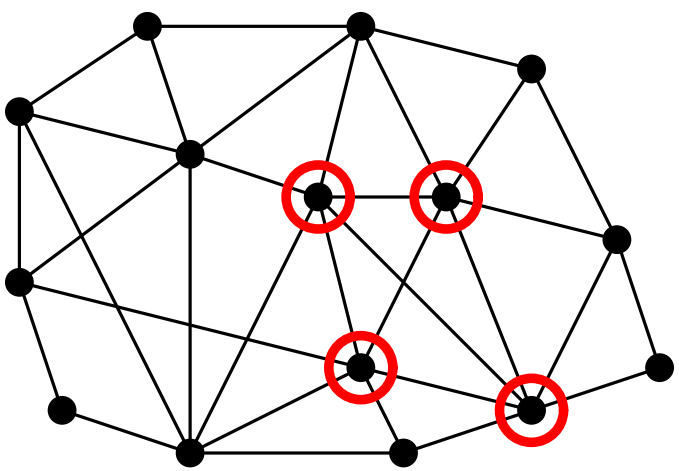
# クリーク問題



## 定義：クリーク問題

無向グラフ $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ と非負整数 $k$ に対して，
$\mathcal{G}$ が頂点数 $k$ のクリークを含むか？

**クリーク** (clique) = 互いに辺で結ばれた頂点の集合



観察：

クリークの頂点数 $= k \Rightarrow$

その中の辺の数 $= \binom{k}{2}$

## 事実 (Karp '72)

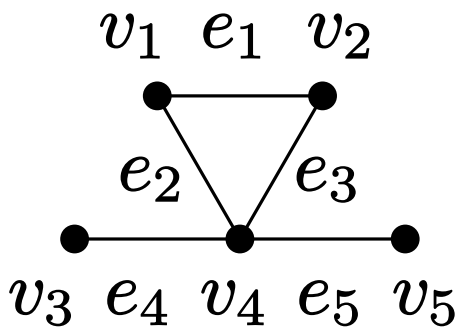
クリーク問題は (強) NP 困難

# $P \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$：証明 (1)



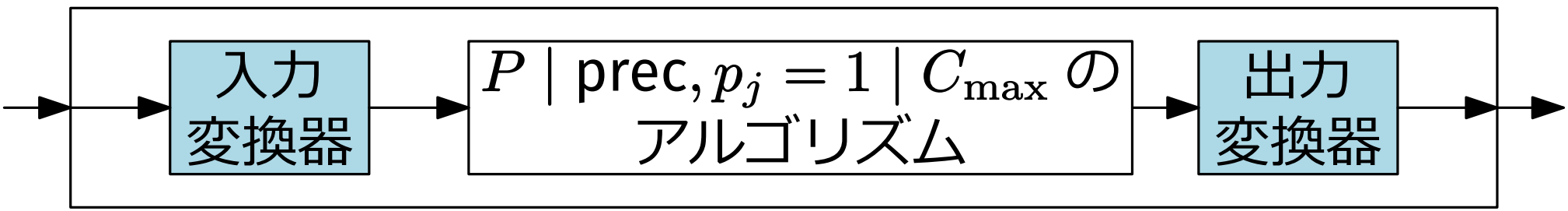
証明：クリーク問題を帰着する

**入力の変換**：




$$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\},\ \mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$$

クリーク問題を解くアルゴリズム

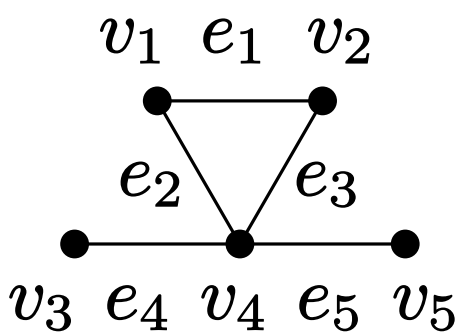


# $P \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$：証明 (1)

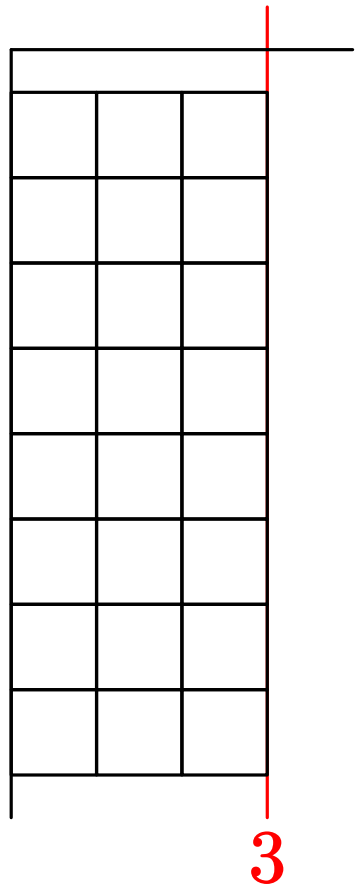


証明：クリーク問題を帰着する

**入力の変換**：




$$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\},\ \mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$$

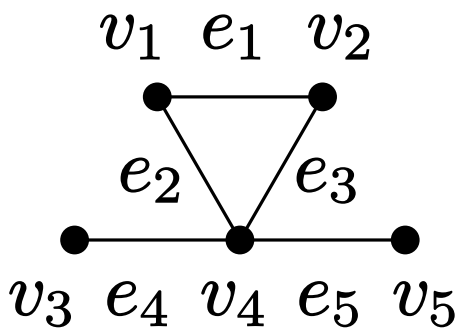


# $P \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$：証明 (1)

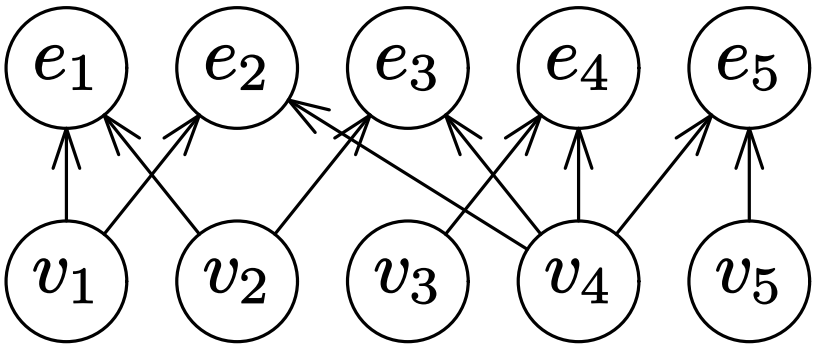


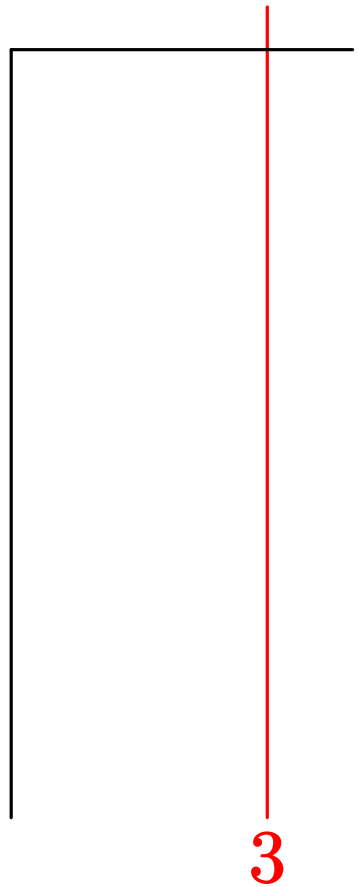
証明：クリーク問題を帰着する

**入力の変換**：




$$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\},\ \mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$$



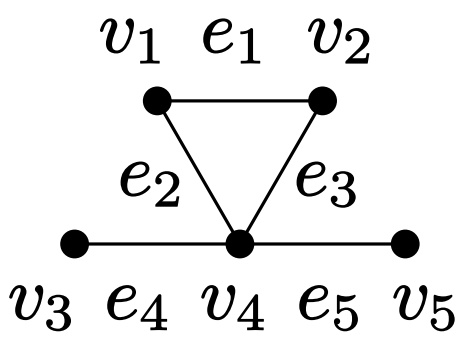



# $P \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$：証明 (1)

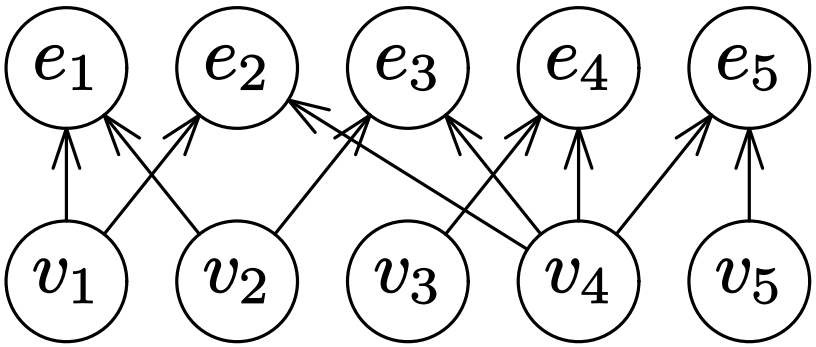


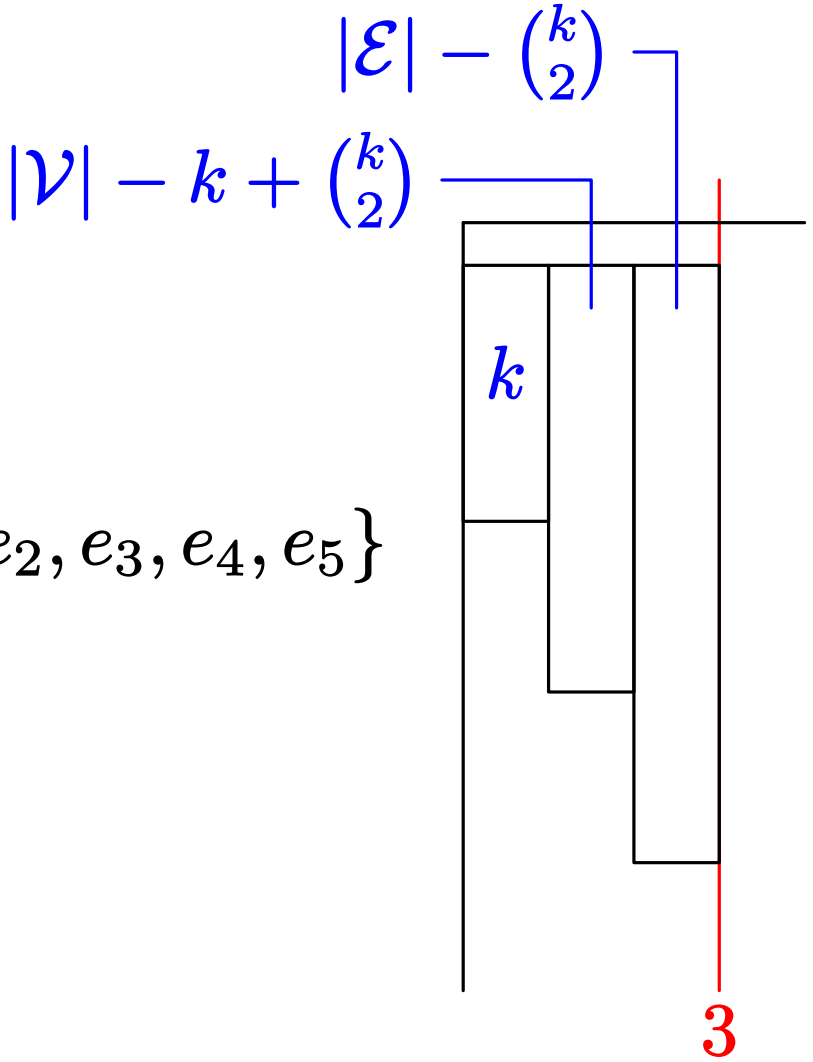
証明：クリーク問題を帰着する

**入力の変換**：




$$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\},\ \mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$$



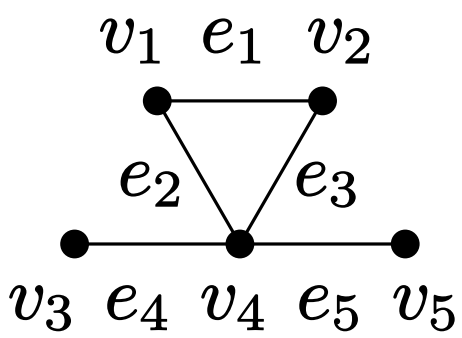



# $P \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$：証明 (1)

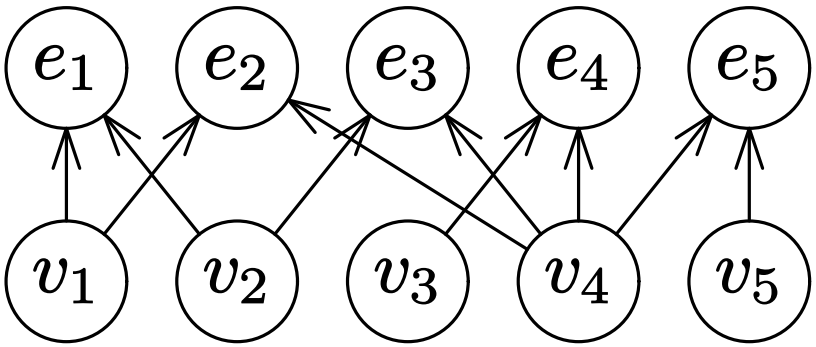


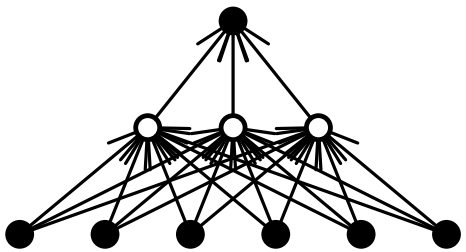
証明：クリーク問題を帰着する

**入力の変換**：




$$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\},\ \mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$$





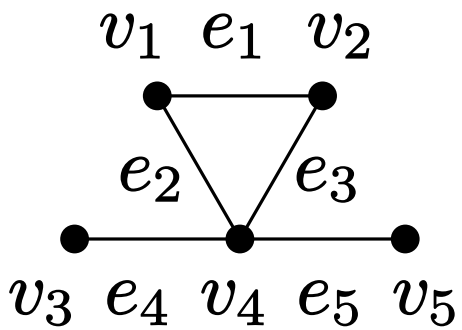

$$m = \max\{k, |\mathcal{V}| - k + \tbinom{k}{2}, |\mathcal{E}| - \tbinom{k}{2}\} + 1$$

# $P \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$：証明 (1)

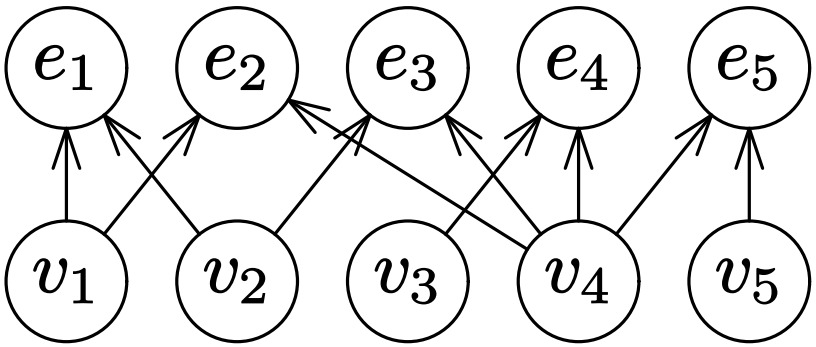


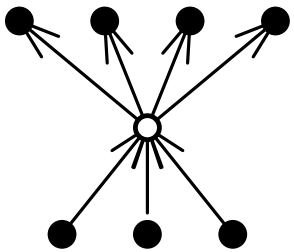
証明：クリーク問題を帰着する

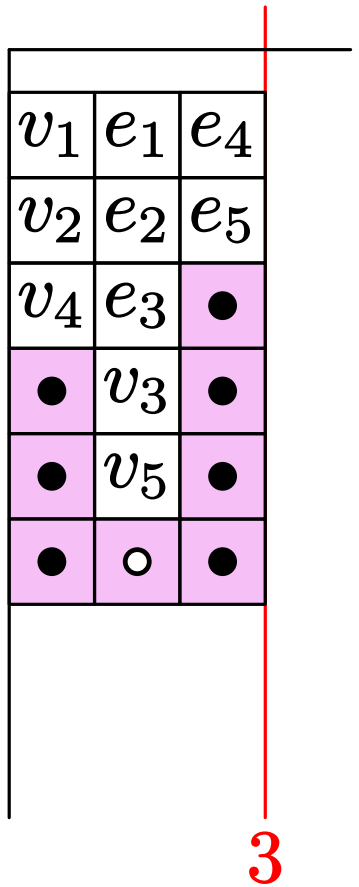
**入力の変換**：




$$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\},\ \mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$$



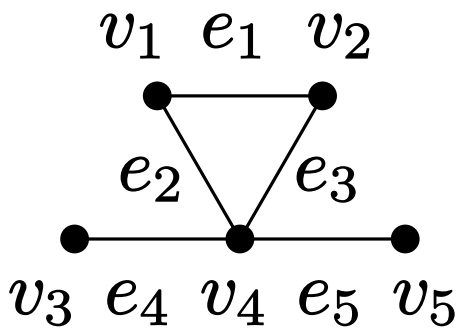


# $P \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$：証明 (1)

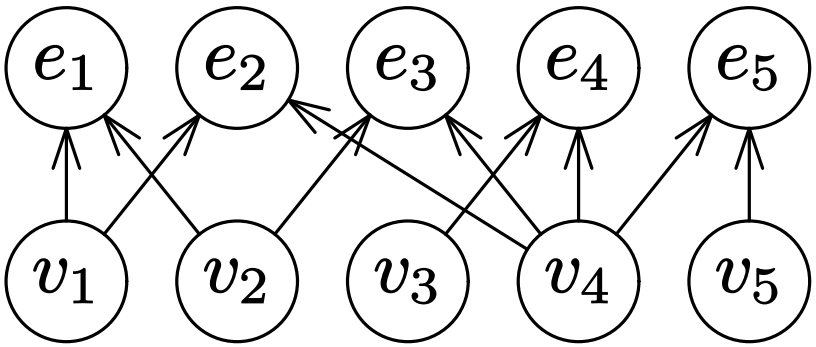


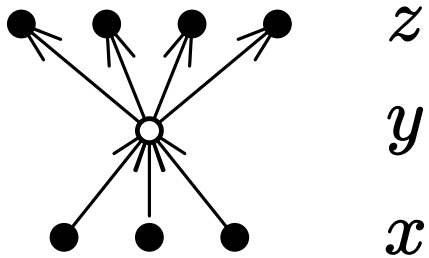
証明：クリーク問題を帰着する

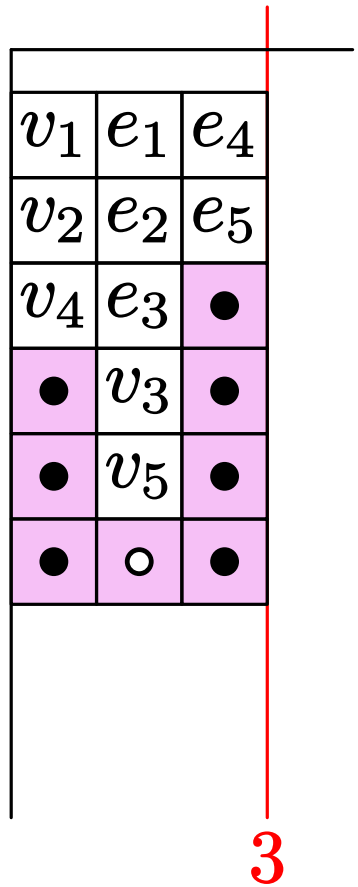
**入力の変換**：




$$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\},\ \mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$$



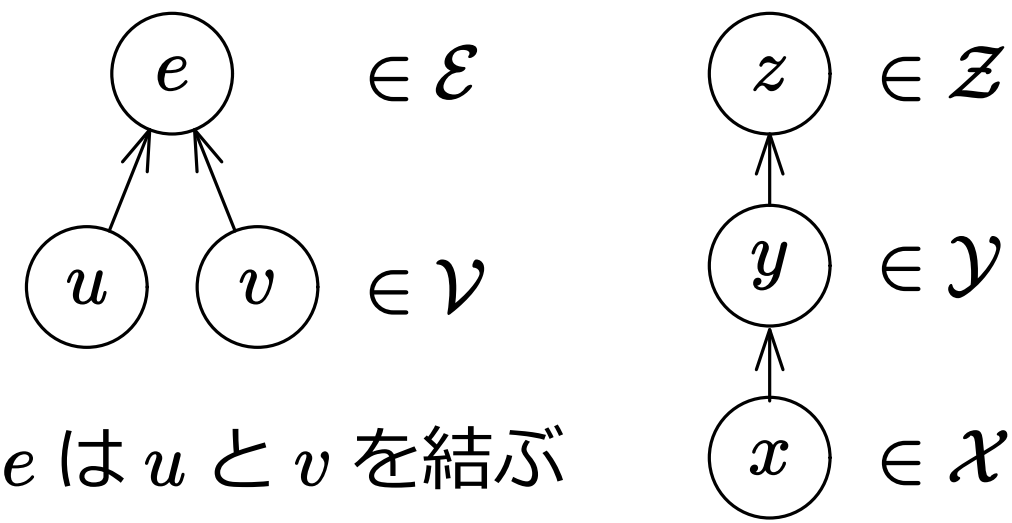
# $P \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$：証明 (2)



証明：クリーク問題を帰着する

**入力の変換**： 無向グラフ $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ と整数 $k$ に対して $\cdots$

- 機械数 $m = \max\{k, |\mathcal{V}| - k + \binom{k}{2}, |\mathcal{E}| - \binom{k}{2}\} + 1$
- ジョブの集合 $\mathcal{J}$ は次で構成
  - $\mathcal{J} = \mathcal{V} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$
  - $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ は $|\mathcal{X}| = m - k$, $|\mathcal{Y}| = m - (|\mathcal{V}| - k + \binom{k}{2})$, $|\mathcal{Z}| = m - (|\mathcal{E}| - \binom{k}{2})$ を満たす集合
- 先行制約は次のとおり

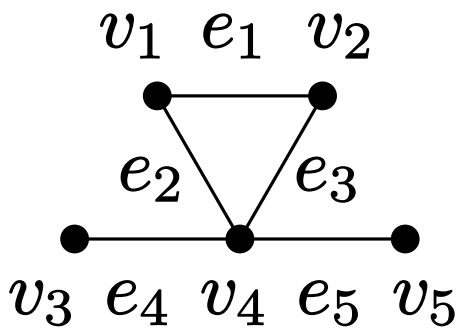


# $P \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$：証明 (3)

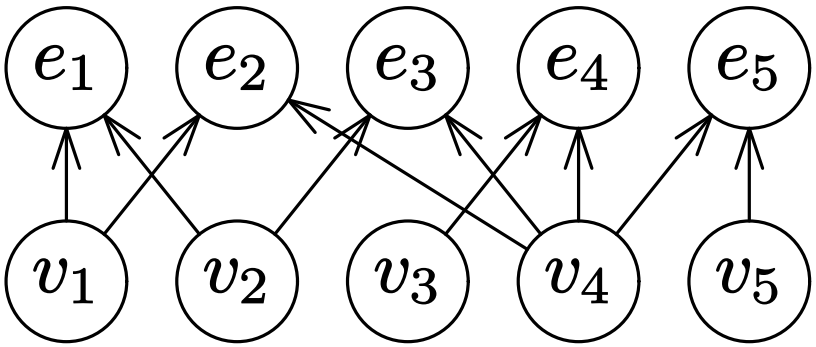


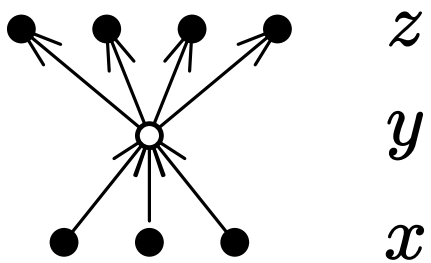
証明：クリーク問題を帰着する

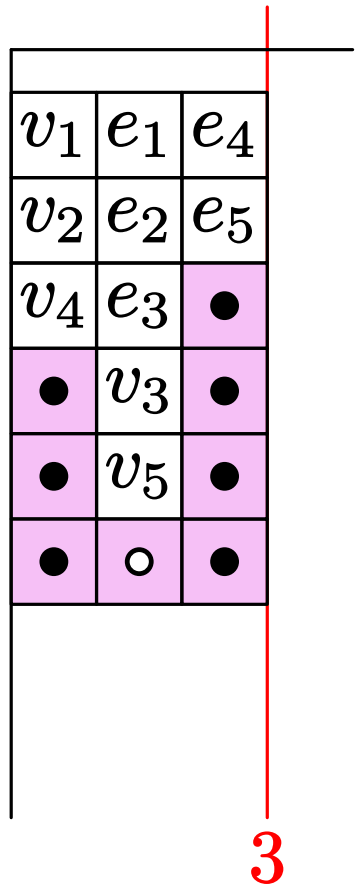
**入力の変換**：




$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$, $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5\}$




**出力の変換**：

最適値 $\leq 3$ $\Rightarrow$ クリーク問題の答えは「含む」

最適値 $> 3$ $\Rightarrow$ クリーク問題の答えは「含まない」 □

# 本日の内容



1. $Pm \mid \text{prec} \mid C_{\max}$ の困難性
2. $P \mid$ **prec** $\mid C_{\max}$ **の近似アルゴリズム**
3. $P2 \mid \text{prec}, p_j = 1 \mid C_{\max}$ のアルゴリズム

---

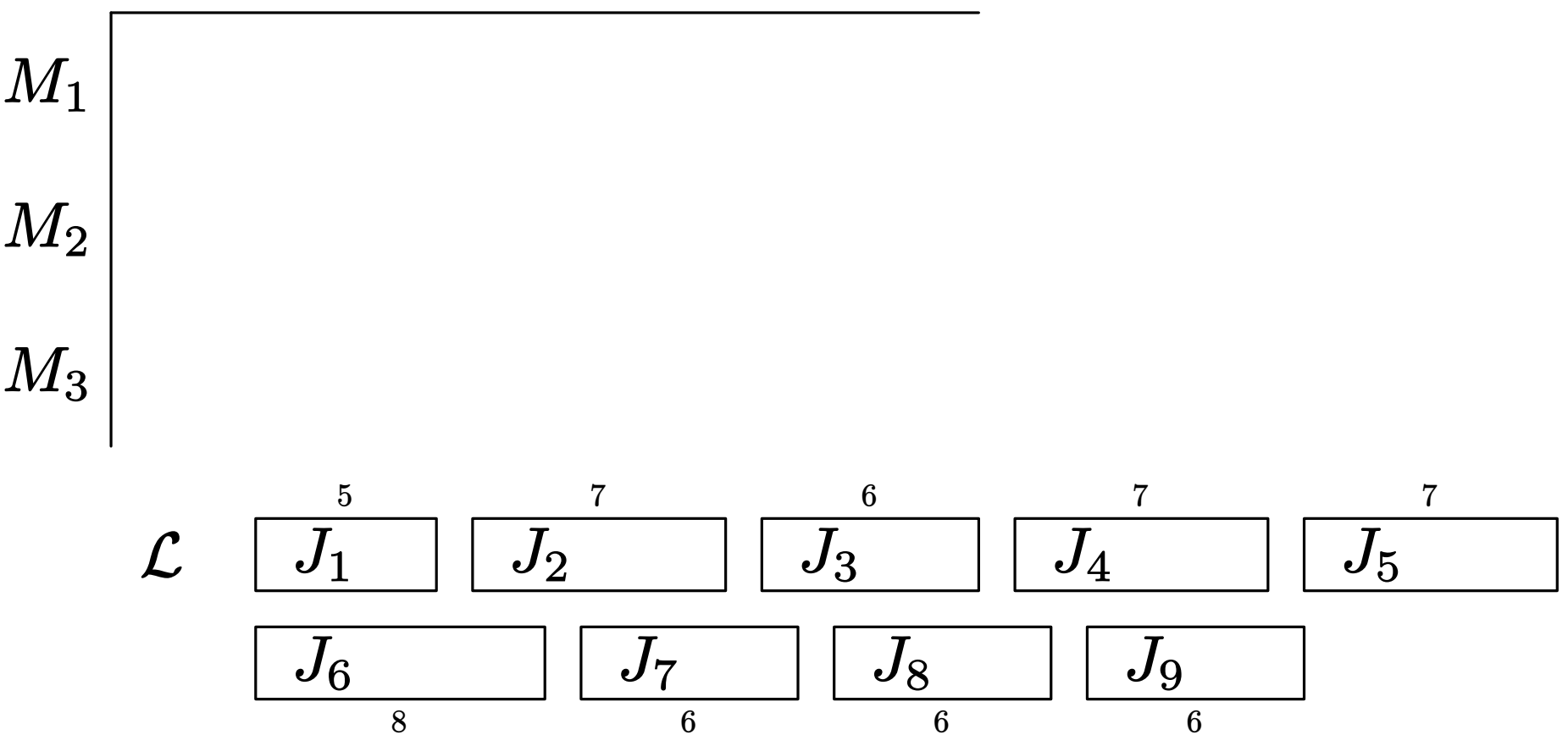

- R. L. Graham, Bounds for certain multiprocessing anomalies, *Bell System Technical Journal* 45 (1966) pp. 1563–1581.

# [復習] リスト・スケジューリング



## リスト・スケジューリング：ひな型

1. すべてのジョブを並べたリスト $\mathcal{L}$ を作る
2. 時刻を 0 から増やし，空いている機械があれば，
   それに $\mathcal{L}$ の先頭にあるジョブを割り当てる

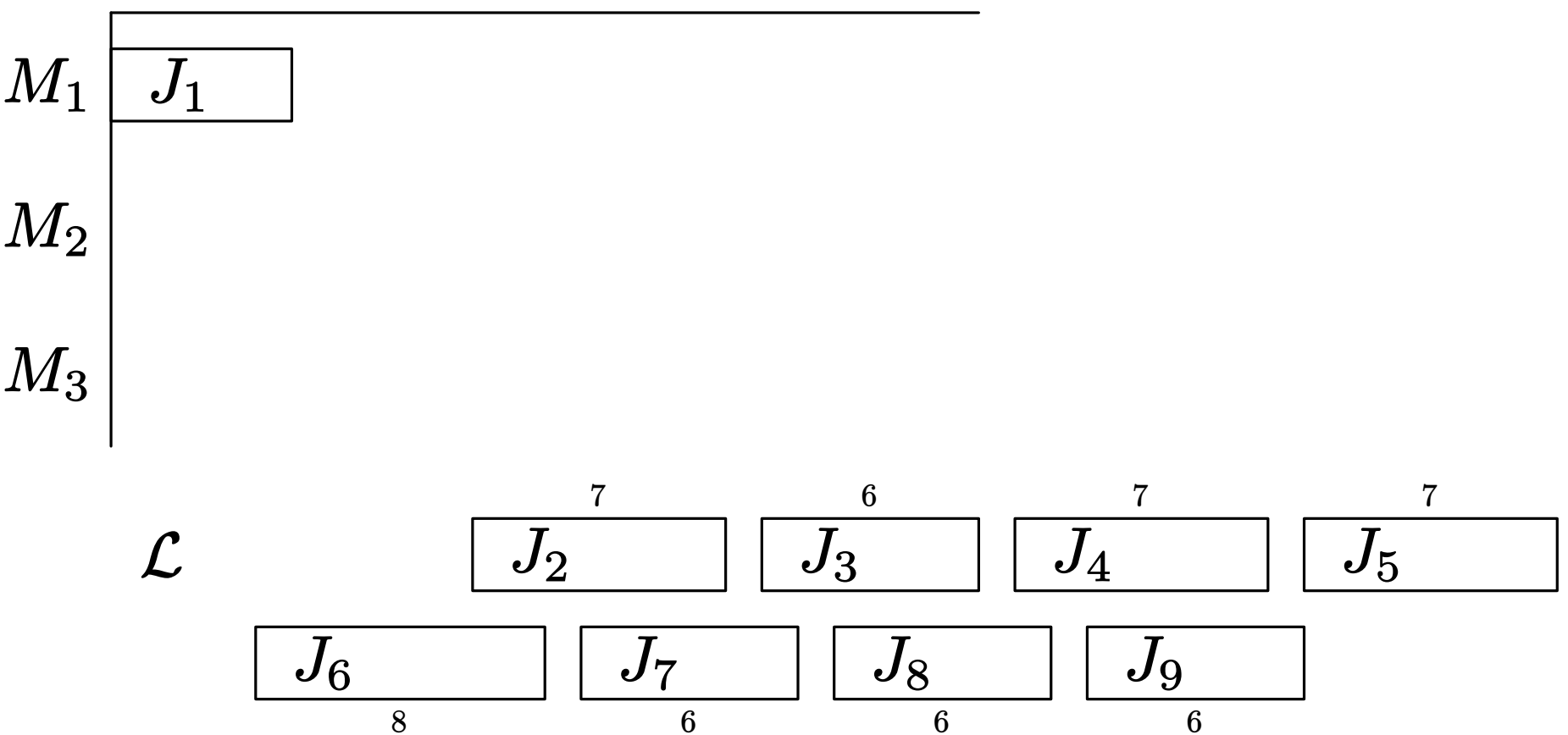


# [復習] リスト・スケジューリング



## リスト・スケジューリング：ひな型

1. すべてのジョブを並べたリスト $\mathcal{L}$ を作る
2. 時刻を 0 から増やし，空いている機械があれば，
   それに $\mathcal{L}$ の先頭にあるジョブを割り当てる

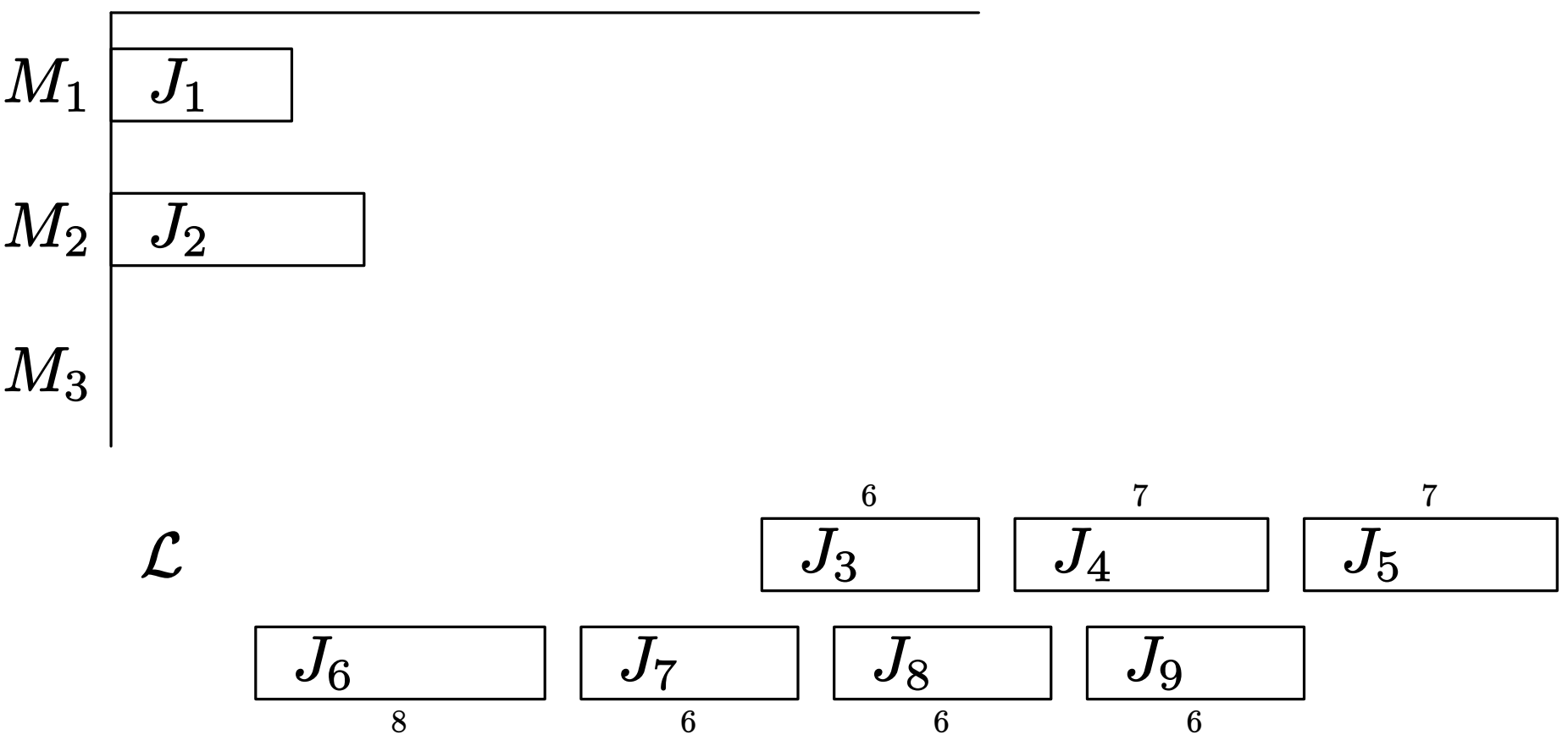


# [復習] リスト・スケジューリング



## リスト・スケジューリング：ひな型

1. すべてのジョブを並べたリスト $\mathcal{L}$ を作る
2. 時刻を 0 から増やし，空いている機械があれば，
   それに $\mathcal{L}$ の先頭にあるジョブを割り当てる

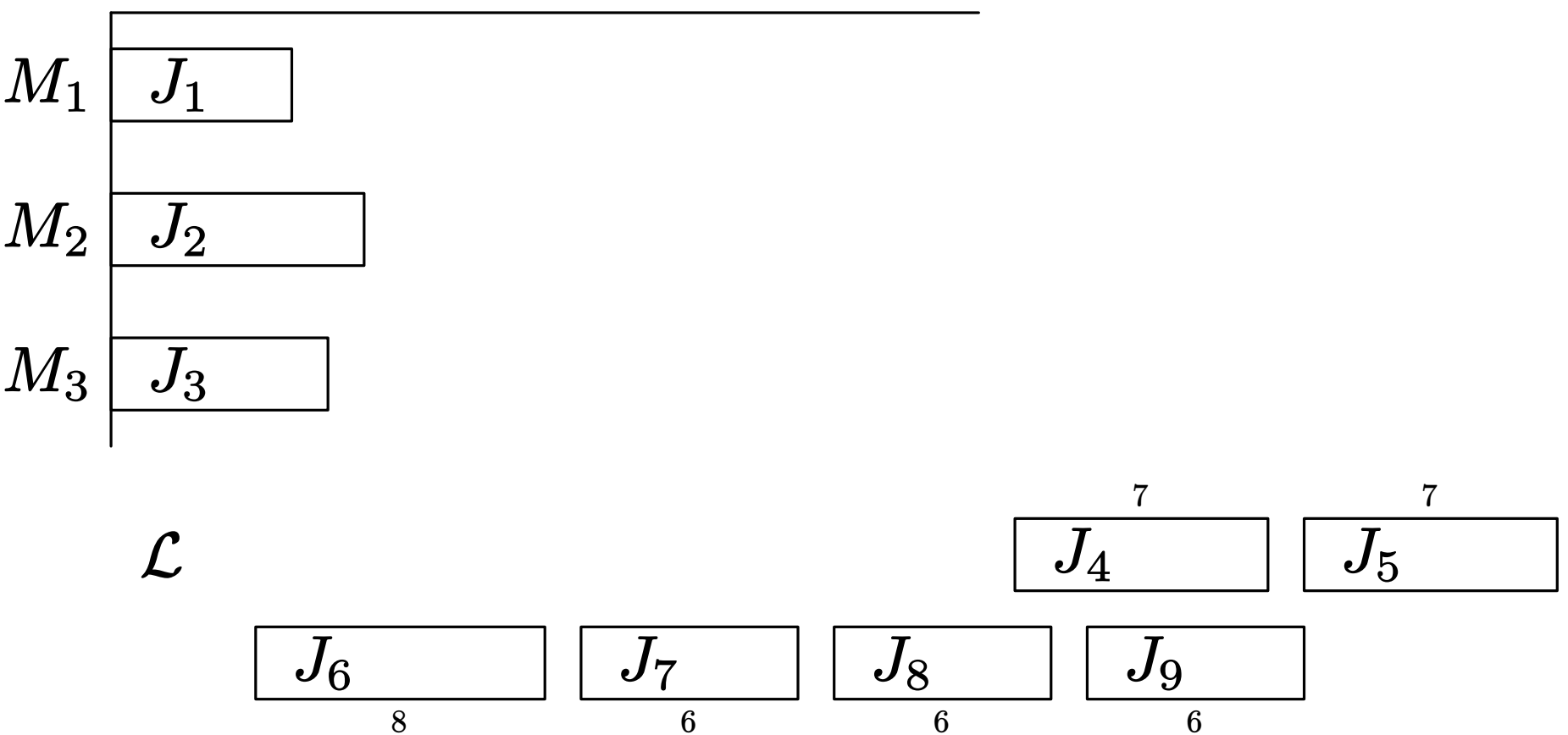


# [復習] リスト・スケジューリング



## リスト・スケジューリング：ひな型

1. すべてのジョブを並べたリスト $\mathcal{L}$ を作る
2. 時刻を 0 から増やし，空いている機械があれば，
   それに $\mathcal{L}$ の先頭にあるジョブを割り当てる

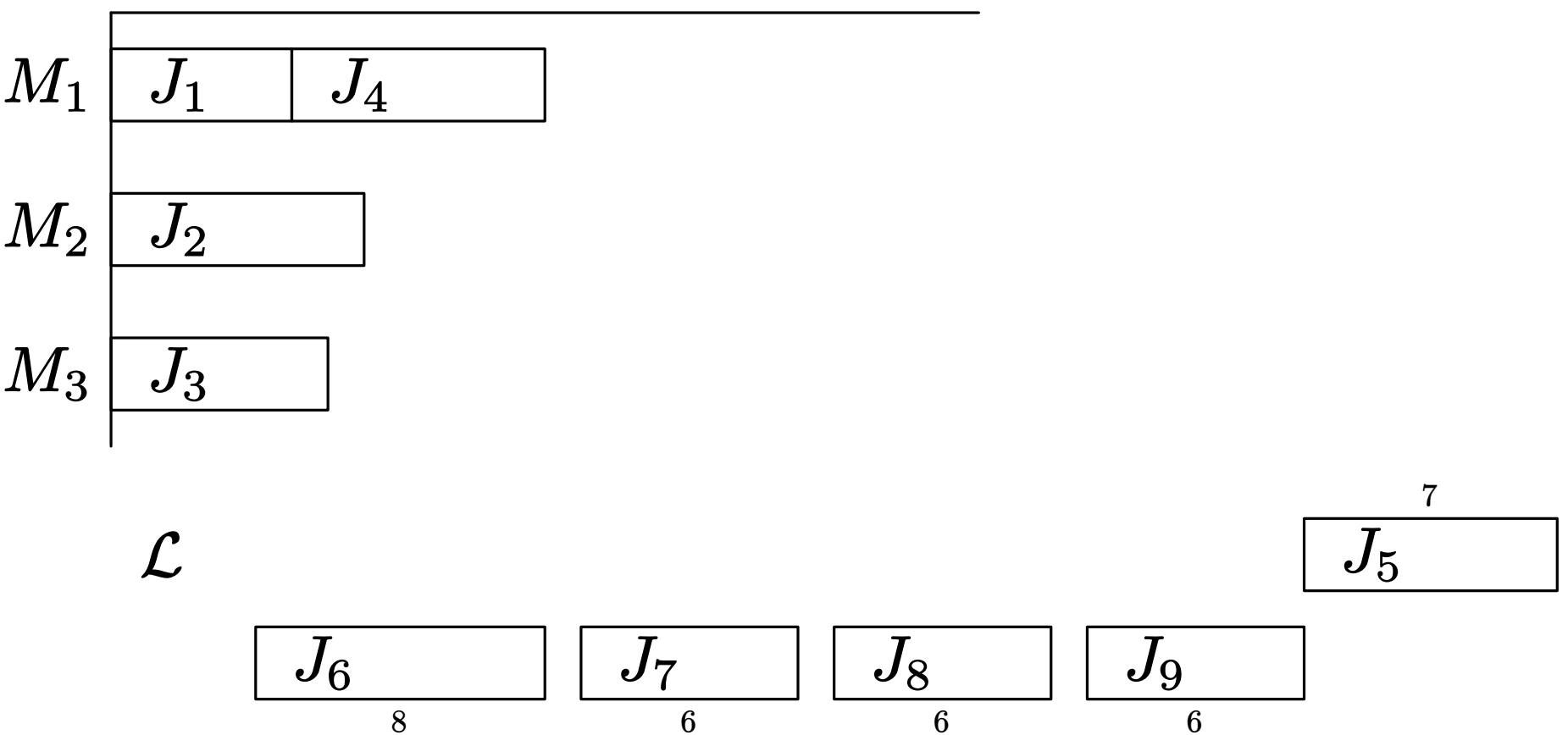


# [復習] リスト・スケジューリング



## リスト・スケジューリング：ひな型

1. すべてのジョブを並べたリスト $\mathcal{L}$ を作る
2. 時刻を 0 から増やし，空いている機械があれば，
   それに $\mathcal{L}$ の先頭にあるジョブを割り当てる

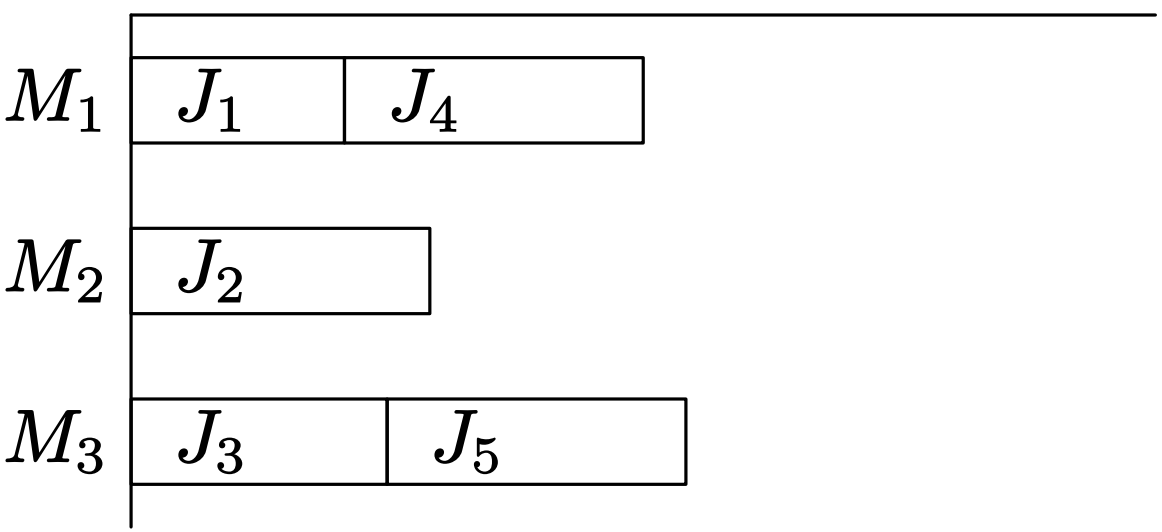


# [復習] リスト・スケジューリング

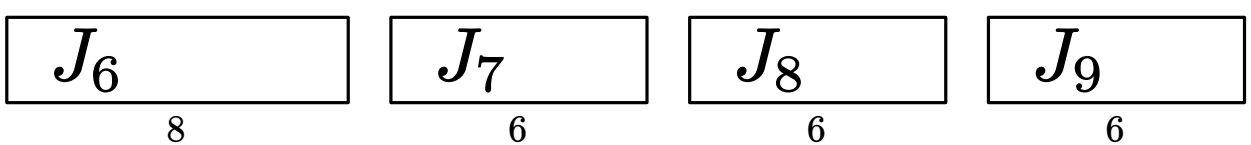


## リスト・スケジューリング：ひな型

1. すべてのジョブを並べたリスト $\mathcal{L}$ を作る
2. 時刻を 0 から増やし，空いている機械があれば，
   それに $\mathcal{L}$ の先頭にあるジョブを割り当てる




$\mathcal{L}$

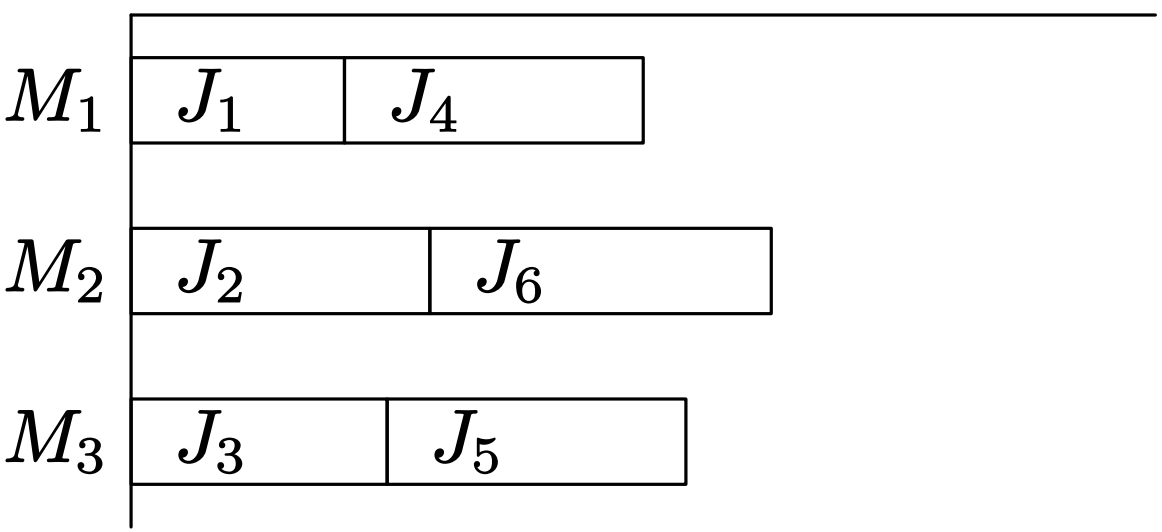


# [復習] リスト・スケジューリング

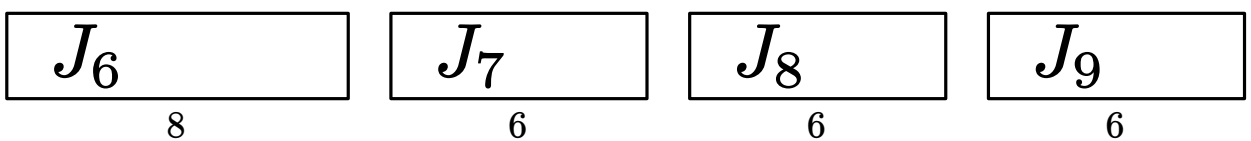


## リスト・スケジューリング：ひな型

1. すべてのジョブを並べたリスト $\mathcal{L}$ を作る
2. 時刻を 0 から増やし，空いている機械があれば，
   それに $\mathcal{L}$ の先頭にあるジョブを割り当てる




$\mathcal{L}$

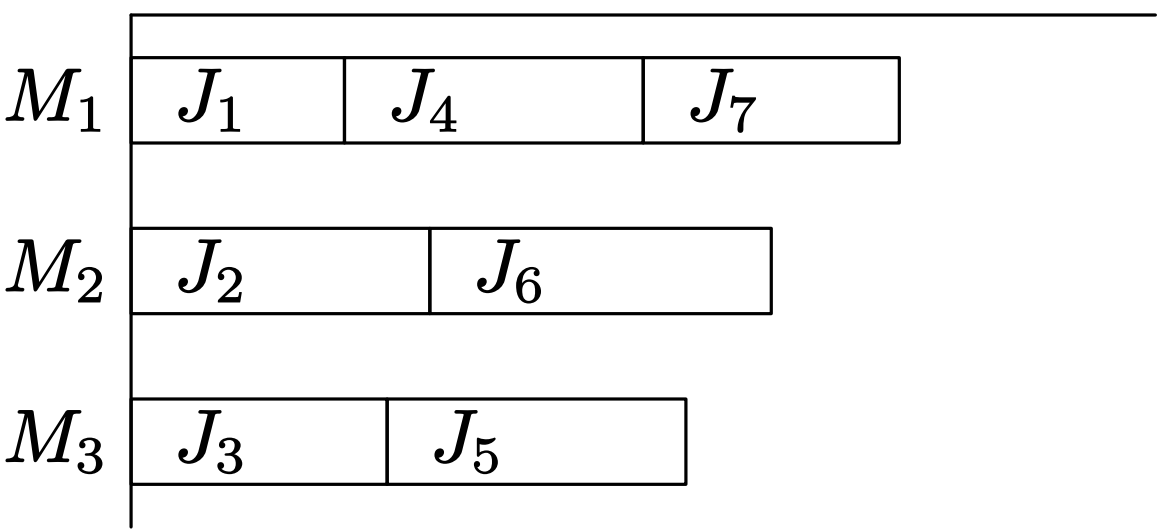


# [復習] リスト・スケジューリング

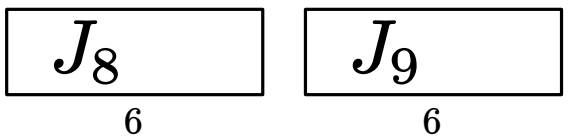


## リスト・スケジューリング：ひな型

1. すべてのジョブを並べたリスト $\mathcal{L}$ を作る
2. 時刻を 0 から増やし，空いている機械があれば，
   それに $\mathcal{L}$ の先頭にあるジョブを割り当てる




$\mathcal{L}$

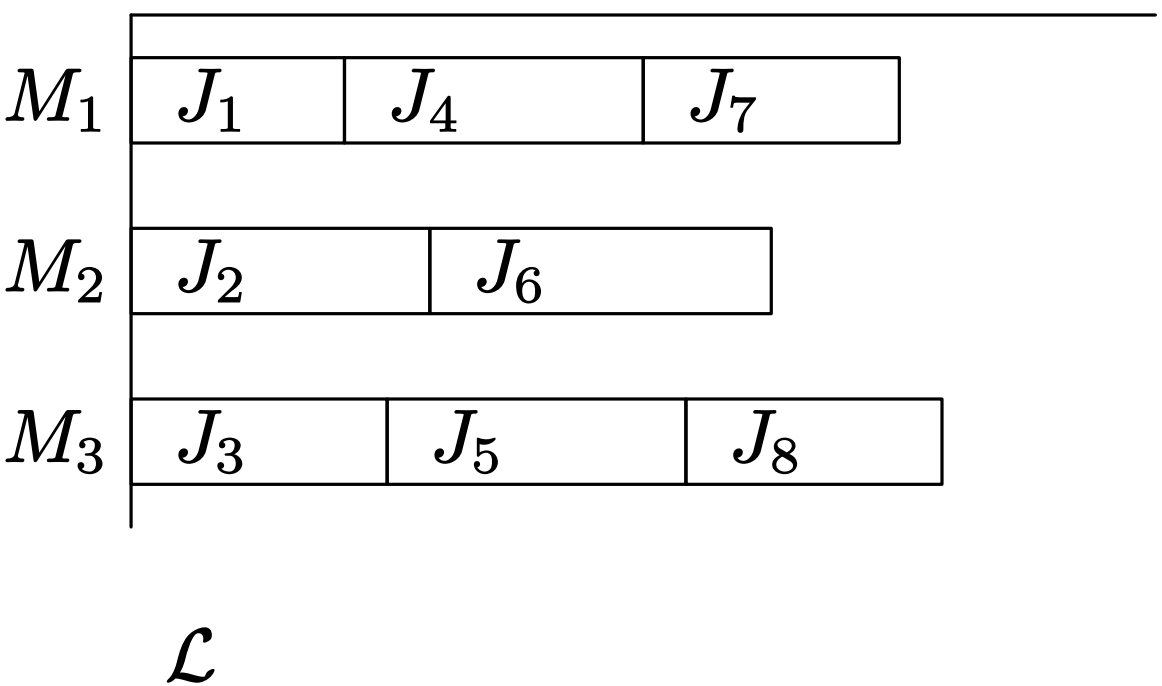


# [復習] リスト・スケジューリング

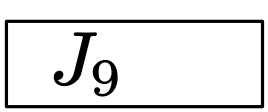


## リスト・スケジューリング：ひな型

1. すべてのジョブを並べたリスト $\mathcal{L}$ を作る
2. 時刻を 0 から増やし，空いている機械があれば，
   それに $\mathcal{L}$ の先頭にあるジョブを割り当てる




$\mathcal{L}$

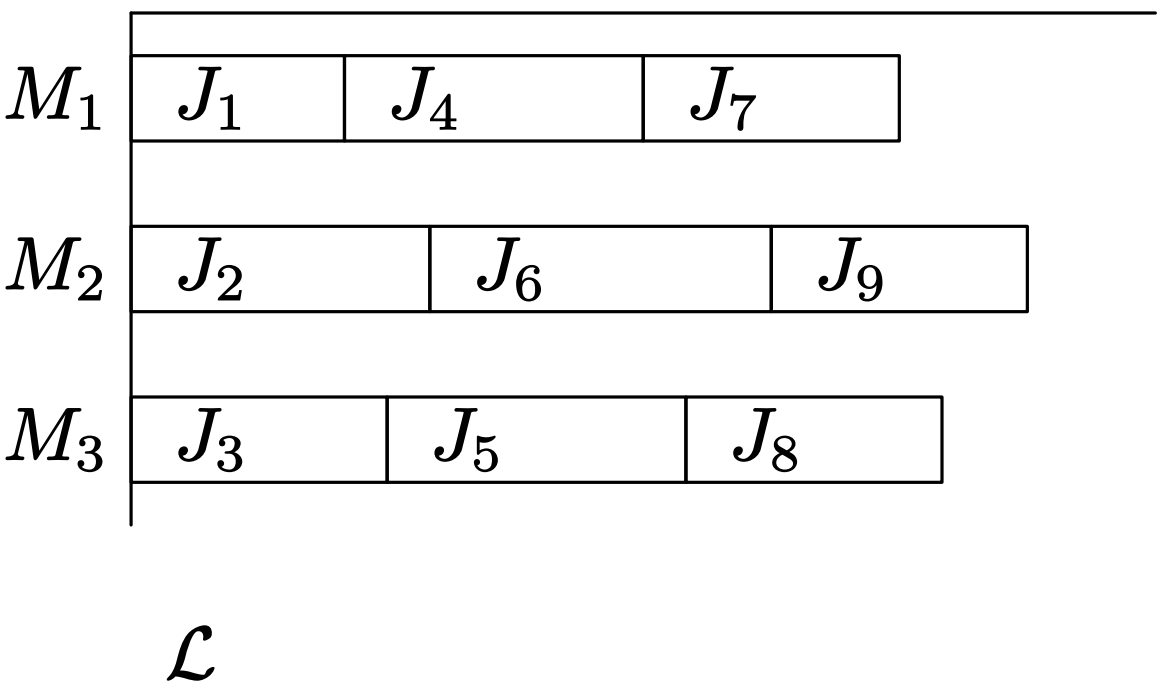


# [復習] リスト・スケジューリング



## リスト・スケジューリング：ひな型

1. すべてのジョブを並べたリスト $\mathcal{L}$ を作る
2. 時刻を 0 から増やし，空いている機械があれば，
   それに $\mathcal{L}$ の先頭にあるジョブを割り当てる




$\mathcal{L}$

なにかしらの

スケジュールが

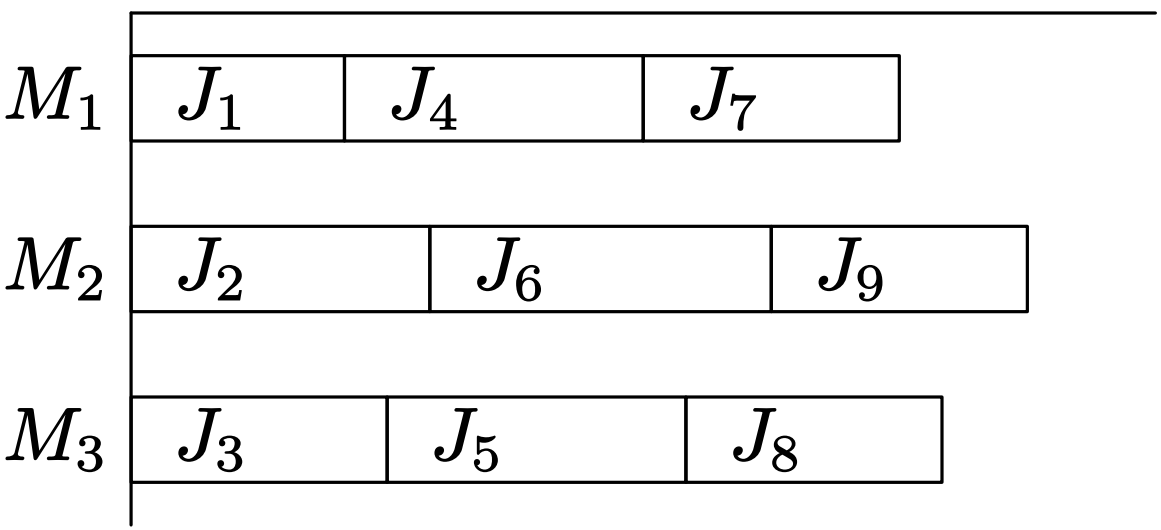
得られる

# [復習] リスト・スケジューリング



## リスト・スケジューリング：ひな型

1. すべてのジョブを並べたリスト $\mathcal{L}$ を作る
2. 時刻を 0 から増やし，空いている機械があれば，
   それに $\mathcal{L}$ の先頭にあるジョブを割り当てる



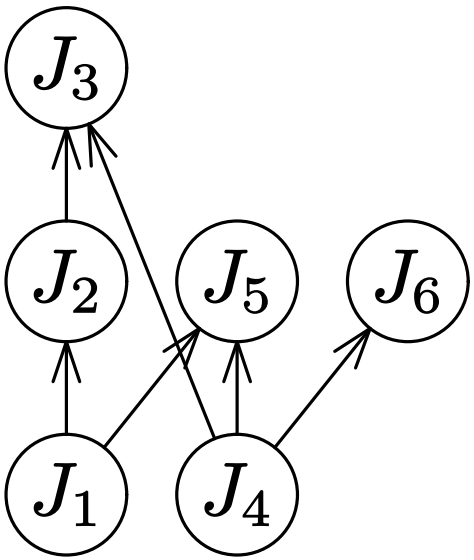

なにかしらの
スケジュールが
得られる

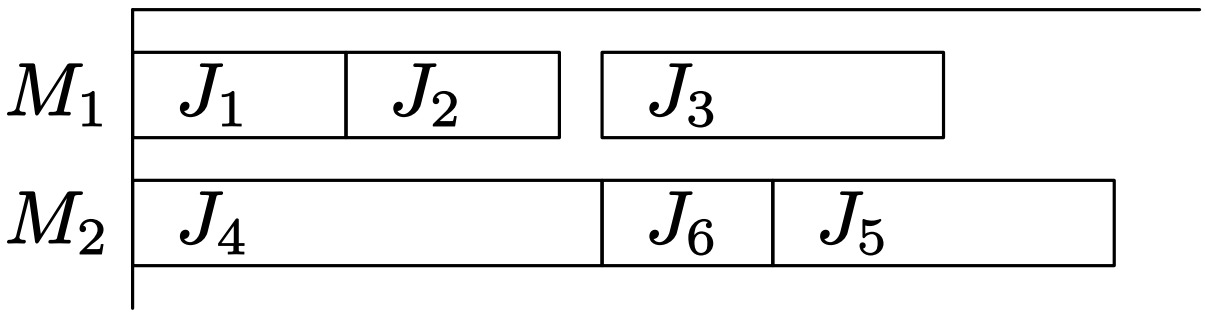
## 疑問

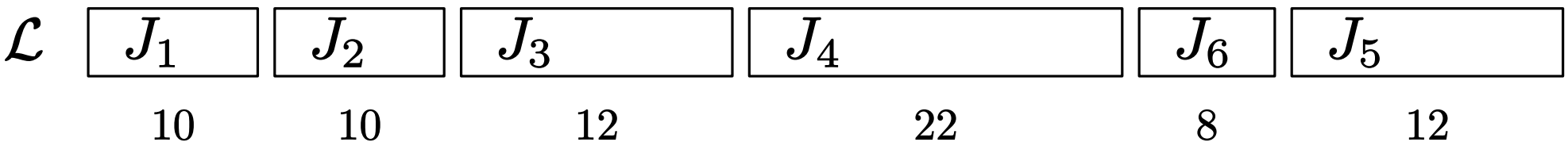
このスケジュールは **どれくらい良い** のか？

## 先行制約がある場合のリスト・スケジューリング

処理を開始できるジョブの中で，
$\mathcal{L}$ において最も前にあるものを 次に処理する



# $P \mid \mathrm{prec} \mid C_{\max}$ と任意のリスト



**定理：$P \mid \mathrm{prec} \mid C_{\max}$ と任意リスト (Graham '66)**

$P \mid \mathrm{prec} \mid C_{\max}$ において，
任意のリストに対するリスト・スケジューリングは
$\left(2 - \frac{1}{m}\right)$ 近似アルゴリズムである ($m$ は機械数)

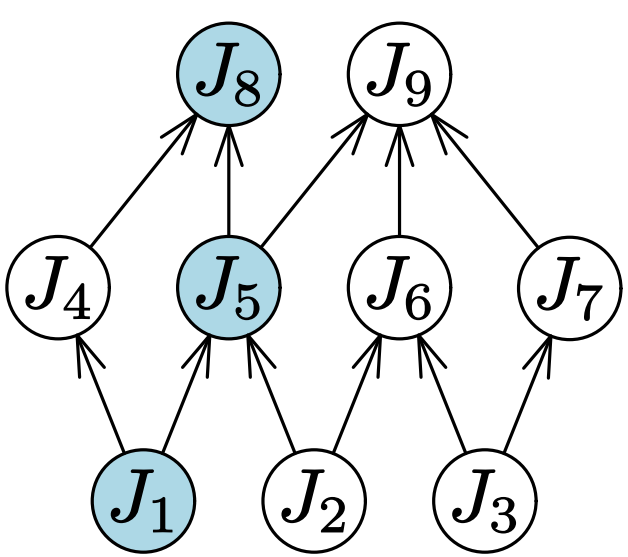
注：$P \mid\mid C_{\max}$ において，(先行制約がないとき)
任意のリストに対するリスト・スケジューリングは
$\left(2 - \frac{1}{m}\right)$ 近似アルゴリズムで，この近似比はタイト

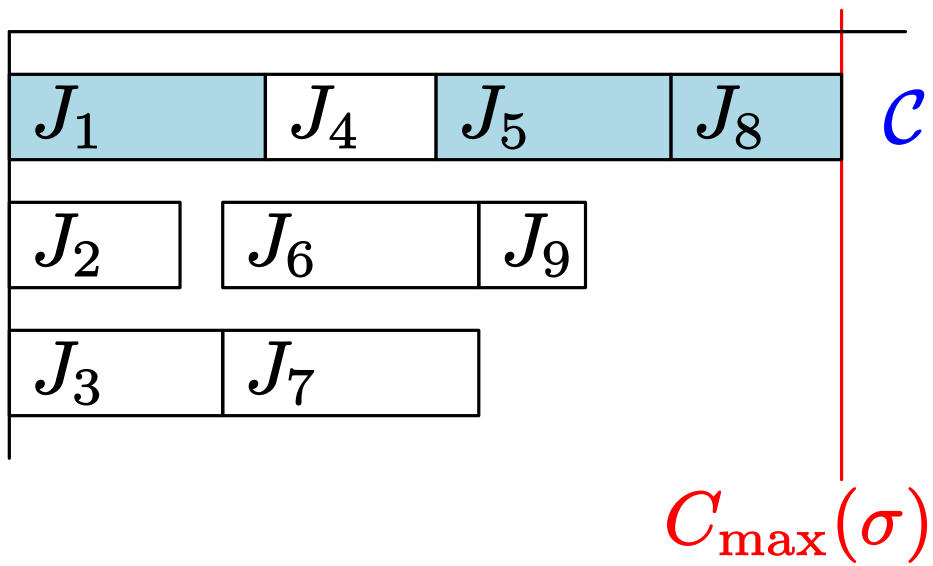
# $P \mid \mathrm{prec} \mid C_{\max}$：証明 (1)



証明：任意の入力を考えて，最適値を $C^*_{\max}$ とする

- アルゴリズムの出力スケジュールを $\sigma$ とする
- $\sigma$ において，次のようなジョブ集合 $\mathcal{C}$ を見つける
  - 最後に完了するジョブ 1 つを $J$ として，$\mathcal{C}$ に入れる
  - $J' \to J$ であるジョブ $J'$ で最後に完了するもの 1 つを $J''$ として，$\mathcal{C}$ に入れる
  - できなくなるまで，これを繰り返す



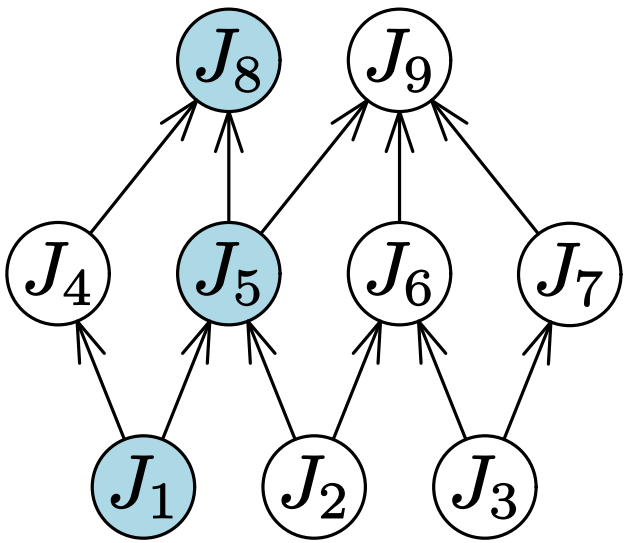



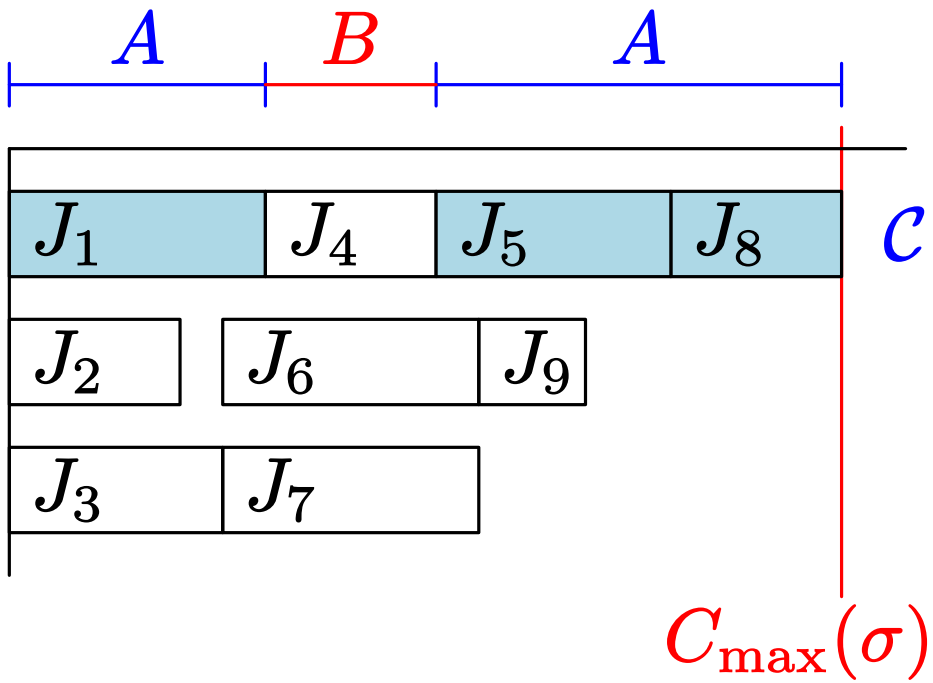
# $P \mid \mathrm{prec} \mid C_{\max}$：証明 (2)



証明 (続)：

- 時間 $[0, C_{\max}(\sigma)]$ を次の 2 つに分ける
  - $A = \mathcal{C}$ のジョブを処理している時間
  - $B =$ それ以外の時間
- $B$ の間は，すべての機械がジョブを処理している (なぜ？)



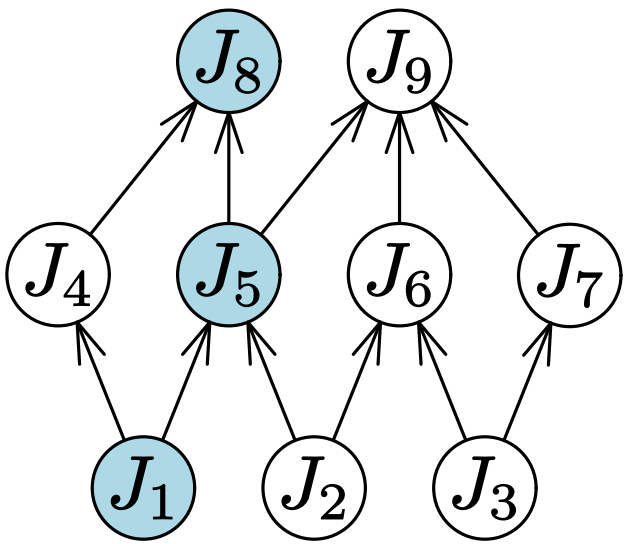
# $P \mid \mathrm{prec} \mid C_{\max}$：証明 (3)

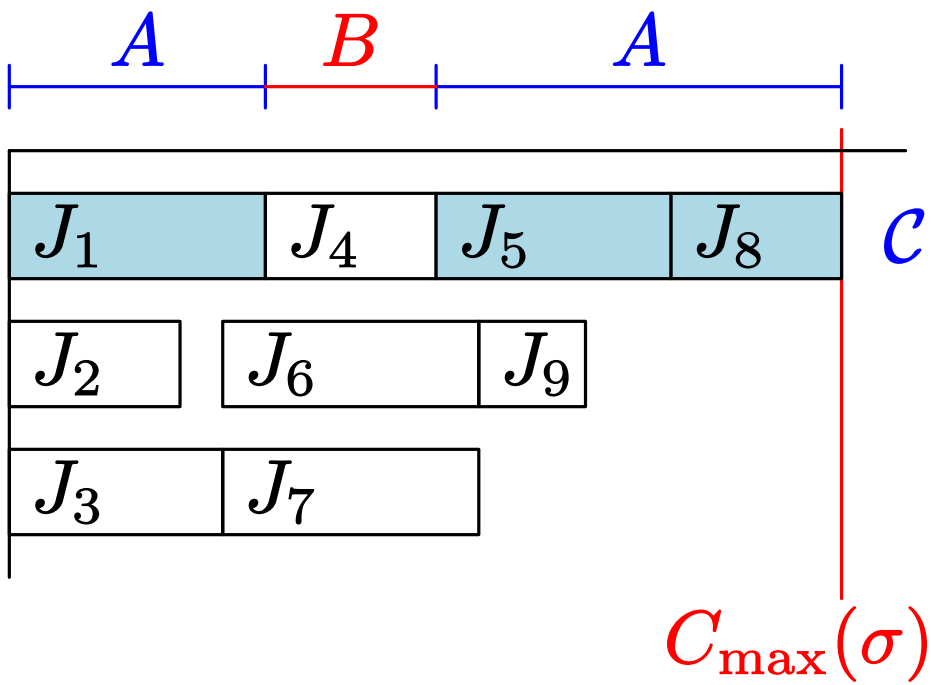


証明 (続)：

- したがって，次が成り立つ
  - $A$ の時間 $= \sum_{J_j \in \mathcal{C}} p_j \leq C^*_{\max}$
  - $B$ の時間 $\leq \frac{1}{m} \sum_{J_j \in \mathcal{J} - \mathcal{C}} p_j \leq \frac{1}{m} \sum_{J_j \in \mathcal{J}} p_j \leq C^*_{\max}$

先行制約がないときと同じ理由



# $P \mid \mathrm{prec} \mid C_{\max}$：証明 (3)



証明 (続)：

- したがって，次が成り立つ
  - $A$ の時間 $= \sum_{J_j \in \mathcal{C}} p_j \leq C^*_{\max}$
  - $B$ の時間 $\leq \frac{1}{m} \sum_{J_j \in \mathcal{J}-\mathcal{C}} p_j \leq \frac{1}{m} \sum_{J_j \in \mathcal{J}} p_j \leq C^*_{\max}$
- ゆえに，$C_{\max}(\sigma) = A$ の時間 $+ B$ の時間

$$\begin{aligned} &\leq \sum_{J_j \in \mathcal{C}} p_j + \frac{1}{m} \sum_{J_j \in \mathcal{J}-\mathcal{C}} p_j \\ &= \frac{1}{m} \sum_{J_j \in \mathcal{J}} p_j + \left(1 - \frac{1}{m}\right) \sum_{J_j \in \mathcal{C}} p_j \\ &\leq \left(2 - \frac{1}{m}\right) C^*_{\max} \end{aligned}$$

□

# 本日の内容



1. $Pm \mid \text{prec} \mid C_{\max}$ の困難性
2. $P \mid \text{prec} \mid C_{\max}$ の近似アルゴリズム
3. $P2 \mid \textbf{prec}, p_j = 1 \mid C_{\max}$ **のアルゴリズム**

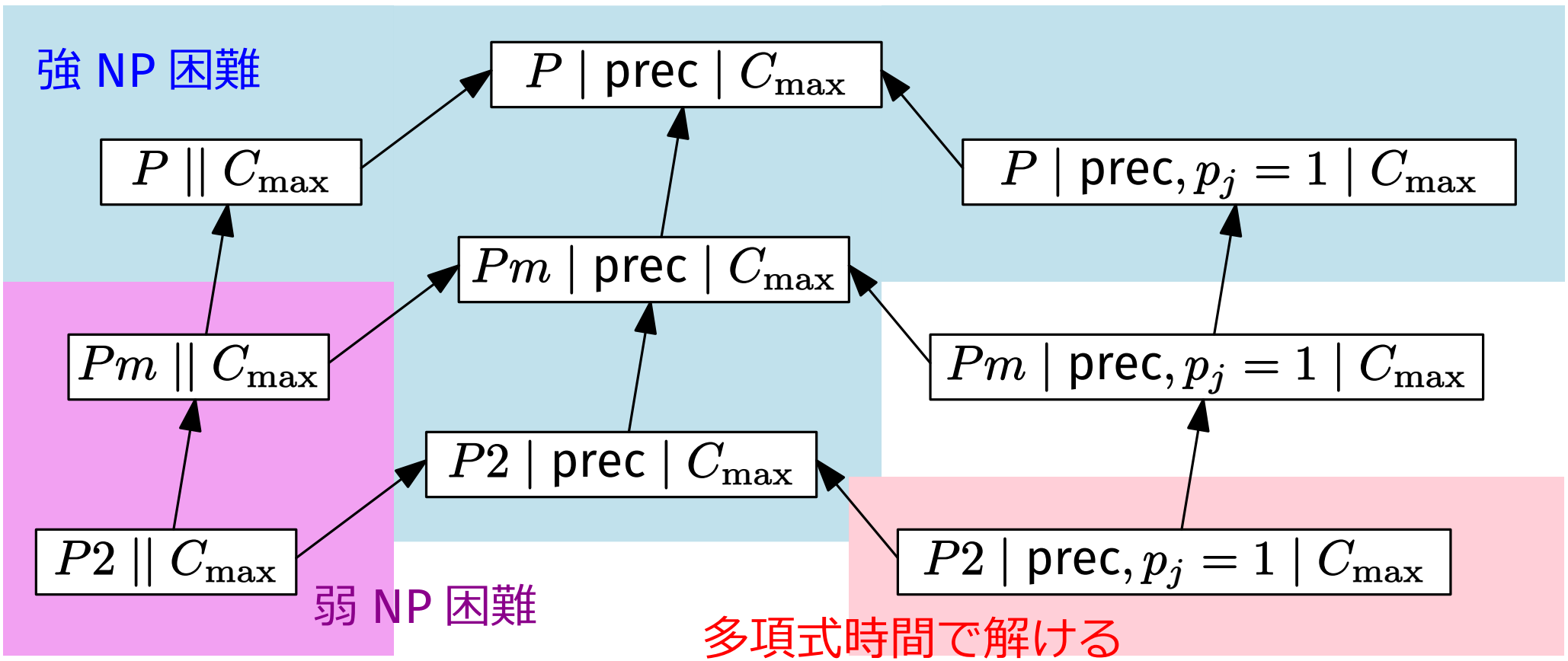
---


- M. Fujii, T. Kasami, K. Ninomiya, Optimal sequencing of two equivalent processors. *SIAM Journal on Applied Mathematics* 17 (1969) pp. 784–789. Erratum, *SIAM Journal on Applied Mathematics* 20 (1971) p. 141.
- E. G. Coffman, Jr., R. L. Graham, Optimal scheduling for two-processor systems. *Acta Informatica* 1 (1972) pp. 200–213.
- R. Sethi, Scheduling graphs on two processors. *SIAM Journal on Computing* 10 (1975) pp. 73–82.
- H. N. Gabow, An almost-linear algorithm for two-processor scheduling. *Journal of the Association for Computing Machinery* 29 (1982) pp. 766–780.
- H. N. Gabow, R. E. Tarjan, A linear-time algorithm for a special case of disjoint set union. *Journal of Computer and System Sciences* 30 (1985) pp. 209–221.

# 今日の内容：$Pm \mid \text{prec} \mid C_{\max}$ 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



# $P2 \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$ のアルゴリズム



**定理**

問題 $P2 \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$ は多項式時間で解ける

| 計算量 | (文献) |
|---|---|
| $O(n^3)$ | (Fujii, Kasami, Ninomiya '69) |
| $O(n^2)$ | (Coffman, Graham '72) |
| $O(n\alpha(n) + h)$ | (Sethi '75, Gabow '82) |
| $O(n + h)$ | (Gabow, Tarjan '85) |
| $\vdots$ | |

$n =$ ジョブ数，$h =$ 先行制約を表す有向グラフの辺数

$\alpha(n) =$ 逆アッカーマン関数 (inverse Ackermann function)

# $P2 \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$ のアルゴリズム



**定理**

問題 $P2 \mid \mathrm{prec}, p_j = 1 \mid C_{\max}$ は多項式時間で解ける

| 計算量 | (文献) |
|---|---|
| $O(n^3)$ | (Fujii, Kasami, Ninomiya '69) |
| $O(n^2)$ | (Coffman, Graham '72) |
| $O(n\alpha(n) + h)$ | (Sethi '75, Gabow '82) |
| $O(n + h)$ | (Gabow, Tarjan '85) |
| $\vdots$ | |

$n =$ ジョブ数，$h =$ 先行制約を表す有向グラフの辺数

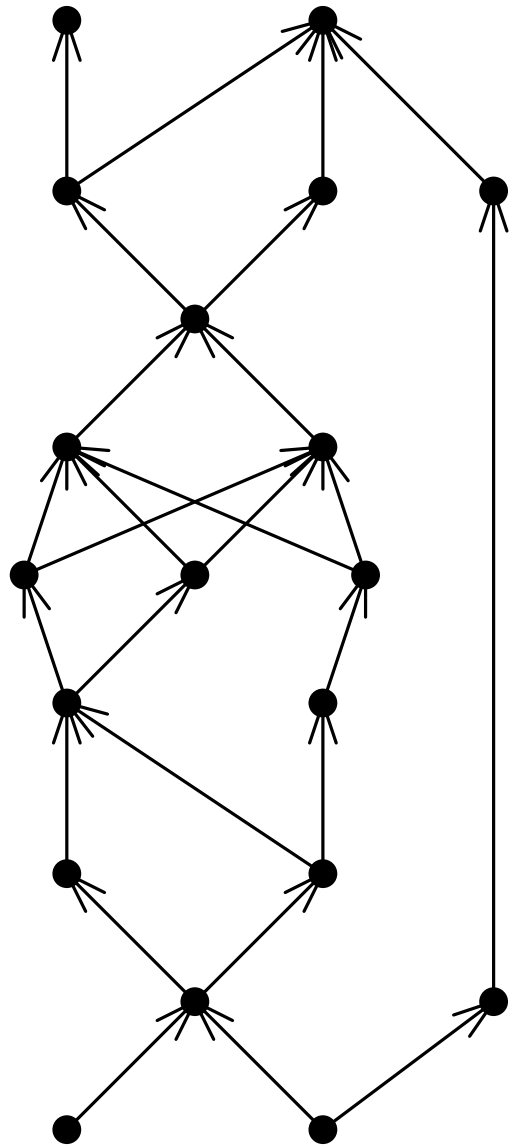
$\alpha(n) =$ 逆アッカーマン関数 (inverse Ackermann function)

# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

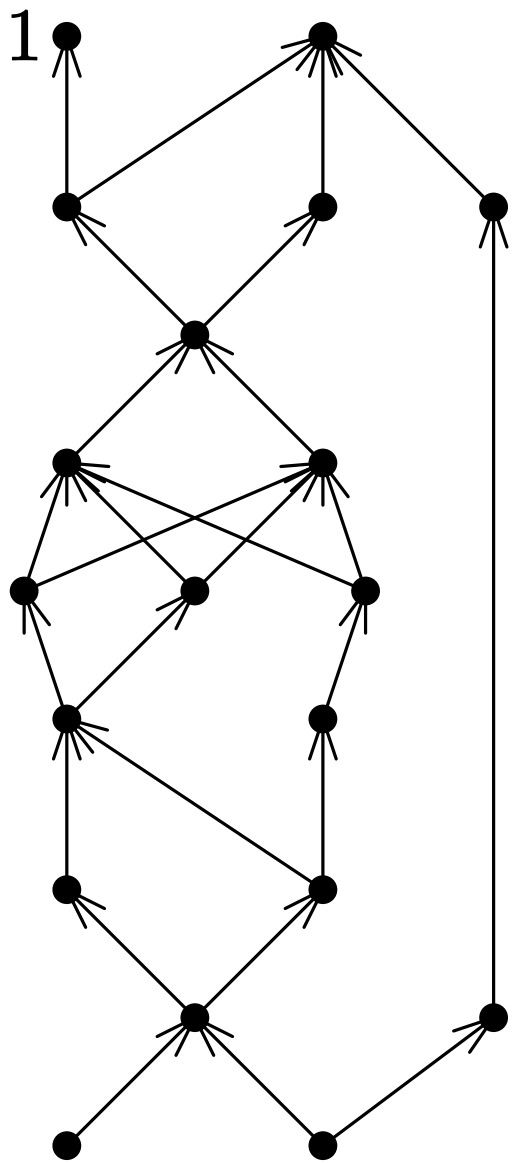


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

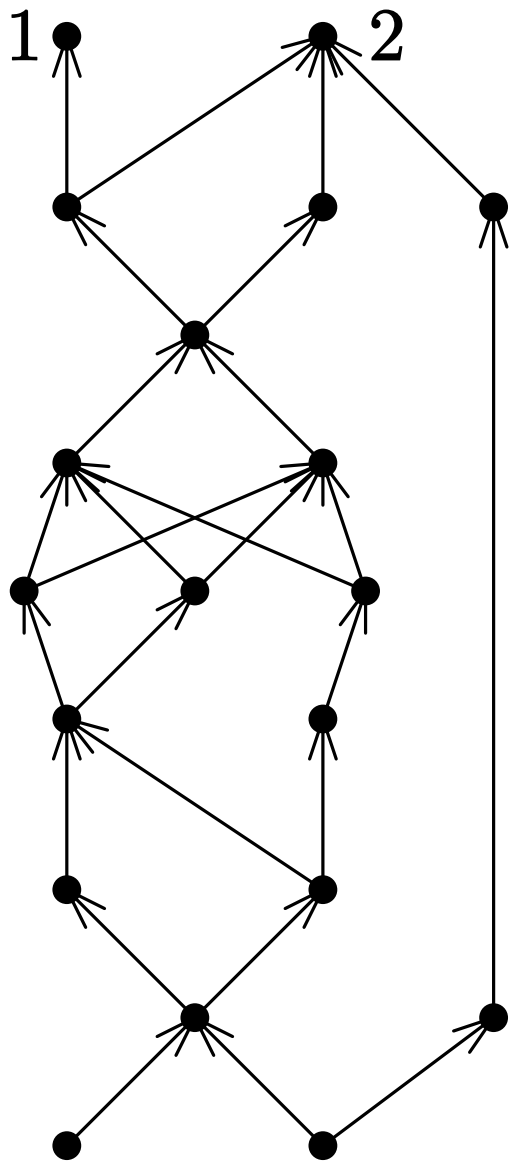


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

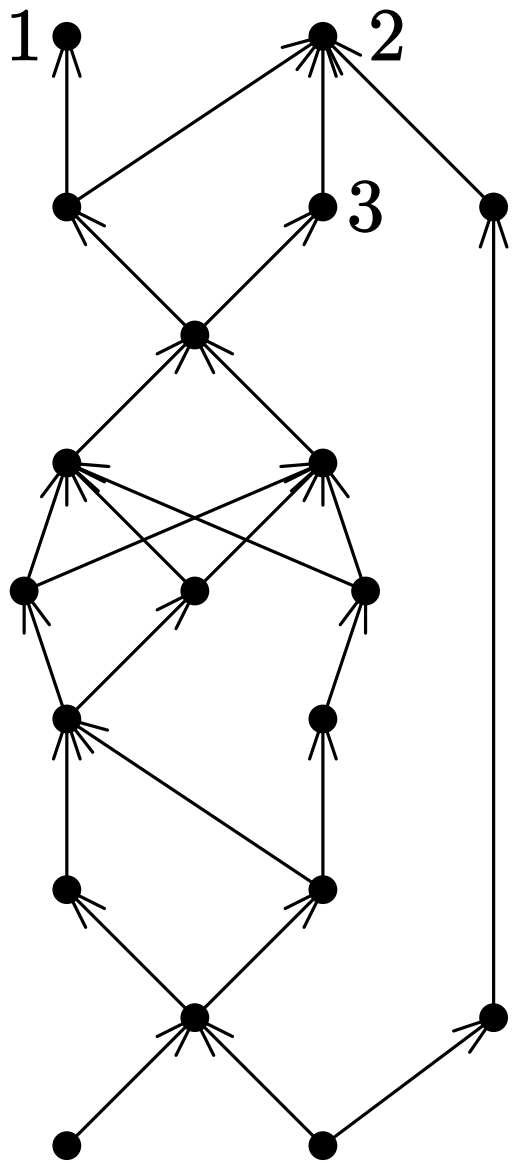


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

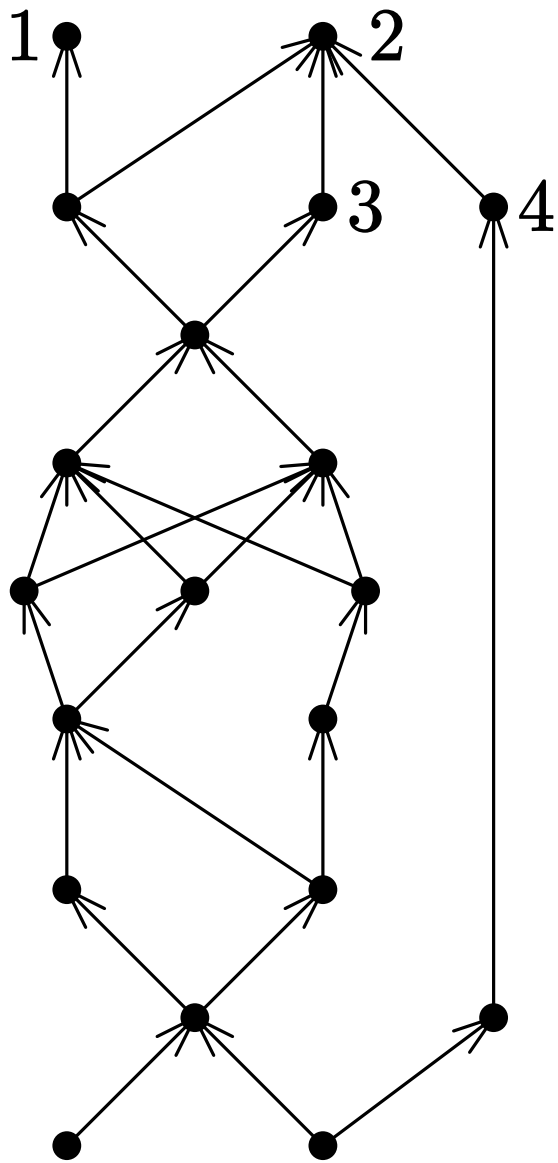


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

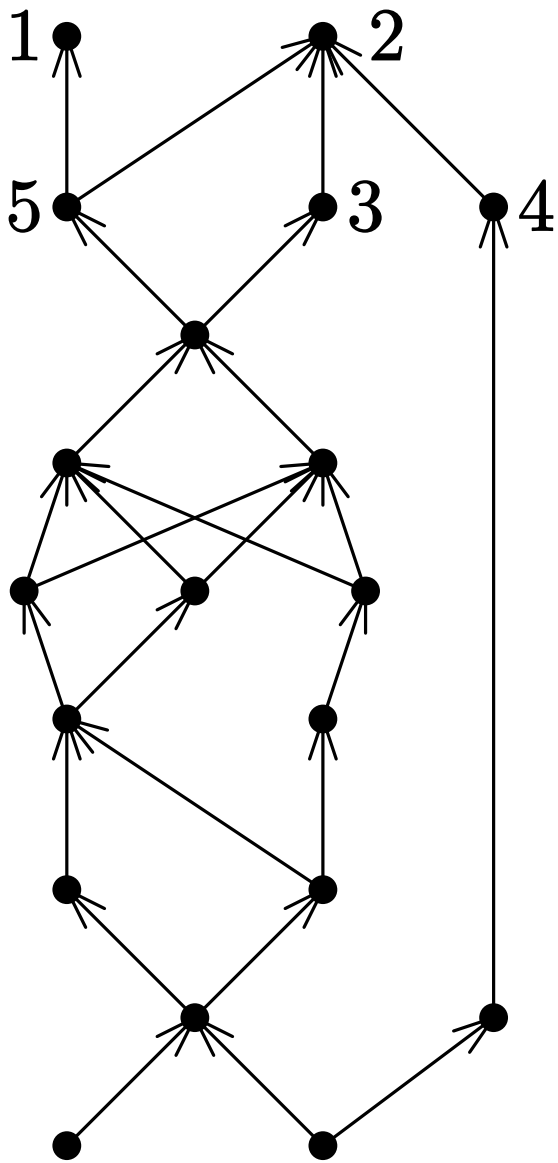


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

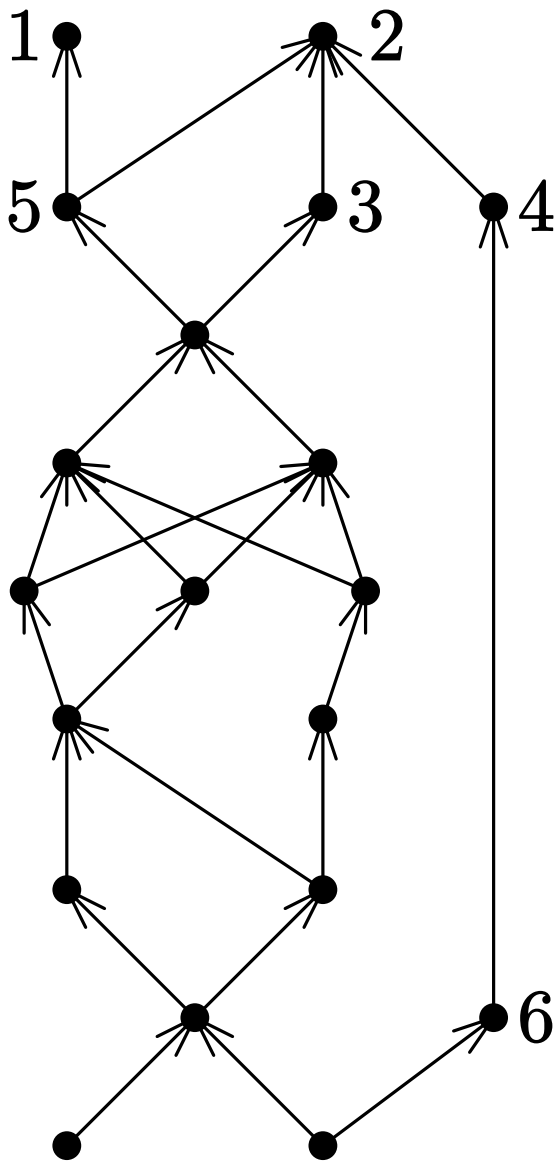


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

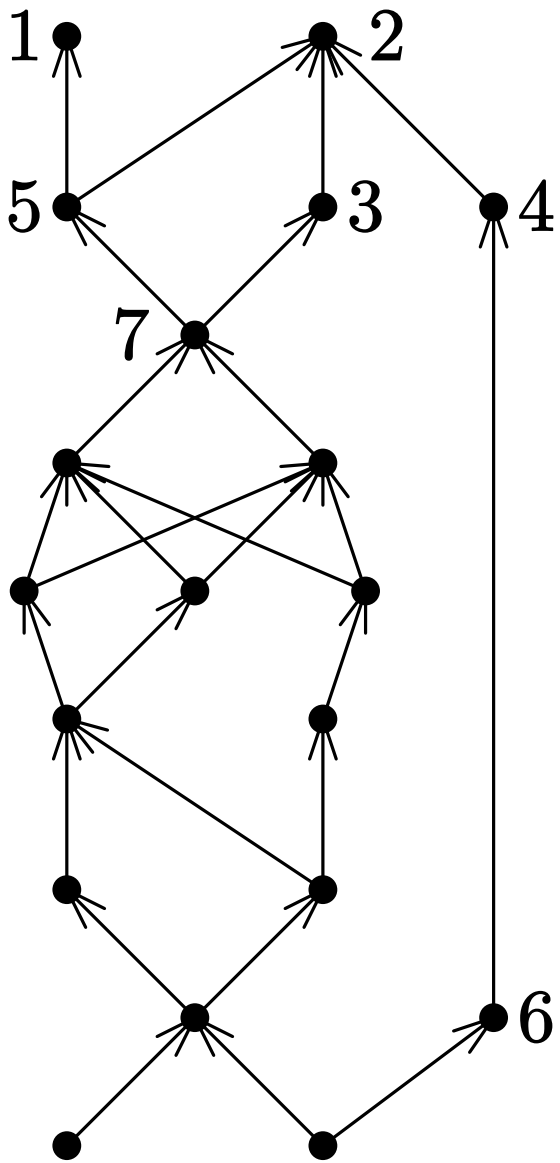


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

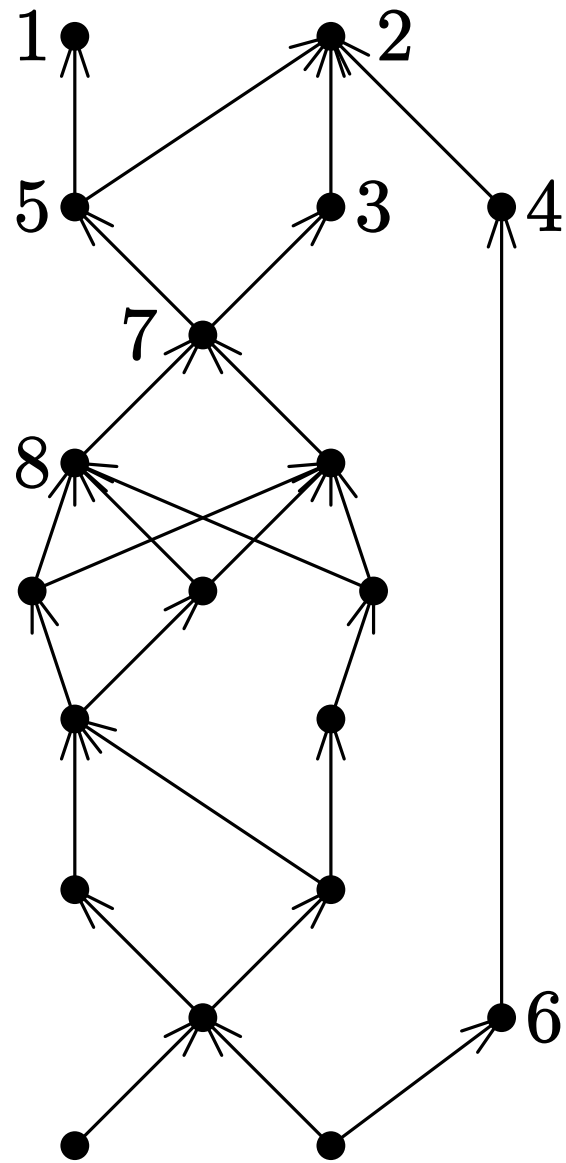


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

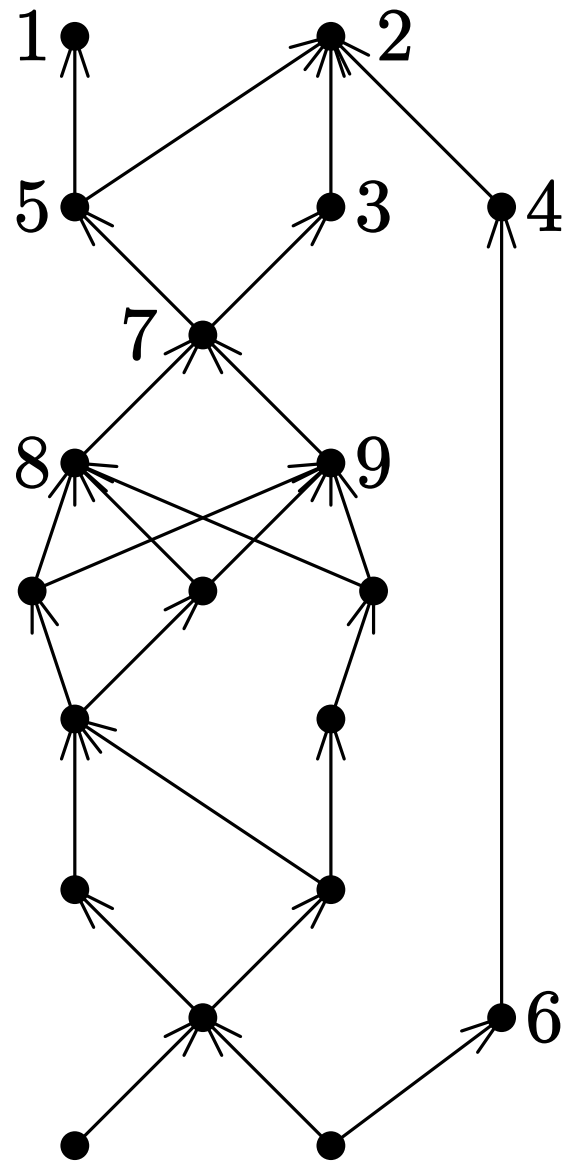


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

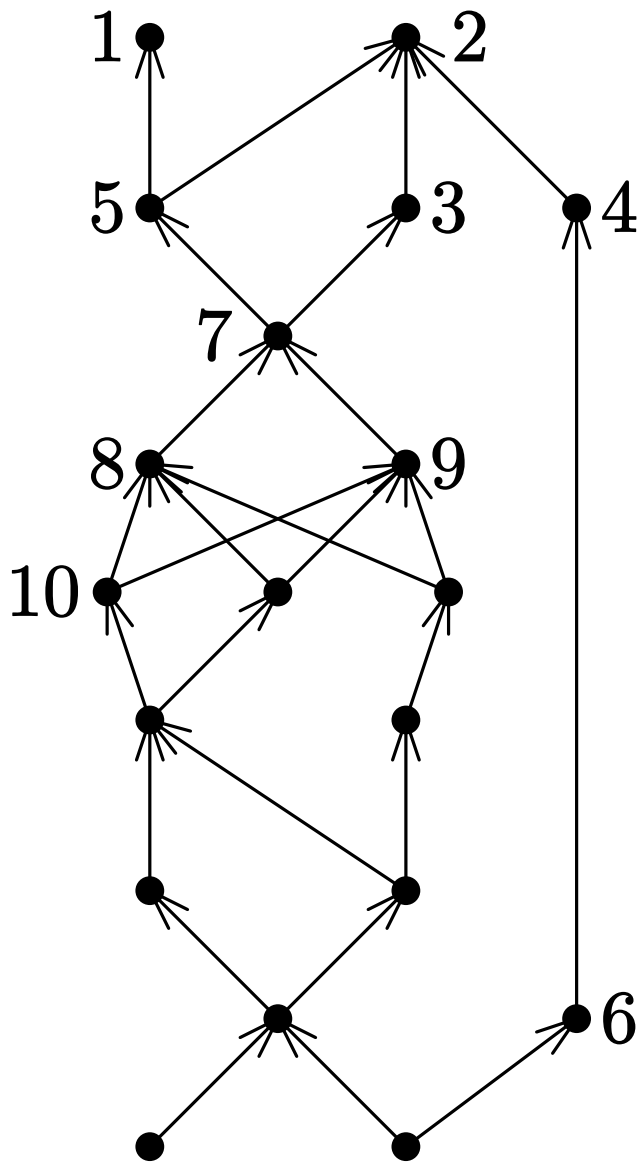


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

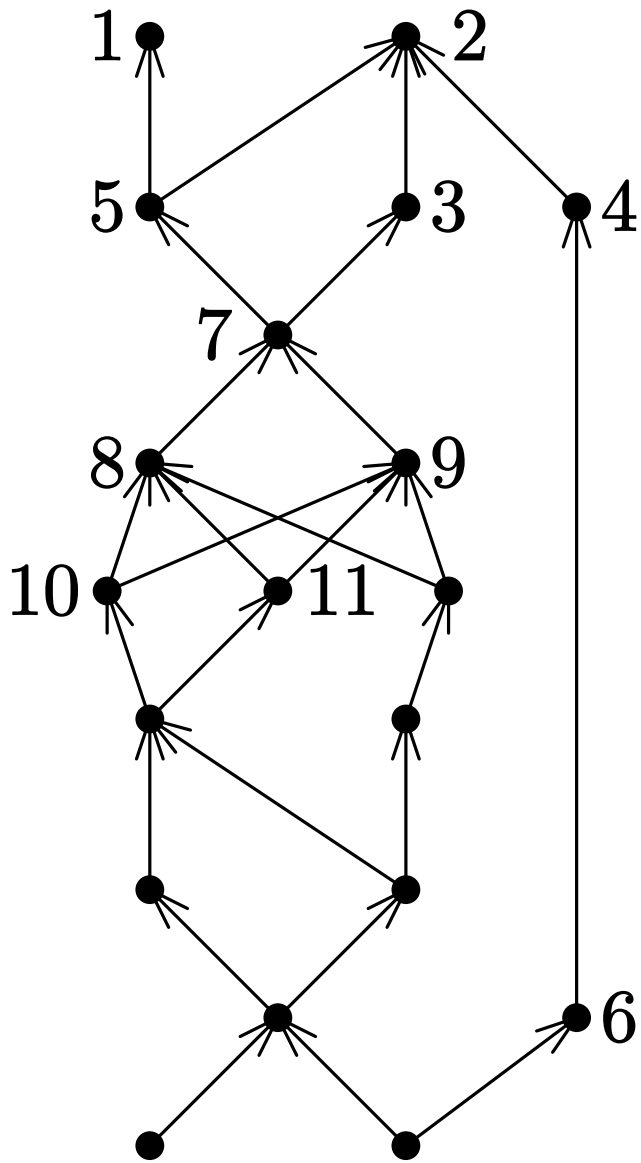


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

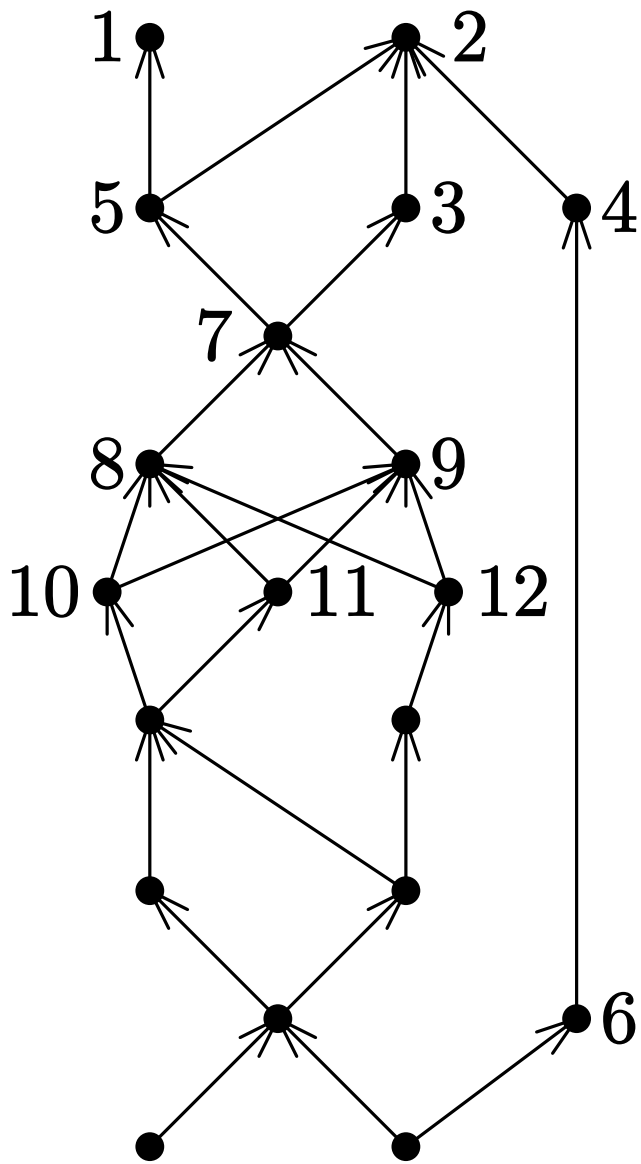


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

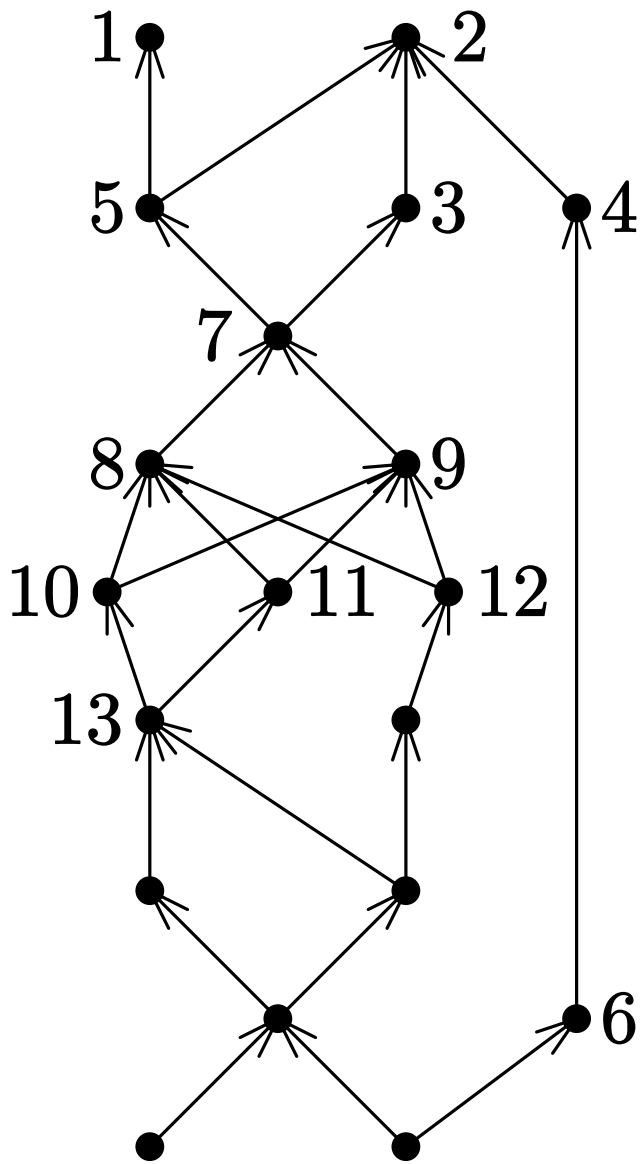


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

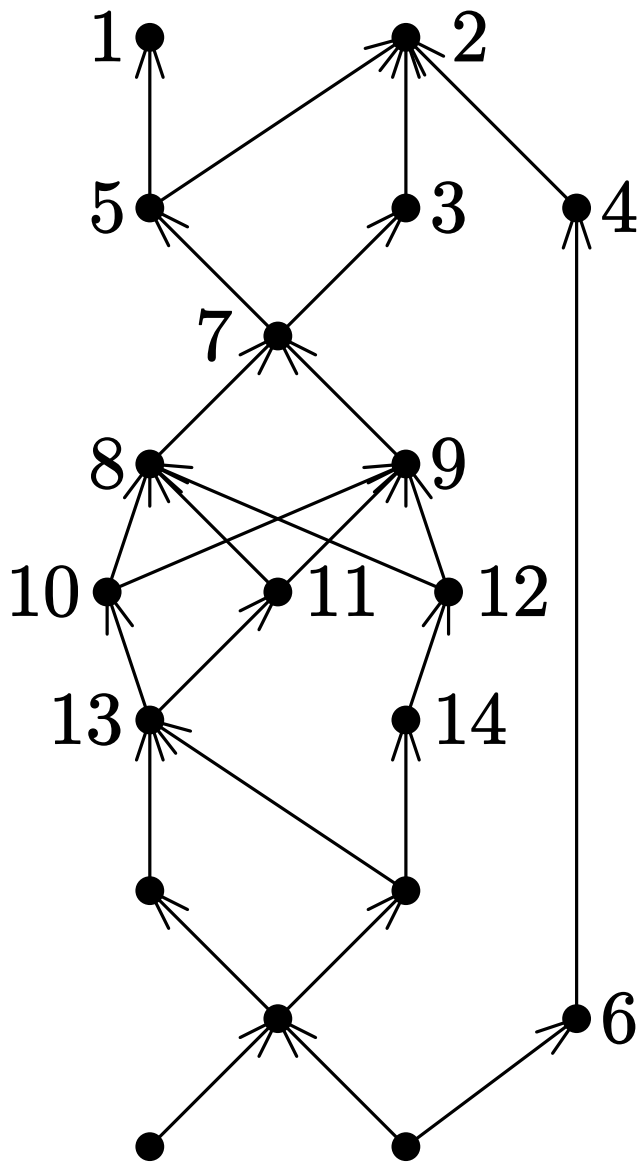


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

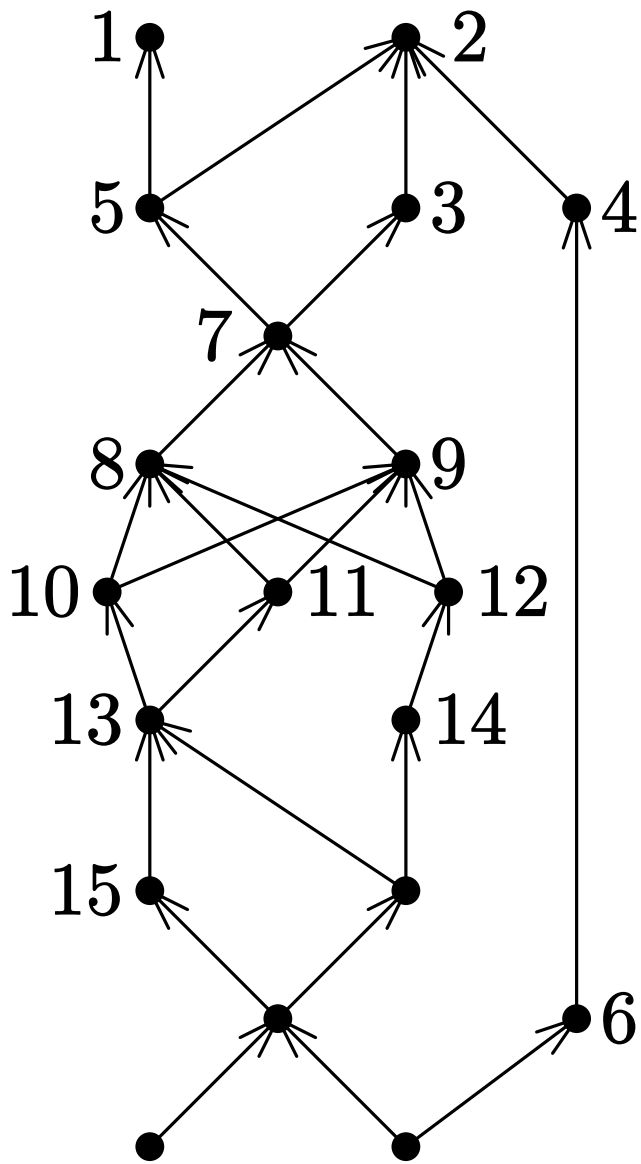


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

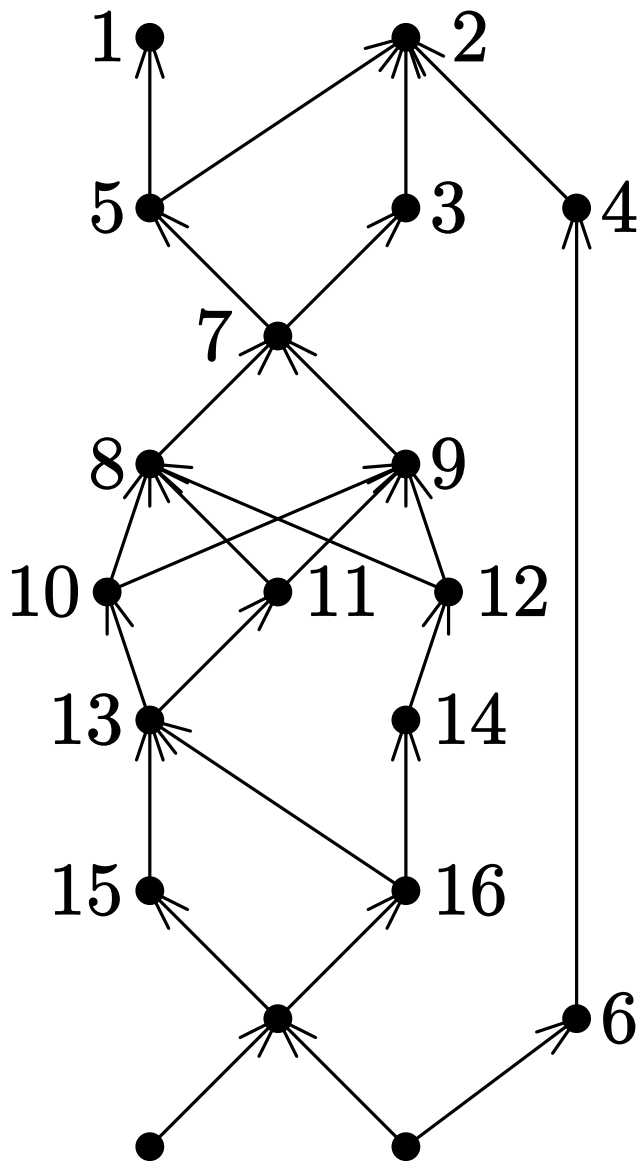


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

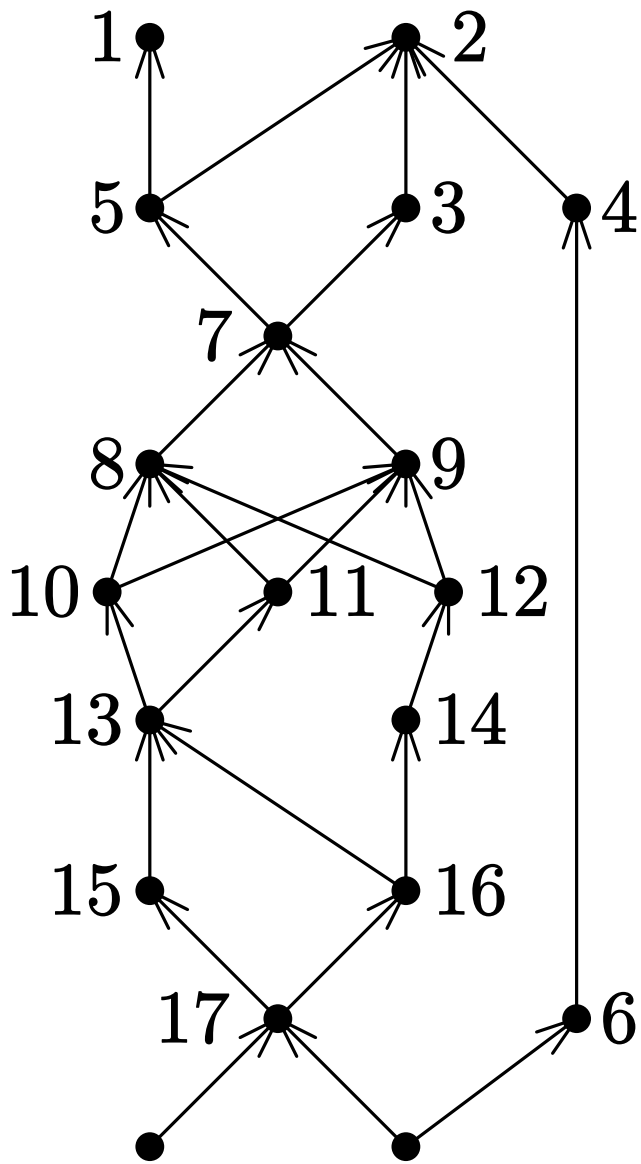


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

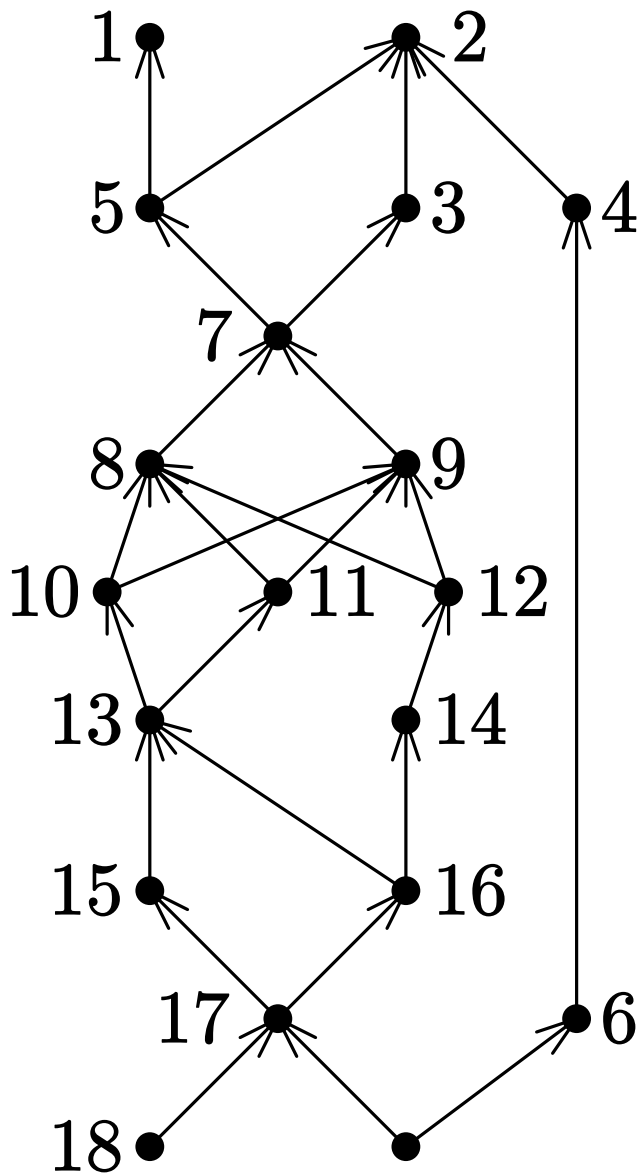


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

大きい方から順に並べたもの

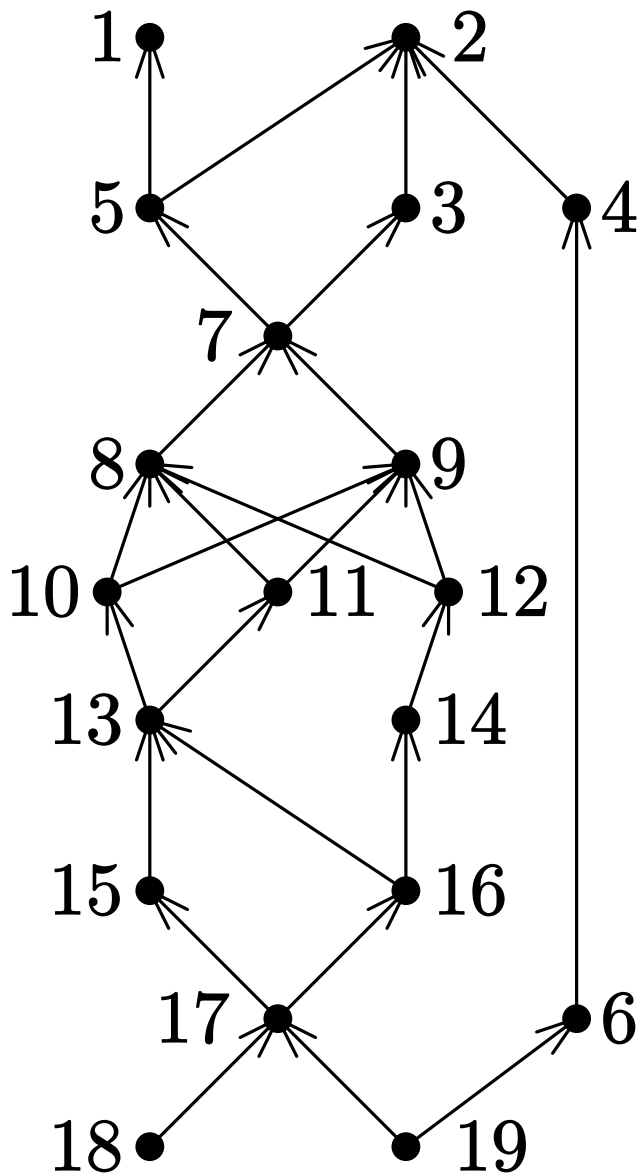


# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる

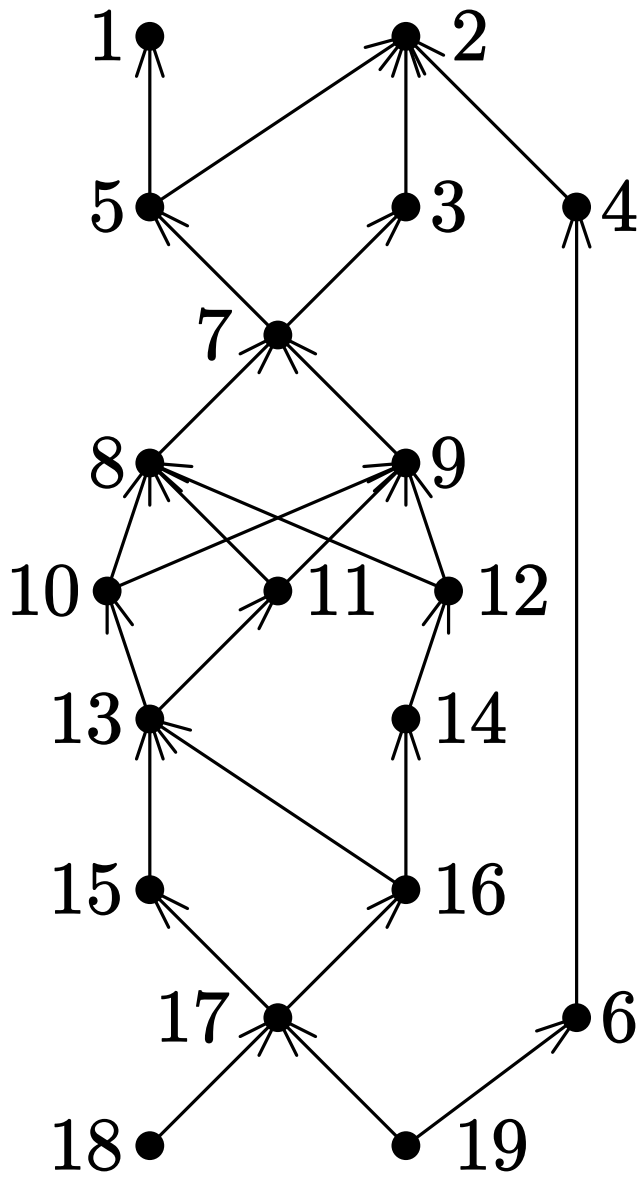
大きい方から順に並べたもの



# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる
- 得られたトポロジカル順序に対してリスト・スケジューリングを適用する

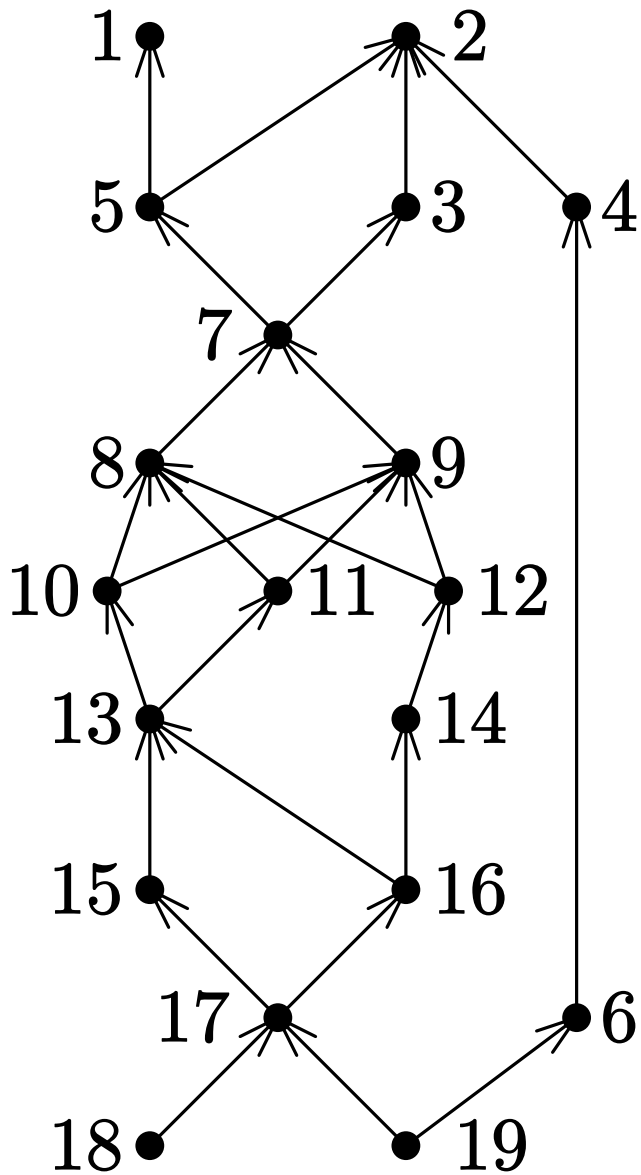



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 19 | 17 | 16 | 14 | 12 | 10 | 9 | 7 | 5 | 2 |
| 18 | 6 | 15 | 13 | 11 | 4 | 8 | | 3 | 1 |

# Coffman–Graham



- トポロジカル順序を後ろから作る
- 後ろから順に $1, 2, \ldots$ と数字を割り当てる
- 出次数が $0$ のジョブの中で出る辺の向かう頂点の数字列が辞書式最小のものに 次の数字を割り当てる
- 得られたトポロジカル順序に対してリスト・スケジューリングを適用する



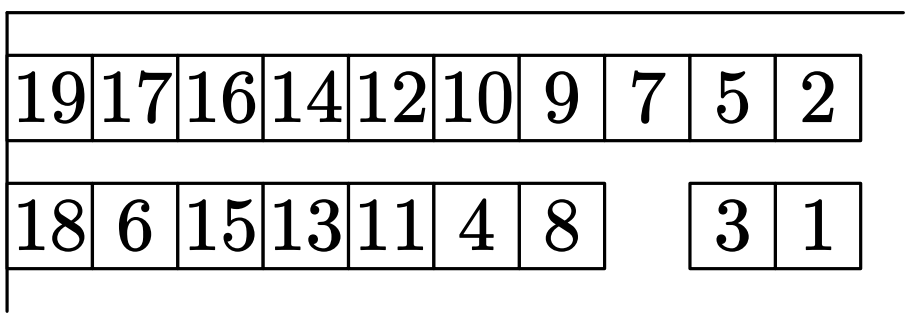

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 19 | 17 | 16 | 14 | 12 | 10 | 9 | 7 | 5 | 2 |
| 18 | 6 | 15 | 13 | 11 | 4 | 8 | | 3 | 1 |

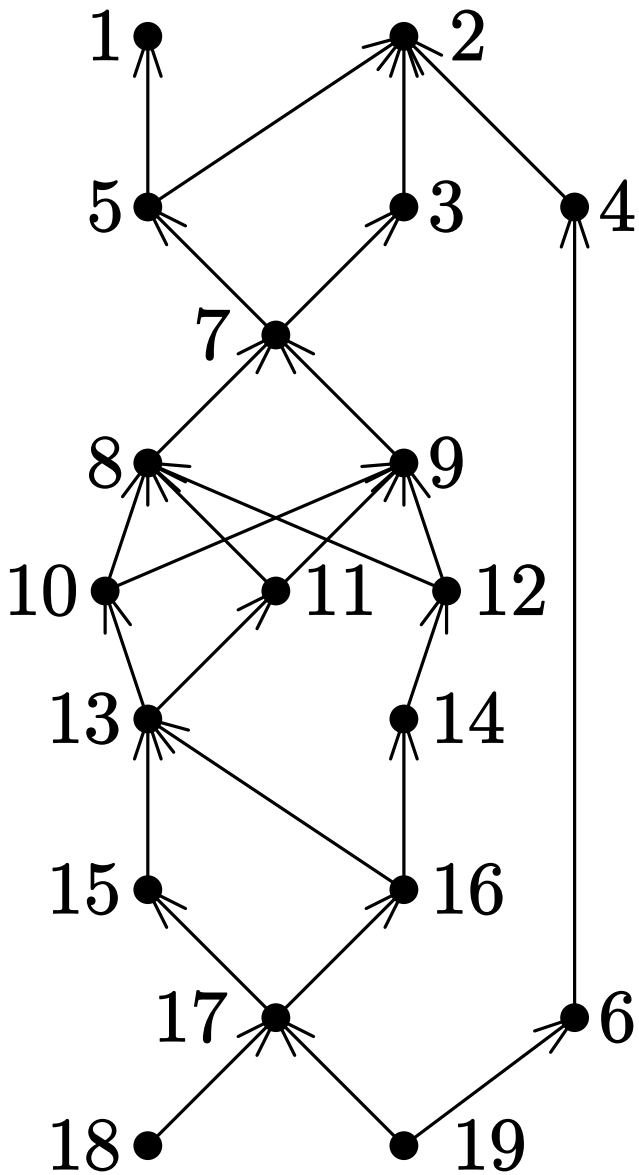
示したいこと：これで最適スケジュールが得られること

# Coffman–Graham：証明 (1)



- WLOG リスト・スケジューリングで $M_1$ が処理するジョブから決めると仮定
- $M_1$ が処理するジョブの数字は単調減少 ($\because\ p_j = 1$)

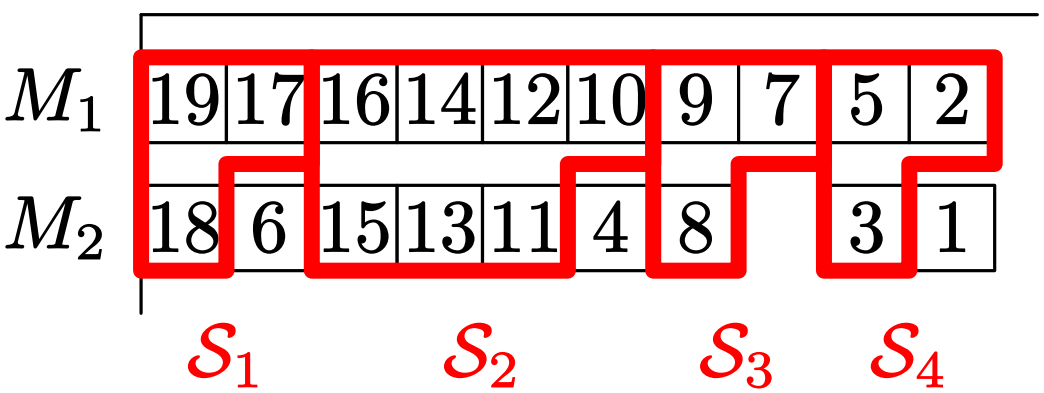


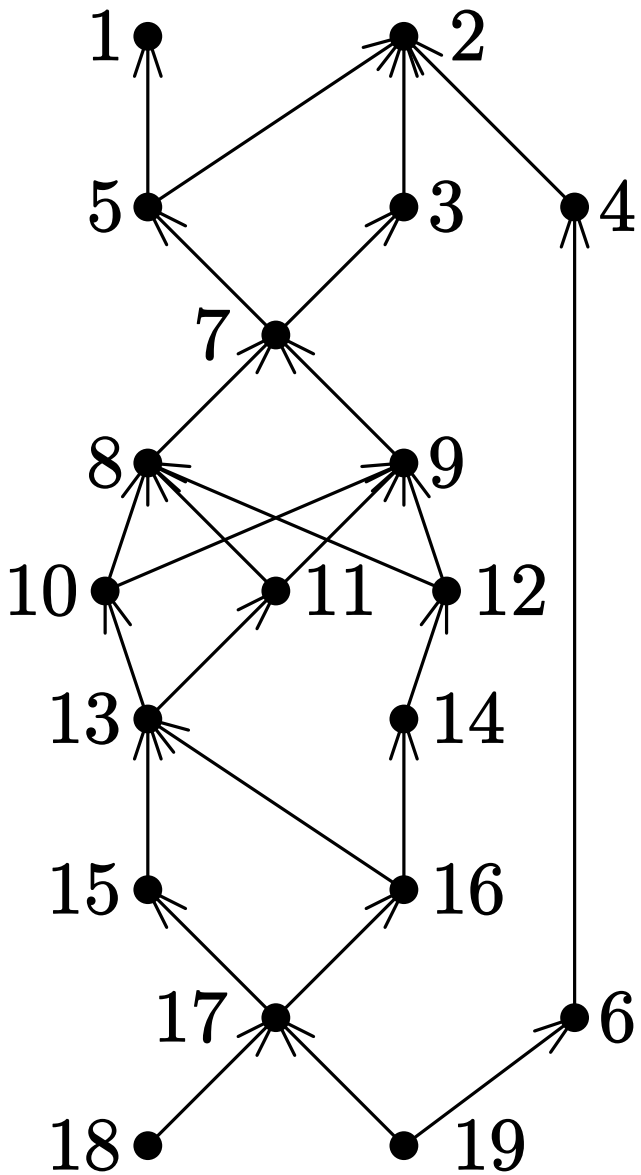



# Coffman–Graham：証明 (1)



- WLOG リスト・スケジューリングで $M_1$ が処理するジョブから決めると仮定
- $M_1$ が処理するジョブの数字は単調減少 ($\because\ p_j = 1$)
- $M_2$ で単調性が崩れるところに着目
- 単調性が崩れる直前までを塊にして，順に $\mathcal{S}_1, \ldots, \mathcal{S}_t$ とする (最後の形は例外)



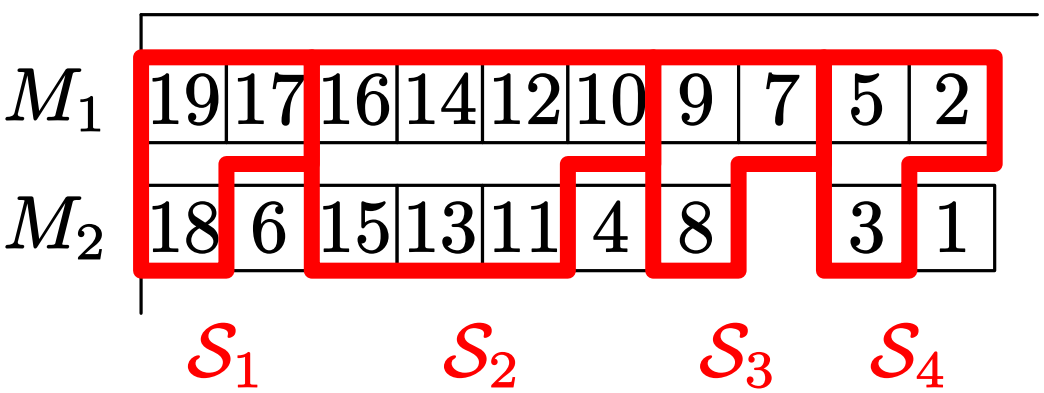
# Coffman–Graham：証明 (2)

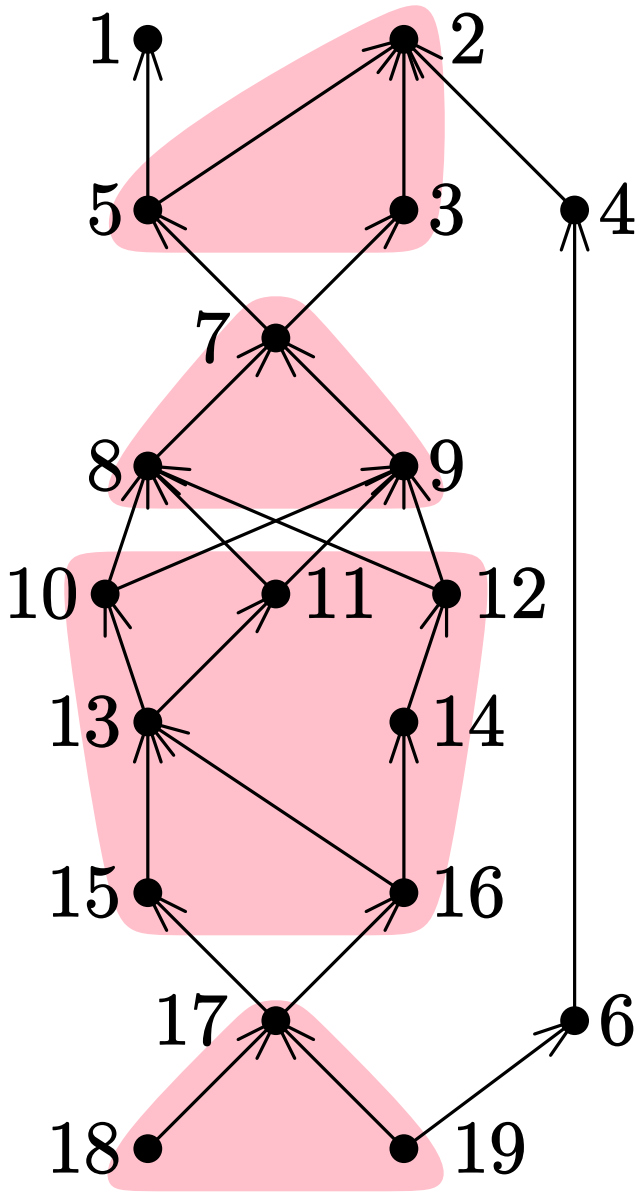


**補題**

$J \in \mathcal{S}_j$，$J' \in \mathcal{S}_{j+1} \Rightarrow$
$J$ の処理を完了しないと，
$J'$ の処理を開始できない

補題から，アルゴリズムの正当性が直ちに導ける



# Coffman–Graham：証明 (3)



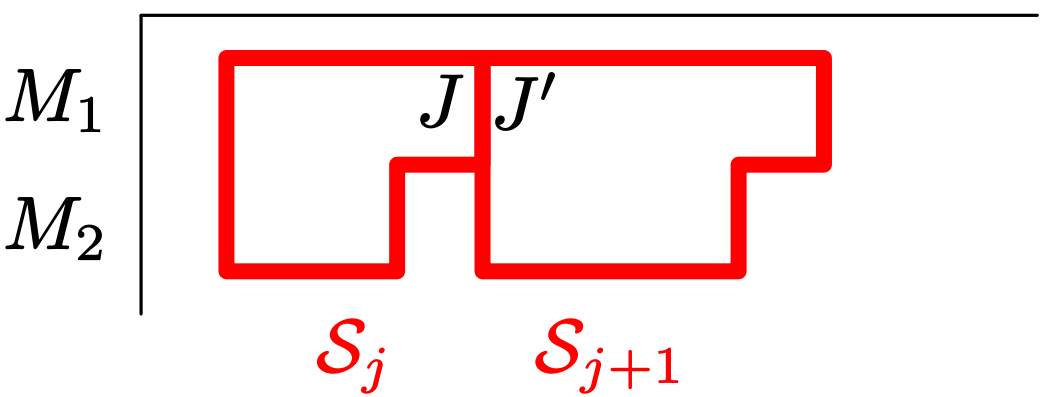
**補題**

$J \in \mathcal{S}_j$，$J' \in \mathcal{S}_{j+1} \Rightarrow$
$J$ の処理を完了しないと，$J'$ の処理を開始できない

補題の証明：ジョブ $I$ に対して

- $n(I) =$ アルゴリズムが $I$ に割り当てた数字
- $t(I) =$ アルゴリズムが $I$ の処理を開始する時刻

証明は数学的帰納法で行う



# Coffman–Graham：証明 (3)



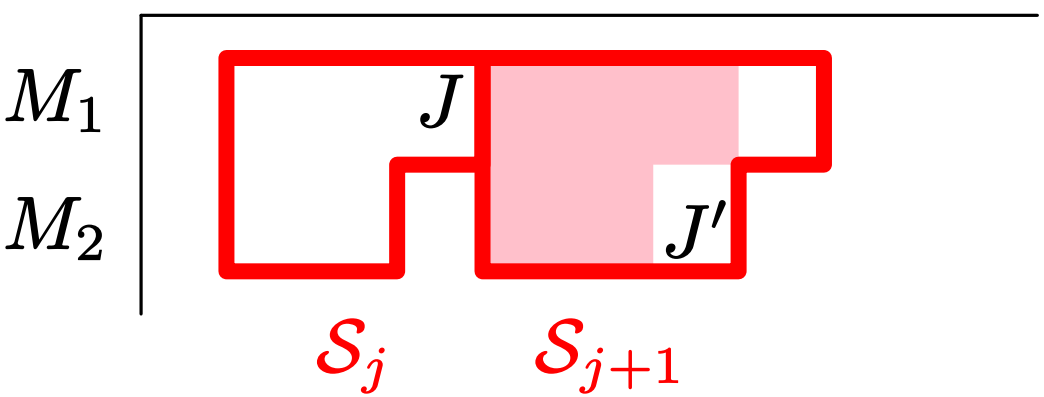
**補題**

$J \in \mathcal{S}_j$，$J' \in \mathcal{S}_{j+1} \Rightarrow$
$J$ の処理を完了しないと，$J'$ の処理を開始できない

補題の証明：ジョブ $I$ に対して

- $n(I) =$ アルゴリズムが $I$ に割り当てた数字
- $t(I) =$ アルゴリズムが $I$ の処理を開始する時刻

証明は数学的帰納法で行う



# Coffman–Graham：証明 (3)



**補題**

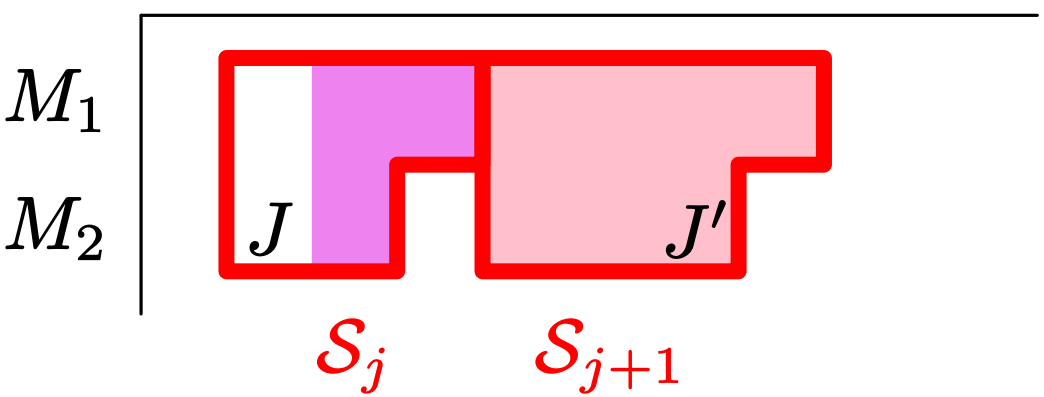
$J \in \mathcal{S}_j$，$J' \in \mathcal{S}_{j+1} \Rightarrow$
$J$ の処理を完了しないと，$J'$ の処理を開始できない

補題の証明：ジョブ $I$ に対して

- $n(I) =$ アルゴリズムが $I$ に割り当てた数字
- $t(I) =$ アルゴリズムが $I$ の処理を開始する時刻

証明は数学的帰納法で行う

$J \to^* J'$ と書くことにする

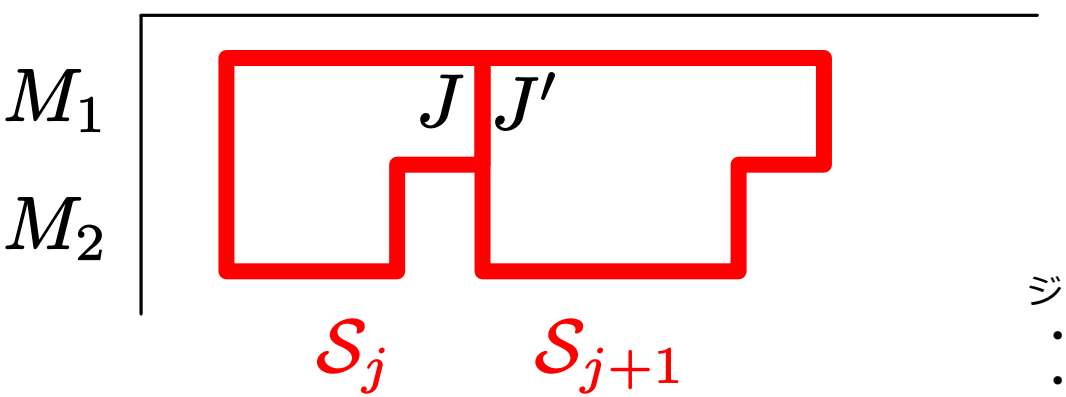


# Coffman–Graham：証明 (4)



$J$ が $\mathcal{S}_j$ の右上，$J'$ が $\mathcal{S}_{j+1}$ の左上のとき

- $n(J) > n(J')$
- あるジョブ $J''$ に対して，$J'' \to J'$ かつ $t(J) \leq t(J'')$
  ($\because J'$ が $J$ と同時に処理されなかった)



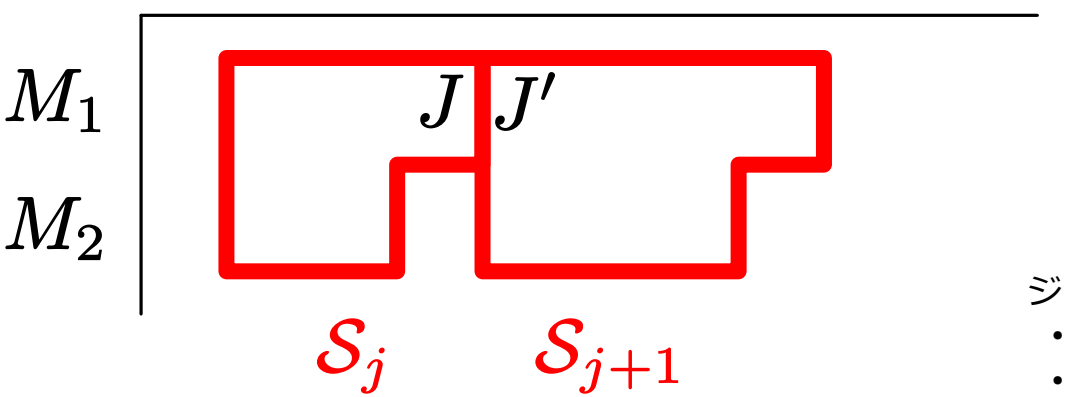

ジョブ $I$ に対して
- $n(I)$ = アルゴリズムが $I$ に割り当てた数字
- $t(I)$ = アルゴリズムが $I$ の処理を開始する時刻

# Coffman–Graham：証明 (4)



$J$ が $\mathcal{S}_j$ の右上，$J'$ が $\mathcal{S}_{j+1}$ の左上のとき

- $n(J) > n(J')$
- あるジョブ $J''$ に対して，$J'' \to J'$ かつ $t(J) \leq t(J'')$
  ($\because J'$ が $J$ と同時に処理されなかった)
- $t(J'') < t(J')$ ($\because J'' \to J'$)
- $n(J'') > n(J')$ ($\because$ 数字はトポロジカル順に従う)



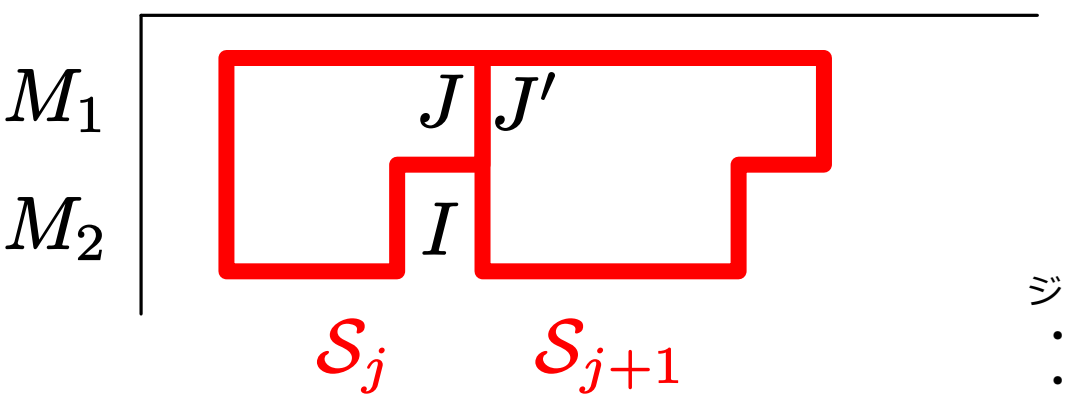

ジョブ $I$ に対して
- $n(I)$ = アルゴリズムが $I$ に割り当てた数字
- $t(I)$ = アルゴリズムが $I$ の処理を開始する時刻

# Coffman–Graham：証明 (4)



$J$ が $\mathcal{S}_j$ の右上，$J'$ が $\mathcal{S}_{j+1}$ の左上のとき

- $n(J) > n(J')$
- あるジョブ $J''$ に対して，$J'' \to J'$ かつ $t(J) \leq t(J'')$
  ($\because J'$ が $J$ と同時に処理されなかった)
- $t(J'') < t(J')$ ($\because J'' \to J'$)
- $n(J'') > n(J')$ ($\because$ 数字はトポロジカル順に従う)
- $\mathcal{S}_j$ の右下にジョブ $I$ があるとすると，$n(I) < n(J')$
- $\therefore J'' = J$ であり，$J \to J'$ となる



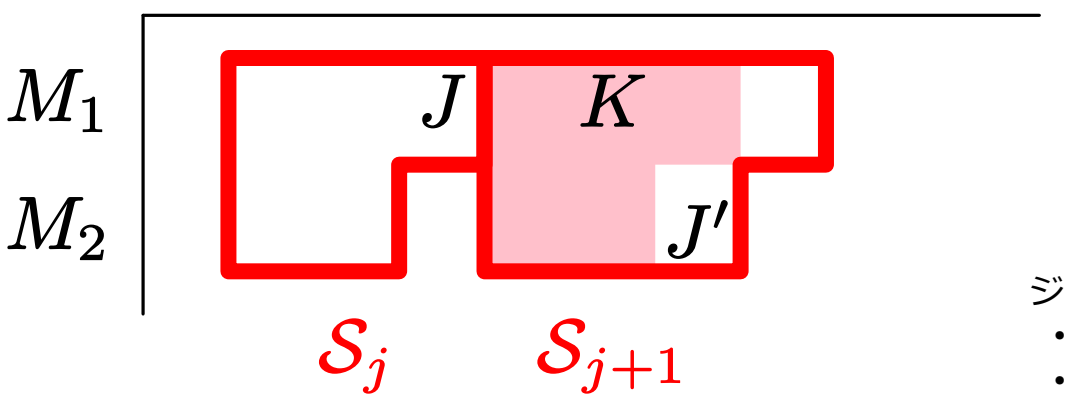

ジョブ $I$ に対して
- $n(I)$ = アルゴリズムが $I$ に割り当てた数字
- $t(I)$ = アルゴリズムが $I$ の処理を開始する時刻

# Coffman–Graham：証明 (5)



$J$ が $\mathcal{S}_j$ の右上，$K$ が $\mathcal{S}_{j+1}$ のピンクにあるときは
$J \to^* K$ が成り立つと仮定して，次の $J' \in \mathcal{S}_{j+1}$ を考える

- あるジョブ $J''$ に対して，$J'' \to J'$ かつ $t(J) \leq t(J'')$
  ($\because J'$ が $M_2$ で処理されなかった)
- $t(J'') < t(J')$ ($\because J'' \to J'$)



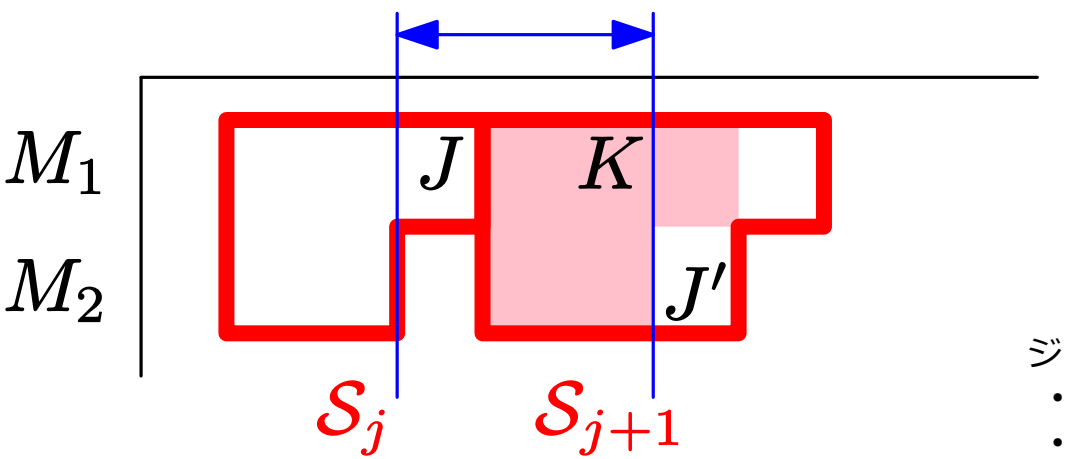

ジョブ $I$ に対して
- $n(I)$ = アルゴリズムが $I$ に割り当てた数字
- $t(I)$ = アルゴリズムが $I$ の処理を開始する時刻

# Coffman–Graham：証明 (5)



$J$ が $\mathcal{S}_j$ の右上，$K$ が $\mathcal{S}_{j+1}$ のピンクにあるときは
$J \to^* K$ が成り立つと仮定して，次の $J' \in \mathcal{S}_{j+1}$ を考える

- あるジョブ $J''$ に対して，$J'' \to J'$ かつ $t(J) \leq t(J'')$
  ($\because J'$ が $M_2$ で処理されなかった)
- $t(J'') < t(J')$ ($\because J'' \to J'$)



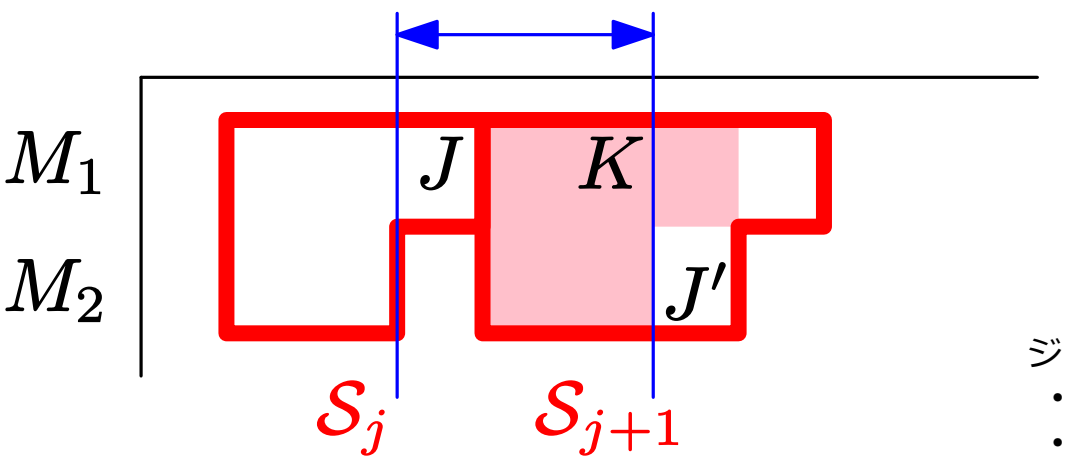

ジョブ $I$ に対して
- $n(I)$ = アルゴリズムが $I$ に割り当てた数字
- $t(I)$ = アルゴリズムが $I$ の処理を開始する時刻

# Coffman–Graham：証明 (5)



$J$ が $\mathcal{S}_j$ の右上，$K$ が $\mathcal{S}_{j+1}$ のピンクにあるときは
$J \to^* K$ が成り立つと仮定して，次の $J' \in \mathcal{S}_{j+1}$ を考える

- あるジョブ $J''$ に対して，$J'' \to J'$ かつ $t(J) \leq t(J'')$
  ($\because J'$ が $M_2$ で処理されなかった)
- $t(J'') < t(J')$ ($\because J'' \to J'$)
- $\therefore J'' = J$ であるか，$J''$ はピンクにある
- どちらであっても，$J \to^* J'$ が成り立つ




ジョブ $I$ に対して
- $n(I)$ = アルゴリズムが $I$ に割り当てた数字
- $t(I)$ = アルゴリズムが $I$ の処理を開始する時刻

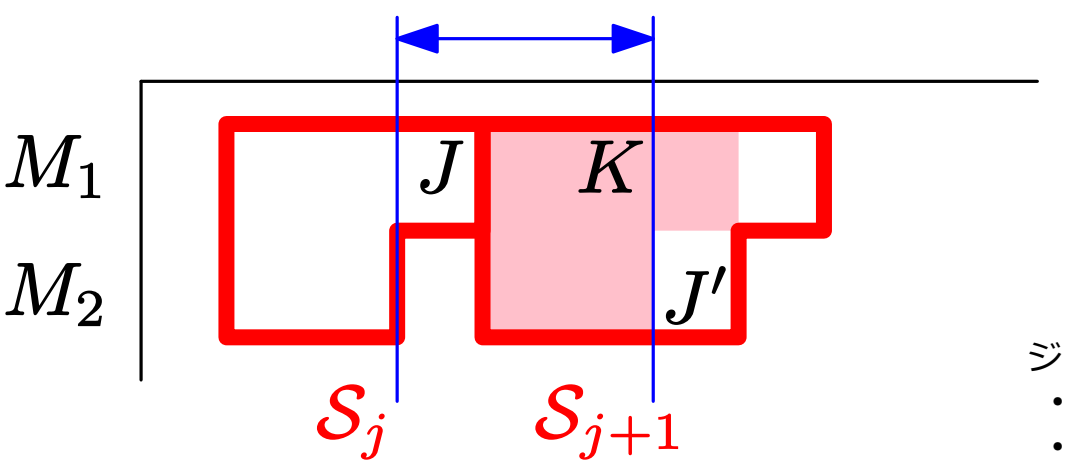
# Coffman–Graham：証明 (5)



$J$ が $\mathcal{S}_j$ の右上，$K$ が $\mathcal{S}_{j+1}$ のピンクにあるときは
$J \to^* K$ が成り立つと仮定して，次の $J' \in \mathcal{S}_{j+1}$ を考える

- あるジョブ $J''$ に対して，$J'' \to J'$ かつ $t(J) \leq t(J'')$
  ($\because J'$ が $M_2$ で処理されなかった)
- $t(J'') < t(J')$ ($\because J'' \to J'$)
- $\therefore J'' = J$ であるか，$J''$ はピンクにある
- どちらであっても，$J \to^* J'$ が成り立つ

したがって，$J$ が $\mathcal{S}_j$ の右上のとき，任意の $J' \in \mathcal{S}_{j+1}$ に対して，
$J \to^* J'$ が成り立つ



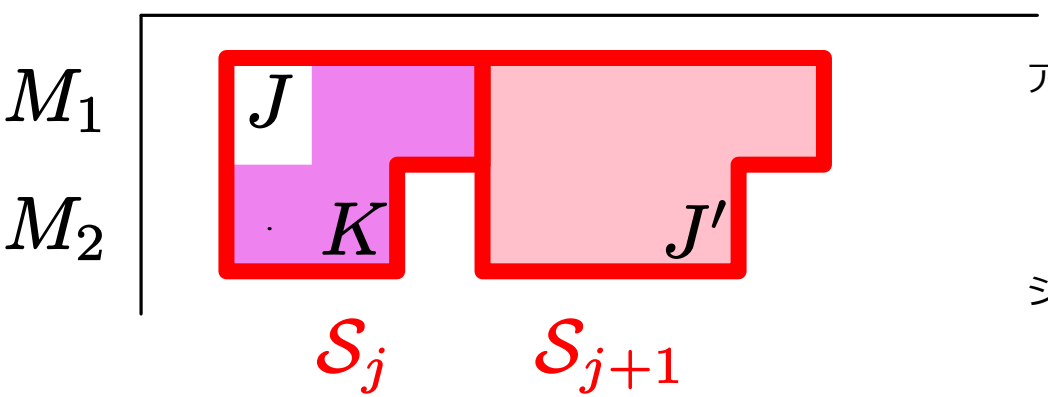

ジョブ $I$ に対して
- $n(I)$ = アルゴリズムが $I$ に割り当てた数字
- $t(I)$ = アルゴリズムが $I$ の処理を開始する時刻

# Coffman–Graham：証明 (6)



$K$ が $S_j$ の紫，$J'$ が $S_{j+1}$ にあるときは $K \to^* J'$ が成り立つと仮定して，紫の１つ前の $J \in S_j$ を考える

- $S_j$ の紫にある，ある $K$ に対して，$J \to^* K$ ならば，$J \to^* J'$ となり，証明終了
- $\therefore$ そのような $K$ はないとする




アルゴリズムでは

- 出次数が 0 のジョブの中で
  出る辺の向かう頂点の数字列が辞書式最小のものに
  次の数字を割り当てる

ジョブ $I$ に対して

- $n(I)$ = アルゴリズムが $I$ に割り当てた数字
- $t(I)$ = アルゴリズムが $I$ の処理を開始する時刻

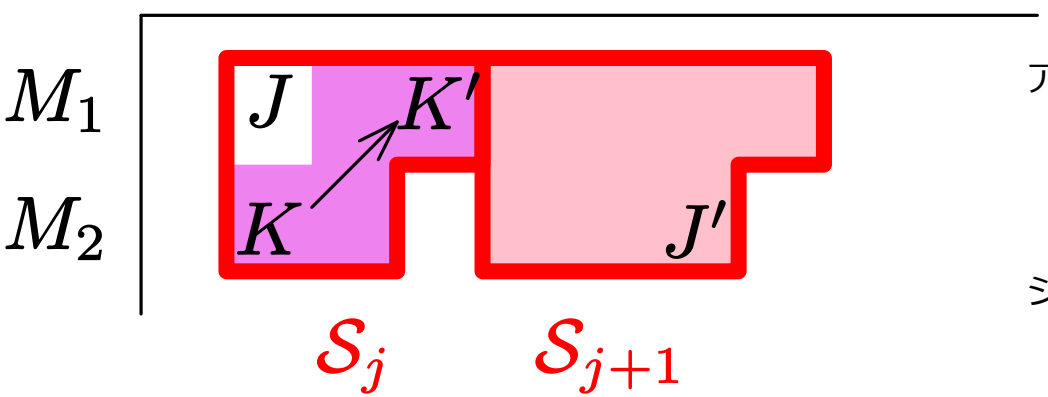
# Coffman–Graham：証明 (7)



$K$ が $\mathcal{S}_j$ の紫，$J'$ が $\mathcal{S}_{j+1}$ にあるときは $K \to^* J'$ が成り立つと仮定して，紫の１つ前の $J \in \mathcal{S}_j$ を考える

$\mathcal{S}_j$ の紫から，任意にジョブを選んで $K$ とする

- $n(J) > n(K)$
- $\mathcal{S}_j$ の紫のジョブ $K'$ に対して，$K \to K'$ とすると，
  $n(J) < n(K)$ となり矛盾　($n(K)$ を決める瞬間を考える)




アルゴリズムでは

- 出次数が 0 のジョブの中で
  出る辺の向かう頂点の数字列が辞書式最小のものに
  次の数字を割り当てる

ジョブ $I$ に対して

- $n(I)$ = アルゴリズムが $I$ に割り当てた数字
- $t(I)$ = アルゴリズムが $I$ の処理を開始する時刻

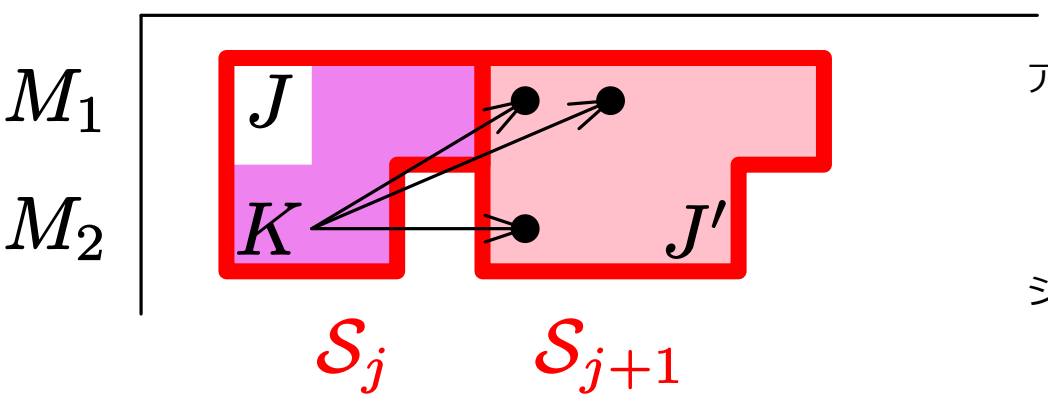
# Coffman–Graham：証明 (7)



$K$ が $\mathcal{S}_j$ の紫，$J'$ が $\mathcal{S}_{j+1}$ にあるときは $K \to^* J'$ が成り立つと仮定して，紫の１つ前の $J \in \mathcal{S}_j$ を考える

$\mathcal{S}_j$ の紫から，任意にジョブを選んで $K$ とする

- $n(J) > n(K)$
- $\mathcal{S}_j$ の紫のジョブ $K'$ に対して，$K \to K'$ とすると，
  $n(J) < n(K)$ となり矛盾　($n(K)$ を決める瞬間を考える)
- $K \to J''$ となる $J'' \in \mathcal{S}_{j+1}$ に対して，
  $J \to J''$ が成り立つ　(なぜ？)




アルゴリズムでは
- 出次数が 0 のジョブの中で
  出る辺の向かう頂点の数字列が辞書式最小のものに
  次の数字を割り当てる

ジョブ $I$ に対して
- $n(I)$ = アルゴリズムが $I$ に割り当てた数字
- $t(I)$ = アルゴリズムが $I$ の処理を開始する時刻

# Coffman–Graham：証明 (7)

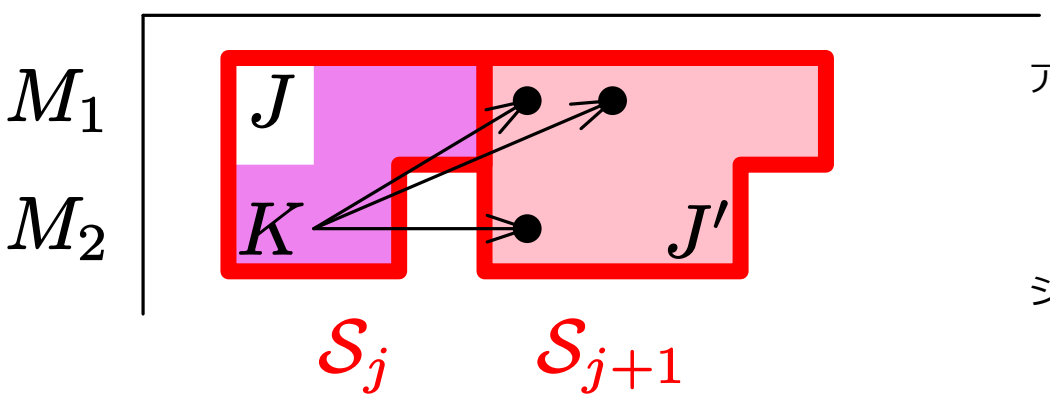


$K$ が $\mathcal{S}_j$ の紫，$J'$ が $\mathcal{S}_{j+1}$ にあるときは $K \to^* J'$ が成り立つと仮定して，紫の１つ前の $J \in \mathcal{S}_j$ を考える

$\mathcal{S}_j$ の紫から，任意にジョブを選んで $K$ とする

- $n(J) > n(K)$
- $\mathcal{S}_j$ の紫のジョブ $K'$ に対して，$K \to K'$ とすると，
  $n(J) < n(K)$ となり矛盾　($n(K)$ を決める瞬間を考える)
- $K \to J''$ となる $J'' \in \mathcal{S}_{j+1}$ に対して，
  $J \to J''$ が成り立つ　(なぜ？)

$\therefore$ 任意の $J' \in \mathcal{S}_{j+1}$ に対して，$J \to^* J'$ □




アルゴリズムでは
- 出次数が 0 のジョブの中で
  出る辺の向かう頂点の数字列が辞書式最小のものに
  次の数字を割り当てる

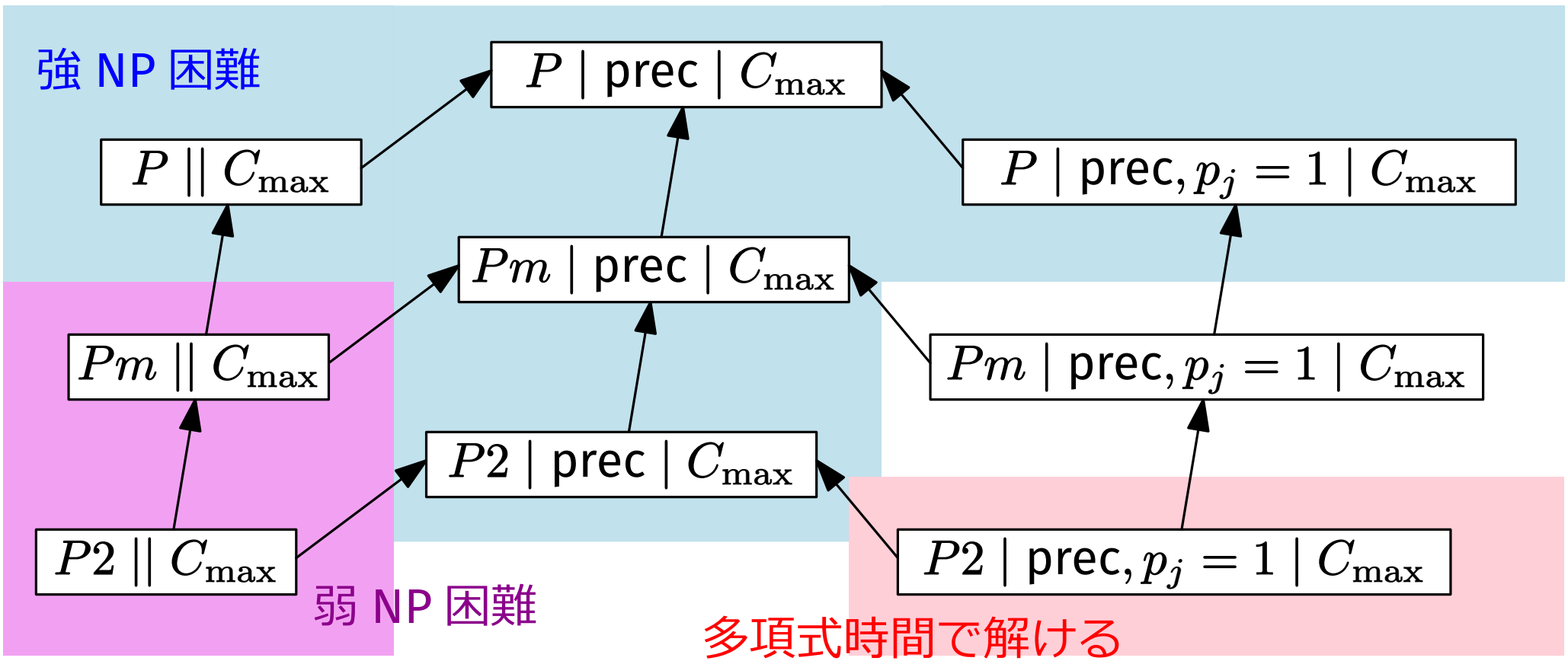
ジョブ $I$ に対して
- $n(I)$ = アルゴリズムが $I$ に割り当てた数字
- $t(I)$ = アルゴリズムが $I$ の処理を開始する時刻

# まとめ：$Pm \mid \mathrm{prec} \mid C_{\max}$ の難しさ



## 定理 (今日の内容)

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



# 次回の予告



## 次回の予告

特殊な構造を持つ先行制約の取り扱い方