離散最適化基礎論 (2025 年後学期)

高速指数時間アルゴリズム

第１回

高速指数時間アルゴリズムの考え方

岡本 吉央　(電気通信大学)

okamotoy@uec.ac.jp

2025 年 10 月 7 日

最終更新：2025 年 10 月 8 日　16:57

講義概要

概要

離散最適化における１つのトピックを集中的に学ぶ

今年度のトピック

高速指数時間アルゴリズム

主に，2000 年代までの **基礎的内容** を扱う

実践ではなく，**理論 (数理)** を扱う

本講義の目標

次の 2 つができるようになること

目標 1

基本的なアルゴリズム設計技法 を使って
基本的な離散最適化問題 を解けるようになる

目標 2

問題の数学的構造がアルゴリズム設計指針を与える
という重要な考え方を実践する

メッセージ：数学的な考え方からアルゴリズムが得られる

解きやすい問題と解きにくい問題

多くの組合せ最適化問題は次のどちらかに分類される

解きやすい問題

- 最小全域木問題
- 最大マッチング問題
- 最大流問題
- …

多項式時間アルゴリズムで解ける

解きにくい問題

- 巡回セールスマン問題
- 最小 Steiner 木問題
- ナップサック問題
- …

NP 困難である

解きやすい問題と解きにくい問題

多くの組合せ最適化問題は次のどちらかに分類される

解きやすい問題

- 最小全域木問題
- 最大マッチング問題
- 最大流問題
- …

多項式時間アルゴリズムで解ける

解きにくい問題

- 巡回セールスマン問題
- 最小 Steiner 木問題
- ナップサック問題
- …

NP 困難である

問 1：解きにくい問題 を解くにはどうすればよいか？

問 2：解きにくい問題の中をさらに分類できるか？

スケジュール (前半)

1. 高速指数時間アルゴリズムの考え方 (10/7)
＊ 休み (体育祭) (10/14)
2. 分枝アルゴリズム：基礎 (10/21)
3. 分枝アルゴリズム：高速化 (10/28)
4. 分枝アルゴリズム：測度統治法 (11/4)
5. 動的計画法：基礎 (11/11)
6. 動的計画法：例 (11/18)

スケジュール (後半)

7. 包除原理：原理 (11/25)
* 休み (秋ターム試験) (12/2)
8. 包除原理：例 (12/9)
9. 部分集合たたみ込み：原理 (12/16)
* 休み (出張) (12/23)
* 休み (冬季休業) (12/30)
10. 部分集合たたみ込み：例 (1/6)
11. 指数時間仮説：原理 (1/13)
12. 指数時間仮説：証明 (1/20)
13. 最近の話題 (1/27)
* 休み (修士論文発表会) (2/3)

授業の進め方：基本情報

教員

岡本吉央 (おかもと よしお)
- 居室：西 4-206
- E-mail：okamotoy@uec.ac.jp

講義 Web ページ

http://dopal.cs.uec.ac.jp/okamotoy/lect/2025/exptime/
- 講義スライド
- コメント回答
- レポート出題
- その他

Google Classroom

内部シラバスのコードを見て，各自が登録

授業の進め方：各回の進行 (予定)

10:40	授業開始
11:20 頃	休憩 (3 分程度)
	授業再開
	授業終了
12:10	コメント投稿
13:00	コメント受付終了

授業の進め方：各回の進行 (予定)

10:40	授業開始
11:20 頃	休憩 (3 分程度)
	授業再開
	授業終了
12:10	コメント投稿
13:00	コメント受付終了

Google Classroom のフォームから投稿

- 質問・疑問
- 誤植の指摘
- 感想
- 要望
- 文句
- 雑談
- など何でも

授業の進め方：評価

評価方法

2 回のレポート提出 **のみ** による

- 1 回 50 点満点

成績

$$\text{素点} = \text{レポート 1 の得点} + \text{レポート 2 の得点}$$

授業の考え方：数学と証明

注意点 1

この授業では，数学を使い，数学的な証明もする

理由

- 授業の内容の正しさが **検証** できるべき
- 証明という **技法** を身につけるべき

$\rightsquigarrow$ この授業では **批判的思考**，**論理的思考** の体得 も目指す

授業の考え方：プログラミング

注意点 2

この授業では，プログラミングを行わない

理由

- 教員がまじめにプログラミングをしたことがない
  (∵ 教員が授業で扱えない)
- 受講生自身が自主的にプログラミングをすればよい

⇝ 授業の内容から **自由に はみ出ていく** ことを推奨する

前提知識

次の内容は既知とする

- **離散数学**
  - 特に，集合・写像の記法 と 数学的帰納法
- **基礎的なプログラミング，アルゴリズム**
  - 特に，再帰 と O 記法

身についていない場合は，自習をするように

参考書

次の文献を参考にして，講義を進める (入手の必要はない)

- F. V. Fomin, D. Kratsch, *Exact Exponential Algorithms*. Springer, 2010.

他に，最近の研究論文も参考にする

本日の内容

1. **高速指数時間アルゴリズムの例**
2. 高速指数時間アルゴリズムの目標

無向グラフの定義と用語

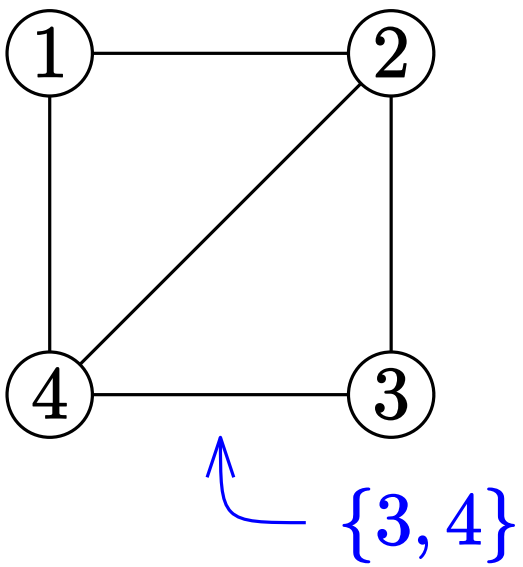
無向グラフ $G = (V, E)$ (undirected graph)

頂点集合 (vertex set)

辺集合 (edge set)

V の要素 = 頂点

E の要素 = 辺



$V = \{1, 2, 3, 4\}$

$E = \{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$

辺は $\{u, v\}$ を省略して uv と書くことも

無向グラフの定義と用語

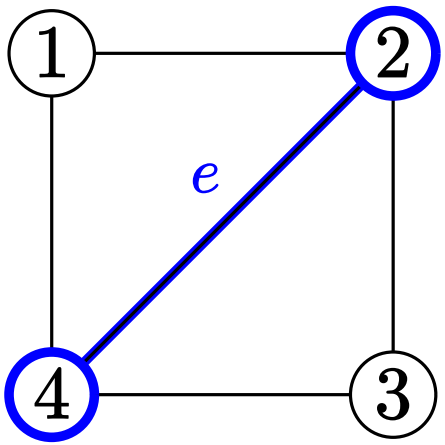
無向グラフ $G = (V, E)$ (undirected graph)

頂点集合 (vertex set)

辺集合 (edge set)

V の要素 = 頂点

E の要素 = 辺



$V = \{1, 2, 3, 4\}$

$E = \{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$

辺は $\{u, v\}$ を省略して uv と書くことも

辺 $e = \{u, v\}$ を考えると

2 頂点 u, v は **隣接** する

辺 e は 2 頂点 u, v に **接続** する

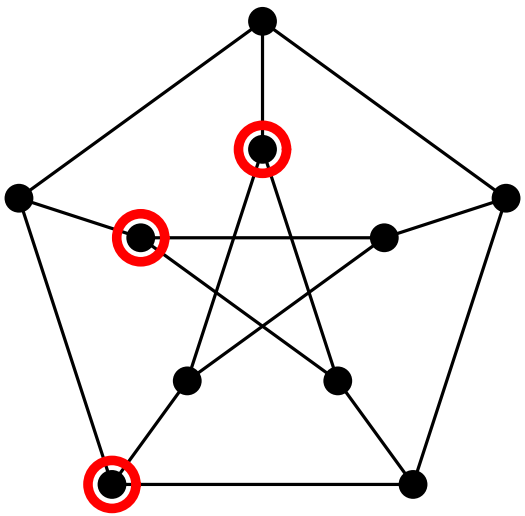
2 頂点 u, v は辺 e の **端点** である

独立集合

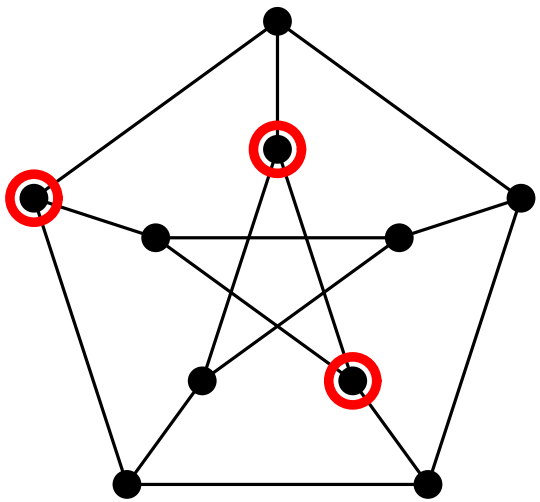
無向グラフ $G = (V, E)$

定義：独立集合 (independent set)

G の **独立集合** とは，頂点部分集合 $S \subseteq V$ で，
どの 2 頂点 $u, v \in S$ も隣接していないもの



独立集合である



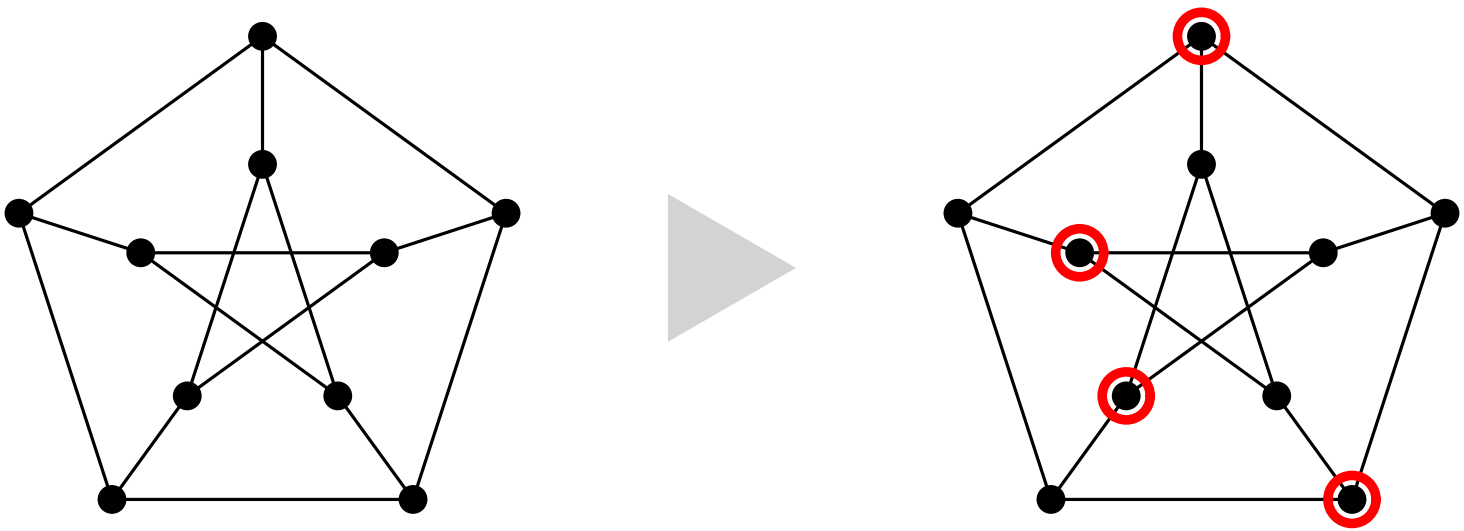
独立集合ではない

最大独立集合問題

問題：最大独立集合問題

入力： 無向グラフ $G = (V, E)$
出力： G の最大独立集合

要素数最大の独立集合



入力　　出力

注：最大独立集合問題は NP 困難 (Karp '72)

しらみつぶし？

(brute force)

すべての

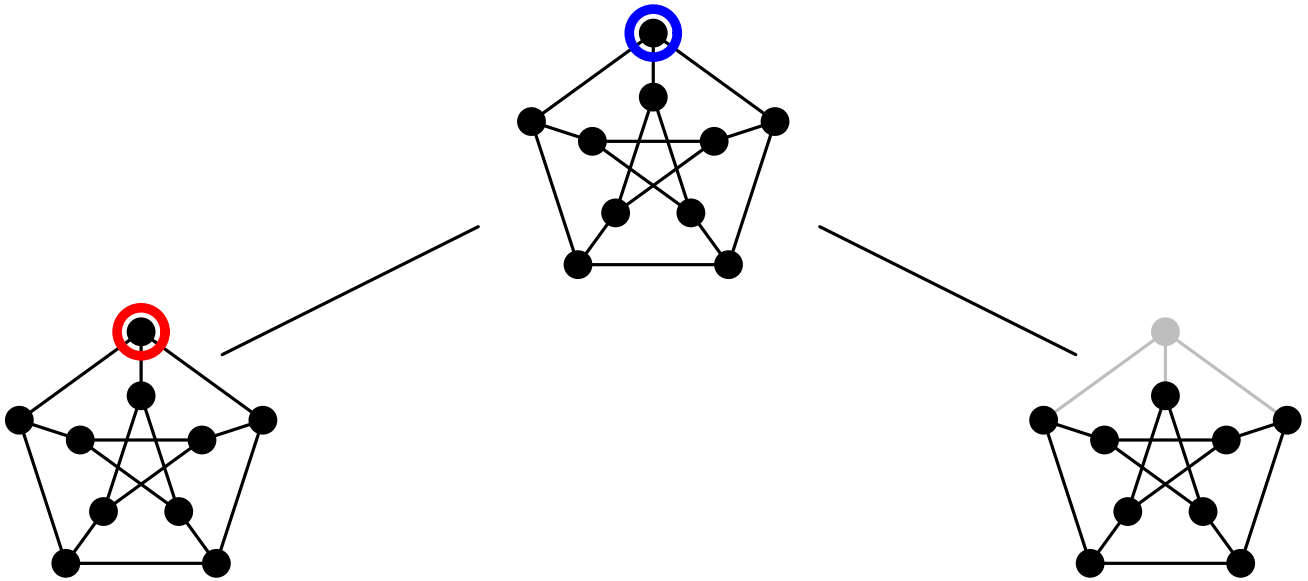
頂点部分集合の数

$= 2^{|V|}$

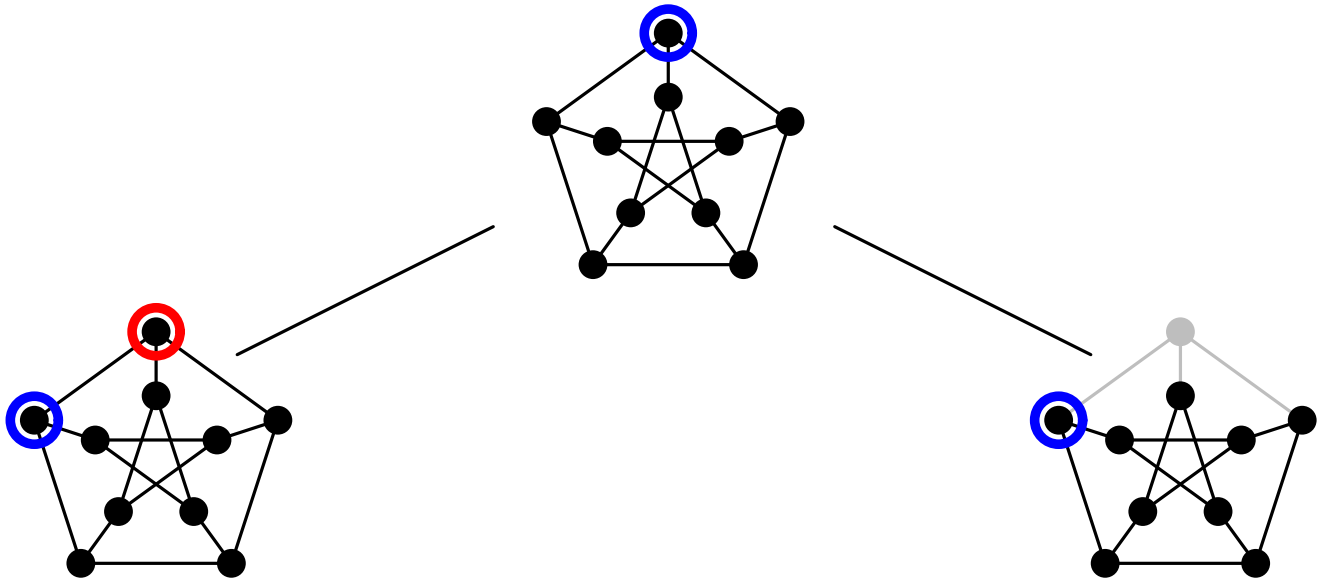
$|V| = 10$

$\rightsquigarrow 2^{|V|} = 1\,024$

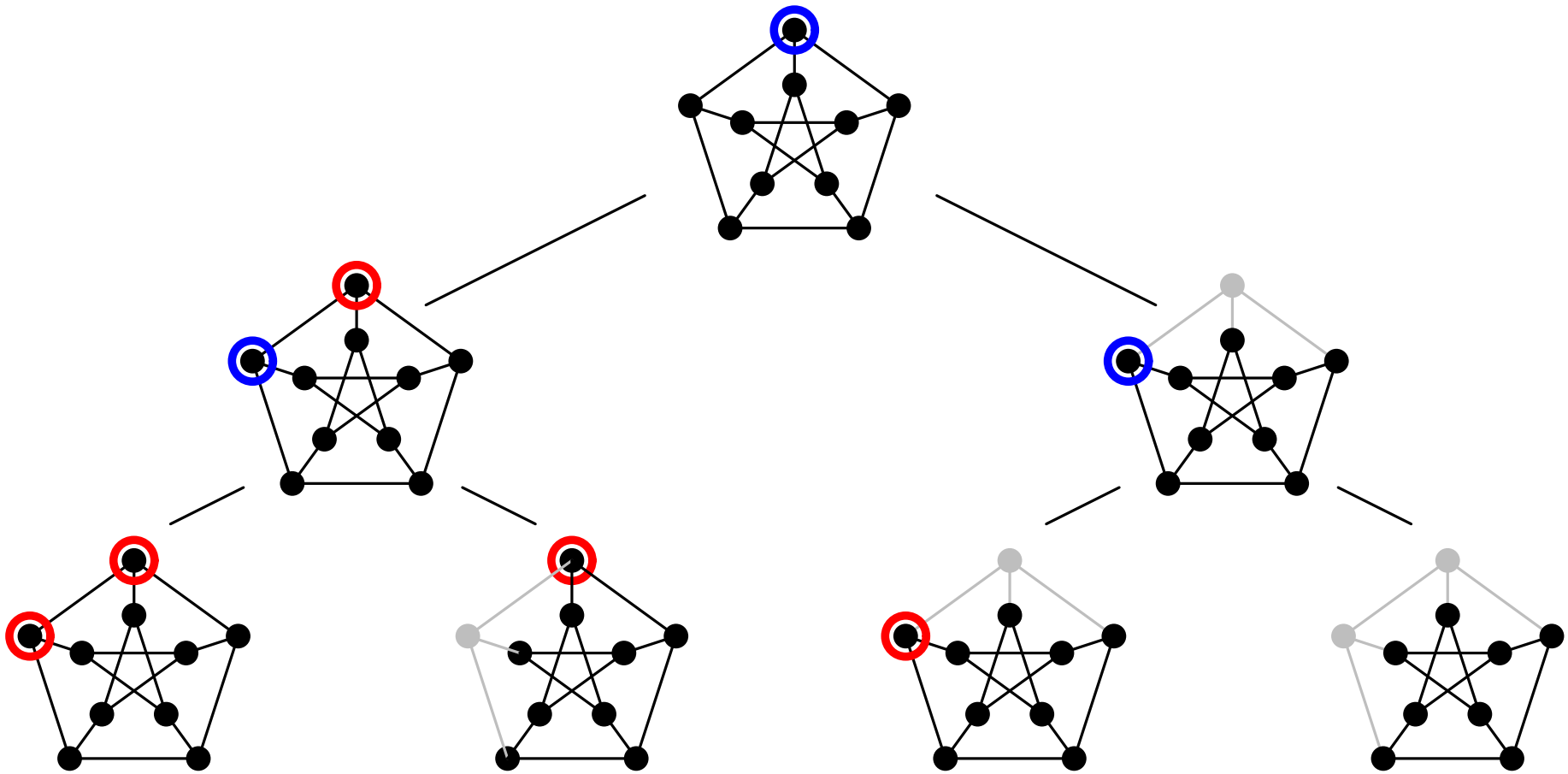
しらみつぶし？ → 木



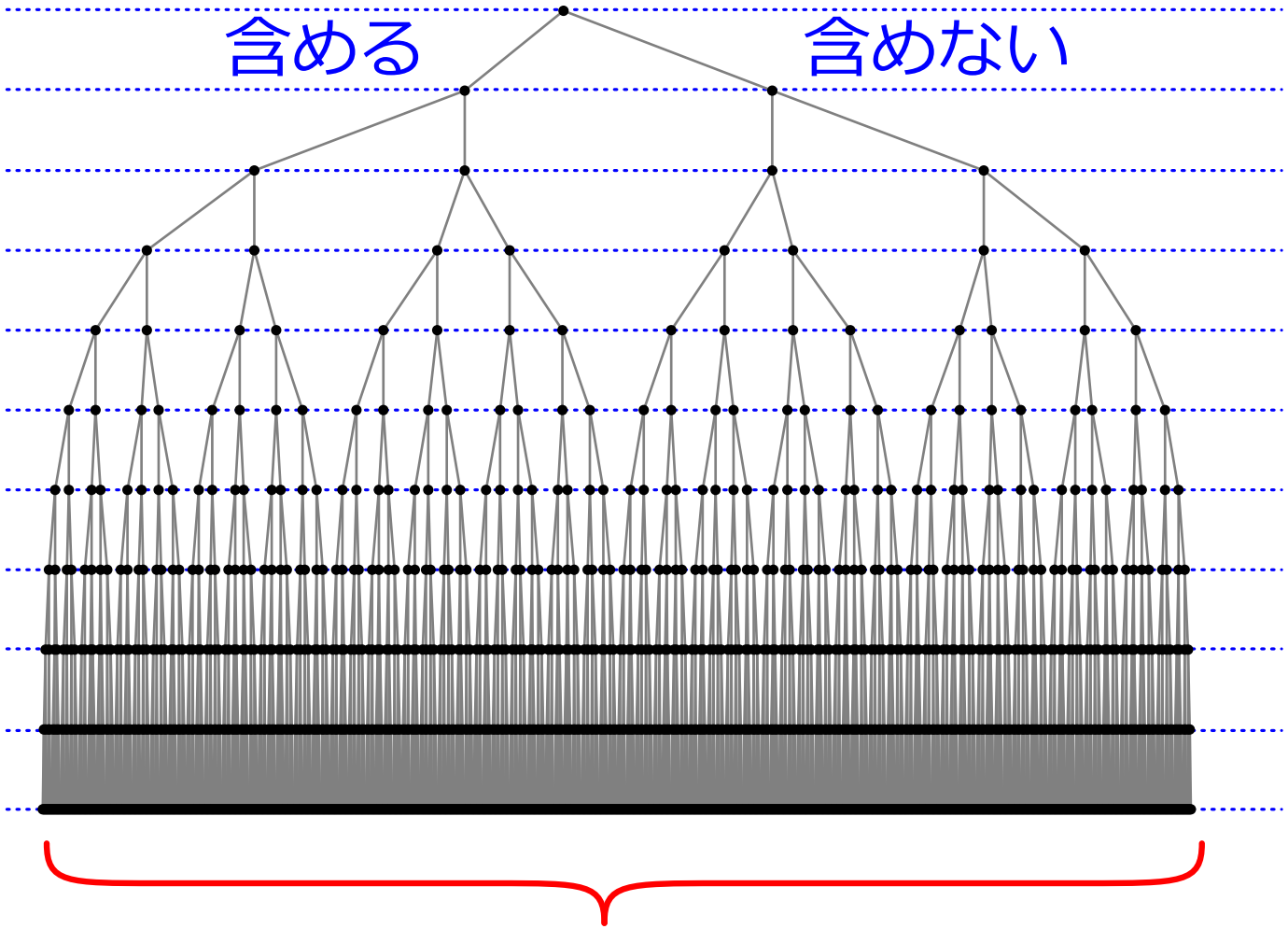
しらみつぶし？ → 木



しらみつぶし？ → 木



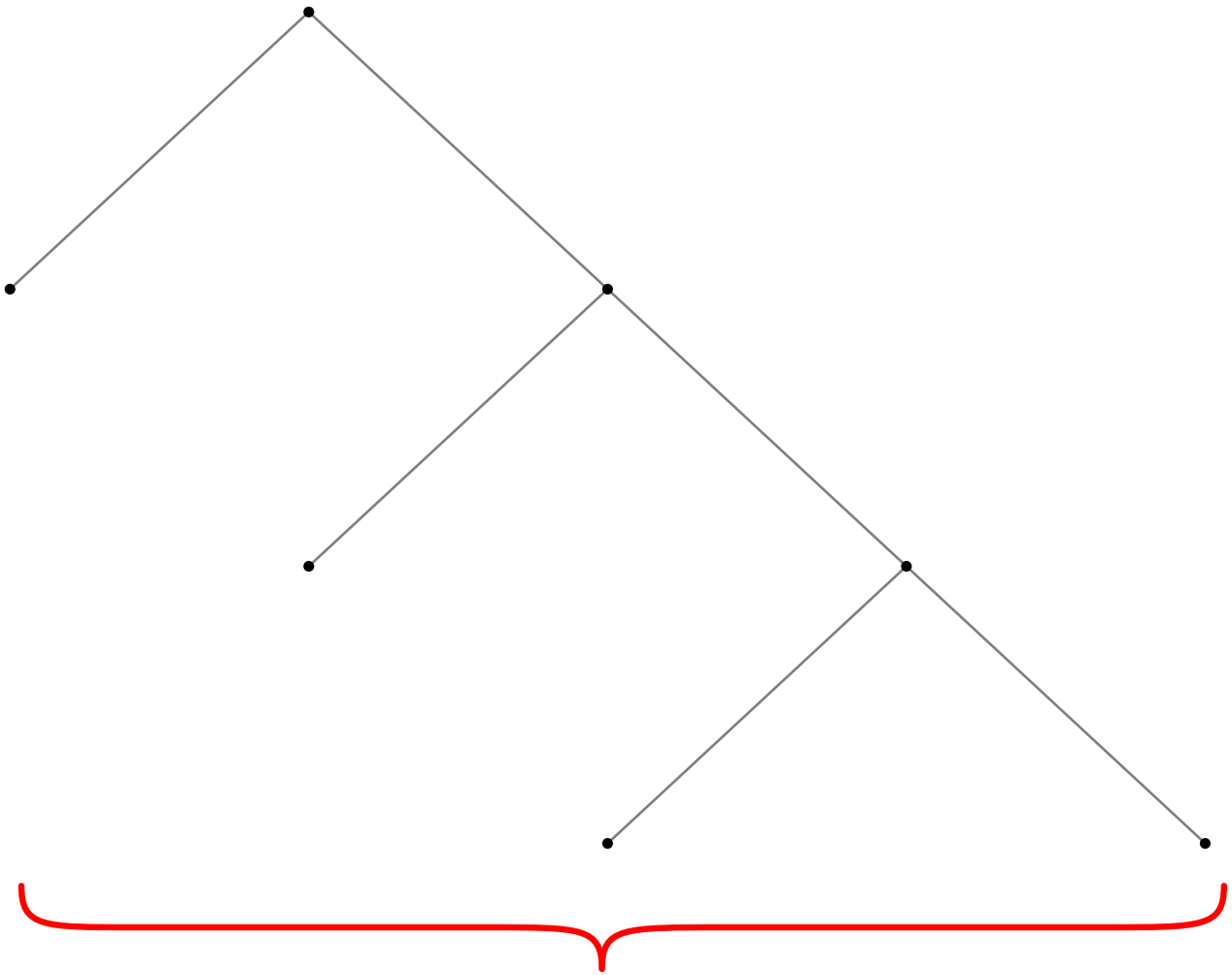
しらみつぶし？ → 木



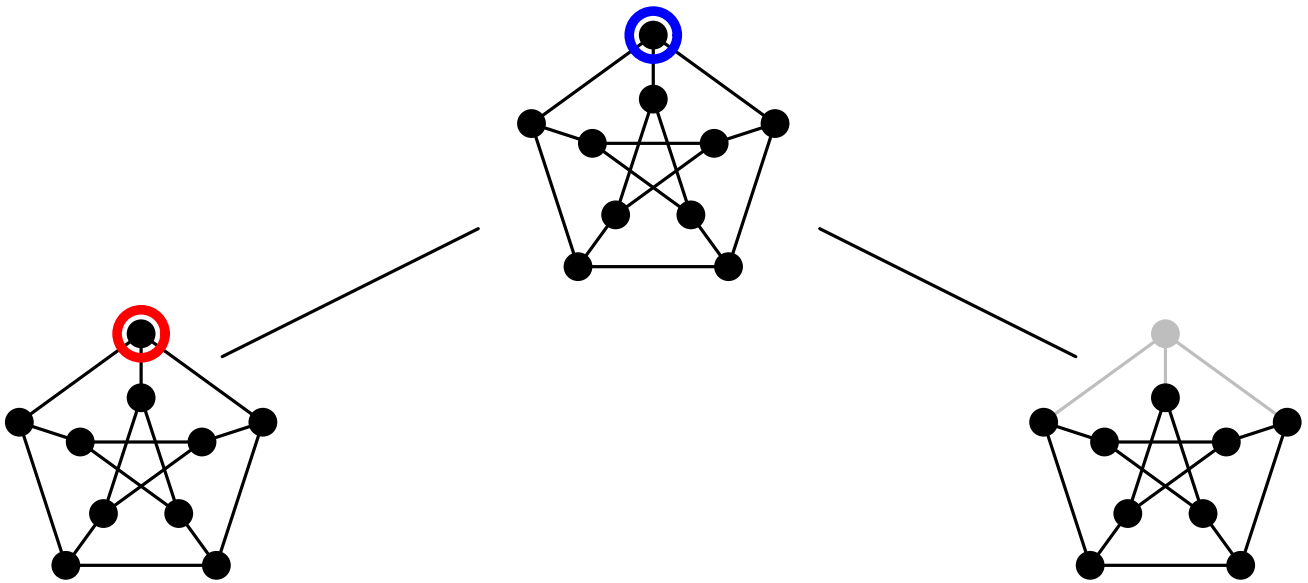
頂点 1
2
3
4
5
6
7
8
9
10

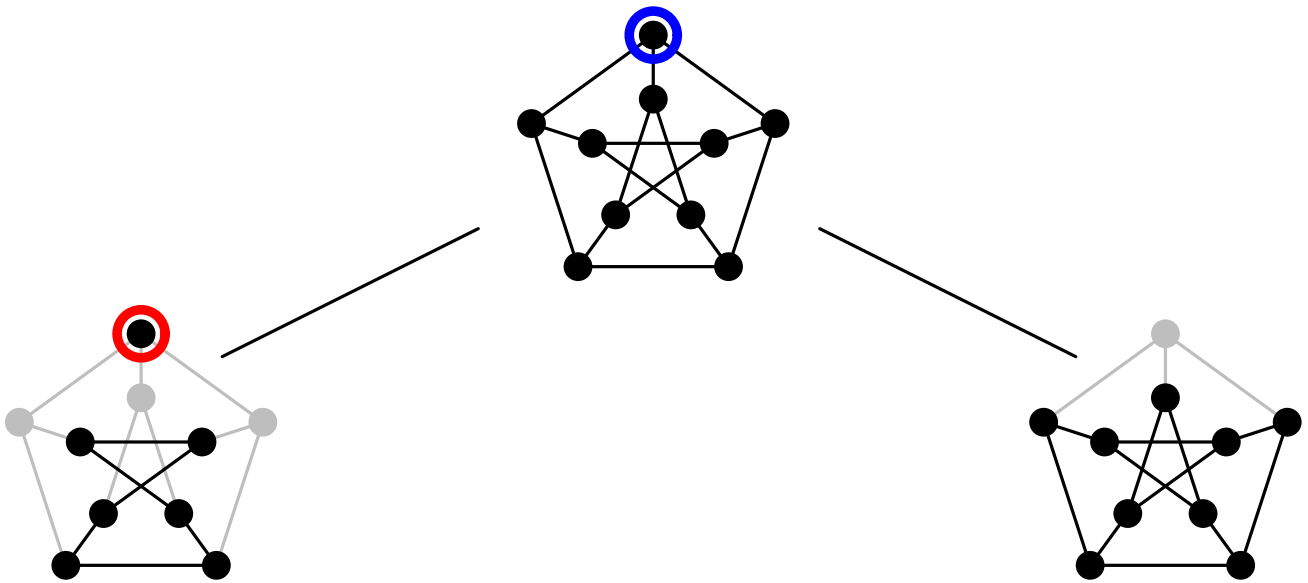
葉の数 = 1 024

工夫をすると…

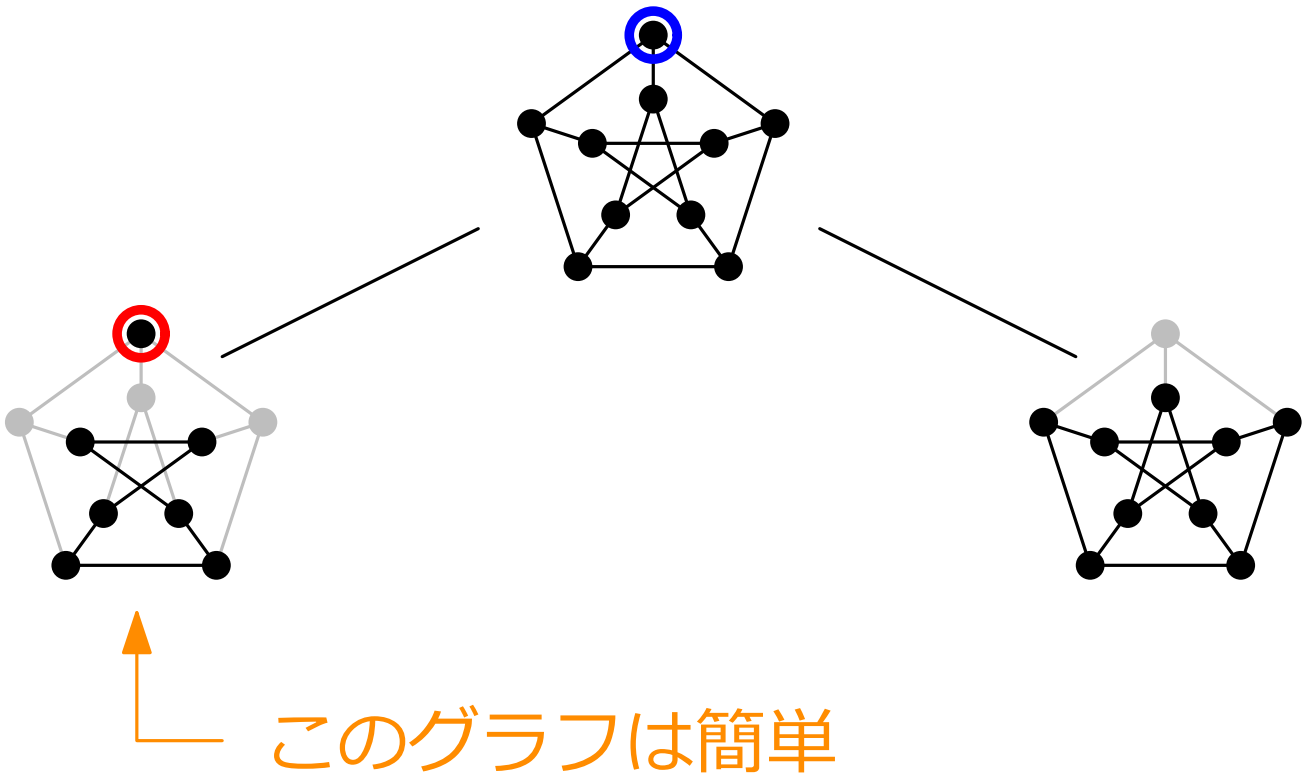


葉の数 $= 4$

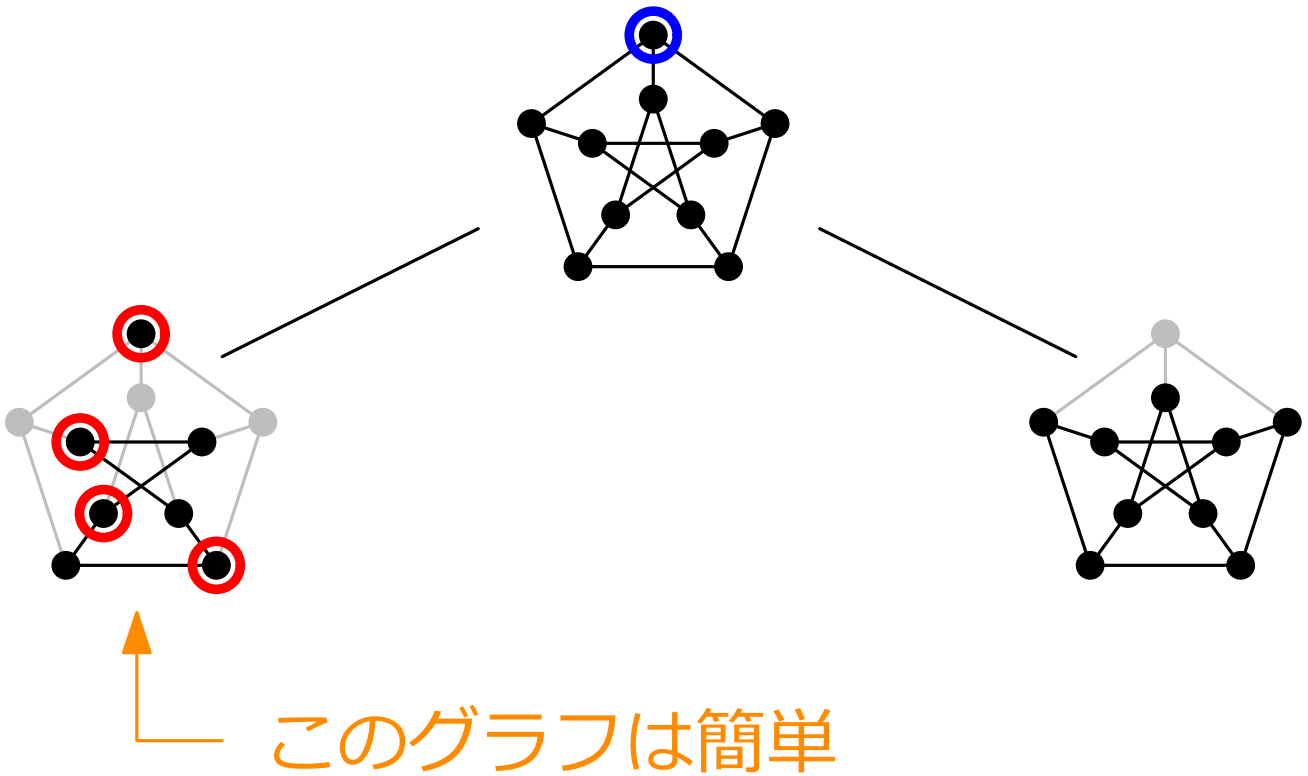




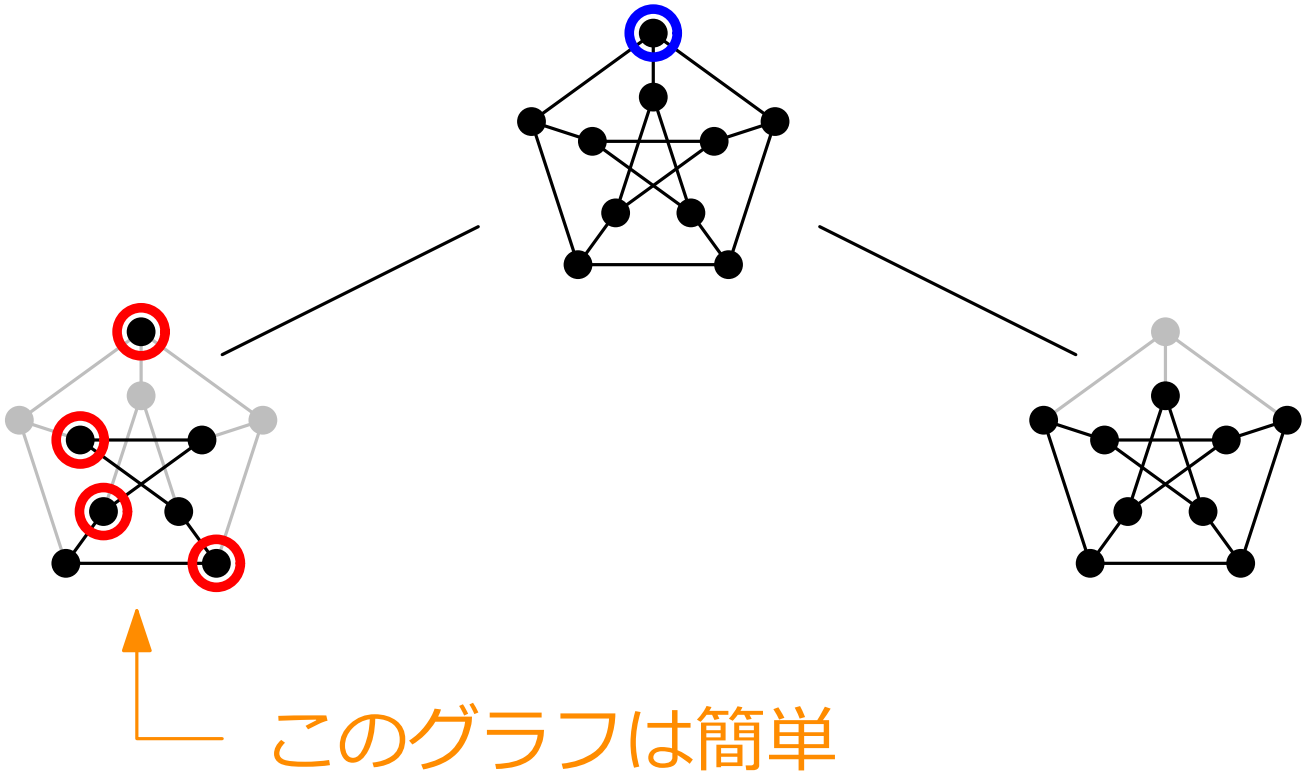
工夫 2

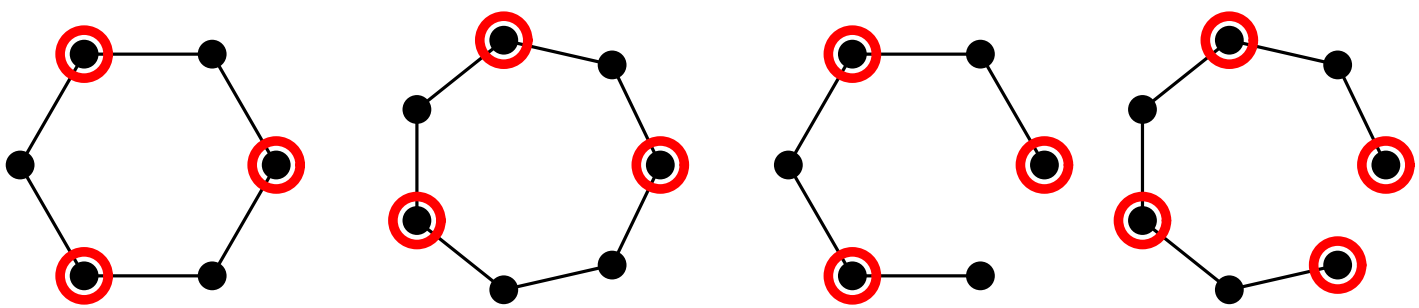


工夫 2

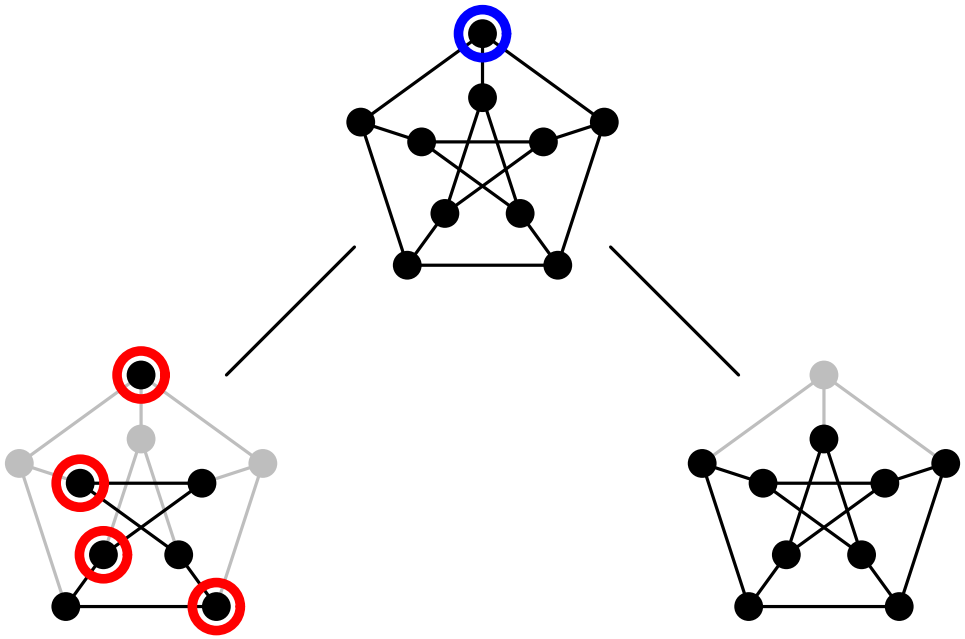


工夫 2

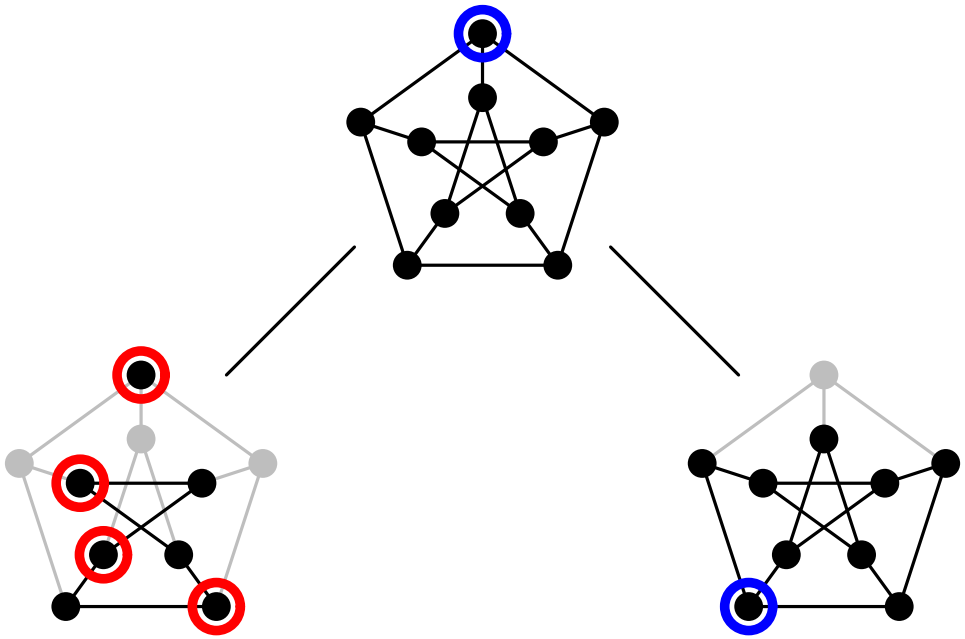




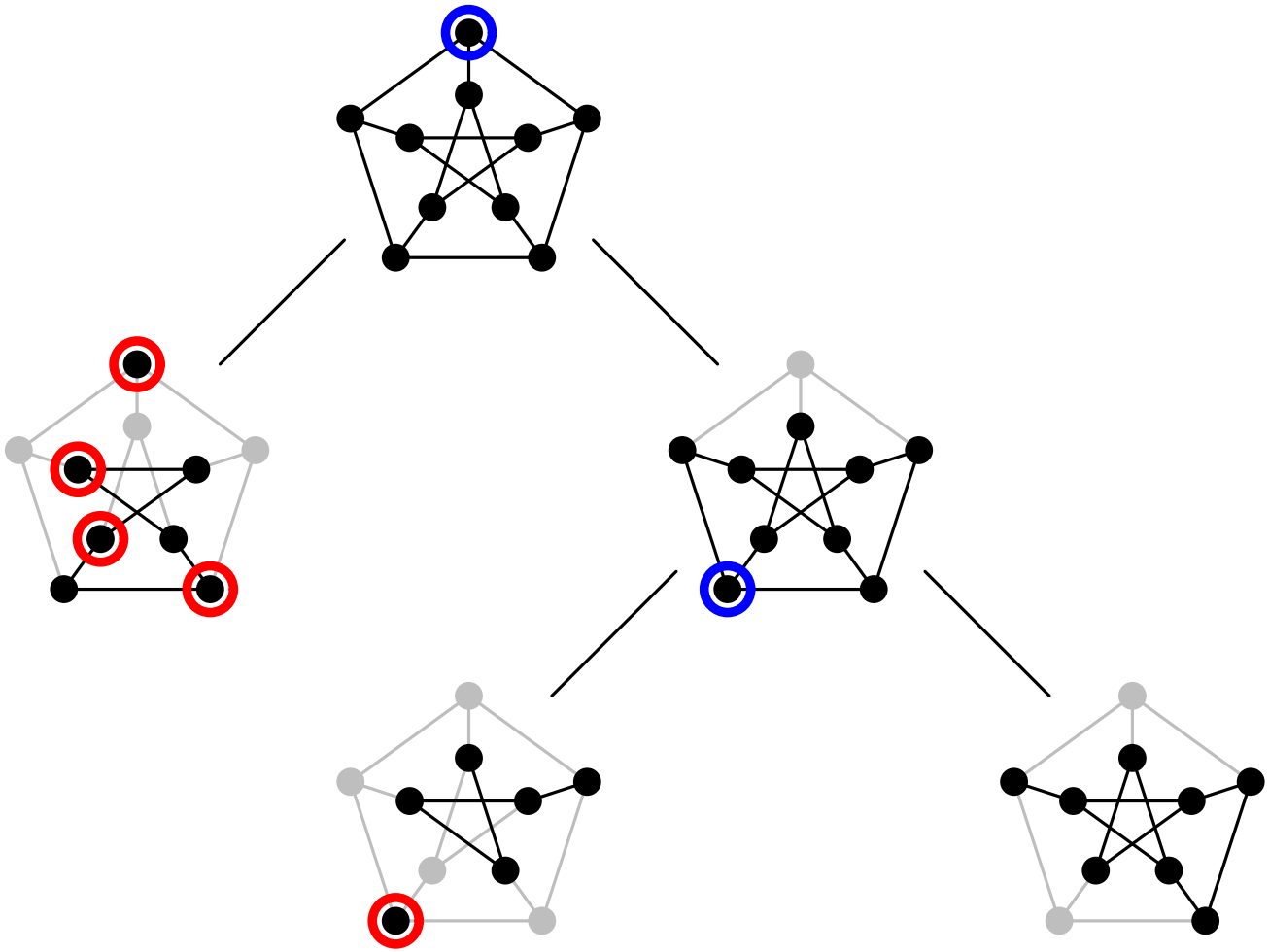
工夫をすると…：例



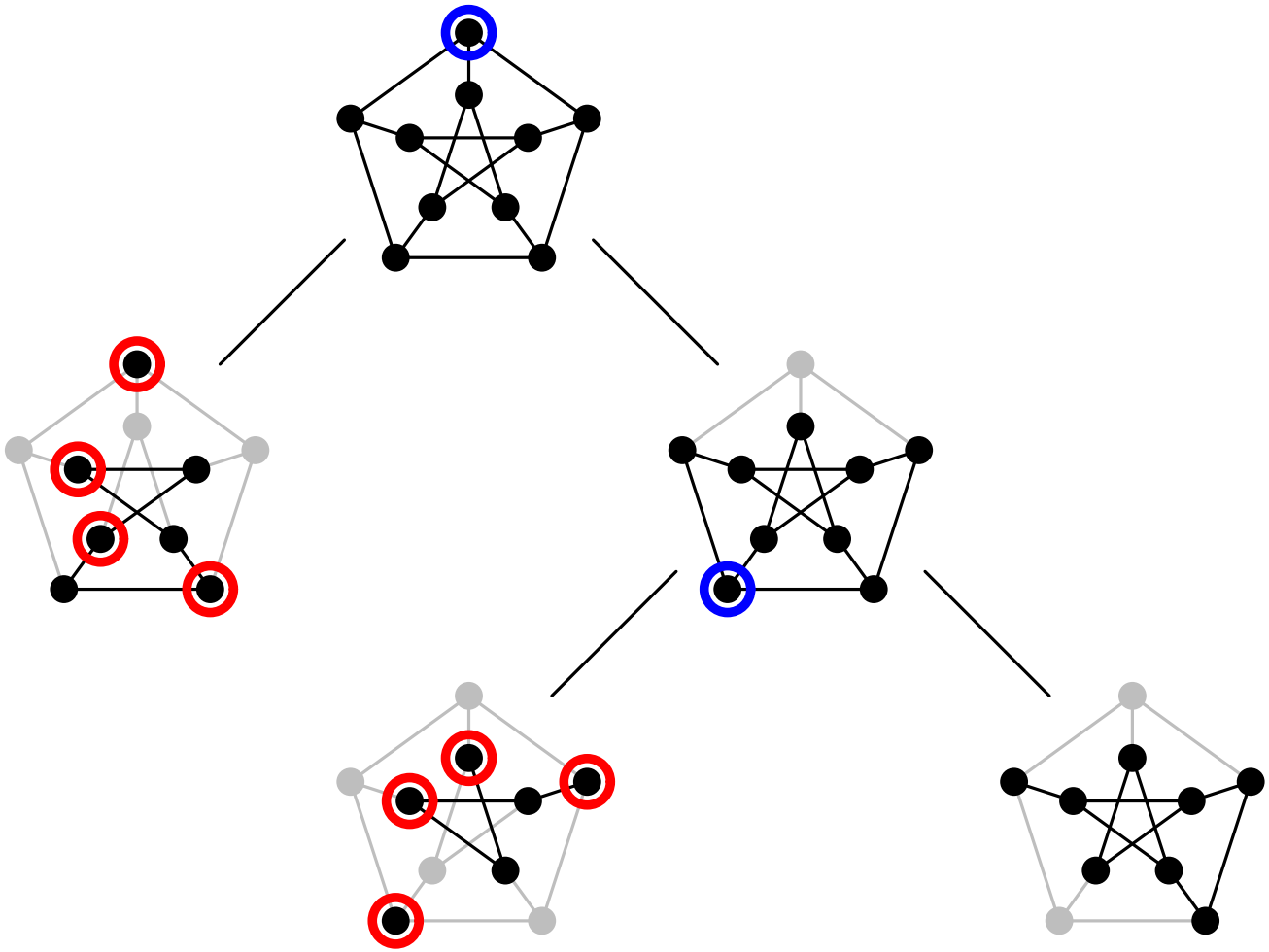
工夫をすると…：例



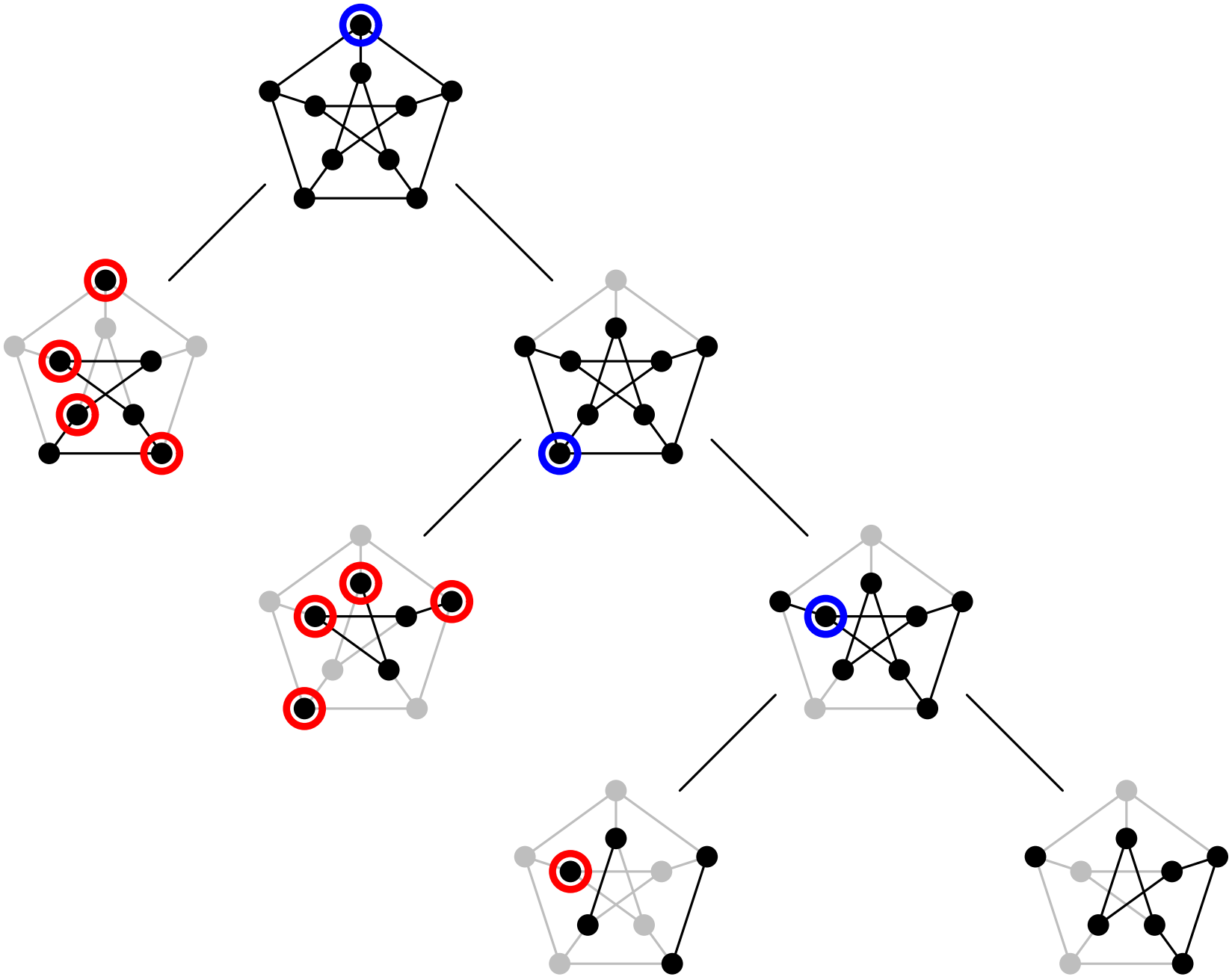
工夫をすると…：例



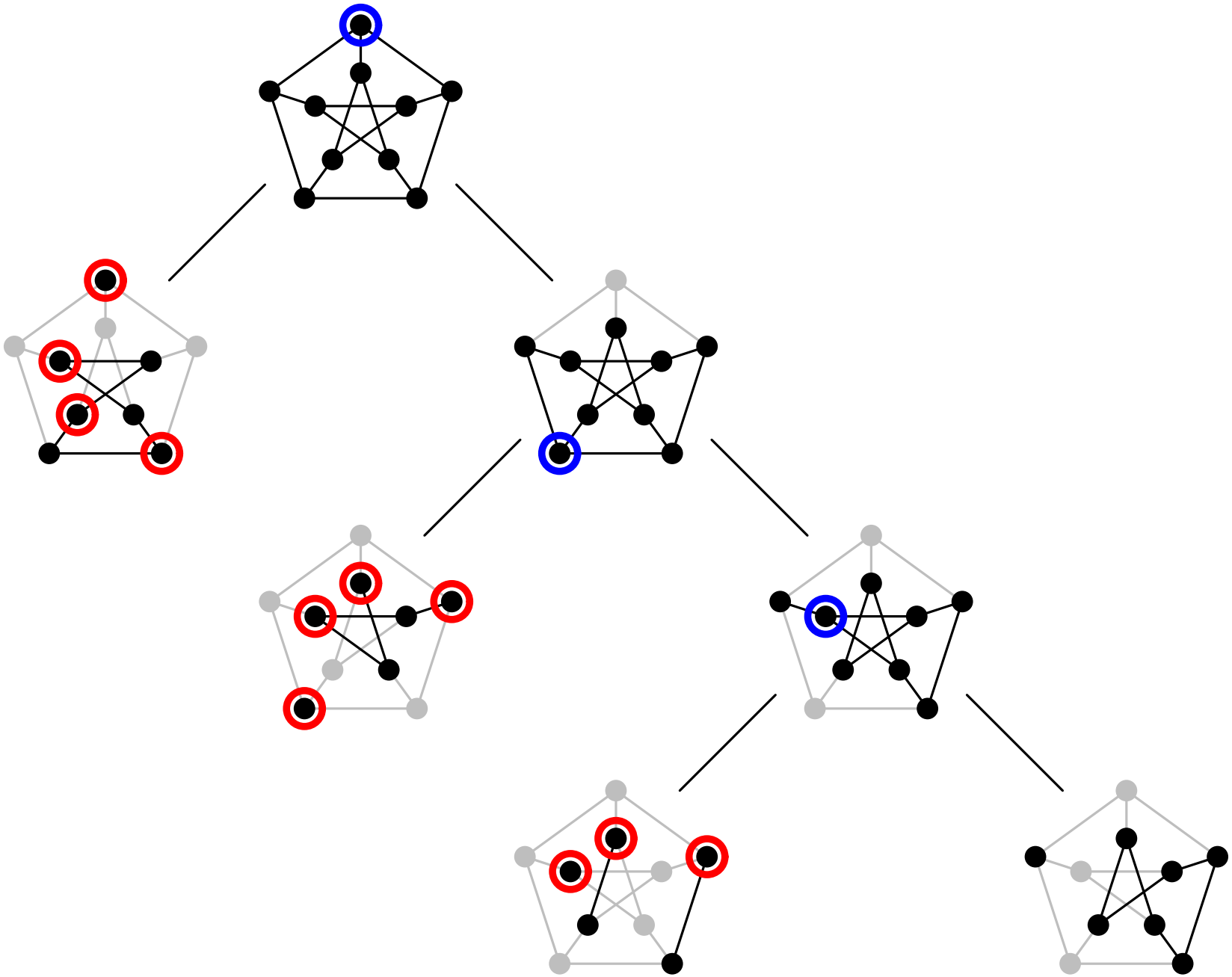
工夫をすると…：例



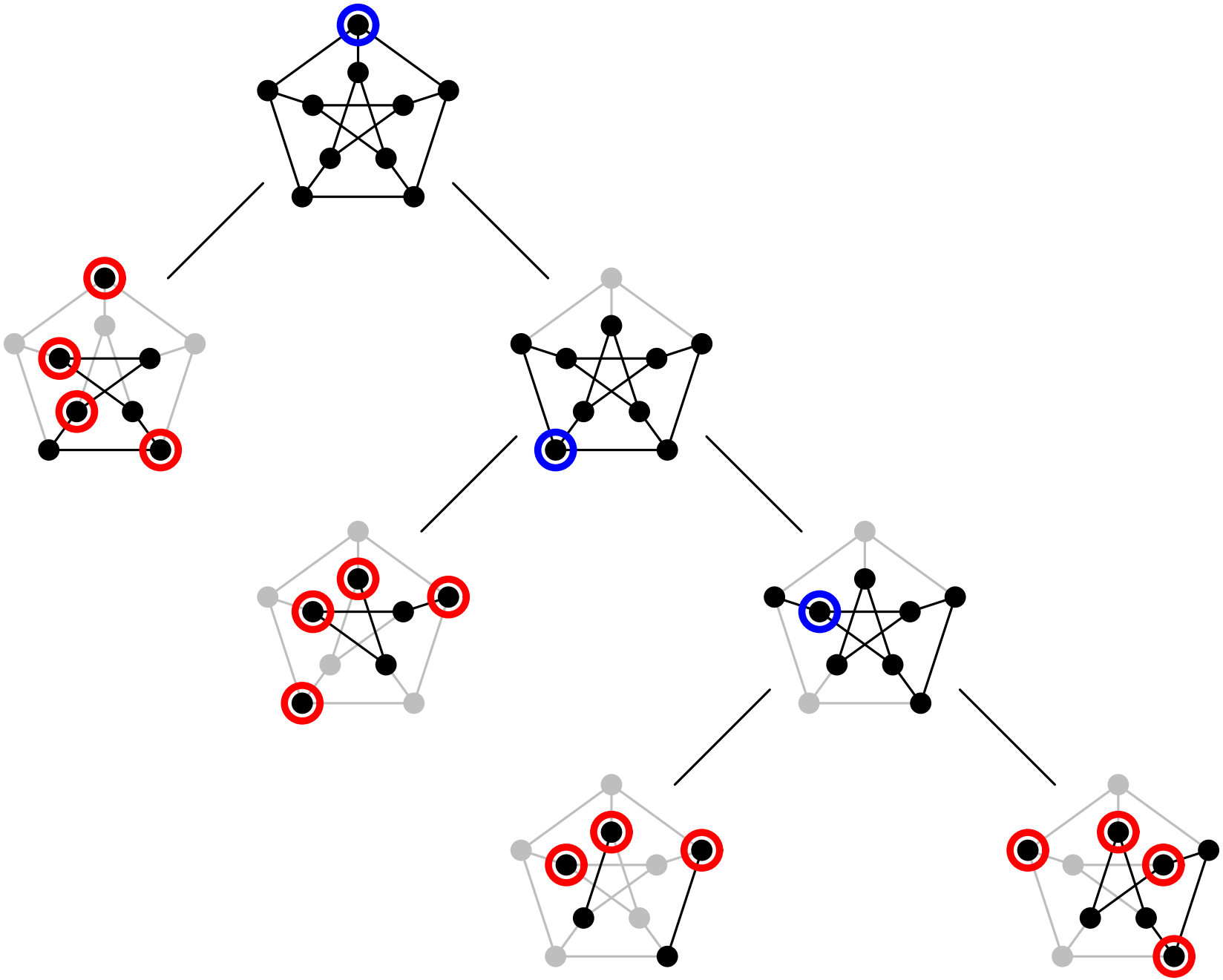
工夫をすると…：例



工夫をすると…：例

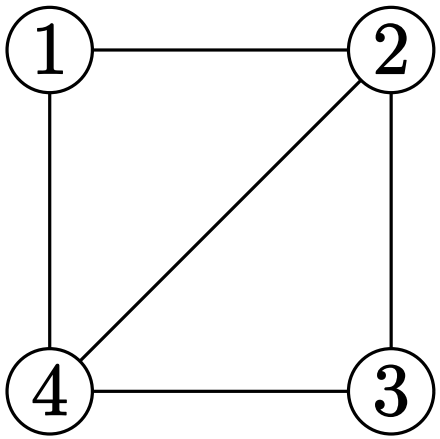


工夫をすると…：例



定義：頂点の次数

定義：頂点 v の **次数** (degree) とは，v に隣接する頂点の数
$\deg(v)$ で表すことがある



- $\deg(1) = 2$
- $\deg(2) = 3$
- $\deg(3) = 2$
- $\deg(4) = 3$

このグラフにおける
- 最大次数 $= 3$
- 最小次数 $= 2$

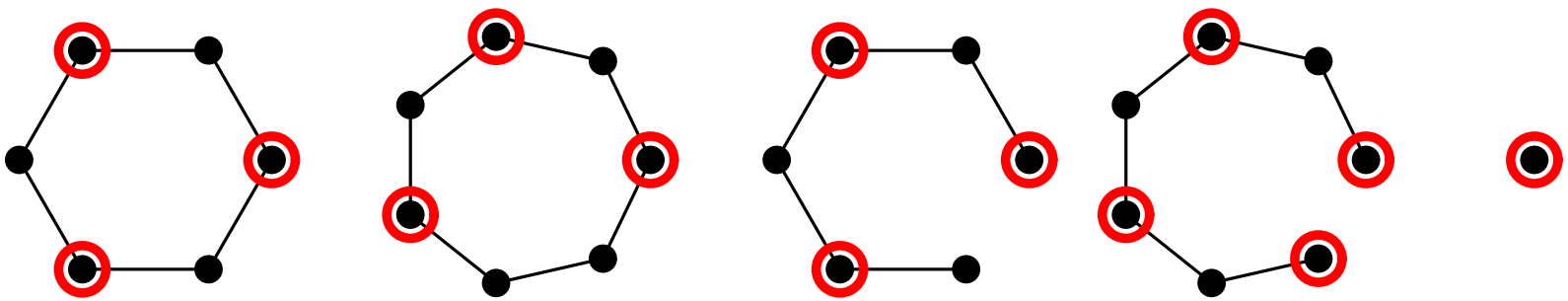
定義：頂点 v の **開近傍** $N(v) =$ 頂点 v の隣接頂点全体の集合
頂点 v の **閉近傍** $N[v] = N(v) \cup \{v\}$

- $N(3) = \{2, 4\}$，$N[3] = \{2, 3, 4\}$

工夫2：補題として

補題

最大次数が 2 以下のグラフに対して，
最大独立集合問題は線形時間で解ける

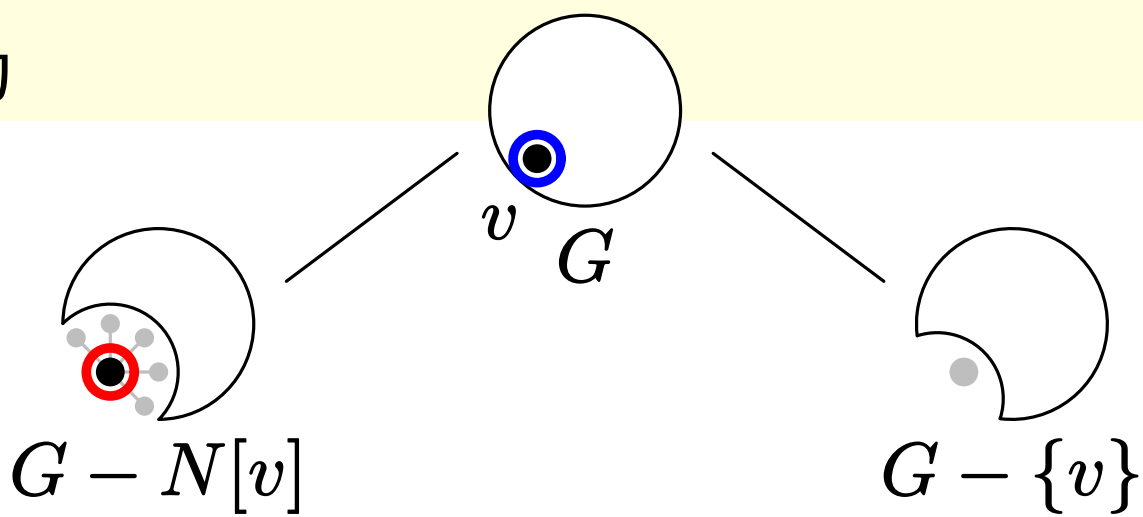


工夫したアルゴリズム

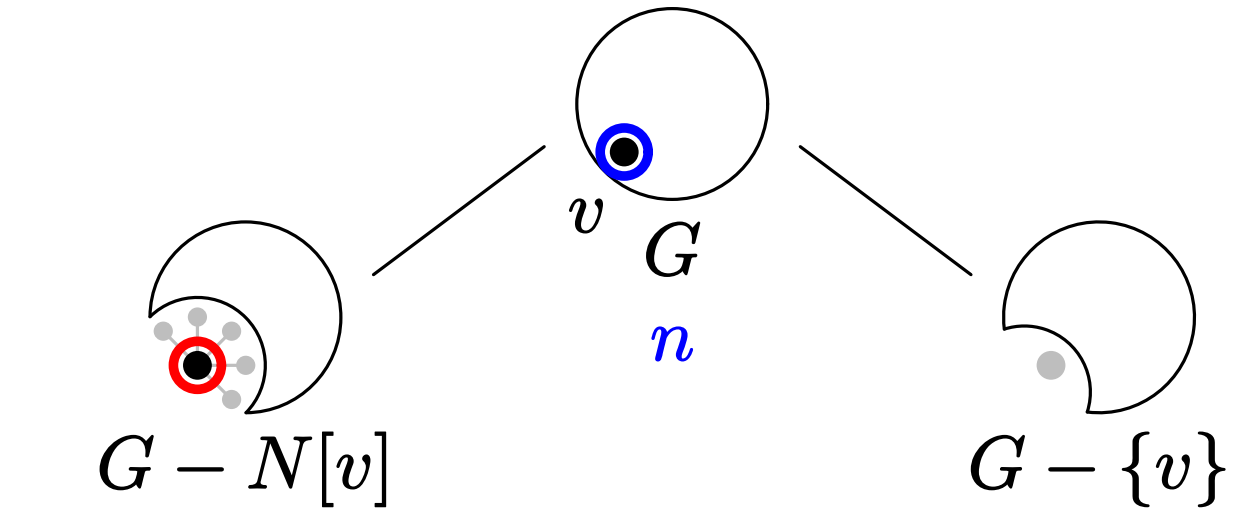
アルゴリズム A(G)

1. if G の最大次数 ≤ 2:
   - 補題を使って，最大独立集合を出力
2. $v = G$ の次数最大の頂点
3. 次の 2 つの大きいほうを出力
   - A($G - N[v]$) の出力 $\cup \{v\}$
   - A($G - \{v\}$) の出力

$G - X = G$ から X を除去してできるグラフ

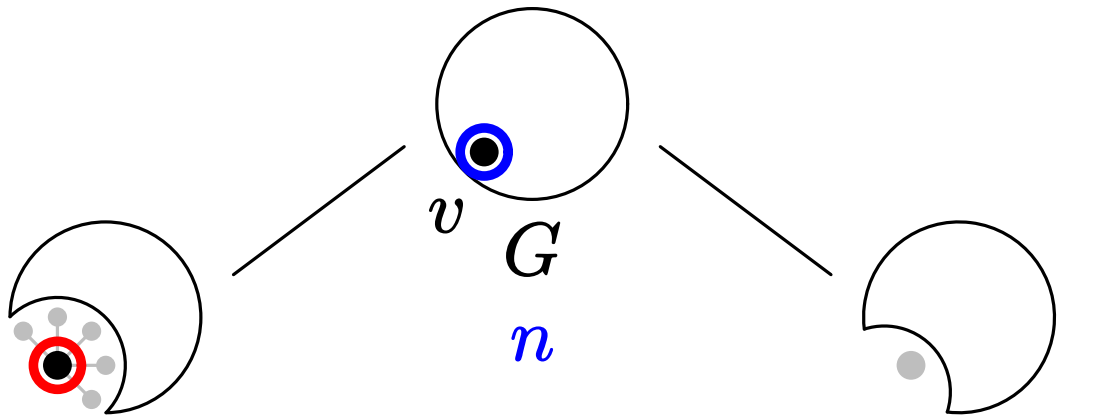


工夫：葉の数



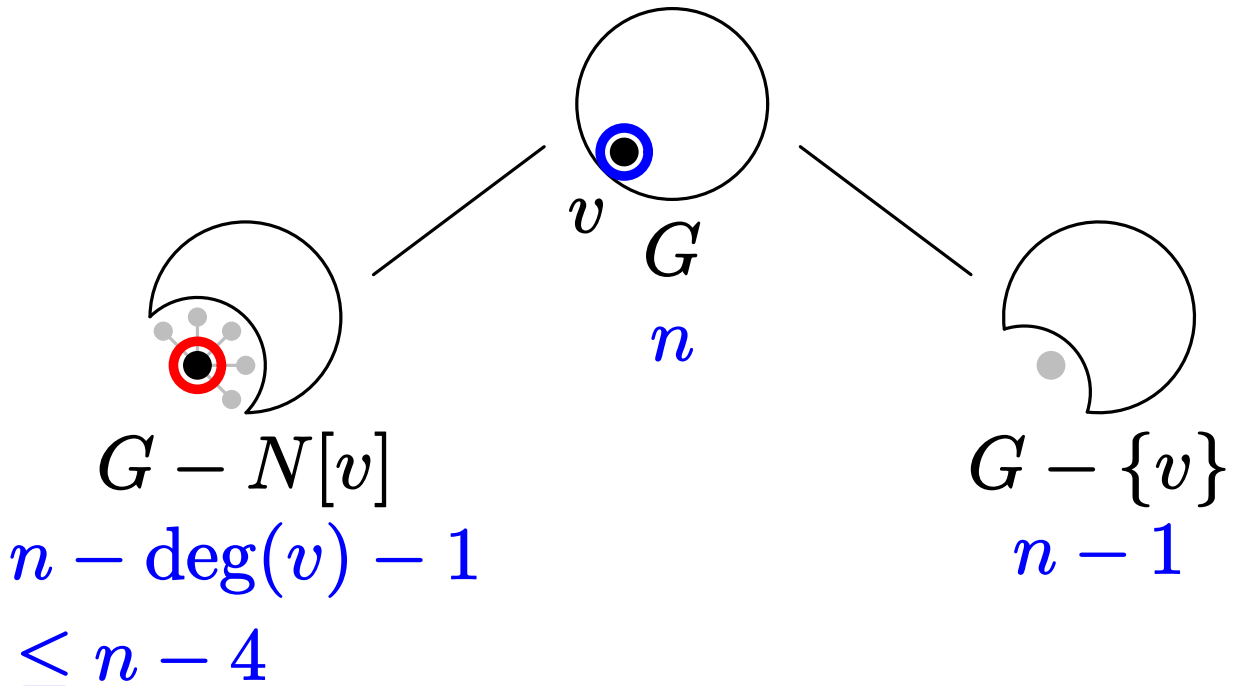
頂点数

工夫：葉の数



頂点数

工夫：葉の数



$T(n) =$ 頂点数 n のグラフに対する葉の数 (の最大値) とすると

$$T(n) \leq T(n-4) + T(n-1)$$

$$\therefore T(n) = O(1.3803^n)$$

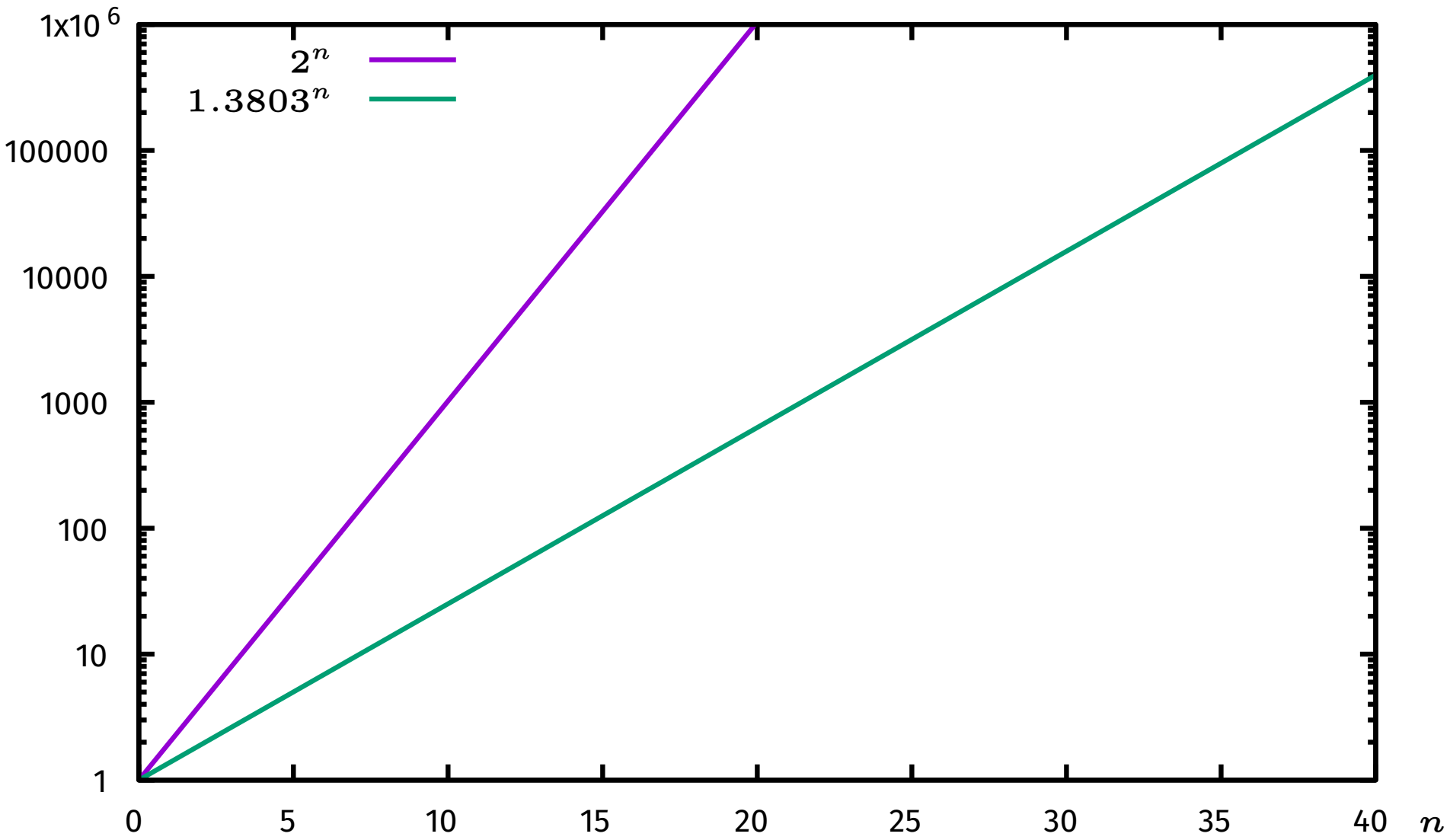
(導出は次回)

1.3803^n のオーダーは 2^n よりも **真に小さい**

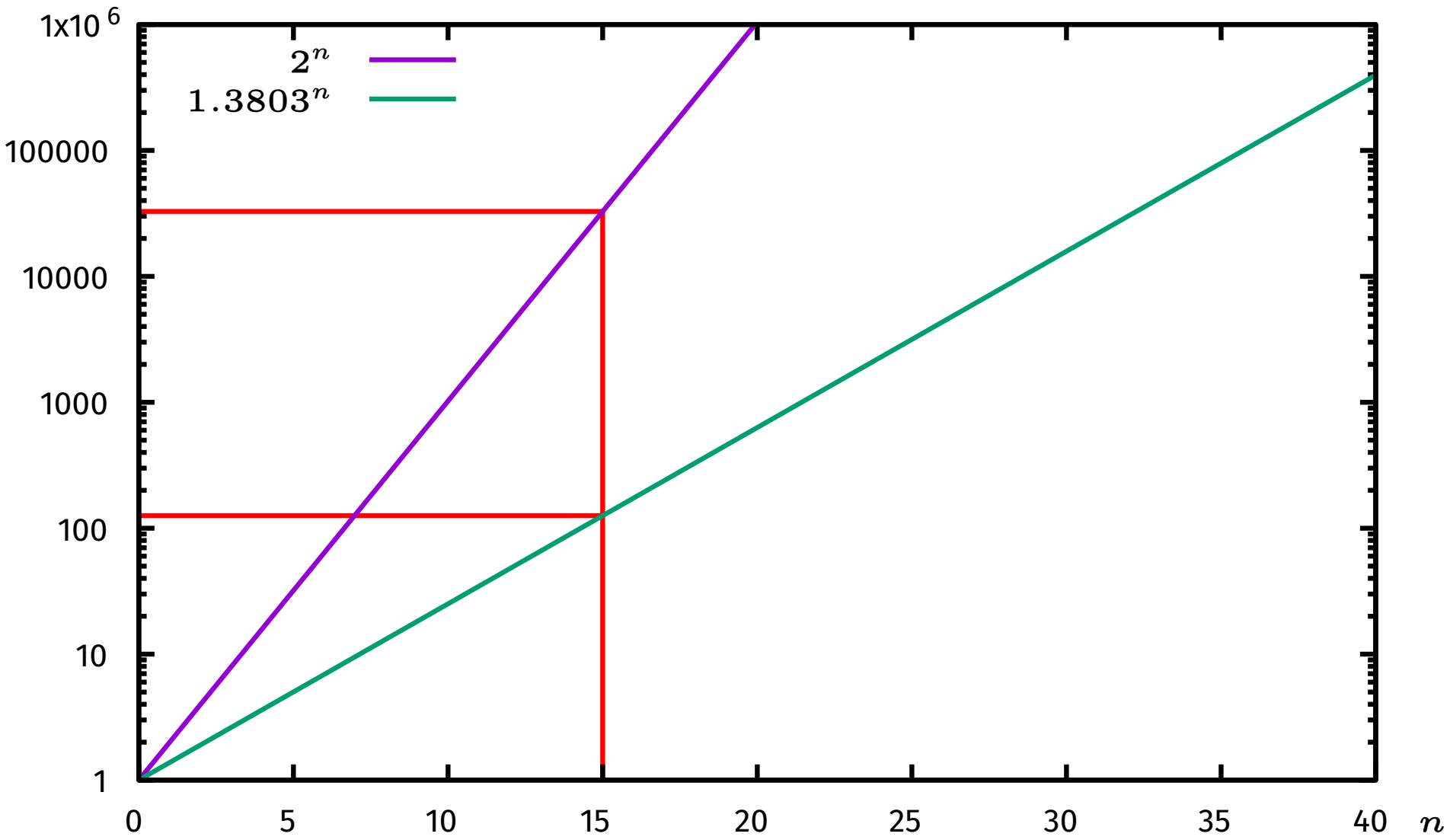
$$\frac{2^n}{1.3803^n} = \left(\frac{2}{1.3803}\right)^n = 1.4490^n \to \infty \quad (n \to \infty)$$

計算量 1.3803^n は計算量 2^n よりも **指数関数的に速い**

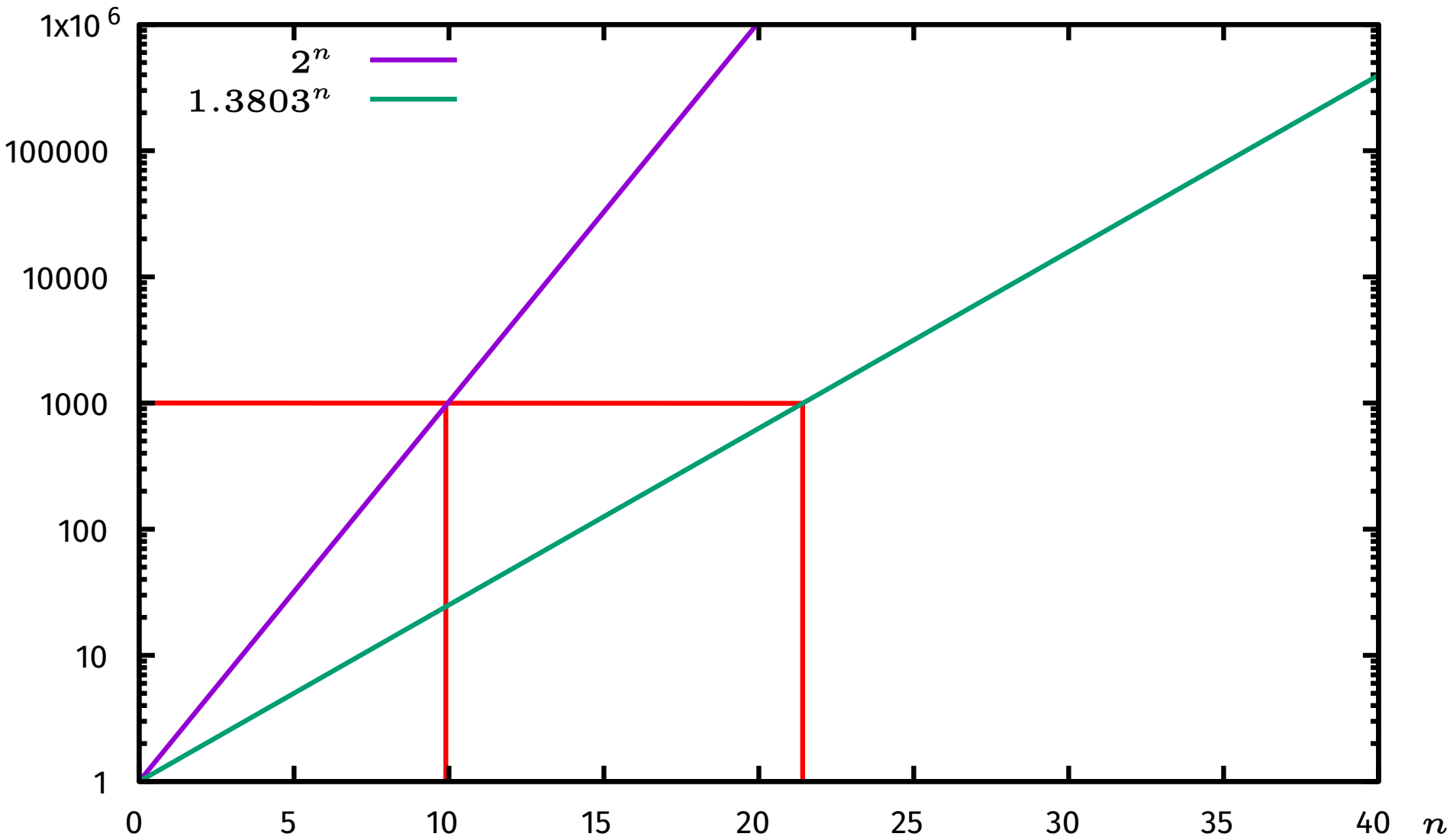
2^n vs 1.3803^n : プロット



2^n vs 1.3803^n : プロット



2^n vs 1.3803^n : プロット



本日の内容

1. 高速指数時間アルゴリズムの例
2. **高速指数時間アルゴリズムの目標**

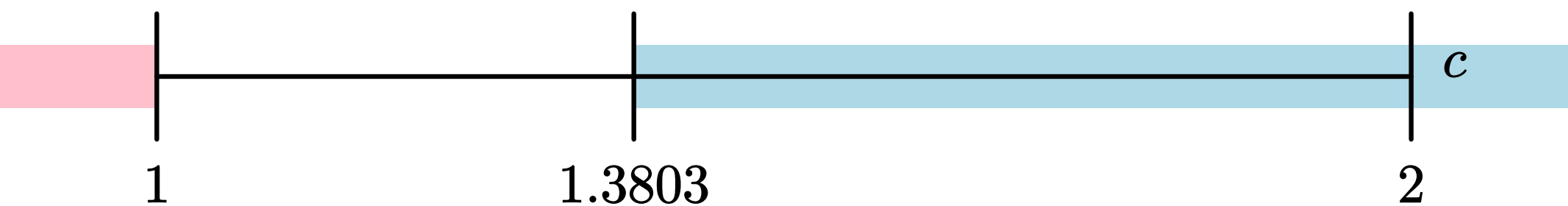
高速指数時間アルゴリズムの目標１

高速指数時間アルゴリズムの目標１

計算量が $O(c^n)$ となるとき

- c をできるだけ小さくすること
- そのような c の限界を知ること

最大独立集合問題に対して



不可能

P ≠ NP の仮定の下で

可能

ポイント

「簡単な工夫」が計算量を **理論的に (数学的に)** 改善する

高速指数時間アルゴリズムの目標 2

高速指数時間アルゴリズムの目標 2

計算量が $2^{O(f(n))}$ となるとき

- $f(n)$ をできるだけ小さくすること
- そのような $f(n)$ の限界を知ること

ポイント

「簡単な工夫」が計算量を **理論的に (数学的に)** 改善する

例：巡回セールスマン問題

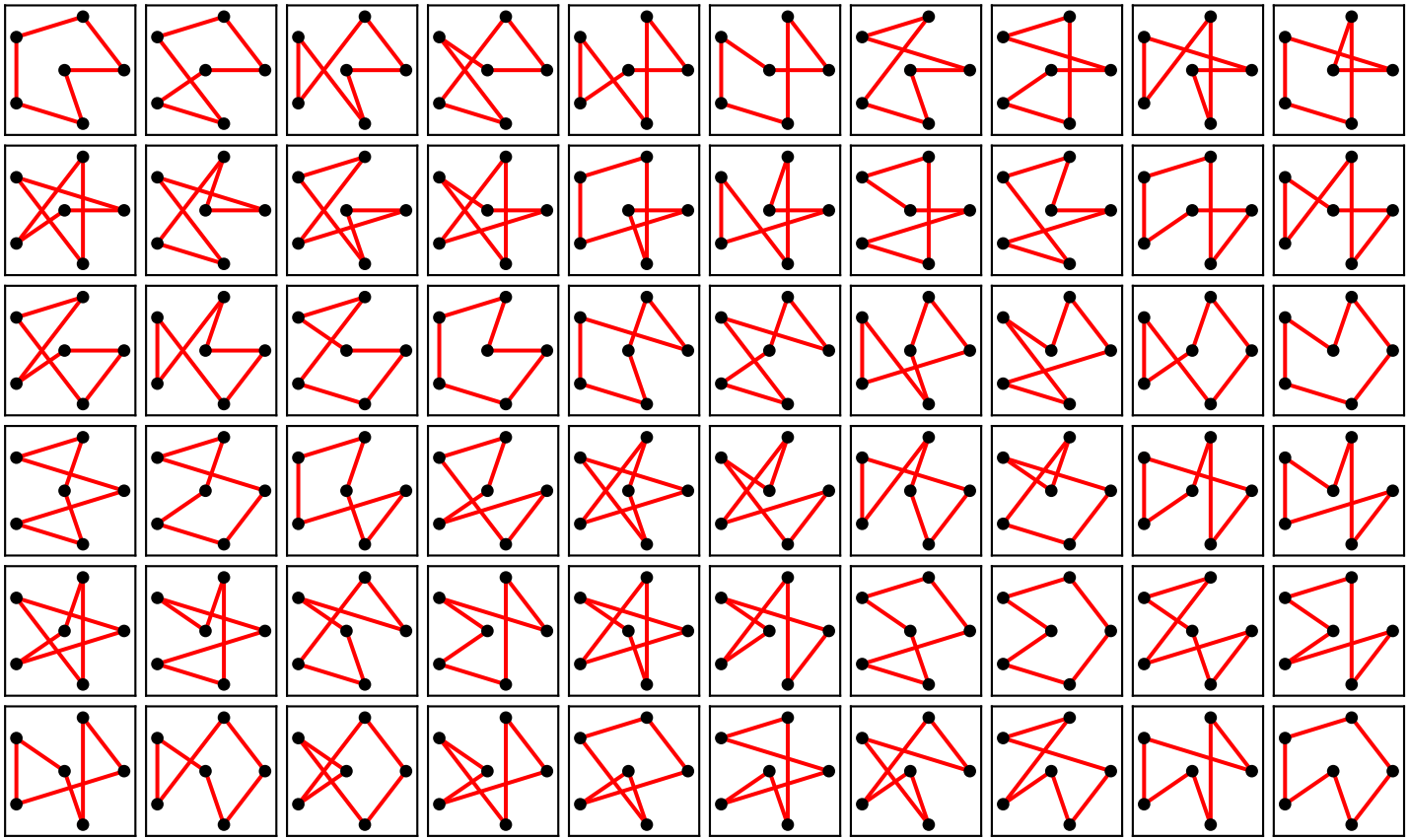
問題：巡回セールスマン問題

入力： 都市の集合 V，都市間の距離行列 $d \in \mathbb{R}^{V \times V}$
出力： V のすべての都市を回って戻る経路の中で，
距離和が最小のもの

入力　　出力

注：巡回セールスマン問題は NP 困難 (Karp '72)

しらみつぶし？



都市数 $n \Rightarrow$ 巡回路数 $= \dfrac{1}{2}(n-1)! = 2^{O(n \log n)}$

補足：スターリングの公式 $n! \approx \sqrt{2\pi n}\left(\dfrac{n}{\mathrm{e}}\right)^n$

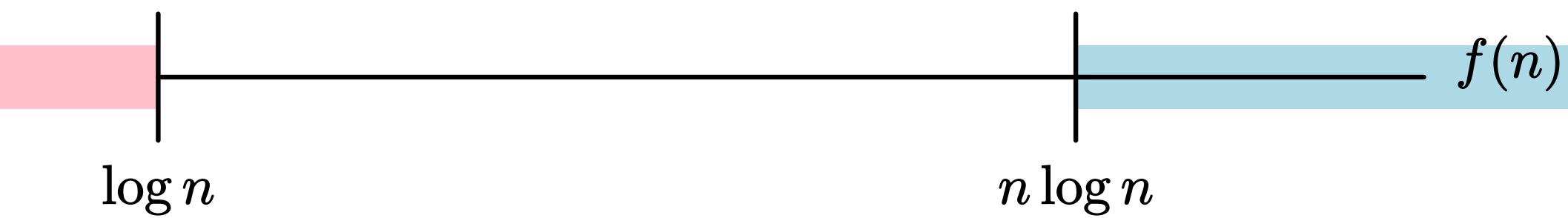
高速指数時間アルゴリズムの目標 2 (再)

高速指数時間アルゴリズムの目標 2

計算量が $2^{O(f(n))}$ となるとき

- $f(n)$ をできるだけ小さくすること
- そのような $f(n)$ の限界を知ること

巡回セールスマン問題に対して



不可能

P ≠ NP の仮定の下で

可能

注：$2^{c \log_2 n} = n^c$

O^* 記法

高速指数時間アルゴリズムに対して，次の記法をよく用いる

記法：O^* 記法

$O^*(f(n))$ と書いたら，
ある多項式 $p(n)$ に対して，$O(f(n)p(n))$ であることを表す

気持ち：多項式時間は無視

例

- $O(2^n n^2) = O^*(2^n)$
- $O(n^3) = O^*(1)$

次回の予告

次回以降 3 回

分枝アルゴリズム (branching algorithm) の設計と解析

第 2 回
- 分枝アルゴリズムの基礎

第 3 回
- 分枝アルゴリズムの改良法

第 4 回
- 測度統治法 (measure & conquer) による改良