

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 3 回

分枝アルゴリズム (2) : 高速化

岡本 吉央 (電気通信大学)

okamotoy@uec.ac.jp

2025 年 10 月 28 日

最終更新 : 2025 年 10 月 28 日 09:57

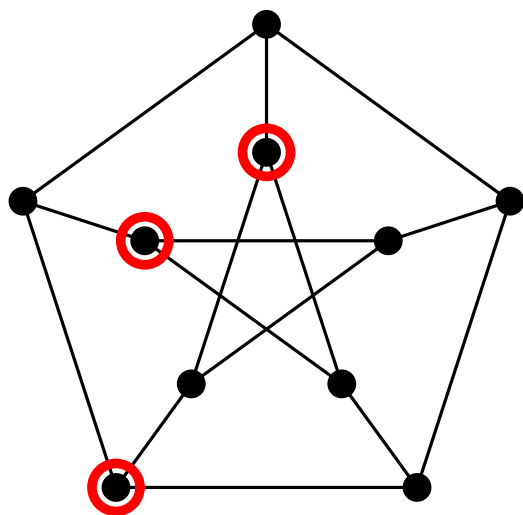
1. 高速指数時間アルゴリズムの考え方 (10/7)
 - * 休み (体育祭) (10/14)
2. 分枝アルゴリズム：基礎 (10/21)
3. **分枝アルゴリズム：高速化** (10/28)
4. 分枝アルゴリズム：測度統治法 (11/4)
5. 動的計画法：基礎 (11/11)
6. 動的計画法：例 (11/18)

- | | |
|-----------------|---------|
| 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) |

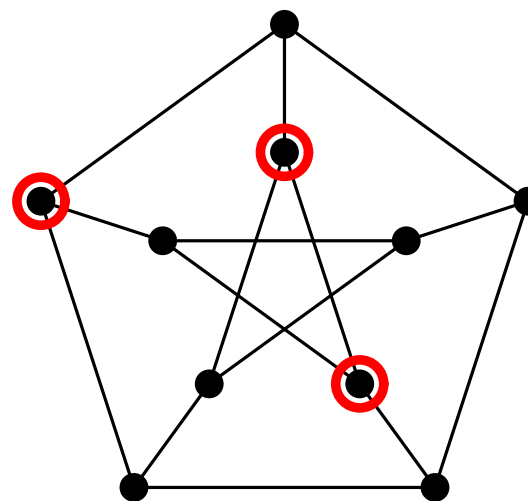
無向グラフ $G = (V, E)$

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



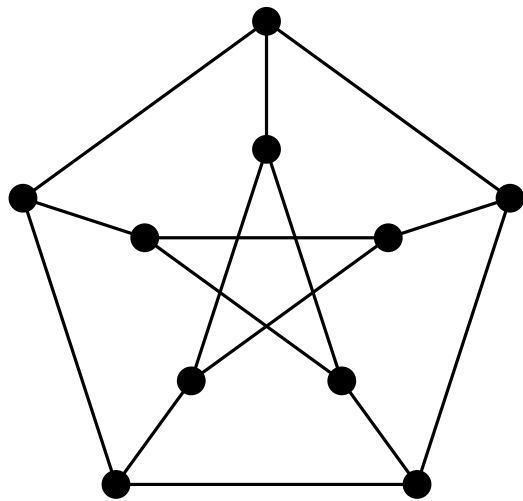
独立集合ではない

問題：最大独立集合問題

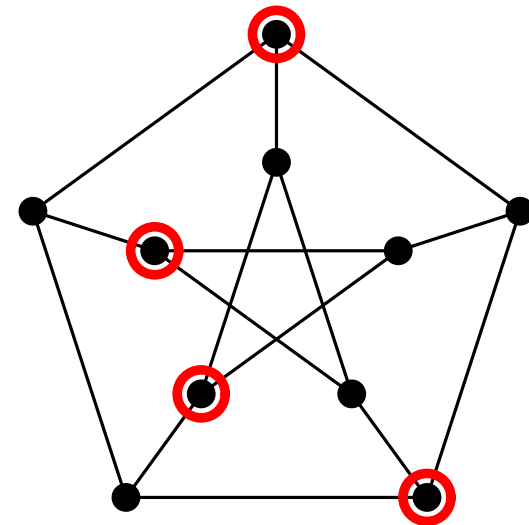
入力：無向グラフ $G = (V, E)$

出力： G の最大独立集合

要素数最大の独立集合



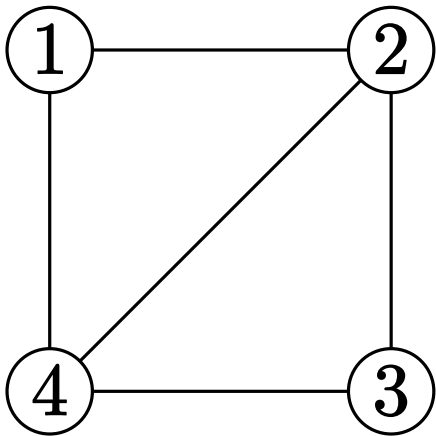
入力



出力

注：最大独立集合問題は NP 困難 (Karp '72)

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

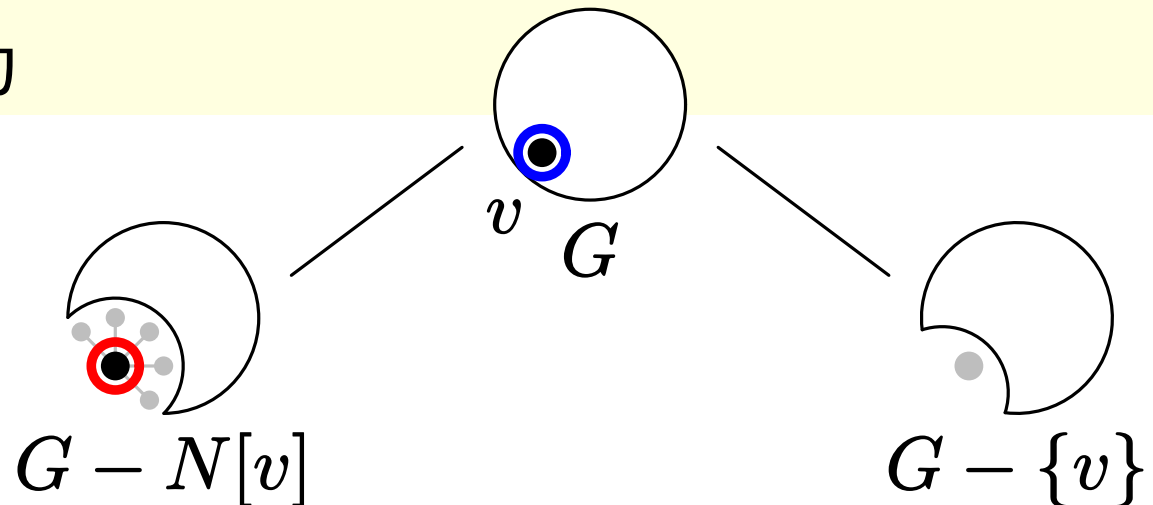
定義 : 頂点 v の **開近傍** $N(v)$ = 頂点 v の隣接頂点全体の集合
頂点 v の **閉近傍** $N[v] = N(v) \cup \{v\}$

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

アルゴリズム A(G)

1. if G の最大次数 ≤ 2 :
 - 多項式時間アルゴリズムで 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

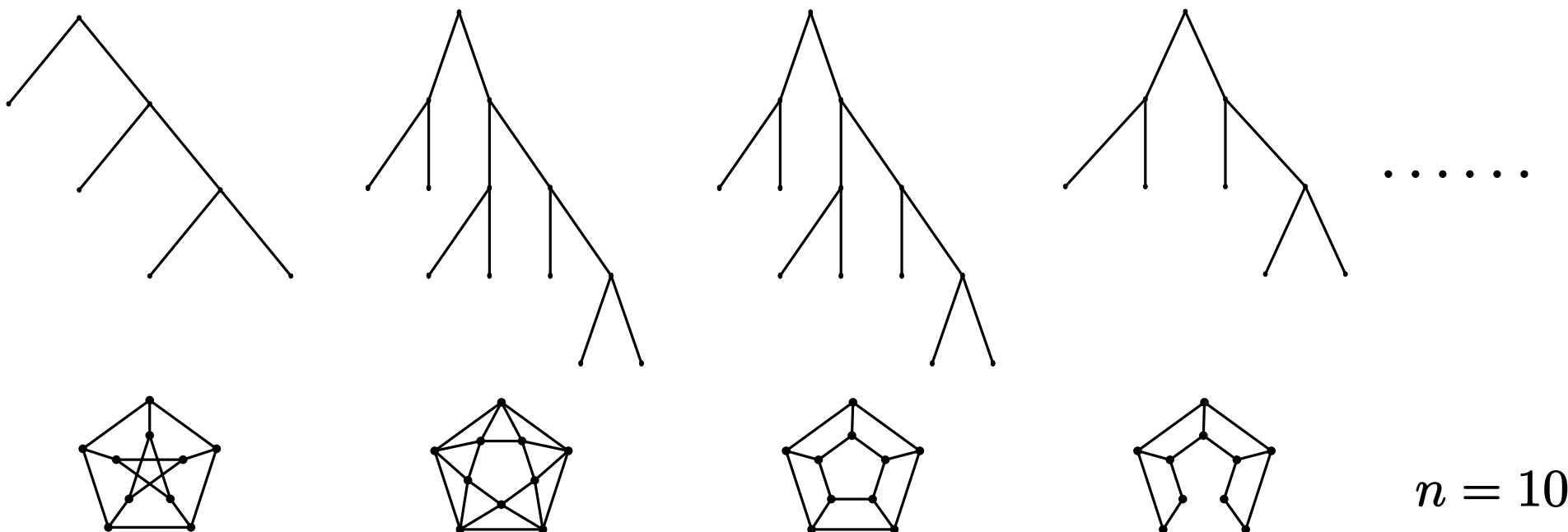
$G - X = G$ から X を
除去してできる
グラフ



記法

アルゴリズム $A(\cdot)$ に対して, 次を定義

- $T(G)$ = 入力を G としたときの探索木の葉の数
- $T(n) = \max\{T(G) \mid G \text{ の頂点数} \leq n\}$



記法

アルゴリズム $A(\cdot)$ に対して, 次を定義

- $T(G)$ = 入力を G としたときの探索木の葉の数
- $T(n) = \max\{T(G) \mid G \text{ の頂点数} \leq n\}$

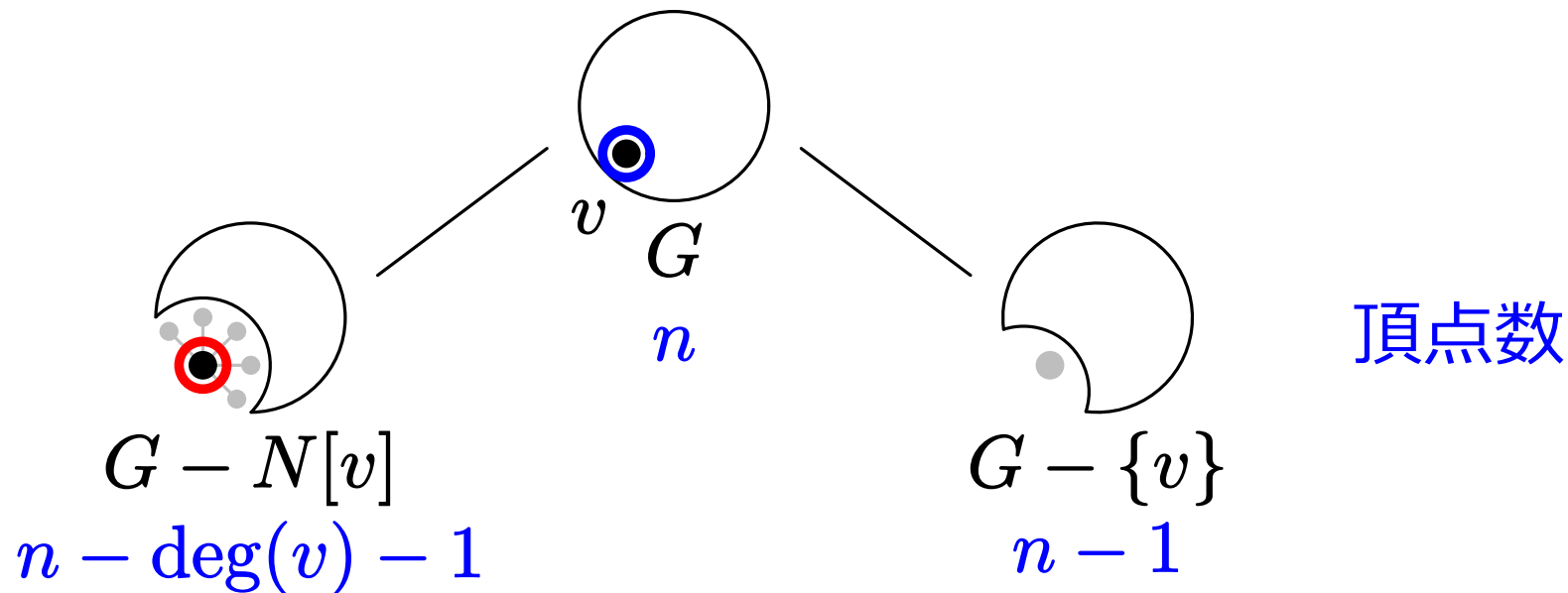
ほしいもの : $T(n)$ に対する小さな上界

性質

$$n \leq n' \Rightarrow T(n) \leq T(n')$$

正しいことは, 定義から直ちに分かる

$n \geq 4$ として, $T(n) = T(G)$ を満たすグラフ G を考える



$$\begin{aligned}
 T(n) = T(G) &= T(G - N[v]) + T(G - \{v\}) \\
 &\leq T(n - \deg(v) - 1) + T(n - 1) \\
 &\leq T(n - 4) + T(n - 1)
 \end{aligned}$$

$\therefore T(n) = \underline{O(1.3803^n)}$ (第2回で導出したとおり)

結論 : アルゴリズム $A(G)$ の計算量は $O^*(1.3803^n)$

本日の目標

アルゴリズム A よりも **数学的に高速な** アルゴリズムを設計する

紹介する補題は 最大独立集合問題 にしか適用できないが
基本的な考え方 は 他の問題 にも適用できる

- 前処理
- 分枝規則の精緻化

結論 : アルゴリズム $A(G)$ の計算量は $O^*(1.3803^n)$

1. **分枝アルゴリズムの高速化：前処理**
 2. 分枝アルゴリズムの高速化：分枝規則の精緻化
 3. 付録：メモ化 (記憶化)
-

仮に...

最大次数 ≤ 3 の場合に，多項式時間で解けたら...

アルゴリズム $A(G)$

1. if G の最大次数 ≤ 2 :
 - 多項式時間アルゴリズムで 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

仮に...

最大次数 ≤ 3 の場合に、多項式時間で解けたら...

アルゴリズム $A(G)$

1. if G の最大次数 $\leq \cancel{3}^3$:
 - 多項式時間アルゴリズムで 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

$$T(n) \leq T(n-5) + T(n-1)$$

$$\therefore T(n) = O(1.3247^n) \quad (\text{注：前は } O(1.3803^n))$$

アルゴリズム $A(G)$

1. if G の最大次数 $\leq \cancel{3}^3$:
 - 多項式時間アルゴリズムで 最大独立集合を出力
2. $v = G$ の次数最大の頂点 $\longleftarrow \deg(v) \geq 4$
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

$$T(n) \leq T(n-5) + T(n-1)$$

$$\therefore T(n) = O(1.3247^n) \quad (\text{注：前は } O(1.3803^n))$$

しかし：最大次数 ≤ 3 のとき，最大独立集合問題は NP 困難
(Garey, Johnson, Stockmeyer '76)

アルゴリズム $A(G)$

1. if G の最大次数 $\leq \cancel{3}^3$:
 - 多項式時間アルゴリズムで 最大独立集合を出力
2. $v = G$ の次数最大の頂点 $\longleftarrow \deg(v) \geq 4$
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

考え方

最大次数が 3 のときを 改善したい

最小次数で場合分け

- 最小次数 = 0 のとき
- 最小次数 = 1 のとき
- 最小次数 = 2 のとき
- 最小次数 = 3 のとき

まず、場合分けの中で役立つ補題をいくつか準備する

無向グラフ $G = (V, E)$

性質：優越

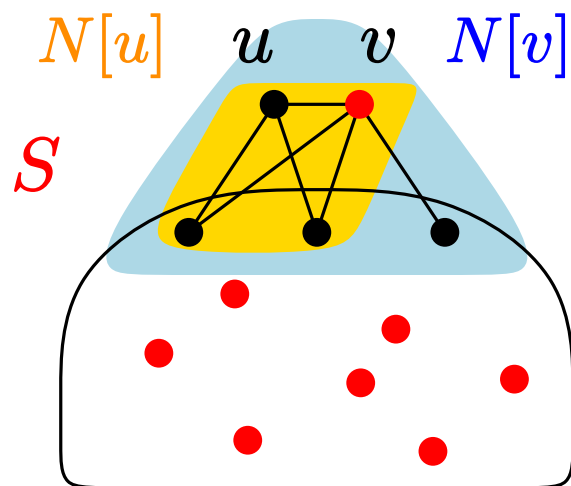
仮定 : $u, v \in V, u \neq v, N[u] \subseteq N[v]$

$$S \subseteq V, v \in S$$

結論： S が G の最大独立集合

$$\Rightarrow (S - \{v\}) \cup \{u\} \text{ は } G \text{ の最大独立集合}$$

証明：



無向グラフ $G = (V, E)$

性質：優越

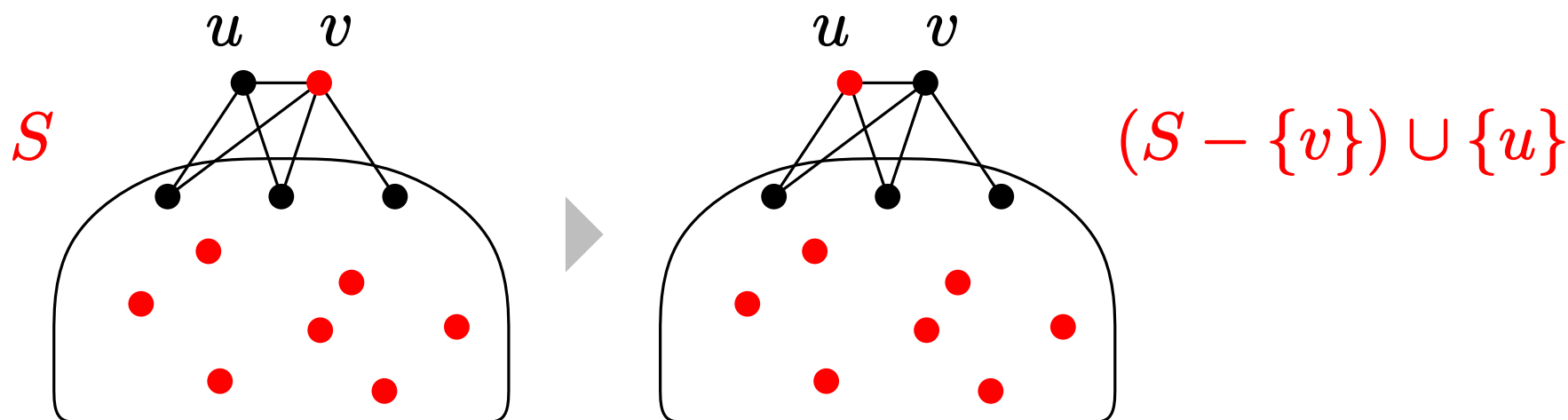
仮定： $u, v \in V, u \neq v, N[u] \subseteq N[v]$

$S \subseteq V, v \in S$

結論： S が G の最大独立集合

$\Rightarrow (S - \{v\}) \cup \{u\}$ は G の最大独立集合

証明：



無向グラフ $G = (V, E)$

性質：優越

仮定： $u, v \in V, u \neq v, N[u] \subseteq N[v]$

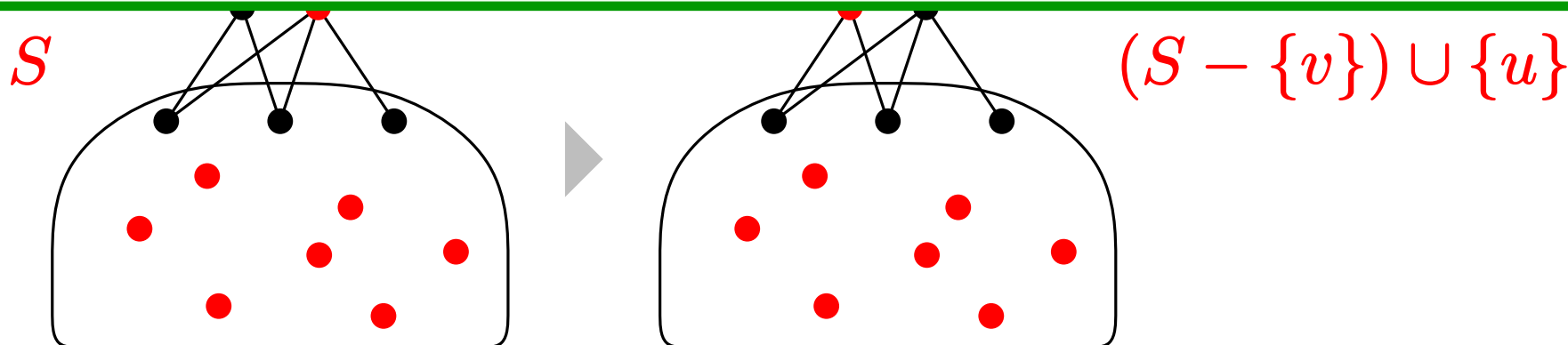
$S \subseteq V, v \in S$

結論： S が G の最大独立集合

$\Rightarrow (S - \{v\}) \cup \{u\}$ は G の最大独立集合

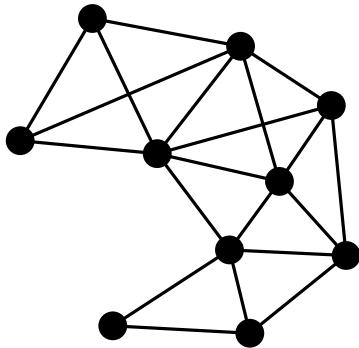
帰結

v を含まない最大独立集合が存在する



アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

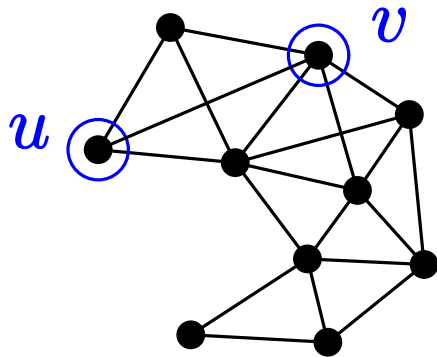


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

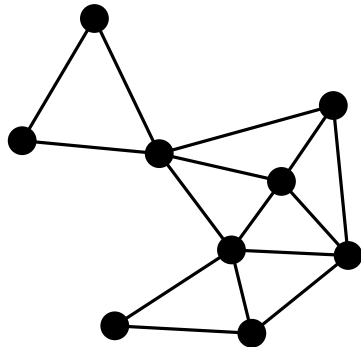


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

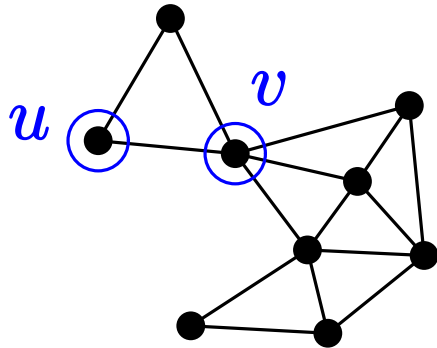


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

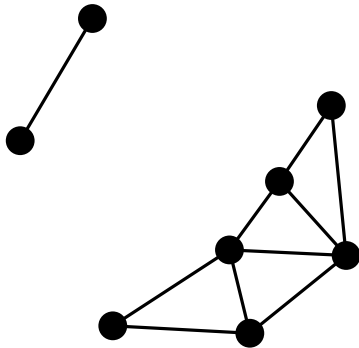


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

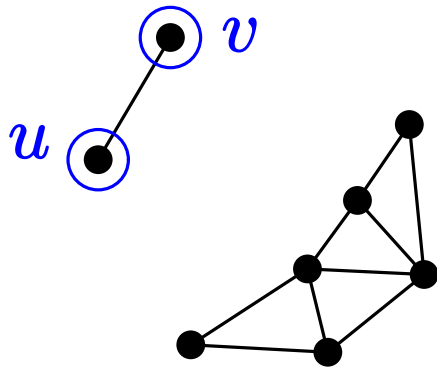


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

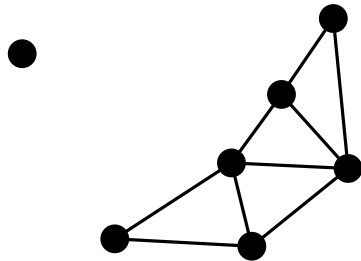


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

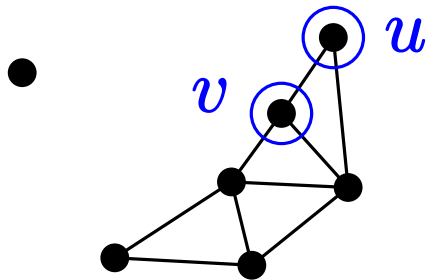


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

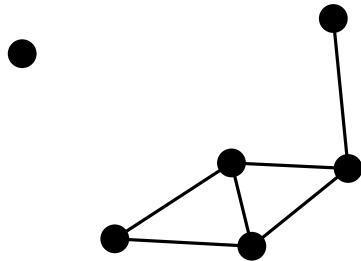


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

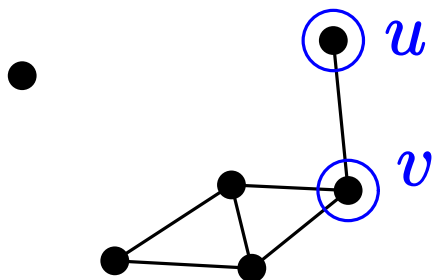


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

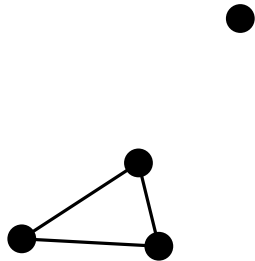


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

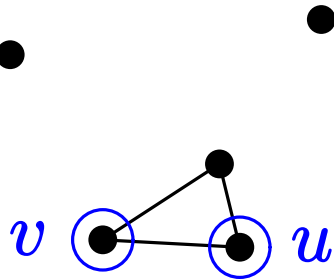


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

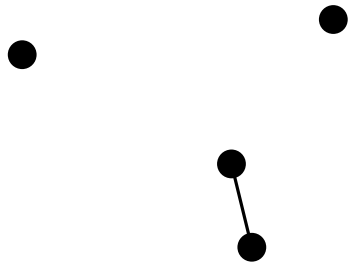


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

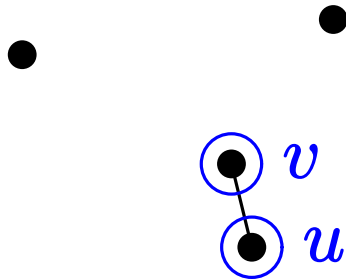


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

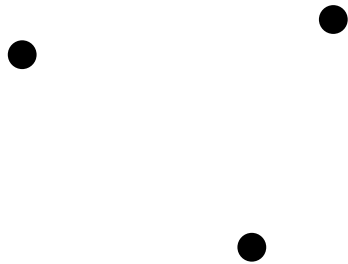


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

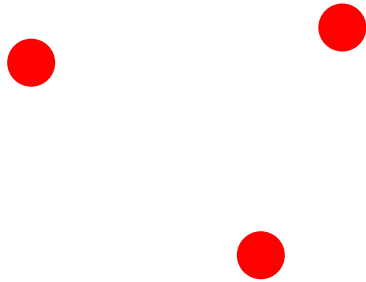


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

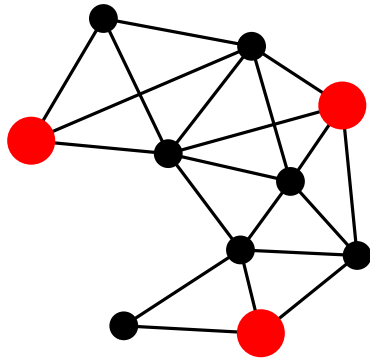


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

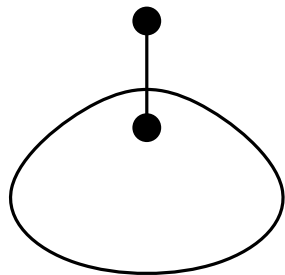


前のページの性質 (とその帰結) より

S が 優越規則(G) の最大独立集合
 $\Rightarrow S$ が G の最大独立集合

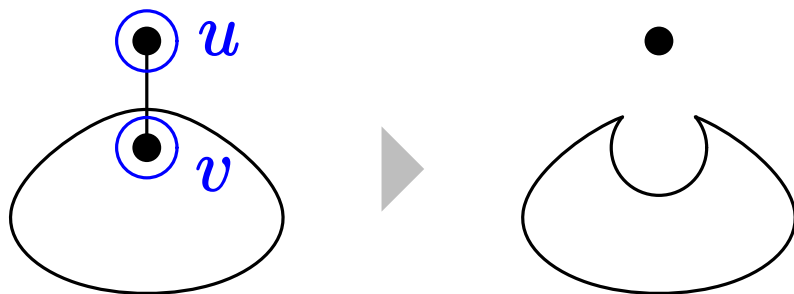
アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力



アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力

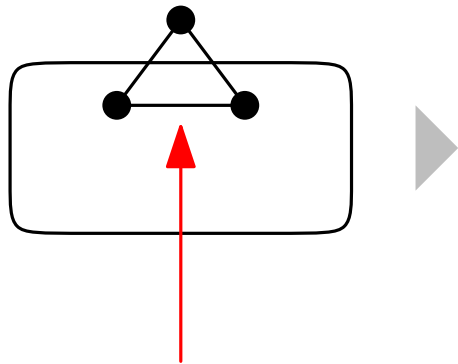


性質：次数1の頂点

優越規則(G) には, 次数1の頂点が存在しない

アルゴリズム 優越規則(G)

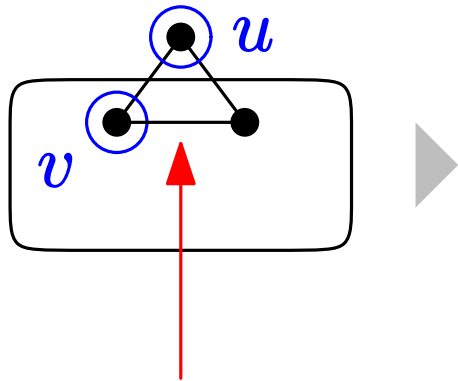
1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力



隣接している場合

アルゴリズム 優越規則(G)

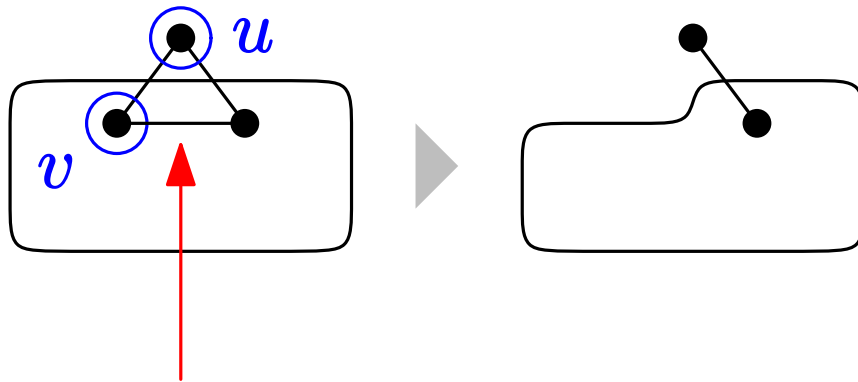
1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力



隣接している場合

アルゴリズム 優越規則(G)

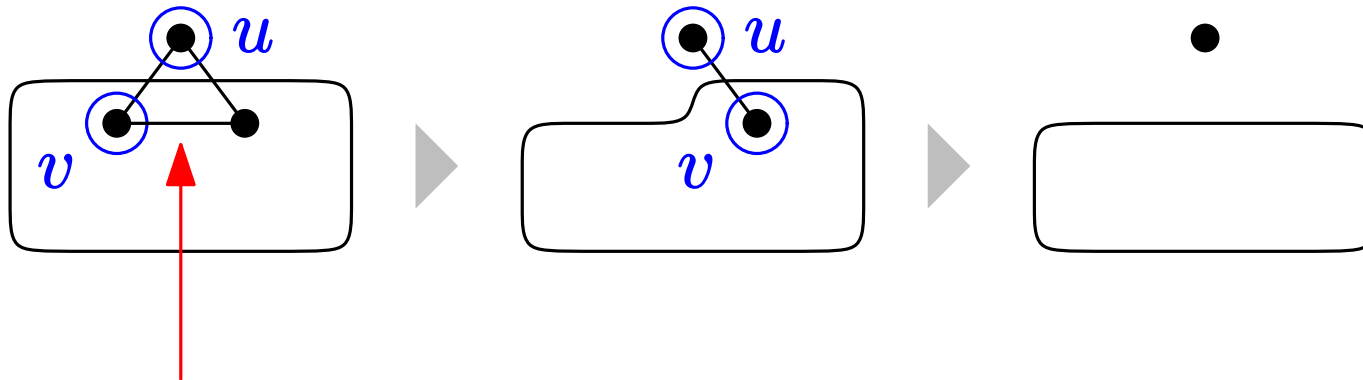
1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力



隣接している場合

アルゴリズム 優越規則(G)

1. ある異なる頂点 u, v が $N[u] \subseteq N[v]$ を満たす限り反復
 - $G = G - \{v\}$
2. G を出力



隣接している場合

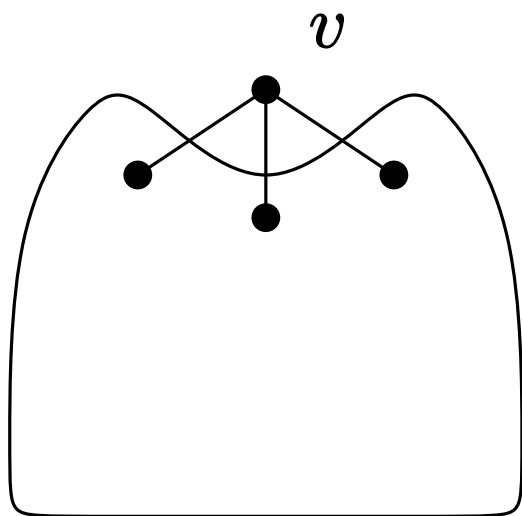
無向グラフ $G = (V, E)$

補題：近傍から 2 頂点

仮定： $v \in V$

G のどの最大独立集合も v を含まない

結論： S が G の最大独立集合 $\Rightarrow |S \cap N(v)| \geq 2$



無向グラフ $G = (V, E)$

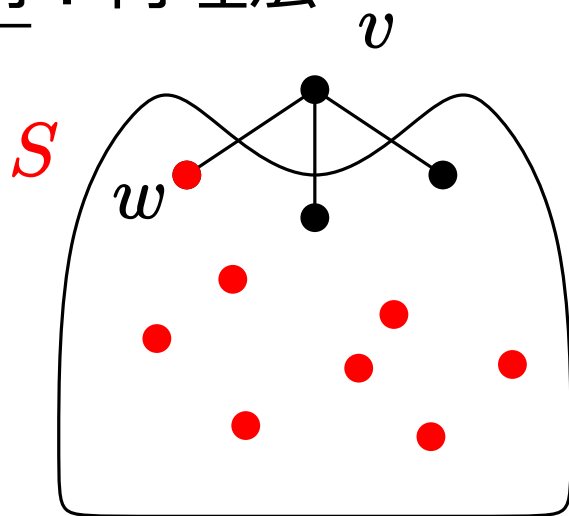
補題：近傍から 2 頂点

仮定： $v \in V$

G のどの最大独立集合も v を含まない

結論： S が G の最大独立集合 $\Rightarrow |S \cap N(v)| \geq 2$

証明： 背理法



無向グラフ $G = (V, E)$

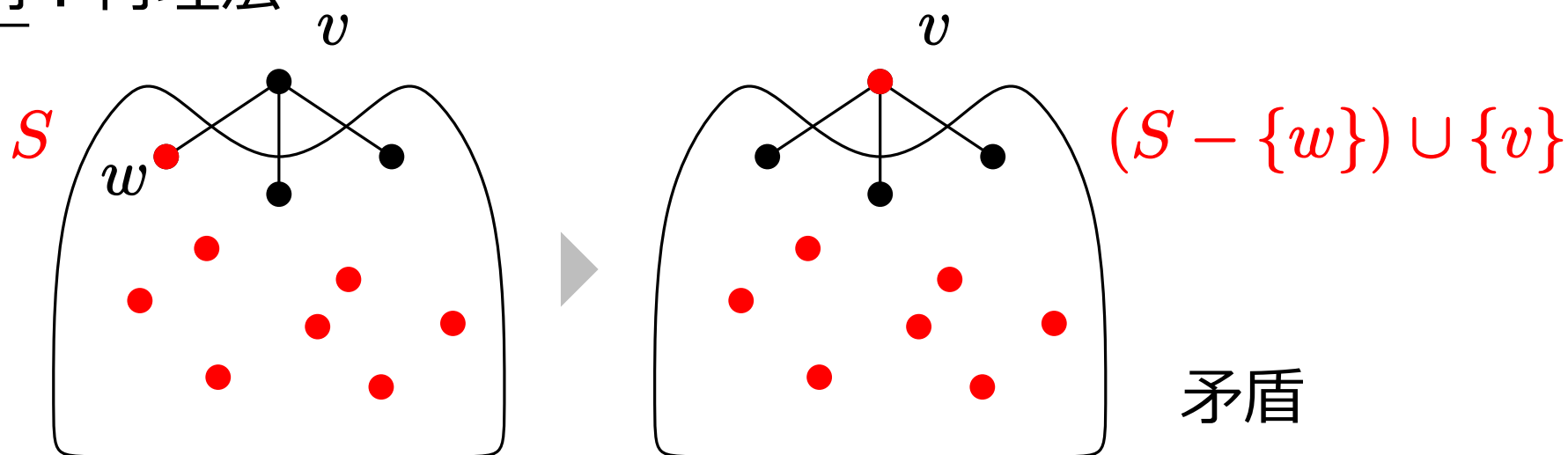
補題：近傍から 2 頂点

仮定： $v \in V$

G のどの最大独立集合も v を含まない

結論： S が G の最大独立集合 $\Rightarrow |S \cap N(v)| \geq 2$

証明：背理法



折畳 (おりたたみ, folding)

20/45

無向グラフ $G = (V, E)$, 頂点 $v \in V$, 最大独立集合 S

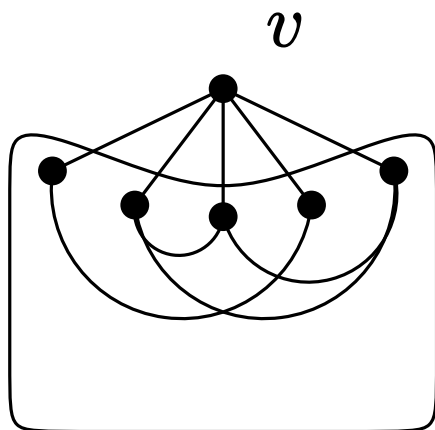
性質：折畳

仮定： $N(v)$ の異なる 3 頂点の間に辺が 1 つはある

G のどの最大独立集合も v を含まない

結論： $|S \cap N(v)| = 2$

証明： 前のページの補題より



折畳 (おりたたみ, folding)

20/45

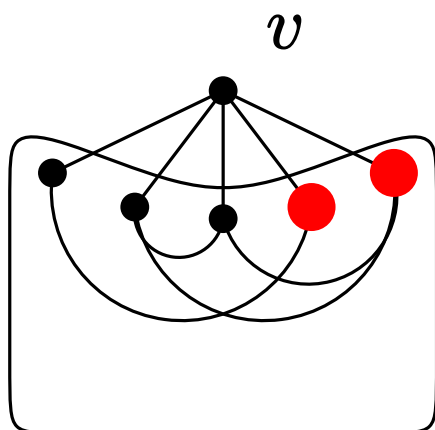
無向グラフ $G = (V, E)$, 頂点 $v \in V$, 最大独立集合 S

性質：折畳

仮定： $N(v)$ の異なる 3 頂点の間に辺が 1 つはある
 G のどの最大独立集合も v を含まない

結論： $|S \cap N(v)| = 2$

証明： 前のページの補題より



補題： 近傍から 2 頂点

仮定： $v \in V$

G のどの最大独立集合も v を含まない

結論： S が G の最大独立集合 $\Rightarrow |S \cap N(v)| \geq 2$

折畳 (おりたたみ, folding)

20/45

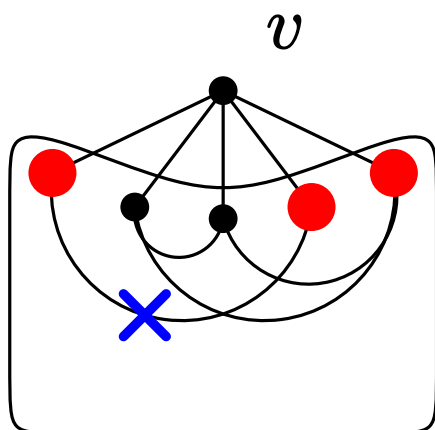
無向グラフ $G = (V, E)$, 頂点 $v \in V$, 最大独立集合 S

性質：折畳

仮定： $N(v)$ の異なる 3 頂点の間に辺が 1 つはある
 G のどの最大独立集合も v を含まない

結論： $|S \cap N(v)| = 2$

証明： 前のページの補題より



補題：近傍から 2 頂点

仮定： $v \in V$

G のどの最大独立集合も v を含まない

結論： S が G の最大独立集合 $\Rightarrow |S \cap N(v)| \geq 2$

アルゴリズム 折畳(G, v)

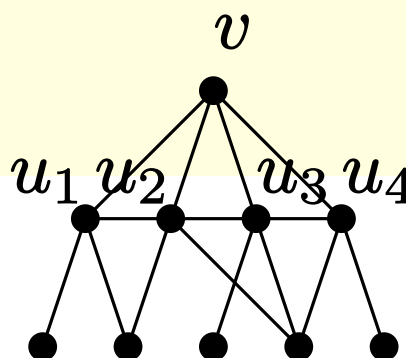
assert $N(v)$ の異なる 3 頂点の間に辺が 1 つはある

1. $\{u_1, u_2, \dots, u_d\} = N(v)$
2. すべての非辺 $\{u_i, u_j\} \subseteq N(v)$ に対して
 - 頂点 $u_i u_j$ を追加
 - 頂点 $u_i u_j$ と $N(u_i) \cup N(u_j)$ の各頂点の間に辺を追加
3. 追加したすべての頂点間に辺を追加
4. $N[v]$ の頂点をすべて削除
5. できあがったグラフを出力

アルゴリズム 折畳(G, v)

assert $N(v)$ の異なる 3 頂点の間に辺が 1 つはある

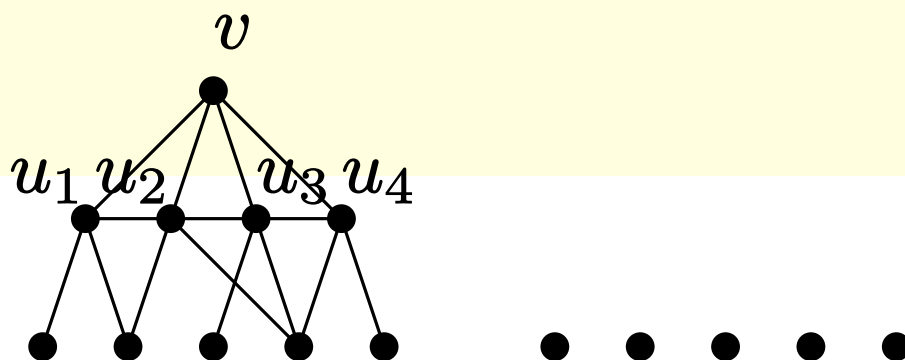
1. $\{u_1, u_2, \dots, u_d\} = N(v)$
2. すべての非辺 $\{u_i, u_j\} \subseteq N(v)$ に対して
 - 頂点 $u_i u_j$ を追加
 - 頂点 $u_i u_j$ と $N(u_i) \cup N(u_j)$ の各頂点の間に辺を追加
3. 追加したすべての頂点間に辺を追加
4. $N[v]$ の頂点をすべて削除
5. できあがったグラフを出力



アルゴリズム 折畳(G, v)

assert $N(v)$ の異なる 3 頂点の間に辺が 1 つはある

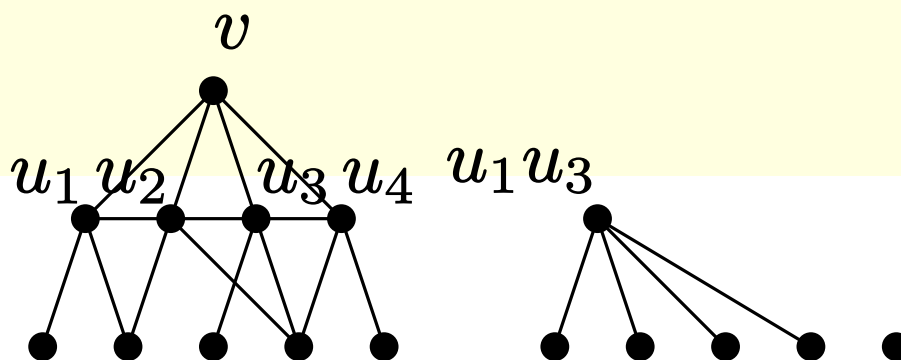
1. $\{u_1, u_2, \dots, u_d\} = N(v)$
2. すべての非辺 $\{u_i, u_j\} \subseteq N(v)$ に対して
 - 頂点 $u_i u_j$ を追加
 - 頂点 $u_i u_j$ と $N(u_i) \cup N(u_j)$ の各頂点の間に辺を追加
3. 追加したすべての頂点間に辺を追加
4. $N[v]$ の頂点をすべて削除
5. できあがったグラフを出力



アルゴリズム 折畳(G, v)

assert $N(v)$ の異なる 3 頂点の間に辺が 1 つはある

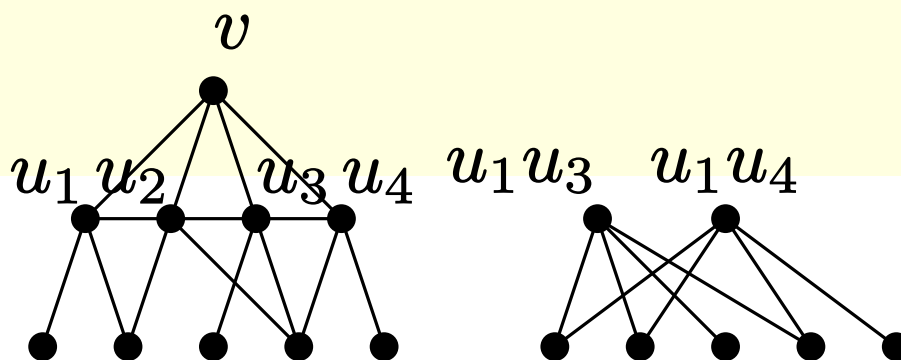
1. $\{u_1, u_2, \dots, u_d\} = N(v)$
2. すべての非辺 $\{u_i, u_j\} \subseteq N(v)$ に対して
 - 頂点 $u_i u_j$ を追加
 - 頂点 $u_i u_j$ と $N(u_i) \cup N(u_j)$ の各頂点の間に辺を追加
3. 追加したすべての頂点間に辺を追加
4. $N[v]$ の頂点をすべて削除
5. できあがったグラフを出力



アルゴリズム 折畳(G, v)

assert $N(v)$ の異なる 3 頂点の間に辺が 1 つはある

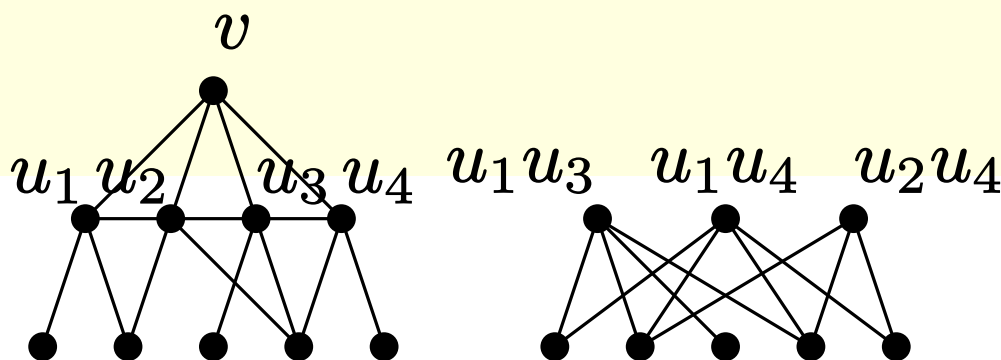
1. $\{u_1, u_2, \dots, u_d\} = N(v)$
2. すべての非辺 $\{u_i, u_j\} \subseteq N(v)$ に対して
 - 頂点 $u_i u_j$ を追加
 - 頂点 $u_i u_j$ と $N(u_i) \cup N(u_j)$ の各頂点の間に辺を追加
3. 追加したすべての頂点間に辺を追加
4. $N[v]$ の頂点をすべて削除
5. できあがったグラフを出力



アルゴリズム 折畳(G, v)

assert $N(v)$ の異なる 3 頂点の間に辺が 1 つはある

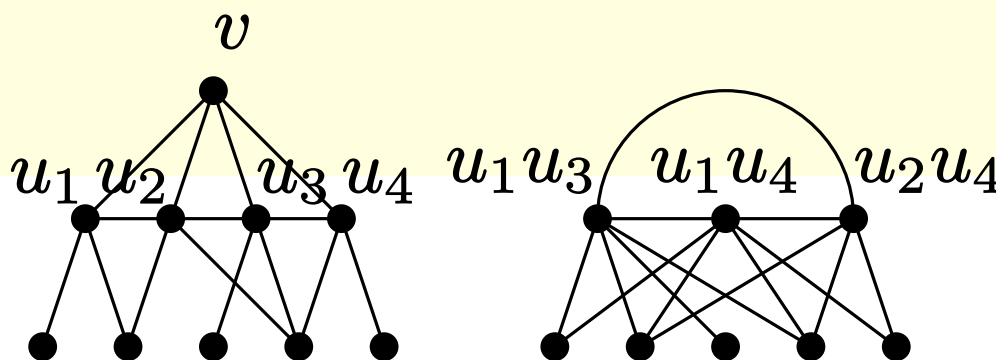
1. $\{u_1, u_2, \dots, u_d\} = N(v)$
2. すべての非辺 $\{u_i, u_j\} \subseteq N(v)$ に対して
 - 頂点 $u_i u_j$ を追加
 - 頂点 $u_i u_j$ と $N(u_i) \cup N(u_j)$ の各頂点の間に辺を追加
3. 追加したすべての頂点間に辺を追加
4. $N[v]$ の頂点をすべて削除
5. できあがったグラフを出力



アルゴリズム 折畳(G, v)

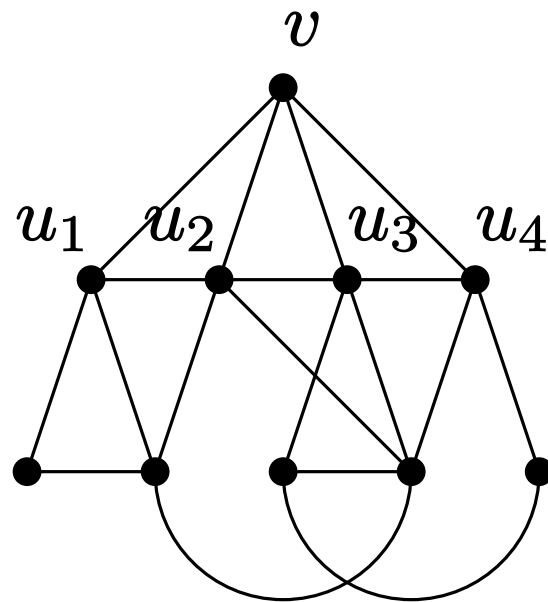
assert $N(v)$ の異なる 3 頂点の間に辺が 1 つはある

1. $\{u_1, u_2, \dots, u_d\} = N(v)$
2. すべての非辺 $\{u_i, u_j\} \subseteq N(v)$ に対して
 - 頂点 $u_i u_j$ を追加
 - 頂点 $u_i u_j$ と $N(u_i) \cup N(u_j)$ の各頂点の間に辺を追加
3. 追加したすべての頂点間に辺を追加
4. $N[v]$ の頂点をすべて削除
5. できあがったグラフを出力



観察

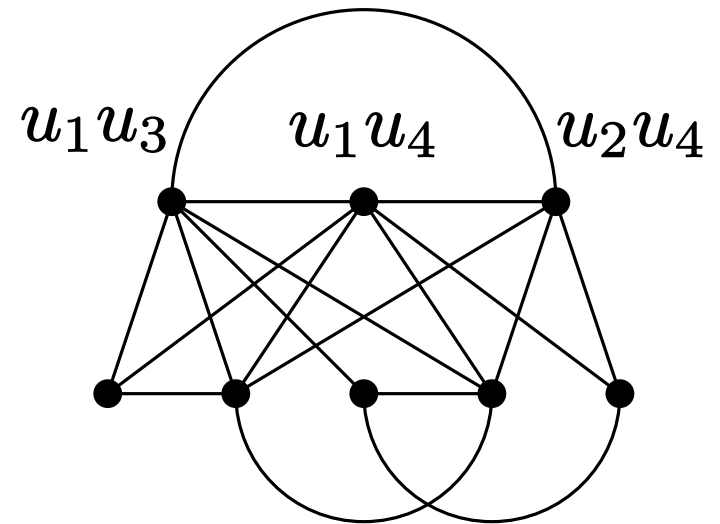
折畳操作で得られたグラフの 任意の最大独立集合 から
折畳操作を行う前のグラフの 任意の最大独立集合 が
復元できる



折畳操作

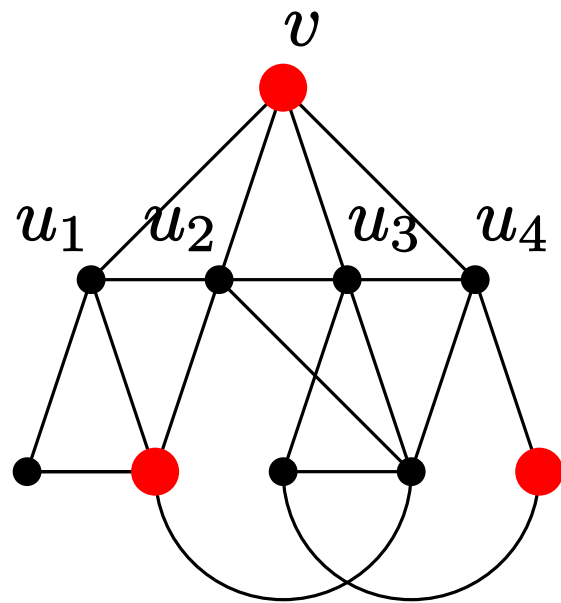


復元操作



観察

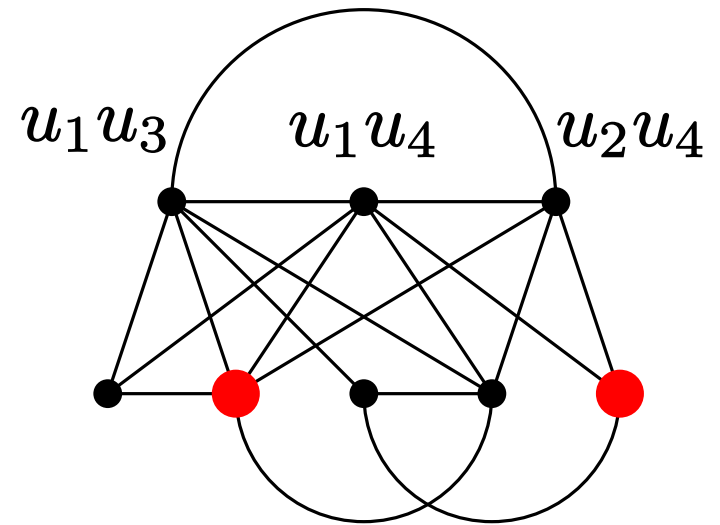
折畳操作で得られたグラフの 任意の最大独立集合 から
折畳操作を行う前のグラフの 任意の最大独立集合 が
復元できる



折畳操作

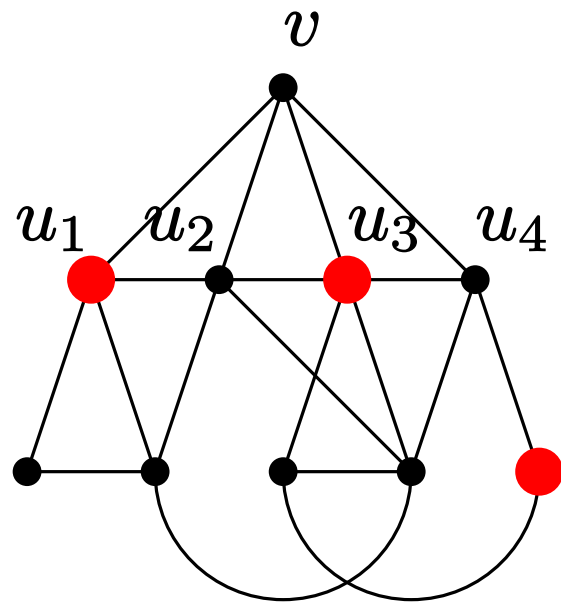


復元操作



観察

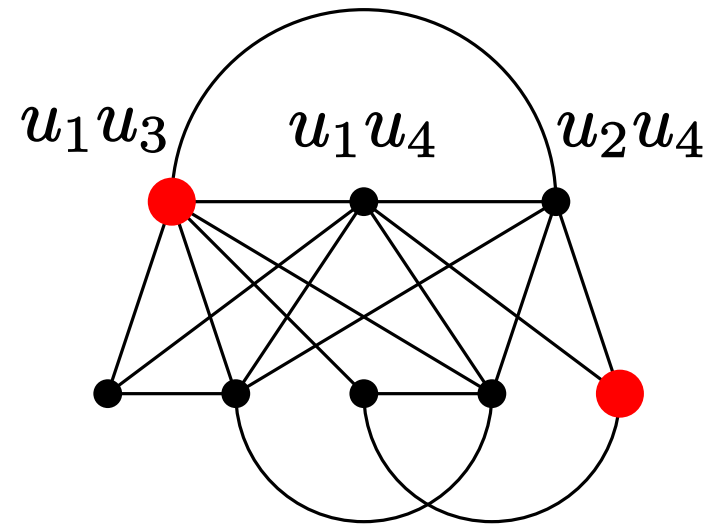
折畳操作で得られたグラフの 任意の最大独立集合 から
折畳操作を行う前のグラフの 任意の最大独立集合 が
復元できる



折畳操作

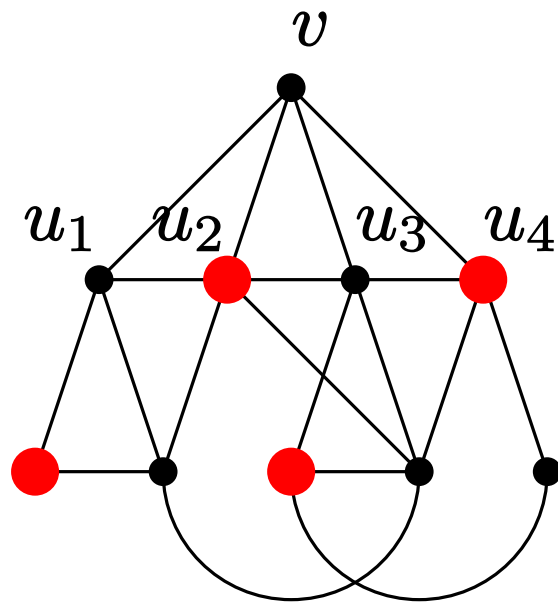


復元操作



観察

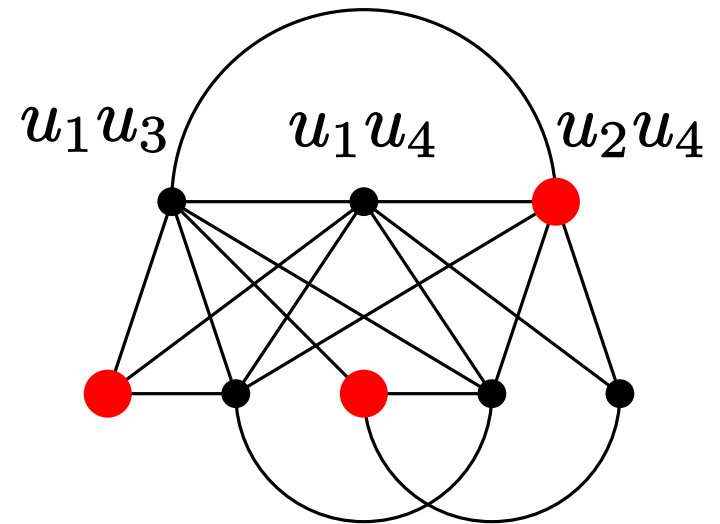
折畳操作で得られたグラフの 任意の最大独立集合 から
折畳操作を行う前のグラフの 任意の最大独立集合 が
復元できる



折畳操作



復元操作



アルゴリズム 折畳規則(G)

1. 折畳操作を行える頂点 v がある限り反復
 - $G = \text{折畳}(G, v)$
2. G を出力

ここまでの議論の帰結

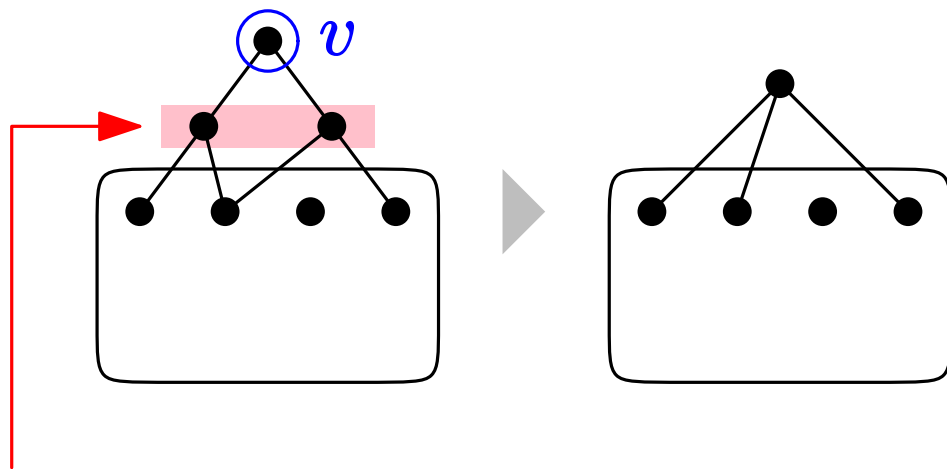
折畳規則(G) の最大独立集合から
 G の最大独立集合が (多項式時間で) 得られる

アルゴリズム 折畳規則(G)

1. 折畳操作を行える頂点 v がある限り反復

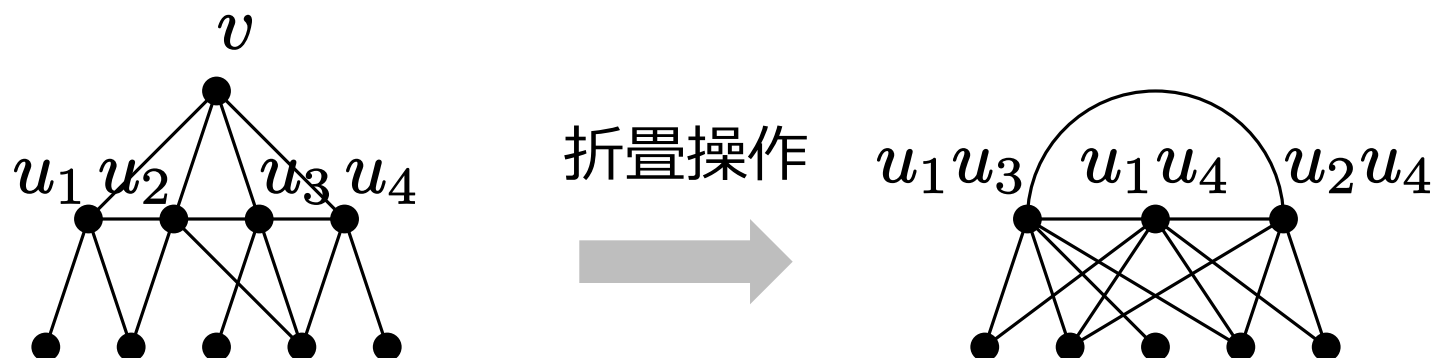
- $G = \text{折畳}(G, v)$

2. G を出力



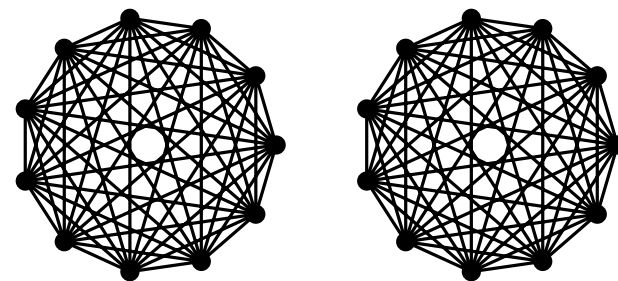
隣接していない場合

注：必ず頂点数が減少

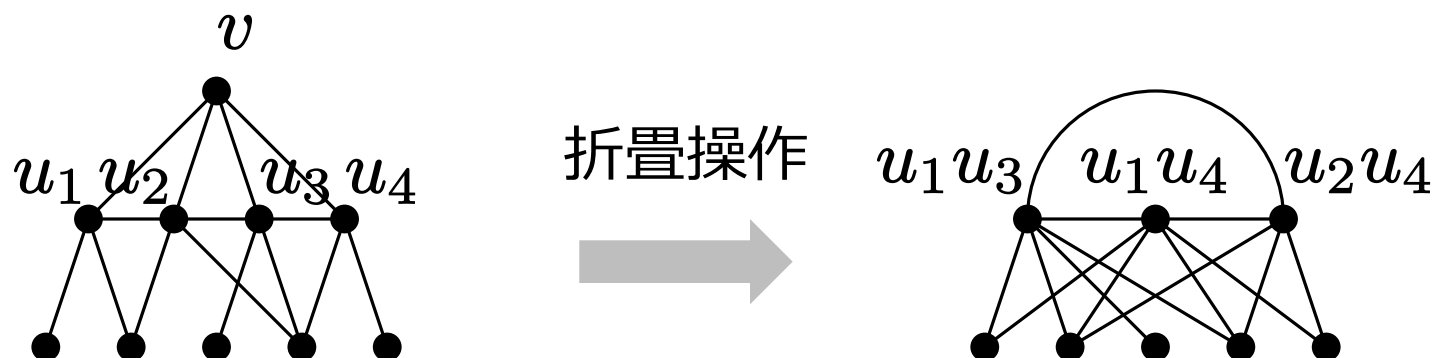


取り除いた頂点の数 $= 1 + \deg(v)$

付け加える頂点の数 $\leq \lceil \deg(v)/2 \rceil \lfloor \deg(v)/2 \rfloor$



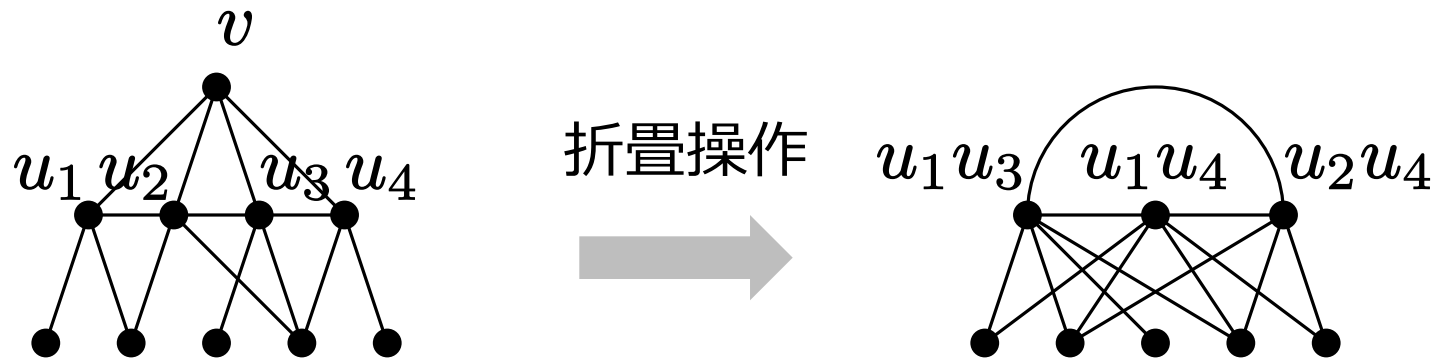
(Mantel の定理)



取り除いた頂点の数 = $1 + \deg(v)$

付け加える頂点の数 $\leq \lceil \deg(v)/2 \rceil \lfloor \deg(v)/2 \rfloor$

d	$1 + d$	$\lceil d/2 \rceil \lfloor d/2 \rfloor$
2	3	1
3	4	2
4	5	4
5	6	6
6	7	9



取り除いた頂点の数 = $1 + \deg(v)$

付け加える頂点の数 $\leq \lceil \deg(v)/2 \rceil \lfloor \deg(v)/2 \rfloor$

d	$1 + d$	$\lceil d/2 \rceil \lfloor d/2 \rfloor$
2	3	1
3	4	2
4	5	4
5	6	6
6	7	9

$\deg(v) \leq 4$ のときのみ
折畳操作を行う

アルゴリズム 折畳規則(G)

1. 折畳操作を行える頂点 v がある限り反復

- $G = \text{折畳}(G, v)$

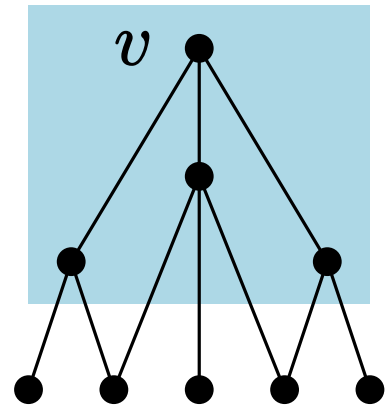
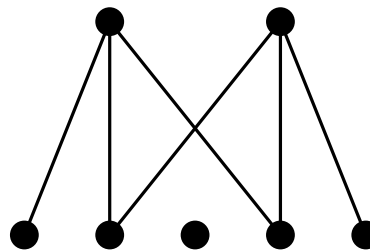
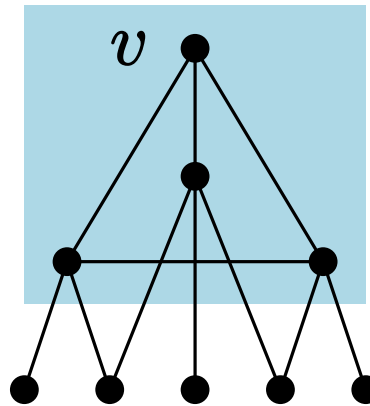
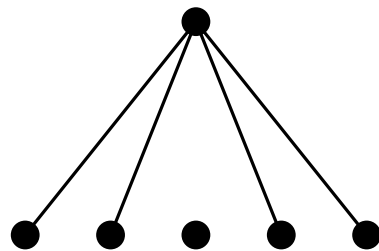
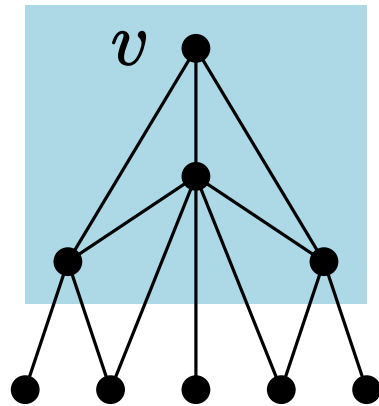
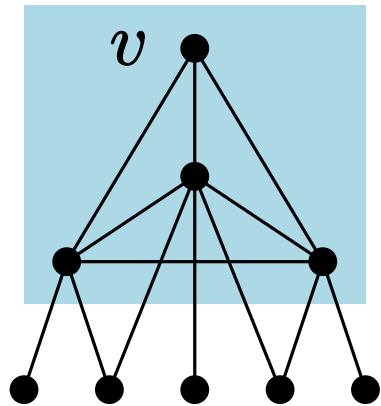
2. G を出力

- 
- $N(v)$ の異なる 3 頂点の間に辺が 1 つはある
 - $\deg(v) \leq 4$

アルゴリズム $A'(G)$

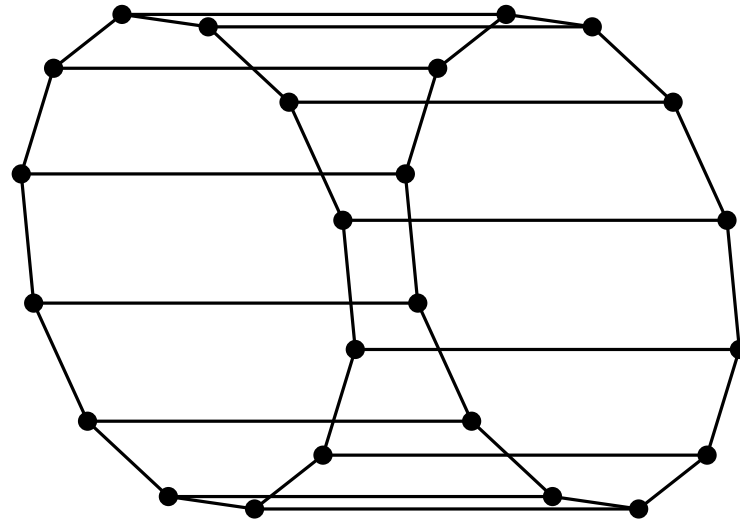
1. 優越規則と折畳規則を可能な限り G に適用
2. $v = G$ の次数最大の頂点
3. 次の2つの大きいほうを出力
 - $A'(G - N[v])$ の出力を復元したもの $\cup \{v\}$
 - $A'(G - \{v\})$ の出力を復元したもの

Step 1 の終了後, 最小次数 ≥ 3



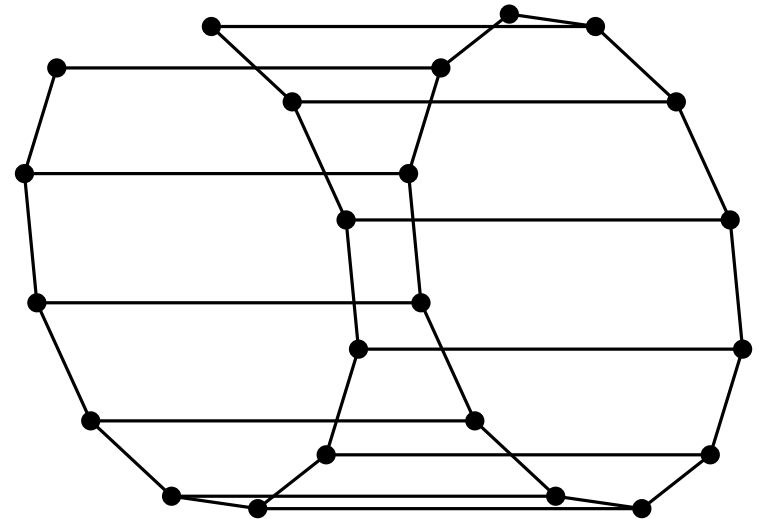
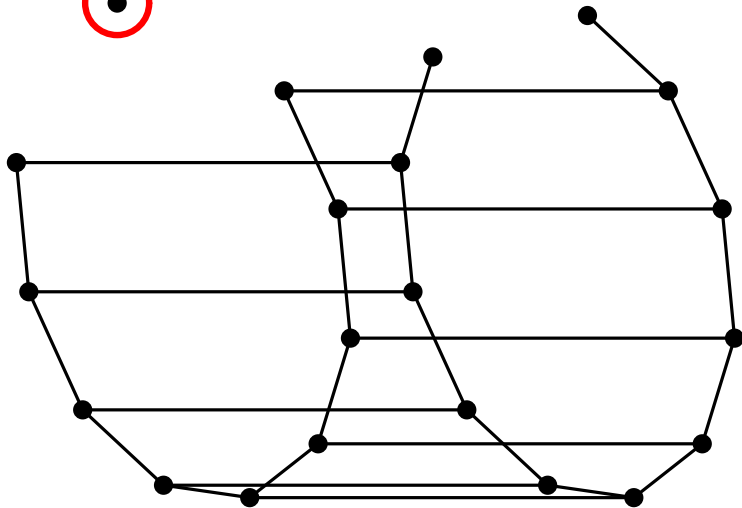
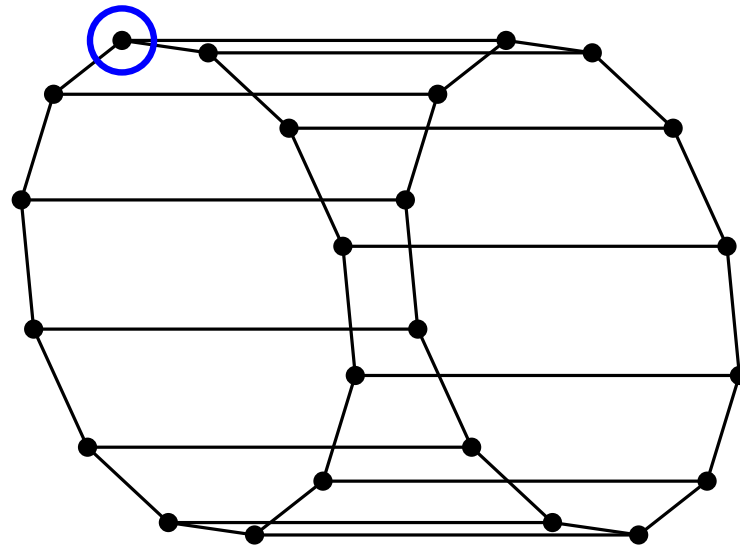
最小次数 3 で 困った場合

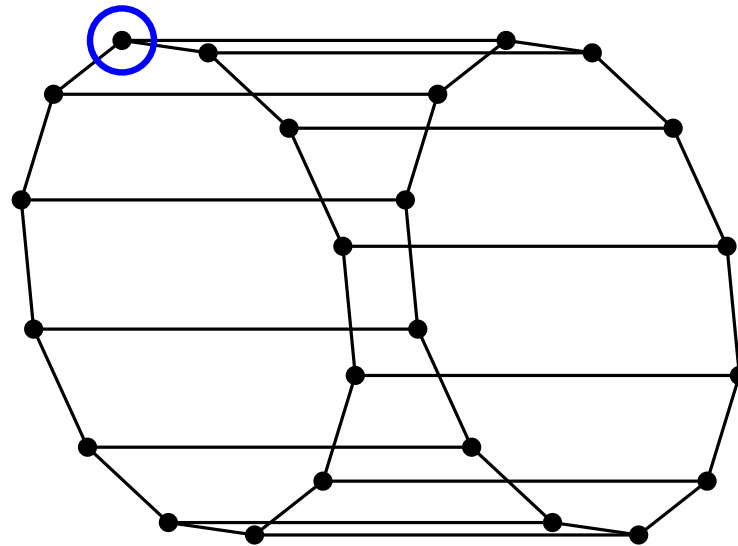
29/45



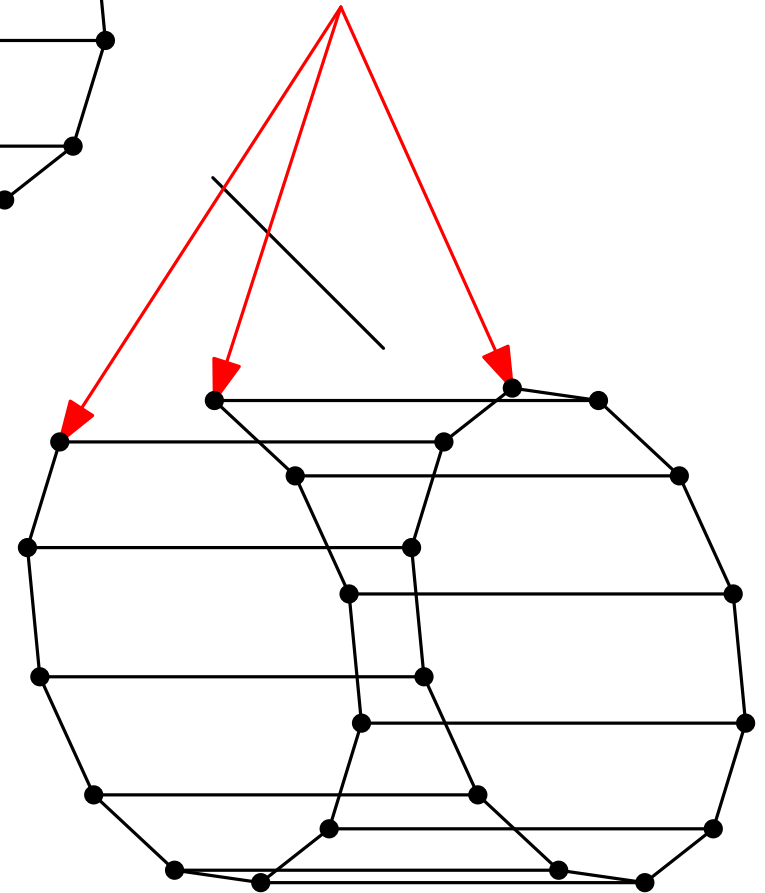
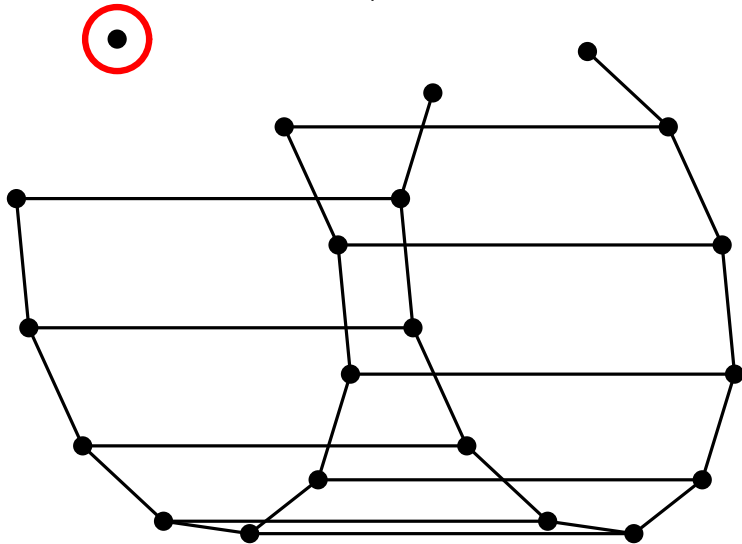
最小次数3で困った場合

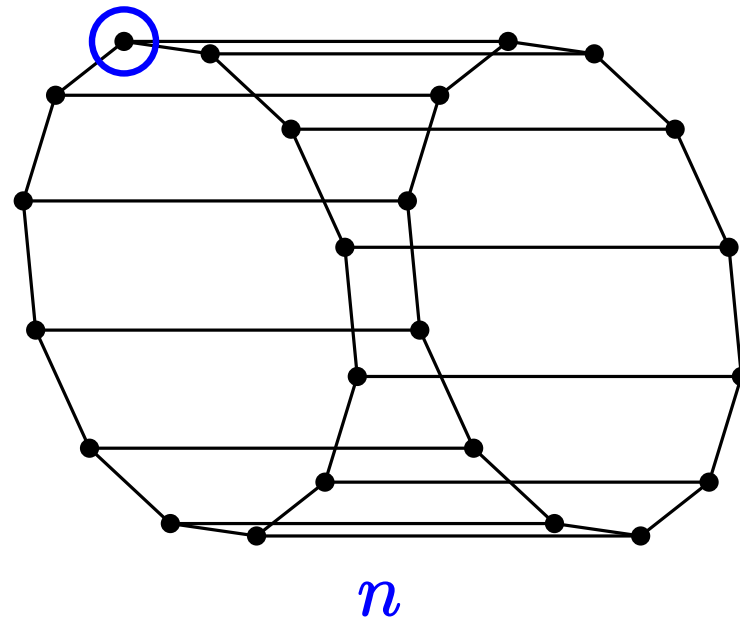
29/45



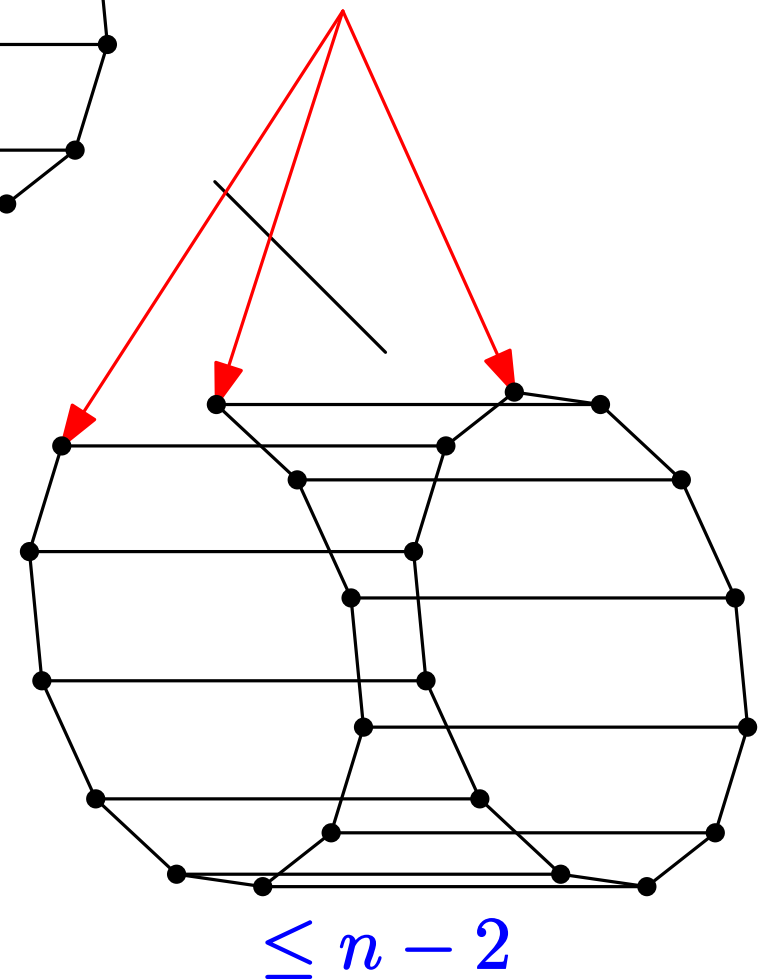
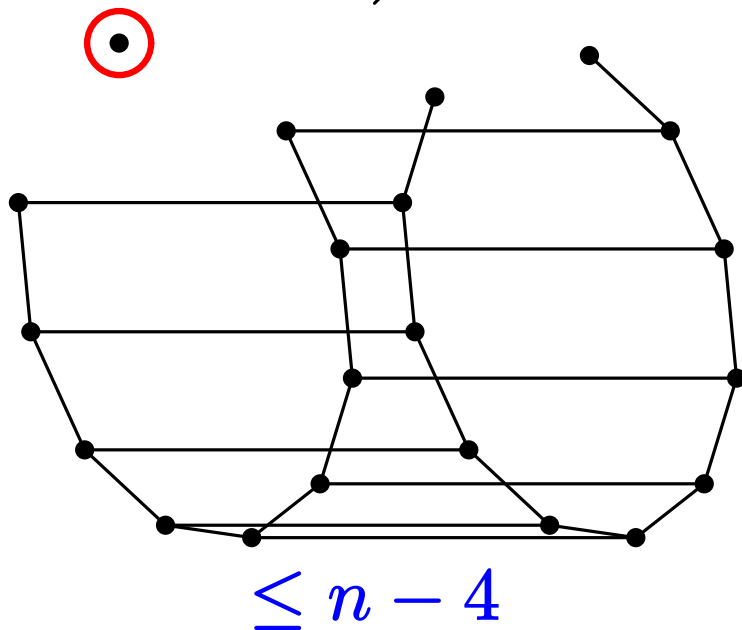


次数は 2
∴ 前処理を適用可





次数は2
∴ 前処理を適用可



アルゴリズム $A'(G)$

1. 優越規則と折畳規則を可能な限り G に適用
2. $v = G$ の次数最大の頂点 $\leftarrow \deg(v) = \text{最大次数}$
 $\geq \text{最小次数} \geq 3$
3. 次の2つの大きいほうを出力
 - $A'(G - N[v])$ の出力を復元したもの $\cup \{v\}$
 - $A'(G - \{v\})$ の出力を復元したもの

計算量を最大にする頂点数 n のグラフを G とすると

Step 2 で $\deg(v) \geq 4$ のとき

$$\begin{aligned} \bullet \quad T(n) = T(G) &= T(G - N[v]) + T(G - \{v\}) \\ &\leq T(n - 5) + T(n - 1) \end{aligned}$$

アルゴリズム $A'(G)$

1. 優越規則と折畳規則を可能な限り G に適用
2. $v = G$ の次数最大の頂点 $\leftarrow \deg(v) = \text{最大次数}$
 $\geq \text{最小次数} \geq 3$
3. 次の2つの大きいほうを出力
 - $A'(G - N[v])$ の出力を復元したもの $\cup \{v\}$
 - $A'(G - \{v\})$ の出力を復元したもの

計算量を最大にする頂点数 n のグラフを G とすると

Step 2 で $\deg(v) = 3$ のとき

$$\begin{aligned} \bullet \quad T(n) = T(G) &= T(G - N[v]) + T(\text{前処理}(G - \{v\})) \\ &\leq T(n - 4) + T(n - 2) \end{aligned}$$

考える漸化式

$$T(n) \leq \max\{T(n-5) + T(n-1), T(n-4) + T(n-2)\}$$

どちらが最大値を取るかわからない

考える漸化式

$$T(n) \leq \max\{T(n-5) + T(n-1), T(n-4) + T(n-2)\}$$

どちらが最大値を取るかわからない

$T(n) \leq T(n-5) + T(n-1)$ に対して

- 特性方程式は $x^5 = 1 + x^4 \leadsto$ 正の実数解 = 1.3247

$T(n) \leq T(n-4) + T(n-2)$ に対して

- 特性方程式は $x^4 = 1 + x^2 \leadsto$ 正の実数解 = 1.2720

考える漸化式

$$T(n) \leq \max\{T(n-5) + T(n-1), T(n-4) + T(n-2)\}$$

どちらが最大値を取るかわからない

$T(n) \leq T(n-5) + T(n-1)$ に対して

- 特性方程式は $x^5 = 1 + x^4 \leadsto$ 正の実数解 = 1.3247

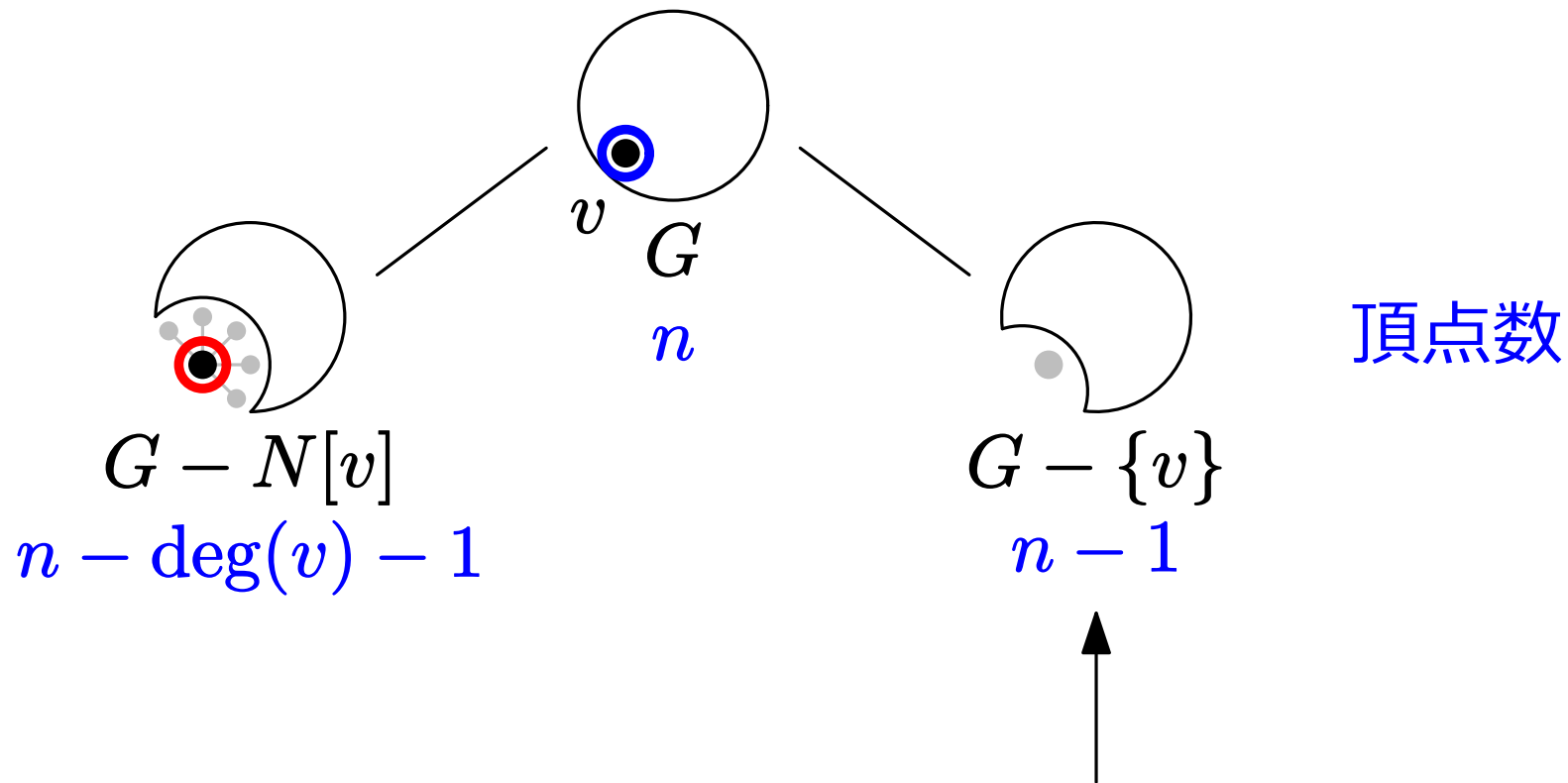
$T(n) \leq T(n-4) + T(n-2)$ に対して

- 特性方程式は $x^4 = 1 + x^2 \leadsto$ 正の実数解 = 1.2720

結論 : アルゴリズム A' の計算量は $O^*(1.3247^n)$

注 : アルゴリズム A の計算量は $O^*(1.3803^n)$

1. 分枝アルゴリズムの高速化：前処理
 2. **分枝アルゴリズムの高速化：分枝規則の精緻化**
 3. 付録：メモ化 (記憶化)
-

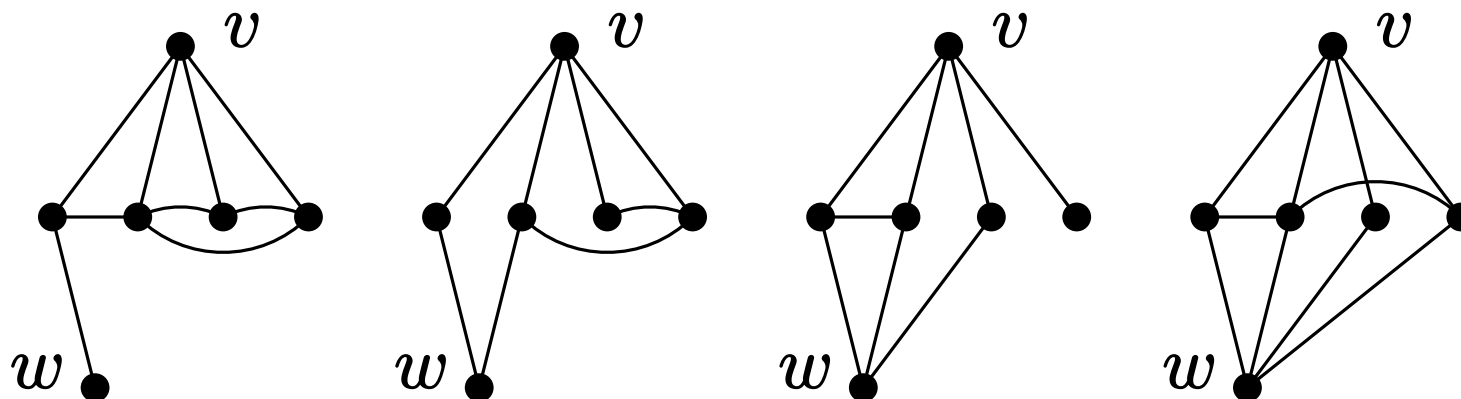


目標 : 頂点数をもっと大きく減らしたい

無向グラフ $G = (V, E)$, 頂点 $v, w \in V$, $\{v, w\} \notin E$

定義：鏡像 (mirror)

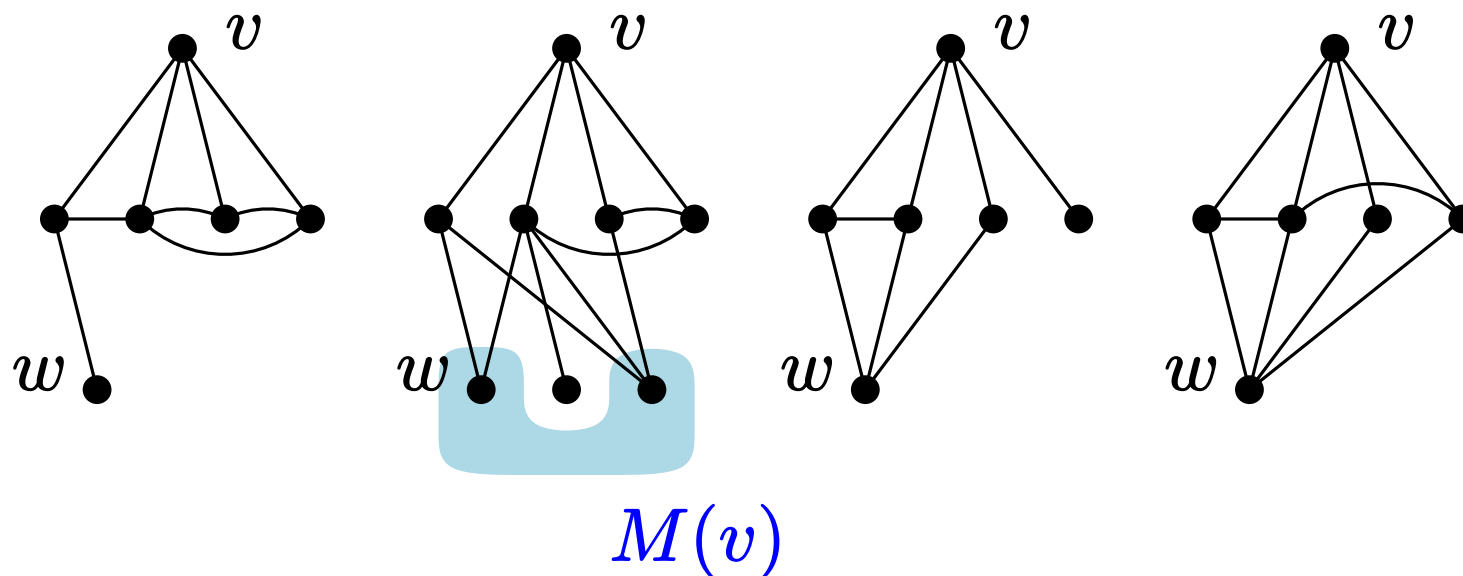
w が v の **鏡像** であるとは, $N(v) \cap N(w) \neq \emptyset$ で $N(v) - N(w)$ の任意の2頂点が隣接していること



無向グラフ $G = (V, E)$, 頂点 $v, w \in V, \{v, w\} \notin E$

定義：鏡像 (mirror)

w が v の **鏡像** であるとは, $N(v) \cap N(w) \neq \emptyset$ で $N(v) - N(w)$ の任意の2頂点が隣接していること



記法： $M(v) = v$ の鏡像全体の集合

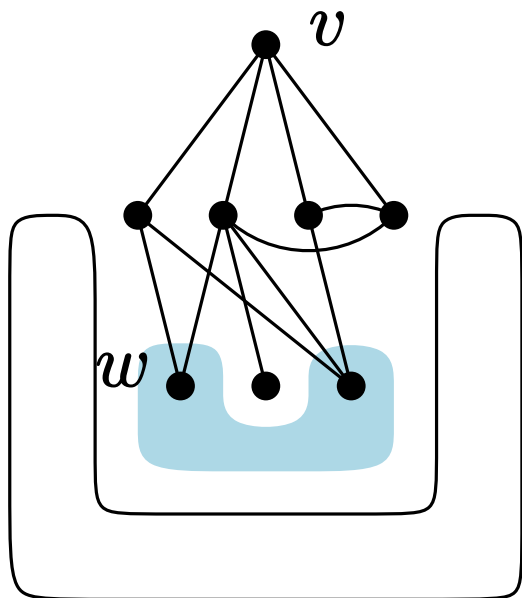
無向グラフ $G = (V, E)$, 頂点 $v, w \in V, \{v, w\} \notin E$

性質：鏡像と最大独立集合

仮定： w は v の鏡像

G のどの最大独立集合も v を含まない

結論： G のどの最大独立集合も w を含まない



無向グラフ $G = (V, E)$, 頂点 $v, w \in V, \{v, w\} \notin E$

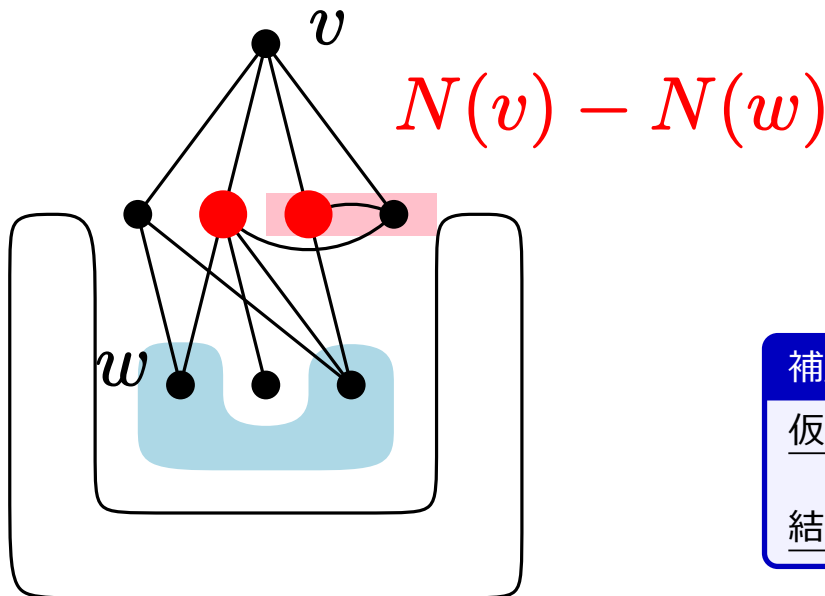
性質：鏡像と最大独立集合

仮定： w は v の鏡像

G のどの最大独立集合も v を含まない

結論： G のどの最大独立集合も w を含まない

証明：補題 (近傍から 2 頂点) を使う



補題：近傍から 2 頂点

仮定： $v \in V$

G のどの最大独立集合も v を含まない

結論： S が G の最大独立集合 $\Rightarrow |S \cap N(v)| \geq 2$

無向グラフ $G = (V, E)$, 頂点 $v, w \in V, \{v, w\} \notin E$

性質：鏡像と最大独立集合

仮定： w は v の鏡像

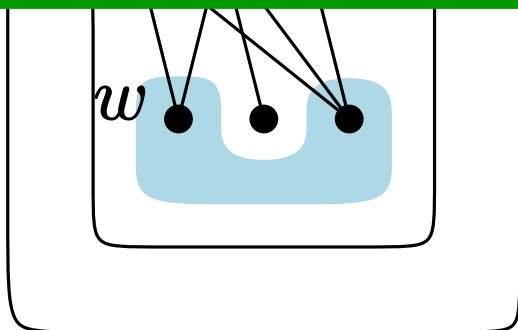
G のどの最大独立集合も v を含まない

結論： G のどの最大独立集合も w を含まない

証明・補題 (近傍から 2 頂点) を使う

帰結

v を含まない最大独立集合を探すときには
 $M(v)$ もグラフから取り除いてよい



補題：近傍から 2 頂点

仮定： $v \in V$

G のどの最大独立集合も v を含まない

結論： S が G の最大独立集合 $\Rightarrow |S \cap N(v)| \geq 2$

アルゴリズム $A''(G)$

1. 優越規則と折畳規則を可能な限り G に適用
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
 - $A''(G - N[v])$ の出力を復元したもの $\cup \{v\}$
 - $A''(G - M(v) - \{v\})$ の出力を復元したもの

計算量の解析は次回

注：マイルストーンのみ記載

時間	領域		
$O^*(1.2599^n)$	多項式	Tarjan, Trojanowski	'77
$O^*(1.2278^n)$	多項式	Robson	'86
$O^*(1.2109^n)$	指数	Robson	'86
($O^*(1.1893^n)$)	指数	Robson	'01)
$O^*(1.2202^n)$	多項式	Fomin, Grandoni, Kratsch	'09
$O^*(1.1996^n)$	多項式	Xiao, Nagamochi	'17

注：マイルストーンのみ記載

時間	領域		
$O^*(1.2599^n)$	多項式	Tarjan, Trojanowski	'77
$O^*(1.2278^n)$	多項式	Robson	'86
$O^*(1.2109^n)$	指数	Robson	'86
($O^*(1.1893^n)$)	指数	Robson	'01)
$O^*(1.2202^n)$	多項式	Fomin, Grandoni, Kratsch	'09
$O^*(1.1996^n)$	多項式	Xiao, Nagamochi	'17

最初の論文

注：マイルストーンのみ記載

時間	領域		
$O^*(1.2599^n)$	多項式	Tarjan, Trojanowski	'77
$O^*(1.2278^n)$	多項式	Robson	'86
$O^*(1.2109^n)$	指数	Robson	'86
($O^*(1.1893^n)$)	指数	Robson	'01)
$O^*(1.2202^n)$	多項式	Fomin, Grandoni, Kratsch	'09
$O^*(1.1996^n)$	多項式	Xiao, Nagamochi	'17

現在の理論上最速

注：マイルストーンのみ記載

時間	領域		
$O^*(1.2599^n)$	多項式	Tarjan, Trojanowski	'77
$O^*(1.2278^n)$	多項式	Robson	'86
$O^*(1.2109^n)$	指数	Robson	'86
($O^*(1.1893^n)$	指数	Robson	'01)
$O^*(1.2202^n)$	多項式	Fomin, Grandoni, Kratsch	'09
$O^*(1.1996^n)$	多項式	Xiao, Nagamochi	'17

指数領域による改善 (メモ化)

注：マイルストーンのみ記載

時間	領域		
$O^*(1.2599^n)$	多項式	Tarjan, Trojanowski	'77
$O^*(1.2278^n)$	多項式	Robson	'86
$O^*(1.2109^n)$	指数	Robson	'86
($O^*(1.1893^n)$)	指数	Robson	'01)
$O^*(1.2202^n)$	多項式	Fomin, Grandoni, Kratsch	'09
$O^*(1.1996^n)$	多項式	Xiao, Nagamochi	'17

計算量評価の新たな手法による改善 (測度統治法)

第2回以降3回

分枝アルゴリズム (branching algorithm) の設計と解析

第2回 (前回)

- 分枝アルゴリズムの基礎

第3回 (今回)

- 分枝アルゴリズムの改良法

第4回 (次回)

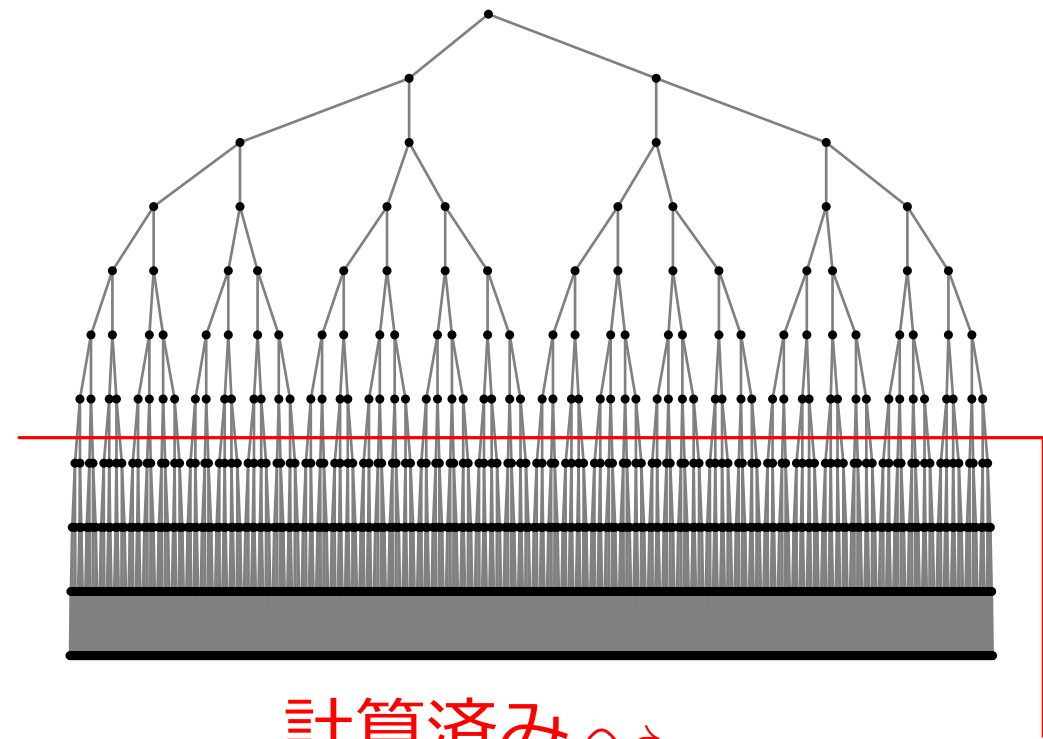
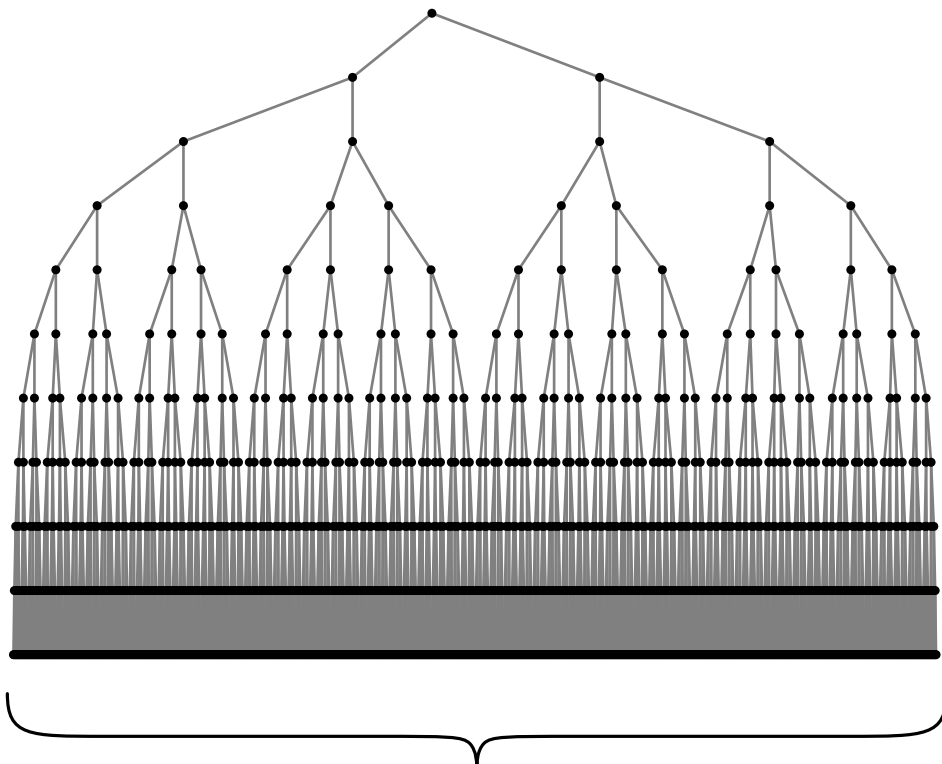
- 測度統治法 (measure & conquer) による改良

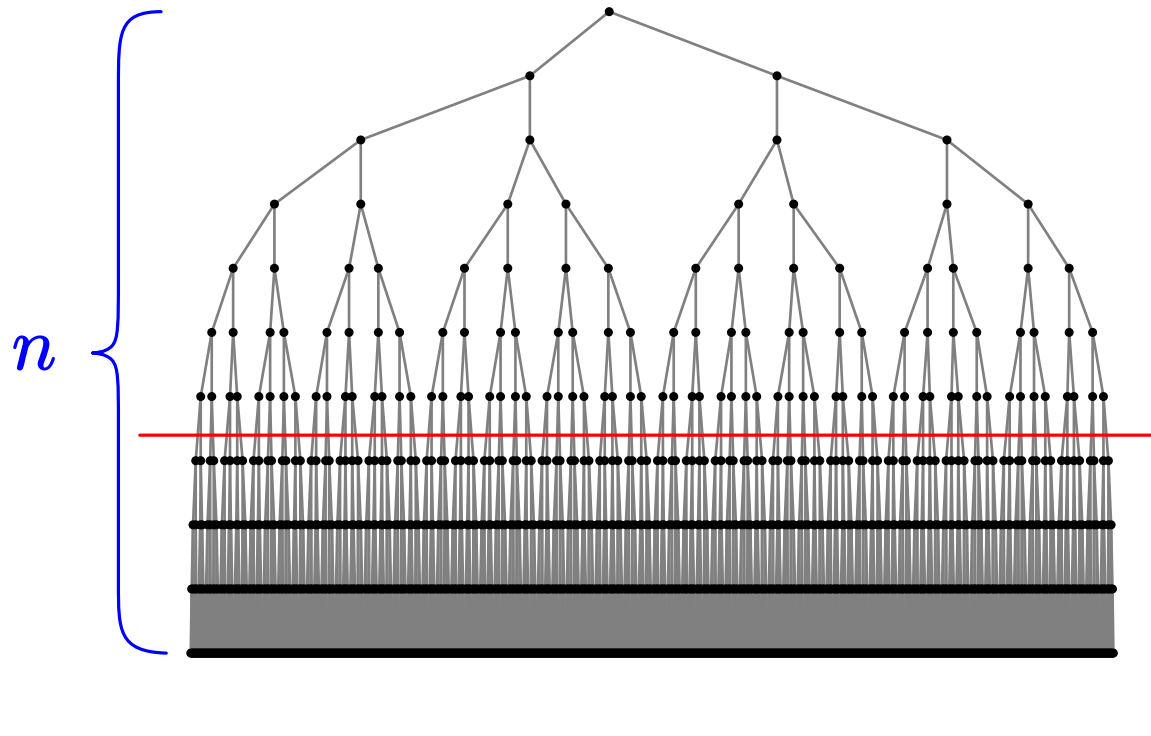
1. 分枝アルゴリズムの高速化：前処理
 2. 分枝アルゴリズムの高速化：分枝規則の精緻化
 3. **付録：メモ化 (記憶化)**
-

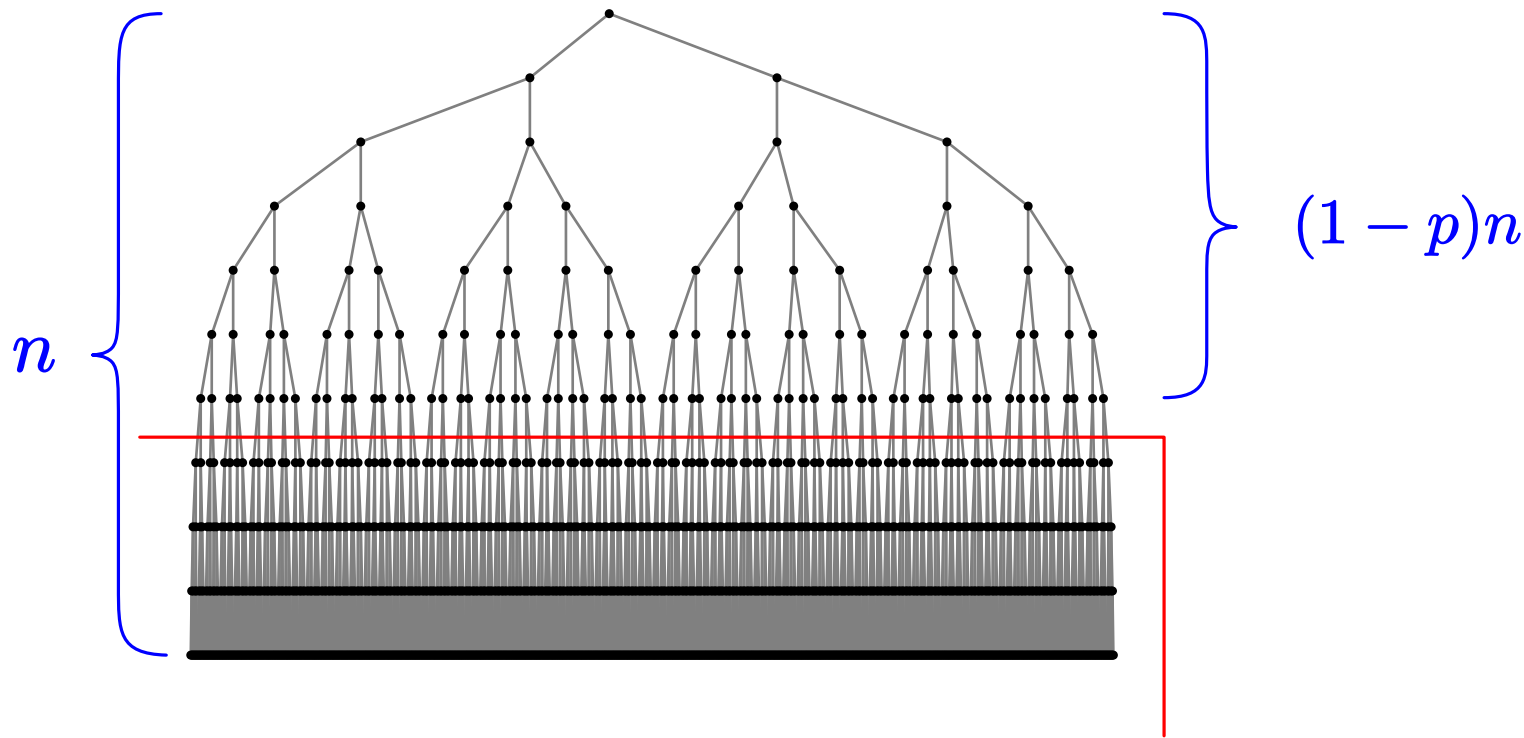
メモ化 (記憶化) の考え方

頂点数が小さい **すべて** のグラフに対する最大独立集合をあらかじめ計算して、覚えておく

メモ化 = memoization, 記憶化 = memorization





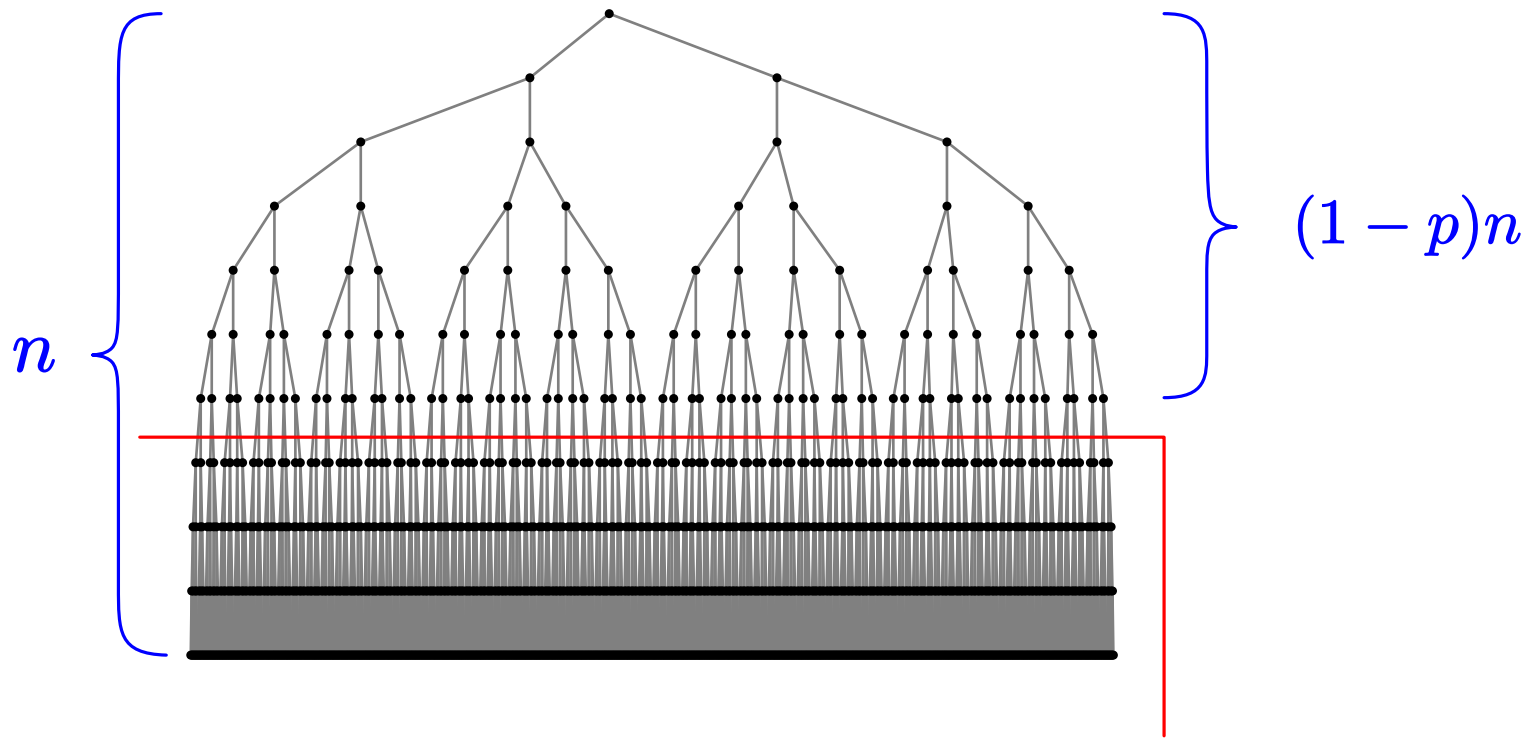


頂点数 pn 以下の部分グラフすべてに対して 覚える

- 頂点数 pn 以下の部分グラフの総数
≤ 要素数 pn 以下の部分集合の総数

$$\leq \sum_{i=0}^{pn} \binom{n}{i}$$

二項係数 (binomial coefficient)



$$\text{計算量} = O^* \left(\underbrace{\sum_{i=0}^{pn} \binom{n}{i}}_{\text{メモ化}} + \underbrace{2^{(1-p)n}}_{\text{しらみつぶし}} \right)$$

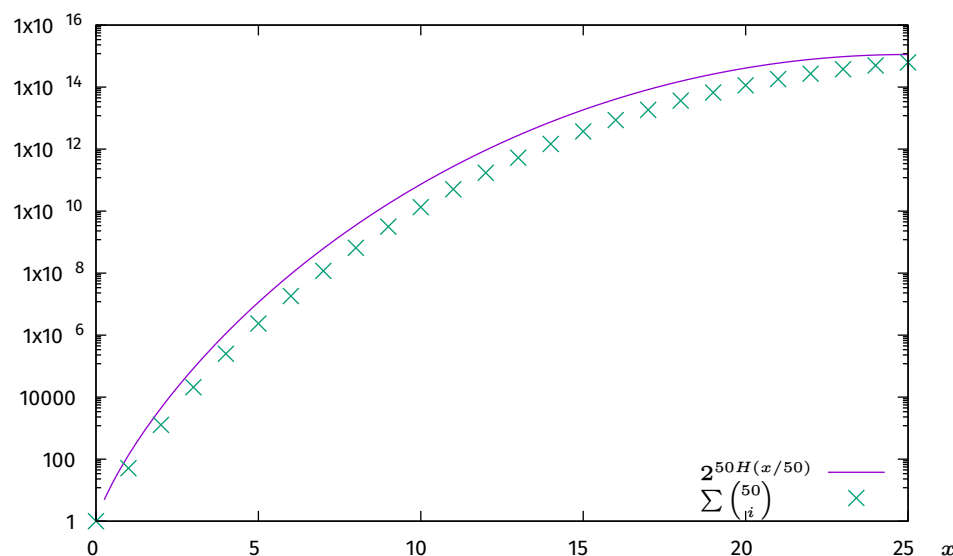
この2項が釣り合うように, p を定める

正整数 n , 実数 $p \in (0, 1/2]$

性質：二項係数の和に対する上界 (証明は省略)

$$\sum_{i=0}^{\lfloor pn \rfloor} \binom{n}{i} \leq 2^{nH(p)}$$

ただし, $H(p) = -p \log_2 p - (1 - p) \log_2 (1 - p)$
(二値エントロピー関数, binary entropy function)



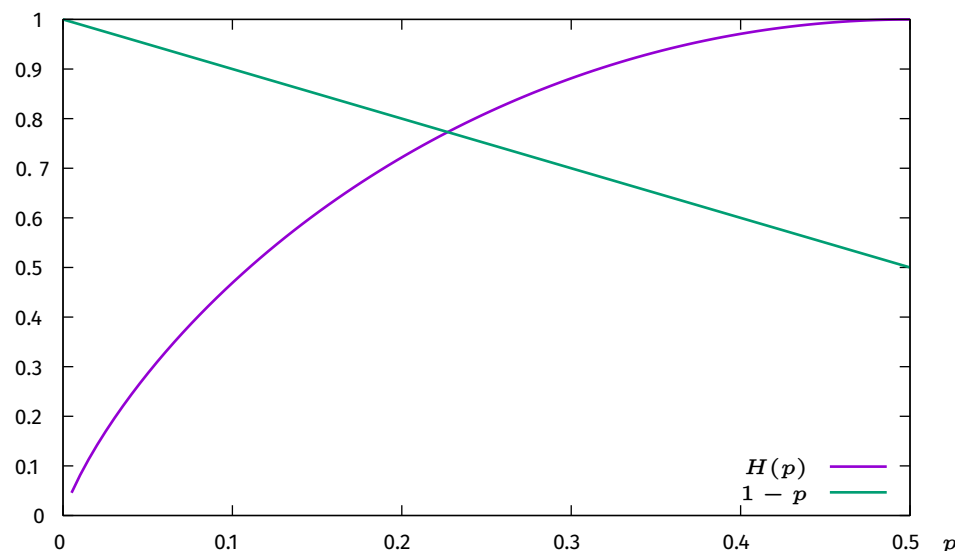
$$\begin{aligned}\text{計算量} &= O^* \left(\sum_{i=0}^{pn} \binom{n}{i} + 2^{(1-p)n} \right) \\ &= O^* \left(2^{nH(p)} + 2^{(1-p)n} \right)\end{aligned}$$

$$\text{計算量} = O^* \left(\sum_{i=0}^{pn} \binom{n}{i} + 2^{(1-p)n} \right)$$

$$= O^* \left(2^{nH(p)} + 2^{(1-p)n} \right)$$

2 項が釣り合う p は次を満たす

$$H(p) = 1 - p \quad \leadsto \quad p = 0.22709$$



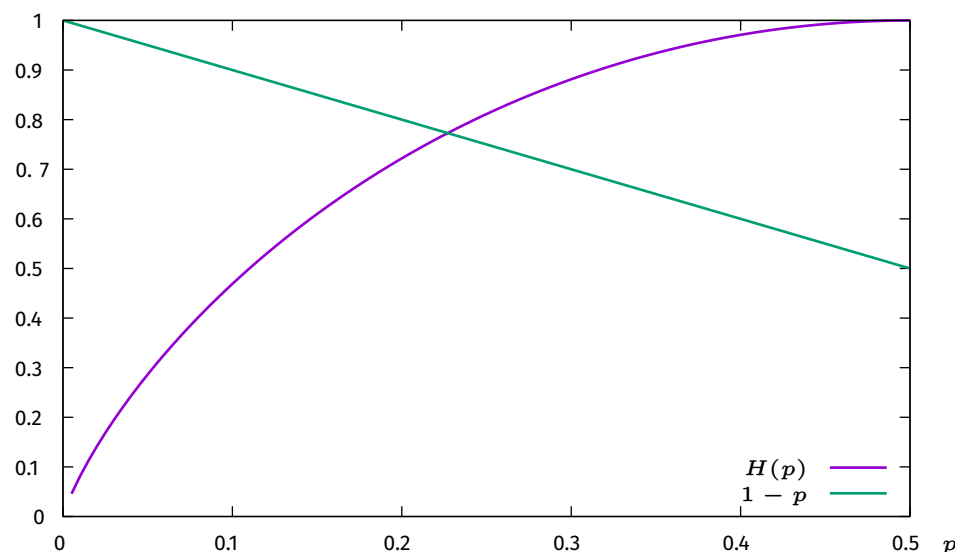
$$\begin{aligned}\text{計算量} &= O^* \left(\sum_{i=0}^{pn} \binom{n}{i} + 2^{(1-p)n} \right) \\ &= O^* \left(2^{nH(p)} + 2^{(1-p)n} \right)\end{aligned}$$

2 項が釣り合う p は次を満たす

$$H(p) = 1 - p \quad \leadsto \quad p = 0.22709$$

したがって,

$$\begin{aligned}\text{計算量} &= O^* \left(2^{(1-0.22709)n} \right) \\ &= O^* (1.7087^n)\end{aligned}$$



$$\begin{aligned}\text{計算量} &= O^* \left(\sum_{i=0}^{pn} \binom{n}{i} + 2^{(1-p)n} \right) \\ &= O^* \left(2^{nH(p)} + 2^{(1-p)n} \right)\end{aligned}$$

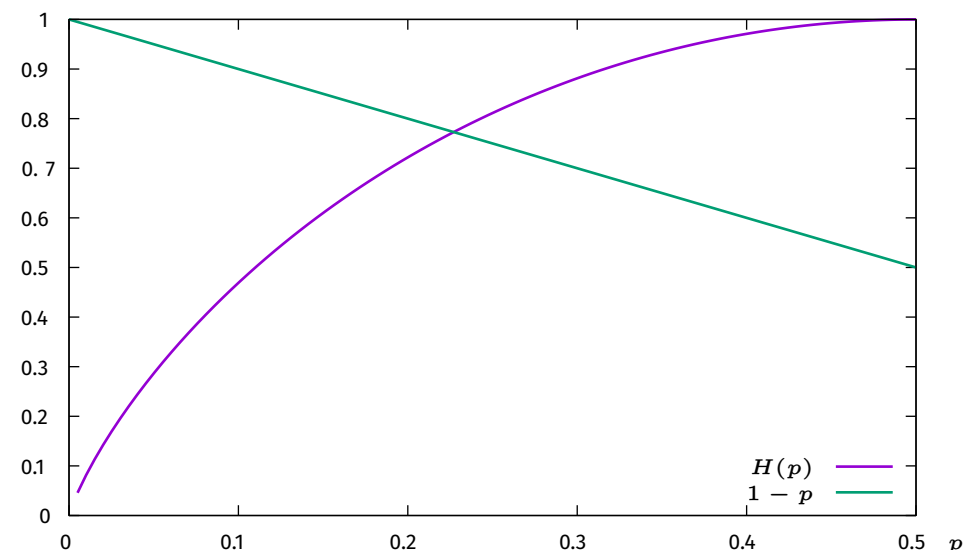
2 項が釣り合う p は次を満たす

$$H(p) = 1 - p \quad \leadsto \quad p = 0.22709$$

したがって,

$$\begin{aligned}\text{計算量} &= O^* \left(2^{(1-0.22709)n} \right) \\ &= O^* (1.7087^n)\end{aligned}$$

$$\begin{aligned}\text{領域} &= O^* \left(2^{nH(0.22709)} \right) \\ &= O^* (1.7087^n)\end{aligned}$$



$$\begin{aligned}\text{計算量} &= O^* \left(2^{nH(p)} + 1.3247^{(1-p)n} \right) \\ &= O^* \left(1.3247^{2.4650nH(p)} + 1.3247^{(1-p)n} \right)\end{aligned}$$

2 項が釣り合う p は次を満たす

$$2.4650H(p) = 1 - p \quad \leadsto \quad p = 0.072765$$

したがって,

$$\text{計算量} = O^* \left(1.3247^{(1-0.072765)n} \right) = O^*(1.2978^n)$$

$$\text{領域} = O^* \left(2^{nH(0.072765)} \right) = O^*(1.2978^n)$$