

離散最適化基礎論 第 13 回
アルゴリズム的問題解決：再考

岡本 吉央
okamotoy@uec.ac.jp

電気通信大学

2020 年 1 月 21 日

最終更新：2020 年 1 月 20 日 08:07

- ★ 休み (大学院入学式) (10/1)
- 1 アルゴリズム的問題解決と計算複雑性 (10/8)
- 2 速習 P vs NP 問題 (10/15)
- ★ 休み (祝日) (10/22)
- 3 充足可能性問題とその変種 (10/29)
- 4 グラフに関する問題 (1) : 部分集合の選択 (11/5)
- 5 グラフに関する問題 (2) : 経路の選択 (11/12)
- 6 集合族に関する問題 (1) : グラフとの関連 (11/19)
- 7 集合族に関する問題 (2) : 発展 (11/26)

- | | | |
|----|-----------------------|-------------|
| 8 | 数値が関わる問題 (1) : 2 分割問題 | (12/3) |
| 9 | 数値が関わる問題 (2) : 3 分割問題 | (12/10) |
| 10 | 平面性が関わる問題 | (12/17) |
| ★ | 冬期休業 | (12/24, 31) |
| 11 | 計算幾何学に関する問題 | (1/7) |
| 12 | 文字列に関する問題 | (1/14) |
| 13 | アルゴリズム的問題解決 : 再考 | (1/21) |
| 14 | 予備 | (1/28) |
| ★ | 休講 | (2/4) |
| ★ | 祝日のため休み | (2/11) |

今日の目標

NP 完全問題の解決法を考える

- ▶ 制限アプローチ
- ▶ 厳密アプローチ
- ▶ 近似アプローチ

- ① アルゴリズム的問題解決と P vs NP 問題：復習
- ② アルゴリズム的問題解決の進展と妥協点
- ③ 制限アプローチ
- ④ 厳密アプローチ
- ⑤ 近似アプローチ
- ⑥ 今日のまとめ と 全体のまとめ

問題解決の流れ (ポリアによる)

- | | |
|---------------|--------------|
| (1) 問題を理解すること | (understand) |
| (2) 計画をたてること | (plan) |
| (3) 計画を実行すること | (carry out) |
| (4) ふり返ってみること | (look back) |

ポリアの射程： 数学的問題解決

本講義の射程： アルゴリズム的問題解決

目標

上の「問題解決の流れ」を「アルゴリズム的問題解決」で行う

(1) 問題を理解すること

アルゴリズム的問題解決における「問題を理解すること」

アルゴリズムがすべきことを定めること

(直感的な言い方)

- 入力 : アルゴリズムが何を受け取るのか
- 出力 : アルゴリズムが何をもたらすのか
- 条件 : 出力が満たすべき性質は何なのか
- (評価) : 出力の良さをどのように測るのか

注釈 1

分野によっては、次のような用語を使うことがある

条件 = 制約 評価 = 目的, 目標

注釈 2

「出力の評価」と「アルゴリズムの評価」は別概念なので、区別する

(2) 計画をたてること

アルゴリズム的問題解決における「計画をたてること」

- 1 アルゴリズム設計の**方針**を立てること
- 2 アルゴリズムを設計すること

アルゴリズム設計の方針とは？

- ▶ 我々が目標とする「実行時間」と「メモリ消費量」の特定
- ▶ 我々が達成できる「実行時間」と「メモリ消費量」の特定
← この講義の内容

現実問題の付帯条件を細かく検討する必要がある

スケジューリングのような問題にも様々

- ▶ 中学校の時間割作成
教員数 30 名程度，学級数 15 程度，1 年に 1 回作成
計算にかけられる時間：1 日程度
- ▶ 銀行 ATM の現金補充計画作成
ATM 数 200 個程度，拠点数 20 個程度，1 年に 1 回程度作成
計算にかけられる時間：1 週間程度
- ▶ 看護師の勤務表作成
看護師数 25 名程度，3 交替制，1 ヶ月に 1 回作成
計算にかけられる時間：1 時間程度

現実問題の付帯条件を細かく検討する必要がある

スケジューリングのような問題にも様々

- ▶ 中学校の時間割作成
教員数 30 名程度，学級数 15 程度，1 年に 1 回作成
計算にかけられる時間：1 日程度 ← 目標とする実行時間
- ▶ 銀行 ATM の現金補充計画作成
ATM 数 200 個程度，拠点数 20 個程度，1 年に 1 回程度作成
計算にかけられる時間：1 週間程度 ← 目標とする実行時間
- ▶ 看護師の勤務表作成
看護師数 25 名程度，3 交替制，1 ヶ月に 1 回作成
計算にかけられる時間：1 時間程度 ← 目標とする実行時間

現実問題の付帯条件を細かく検討する必要がある

スケジューリングのような問題にも様々

- ▶ 中学校の時間割作成
教員数 30 名程度, 学級数 15 程度, 1 年に 1 回作成
計算にかけられる時間: 1 日程度 ← 目標とする実行時間
- ▶ 銀行 ATM の現金補充計画作成
ATM 数 200 個程度, 拠点数 20 個程度, 1 年に 1 回程度作成
計算にかけられる時間: 1 週間程度 ← 目標とする実行時間
- ▶ 看護師の勤務表作成
看護師数 25 名程度, 3 交替制, 1 カ月に 1 回作成
計算にかけられる時間: 1 時間程度 ← 目標とする実行時間

「我々の目標とするもの」と「我々の達成できるもの」のギャップが重要

- ▶ 目標が達成できる ⇒ 達成すればよい
- ▶ 目標が達成できない ⇒ どうすれば? (今日の内容)

(3) 計画を実行すること

アルゴリズム的問題解決における「計画を実行すること」

- ▶ アルゴリズムをプログラムとして実現 (実装) すること
- ▶ プログラムを実行すること

注意

「アルゴリズム」と「プログラム」は明確に区別される

アルゴリズム	手続きの抽象的記述	(主に数学による)
プログラム	手続きの具体的記述	(主にプログラミング言語による)

つまり,

- ▶ 「速いアルゴリズム」と「速いプログラム」は区別しないとイケない
- ▶ 「アルゴリズムがプログラムとして実現できるか」は分からない

(4) 振り返ってみること

アルゴリズム的問題解決における「振り返ってみること」

- ▶ 問題が正しく理解できているか，評価する
- ▶ アルゴリズムとプログラムに満足できるか，評価する

評価ができたなら，また (1) に戻って，繰り返す

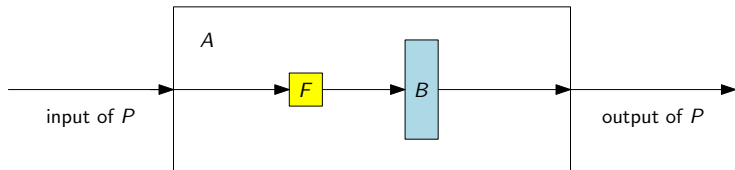
- ▶ 解くべき問題が理解できてないと 判明することも多い

性質

ある NP 完全問題が多項式時間で解ける $\Rightarrow P = NP$

証明：仮定にある「ある NP 完全問題」を Q とする

- ▶ Q を多項式時間で解くアルゴリズム B が存在 (仮定)
- ▶ NP に所属する任意の判定問題を P とすると、
 P は Q に多項式時間多対一帰着可能 (NP 完全性の定義)
- ▶ P を Q に帰着するアルゴリズムを F とする
- ▶ このとき、 B と F から、 P を解く多項式時間アルゴリズムが作れる



性質

ある NP 完全問題が多項式時間で解ける \Rightarrow $P = NP$

直感

NP 完全問題はクラス NP の中で最も難しい問題

未解決問題

NP 完全問題は多項式時間で解けるか？

性質の対偶

$P \neq NP \Rightarrow$ どの NP 完全問題も多項式時間で解けない

(2) 計画をたてること

アルゴリズム的問題解決における「計画をたてること」

- 1 アルゴリズム設計の**方針**を立てること
- 2 アルゴリズムを設計すること

アルゴリズム設計の方針とは？(再)

- ▶ 我々が目標とする「実行時間」と「メモリ消費量」の特定
- ▶ 我々が達成できる「実行時間」と「メモリ消費量」の特定
← この講義の内容

NP 完全性の使い方

対象とする問題が NP 完全問題であると分かれば

$P \neq NP$ という仮定の下で、

「多項式時間で解くこと」が達成できないと分かる

(←落とせない仮定)

- ① アルゴリズム的問題解決と P vs NP 問題：復習
- ② アルゴリズム的問題解決の進展と妥協点
- ③ 制限アプローチ
- ④ 厳密アプローチ
- ⑤ 近似アプローチ
- ⑥ 今日のまとめ と 全体のまとめ

NP 完全性の使い方

対象とする問題が NP 完全問題であると分かれば

$P \neq NP$ という仮定の下で、

(←落とせない仮定)

「多項式時間で解くこと」が達成できないと分かる

しかし、問題を解かないといけない！ (←社会的要請)

どうすればよいのか？

NP 完全性の使い方

対象とする問題が NP 完全問題であると分かれば

$P \neq NP$ という仮定の下で，

(←落とせない仮定)

「多項式時間で解くこと」が達成できないと分かる

しかし，問題を解かないといけない！ (←社会的要請)

どうすればよいのか？

- ▶ $P = NP$ であることを証明する

NP 完全性の使い方

対象とする問題が NP 完全問題であると分かれば

$P \neq NP$ という仮定の下で，

(←落とせない仮定)

「多項式時間で解くこと」が達成できないと分かる

しかし，問題を解かないといけない！ (←社会的要請)

どうすればよいのか？

- ▶ $P = NP$ であることを証明する
- ▶ 全面的に諦める

NP 完全性の使い方

対象とする問題が NP 完全問題であると分かれば

$P \neq NP$ という仮定の下で，

(←落とせない仮定)

「多項式時間で解くこと」が達成できないと分かる

しかし，問題を解かないといけない！ (←社会的要請)

どうすればよいのか？

- ▶ $P = NP$ であることを証明する
- ▶ 全面的に諦める
- ▶ 妥協する
 - ▶ 全面的に諦めるわけではなく，部分的に諦める

NP 完全問題を解くことは諦めるのか？

NP 完全性の使い方

対象とする問題が NP 完全問題であると分かれば
P \neq NP という仮定の下で， (←落とせない仮定)
「多項式時間で解くこと」が達成できないと分かる

しかし，問題を解かないといけない！ (←社会的要請)

どうすればよいのか？

- ▶ P = NP であることを証明する
- ▶ 全面的に諦める
- ▶ 妥協する
 - ▶ 全面的に諦めるわけではなく，部分的に諦める

考えるべきこと

何を諦めて，何を諦めないのか？

アルゴリズムに求められていること

- ▶ すべての入力に対して (入力の普遍性)
- ▶ 素早く (処理の効率性)
- ▶ 正しい出力を行うこと (出力の正当性)

これらの中の、**1つだけ**を諦めることを考える

- ▶ 「すべての入力に対して」を諦める ⇨ 制限アプローチ
- ▶ 「素早く」を諦める ⇨ 厳密アプローチ
- ▶ 「正しい出力を行うこと」を諦める ⇨ 近似アプローチ

アルゴリズムに求められていること

- ▶ すべての入力に対して (入力の普遍性)
- ▶ 素早く (処理の効率性)
- ▶ 正しい出力を行うこと (出力の正当性)

制限アプローチでアルゴリズムに求められていること

- ▶ 一部分の入力に対して (入力の普遍性を緩和)
- ▶ 素早く (処理の効率性)
- ▶ 正しい出力を行うこと (出力の正当性)

アルゴリズムに求められていること

- ▶ すべての入力に対して (入力の普遍性)
- ▶ 素早く (処理の効率性)
- ▶ 正しい出力を行うこと (出力の正当性)

厳密アプローチでアルゴリズムに求められていること

- ▶ すべての入力に対して (入力の普遍性)
- ▶ ある程度の時間で (処理の効率性を妥協)
- ▶ 正しい出力を行うこと (出力の正当性)

アルゴリズムに求められていること

- ▶ すべての入力に対して (入力の普遍性)
- ▶ 素早く (処理の効率性)
- ▶ 正しい出力を行うこと (出力の正当性)

近似アプローチでアルゴリズムに求められていること

- ▶ すべての入力に対して (入力の普遍性)
- ▶ 素早く (処理の効率性)
- ▶ 正しい出力に近い出力を行うこと (出力の正当性を妥協)

なぜ 3 つのアプローチは正当化されるか？

なぜ 3 つのアプローチは正当化されるか？

立場 1：理論的 (数学的) 立場

$P = NP$ を証明することが難しい

立場 2：実践的立場

現実問題の解決において、
「すべての入力において、素早く、正しい出力を行うこと」
が必要とされるわけではない

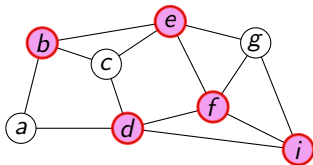
現実問題の解決において

- ▶ 現れる入力は限られるかもしれない ⇨ 制限アプローチ
- ▶ 時間はたっぷりあるかもしれない ⇨ 厳密アプローチ
- ▶ 少々の間違いは許されるかもしれない ⇨ 近似アプローチ

最小頂点被覆問題

- ▶ 入力：無向グラフ $G = (V, E)$
- ▶ 出力： G の頂点被覆 S
- ▶ 評価：要素数 $|S|$ の最小化

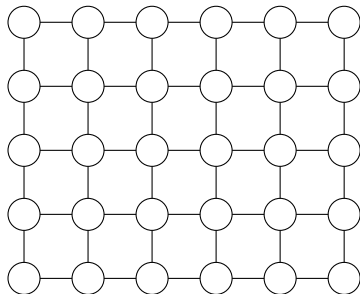
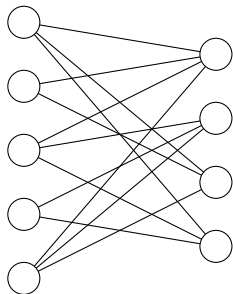
無向グラフ G の頂点被覆とは、 G の頂点部分集合 S で G のどの辺も S に端点を持つもの



復習：最小頂点被覆問題の判定問題版は NP 完全 (第 4 回講義)

- ① アルゴリズム的問題解決と P vs NP 問題：復習
- ② アルゴリズム的問題解決の進展と妥協点
- ③ 制限アプローチ
- ④ 厳密アプローチ
- ⑤ 近似アプローチ
- ⑥ 今日のまとめ と 全体のまとめ

二部グラフとは、頂点集合を2つの独立集合に分割できるグラフ
(2彩色を持つグラフ と言い換えることもできる)

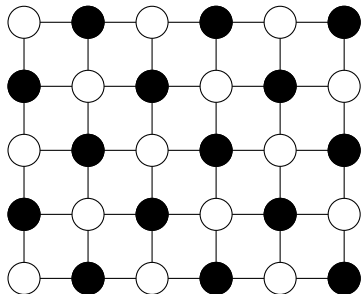
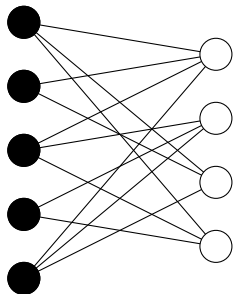


定理

二部グラフに入力を限定した最小頂点被覆問題は
多項式時間で解ける

多項式時間で解ける仕組みを説明したい

二部グラフとは、頂点集合を2つの独立集合に分割できるグラフ
(2彩色を持つグラフ と言い換えることもできる)



定理

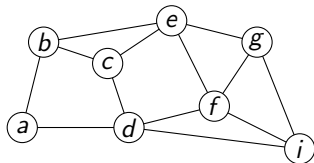
二部グラフに入力を限定した最小頂点被覆問題は
多項式時間で解ける

多項式時間で解ける仕組みを説明したい

マッチング問題

- ▶ 入力：無向グラフ G , 自然数 $k \in \mathbb{N}$
- ▶ 出力：Yes か No
- ▶ 条件： G が要素数 k 以上のマッチングを持つ \Rightarrow Yes
 G が要素数 k 以上のマッチングを持たない \Rightarrow No

無向グラフ G の **マッチング** とは, G の辺部分集合 S で S のどの2辺も端点を共有しないもの



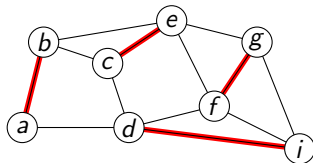
多項式時間で解ける

(Edmonds '65)

マッチング問題

- ▶ 入力：無向グラフ G , 自然数 $k \in \mathbb{N}$
- ▶ 出力：Yes か No
- ▶ 条件： G が要素数 k 以上のマッチングを持つ \Rightarrow Yes
 G が要素数 k 以上のマッチングを持たない \Rightarrow No

無向グラフ G の **マッチング** とは, G の辺部分集合 S で S のどの2辺も端点を共有しないもの



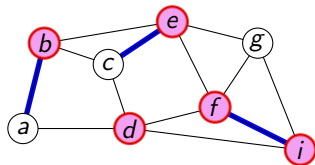
多項式時間で解ける

(Edmonds '65)

事実

任意のグラフに対して

任意のマッチングの要素数 \leq 任意の頂点被覆の要素数



特に、任意のグラフに対して

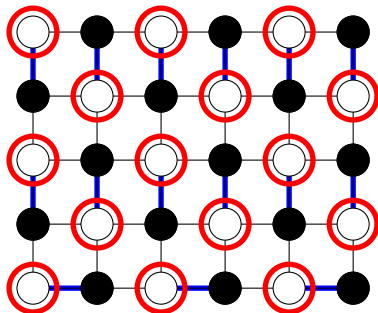
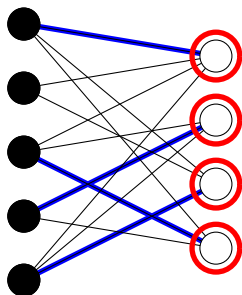
マッチングの最大要素数 \leq 頂点被覆の最小要素数

事実

(König '31; Egerváry '31)

二部グラフに対して,

マッチングの最大要素数 = 頂点被覆の最小要素数



事実 (König '31; Egerváry '31)

二部グラフに対して,

$$\text{マッチングの最大要素数} = \text{頂点被覆の最小要素数}$$

つまり, 二部グラフに対しては

▶ マッチングの最大要素数が分かれば, 頂点被覆の最小要素数も分かる
どんなグラフに対しても

▶ マッチングの最大要素数は多項式時間で計算できる

∴ 二部グラフに対しては

▶ 頂点被覆の最小要素数が多項式時間で計算できる

要素数から頂点被覆自体も多項式時間で計算できる (第1回講義参照) □

制限アプローチの利点

- ▶ 入力を制限すると、多項式時間で解ける可能性が上がる

制限アプローチの欠点

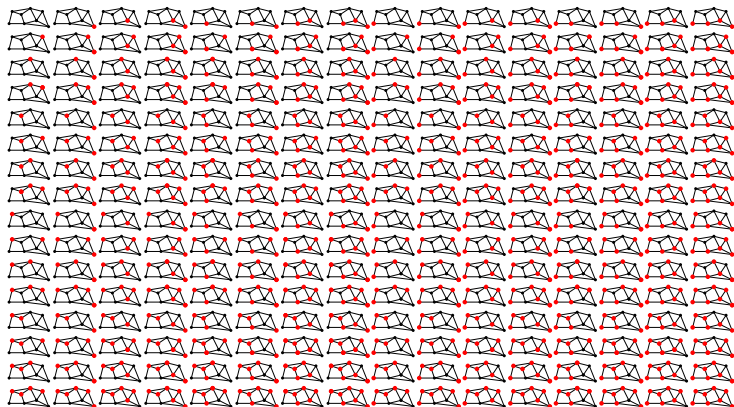
- ▶ 入力を制限すると、扱える入力の種類が小さくなる
- ▶ 入力を制限しても、多項式時間で解けないかもしれない

例：平面的グラフに対する最小頂点被覆問題 (判定問題版) は NP 完全
(第 10 回講義参照)

- ① アルゴリズム的問題解決と P vs NP 問題：復習
- ② アルゴリズム的問題解決の進展と妥協点
- ③ 制限アプローチ
- ④ 厳密アプローチ
- ⑤ 近似アプローチ
- ⑥ 今日のまとめ と 全体のまとめ

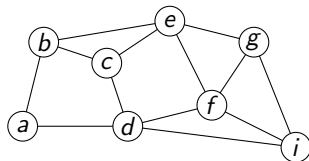
単純な厳密アルゴリズム (正しい出力をするアルゴリズム)

すべての頂点部分集合を見ていく

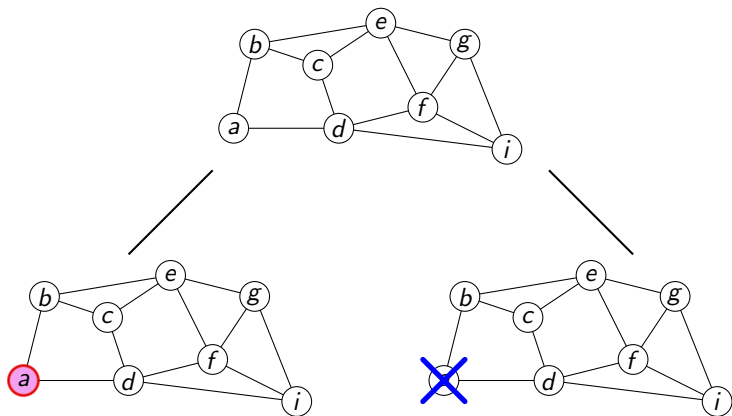


頂点部分集合の総数 = 2^n (n は頂点の総数)

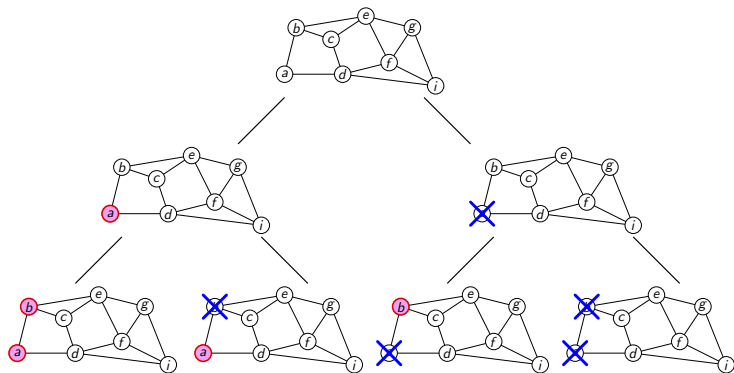
場合分けの木を通して，アルゴリズムを見直す



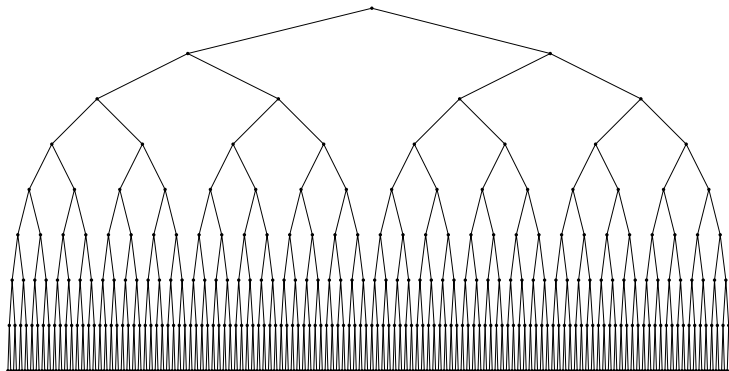
場合分けの木を通して、アルゴリズムを見直す



場合分けの木を通して、アルゴリズムを見直す

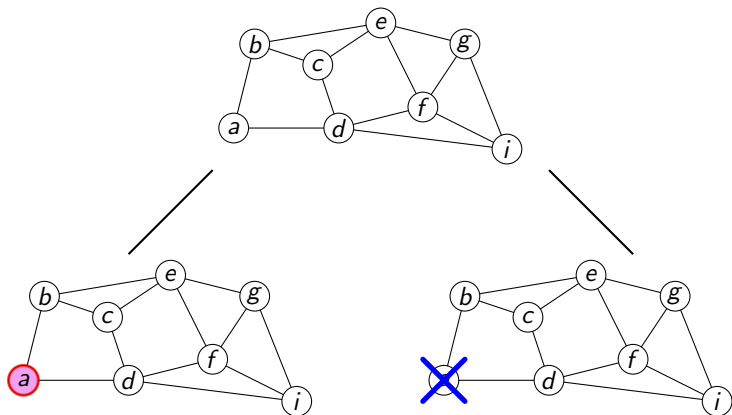


場合分けの木を通して，アルゴリズムを見直す

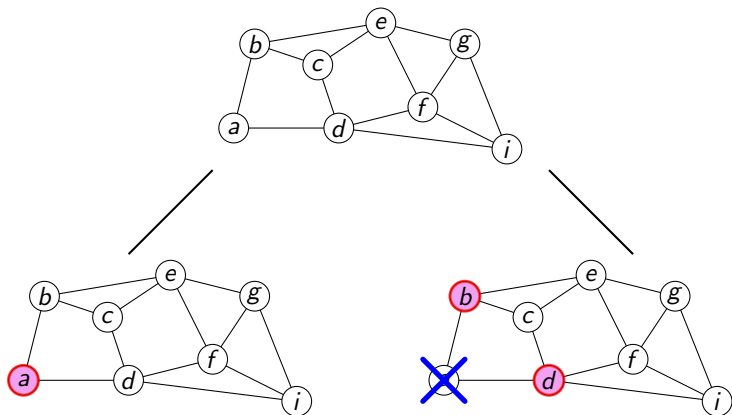


この木の葉 \leftarrow 1対1対応 \rightarrow 頂点部分集合

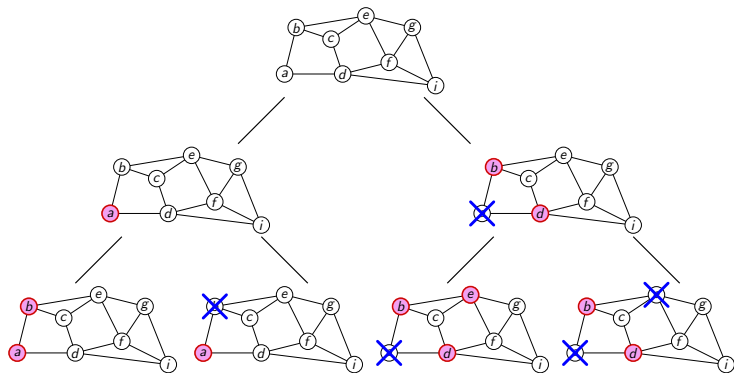
最小頂点被覆問題：少し頭を使う (1)



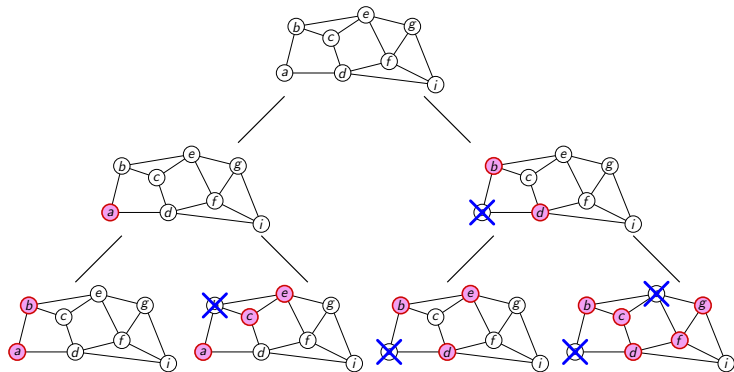
最小頂点被覆問題：少し頭を使う (1)



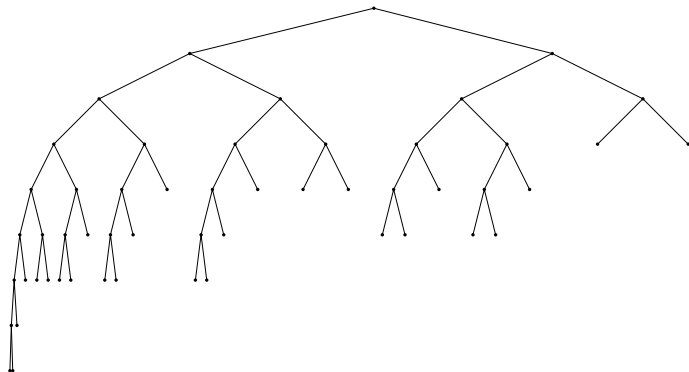
最小頂点被覆問題：少し頭を使う (2)



最小頂点被覆問題：少し頭を使う (2)



場合分けの木を通して、アルゴリズムを見直す



木の葉の数がかなり減った \rightsquigarrow アルゴリズムの効率がよくなった

再帰として考える

- 1 G の頂点 v を頂点被覆に含めるとき
 $\rightsquigarrow G - v$ の頂点被覆を考えればよい
- 2 G の頂点 v を頂点被覆に含めるとき
 v に隣接する端点 u を頂点被覆に含めないといけない
 $\rightsquigarrow G - v - u$ の頂点被覆を考えればよい

頂点数 n のグラフに対する最悪時実行時間を $t(n)$ とすれば，次が成立

$$t(n) \begin{cases} = O(1) & (n = 1 \text{ のとき}) \\ \leq t(n-1) + t(n-2) + O(1) & (n \geq 2 \text{ のとき}) \end{cases}$$

これを解くと， $t(n) = O\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right) = O(1.618^n)$ が得られる

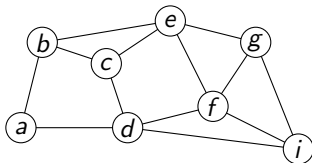
教訓

厳密アプローチであっても，工夫をすると，理論的にも実践的にも「単純なアルゴリズム」より効率のよいアルゴリズムを作れる

- ① アルゴリズム的問題解決と P vs NP 問題：復習
- ② アルゴリズム的問題解決の進展と妥協点
- ③ 制限アプローチ
- ④ 厳密アプローチ
- ⑤ 近似アプローチ
- ⑥ 今日のまとめ と 全体のまとめ

よさそうな頂点被覆を高速に作るアルゴリズム

(1) 極大マッチング M を 1 つ 見 つ け る

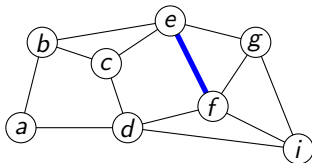


M が極大マッチング :

M に属さない辺 e に対して, $M \cup \{e\}$ がマッチングではない

よさそうな頂点被覆を高速に作るアルゴリズム

(1) 極大マッチング M を 1 つ 見 つ け る

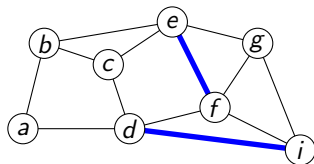


M が極大マッチング :

M に属さない辺 e に対して, $M \cup \{e\}$ がマッチングではない

よさそうな頂点被覆を高速に作るアルゴリズム

(1) 極大マッチング M を 1 つ 見 っ け る

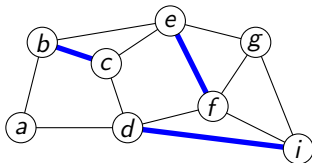


M が極大マッチング :

M に属さない辺 e に対して, $M \cup \{e\}$ がマッチングではない

よさそうな頂点被覆を高速に作るアルゴリズム

(1) 極大マッチング M を 1 つ 見 つ け る

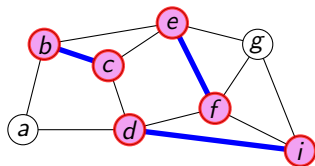


M が極大マッチング :

M に属さない辺 e に対して, $M \cup \{e\}$ がマッチングではない

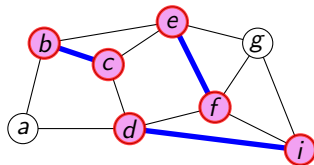
よさそうな頂点被覆を高速に作るアルゴリズム

(2) M の辺の端点である頂点をすべて集めて、出力する



考えるべき事項

- 1 この方法で，頂点被覆が必ず出力されるのか？
- 2 この方法が出力する頂点被覆は，どれぐらいよいのか？



よい = 要素数が最小頂点被覆の要素数に近い

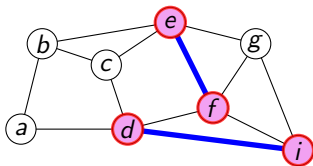
証明すること

先ほどの方法は、頂点被覆を必ず出力する

証明：

- ▶ (背理法) 出力が頂点被覆ではないとする
- ▶ このとき、両端点が出力に含まれない辺 e が存在する
- ▶ しかし、このとき、 $M \cup \{e\}$ はマッチングなので、 M が極大マッチングであることに矛盾

□

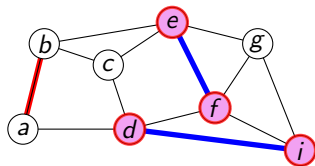


証明すること

先ほどの方法は、頂点被覆を必ず出力する

証明：

- ▶ (背理法) 出力が頂点被覆ではないとする
- ▶ このとき、両端点が出力に含まれない辺 e が存在する
- ▶ しかし、このとき、 $M \cup \{e\}$ はマッチングなので、 M が極大マッチングであることに矛盾

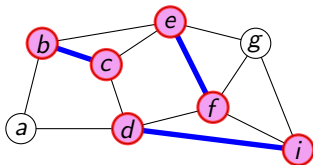


証明すること

先ほどの方法が出力する頂点被覆の要素数は
最小頂点被覆の要素数の2倍以下

証明：先ほどの方法の出力を S ，任意の最小頂点被覆を S^* とする

- ▶ S を作るために使った極大マッチングを M とする
- ▶ M はマッチングなので， $|S| = 2|M|$
- ▶ M はマッチング， S^* は頂点被覆なので， $|M| \leq |S^*|$
- ▶ $\therefore |S| = 2|M| \leq 2|S^*|$



- ① アルゴリズム的問題解決と P vs NP 問題：復習
- ② アルゴリズム的問題解決の進展と妥協点
- ③ 制限アプローチ
- ④ 厳密アプローチ
- ⑤ 近似アプローチ
- ⑥ 今日のまとめ と 全体のまとめ

NP 完全問題を解くことは諦めるのか？

全面的に諦めるわけではなく、部分的に諦める (妥協する)

アルゴリズムに求められていること

- ▶ すべての入力に対して (入力の普遍性)
- ▶ 素早く (処理の効率性)
- ▶ 正しい出力を行うこと (出力の正当性)

これらの中の、**1つだけ**を諦めることを考える

- ▶ 「すべての入力に対して」を諦める ⇨ 制限アプローチ
- ▶ 「素早く」を諦める ⇨ 厳密アプローチ
- ▶ 「正しい出力を行うこと」を諦める ⇨ 近似アプローチ

例として、最小頂点被覆問題を考察対象とした

これらの中の、**1つだけ**を諦めることを考える

- ▶ 「すべての入力に対して」を諦める ⇨ 制限アプローチ
- ▶ 「素早く」を諦める ⇨ 厳密アプローチ
- ▶ 「正しい出力を行うこと」を諦める ⇨ 近似アプローチ

例として、最小頂点被覆問題を考察対象とした

疑問

これら3つのアプローチには限界があるのか？

答 : あると思われる

- ▶ 次の記事に (少し古い内容であるが) 解説がある
岡本吉央, 「組合せ最適化理論の三次元描像」, 第21回回路とシステム軽井沢ワークショップ, 2008年.

<http://dopal.cs.uec.ac.jp/okamotoy/PDF/2008/comb3d.pdf>

主題

離散最適化のトピックの1つとして離散最適化における計算困難性を取り上げ、計算困難性の考え方が離散最適化にどう応用されるのか、理解する

なぜ講義で取り扱う？

- ▶ 重要な概念であるから
- ▶ アルゴリズム設計の根幹を成すから
- ▶ まとまって講義として取り扱われることが、あまりないから