計算の理論と現実
難しいはずの計算が実はいとも？簡単に

国立情報学研究所
情報学プリンシプル研究系　助教
岩田　陽一

自己紹介

岩田 陽一

所属: 国立情報学研究所
情報学プリンシプル研究系
助教（2016年～）

専門: 組合せ最適化
パラメータ化計算量

趣味: プログラミングコンテスト（優勝多数）

秋葉・岩田・北川
（マイナビ出版）

本日の内容

アルゴリズムとは？

計算の難しさとは？

本当に難しいのか？

私の研究の紹介

キーワード:

- fine-grained complexity
- parameterized complexity

アルゴリズムとは？

アルゴリズムとは

「問題を解くための手順」

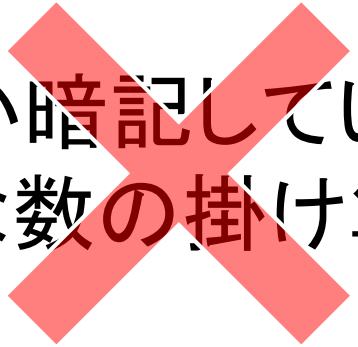
コンピュータが生まれるより前からある

- 筆算
- ユークリッドの互除法
- ガウスの消去法
- エラトステネスの篩（ふるい）

など

掛け算

$$123 \times 456 = ?$$



九九までしか暗記していないから
こんな大きな数の掛け算は出来ない…☹

筆算

$$123 \times 456 = ?$$

			1	2	3	
×			4	5	6	
			7 ^{1}	3 ^{1}	8	
		6 ^{1}	1 ^{1}	5		
	4	9 ^{1}	2			
	5	6	0	8	8	

九九までしか覚えていない人間
＋筆算
＝何桁の掛け算でも出来る！

単純な計算しか出来ない機械
＋アルゴリズム
＝複雑な計算が出来る！

筆算の手間

$$123 \times 456 = ?$$

			1		2		3	
×			4		5		6	
			7	1	3	1	8	
		6	1	1	5			
	4	9	2					
	5	6	0		8		8	

n桁のとき

$O(n^2)$回の計算

厳密な計算回数の議論は本質ではないのでO記法を用いる

オーダー

O記法: $O(f(n))$

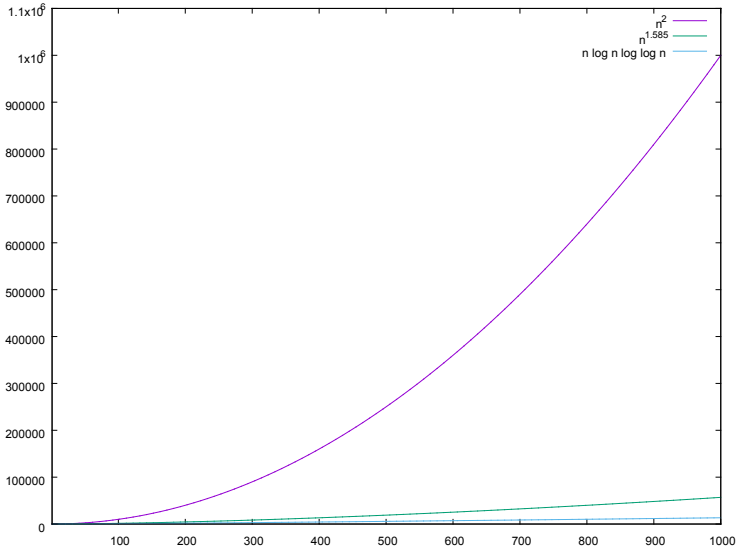
$f(n)$の定数倍で抑えられる

アルゴリズムと計算時間

使うアルゴリズムによって計算時間が大きく異なる

例：n桁の掛け算

- 筆算: $O(n^2)$時間
- Karatsuba法: $O(n^{1.585})$時間
- FFT: $O(n \log n \log \log n)$時間



用語:「問題」と「入力」

問題: 入出力の仕様

整数 a と b が与えられるので $a \times b$ を計算せよ。

入力: 具体的な値

$123 \times 456 = ?$

筆算は 123×456 だけでなく、任意の $a \times b$ に対して適用可能。

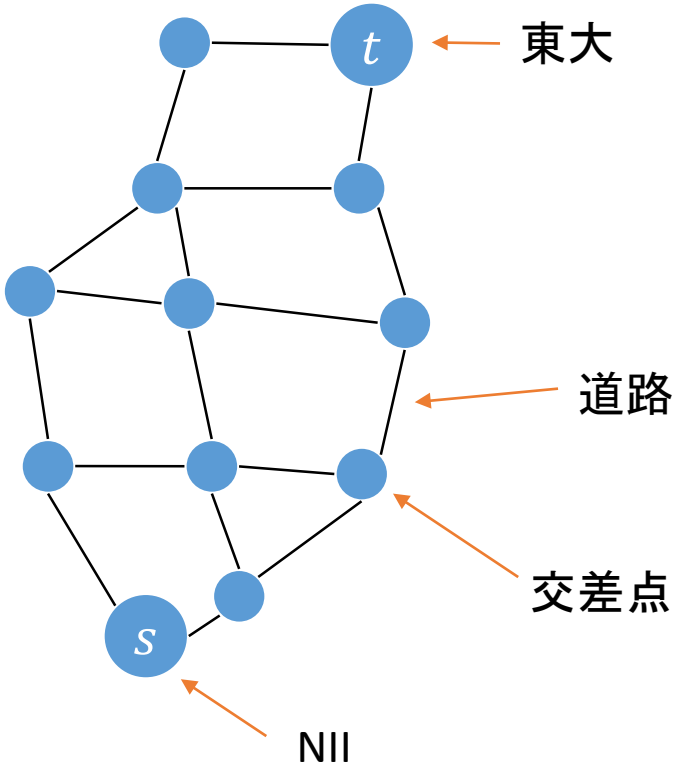
現実の問題を計算機で解く

NIIから東大へ最短経路で移動したい

Google map より

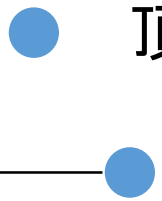
モデル化

グラフの最短路問題



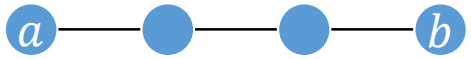
グラフとは？

要素の「つながり」を抽象化したもの

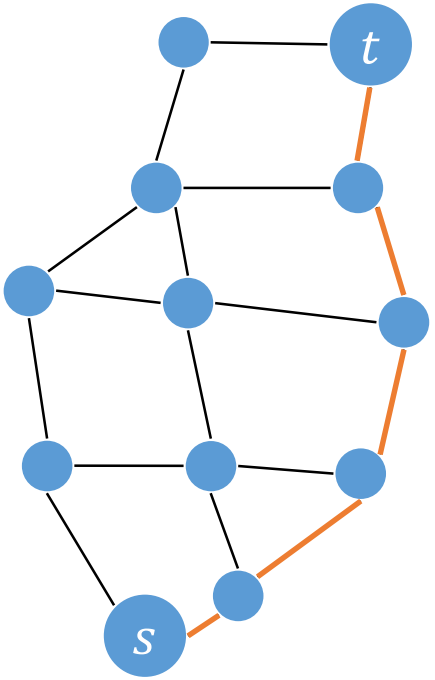


頂点

辺



aからbへの「パス」



最短路問題
n点m辺のグラフが与えられるので、
s から t への最短パスを計算せよ

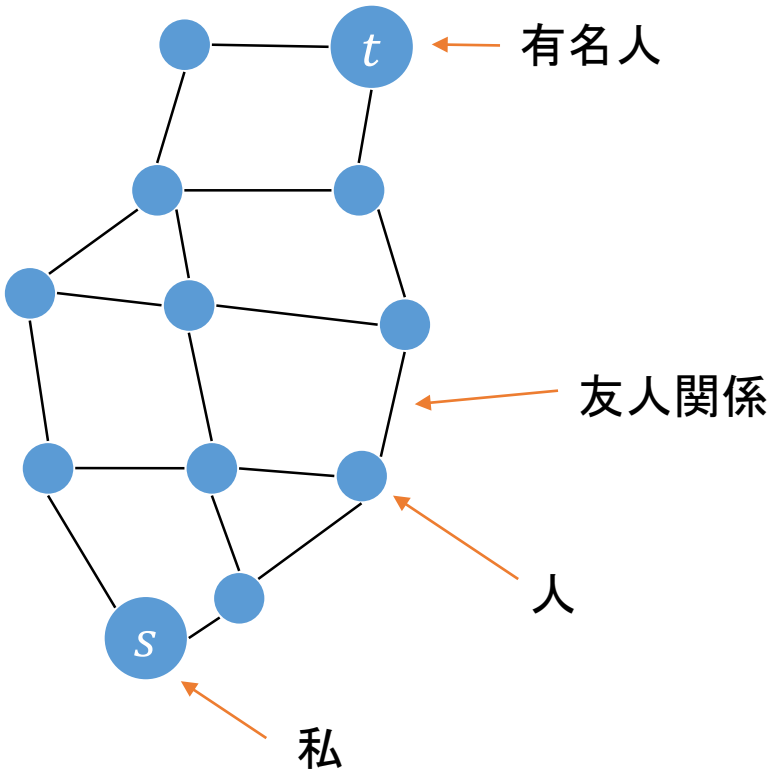
様々な現実の問題を統一的に扱える

グラフの最短路問題

有名人までのホップ数
例「友人の友人がアルカイダ」

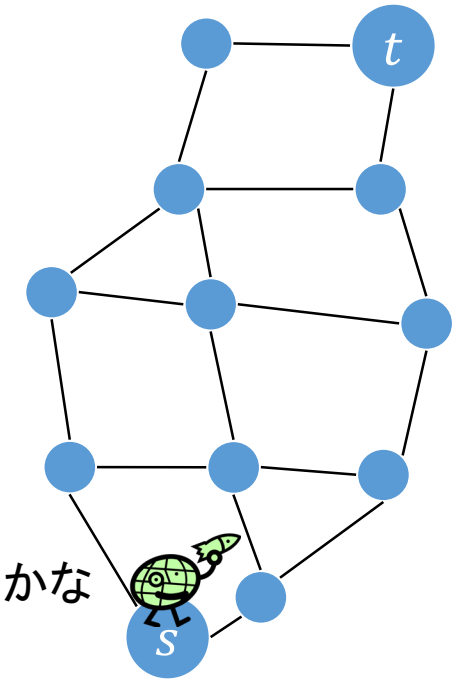
私の友人：A,B
Aの友人：私,C,D
Bの友人：私,C,E
⋮

モデル化



しらみつぶし法

あらゆる可能性を全通り試して、
一番良いものを答える



左と右どっちに行こうかな

同じ点を二回通る必要はないので、
$O(2^n)$時間

多項式時間と指数時間

線形時間	n	10	100	1,000
多項式時間	n^2	100	10,000	1,000,000
	n^3	1,000	10^6	10^9
指数時間	2^n	1,024	約10^{30}	約10^{301}

天文学的数字

宇宙の年齢＝138億年＝約 4×10^{17} 秒

指数時間はヤバイ！（指数爆発）

同じ計算をするなら、出来るだけ高速に行いたい。
どこまで効率化が可能？
実は最短路問題は$O(m \log n)$時間で解ける。
(ダイクストラ法）

本日の内容

アルゴリズムとは？

計算の難しさとは？

本当に難しいのか？

私の研究の紹介

計算量理論 (Complexity Theory)

「計算の難しさ」を科学する研究分野

効率化には「限界」があると信じられている

- 例: 暗号解読は効率的には出来ないはず

難しさの証明は非常に難しい

P ≠ NP 予想 （ミレニアム懸賞問題の一つ）

ある性質（**NP完全**）を満たす問題は多項式時間で解けない

解決したら100万ドル貰える

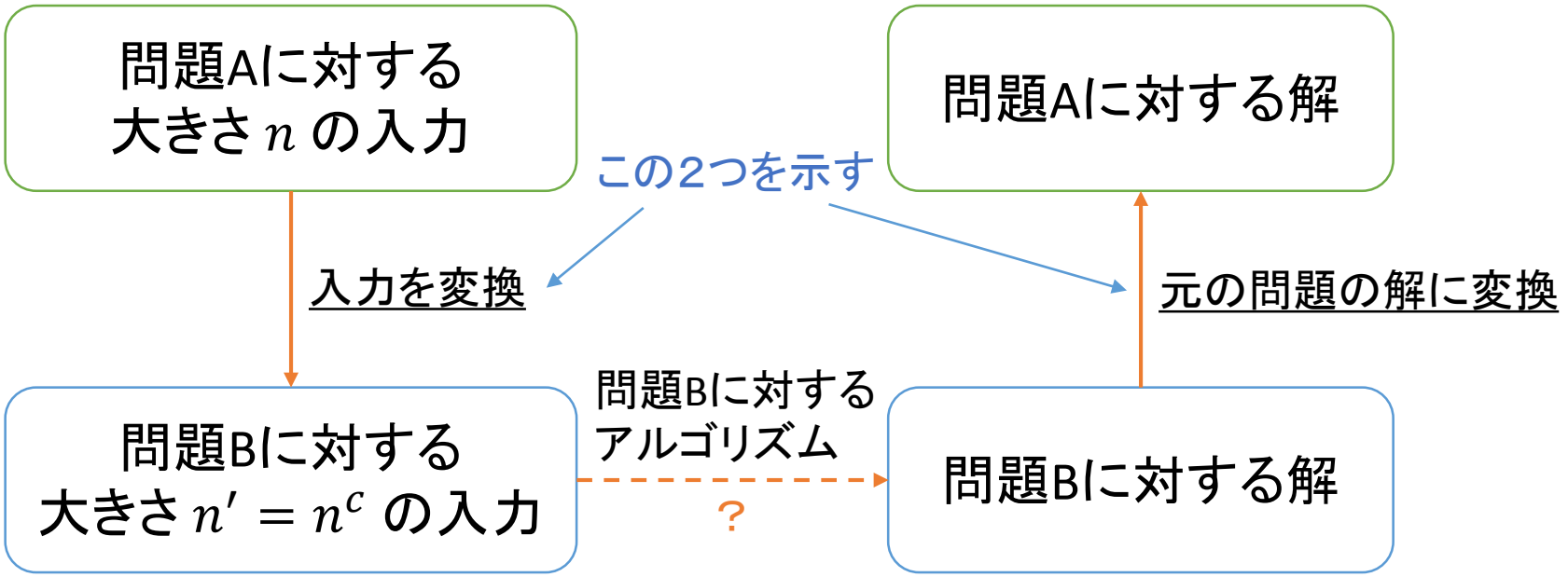
「計算の難しさ」の証明

「**絶対的**」な難しさの証明は非常に難しいが、
「**相対的**」な難しさは証明出来ることが多い。

問題Aが多項式時間で解けないなら、
問題Bも多項式時間で解けない。
つまり、問題Bは問題A以上に難しい。

「帰着(Reduction)」という手法を用いる。

帰着



もし、問題Bに対する多項式時間アルゴリズムがあれば、問題Aも多項式時間で解ける。

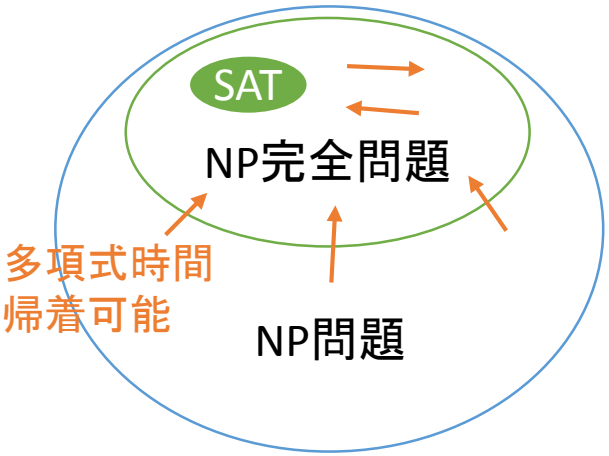
よって、「Aが多項式時間で解けないならば、Bも多項式時間で解けない」ことが証明できた。

充足可能性問題 (SAT)

以下の形式の n 変数の論理式が与えられるので、
全体を真にする真偽値割当があるか判定せよ。

$$(a \vee \bar{b} \vee \cdots \vee c) \wedge \cdots \wedge (\bar{d} \vee e \vee \cdots \vee f)$$

否定　　または　　かつ



NP完全問題の代表

$$\mathrm{P} \neq \mathrm{NP}$$
$$\Updownarrow$$
SATが多項式時間で解けない

例

3変数 $\{a, b, c\}$ の論理式

$$(a \vee b) \wedge \left(\bar{b} \vee c\right) \wedge (\bar{a} \vee \bar{b} \vee \bar{c})$$

$a =$ 真, $b =$ 真, $c =$ 真 とすると、
真 ∧ 真 ∧ 偽 = 偽

$a =$ 真, $b =$ 真, $c =$ 偽 とすると、
真 ∧ 偽 ∧ 真 = 偽

$a =$ 真, $b =$ 偽, $c =$ 真 とすると、
真 ∧ 真 ∧ 真 = 真

見つかった☺

SATの難しさ

しらみつぶし法を用いると$O(2^n)$時間

大きなギャップ

SATがもしも多項式時間で解けたら、$\mathrm{P} = \mathrm{NP}$

実は、$O(2^n)$ より速いアルゴリズムは見つかっていない。

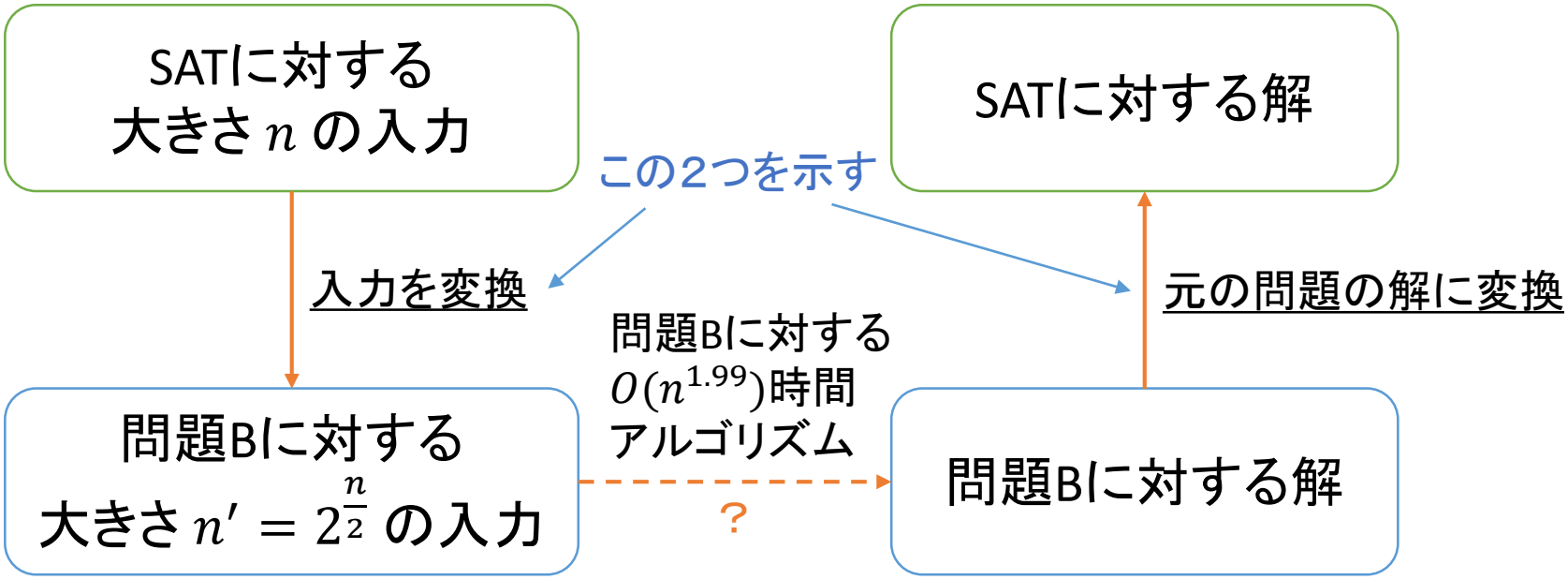
⇒ 不可能という予想も

SETH: Strong Exponential-Time Hypothesis

Fine-Grained Complexity

近年注目を集めている計算量理論の発展。

Fine-grained帰着という手法を用いることで、より細かな限界を示す。



SETHの仮定のもと、問題Bが$O(n^{1.99})$時間で解けないと示せた。

Fine-Grained Complexity

ある仮定の元で、

- グラフの半径・内径は$O(nm)$より速く計算出来ない。
- 最短路クエリを$O(nm)$より速い前処理かつ$O(m)$より速い応答速度で処理出来ない。

などなど [LWW18]

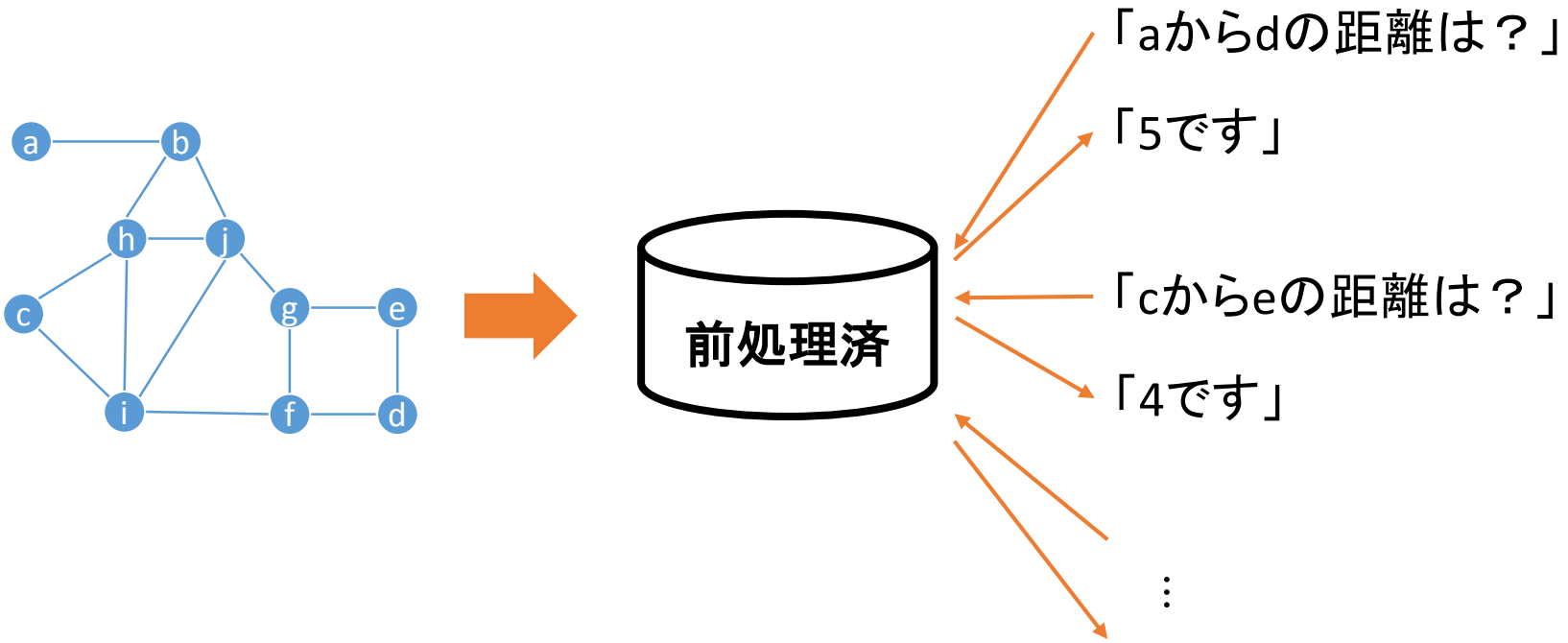
つまり、既存手法が理論上最速。

これ以上研究しても無駄！？

[LWW18] Lincoln, V. Williams, R. Williams: Tight Hardness for Shortest Cycles and Paths in Sparse Graphs. SODA 2018: 1236-1252

最短路クエリ

グラフを前処理することで、高速に二点間の最短路を答える。



自明な方法1：前処理$O(nm)$、クエリ$O(1)$

自明な方法2：前処理$O(1)$、クエリ$O(m)$

本日の内容

アルゴリズムとは？

計算の難しさとは？

本当に難しいのか？

私の研究の紹介

本当に難しいのか？

理論上難しいはずの問題が、実は現実には高速に解けている。

- 充足可能性問題（SAT）: $O(2^n)$で解けないという予想
  - CDCLという手法を用いたソルバが数万変数規模の論理式を処理できている。
- 巡回セールスマン問題（TSP）: NP完全
  - 分枝カット法という手法を用いたソルバが数万点規模のグラフを処理できている。
- 最短路クエリ問題: 前処理かクエリのどちらかは遅い
  - 2-hopラベリングという手法を用いて数億辺規模のグラフを数時間で前処理し、クエリ時間1ms以下を達成できている。

なぜ？　実はP = NPだった？？

効率化の限界とは

× 「どんな入力に対しても」速いアルゴリズムは存在しない
○ 「どんな入力に対しても速いアルゴリズム」は存在しない
特殊ケースに対しては，もっと速いアルゴリズムがあるかも

例: $1003 \times 997 = (1000 + 3) \times (1000 - 3)$
$= 1000^2 - 3^2$
$= 999991$

難しいはずの問題が現実には簡単に解けるのは、理論が間違っているのではなく、現実の入力が簡単なだけでした。
めでたしめでたし？

理論的に特殊ケースを扱いたい…

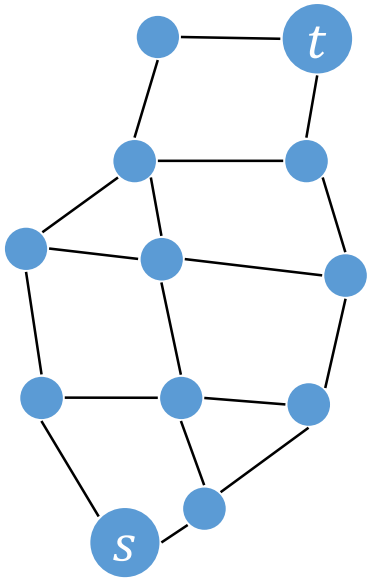
現実の入力は特殊ケース

東大からNIIへ最短
経路で移動したい

Google map より

グラフの最短路問題

モデル化



モデル化

有名人までのホップ数

私の友人：A,B
Aの友人：私,C,D
Bの友人：私,C,E
⋮

このモデル化は一般化しすぎ！

地図のグラフ、友人のグラフは一般のグラフに比べて
何か特殊な構造があるはず。

特殊ケースを理論的に扱うには

```
if 知っている入力:
    return 記憶しておいた答え
else:
    真面目に計算する
```

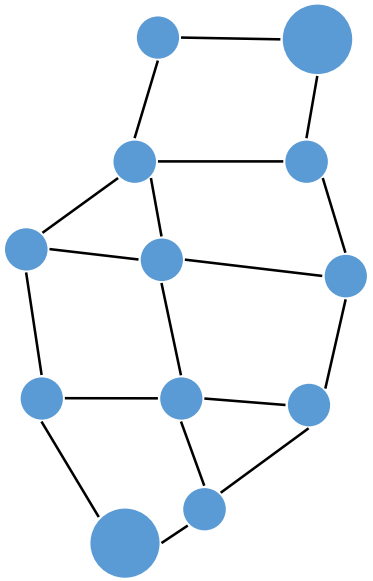
有限の入力に特化した
アルゴリズム

(有限個の)特定の入力に対する計算量は意味がない。

地図グラフの性質、友人グラフの性質、などを考えて、「～という性質がある入力」に対するアルゴリズムと計算量を考える。

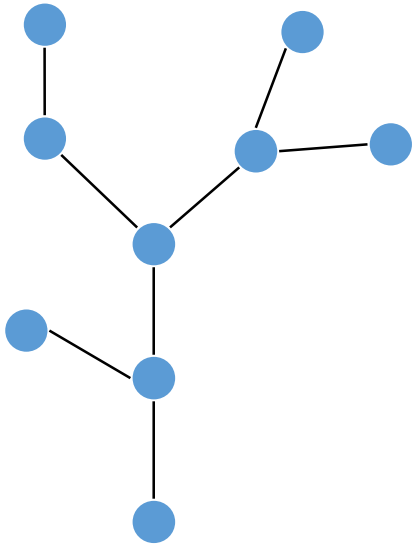
特殊なグラフの例

平面グラフ



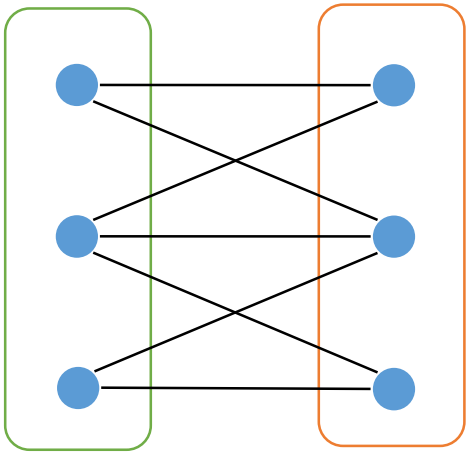
二次元平面に辺の交差
なく描画出来る

木



閉路がない

二部グラフ



頂点が2つのグループに別れ
同じグループ間には辺が無い

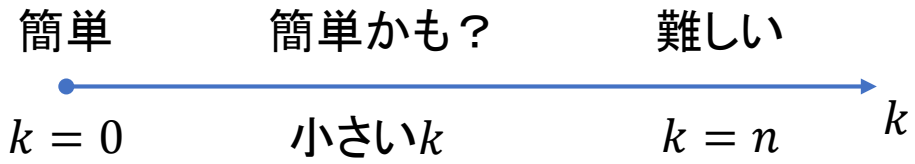
高速なアルゴリズムが存在するが、制約が強すぎて応用先が限られる。

パラメータ化計算量

計算時間の評価で、入力サイズ n の他に**パラメータ k** を導入。

パラメータの選び方は任意（扱う特殊ケースが変わる）

「簡単な超特殊ケース」からの距離をkとする気分。



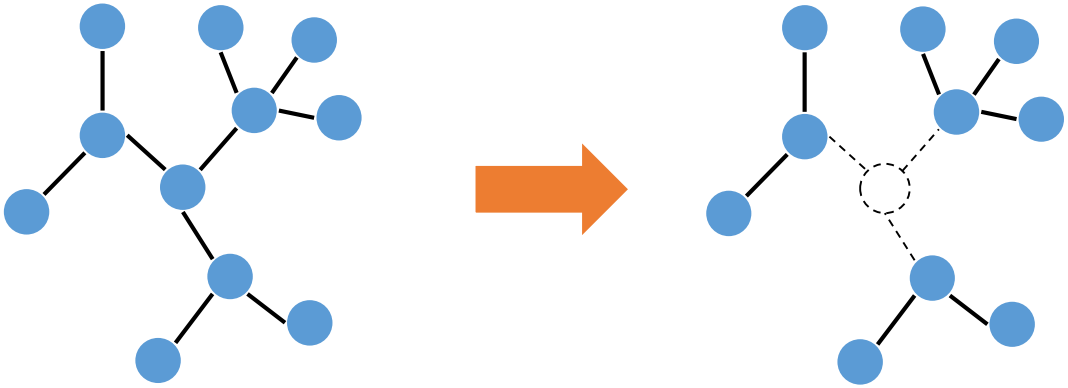
一般には$O(2^n)$時間かかるものが、もし $O(2^k \mathrm{poly}(n))$ 時間で解けると示せたら、kが小さい特殊ケースに対しては効率的であると見なせる。

FPT: **F**ixed **P**arameter **T**ractable という。

パラメータの例1：グラフの「木っぽさ」

木

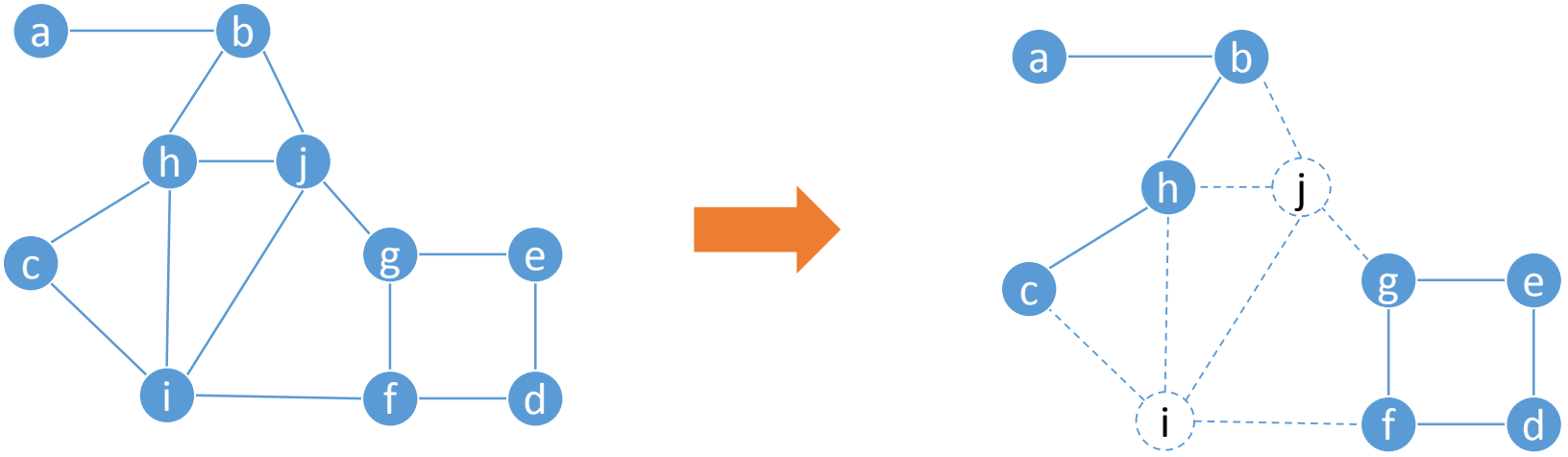
- 閉路が無いグラフ
- ある一点を取り除くことで、大きさ $\frac{n}{2}$ 以下の木に分割出来る



パラメータの例1：グラフの「木っぽさ」

木っぽさが k のグラフ

- ある k 点を取り除くことで、大きさ $\frac{n}{2}$ 以下の木っぽさが k のグラフに分割出来る



木の「木っぽさ」は定数、
平面グラフの「木っぽさ」は$O(\sqrt{n})$で抑えられる。

木っぽさのデモ

「木っぽさ」の指標

色々ある

- Tree-width (木幅)
- Branch-width
- Tree-depth

などなど

どれも大体同じ

$$\mathrm{bw} \leq \mathrm{tw} + 1 \leq \frac{3}{2}\mathrm{bw}$$

$$\mathrm{tw} + 1 \leq \mathrm{td} \leq (\mathrm{tw} + 1)\log n$$

「木っぽい」入力に対するアルゴリズムの例

- 様々なNP完全問題が、入力の木幅がkの場合は動的計画法を用いて $O(\text{定数}^{k} n)$ 時間で解ける。
- SATは一般には$O(2^n)$が最速だが、CDCLは木幅がkの論理式を$O(n^{2k+2})$時間で処理出来る [AFT11]
- 最大マッチングは一般には $O(\sqrt{n}m)$ が最速だが、木幅がkのグラフは $O(k^4 n)$時間で処理出来る [FLSPW18]

[AFT11] Atserias, Fichte, Thurley: Clause-Learning Algorithms with Many Restarts and Bounded-Width Resolution. J. Artif. Intell. Res. 40: 353-373 (2011)

[FLSPW18] Fomin, Lokshtanov, Saurabh, Pilipczuk, Wrochna: Fully Polynomial-Time Parameterized Computations for Graphs and Matrices of Low Treewidth. ACM Trans. Algorithms 14(3): 34:1-34:45 (2018)

パラメータの例2: Max 2-SAT

以下の形式の n 変数の論理式が与えられるので、
全体を真にする真偽値割当があるか判定せよ。

$$(a \vee \bar{b}) \wedge \cdots \wedge (\bar{d} \vee e)$$

（ … ）の中（節という）が2変数まで

実はこの問題（2-SATという）は線形時間で解ける。

パラメータの例2: Max 2-SAT

全体を真に出来ない場合には、出来るだけ沢山の節を真にしたい（Max 2-SAT）

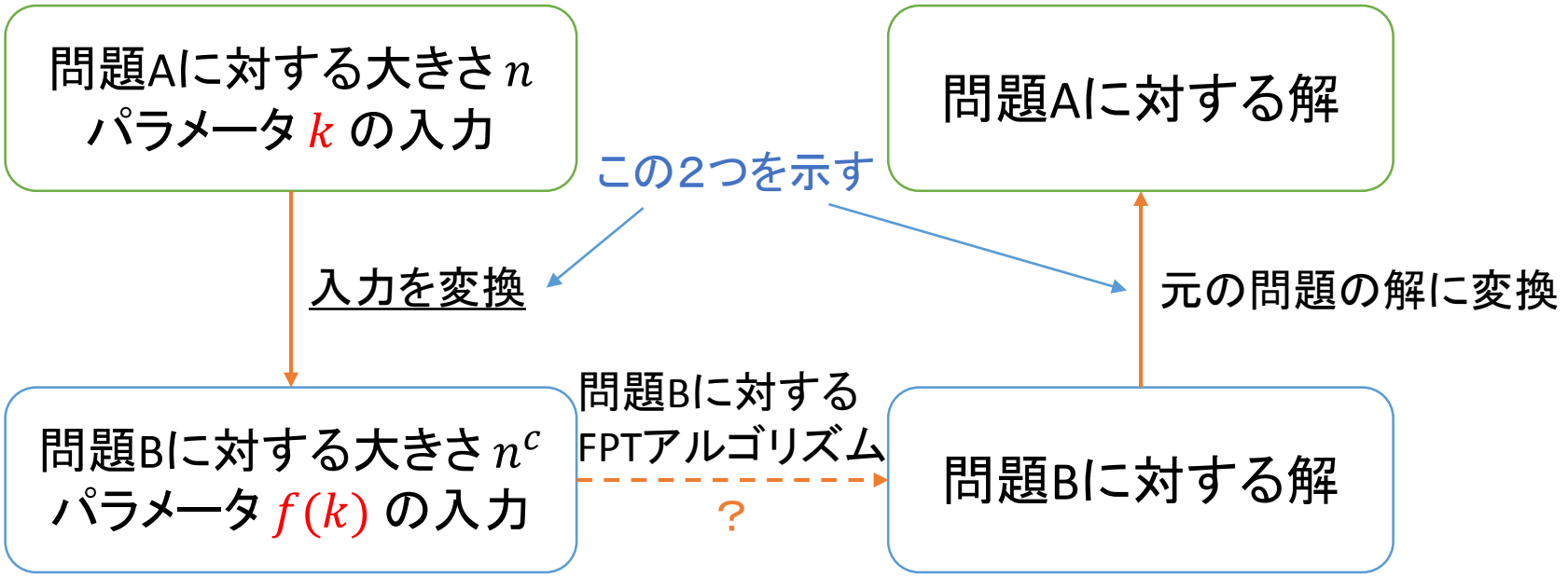
⇒ NP完全となり、一気に難しくなる

真に出来ない節の数をパラメータ k とする

- $k = 0$ は 2-SAT なので線形時間
- 任意の定数kに対して、$O(m^3)$時間で解ける [RO09]

[RO09] Razgon, O'Sullivan: Almost 2-SAT is fixed-parameter tractable. J. Comput. Syst. Sci. 75(8): 435-450 (2009)

特殊ケースの難しさの証明



問題Aの特殊ケースを問題Bの特殊ケースに変換する帰着を示すことが出来ると、「Aの特殊ケースが難しい（つまり、FPTアルゴリズムが存在しない）ならば、Bの特殊ケースも難しい」ことが証明出来る。

- 例: リスト彩色問題は木っぽいグラフでも難しい

本日の内容

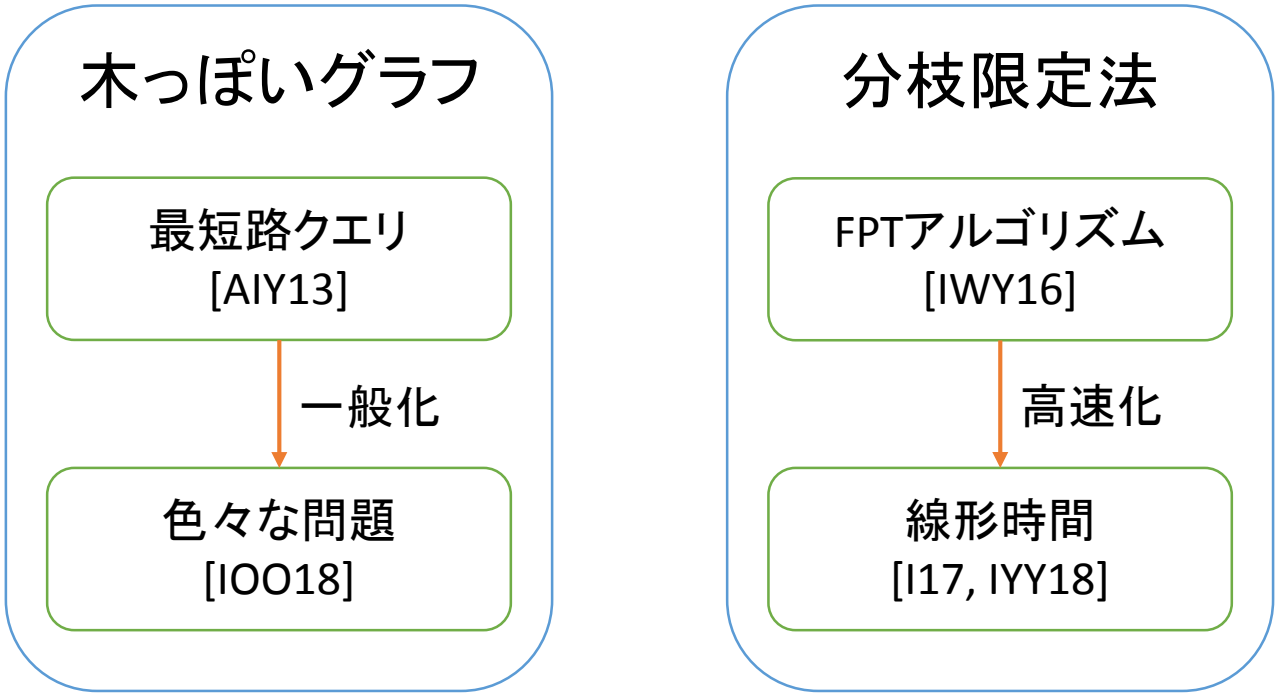
アルゴリズムとは？

計算の難しさとは？

本当に難しいのか？

私の研究の紹介

私の研究の紹介

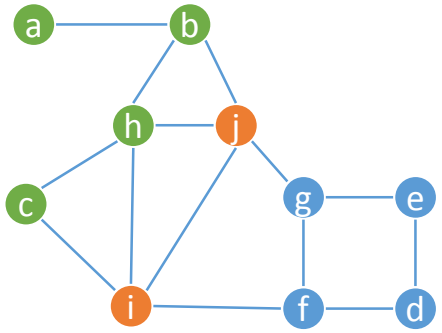


[AIY13] Akiba, Iwata, Yoshida: Fast exact shortest-path distance queries on large networks by pruned landmark labeling. SIGMOD 2013: 349-360
[IOO18] Iwata, Ogasawara, Ohsaka: On the Power of Tree-Depth for Fully Polynomial FPT Algorithms. STACS 2018: 41:1-41:14
[IWY16] Iwata, Wahlström, Yoshida: Half-integrality, LP-branching, and FPT Algorithms. SIAM J. Comput. 45(4): 1377-1411 (2016)
[I17] Iwata: Linear-Time Kernelization for Feedback Vertex Set. ICALP 2017: 68:1-68:14
[IYY18] Iwata, Yamaguchi, Yoshida: 0/1/All CSPs, Half-Integral A-Path Packing, and Linear-Time FPT Algorithms. FOCS 2018: 462-473

木っぽいグラフ

最短路クエリ [AIY13]

アイデア： 遠くに行くためには、必ず通る頂点集合Sがある。
S からの距離を前計算しておけば、近くだけの問題に
→ 再帰的に適用



左半分から右半分に行くには、
かならずiかjを通る。

「木っぽい」グラフで理論上高速。現実のグラフは割と木っぽい。
１億辺程度の現実グラフで

	前処理時間	クエリ時間
前処理なし	–	数秒
全点間前計算	数年	10^{-7}秒
提案手法	1時間	10^{-6}秒

木っぽいグラフのアルゴリズム [IOO18]

様々な問題について、入力が木っぽい（tree-depthが小さい）場合に高速なアルゴリズムを示した。

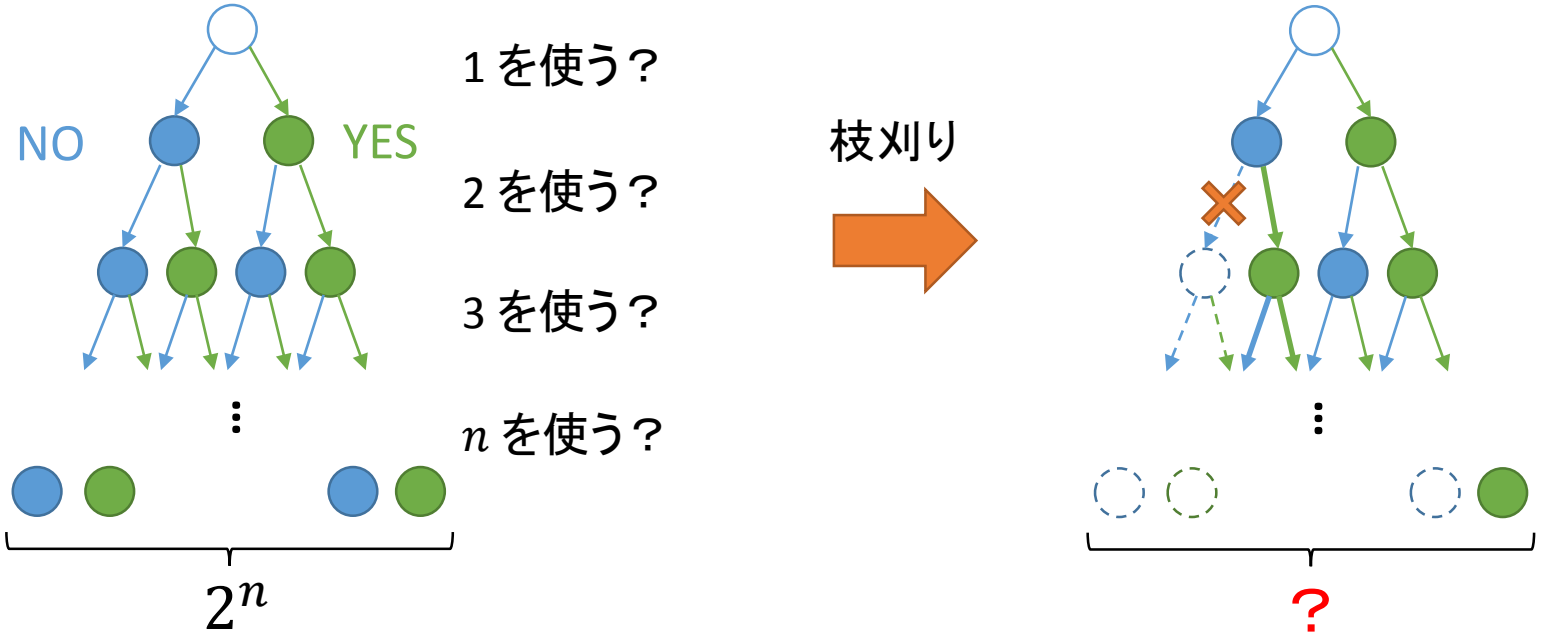
- 既存手法より高速で、適用範囲も広い

問題	一般の場合	木っぽさがk
最大マッチング	$O(\sqrt{n}m)$	$\tilde{O}(km)$
重み付き最大マッチング	$\tilde{O}(nm)$	$\tilde{O}(km)$
負閉路検出	$\tilde{O}(nm)$	$\tilde{O}(km)$
内径	$\tilde{O}(nm)$	$\tilde{O}(km)$
代替パス	$\tilde{O}(nm)$	$\tilde{O}(km)$
最短路クエリ	クエリ $\tilde{O}(m)$ or 前処理 $\tilde{O}(nm)$	クエリ $O(k)$ and 前処理 $\tilde{O}(km)$

分枝限定法

分枝限定法

$\{1, \ldots, n\}$ の部分集合で，～を満たす最小のものを求める



分枝限定法: しらみつぶし法の途中で、「この先何個必要か」を見積もる。欲しい大きさを超えていたら、枝刈り出来る。
計算時間は見積もれないが、実用上とても高速。

分枝限定法の計算量 [IWY16]

「この先何個必要か？」→下界という

証明したこと:
下界がある性質（k劣モジュラ性）を満たすなら、
分枝限定法の計算時間は

$$O(\text{定数}^{\text{gap}} \times \text{下界の計算時間})$$

で抑えられる。

gap ＝ 最適解と初期下界の差

現実の入力では、gapは割と小さいことが多い。

gapが小さいとすぐに枝刈りされて直感的には高速に解けそうな気がするが、実際に正しい。

分枝限定法の計算量 [IWY16]

様々な問題がこの性質を満たす。

特に、gap = 最適解 − 初期下界 ≤ 最適解 なので、最適解 k が小さい場合にも高速であると言える。

既存手法に比べ、適用範囲が広く、kへの依存が小さい。

代表的な問題	計算時間
Max 2-SAT	$O\left(4^k \mathrm{poly}(n)\right)$
フィードバック点集合	$O\left(4^k \mathrm{poly}(n)\right)$
多分割カット	$O\left(2^k \mathrm{poly}(n)\right)$

他にも多数

高速化 [I17,IYY18]

組合せ最適化のテクニック（増大路）を用いると、
下界を線形時間で計算出来るようになる。

代表的な問題	計算時間
Max 2-SAT	$O(4^k m)$
フィードバック点集合	$O(4^k m)$
多分割カット	$O(2^k m)$

他にも多数

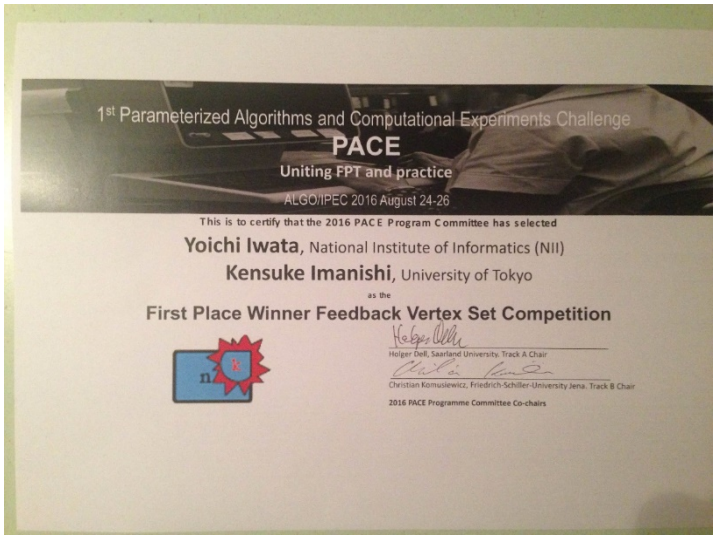
2-SAT は線形時間で解ける。
Max 2-SATは一般にはNP完全だが、
ほぼ充足可能な特殊ケースは線形時間で解ける！

競技会

フィードバック点集合を高速に計算するソルバの性能を競う競技会(PACE Challenge 2016)にて、先のアルゴリズムを実装して優勝！

https://github.com/wata-orz/fvs

理論だけでなく、実際に有用



まとめ

- 同じ問題を解くのにも、使うアルゴリズムによって必要な計算時間が大きく異なる
- 計算が難しいとは、「どんな入力に対しても速いアルゴリズム」は存在しない（だろう）ということ
- 「実際に解きたい特殊ケース」に対して速いアルゴリズムは存在するかもしれない
- 特殊ケースの難しさを理論的に扱う「パラメータ化計算量」と、特殊ケースでの速さが理論的に保証されたアルゴリズムの紹介

まだまだ分からないことだらけ