

全ソートアルゴリズム入場！ 地下ソート場最大トーナメント！



- Credit: 板垣恵介、「グラッpler刃牙」21巻 (秋田書店)

自己紹介

- さめ (мeг-ccк)
 - 🚗 フリーランスのソフトウェアエンジニア
 - 🎓 社会人学生として通信制大学在学中
- 得意分野:
 - 📸 コンピュータビジョン
(画像認識/点群処理)
 - 🌎 空間情報処理 (地理情報/リモートセンシング)
 - ☁ クラウドインフラ設計/IaC (AWS, GCP)
- GitHub
- YouTube
- Speaker Deck



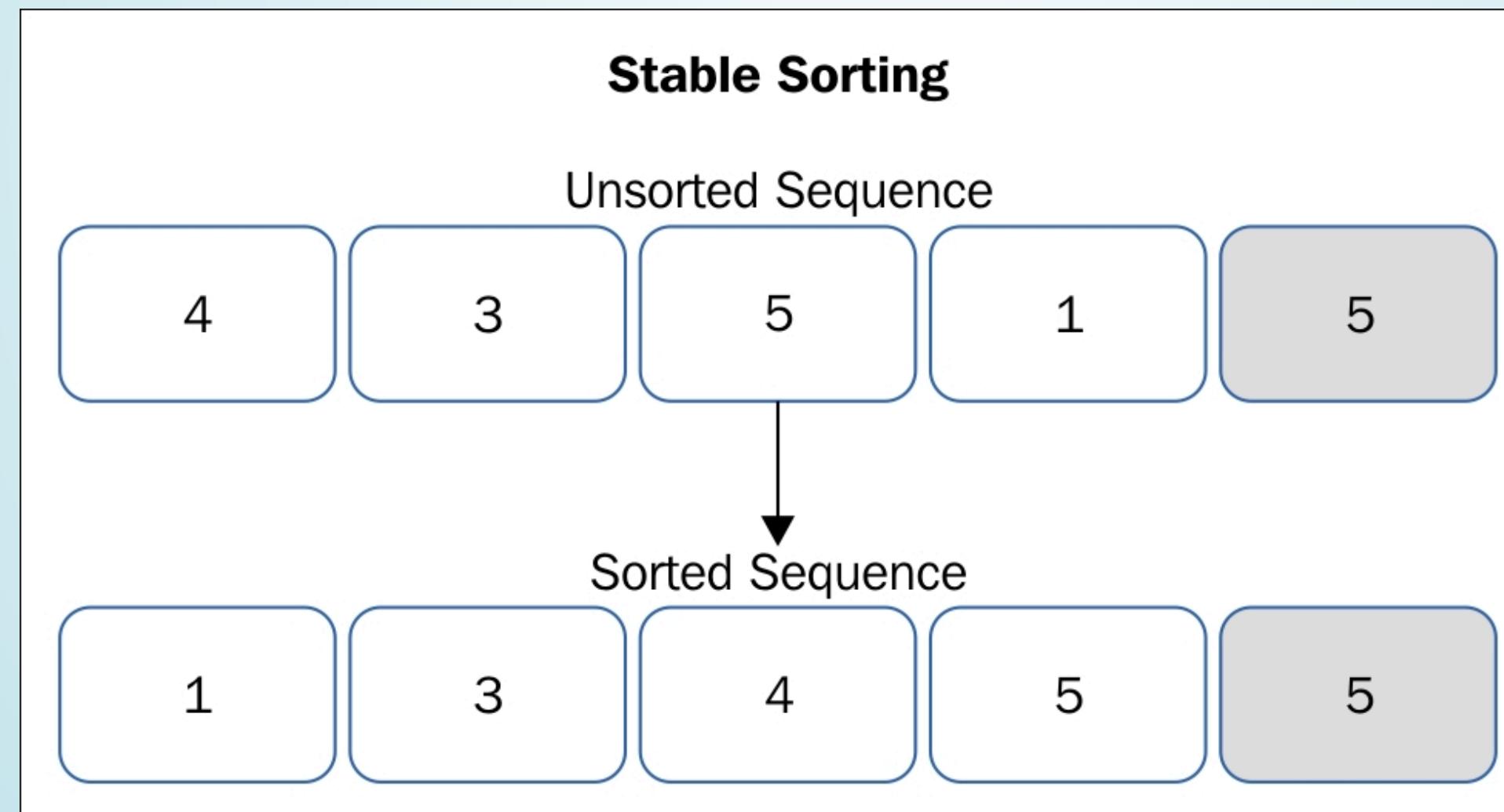
今日話すこと

- 「最強のソートアルゴリズム」を決めるトーナメントを開催するッ！
- 最大ソートアルゴリズムのトーナメントの意外な決着とはッ！？
- 「最強」とは？

- 刃牙を読んだことのない人は意味不明なネタですみません...

ソートとはッ！？

- ソートとはデータを特定の順序に並べ替えること
- 人類は研鑽を積み様々なソートアルゴリズムを開発してきた...



- Credit: Learning Functional Data Structures And Algorithms

代表的なソートアルゴリズム

- マージソート
- クイックソート
- ヒープソート
 - and so on...

- 誰もが思った、「最強のソートアルゴリズム」はど
れか？
- 今夜、「最強のソートアルゴリズム」を決めるト
ーナメントを開催ッ！

全アルゴリズム入場！



- Credit: 板垣恵介、「グラッパー刃牙」21巻 (秋田書店)

クイックソートッ！

ピボット選択は生きていた!!更なる分割を積みインプ
レース凶器が甦った!!!
武神!!クイックソートだア—————!!!



- Credit: 板垣恵介、「グラップラー刃牙」21巻 (秋田書店)

ヒープソートッ！

並べたいからここまできたッ 最初の考案者は不明!!!!
完全二分木のピットファイター ヒープソートだ!!!



- Credit: 板垣恵介、「グラップラー刃牙」21巻 (秋田書店)

ティムソートツ！

Pythonistの前でならオレはいつでも全盛期だ!!
燃える闘魂 ティム・ピーターズのティムソート 本
名で登場だ!!!



- Credit: 板垣恵介、「グラップラー刃牙」21巻 (秋田書店)

マージンートッシ！

めい土の土産にメモリーとはよく言ったもの!!
フォン・ノイマンの奥義が今 実戦でバクハツする!!
ロスアラモス流ソート術 マージソートだ——!!



- Credit: 板垣恵介、「グラップラー刃牙」21巻(秋田書店)

イントロソートッ！

ソートは実装で使ってナンボのモン!!! 超実戦
C++術!!
本家STLからイントロソートの登場だ!!!



- Credit: 板垣恵介、「グラップラー刃牙」21巻 (秋田書店)

シェルソートッ！

挿入ソートだったらこのアルゴリズムを外せない!!
超A級ソート師 シェルソートだ!!!



- Credit: 板垣恵介、「グラップラー刃牙」21巻 (秋田書店)

スムースソートッ！

ほとんどソート済みの数列のソートはこのアルゴリズムが完成させた!!
ダイクストラの切り札!! スムースソートだ!!!



- Credit: 板垣恵介、「グラップラー刃牙」21巻 (秋田書店)

ペーションスソートッ！

ソリティア四千年の歴史が今ベールを脱ぐ!!
最長増加部分列から ペーションスソートだ!!!



- Credit: 板垣恵介、「グラップラー刃牙」21巻 (秋田書店)

以上8名によってベルト争奪戦を行いますッ！



- Credit: 板垣恵介、「グラップラー刃牙」21巻 (秋田書店)

地下ソート場最大トーナ
メントの意外な決着ッ！

全員引き分け、全員優勝！

- 全試合が引き分けとなり、全員優勝となりました！
- なぜ格闘漫画だったら読者からのブーイング必須な結果になってしまったのか？

ソートアルゴリズムの理 論下限

ソートアルゴリズムの理論下限

- ソートアルゴリズムはこれ以上速くできない理論下限がある
- 今日の選手はすべてこの理論下限に達していた

- なぜソートにはこれ以上速くできない理論下限があるのか？

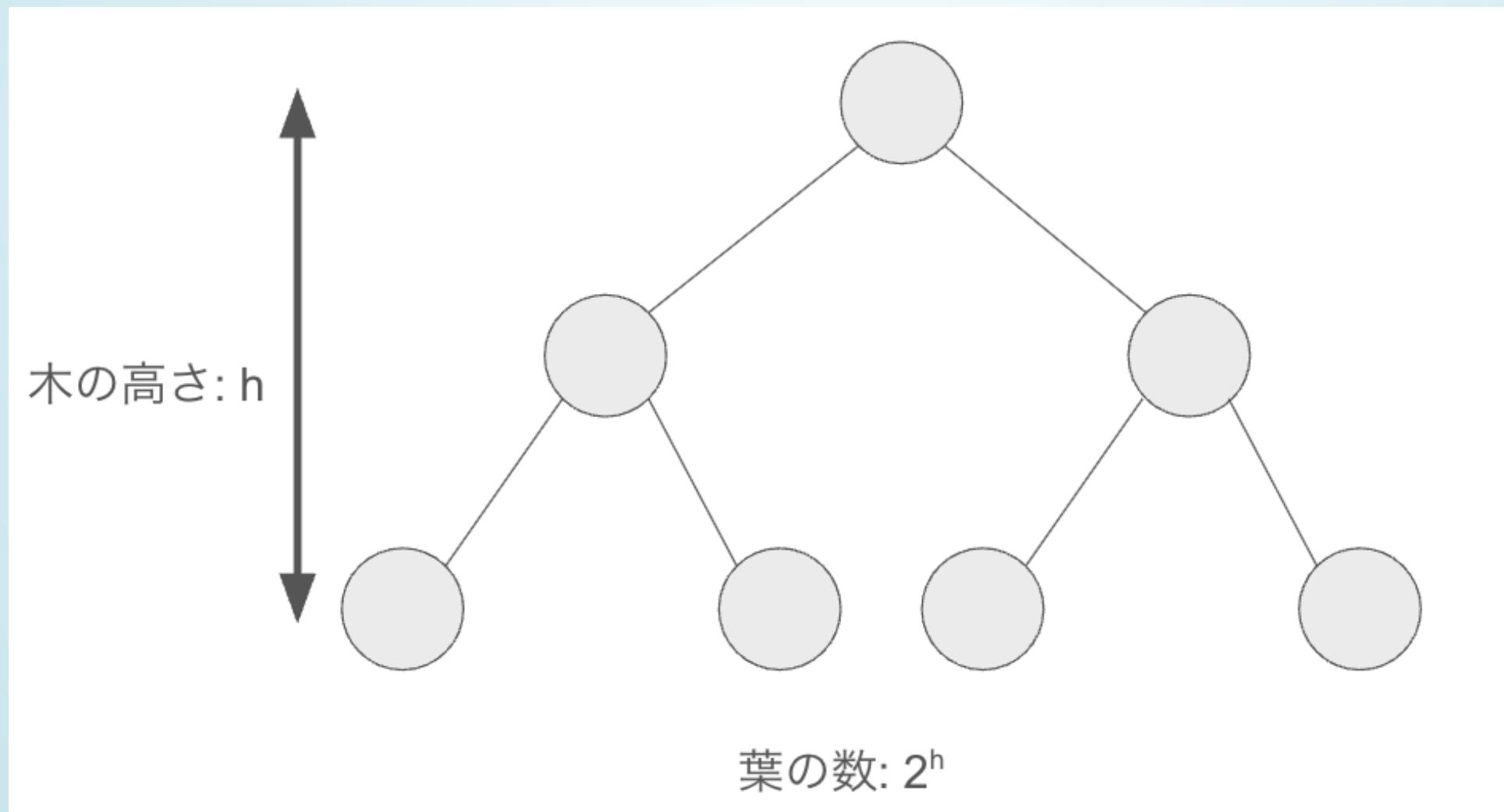
すべての並び替えの組み合わせの 数

n 個の数値を並び替える組み合わせの数は、 $n!$ 通り

- $n!$ 通りの中からたった一つの正解を見つける必要がある

二分木の高さと葉の数

- 二分木の高さを h とすると、最大で 2^h 個の並べ替えの組み合わせを比較できる



必要な木の高さは？

すべての数字の並びを比較するには

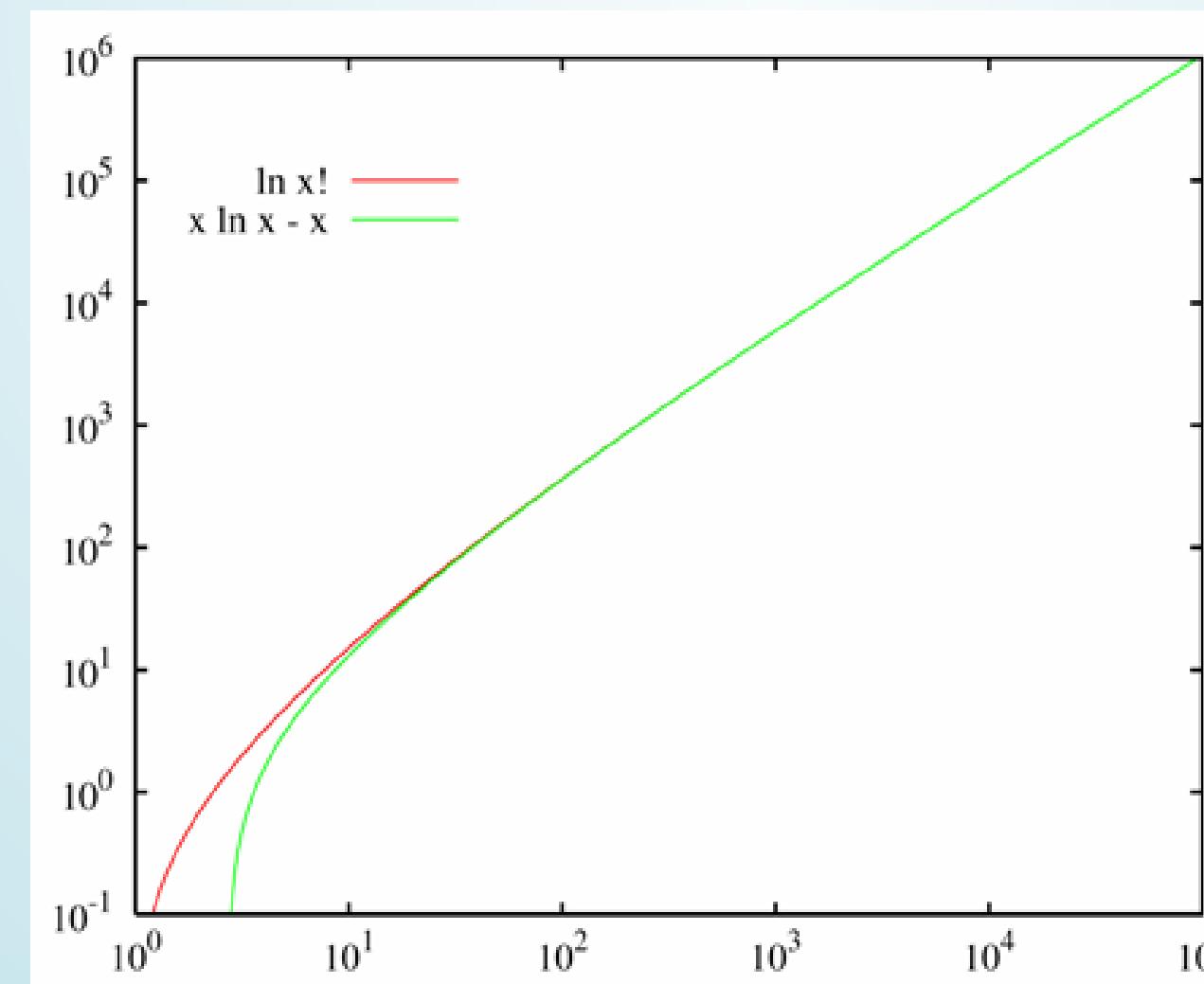
$$n! \leq 2^h$$

が必要！

スターリングの公式

- ガンマ関数(階乗の一般化)を近似できる便利な公式

$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e} \right)^n$$



木の高さを求める不等式を解く

$$n! \leq 2^h$$

$$\log_2 n! \leq h$$

左辺をスターリングの公式で近似すると

$$\log_2 n! \approx \log_2 \left(\sqrt{2\pi n} \left(\frac{n}{e} \right)^n \right)$$

さらに計算していく

$$\log_2 \sqrt{2\pi n} + n \log_2 \left(\frac{n}{e} \right)$$

$$\frac{1}{2} \log_2(2\pi n) + n \log_2 n - n \log_2 e$$

- 残った3つの項のうち、最も大きな項が計算量に影響する
- 一番大きな項は？

数学的補足: 計算量の表現記法

- 上界: $O(\cdot)$ 、必要な計算量の上限
- 下界: $\Omega(\cdot)$ 、最低でも絶対に必要な計算量
- 厳密計算量: $\Theta(\cdot)$ 、上界と下界が一致する場合

オーダーの比較

- $\frac{1}{2} \log_2(2\pi n) = O(\log n)$
- $n \log_2 n = O(n \log n)$ ➡ これが一番大きい
- $n \log_2 e = O(n)$

- 一番計算量が多い項は $n \log_2 n$
- n が十分に大きくなると、この項以外は無視してもOK！

$$\log_2 n! = O(n \log n)$$

数列をソートするのに必要な木の高さは？

$$h \geq \log_2 n! = \Omega(n \log n)$$

- いろんな近似をしているが、計算量の下界($\Omega(\cdot)$)の観点からは厳密に正しく、等号が成り立つ

ソートアルゴリズムの「最速」

- どんなにソートの計算を効率化しても、 $\Omega(n \log n)$ の計算量は避けられない
- これが理論上最速、これ以上に速い「汎用的」なソートアルゴリズムは存在しない！

今日の選手たちの「平均」計算量

- クイックソート: $O(n \log n)$
- ヒープソート: $O(n \log n)$
- マージソート: $O(n \log n)$
 - and so on...

- 今日の選手たちは皆、平均計算量は理論下限の $O(n \log n)$ に達していた！
- だから全員引き分けの全員優勝！

各アルゴリズムの得意・
不得意

クイックソート

- 平均計算量は $O(n \log n)$
- 最悪計算量は $O(n^2)$

- 長所: インプレース(追加メモリ不要)
- 短所: 最悪計算量が $O(n^2)$
 - 逆順に並んでいる場合、最悪計算量になる
 - $(5, 4, 3, 2, 1, 0) \rightarrow (0, 1, 2, 3, 4, 5)$ と並べ替えるのは苦手

ヒープソート

- 平均計算量は $O(n \log n)$
- 最悪計算量は $O(n \log n)$

- 長所: 最悪計算量が $O(n \log n)$ 、インプレース
- 短所: 不安定ソート(ソート途中で順序が維持されない)、ランダムアクセスが必要(並列化が難しい)

マージソート

- 平均計算量は $O(n \log n)$
- 最悪計算量は $O(n \log n)$

- 長所: 最悪計算量が $O(n \log n)$ 、安定ソート
- 短所: 追加で $O(n)$ のメモリが必要
 - 元々の数列以外にもう一つの数列を覚えておく必要がある
 - 小さな数列の場合はメモリを確保するのがオーバーヘッドになる

ソートアルゴリズムの使い分け

- 一般的な用途 → クイックソート（多くの言語の標準）
- 安定性が必要 → マージソート
- リアルタイム性重視 → ヒープソート（最悪保証かつインプレース）

- ソートアルゴリズムに「**最強**」は存在しない
- 得意・不得意があり、「**最適**」があるのみ

特定条件下で強いアルゴリズム

- 平均パフォーマンスでは今日のトーナメント出場選手たちに劣るが、特定条件下ではより高速

- 基数ソート: $O(kn)$
 - 入力データが限定されている場合に有効
- カウンティングソート: $O(n + k)$
 - 値の範囲が小さい場合に有効
- バケットソート: $O(n)$
 - 入力データが一様分布している場合に有効

まとめ

- ソートアルゴリズムに「最強」は存在しない
- 得意・不得意があり、「最適」があるのみ
- 汎用的にはパフォーマンスが劣っていても、特定条件下では圧倒的に強くなるアルゴリズムもある

- みんなちがって、みんないい
- それぞれの特性を把握して、状況に応じた適切なアルゴリズムを選ぶことが重要！

おまけ

- ただしバブルソート、テーマはダメだ
 - 「最強」は存在しないが明確な悪手は存在する
 - Leet Codeだとバブルソート的な計算量 $O(n^2)$ のアルゴリズムはTLEになるので効率化が重要！



- Credit: 澤井啓夫、「ボボボーボ・ボーボボ」1巻(集英社)