

<https://bit.ly/3EZI6Yc>

KYOTO UNIVERSITY

# アルゴリズムとデータ構造①

## ～ 概要 ～

鹿島久嗣  
(計算機科学コース)

DEPARTMENT OF INTELLIGENCE SCIENCE  
AND TECHNOLOGY

## 講義についての情報：

最新の情報はPandAで確認すること

---

- 担当教員： 鹿島久嗣  
（工学部情報学科計算機科学コース）
  - 連絡先： [kashima@i.kyoto-u.ac.jp](mailto:kashima@i.kyoto-u.ac.jp)
- PandAのページ： <http://bit.ly/3PWLWYq>
- 常に最新のスケジュール等をPandA で確認すること
- 評価方法： 中間試験 + 期末試験
  - 平常点（クイズ）は補助的に利用

## 参考書:

標準的なものであればなんでもよいが...

### ■ 基本 :

杉原厚吉「データ構造とアルゴリズム」 (共立出版)

- 本講義の多くの内容はこの本に依る
- とても読みやすい

### ■ より高度な内容 :

Cormen, Leiserson, Rivest, & Stein  
「Introduction to Algorithms」

- 翻訳 : 「アルゴリズムイントロダクション」 (近代科学社)
- 講義内容は部分的に参照



## 内容（前半）：

アルゴリズムの基本的な概念、評価法、基本的な道具

---

1. 算法とは・算法の良さの測り方：  
アルゴリズムとデータ構造、計算のモデル、計算複雑度、...
2. 基本算法：  
挿入、削除、整列、検索、...
3. 基本データ構造：  
リスト、スタック、キュー、ヒープ、...
4. 算法の基本設計法：  
再帰、分割統治、動的計画、...
5. 探索：  
二分探索、ハッシュ、...

内容（後半）：

## グラフ・計算量・難しい問題への対処法

---

6. グラフ算法：  
深さ・幅優先探索、最短路、最大流
7. 計算複雑度：  
PとNP、NP完全、NP困難
8. 難しい問題の解き方：  
分枝限定法、貪欲法
9. 発展的話題：  
近似アルゴリズム、オンラインアルゴリズム

※ 変更・追加の可能性あり

# アルゴリズムとデータ構造は

# 動機:

## 「良い」プログラムを書きたい

- プログラムの良し悪しとは：
  - 正しく動く： 想定したように動く
  - 速く動く： プログラムは速いほど良い！
  - 省資源： メモリや電気代
  - 例： お店の顧客管理
- 特定のプログラム言語やハードウェアとはなるべく独立に：
  - プログラムの良し悪しを測りたい
  - ひいては良いプログラムを書きたい



# アルゴリズム:

## 与えられた問題を解くための有限の手続き

### ■ アルゴリズム (algorithm) とは :

– プログラム言語 (CやPythonなど) やハードウェア (CPUやメモリなど) とは別に、どのような手続きを表現しようとするかという「問題の解き方の手順書」

– もうすこし厳密にいうと「与えられた問題を解くための機械的操作からなる、有限の手続き」

- 機械的操作 : 四則演算やジャンプなど

- かならず有限ステップで終わるべし



- 手続き (procedure) : 有限ステップでの終了が保証されない



# データ構造:

データを管理し、アルゴリズムを効率化する

---

- 多くのプログラムは「データ」を扱う
  - データは繰り返し使用するもの
  - 使用の仕方が予め決められているわけではない
- アルゴリズムがうまく動くためには、データをどのようにもっておくか（＝データ構造）が重要
  - 名前を、入力順に格納？ アイウエオ順？
- データ構造はアルゴリズムと切り離せないもの
  - お互いの良さに影響を与え合う

# アルゴリズムの例: 指数演算のアルゴリズム

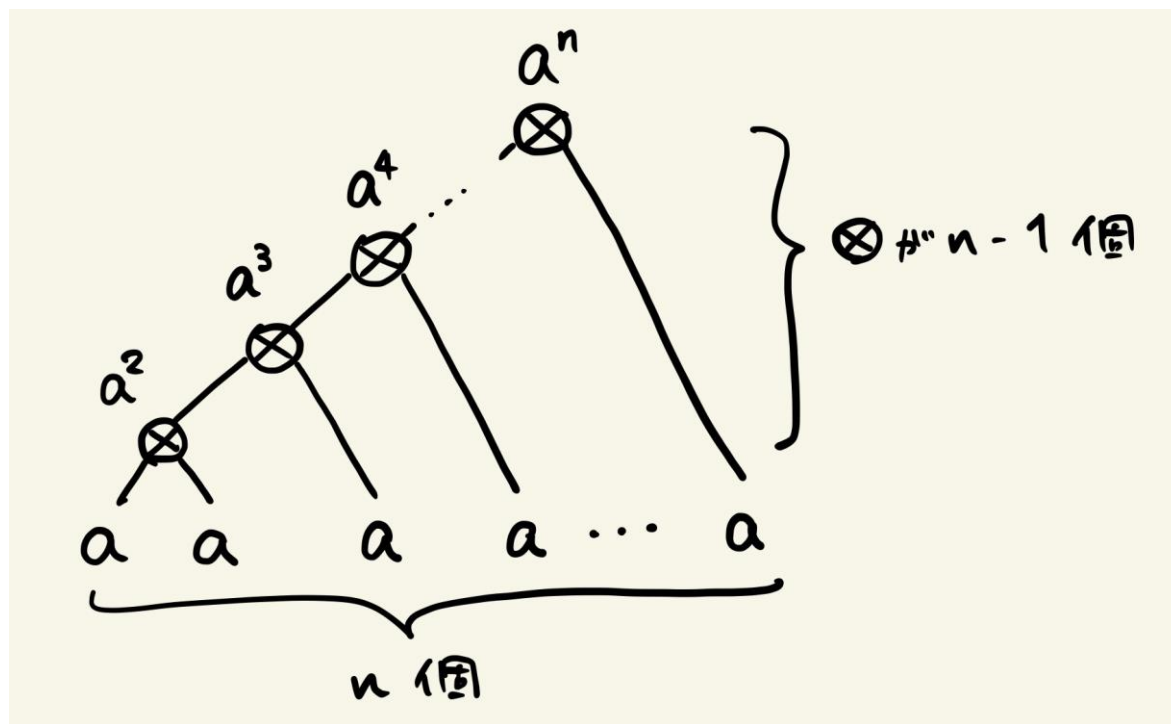
---

- 問題：べき乗計算
  - 入力：2つの正整数  $a$  と  $n$
  - 出力： $a^n$
  - 仮定：許されるのは四則演算のみとする  
(いきなり $n$ 乗するのはダメとする)
- $a^n$ の計算には四則演算が何回必要か？

# 指数演算のアルゴリズム①:

単純な掛け算の繰り返したと**線形時間**がかかる

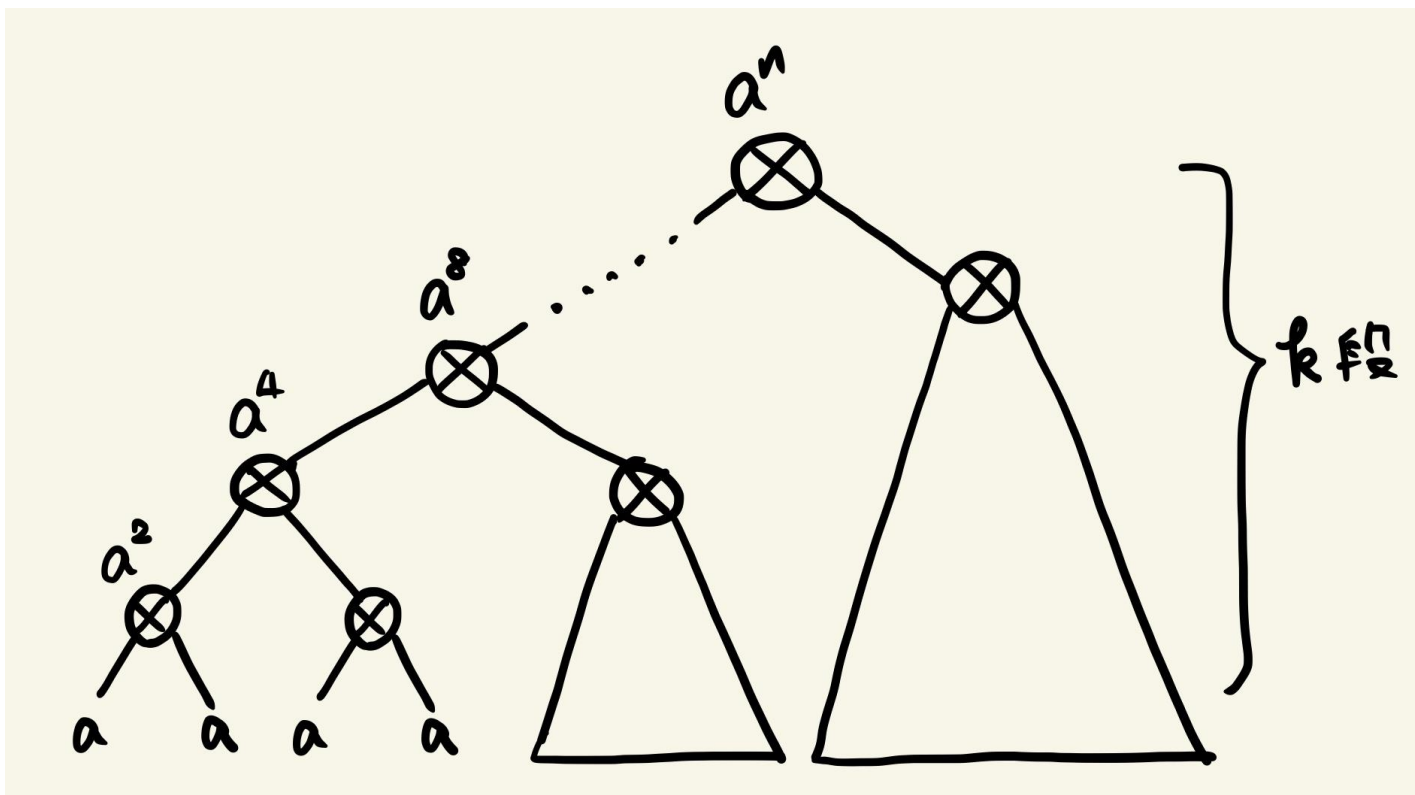
- $a^n = ((\dots((a \times a) \times a) \times \dots) \times a)$  で計算
- $n - 1$ 回の掛け算でできる (演算回数が  $n$  の線形関数)



## 指数演算のアルゴリズム②:

ちょっと工夫すると**対数時間**で計算可能

- 少し工夫すると  $k = \log_2 n$  回の掛け算でできる
  - 仮定:  $n = 2^k$  (この仮定はあとで取り除ける)



## 指数演算のアルゴリズム②（補足）： ちょっと工夫すると**対数時間**で計算可能

- なお、 $n \neq 2^k$ の場合も凡そ  $3\log_2 n$  回の演算で可能
  - $n$ を2進表現する ( $\lceil \log_2 n \rceil - 1 = 5$  回の割り算)
    - 例：  $n = 22 = 10110$
  - 1が立っている桁数に対し2の冪を求める ( $\lceil \log_2 n \rceil - 1$  回の掛け算)
    - 例：  $n = \cancel{2^0} + 2^1 + 2^2 + \cancel{2^3} + 2^4$
  - すべて足す ( $\lceil \log_2 n \rceil - 1$ 回の足し算)

# アルゴリズムの重要性:

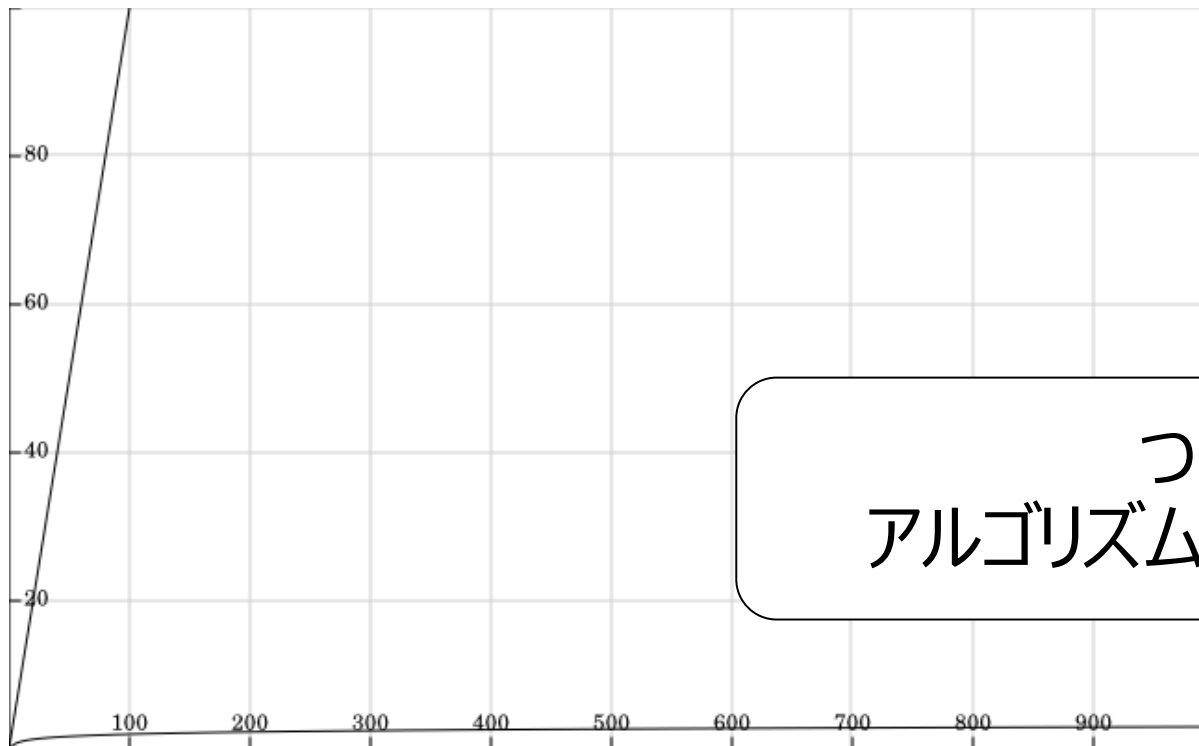
アルゴリズムの工夫で計算効率に大きな差が生じる

■  $n = 1024 = 2^{10}$  のとき、掛け算の回数は

– 解法① では 1023回 (大体  $n$  回)

– 解法② では 10回 (大体  $\log n$  回)

} 指数的な差



つまり  
アルゴリズムはすごく重要

## データ構造の例:

### データに対して繰り返し操作を行う場合に有効

- 前のアルゴリズムの例では1回限りの計算を対象としていた
- データに対して繰り返し計算を行う場合には、予めデータを処理して、うまい構造 (=データ構造) を作っておくことでその後の計算を高速に行えるようになる (ことがある)
- 例えば、これから  $S$  回の計算を行うとして
  - ① (1回限りの計算時間)  $\times S$  回
  - ② (データ構造の構築にかかる計算時間) + (データ構造を利用した1回分の計算時間)  $\times S$で①  $>$  ② となる場合にはデータ構造を考えることが有効

## 具体的な問題例:

### 店舗における顧客情報管理システム

- $n$  人の顧客情報  $\{(n_i, p_i)\}_{i=1, \dots, n}$  が載った名簿を考える
  - $n_i$  : 名前、 $p_i$  : 情報
  - 例 : (元田中 将大, mmototanaka@kyoto-u.ac.jp)
- 客が来るたびに名前を聞いて入力すると、その人の情報が得られるシステムを考える
  - $s$  人分の問い合わせ  $n_{k_1}, n_{k_2}, \dots, n_{k_s}$  が順に与えられる
  - それぞれに対して  $p_{k_j}$  を返す



## 単純なアルゴリズム：

並び順がでたらめな場合は最悪で約 $nS$ 回のチェックが必要

- 名簿の並び順が登録順（つまり、特に規則性なく並んでいる）の場合を考える
- 探索のアルゴリズムとしては、前から順に探していく（しかない）とする
- この場合、各問い合わせで、最悪 $n$ 回のチェックが必要
  - 名簿上の位置（ページ）を指定してチェックすることは単位時間でできるものとする
- 合計 約  $nS$  回のチェックが（最悪ケースで）必要となる

ちょっと頭を使ったアルゴリズム：

辞書順に並んでいる場合は $S \log n$ 回のチェックで可能

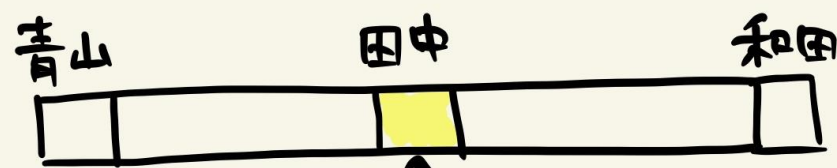
- 予め名簿を辞書順に並べてデータを保存しておくとする
  - そのような「データの持ち方」をする = 「データ構造」を作る
- 問い合わせ名を名簿の「ほぼ真ん中」の人の名前と比較
  - 前者が辞書順で前ならば、目的の人は名簿の前半分にいるはず
  - 今後は前半分だけを調べればよい
  - こんどは前半分の「ほぼ真ん中」の人と比較
  - …
  - 計 $S \log n$ 回のチェックで可能

ちょっと頭を使ったアルゴリズム：

辞書順に並んでいる場合は  $S \log n$  回のチェックで可能

- 計  $S \log n$  回のチェックで発見可能

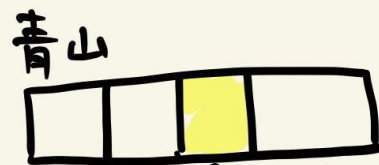
問合せ：木村



ちょうど真ん中

木村 < 田中なので、田中より左に絞る

↓ 探索範囲が 半分 になる



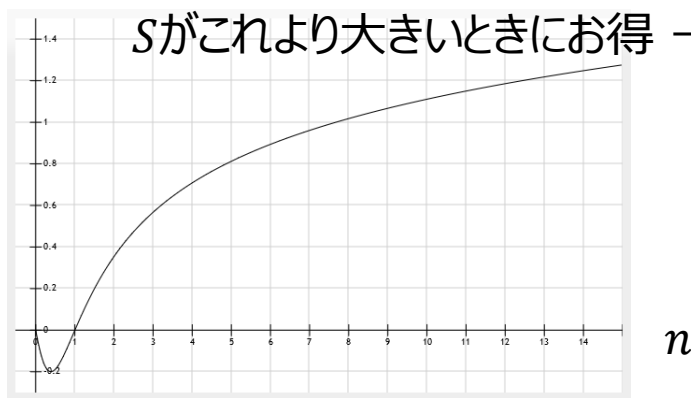
真ん中(加藤)と木村で比較

# データ構造の重要性:

データを正しく持つことで計算コストが大きく削減される

- データ構造 = 「データをどのように管理するか」
- 先ほどの「賢いほうのアルゴリズム」の恩恵にあずかるにはデータが予め整列（ソート）されている必要がある
  - 一般に整列は  $n \log n$  回に比例する演算回数が必要
- よって ①  $nS$  と ②  $n \log n + S \log n$  の比較
  - $S$  が大きくなると②の方がお得になってくる

$\frac{n \log n}{n - \log n}$   $S$ がこれより大きいときにお得  $\rightarrow \frac{n \log n}{n - \log n} = O(\log n)$ 回使えばお得



つまり  
データ構造もすごく重要

まとめ:

## アルゴリズムとデータ構造を学ぶ意義

---

- アルゴリズムはソフト／ハードに（あまり）依存しない、計算機による問題解決の手順書
- データ構造は、データの管理を行うアルゴリズム
- いずれも賢く設計すると、すごく得する  
（逆に、賢くやらないと、すごく損する）