集合の濃度

集合 A, B **に対し、**

$$A \prec B \overset{\Leftarrow}{\Longleftrightarrow} \exists \iota : A \longrightarrow B : \text{ 単射}$$

$$\Longleftrightarrow \exists \pi : B \longrightarrow A : \text{ 全射}$$

（$\Longleftarrow$ **には選択公理が必要）**

$$A \sim B \overset{\Leftarrow}{\Longleftrightarrow} \exists \varphi : A \longrightarrow B : \text{ 全単射}$$

$$\Longleftrightarrow A \prec B \text{ かつ } B \prec A$$

（Bernstein の定理）

$\sim$ **は “集合全体の集まり” の上の “同値関係”**

集合の濃度

A の属する“同値類”：A の**濃度 (cardinality)**
（$\#A, |A|, \mathrm{card}(A)$ 等と書く）

- $\aleph_0 = \mathfrak{a} := \#\mathbb{N}$：**可算濃度**
  (countable, enumerable)
- $\aleph = \mathfrak{c} := \#\mathbb{R}$：**連続体濃度 (continuum)**

濃度の比較：$\#A \leq \#B \overset{\leftarrow}{\Longleftrightarrow} A \prec B$

- $\#A \leq \#A$
- $\#A \leq \#B, \#B \leq \#A \Longrightarrow \#A = \#B$
- $\#A \leq \#B, \#B \leq \#C \Longrightarrow \#A \leq \#C$

（$\leq$ は濃度の間の“順序関係”である）

対角線論法の例：冪集合の濃度

集合 X の冪集合 (power set)

$$\mathcal{P}(X) = \{S | S \subset X\}$$

について、

$$\#X \leq \#\mathcal{P}(X)$$

応用：$\#\mathbb{N} = \#\mathbb{Q} = \aleph_0$（可算濃度）だが、

$$\#\mathbb{R} = \aleph \geq \aleph_0$$ **（連続体濃度）**

注：$\aleph$ は $\aleph_0$ の 次の 大きさ、とは言えない

（連続体仮説）

定理

チューリングマシンで認識可能でない言語が存在する。

- チューリングマシン全体の集合
- 言語全体の集合

の濃度とを比較せよ

定理

チューリングマシンで認識可能で<u>ない</u>言語が存在する。

- **チューリングマシン全体の集合**
- **言語全体の集合**

の濃度とを比較せよ

さて、本講義最後の話題は、

計算量

について

問題の難しさを如何に計るか？

さて、本講義最後の話題は、

計算量

について

問題の難しさを如何に計るか？

Church-Turing の提唱（再掲）

「全てのアルゴリズム（計算手順）は、
チューリングマシンで実装できる」

（アルゴリズムと呼べるのは
チューリングマシンで実装できるものだけ）

. . . 「アルゴリズム」の定式化

計算量 (complexity)

- **時間計算量**：計算に掛かるステップ数
  （TM での計算の遷移の回数）
- **空間計算量**：計算に必要なメモリ量
  （TM での計算で使うテープの区画数）

通常は、決まった桁数の四則演算 1 回を
1 ステップと数えることが多い

入力データ長 n に対する
増加のオーダー（Landau の O-記号）で表す

計算量 (complexity)

- **時間計算量**：計算に掛かるステップ数
  （TM での計算の遷移の回数）
- **空間計算量**：計算に必要なメモリ量
  （TM での計算で使うテープの区画数）

通常は、決まった桁数の四則演算 1 回を
1 ステップと数えることが多い

入力データ長 n に対する
増加のオーダー（Landau の O-記号）で表す

計算量 (complexity)

- **時間計算量**：計算に掛かるステップ数
  （TM での計算の遷移の回数）
- **空間計算量**：計算に必要なメモリ量
  （TM での計算で使うテープの区画数）

通常は、決まった桁数の四則演算 1 回を
1 ステップと数えることが多い

入力データ長 n に対する
増加のオーダー（Landau の O-記号）で表す

Landau の O-記号・o-記号

$f, g : \mathbb{N} \longrightarrow \mathbb{R}_{>0}$ **に対し、**

$$f = O(g) \overset{\leftarrow}{\Longleftrightarrow} \exists N > 0, \exists C > 0 : \forall n : \\ (n \geq N \Longrightarrow f(n) \leq Cg(n))$$

$$f = o(g) \overset{\leftarrow}{\Longleftrightarrow} \frac{f(n)}{g(n)} \longrightarrow 0 \ (n \to \infty) \\ \Longleftrightarrow \forall \varepsilon > 0 : \exists N > 0 : \forall n : \\ (n \geq N \Longrightarrow f(n) \leq \varepsilon g(n))$$

Landau の O-記号・o-記号

$f, g : \mathbb{N} \longrightarrow \mathbb{R}_{>0}$ **に対し、**

$$f = O(g) \overset{\leftarrow}{\Longleftrightarrow} \exists N > 0, \exists C > 0 : \forall n : \\ (n \geq N \Longrightarrow f(n) \leq Cg(n))$$

$$f = o(g) \overset{\leftarrow}{\Longleftrightarrow} \frac{f(n)}{g(n)} \longrightarrow 0 \ (n \to \infty) \\ \Longleftrightarrow \forall \varepsilon > 0 : \exists N > 0 : \forall n : \\ (n \geq N \Longrightarrow f(n) \leq \varepsilon g(n))$$

計算量 (complexity)

問題を解くアルゴリズムによって決まる

$\cdots$ アルゴリズムの計算量

$\longrightarrow$ アルゴリズムの効率 の評価

問題の計算量：

その問題を解くアルゴリズムの計算量の下限

最も効率良く解くと、どれ位で解けるか

$=$ どうしてもどれ位必要か

$=$ どれ位難しい問題か

$\longrightarrow$ 問題の難しさ の評価

計算量 (complexity)

問題を解くアルゴリズムによって決まる

$\cdots$ アルゴリズムの計算量

$\longrightarrow$ アルゴリズムの効率 の評価

問題の計算量：

その問題を解くアルゴリズムの計算量の下限

最も効率良く解くと、どれ位で解けるか

$=$ どうしてもどれ位必要か

$=$ どれ位難しい問題か

$\longrightarrow$ 問題の難しさ の評価

計算量 (complexity)

問題を解くアルゴリズムによって決まる

$\cdots$ アルゴリズムの計算量

$\longrightarrow$ アルゴリズムの効率 の評価

問題の計算量：

その問題を解くアルゴリズムの計算量の下限

最も効率良く解くと、どれ位で解けるか

$=$ どうしてもどれ位必要か

$=$ どれ位難しい問題か

$\longrightarrow$ 問題の難しさ の評価

計算量 (complexity)

問題を解くアルゴリズムによって決まる

$\cdots$ アルゴリズムの計算量

$\longrightarrow$ アルゴリズムの効率 の評価

問題の計算量：

その問題を解くアルゴリズムの計算量の下限

最も効率良く解くと、どれ位で解けるか

$=$ どうしてもどれ位必要か

$=$ どれ位難しい問題か

$\longrightarrow$ 問題の難しさ の評価

基本的な例

- **加法：** $O(n)$
- **乗法：** $O(n^2)$ **かと思いきや** $O(n \log n \log \log n)$
  （高速フーリエ変換 (FFT)）

基本的な例

- **加法：** $O(n)$

- **乗法：** $O(n^2)$ **かと思いきや** $O(n \log n \log \log n)$
  （高速フーリエ変換 (FFT)）

例：互除法

- **入力：正整数** x, y
  入力データ長：
  $$n = \lceil \log_2 x \rceil + \lceil \log_2 y \rceil \sim \max\{\log x, \log y\}$$
- **出力：最大公約数** $d = \gcd(x, y)$

計算量の評価：

- **割算の回数：** $O(n)$
- **1 回の割算：素朴な方法でも** $O(n^2)$
  （FFT を使えば $O(n \log n \log\log n)$**）**

$\longrightarrow$ **併せて** $O(n^3)$ **（FFT で** $O(n^2 \log n \log\log n)$**）**

$\cdots$ **充分に高速なアルゴリズム**

例：互除法

- **入力：正整数** x, y
  入力データ長：
  $$n = \lceil \log_2 x \rceil + \lceil \log_2 y \rceil \sim \max\{\log x, \log y\}$$
- **出力：最大公約数** $d = \gcd(x, y)$

計算量の評価：

- **割算の回数：** $O(n)$
- **1 回の割算：素朴な方法でも** $O(n^2)$
  （FFT を使えば $O(n \log n \log\log n)$**）**

$\longrightarrow$ **併せて** $O(n^3)$ **（FFT で** $O(n^2 \log n \log\log n)$**）**

$\cdots$ **充分に高速なアルゴリズム**