アルゴリズムとデータ構造

第2週目

担当　情報システム部門　徳光政弘
2025年4月16日

今日の内容

- アルゴリズムの定義
- アルゴリズムの評価基準（計算量）
- 計算量の漸近的評価

アルゴリズムの考え方

- アルゴリズム　うまくやるための手順
- 例
  - 料理のレシピ
  - 料理本の手順
  - ゲームの攻略本
  - 作業の指示書

アルゴリズムの考え方

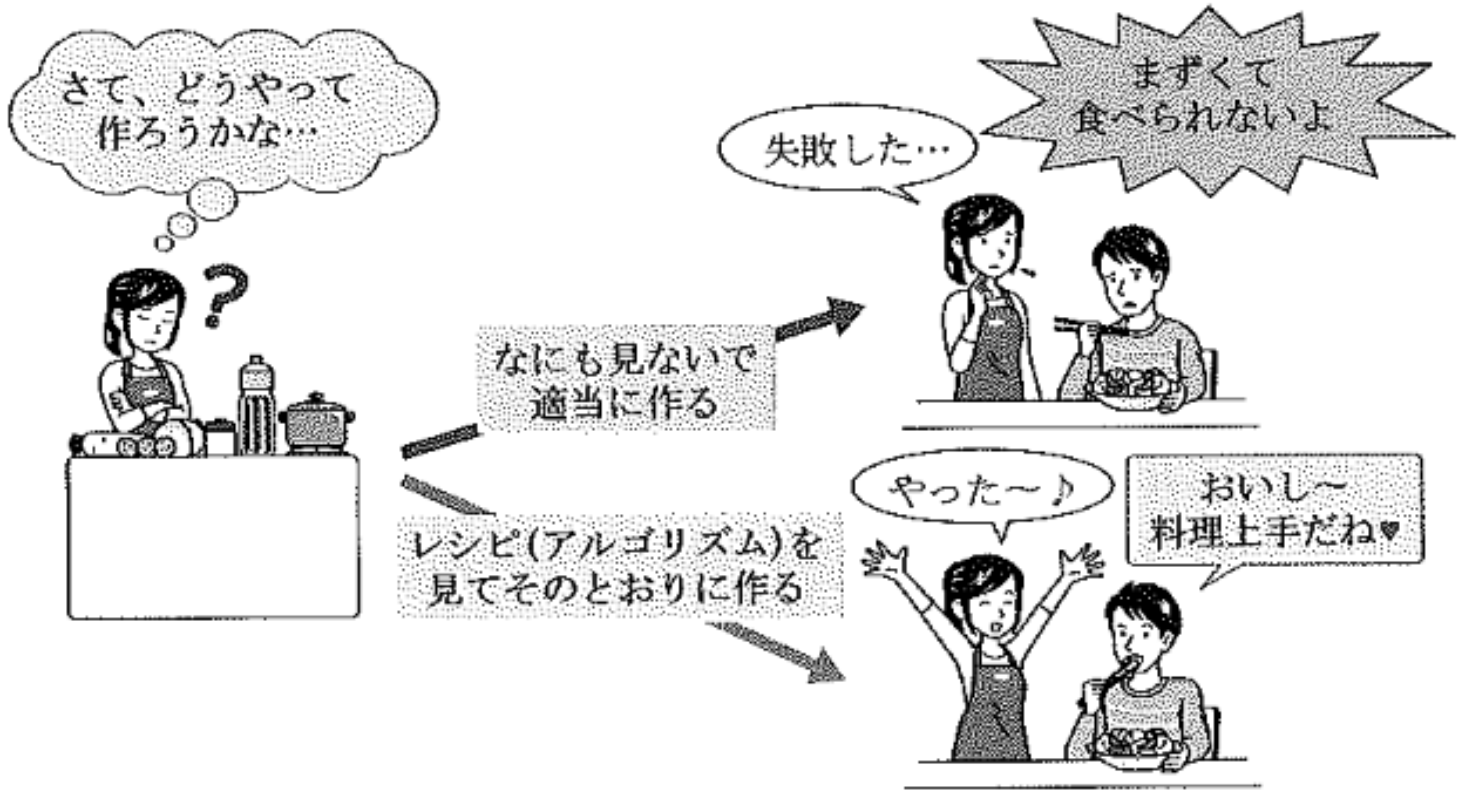


図 1.1　料理におけるアルゴリズム

アルゴリズムの定義

- 教科書の定義
- コンピュータで動作させるためにはプログラミング言語で

◆定義 1.1　アルゴリズム

与えられた問題の正しい答えを求めるための“うまいやり方”であり，一般に文章やプログラミング言語で記述される．

アルゴリズムの例

- 教科書の定義

[問題 1.1]

n 桁の整数が与えられた場合に，その整数が 3 の倍数であるかどうかを答えよ．

“小学校で習った筆算を使って，与えられた n 桁の整数を 3 で割算し，余りが 0 ならば 3 の倍数であると答える．”

アルゴリズムの例

- 例で考えてみる
- 例1　300 / 3 = 100（3の倍数）
- 例2　999 / 3 = 333（3の倍数）
- 例3　1000 / 3 = 333と余り1（3の倍数ではない）
- 極端な例
- 例4 100000000000000000000000000000000
  - 64ビットの整数では表現できなくて単純には計算できない

アルゴリズムの評価基準

- 例　1893206753214

$$1+8+9+3+2+0+6+7+5+3+2+1+4=51$$

- 割り算より単純に足すだけのでわかりやすい
- 長い桁数でも一桁ずつ数値を足せばよい

アルゴリズムの評価基準

- 問題を解くためのアルゴリズムは複数ある場合がある
- どのようにしてアルゴリズムを比較するのかが問題
- コンピュータは「計算する」機械である
- 直感的には「速いが優れている」気もする

アルゴリズムの比較例

- テニスボールの山から不良品のボールを探す

[問題 1.2]

n 個のテニスボールがある．このテニスボール 1 個の重さは 100 g であるが，n 個のうち 1 つだけ重さが 95 g の不良品である．重さが測定できるはかりを用いて，この不良品のテニスボールをみつけよ．

テニスボールの比較

- ボールをb1、b2、…、bnと区別する

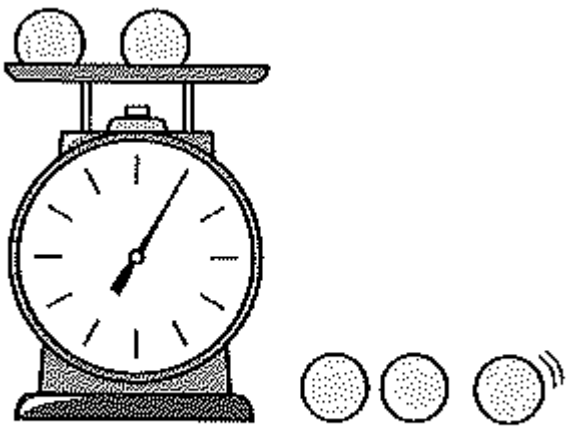


図 1.2　テニスボールとはかり

テニスボールの比較

- 単純なアルゴリズム　そのアイデアとは？

アルゴリズム 1.2

入力：n 個のテニスボール $\{b_1, b_2, \ldots, b_n\}$
アルゴリズム：

① $i = 1$ とする．
② テニスボール b_i をはかりに載せる．
③ b_i の重さが 100 g ならば，i を 1 だけ増加させて②，③の操作を繰り返す．b_i の重さが 95 g ならば，そのボールを不良品としてアルゴリズムを終了する．

テニスボールの比較

アルゴリズム 1.3

入力：n 個のテニスボール $\{b_1, b_2, \ldots, b_n\}$
アルゴリズム：

① テニスボールを約半分ずつの 2 つの集合 $B_1 = \{b_1, b_2, \ldots, b_{\lceil \frac{n}{2} \rceil}\}$, $B_2 = \{b_{\lceil \frac{n}{2} \rceil + 1}, b_{\lceil \frac{n}{2} \rceil + 2}, \ldots, b_n\}$ に分ける[1)].

② テニスボールの集合 B_1 をはかりに載せる.

③ B_1 の重さが 100 の倍数ならば，テニスボールの集合 B_2 に不良品があり，B_1 の重さが 100 の倍数でなければ，不良品は B_1 の中にある．このとき，不良品の含まれているほうのボールの集合に対して，以下の操作を行う.

a. 不良品の含まれているボールの集合に 1 つのボールしかなければ，そのボールを不良品としアルゴリズムを終了する.

b. 不良品の含まれているボールの集合に複数のボールが含まれていれば，そのボールの集合を①と同様に 2 つの集合 B_1 と B_2 に分けて，②，③の操作を繰り返す.

復習　集合と天井関数・床関数

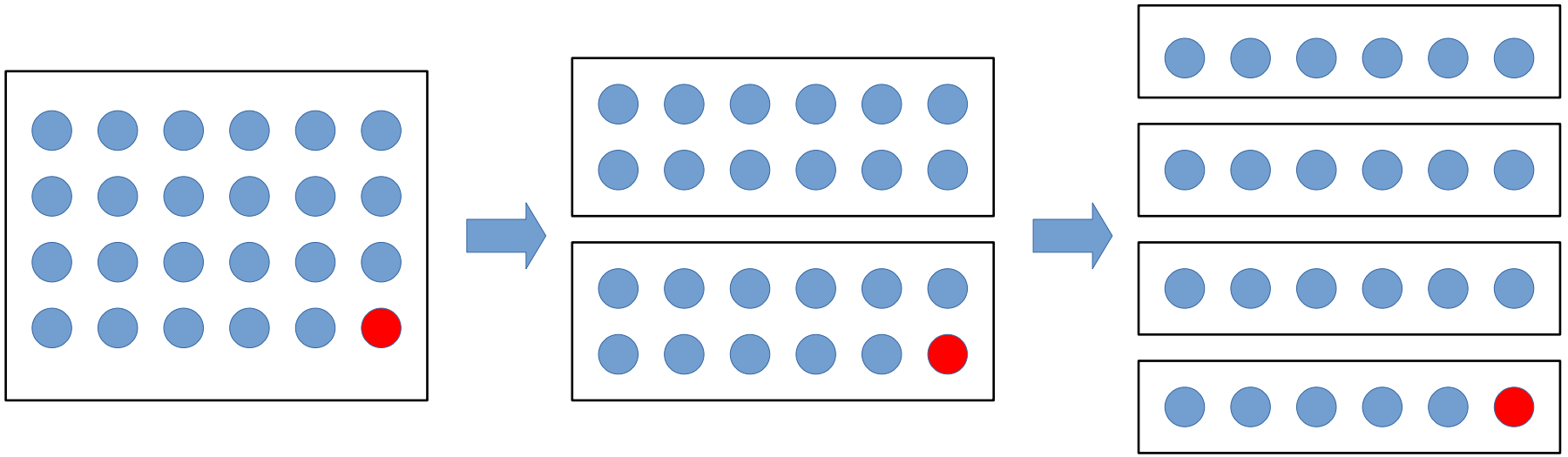
アルゴリズム 1.3

入力：n 個のテニスボール $\{b_1, b_2, \ldots, b_n\}$

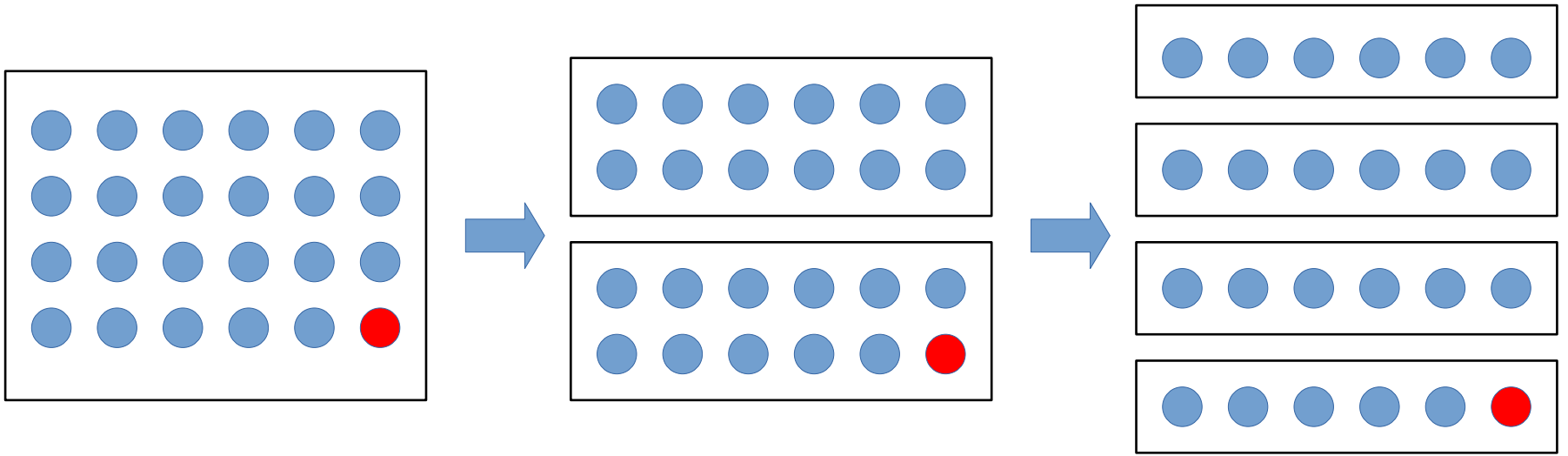
アルゴリズム：

① テニスボールを約半分ずつの 2 つの集合 $B_1 = \{b_1, b_2, \ldots, b_{\lceil \frac{n}{2} \rceil}\}$, $B_2 = \{b_{\lceil \frac{n}{2} \rceil + 1},$

テニスボールの比較　アルゴリズム1.3の考え方



アルゴリズム1.3の考え方　比較回数



比較回数

$$\left(\frac{1}{2}\right)^k \times n$$

終了条件

$$\left(\frac{1}{2}\right)^k \times n = 1$$

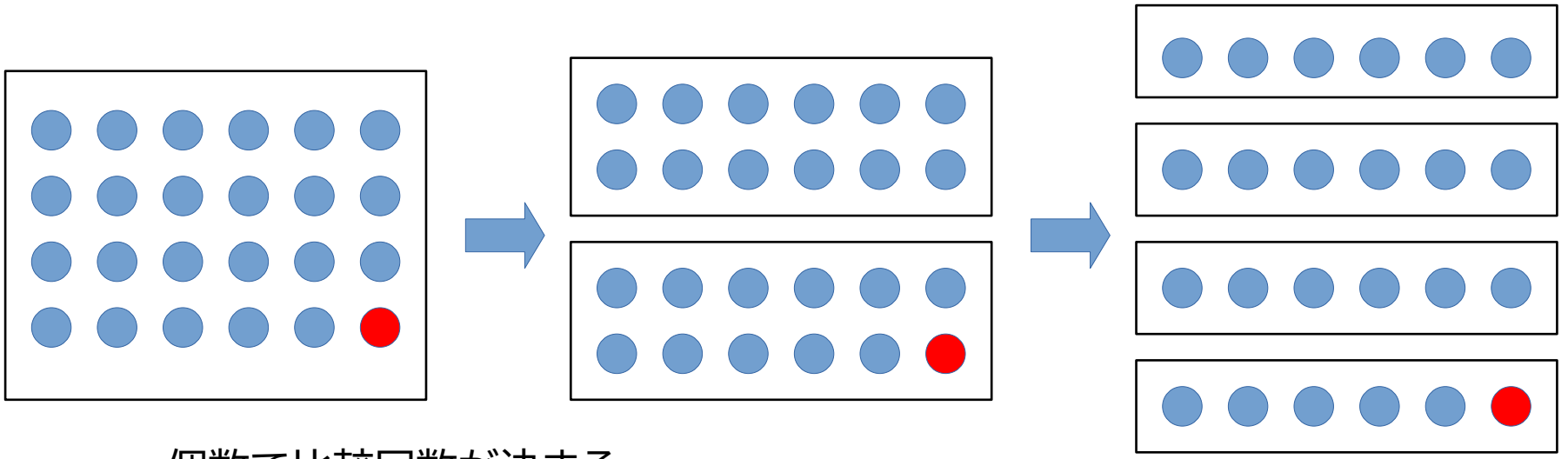
$$\left(\frac{1}{2}\right)^k \times n = 1$$

$$2^k = n$$

$$\log_2 2^k = \log_2 n$$

$$k = \log_2 n$$

アルゴリズム1.3の考え方　比較回数



個数で比較回数が決まる

$$\left(\frac{1}{2}\right)^k \times n = 1$$

$$2^k = n$$

$$\log_2 2^k = \log_2 n$$

$$k = \log_2 n$$

アルゴリズム1.3の考え方　比較回数

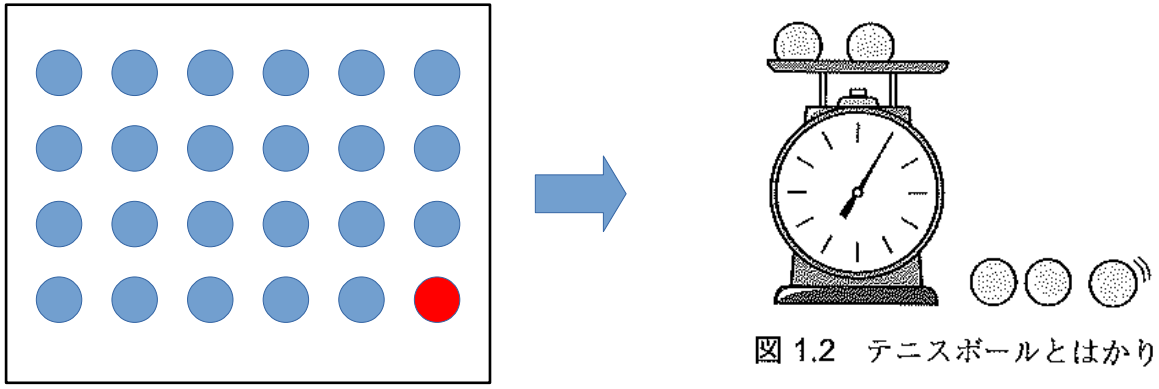


図 1.2　テニスボールとはかり

1回10秒要すると仮定すると、n個の場合は何秒かかるか

アルゴリズム1.3の考え方　比較回数

表 1.1　アルゴリズムの実行時間

テニスボールの数 n	アルゴリズム 1.2		アルゴリズム 1.3
	b_1 が不良品の場合	b_n が不良品の場合	
10	10 秒	100 秒	$10 \times \log_2 10 =$ 約 40 秒
100	10 秒	1000 秒	$10 \times \log_2 100 =$ 約 70 秒
1000	10 秒	10000 秒	$10 \times \log_2 1000 =$ 約 100 秒
10000	10 秒	100000 秒	$10 \times \log_2 10000 =$ 約 140 秒
100000	10 秒	1000000 秒	$10 \times \log_2 100000 =$ 約 170 秒

アルゴリズムの計算量

- 入力の大きさで計算時間が変わる
- 時間計算量　計算時間で比較する
- もっと速いケース　最良時間計算量
- もっとも悪いケース　最悪時間計算量
- 領域計算量　データの記憶容量で比較する

アルゴリズムの計算量

表 1.2　アルゴリズムの時間計算量

	アルゴリズム 1.2	アルゴリズム 1.3
最良時間計算量	10	$10 \log_2 n$
最悪時間計算量	$10n$	$10 \log_2 n$

計算量の漸近的評価

- 計算量の比較を考える
- 入力のサイズ「n」と仮定する
  - データの個数、ボールの個数を想像するとよい

アルゴリズム A ：$10n^2 + 100n + 10000$

アルゴリズム B ：$n^4 - n^3 - n$

アルゴリズム C ：$100n^3$

計算量のオーダー記法

このオーダ記法の定義は少々わかりにくいので，以下のような理解で十分である．まず，アルゴリズムの時間計算量を入力サイズ n を用いた関数として求める．つぎに，その関数のなかで主要項(n が無限大に近い場合にもっとも大きな項)をみつける．この主要項の係数を削除した関数が $f(n)$ であるとき，“アルゴリズムの時間計算量は $O(f(n))$ である”，もしくは“アルゴリズムは $O(f(n))$ 時間で実行できる”という．ア

アルゴリズム A ： $10n^2 + 100n + 1000$

オーダー記法 $O(n^2)$

計算量の漸近的評価

- テニスボールの計量の例

表 1.3　アルゴリズムの漸近的な時間計算量

	アルゴリズム 1.2	アルゴリズム 1.3
最良時間計算量	$O(1)$	$O(\log n)$
最悪時間計算量	$O(n)$	$O(\log n)$

計算量の比較

$$\log n \ ^{1)} < \sqrt{n} < n < n \log n < n^2 < n^3 < \cdots < 2^n < n!$$

なお，関数 $f(n)$ が n に依存しない定数である場合は，その時間計算量を特別に $O(1)$ と記述し，その時間計算量を**定数時間**とよぶ．

計算量の比較

表 1.4　アルゴリズム A，B，C の実行時間

入力サイズ (n)	アルゴリズム A $(10n^2+100n+10000)$	アルゴリズム B (n^4-n^3-n)	アルゴリズム C $(100n^3)$
10	12000	8990	100000
100	1210000	98999900	100000000
1000	10110000	9.99×10^{11}	10×10^{10}
10000	1001010000	9.99×10^{15}	10×10^{13}
100000	10×10^{10}	9.99×10^{19}	10×10^{16}

アルゴリズムの記述

- 教科書の「アルゴリズムの記述」の節を参照

計算量の例

- 教科書の「アルゴリズムの記述」の節を参照

アルゴリズム **1.4**　最大値の計算

入力：n 個の整数 x[1], x[2], ..., x[n]

```
max=x[1];
for (i=2; i<n+1; i=i+1) {
  if (max<x[i]) { max=x[i]; }
}
```

計算量　$O(1) \times (n-1) = O(n)$

計算量の例

アルゴリズム 1.5　等しい整数の出力

```
入力：n 個の整数 x[1], x[2], ..., x[n]
for (i=1; i<n; i=i+1) {
  for (j=i+1; j<n+1; j=j+1) {
    if (x[i]==x[j]) { x[i]とx[j]は同じであると出力; }
  }
}
```

計算量

$$\sum_{i=1}^{n-1}(n-i)\times O(1) = O(1)\times\sum_{i=1}^{n-1} i$$

$$= O(1)\times\frac{n(n-1)}{2}$$

$$= O(n^2)$$