cs-15. コンピューターサイエンスの概念の広がり

（コンピューターサイエンス）

金子邦彦

資料中の図などは Google Nano Banana 2 を用いて作成

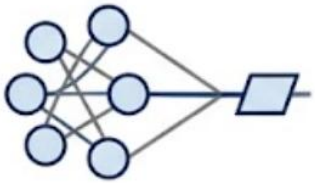
「コンピューターサイエンス」第15回の内容

Database Lab.

対象を抽象化し、数式や図として表現し、計算の仕組みとして扱う

抽象化

目的に応じて対象の本質を抽出する



モデル化

数式・図・プログラムなどの形式で表現する

計算モデル

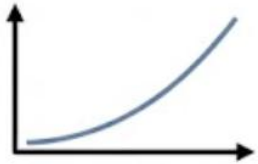
入力・記憶・操作・結果に注目する

例：有限オートマトン、チューリングマシン

アルゴリズムと計算量

アルゴリズムの選択で処理の手間が変わる

入力の量が増えると手間も増える。この増え方（計算量）の把握が重要



数値計算における丸め

入力・内部表現・表示のそれぞれの段階で丸めが生じる

15-1. 抽象化・モデル化・計算モデル

現実の対象をどう単純化するか？

- 抽象化には目的（何に注目するか）に応じて、本質を抽出

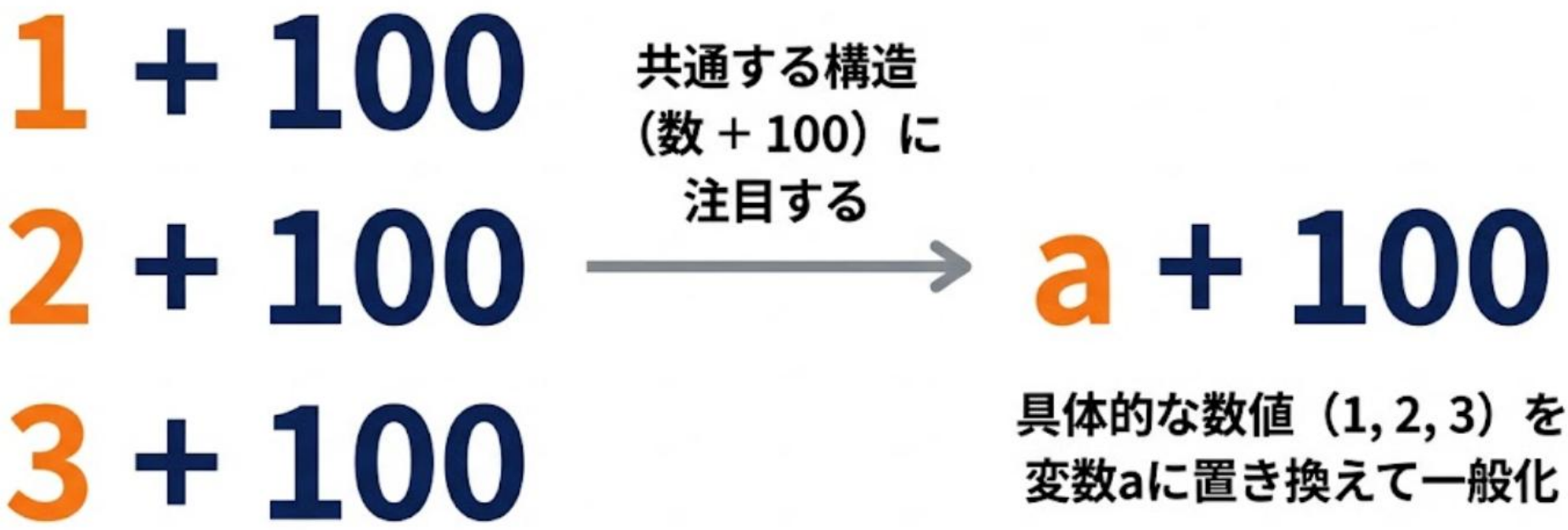
  （例）**1+100, 2+100, 3+100 ⇒ a + 100**

  （例）**様々な自動販売機 ⇒ 投入金額が１５０以上かに注目**

- 計算の枠組み（有限オートマトン、チューリングマシン）にも触れる

抽象化

具体的な違いを無視し、共通の構造を抜き出すプロセス（抽象化）



抽象化の例

[1] 目的による性質の選別

自動販売機がもつ機能

おつりの計算
在庫管理
返金
商品選択
故障対応
投入金額の判定

目的達成のための抽象化

目的：投入金額が150円以上かどうかを判定する
↓
『投入金額の判定』に注目

[2] 具体的な値の一般化

金額としては異なるが、150円以上という点では同じ

150円未満（個別）			150円以上	
0円	50円	100円	150円	200円

[3] 抽象化によって失われる情報

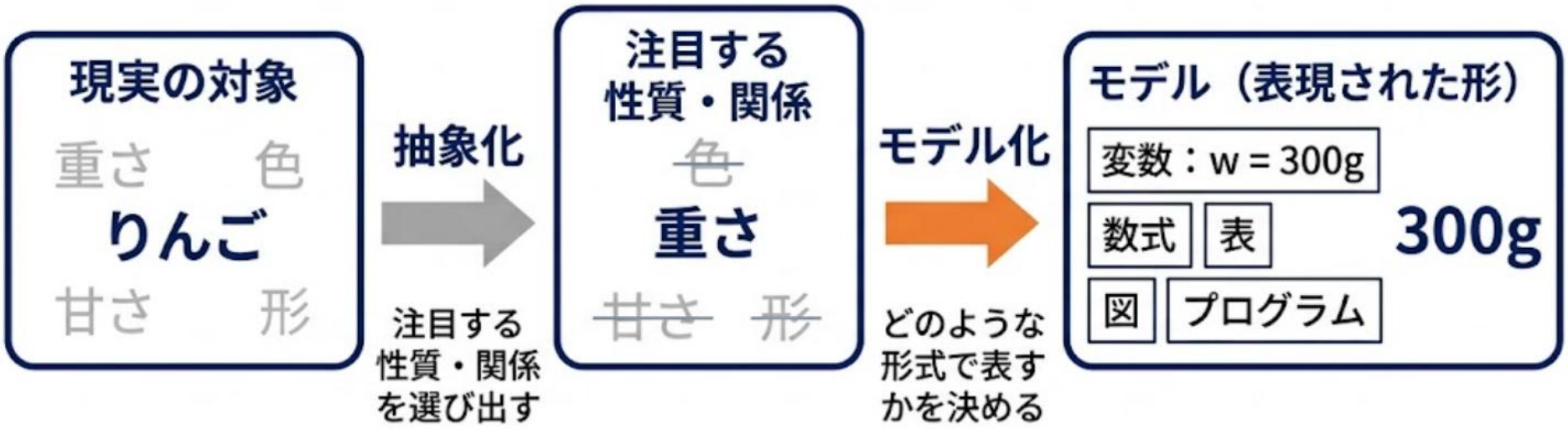
『150円以上』という区分だけを見てくても、実際の投入金額が150円か200円かは分からない

[4] 抽出された共通の本質

『投入金額の判定』

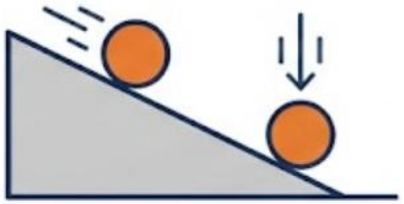
モデル化

モデル化とは、注目する性質や関係を、何らかの形式で表すこと



現実を数式でモデル化する例

モデル化とは、注目した性質や関係を、変数・数式・図などの形式で表すことである

現実の対象	抽象化：注目する性質・関係を選び出す	表現形式
物体の運動 	位置・速度・時間の関係に注目	微分方程式で表す $\frac{d^2x}{dt^2} = -g$ 
窓口に並ぶ行列 	到着間隔・待ち時間・人数に注目	数式（待ち行列モデル）で表す $W = \frac{1}{\mu - \lambda}$ 

形式を決める作業

表現形式の例：変数、状態、数式、表、図、プログラム

計算モデル

計算モデルとは、"計算をどのような仕組みとして考えるか"を定めた共通の枠組み

入力の扱い方

何をどのように読み込むか

記憶の持ち方

情報をどこにどう保持するか

計算モデル

あらゆるコンピュータ・あらゆる計算に共通する仕組みの捉え方

許される操作

どんな処理・計算が可能か

結果の読み取り方

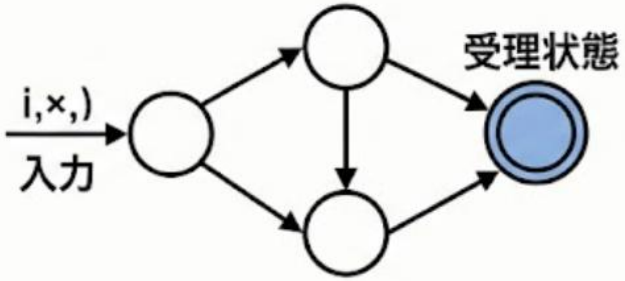
答えをどう取り出すか

- 有限オートマトン
- チューリングマシン
- ラムダ計算

計算モデル：有限オートマトンとチューリングマシン

有限オートマトンとチューリングマシンの計算結果の判定方法を比較する

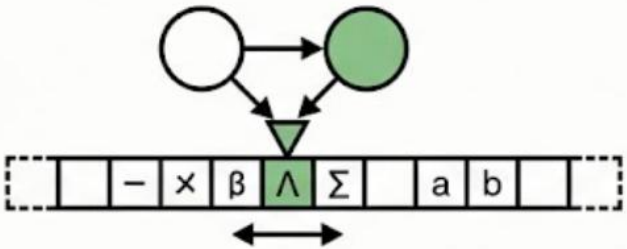
有限オートマトン



入力を読み終えたときに
受理状態にいるかどうかで判定する

Yes / No を判定する問題に向いた単純な計算モデル

チューリングマシン



停止するかどうか、そして停止したときに
テープに何が書かれているかで判定する

必ずしも停止するとは限らない。計算結果そのものを出定力できる、より強力なモデル

有限オートマトンは『状態』だけで判定し、
チューリングマシンは『テープの書き込み』と『停止』で判定する

計算モデルごとの違い

同じ"4つの観点"で見ると、計算モデルごとの違いがはっきり分かる

	有限オートマトン	チューリングマシン
入力	一方向に読むだけの文字列	自由に読み書きできるテープ
記憶	状態だけ（限られた記憶）	テープに自由に書き込める記憶
操作	決まった状態遷移のみ	読む・書く・移動を 自由に組み合わせ
結果	受理／拒否の判定のみ	テープ上に自由な形で出力

限界：チューリングマシンは停止しないことがある

計算モデルと現実との関係

チューリングマシン

パソコン・スマホ本体の計算のしくみ

パソコン
スマートフォン

テープに自由に読み書きできる仕組みに
近く、多様なアプリ・処理を実行できる

有限オートマトン

機械の中の"一部分"を支えるしくみ

- 文字列検索（正規表現）
- 自動販売機の制御
- 通信の状態管理
- エレベーターの制御

決まったパターンの繰り返し処理を、
少ない記憶で確実にこなす

"何でもできる大きな脳"と"決まった仕事を確実にこなす部品"、それぞれに向き不向きがある

チューリングマシンとパソコン・スマートフォンの関係

セルが際限なく続くと仮定する

↓ 現実には有限が、桁違いに大きい

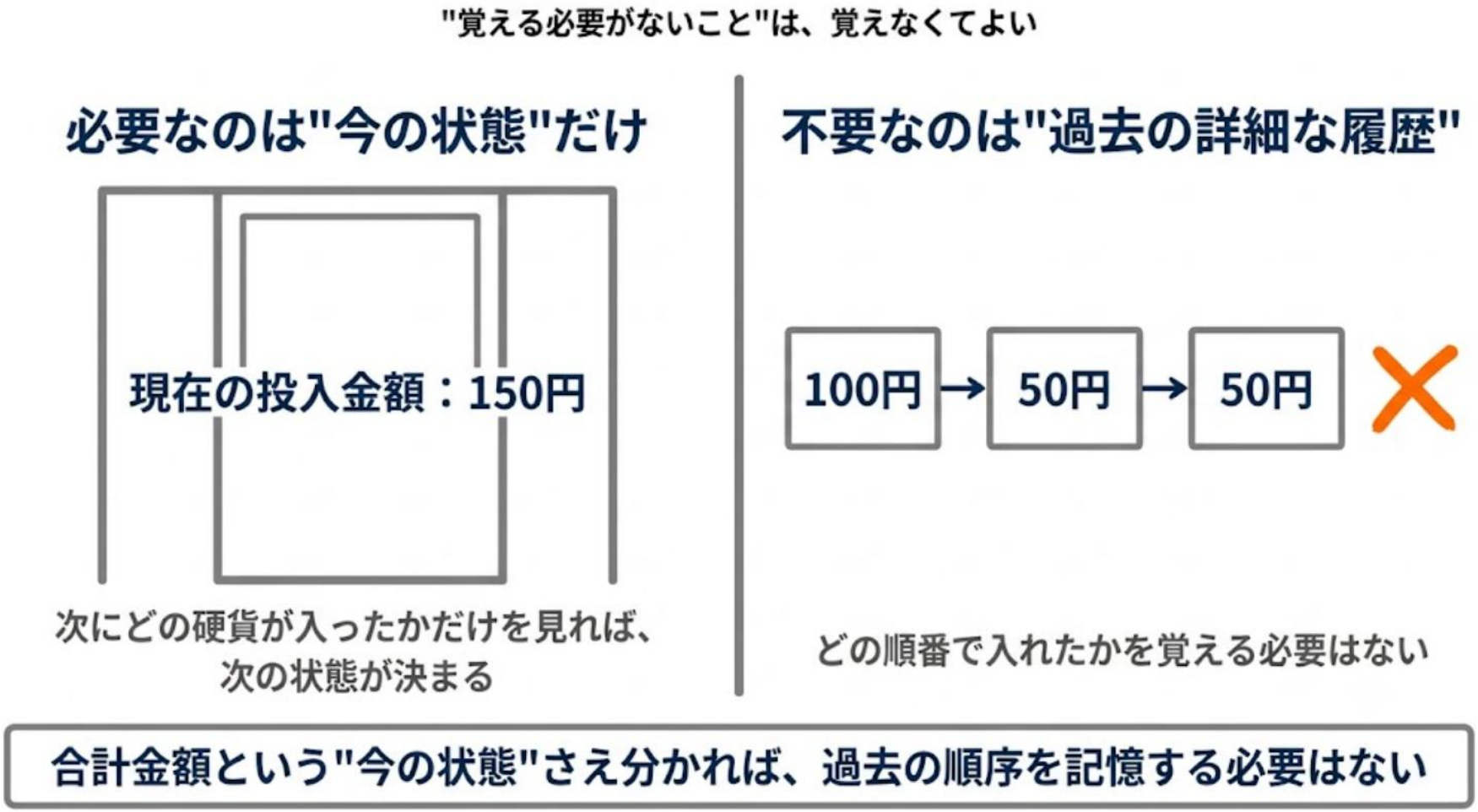
パソコン・スマートフォンの実際のメモリ

メモリ容量	セルの数の目安	実用上の扱い
数GB～数十GB	数百億個以上	事実上無限とみなせる

↓

だから"パソコン＝チューリングマシンの近似"として扱える

有限オートマトンと現実の機械の関係



「実際の機械は、全体としてはチューリングマシンの近似とみなし、その中の特定の制御部分（自動販売機やエレベーター等）を有限オートマトンとしてモデル化する、という使い分けをする」

記憶の持ち方

"記憶"という観点を、もう一段くわしく言うと"状態"になる

記憶の持ち方（これまでの4観点のひとつ）
情報をどこにどう保持するか

もう一段くわしく言うと

有限オートマトンの"状態"

内部状態

機械が今どの段階にいるかを表す、
有限個の内部状態のみ

チューリング機械の"状態"(構成)

内部状態 | テープの内容 | ヘッドの位置

まとめて"構成"と呼ぶ

この3つをすべて含めて、
その時点の計算状況を表す

"状態"と言うとき、内部状態だけを指すのか、
構成全体を指すのか、区別して使う必要がある

15-2. アルゴリズムと計算量

問題を解く手順をどう設計し、その手間をどう測るか

- アルゴリズム（問題を解くための定められた手順）によって手間が違う
- データ構造（例：整列済みか）によって、利用できるアルゴリズムが変わる場合がある
- 探索の手間と、探索の結果が正しいかを確認する手間は大きな違いがある場合が多い

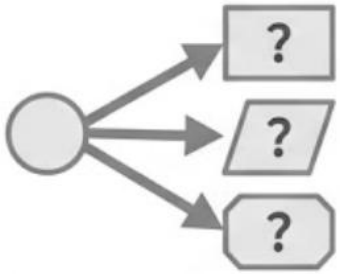
アルゴリズム

アルゴリズムとは、問題を解くための、明確に定められた手順のことである。

アルゴリズムの構造

なぜ明確な手順が必要か

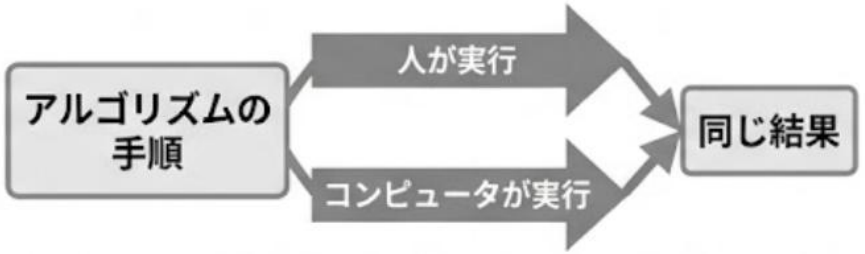
手順があいまい



手順が明確

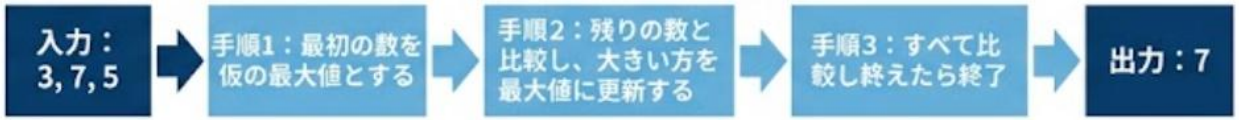
手順があいまいだと、人によって結果が変わったり、機械が実行できなかったりする

何の役に立つか



誰が実行しても、何度実行しても、同じ結果が得られる（再現性・自動化）

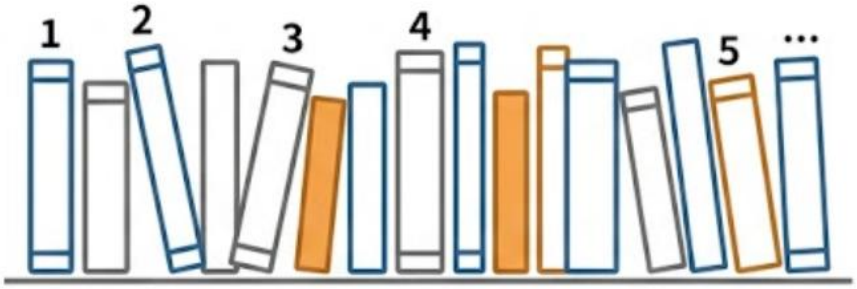
実例：３つの数字の中から最大値を見つける手順



本の並び方による探索の手間の違い

本の並び方によって、目的の一冊を探し出す手間は大きく変わります。

本がバラバラに並んでいる場合



目的の本が見つかるまで、
最悪すべての本を確認する必要がある

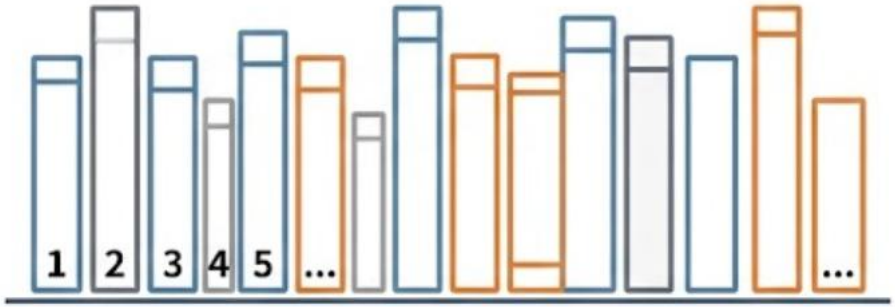
本があいうえお順に並んでいる場合

順序があると、規則性を利用して
手間を大きく減らせる

探索のアルゴリズムによる手間の違い

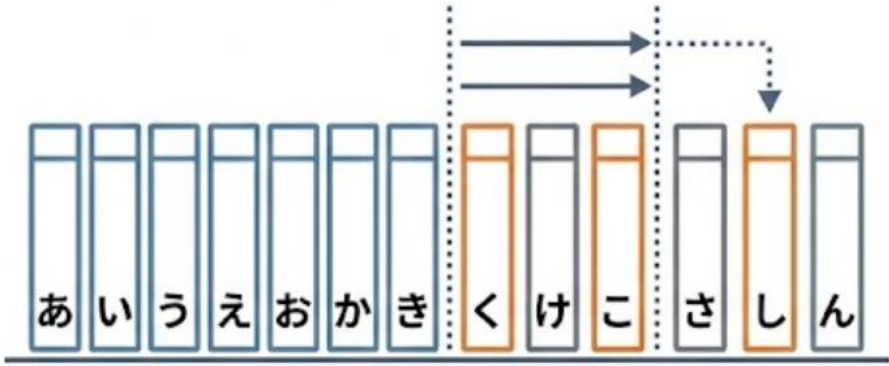
探す手間は手順（アルゴリズム）で変わり、データの並び方で使える手順が決まる

バラバラな本棚では:1冊ずつ確認する(線形探索)



順番に1冊ずつ確認するしかなく、
最悪すべての本を調べる必要がある

あいうえお順の本棚では:半分ずつ絞り込める(二分探索



中央の値と目的の値を比較し、大小関係
により探索範囲を半分に絞り込める

バラバラなデータ → 1冊ずつ確認する方法(線形探索)しか使えない

整列したデータ → 半分ずつ絞り込む方法(二分探索)も使える

同じ『探す』という仕事でも、手順(アルゴリズム)が違えば手間が変わり、
データの並び方によって使える手順そのものが変わる

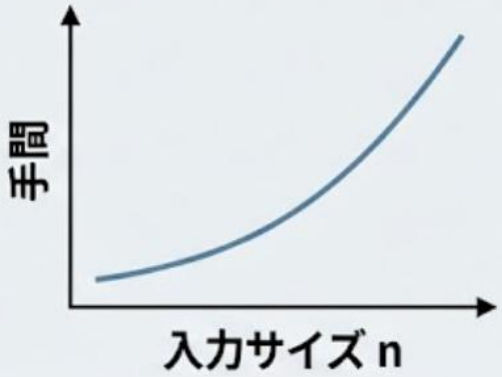
計算量

Database Lab.

計算量は、入力データが大きくなった時の処理の手間を予測する基準です。

nが増えると手間はどう増えるか

入力サイズ n → 手間（処理量）



計算量とは、入力サイズnに応じて必要な手間がどのように増えるかを表す考え方

- 比較回数
- 足し算（演算）の回数
- 実行時間
- 使用メモリ量

- 最悪の場合
- 平均的な場合
- 最良の場合

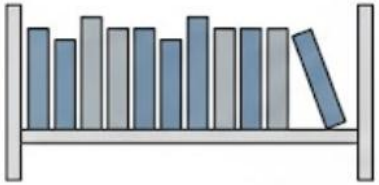
ふつうは、最悪の場合の手間の増え方に注目する

計算量の重要性

データの規模が大きくなるほど、手順の違いが致命的な差を生むため、計算量による事前予測が重要である

本の冊数が少ないとき

本が10冊のとき



1冊ずつ確認しても → すぐに見つかる

半分ずつ絞り込んでも → すぐに見つかる

冊数が少ないうちは、どちらの方法でも
ほとんど差を感じない

本の冊数が非常に多いとき

本が100万冊のとき

1冊ずつ確認すると → 見つかるまでに膨大な時間がかかる

半分ずつ絞り込むと → 短時間で見つかる

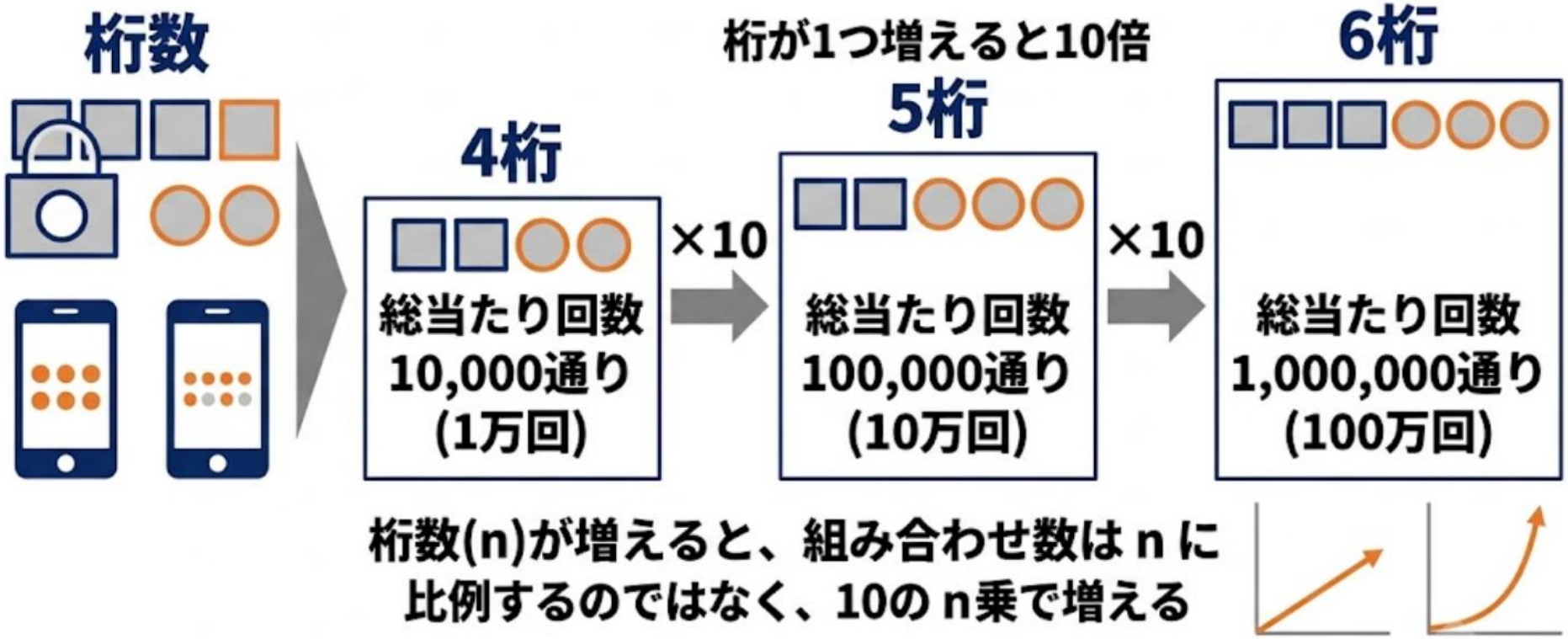
冊数が多くなると、手順の違いが致命的な差になる

実際に動かして困る前に、計算量という考え方を使えば、
規模が大きくなったときにどれくらい手間がかかるかを事前に予測できる

これが、計算量を学ぶ意味である

計算量が急激に増える場合

桁数が増えるだけで、総当たり攻撃の手間が急激に増える仕組み



桁数(n)が増えると、組み合わせ数はnに比例するのではなく、10のn乗で増える

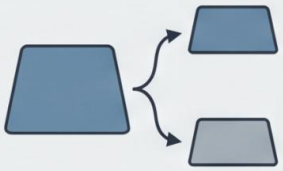
桁数を増やすだけで、総当たり攻撃に必要な試行回数が爆発的に増え、パスワードや鍵の安全性が大きく高まる

8桁のパスワードは危険かもしれない。10文字、12文字に増やすと安全

分割統治法

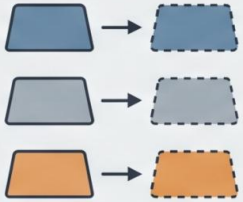
大きな問題を分割して解決する『分割統治法』の考え方を、二分探索という例で理解する

1.『分割』



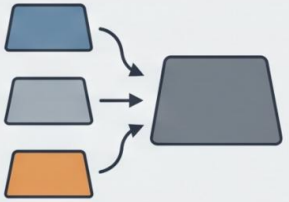
大きな問題を扱いやすい
小問題に分ける

2.『解決』

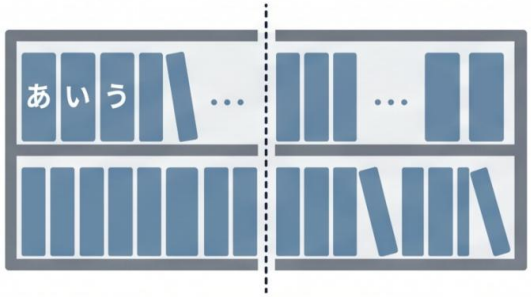


それぞれの小問題を
個別に解決する

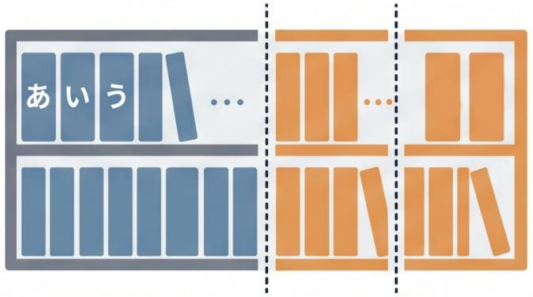
3.『統合』



小問題の解決結果を組み合わせて、
全体の解を導き出す



本棚を半分に分ける



目的の本がある側を選び、さらに半分に分ける
目的の本が無い側は、探索範囲から除外

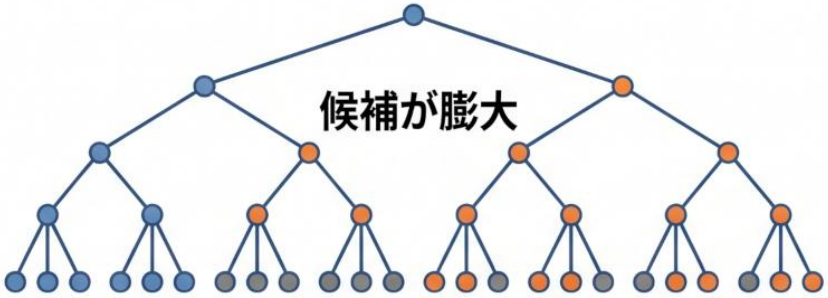
目的の本が1冊見つかる

二分探索は、本棚を半分に分け、目的の本がある側だけを選び、それを繰り返して答え(目的の本)にたどり着く、分割統治法の一例である

探索の手間と、探索の結果が正しいかを確認する手間の違い

同じ問題でも、答えを探す作業と、答えを確かめる作業では、必要な手間が大きく異なることがある

探索：条件を満たす答えを見つける作業



候補をひとつずつ確かめる必要がある

検証：与えられた候補が正しいか確かめる作業

候補 → 確認 → ○か×

候補が1つ与えられれば、それを確かめるだけでよい

{3, 7, 2, 9, 5}から数字を選び、足して15にできるか

目標値15と一致するか、
一つずつ確認

{3, 7, 5} → 3 + 7 + 5 → 15

合計を計算し、目標値と比較するだけでよい

この性質は、計算の理論とも関係する

すべての問題でこの差が大きいわけではない。総当たりが遅いからといって、他の解き方も必ず遅いとは限らない

この性質は、P対NP問題と呼ばれる、計算理論における未解決問題と関係する

15-3. 丸めと計算誤差

コンピュータが数値をどこまで正確に扱えるか

丸めは『いつ・どこで』生じるかによって3種類に区別される。以降のスライドでは、具体的な数値例（1/3の計算や 0.1）を用いてこの3種類の違いを追う

丸めとは、値を扱える桁数に近似する処理であり、いつ・どこで起こるかを区別することが重要である

入力値の丸め

人間やプログラムが、桁数を指定して明示的に行う丸め処理

内部表現による誤差

コンピュータが値を2進数で保持する際に生じる、意図しない微小な誤差

表示上の丸め

画面表示のためだけに行う丸めであり、内部に保存された値そのものは変化しない

これら3つは別々の現象であり、混同してはならない

表現誤差

10進数の小数は、2進数では正確に表せないことがある

原因（原因）

10進数：0.1

↓

2進数：
0.0001100110011...
（無限に続く）

有限桁で打ち切り

10進数では有限桁の小数が、2進数では無限桁になることがある

結果（結果）

本来の値：0.1

≠

実際に保存される値：
0.1000000000000555
11115123125782 7...

近い別の値として保持される
（表現誤差／丸め誤差）

値 → 計算 → 計算 → 誤差拡大

3つの丸めの違い

入力値の丸め
人間が入力する際に生じる丸め

▼

内部表現による誤差（表現誤差）
浮動小数点数の内部保存形式の制約によって生じる誤差

▼

表示上の丸め
画面表示のために小数第2位などに丸めて見せるだけで、内部に保存された値そのものは変化しない

↳ 内部の値は変わっていない

演習

演習1

計算モデルのチューリングマシンを実際に体験するもの

```
開始: 状態=q_right, 位置=0, テープ=1 1 1 _ ...
状態=q_right, 位置=1, テープ=1 1 1 _ ...（右へ進んだ）
状態=q_right, 位置=2, テープ=1 1 1 _ ...（右へ進んだ）
状態=q_right, 位置=3, テープ=1 1 1 _ ...（右へ進んだ）
状態=q_halt, 位置=3, テープ=1 1 1 1 _ ...（1を書いて停止）
最終テープ: 1 1 1 1 _ ...
```

「計算とは」、「計算できるとは」の謎の解明に役立ってきたもの。体験をお願い

演習 2

アルゴリズムの違いによって手間が違うという事実の確認

```
線形探索: (13, True)
二分探索: (4, True)
```

演習３

コンピュータでも解けないほど手間がかかる問題があるという事実

〇但し、解いた結果が正しいかの確認は手早く確実にできる

→　実は暗号やパスワードの基礎にもなっている

演習４

「丸め」があり、コンピュータでの計算は完ぺきに正確というわけではない

- 正常動作
- 普段は気づかない
- わずかな誤差？と思うことも将来あるかも知れない。これは、コンピュータの特質によるもの。故障やミスではない