理論計算機科学特論 (2026 年前学期)

計算複雑性の基礎

第１回

計算理論の復習

岡本 吉央　(電気通信大学)

okamotoy@uec.ac.jp

2026 年 4 月 7 日

最終更新：2026 年 4 月 8 日　09:57

講義の概要

概要

理論計算機科学 における１つのトピックを集中的に学ぶ

理論計算機科学 = Theoretical Computer Science

今年度のトピック

計算複雑性の基礎

計算複雑性理論 = Computational Complexity Theory

(計算複雑さの理論，計算量理論 とも)

主に，1990 年代初頭までの **基礎的内容** を扱う

講義の目標

次の 2 つができるようになること

目標 1

計算複雑性 に関わる **用語・概念** を
正しく使う ことができるようになる

目標 2

問題解決の難しさ を **計算複雑性** の視点から議論し
今後の研究や日常生活で使えるようになる

理論計算機科学？

論文誌 *Theoretical Computer Science* によると

- *Theoretical Computer Science* is mathematical and abstract in spirit, but it derives its motivation from practical and everyday computation. Its aim is to understand the nature of **computation** and, as a consequence of this understanding, provide more efficient methodologies. All papers introducing or studying mathematical, logic and formal concepts and methods are welcome, provided that their motivation is clearly drawn from the field of **computing**.
- 3 つのセクション
  1. Algorithms, automata, complexity and games
  2. Logic, semantics and the theory of programming
  3. Natural computing

計算可能性 と 計算複雑性

計算可能性

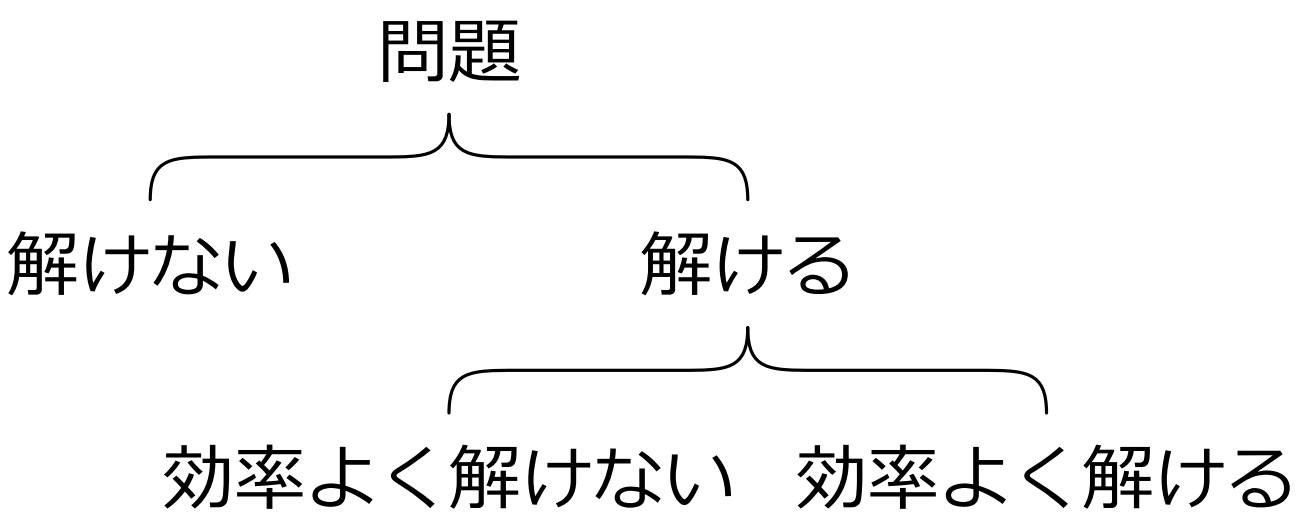
基本的な問い
- 何を計算できるか？

Computability

計算複雑性 (計算量)

基本的な問い
- 計算に必要な資源は？

Computational complexity



計算可能性 と 計算複雑性

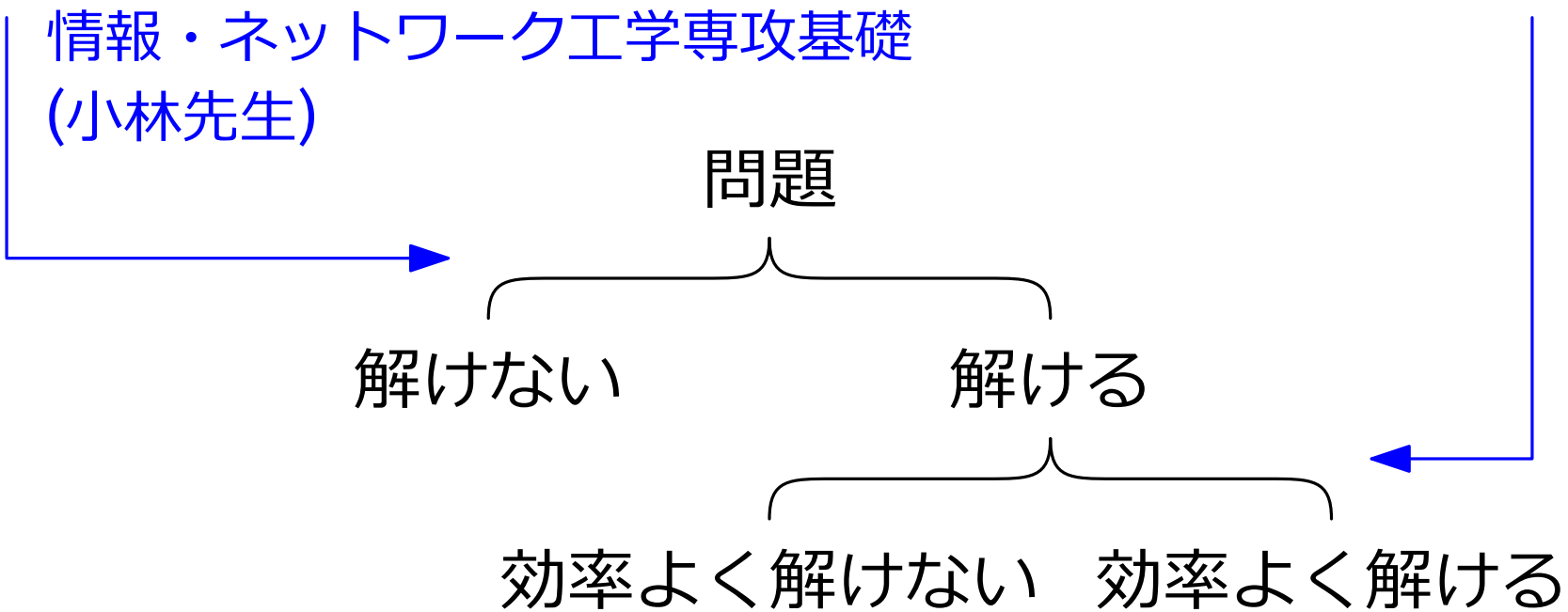
計算可能性

基本的な問い
- 何を計算できるか？

計算複雑性 (計算量)

基本的な問い
- 計算に必要な資源は？

Computability　　Computational complexity



計算複雑性における課題

課題 1

計算における **資源** とは何か？

$\rightsquigarrow$ 時間，領域，非決定性，ランダム性，インタラクション

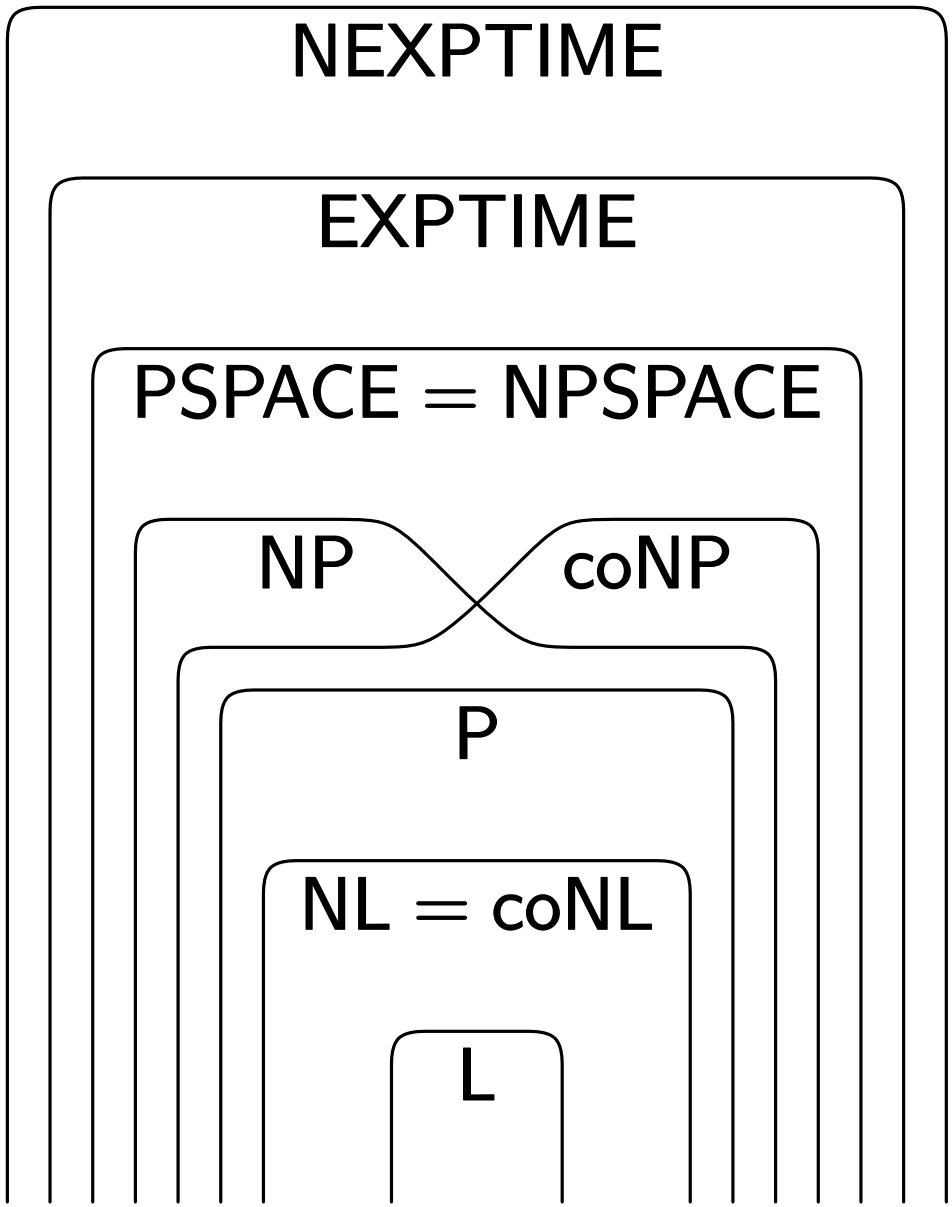
課題 2

資源どうしに **関係** はあるか？

$\rightsquigarrow$ 計算複雑性クラス

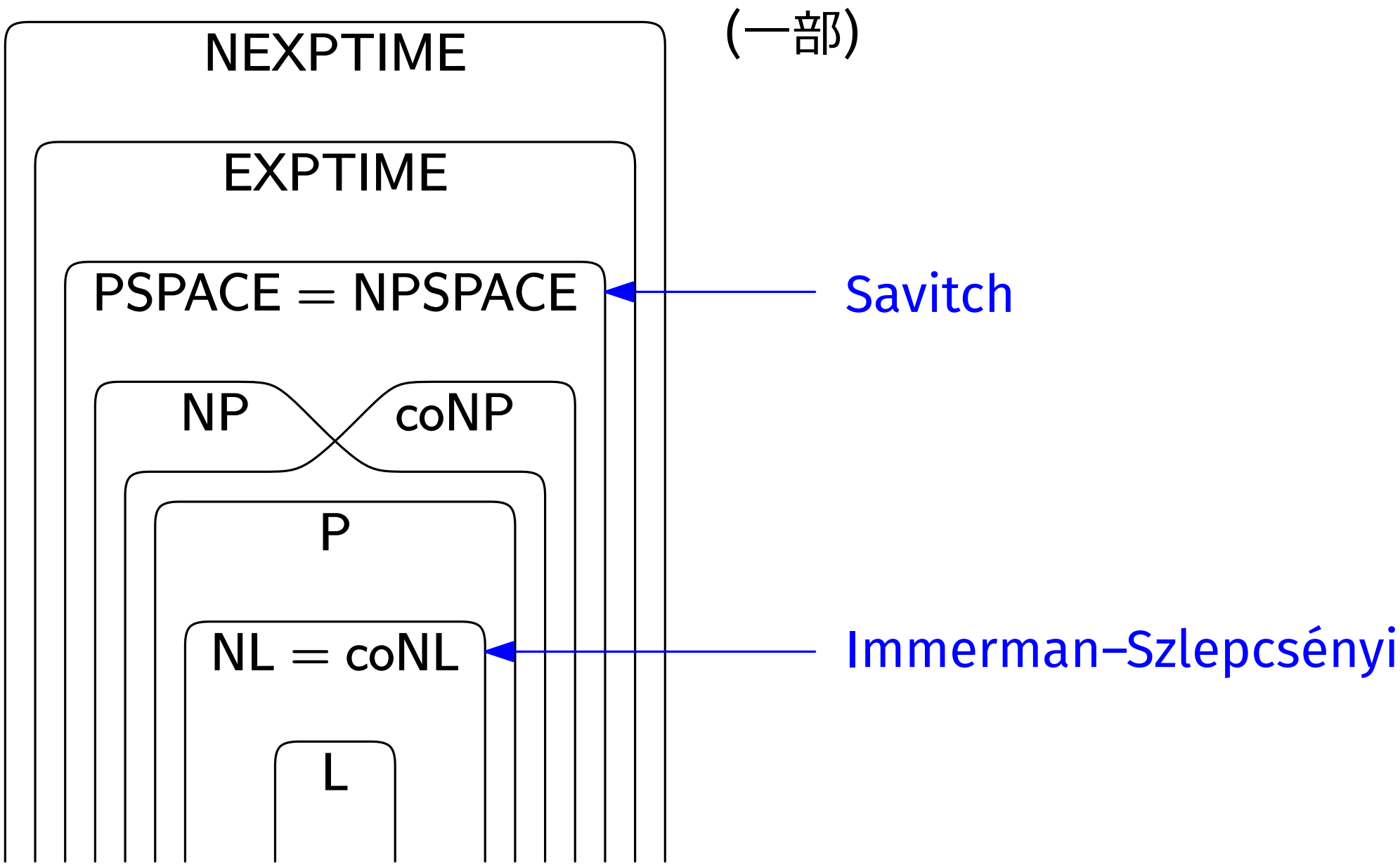
注：この 2 つだけが課題であるわけではない

計算複雑性クラス



(一部)

計算複雑性クラス


(一部)
NEXPTIME
EXPTIME
PSPACE = NPSPACE
NP
coNP
P
NL = coNL
L
Savitch
Immerman–Szlepcsényi

スケジュール (前半)

1. 計算理論の復習 (4/7)
2. 時間計算量：P, NP, coNP (4/14)
3. 帰着と完全性：NP 完全 (4/21)
4. 領域計算量：L, NL, PSPACE (4/28)

* 休み (祝日) (5/5)

5. 時間と領域の関係：$\mathrm{P} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXPTIME}$ (5/12)
6. 階層定理：$\mathrm{P} \neq \mathrm{EXPTIME}$ (5/19)
7. Ladner の定理：$\mathrm{NP} - \mathrm{P} = \mathrm{NPC} \Rightarrow \mathrm{P} = \mathrm{NP}$ (5/26)

スケジュール (後半)

8. Savitch の定理：PSPACE = NPSPACE (6/2)
9. Immerman-Szlepcsényi の定理：NL = coNL (6/9)
10. 多項式階層：P = NP $\Rightarrow$ P = PH (6/16)
11. 交代性計算：AP = PSPACE (6/23)
12. 確率的計算：P $\subseteq$ BPP $\subseteq$ PP, NP $\subseteq$ PP (6/30)
13. 対話証明系 (1)：NP $\subseteq$ MA $\subseteq$ AM (7/7)
14. 対話証明系 (2)：IP $\subseteq$ PSPACE (7/14)
15. 対話証明系 (3)：PSPACE $\subseteq$ IP (7/21)

* 休み (授業のない日) (7/28)

授業の進め方：基本情報

教員

岡本吉央 (おかもと よしお)
- 居室：西 4-206
- E-mail：okamotoy@uec.ac.jp

講義 Web ページ

http://dopal.cs.uec.ac.jp/okamotoy/lect/2026/tcs/
- 講義スライド
- コメント回答
- レポート出題
- その他

Google Classroom

内部シラバスのコードを見て，各自が登録

授業の進め方：各回の進行 (予定)

09:00	授業開始
09:40 頃	休憩 (3 分程度)
	授業再開
	授業終了
10:30	クイズ・コメント投稿
13:00	クイズ・コメント受付終了

授業の進め方：各回の進行 (予定)

09:00　授業開始

09:40 頃　休憩 (3 分程度)

授業再開

授業終了

10:30　クイズ・コメント投稿

13:00　クイズ・コメント受付終了

Google Classroom のフォームから投稿

- 質問・疑問
- 誤植の指摘
- 感想
- 要望
- 文句
- 雑談
- など何でも

授業の進め方：評価

評価方法

毎回のクイズ
- 1 回 1 点満点

2 回のレポート提出
- 1 回 45 点満点

$\rightsquigarrow$ 素点合計の上限 $= 1 \times 15 + 45 \times 2 = 105$

成績

成績 $= \min\{$素点合計$, 100\}$

授業の考え方：数学と証明

注意点 1

この授業では，数学を使い，数学的な証明もする

理由

- 授業の内容の正しさが **検証** できるべき
- 証明という **技法** を身につけるべき

$\rightsquigarrow$ この授業では **批判的思考**，**論理的思考** の体得 も目指す

授業の考え方：プログラミング

注意点 2

この授業では，プログラミングを行わない

理由

- 教員がまじめにプログラミングをしたことがない
  (∵ 教員が授業で扱えない)
- 受講生自身が自主的にプログラミングをすればよい

⇝ 授業の内容から **自由に はみ出ていく** ことを推奨する

前提知識

次の内容は既知とする

- **基礎的な離散数学**
  - 特に，集合・写像の記法 と 数学的帰納法
- **基礎的なプログラミング，アルゴリズム**
  - 特に，再帰 と O 記法
- **基礎的な確率論**
  - 特に，離散確率

身についていない場合は，自習をするように

参考書

主に，次の文献を参考にして進める (入手の必要はない)

- M. Sipser, *Introduction to the Theory of Computation, 3rd Edition*. Cengate Learning, 2012.
- S. Arora, B. Barak, *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
- D. Kozen, *Theory of Computation*. Springer, 2006.
- 荻原光徳，複雑さの階層，共立出版，2006.
- 守屋悦朗，チューリングマシンと計算量の理論，培風館，1997.

注意 1

注意 1

この授業は　**P vs NP 問題**　を扱うものではない

ただし，P，NP の定義や性質は扱う

注意 1

この授業は　**P vs NP 問題**　を扱うものではない

ただし，P，NP の定義や性質は扱う

注：クレイ数学研究所ミレニアム問題 (’00)

- ポアンカレ予想 (解決)
- ヤン・ミルズ方程式と質量ギャップ
- リーマン予想
- P vs NP
- ナビエ・ストークス方程式
- ホッジ予想
- BSD 予想

注意 2

注意 2

この授業は **個々の問題の計算複雑性** を扱うものではない

例えば，次のような問題意識は扱わない

- テトリスは NP 完全 (Bleukelaar et al. '04)
- オセロは PSPACE 完全 (Iwata, Kasai '94)
- チェッカーは EXPTIME 完全 (Robson '84)

ただし，次は扱う

- NP 完全，PSPACE 完全，EXPTIME 完全という用語の定義や性質

本日の内容

1. **計算理論の復習：問題と計算モデル**
2. 計算理論の復習：符号化
3. 計算理論の復習：万能性

問題？

次のようなものを 問題 だと思うかもしれない

方程式 $x^2 - 3x + 2 = 0$ を解け

問題？

次のようなものを 問題 だと思うかもしれない

方程式 $x^2 - 3x + 2 = 0$ を解け

課題

- この方程式「だけ」が解けても，うれしくない
- 「解く」とは何なのか，明確ではない

問題？

次のようなものを 問題 だと思うかもしれない

方程式 $x^2 - 3x + 2 = 0$ を解け

課題

- この方程式「だけ」が解けても，うれしくない
- 「解く」とは何なのか，明確ではない

こんな感じのものを 問題 と言いたい

入力： 実数 a, b, c
出力： $ax^2 + bx + c = 0$ を満たす複素数 x をすべて

判定問題とインスタンス

この講義で扱うのは たいてい **判定問題** である

定義 (非形式) : 判定問題 (decision problem)

判定問題 とは，次の形で書けるもののこと

入力： I
出力： I が性質 $\mathcal{P}$ を満たす $\Rightarrow$ Yes
I が性質 $\mathcal{P}$ を満たさない $\Rightarrow$ No

「性質 $\mathcal{P}$ を判定する問題」という言い方をすることがある

判定問題とインスタンス

この講義で扱うのは たいてい **判定問題** である

定義 (非形式) : 判定問題 (decision problem)

判定問題 とは，次の形で書けるもののこと

入力： I
出力： I が性質 $\mathcal{P}$ を満たす $\Rightarrow$ Yes
I が性質 $\mathcal{P}$ を満たさない $\Rightarrow$ No

「性質 $\mathcal{P}$ を判定する問題」という言い方をすることがある

定義 (非形式) : インスタンス (instance)

判定問題 P の **インスタンス** とは，P の入力 1 つのこと

- **Yes インスタンス** とは，出力が Yes であるインスタンス
- **No インスタンス** とは，出力が No であるインスタンス

定義 (非形式) : アルゴリズム (algorithm)

判定問題 P を **解く アルゴリズム** とは，
任意の入力 I に対して，次を行うもの

- I が性質 $\mathcal{P}$ を満たす ⇒ Yes を出力
- I が性質 $\mathcal{P}$ を満たさない ⇒ No を出力

定義 (非形式) : 判定問題 (decision problem)

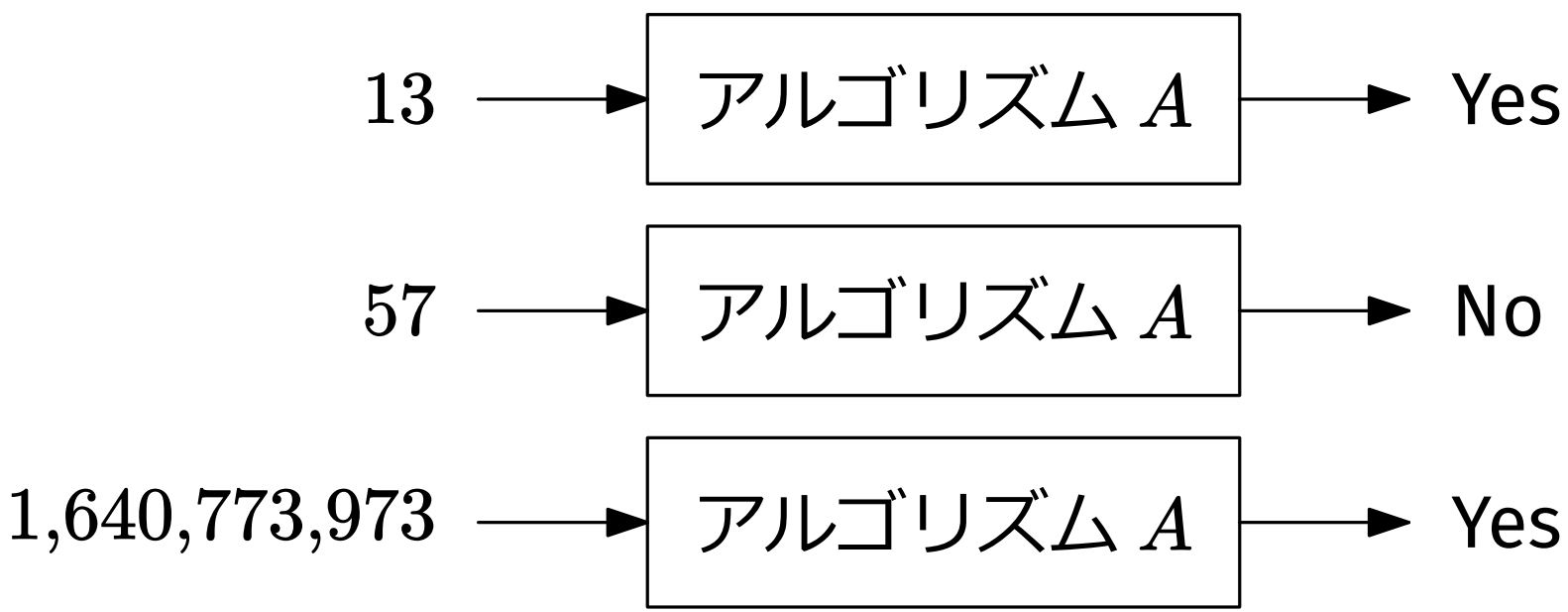
判定問題 とは，次の形で書けるもののこと

入力 : I
出力 : I が性質 $\mathcal{P}$ を満たす ⇒ Yes
I が性質 $\mathcal{P}$ を満たさない ⇒ No

判定問題の例

問題：素数判定

入力： 正整数 n
出力： n が素数である　⇒ Yes
n が素数ではない ⇒ No



これを $A(13) = \text{Yes}$, $A(57) = \text{No}$ などと書く

計算モデル

定義 (非形式)：計算モデル (model of computation)

計算モデル とは, アルゴリズムができることを定めたもの

歴史の中で，さまざまな計算モデルが提案されている

- 帰納的関数
- ラムダ計算
- チューリング機械
- ランダム・アクセス機械
- さまざまなプログラミング言語
- 有限オートマトン
- タグ・システム
- ライフ・ゲーム
- ...

計算モデル

定義 (非形式)：計算モデル (model of computation)

計算モデル とは, アルゴリズムができることを定めたもの

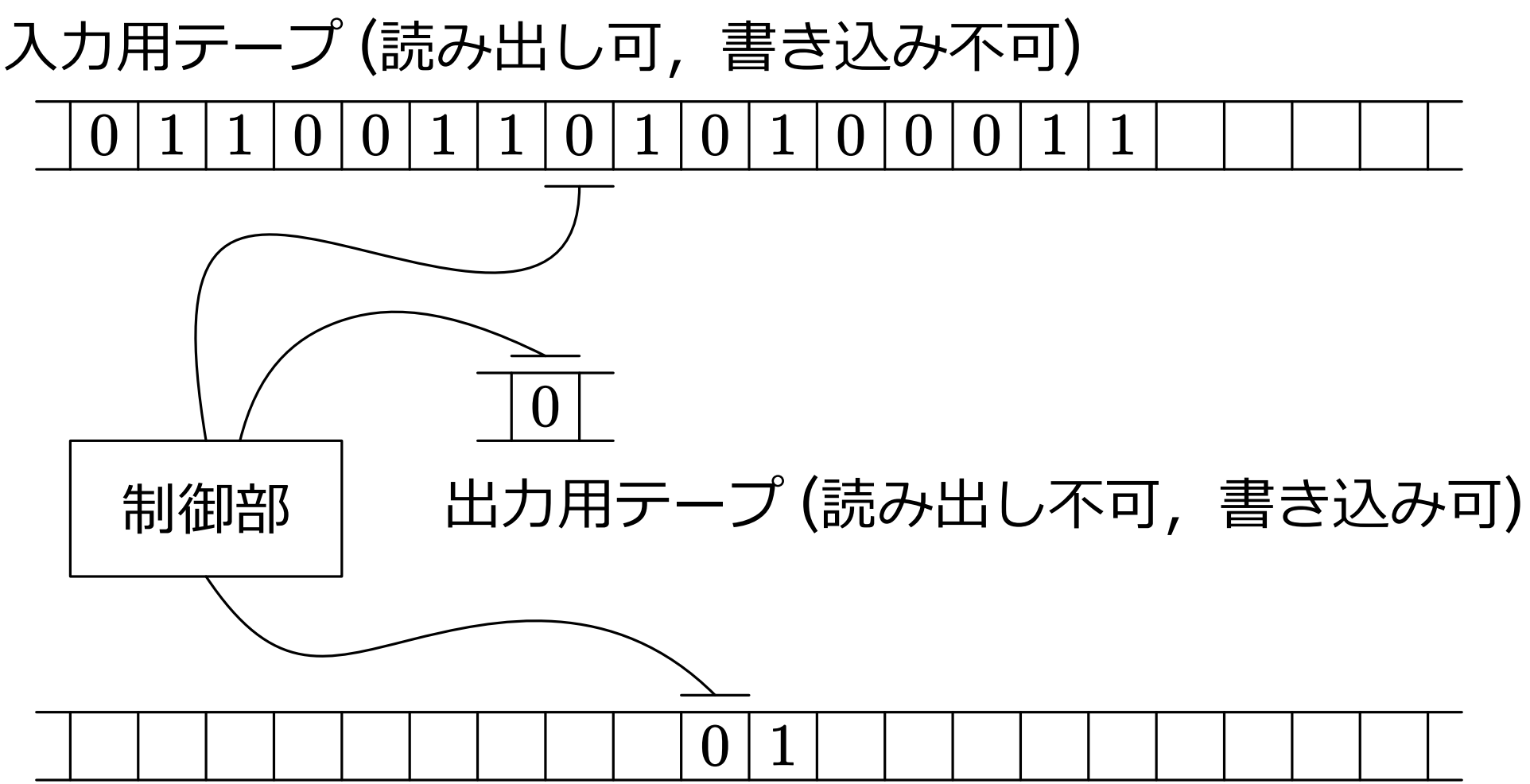
歴史の中で，さまざまな計算モデルが提案されている

- 帰納的関数
- ラムダ計算
- チューリング機械
- ランダム・アクセス機械
- さまざまなプログラミング言語
- 有限オートマトン
- タグ・システム
- ライフ・ゲーム
- ...

重要な注意

この講義では **チューリング機械** を計算モデルとして使う

チューリング機械：イメージ



チューリング機械：要点

要点

「ふつう」のプログラミング言語ができることは
チューリング機械でできると思ってよい

- 整数の四則演算，条件分岐と反復
- ポインタとアドレッシング (直接/間接)
- 再帰

プログラミング言語における単位操作と
チューリング機械における単位操作は
たがいに，多項式回の操作で模倣できる

ショートカット

チューリング機械におけるアルゴリズムを
プログラミング言語におけるアルゴリズムのように書く

本日の内容

1. 計算理論の復習：問題と計算モデル
2. **計算理論の復習：符号化**
3. 計算理論の復習：万能性

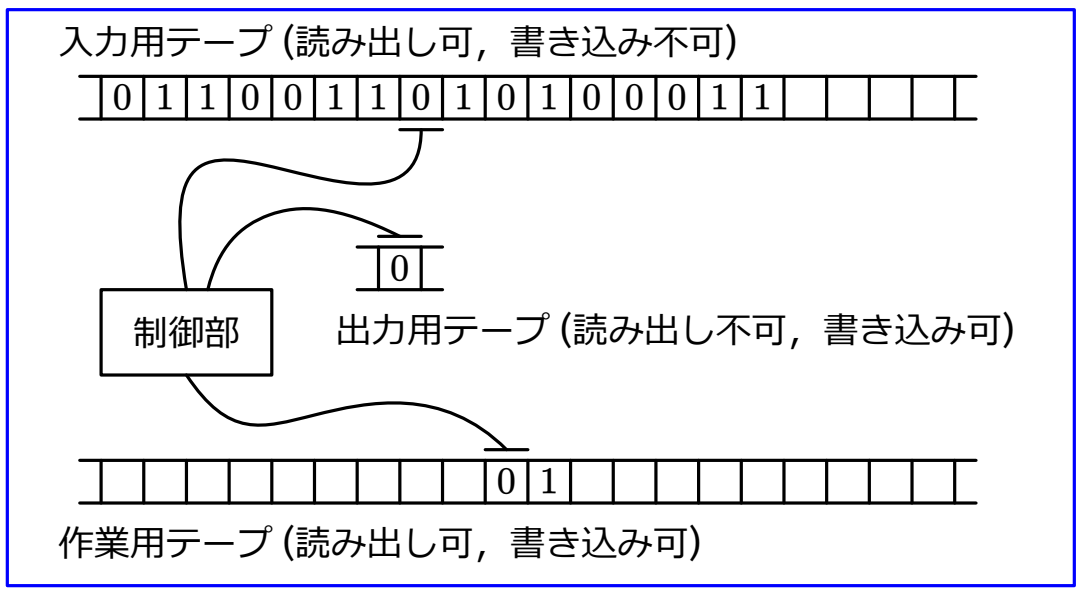
チューリング機械で計算するときの課題

願望

- いろいろなものを計算の対象としたい
  (整数，有理数，文字列，集合，グラフ，論理式，…)

課題

- それらを 0 と 1 の列で表さないといけない



符号化

符号化 (encoding)：考え方

「紙に書けるもの」は，0 と 1 の列で表現できる

符号化 = 0 と 1 の列による表し方 (この授業では)

例：ASCII (アスキー) コード

文字	符号	文字	符号	文字	符号
0	00110000	1	00110001	2	00110010
3	00110011	4	00110100	5	00110101
A	01000001	B	01000010	C	01000011
a	01100001	b	01100010	c	01100011
,	00101100	{	01111011	}	01111100

「紙に書けるもの」の例

符号化 (encoding)：考え方

「紙に書けるもの」は，0 と 1 の列で表現できる

文字列

The quick brown fox ...

数式

$23 \times (45 + 67)$

論理式

$P \wedge (Q \vee \neg R)$

集合

$\{2, 3, a, t, \{4, b, w\}\}$

グラフ

$(\{a, b, c\}, \{\{a, b\}, \{b, c\}\})$

プログラム

$\texttt{print}(\texttt{'Hello, World!'})$

符号長

定義 (非形式)：符号長 (encoding length)

ものの o の **符号長** とは,
それを符号で表したときの 0 と 1 の総数 (単位：ビット)

記法：$|o|$ = ものの o の符号長

例：ASCII コードで考えるとき

- $|(\{a,b,c\},\{\{a,b\},\{b,c\}\})| = 23 \cdot 8 = 184$ ビット

符号長：整数

性質：整数の符号長

整数 n の符号長は $O(\log n)$ ビット

略証：n の桁数は $O(\log n)$ ␣

例：ASCII コードで考えるとき

- $|1| = 8$ ビット
- $|12| = 16$ ビット
- $|123| = 24$ ビット
- $|1234| = 32$ ビット
- $|-1| = 16$ ビット
- $|-12| = 24$ ビット
- $|-123| = 32$ ビット
- $|-1234| = 40$ ビット

符号長：グラフ . . . の前に

定義：無向グラフ (undirected graph)

無向グラフ とは，次を満たす V と E の組 $G = (V, E)$

- V は有限集合 (**頂点** の集合)
- E は V の要素数 2 の部分集合の集合 (**辺** の集合)

例：$(\{a, b, c\}, \{\{a, b\}, \{b, c\}\})$

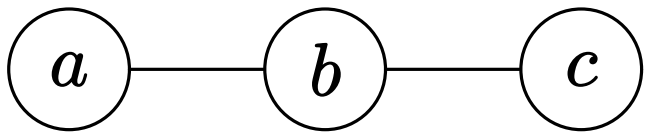
符号長：グラフ . . . の前に

定義：無向グラフ (undirected graph)

無向グラフ とは，次を満たす V と E の組 $G = (V, E)$

- V は有限集合 (**頂点** の集合)
- E は V の要素数 2 の部分集合の集合 (**辺** の集合)

例：$(\{a, b, c\}, \{\{a, b\}, \{b, c\}\})$



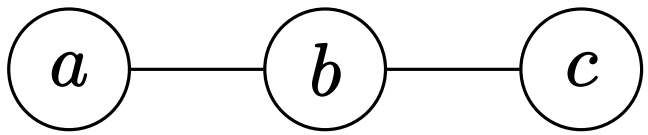
符号長：グラフ . . . の前に

定義：無向グラフ (undirected graph)

無向グラフ とは，次を満たす V と E の組 $G = (V, E)$

- V は有限集合 (**頂点** の集合)
- E は V の要素数 2 の部分集合の集合 (**辺** の集合)

例：$(\{a, b, c\}, \{\{a, b\}, \{b, c\}\})$



定義：有向グラフ (directed graph)

有向グラフ とは，次を満たす V と A の組 $G = (V, A)$

- V は有限集合 (**頂点** の集合)
- A は V の順序対の集合 (**弧** の集合)

例：$(\{a, b, c\}, \{(a, b), (b, c)\})$

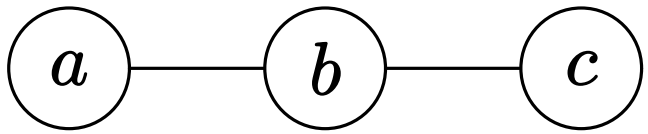
符号長：グラフ . . . の前に

定義：無向グラフ (undirected graph)

無向グラフ とは，次を満たす V と E の組 $G = (V, E)$

- V は有限集合 (**頂点** の集合)
- E は V の要素数 2 の部分集合の集合 (**辺** の集合)

例：$(\{a, b, c\}, \{\{a, b\}, \{b, c\}\})$

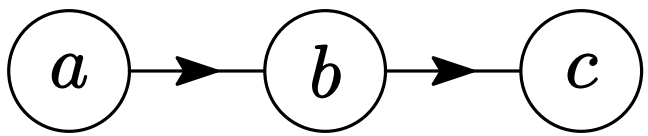


定義：有向グラフ (directed graph)

有向グラフ とは，次を満たす V と A の組 $G = (V, A)$

- V は有限集合 (**頂点** の集合)
- A は V の順序対の集合 (**弧** の集合)

例：$(\{a, b, c\}, \{(a, b), (b, c)\})$



符号長：グラフ

性質：グラフの符号長

- 頂点数 n，辺数 m の無向グラフの符号長は $O((n+m)\log n)$ ビット
- 頂点数 n，弧数 m の有向グラフの符号長は $O((n+m)\log n)$ ビット

略証：

- 各頂点の符号化 $\to O(\log n)$
- 各辺・弧の符号化 $\to$ 2 頂点の符号化 $\to O(\log n)$
- 頂点数 $= n$，辺・弧数 $= m \to$ 全体で $O((n+m)\log n)$ □

例：ASCII コードで考えるとき

- $|(\{a,b,c\},\{\{a,b\},\{b,c\}\})| = 23 \cdot 8 = 204$ ビット

本日の内容

1. 計算理論の復習：問題と計算モデル
2. 計算理論の復習：符号化
3. **計算理論の復習：万能性**

万能性

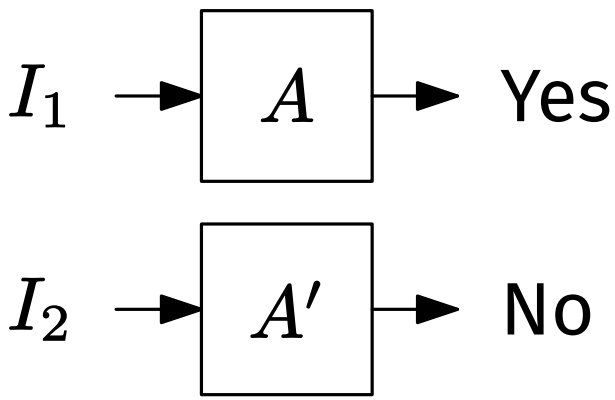
チューリング機械やふつうのプログラミング言語が持つ性質

性質：万能性 (universality)

次の問題を解くアルゴリズム U が存在する

入力： アルゴリズム A，入力 I
出力： I を A に入力したときの A の出力

注意：万能性 $\neq$ 何でもできる



U は その計算モデルの **インタプリタ**

万能性

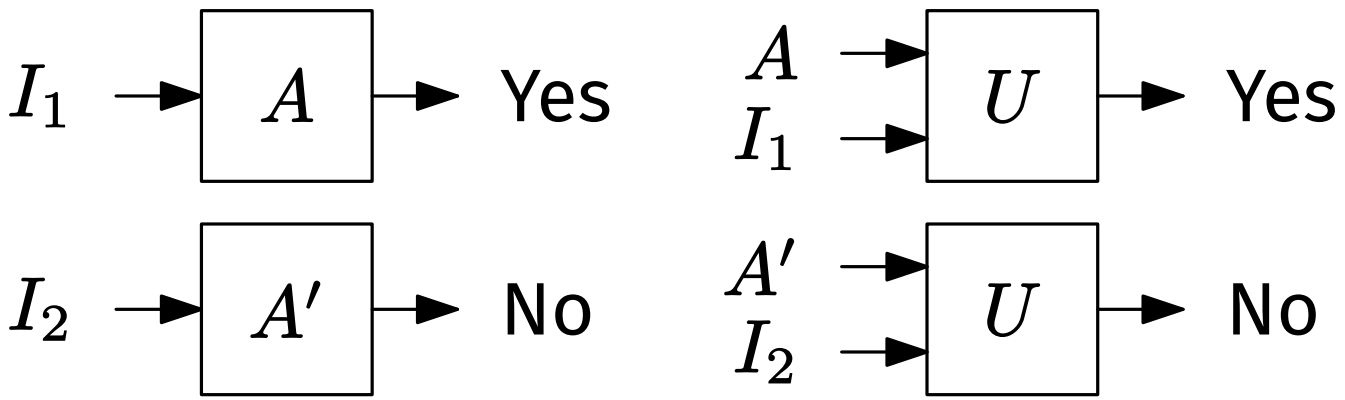
チューリング機械やふつうのプログラミング言語が持つ性質

性質：万能性 (universality)

次の問題を解くアルゴリズム U が存在する

入力： アルゴリズム A，入力 I
出力： I を A に入力したときの A の出力

注意：万能性 $\neq$ 何でもできる



U は その計算モデルの **インタプリタ**

万能性：実行時間

性質：万能性 (universality)

次の問題を解くアルゴリズム U が存在する

入力： アルゴリズム A，入力 I
出力： I を A に入力したときの A の出力

注：$U(A, I)$ の実行時間 $\leq A(I)$ の実行時間の多項式となるように U を構成できる

例えば，$A(I)$の実行時間 $= T$ のとき，
$U(A, I)$の実行時間 $\leq T^2$ のように

次回の予告

次回

- 計算資源として **時間** を扱う
- 計算モデルに **非決定性** を加える

考える計算複雑性クラス

- P，EXPTIME
- NP，NEXPTIME
- coNP，coNEXPTIME

クイズ

クイズはここに手書きで伝える

Q

1.

2.

3.

4.