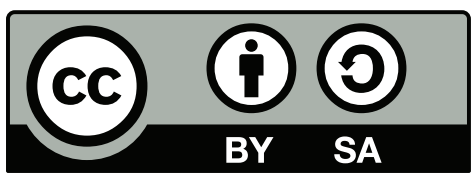
オペレーションズ・リサーチ III (2)

田中 俊二

shunji.tanaka@okayama-u.ac.jp



スケジュール

No.	内容
1	導入 (組合せ最適化，グラフ・ネットワーク，整数計画問題)
2	計算複雑さの理論
3	グラフ・ネットワーク 1 (グラフの分類，用語，種々の問題)
4	グラフ・ネットワーク 2 (最短経路問題，動的計画法)
5	グラフ・ネットワーク 3 (最小全域木，最大フロー問題)
6	グラフ・ネットワーク 4 (マッチング)
7	整数計画 (緩和問題，分枝限定法，切除平面法)

計算量とインスタンス

計算量

- 組合せ最適化問題には簡単な問題も難しい問題もある
  - 割当問題・最短経路問題：簡単
  - ナップサック問題：ちょっと難しい
  - 巡回セールスマン問題：とっても難しい
- 問題を解く方法 (アルゴリズム) の効率

  計算時間： **時間計算量** (time complexity)

  記憶容量： **空間計算量** (space complexity)

問題 (problem) とインスタンス (instance) の違い

- 問題：問題のパラメータは抽象化 (記号で表されている)
  - n 人のバイトを n 個のシフトに総コストが最小になるよう割り当てる
  - バイト i をシフト j に割り当てるコストは c_{ij}
- インスタンス (問題例)：パラメータの具体的な値
  - $n = 3$，バイト X, Y, Z を午前，午後，深夜のシフトに割り当てる
  - コスト c_{ij} の具体的な値も与えられている

計算量とインスタンス

計算量

- 組合せ最適化問題には簡単な問題も難しい問題もある
  - 割当問題・最短経路問題：簡単
  - ナップサック問題：ちょっと難しい
  - 巡回セールスマン問題：とっても難しい
- 問題を解く方法 (アルゴリズム) の効率

  計算時間： **時間計算量** (time complexity) ⇒ **単に「計算量」といえばこれ**

  記憶容量： 空間計算量 (space complexity)

問題 (problem) とインスタンス (instance) の違い

- 問題：問題のパラメータは抽象化 (記号で表されている)
  - n 人のバイトを n 個のシフトに総コストが最小になるよう割り当てる
  - バイト i をシフト j に割り当てるコストは c_{ij}
- インスタンス (問題例)：パラメータの具体的な値
  - $n = 3$，バイト X, Y, Z を午前，午後，深夜のシフトに割り当てる
  - コスト c_{ij} の具体的な値も与えられている

計算量の表し方

問題を解くアルゴリズム (algorithm)

- 解を求めるための有限個の手続き
- インスタンスを入力すると，解が出力される

計算量の表し方

- アルゴリズムの総ステップ数
- 基本的な操作は 1 ステップで実行可能
  - 四則演算
  - 値の書き込み・読み出し
  - etc.

インスタンスのサイズ・規模

- 計算量にとくに影響を与えるのはインスタンスのサイズ
  - 割当問題：集合の要素数 n (バイトやシフトの数)
  - ナップサック問題：アイテム数 n
  - グラフの問題：頂点数 $|V|$，辺の数 $|E|$ ($|X|$ は集合 X の要素数)
- 計算量は**インスタンスのサイズの関数**として表す

最悪計算量と平均計算量

計算量はインスタンスのパラメータ値にも依存

割当問題の例

割り当てコスト c_{ij} がすべて等しいなら，
バイト i をシフト i に割り当てれば最適

	午前	午後	深夜
バイト A	1	1	1
バイト B	1	1	1
バイト C	1	1	1

最悪計算量と平均計算量

同じサイズのインスタンスすべてについて，以下を求めたもの

最悪計算量 (worst-case complexity)： 計算量の最大値．通常はこちらを使う

平均計算量 (average complexity)： 計算量の平均値

整列 (ソート) アルゴリズムの例 (数値の個数 n)

クイックソート： 最悪計算量 $O(n^2)$，平均計算量 $O(n \log_2 n)$

ヒープソート： 最悪計算量 $O(n \log_2 n)$

実用上はクイックソートのほうが速い

$O(n^2)$：**オーダー記法**

最悪計算量と平均計算量

計算量はインスタンスのパラメータ値にも依存

割当問題の例

割り当てコスト c_{ij} がすべて等しいなら，
バイト i をシフト i に割り当てれば最適

	午前	午後	深夜
バイト A	**1**	1	1
バイト B	1	**1**	1
バイト C	1	1	**1**

最悪計算量と平均計算量

同じサイズのインスタンスすべてについて，以下を求めたもの

最悪計算量 (worst-case complexity)： 計算量の最大値．通常はこちらを使う

平均計算量 (average complexity)： 計算量の平均値

整列 (ソート) アルゴリズムの例 (数値の個数 n)

クイックソート： 最悪計算量 $O(n^2)$，平均計算量 $O(n \log_2 n)$

ヒープソート： 最悪計算量 $O(n \log_2 n)$

実用上はクイックソートのほうが速い

$O(n^2)$：**オーダー記法**

計算量のオーダー記法

オーダー記法 (order notation)・Big O 記法 (Big O notation)

- サイズ n に対する計算量の増加速度が重要
- 最高次の項だけ考える
  n が大きいとき，次数の低い項は無視できる
- 定数倍も省略
- **ランダウ (Landau) の記号 O**

例

- 総ステップ数の最大値が $5n^3 + 2n^2 + 4$
  $\Rightarrow$ 計算量 $O(n^3)$，計算量は n^3 のオーダー
- 総ステップ数の最大値が $4n^3 \log_2 n + n^3 + n \log n$
  $\Rightarrow$ 計算量 $O(n^3 \log_2 n)$，計算量は $n^3 \log_2 n$ のオーダー (log の底 2 は省略してもよい)

対数 (logarithmic)，多項式 (polynomial)，指数 (exponential) オーダー

- $O(\log_2 n)$: **対数オーダー**
- $O(n^2)$, $O(n^3)$, etc.: **多項式オーダー**
- $O(2^n)$, $O(3^n)$, etc.: **指数オーダー**

Big O 記法の練習問題

Big O 記法で表す

1. $4n^3 + 2n^2 + 100n + 3000$
2. $3n^2 \log_2 n + 5n^3 + 7(\log_2 n)^2$
3. $5 \cdot 2^{n+3} + 4 \cdot 3^n$

Big O 記法の練習問題

Big O 記法で表す

1. $4n^3 + 2n^2 + 100n + 3000$
2. $3n^2 \log_2 n + 5n^3 + 7(\log_2 n)^2$
3. $5 \cdot 2^{n+3} + 4 \cdot 3^n$

解答

1. $O(n^3)$
2. $O(n^3)$
3. $O(3^n)$

Big Θ 記法 (Big Theta Notation)

Big Θ 記法 (Big Theta Notation)

- 計算量 $O(n^3)$
  - 計算量 $O(n^2)$ や計算量 $O(n^2 \log_2 n)$ なども含む
  - より小さいオーダーで表せる可能性あり
- 計算量 $\Theta(n^3)$
  - n^3 より小さなオーダーで表すことはできない
- 他にも，$O(\cdot)$ の逆の意味を持つ $\Omega(\cdot)$ (Big Omega) も (たまに) 使われる

$f(x) = O(g(x))$ の定義

- ある x_0 および $c > 0$ が存在して，$x > x_0$ のとき $|f(x)| \leq c|g(x)|$ が成り立つ
- 微積でも出てくる (おもに x が微小量のとき)

$f(x) = \Theta(g(x))$ の定義

ある x_0 および $c_1, c_2 > 0$ が存在して，$x > x_0$ のとき $c_1|g(x)| \leq |f(x)| \leq c_2|g(x)|$ が成り立つ

組合せ爆発

組合せ最適化問題

- 解は有限個なので，すべて列挙 (全列挙) すれば最適解が求まる!

  割当問題： $n!$ 通り
  0-1 ナップサック問題： 2^n 通り
  最短経路問題： $O((n-2)!)$ 通り †
  巡回セールスマン問題： $(n-1)!$ 通り ‡

- n が大きくなると，組合せの数は急激に増大
  ⇒ **組合せ爆発** (combinatorial explosion)

† $\sum_{k=0}^{n-2}(n-2)!/k! = O((n-2)!)$, ‡ $n!/n = (n-1)!$. 任意の 2 地点が接続しているとして計算

n と各関数の関係

n	$\log_2 n$	n	$n\log_2 n$	n^2	n^3	2^n	$n!$
10	3.3×10^{0}	1.0×10^{1}	3.3×10^{1}	1.0×10^{2}	1.0×10^{3}	1.0×10^{3}	3.6×10^{6}
20	4.3×10^{0}	2.0×10^{1}	8.6×10^{1}	4.0×10^{2}	8.0×10^{3}	1.0×10^{6}	2.4×10^{18}
50	5.6×10^{0}	5.0×10^{1}	2.8×10^{2}	2.5×10^{3}	1.3×10^{5}	1.1×10^{15}	3.0×10^{64}
100	6.6×10^{0}	1.0×10^{2}	6.6×10^{2}	1.0×10^{4}	1.0×10^{6}	1.3×10^{30}	9.3×10^{157}
1000	1.0×10^{1}	1.0×10^{3}	1.0×10^{4}	1.0×10^{6}	1.0×10^{9}	1.1×10^{301}	–
10000	1.3×10^{1}	1.0×10^{4}	1.3×10^{5}	1.0×10^{8}	1.0×10^{12}	–	–

計算複雑さの理論

組合せ最適化問題

- 全列挙で解くのは難しい
- 問題によっては多項式オーダーの計算量で解ける

  割当問題： $O(n^3)$ のハンガリー法 (Hungarian method)

  単一始点最短経路問題： $O(n^2)$ † のダイクストラ法 (Dijkstra's algorithm)
- 難しい問題と簡単な問題?

† $O(n^2) = O(|V|^2)$．工夫すればもう少し減らせる

計算複雑さの理論の概要

- ある計算機モデル (**チューリング機械**) を使って計算量を評価
- 対象とするのは**決定問題**：解は yes/no のみ
- インスタンスのサイズ：パラメータを**文字列**で表したときの長さ
- 計算量で判定問題をクラス分け

  クラス P： **決定性**チューリング機械で多項式時間

  クラス NP： **非決定性**チューリング機械で多項式時間

きっちり議論するのはとても大変なので，ここでは概略だけ説明

決定問題

決定問題・判定問題 (decision problem)

解が yes/no で決まる問題

決定問題の例その 1：ハミルトン閉路問題

ハミルトン閉路が存在するなら yes，存在しないなら no

ハミルトン閉路：すべての地点をちょうど 1 回ずつ巡って出発地点に戻る巡回路

決定問題の例その 2: 決定問題版の組合せ最適化問題

巡回セールスマン問題において，長さ L のハミルトン閉路が存在するなら yes，存在しないなら no

決定問題の計算量

- チューリング機械を使って解くときの計算量
- チューリング機械への入力：**ビット** (0, 1) と特殊文字の列
  $\Rightarrow$ インスタンスをビット列で表す
- 計算量は**文字列長**の関数

インスタンスの表し方

インスタンスの表し方 (簡単のため，パラメータはすべて非負整数)

- パラメータを 2 進数に直して並べる ⇒ **文字列** (ビット列)
- 文字列長 ⇒ インスタンスのサイズ

非負整数の 2 進数表現

- n 桁の 2 進数で表せる数：$0 \sim 2^n - 1$
- (10 進数で) $0 \sim n'$ $(n' > 0)$ の数を 2 進数で表すのに必要な桁数：$\lfloor \log_2 n' \rfloor + 1$

10 進	2 進	10 進	2 進	10 進	2 進	10 進	2 進
0	0	4	100	8	1000	12	1100
1	1	5	101	9	1001	13	1101
2	10	6	110	10	1010	14	1110
3	11	7	111	11	1011	15	1111

床関数・天井関数

床関数 (floor function) $\lfloor x \rfloor$： x 以下の最大の整数

天井関数 (ceiling function) $\lceil x \rceil$： x 以上の最小の整数

床関数・天井関数の例

$\lfloor 5.3 \rfloor = 5, \lceil 5.3 \rceil = 6, \quad \lfloor 5 \rfloor = 5$

インスタンスを表す文字列の例：巡回セールスマン問題

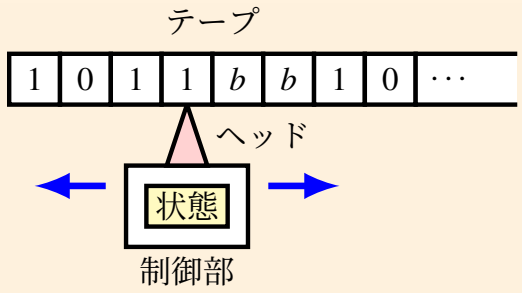
インスタンスを表す文字列の例：巡回セールスマン問題

- インスタンスのパラメータ
  - 地点の数 n
  - 2 地点間の距離 d_{ij}：$n(n-1)/2$ 個 ($d_{ij} = d_{ji}$ の場合)．0 から $d_{\max}$ の整数とする
- d_{ij} を 2 進数で表す桁数：$\lfloor \log_2 d_{\max} \rfloor + 1$
- n を 2 進数で表す桁数：$\lfloor \log_2 n \rfloor + 1$
- 文字列長：$N = (n(n-1)/2)(\lfloor \log_2 d_{\max} \rfloor + 1) + \lfloor \log_2 n \rfloor + 1 = O(n^2 \log_2 d_{\max})$

チューリング機械

チューリング機械 (Turing machine)

- チューリング (Alan Turing) による計算機モデル
- 無限長のテープおよび制御部を備えた機械
- テープ
  - 文字列を記憶できる
  - 使用可能な文字は 0, 1 の他，空白を表す b などの特殊文字
- 制御部
  - 文字の書き込み・読み出し用ヘッド
  - 有限個の内部状態



チューリング
写真

Alan Turing (1912–1954)

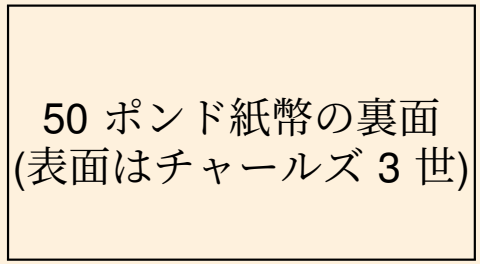
各ステップでの動作

1. テープの現在位置から 1 文字を読み込む
2. 文字と現在の内部状態に応じて次の内部状態へ遷移 (遷移しなくてもよい)
3. テープの現在位置に 1 文字を書き出す (書き出さなくてもよい)
4. テープ上を左か右へ 1 文字分移動する (移動しなくてもよい)

アラン・チューリング

アラン・チューリング (Alan Turing)

- イギリスの数学者
- 第 2 次世界大戦中，イギリス軍に協力しナチス・ドイツ軍のエニグマ暗号の解読に成功
- 人間と AI を判別するための「チューリングテスト」でも有名
- 1952 年，同性愛者であることが発覚し逮捕される
- 化学的去勢措置を受けることを条件に釈放されるが，1954 年に自宅で自殺
- 2013 年に正式に恩赦



エニグマ (Eniguma)

- ナチス・ドイツ軍が用いていた暗号機の名前
- 当時，解読は不可能と言われていた

イミテーション・ゲーム/エニグマと天才数学者の秘密

- チューリングの伝記をもとにした 2014 年のアメリカ映画
- ベネディクト・カンバーバッチがチューリングを演じた

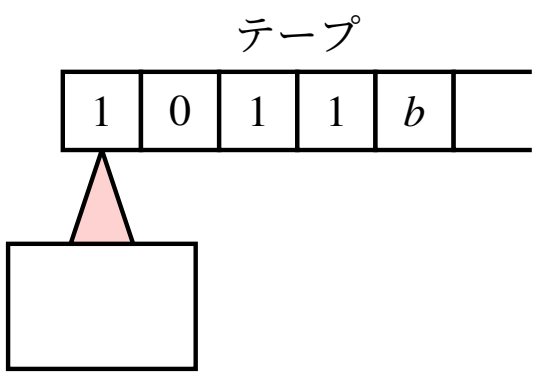
決定問題を解くチューリング機械

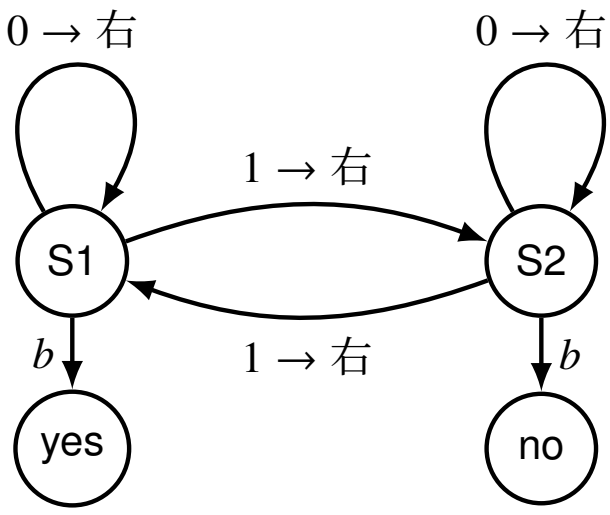
決定問題を解くチューリング機械

- テープには，初期状態ではインスタンス (文字列) が書き込まれている
- 内部状態の遷移，書き込む文字，ヘッドの移動などを適切に設計
- 有限ステップで停止
- 停止時の内部状態は「yes」または「no」(決定問題の解を表す)

例：1 の個数の偶奇判定

ビット列の 1 の個数を空白 (b) まで数え，偶数なら yes，奇数なら no を返す





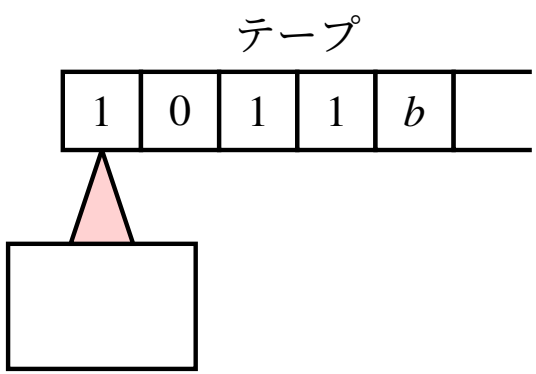
決定問題を解くチューリング機械

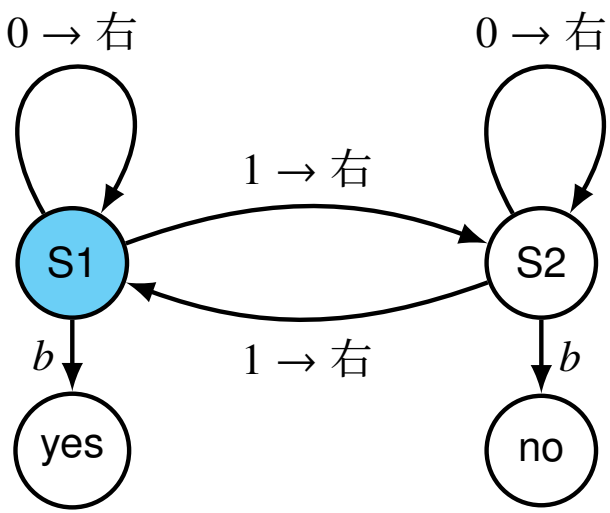
決定問題を解くチューリング機械

- テープには，初期状態ではインスタンス (文字列) が書き込まれている
- 内部状態の遷移，書き込む文字，ヘッドの移動などを適切に設計
- 有限ステップで停止
- 停止時の内部状態は「yes」または「no」(決定問題の解を表す)

例：1 の個数の偶奇判定

ビット列の 1 の個数を空白 (b) まで数え，偶数なら yes，奇数なら no を返す





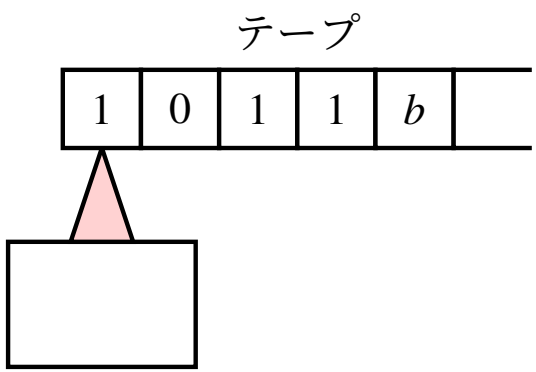
決定問題を解くチューリング機械

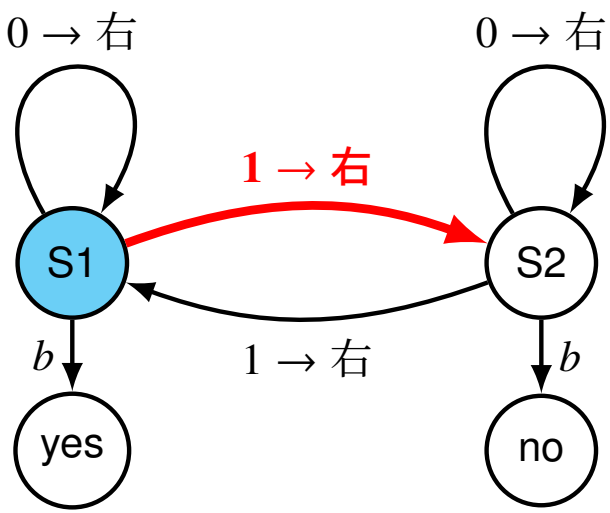
決定問題を解くチューリング機械

- テープには，初期状態ではインスタンス (文字列) が書き込まれている
- 内部状態の遷移，書き込む文字，ヘッドの移動などを適切に設計
- 有限ステップで停止
- 停止時の内部状態は「yes」または「no」(決定問題の解を表す)

例：1 の個数の偶奇判定

ビット列の 1 の個数を空白 (b) まで数え，偶数なら yes，奇数なら no を返す





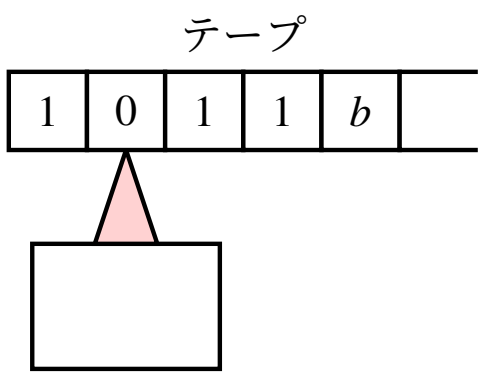
決定問題を解くチューリング機械

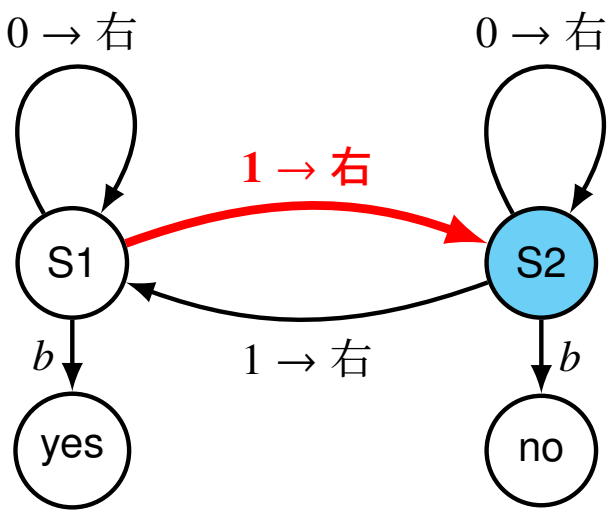
決定問題を解くチューリング機械

- テープには，初期状態ではインスタンス (文字列) が書き込まれている
- 内部状態の遷移，書き込む文字，ヘッドの移動などを適切に設計
- 有限ステップで停止
- 停止時の内部状態は「yes」または「no」(決定問題の解を表す)

例：1 の個数の偶奇判定

ビット列の 1 の個数を空白 (b) まで数え，偶数なら yes，奇数なら no を返す





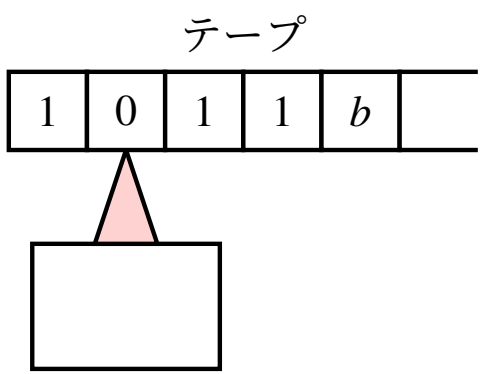
決定問題を解くチューリング機械

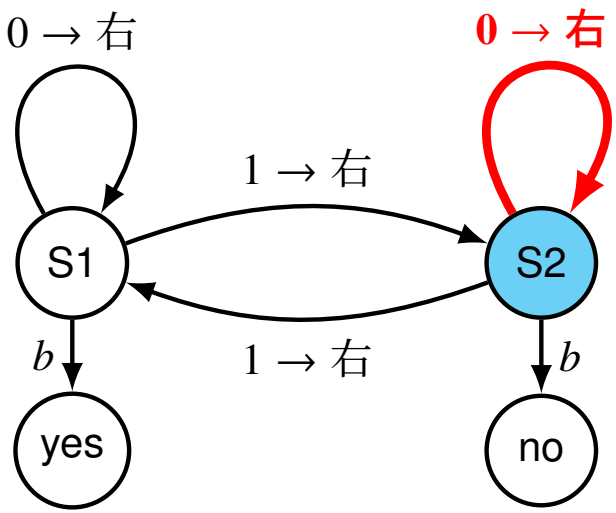
決定問題を解くチューリング機械

- テープには，初期状態ではインスタンス (文字列) が書き込まれている
- 内部状態の遷移，書き込む文字，ヘッドの移動などを適切に設計
- 有限ステップで停止
- 停止時の内部状態は「yes」または「no」(決定問題の解を表す)

例：1 の個数の偶奇判定

ビット列の 1 の個数を空白 (b) まで数え，偶数なら yes，奇数なら no を返す





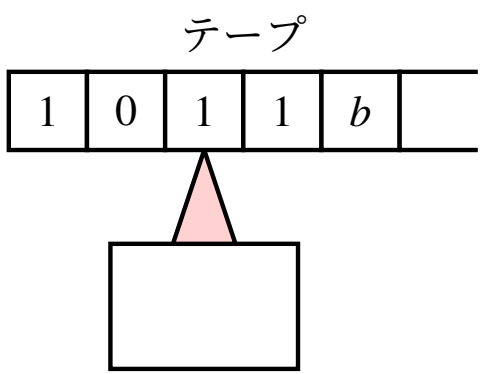
決定問題を解くチューリング機械

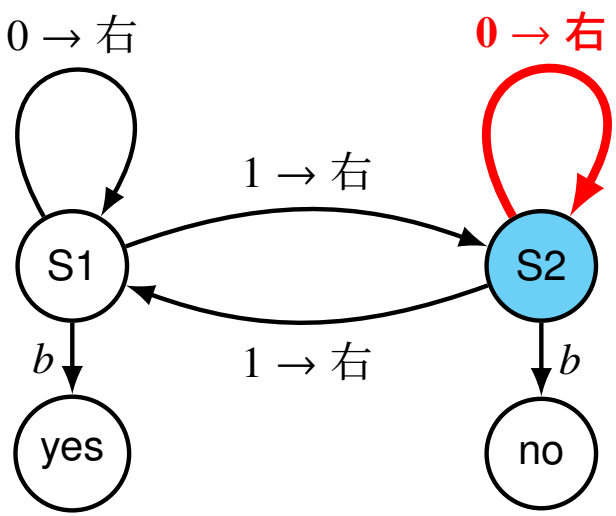
決定問題を解くチューリング機械

- テープには，初期状態ではインスタンス (文字列) が書き込まれている
- 内部状態の遷移，書き込む文字，ヘッドの移動などを適切に設計
- 有限ステップで停止
- 停止時の内部状態は「yes」または「no」(決定問題の解を表す)

例：1 の個数の偶奇判定

ビット列の 1 の個数を空白 (b) まで数え，偶数なら yes，奇数なら no を返す





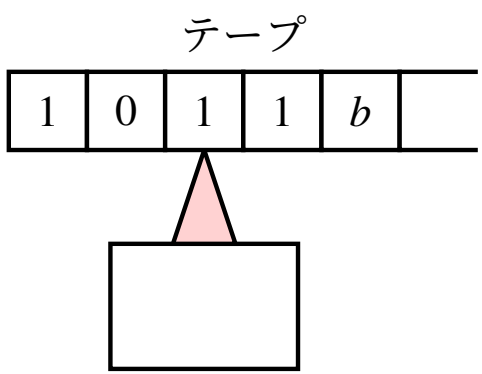
決定問題を解くチューリング機械

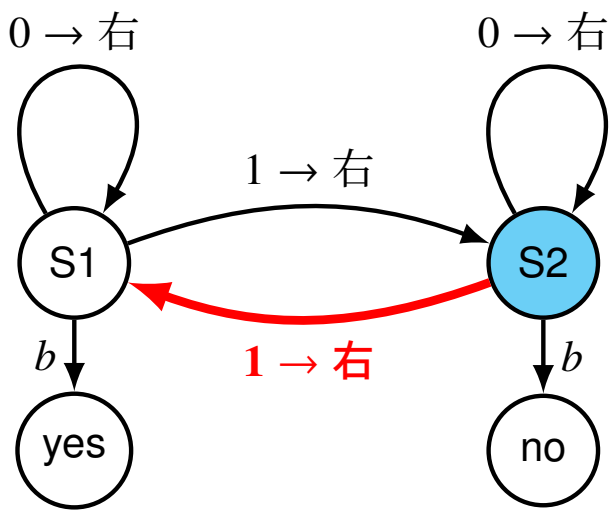
決定問題を解くチューリング機械

- テープには，初期状態ではインスタンス (文字列) が書き込まれている
- 内部状態の遷移，書き込む文字，ヘッドの移動などを適切に設計
- 有限ステップで停止
- 停止時の内部状態は「yes」または「no」(決定問題の解を表す)

例：1 の個数の偶奇判定

ビット列の 1 の個数を空白 (b) まで数え，偶数なら yes，奇数なら no を返す





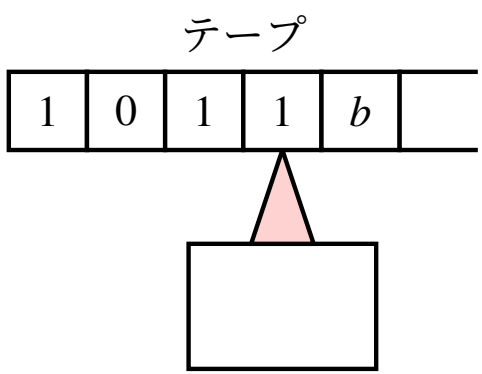
決定問題を解くチューリング機械

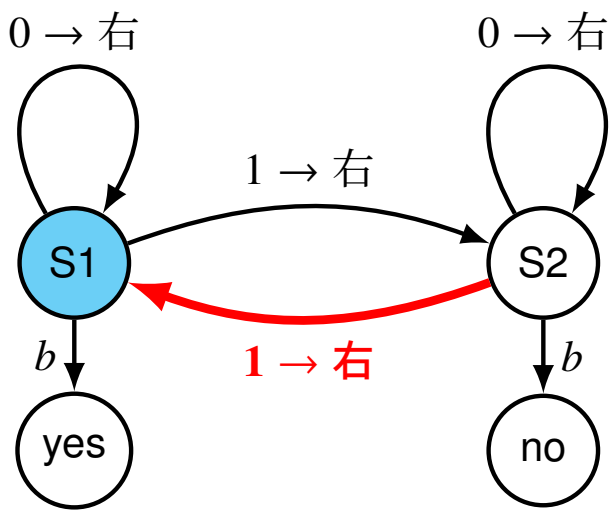
決定問題を解くチューリング機械

- テープには，初期状態ではインスタンス (文字列) が書き込まれている
- 内部状態の遷移，書き込む文字，ヘッドの移動などを適切に設計
- 有限ステップで停止
- 停止時の内部状態は「yes」または「no」(決定問題の解を表す)

例：1 の個数の偶奇判定

ビット列の 1 の個数を空白 (b) まで数え，偶数なら yes，奇数なら no を返す





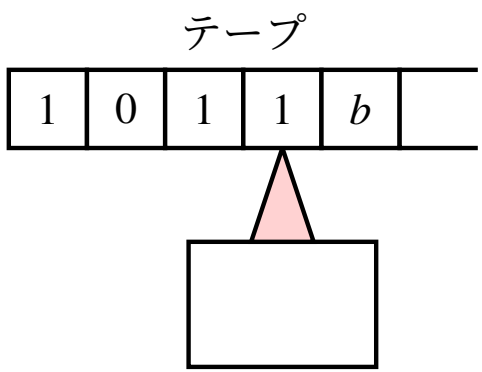
決定問題を解くチューリング機械

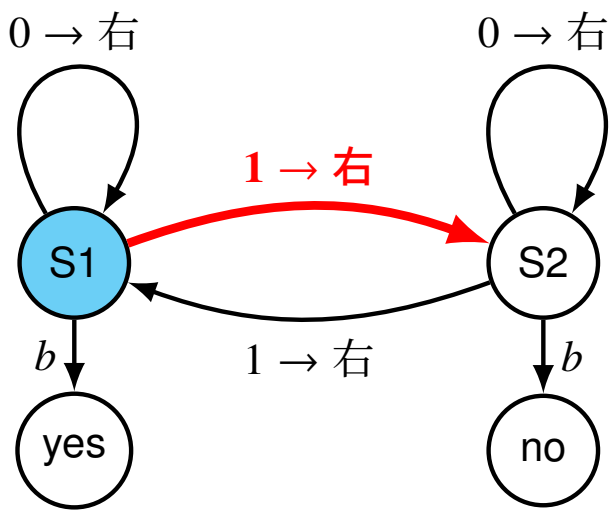
決定問題を解くチューリング機械

- テープには，初期状態ではインスタンス (文字列) が書き込まれている
- 内部状態の遷移，書き込む文字，ヘッドの移動などを適切に設計
- 有限ステップで停止
- 停止時の内部状態は「yes」または「no」(決定問題の解を表す)

例：1 の個数の偶奇判定

ビット列の 1 の個数を空白 (b) まで数え，偶数なら yes，奇数なら no を返す





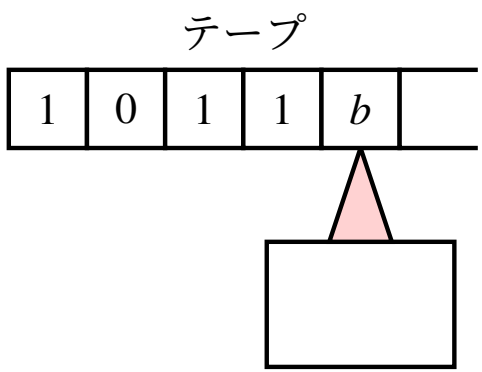
決定問題を解くチューリング機械

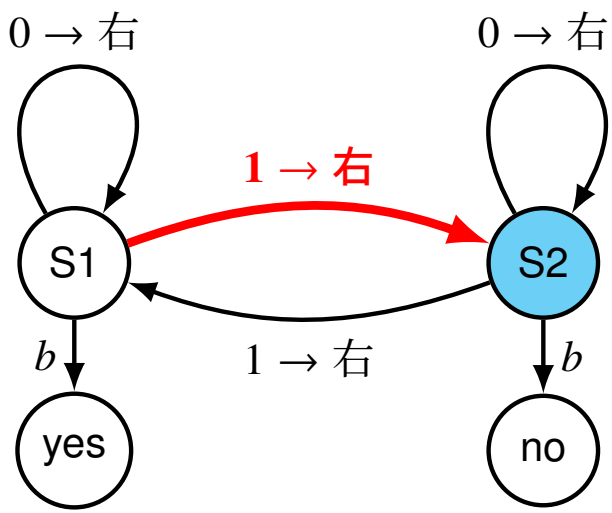
決定問題を解くチューリング機械

- テープには，初期状態ではインスタンス (文字列) が書き込まれている
- 内部状態の遷移，書き込む文字，ヘッドの移動などを適切に設計
- 有限ステップで停止
- 停止時の内部状態は「yes」または「no」(決定問題の解を表す)

例：1 の個数の偶奇判定

ビット列の 1 の個数を空白 (b) まで数え，偶数なら yes，奇数なら no を返す





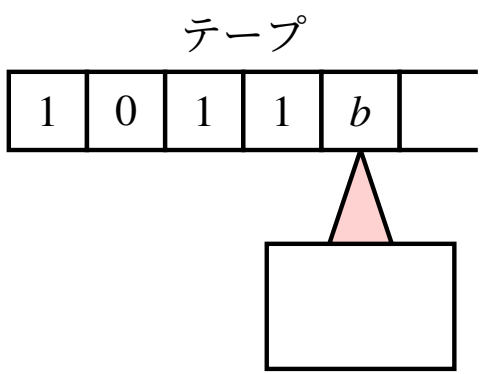
決定問題を解くチューリング機械

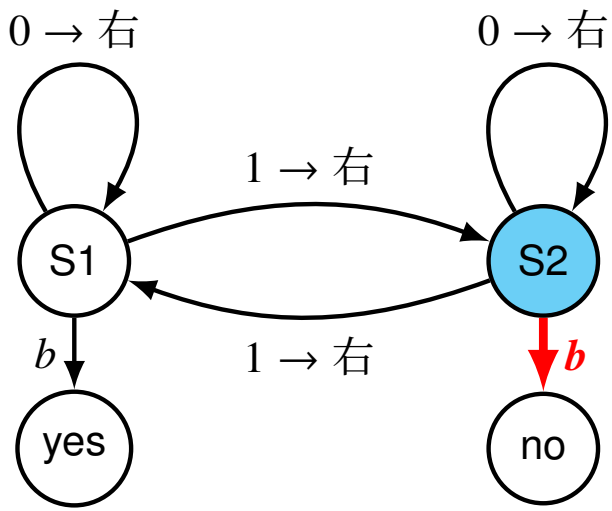
決定問題を解くチューリング機械

- テープには，初期状態ではインスタンス (文字列) が書き込まれている
- 内部状態の遷移，書き込む文字，ヘッドの移動などを適切に設計
- 有限ステップで停止
- 停止時の内部状態は「yes」または「no」(決定問題の解を表す)

例：1 の個数の偶奇判定

ビット列の 1 の個数を空白 (b) まで数え，偶数なら yes，奇数なら no を返す





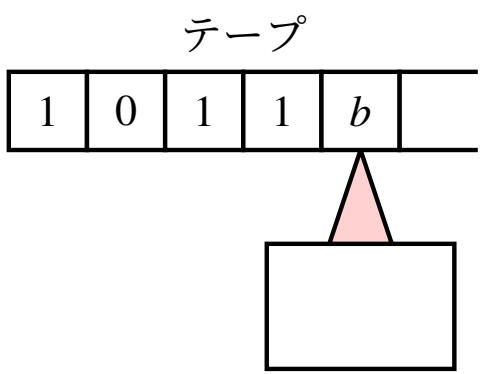
決定問題を解くチューリング機械

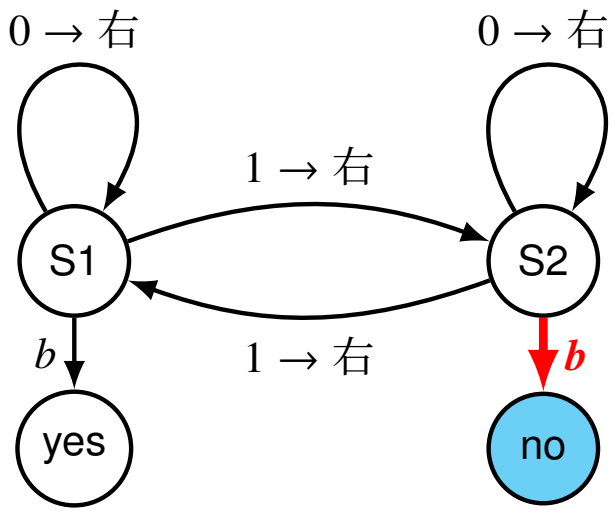
決定問題を解くチューリング機械

- テープには，初期状態ではインスタンス (文字列) が書き込まれている
- 内部状態の遷移，書き込む文字，ヘッドの移動などを適切に設計
- 有限ステップで停止
- 停止時の内部状態は「yes」または「no」(決定問題の解を表す)

例：1 の個数の偶奇判定

ビット列の 1 の個数を空白 (b) まで数え，偶数なら yes，奇数なら no を返す





クラス P

多項式時間で求解可能 (polynomially solvable)

入力文字列長 N の多項式オーダーの計算量 (ステップ数) で正解を返すチューリング機械が存在すること

クラス P (class P)

多項式時間で求解可能な決定問題全体

クラス P に属する問題は，多項式オーダーの計算時間で解ける！

クラス NP

決定問題版の巡回セールスマン

- 「巡回セールスマン問題において，長さが L 以下の巡回路が存在するか」
- 解が yes である「証拠」(certificate) として，具体的な巡回路が与えられた
  $\Rightarrow$ 合っているのか確かめるのは容易 (長さを計算すればよい)

非決定性計算により多項式時間で求解可能 (nondeterministic polynomially solvable)

インスタンスと，その解が yes である証拠 † が与えられたとき，N の多項式オーダーの計算量でその証拠の正しさを (チューリング機械で) 検証できる

† ただし，証拠の文字列長は N の多項式オーダー

クラス NP (class NP)

非決定性計算により多項式時間で求解可能な決定問題全体

クラス NP

決定問題版の巡回セールスマン

- 「巡回セールスマン問題において，長さが L 以下の巡回路が存在するか」
- 解が yes である「証拠」(certificate) として，具体的な巡回路が与えられた
  $\Rightarrow$ 合っているのか確かめるのは容易 (長さを計算すればよい)

非決定性計算により多項式時間で求解可能 (nondeterministic polynomially solvable)

インスタンスと，その解が yes である証拠 † が与えられたとき，N の多項式オーダーの計算量でその証拠の正しさを (チューリング機械で) 検証できる

† ただし，証拠の文字列長は N の多項式オーダー

クラス NP (class NP)

非決定性計算により多項式時間で求解可能な決定問題全体

非決定性計算とは?

非決定性計算の意味

決定性 (deterministic) チューリング機械

- 内部状態の遷移が決定的 (ひとつに決まる)
- 先程の 1 の偶奇を判定するチューリング機械

非決定性 (nondeterministic) チューリング機械

- 内部状態の遷移が確率的
- 同時に複数の遷移を考えることができる

クラス NP (等価な定義)

非決定性チューリング機械により多項式時間で求解可能な決定問題全体

NP は not polynomial でも non-polynomial でもなく，nondeterministic polynomial

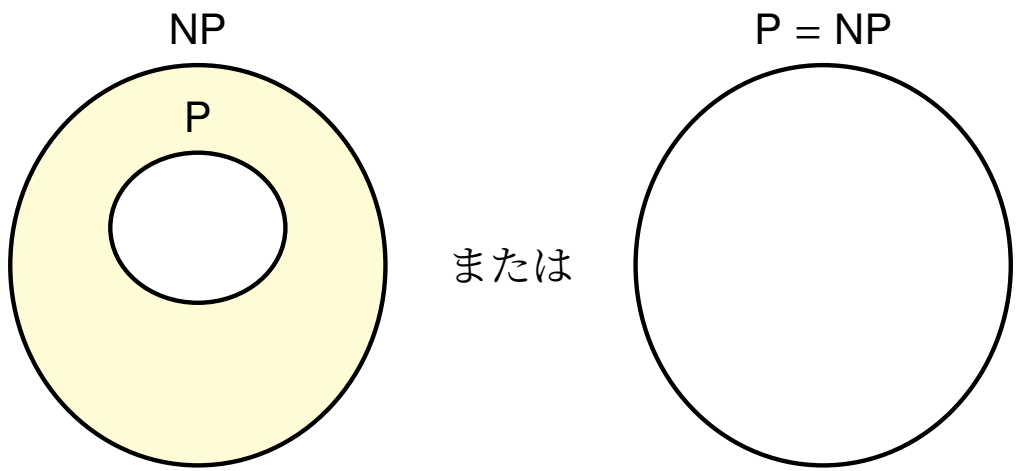
クラス P とクラス NP の関係

クラス P ⊂ クラス NP

長さ 0 の文字列を証拠とすればよい (証拠がなくても多項式オーダー)

クラス P = クラス NP?

- 未解決
- ミレニアム懸賞問題「P versus NP」．100 万ドルの懸賞金
- 大方の予想では (クラス P) ≠ (クラス NP)



問題の難しさの比較：多項式時間帰着

2 つの決定問題 Π_1，Π_2 の比較

- 問題 Π_2 は問題 Π_1 に変換可能：問題 Π_1 が解ければ問題 Π_2 も解ける
  $\Rightarrow$ 問題 Π_1 は問題 Π_2 と同程度に難しいか，より難しい
- ただし，変換に多大な計算量が必要なのは困る

多項式時間帰着 (polynomial-time reduction)

決定問題 Π_1, Π_2 について，以下の 2 条件が成り立つとき，Π_2 は Π_1 に **(多項式時間) 帰着可能** ((polynomial-time) reducible) であるという．また，この変換を，**多項式時間帰着** (polynomial-time reduction) という．

- Π_2 の任意のインスタンス I_2 を，解 (yes/no) が一致する Π_1 のインスタンス I_1 に変換可能
- 変換に必要な計算量は I_2 のサイズの多項式オーダー

多項式時間帰着の例

ハミルトン閉路問題

ハミルトン閉路が存在するなら yes，そうでないなら no

決定問題版の巡回セールスマン問題

総距離 L 以下の巡回路 (ハミルトン閉路) が存在するなら yes，そうでないなら no

(地点間の距離と L を適切に設定した) 決定問題版の巡回セールスマン問題を解けば，ハミルトン閉路問題の答がわかる

決定問題版の巡回セールスマン問題への多項式帰着

- 地点間の接続関係はそのまま
- 接続する 2 地点間の距離はすべて 1
- $L =$ (地点数)　(n 地点を訪れる巡回路の長さが n となることに注意)

NP 完全と NP 困難

多項式時間帰着を使って，クラス NP の中で一番難しい問題を考える

NP 完全 (NP-complete)

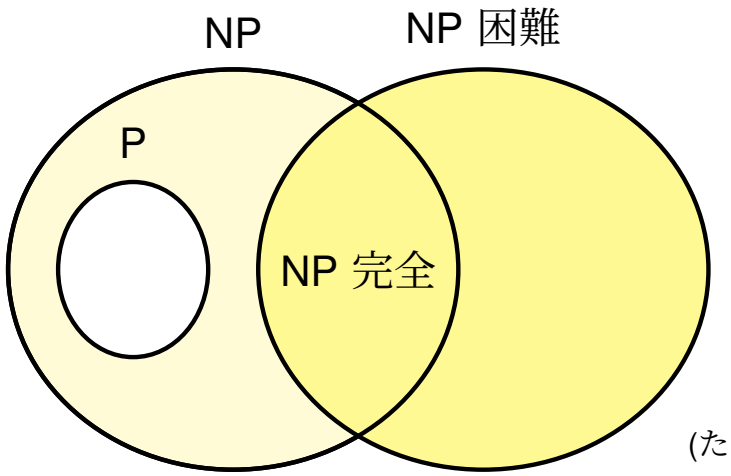
決定問題 Π が **NP 完全**であるとは，

- $\Pi \in$ (クラス NP)
- クラス NP の任意の決定問題を Π に帰着可能

NP 困難 (NP-hard)

決定問題 Π が **NP 困難**であるとは，

- $\Pi \in$ (クラス NP) **とは限らない**
- クラス NP の任意の決定問題を Π に帰着可能



(ただし，P ≠ NP のとき)

クックの定理

NP 完全問題の種類

- クラス NP の中でもっとも難しい問題のクラス
- どんな問題が NP 完全?

クック (Cook) の定理

充足可能性問題 (satisfiability problem) は NP 完全

充足可能性問題：与えられた論理式を満足する論理変数が存在するかを判定する問題

- その後，色々な問題の NP 完全性が示された
- 決定問題版の巡回セールスマン問題も NP 完全
- **巡回セールスマン問題は NP 困難**
  - 巡回セールスマン問題が解ければ，決定問題版も解ける
  - (巡回セールスマン問題) $\notin$ NP　(巡回セールスマン問題は決定問題ではない)
- **ナップサック問題も NP 困難**

NP 完全 (NP 困難) 問題同士での違い

ナップサック問題の方が巡回セールスマン問題よりも解きやすい

擬多項式時間の例：0-1 ナップサック問題

0-1 ナップサック問題

- アイテム数 n
- ナップサック容量 C
- 各アイテム i について，サイズ a_i $(\leq C)$ と利得 c_i
- **計算量 $O(nC)$ のアルゴリズムが存在 $\Rightarrow$ 擬多項式時間**という

利得 c_i の最大値を D とする

インスタンスを表す文字列の長さ N

$$N = \log_2\lfloor n\rfloor + 1 + (n+1)(\log_2\lfloor C\rfloor + 1) + n(\log_2\lfloor D\rfloor + 1) = O(n(\log_2 C + \log_2 D))$$

簡単のため，$N = n(\log_2 C + \log_2 D)$ とする

計算量のオーダー

$$C = 2^{\log_2 C} = 2^{N/n - \log_2 D} \qquad (C \text{ は } N \text{ の指数オーダー})$$

$$O(nC) = O(n2^{N/n - \log_2 D})$$

n と D が定数のとき，計算量は $O(nC) = O(2^{N/n}) = O((\sqrt[n]{2})^N)$ $\Rightarrow$ **指数オーダー**

擬多項式時間の例：0-1 ナップサック問題 (続き)

1 進数表現 (unary encoding)

値を 1 の個数で表現．例：$5 \Rightarrow 11111$

インスタンスを表す文字列の長さ N

$$N = n + (n+1)C + nD = O(n(C+D))$$

簡単のため，$N = n(C+D)$ とする

計算量のオーダー

$$C = N/n - D \qquad (C \text{ は } N \text{ の多項式オーダー})$$

$$O(nC) = O(nC) = O(n(N/n - D)) = O(N - nD)$$

n と D が定数のとき，計算量 $O(nC) = O(N)$ $\Rightarrow$ **多項式オーダー**

2 進数表現よりも入力文字列の長さを長めに見積もったため，計算量が多項式オーダーにおさまった！

擬多項式時間 (pseudopolynomial-time)

$O(nC)$ のように，インスタンスのサイズだけでなく**パラメータ**の多項式を含む

強 NP 完全・強 NP 困難

強 NP 完全 (strongly NP-complete, NP-complete in the strong sense, unary NP-complete)

パラメータ値の最大値が文字列長 N の多項式オーダーのときでも NP 完全

強 NP 完全：1 進数表現で N を長めに見積もってもダメだった

(弱)NP 完全 (weakly NP-complete, NP-complete in the ordinary sense, binary NP-complete)

NP 完全問題のうち，(P ≠ NP のもとで) 擬多項式時間アルゴリズムが存在する

(弱)NP 完全：1 進数表現で N を長めに見積もれば，多項式オーダー

強 NP 困難，(弱)NP 困難

強 NP 完全，(弱)NP 完全に対応

- 0-1 ナップサック問題
  (弱)NP 困難，擬多項式時間アルゴリズムが存在
- 巡回セールスマン問題
  強 NP 困難，(P ≠ NP ならば) 擬多項式時間アルゴリズムは存在しない