PCP定理とその証明

清水 伸高 (東京科学大学)

Contents

序文

このノートは, 計算量理論で 90 年代に証明された重要な結果である PCP 定理とその証明についての講義ノートである. 計算量 (computational complexity) とは, 問題を解くために必要な計算リソースの量 (例えば計算時間, 記憶領域のサイズ, 乱択や量子性の有無や量) を意味し, 計算量理論 (computational complexity theory) とはそれぞれの問題の計算量を明らかにするための理論である. PCP 定理とは, 判定問題 (Yes か No で答える問題) の検証に要する計算量に関する結果であり, 端的に言うと, ある命題が真であると主張する証明が文字列として与えられたとき, その証明を検証するためには, 通常, 全ての文字を見て確認する必要があるが, PCP 定理によれば, その証明の一部だけを見ることで, その命題が真であるか否かを確率的に検証することができるという驚くべき結果である. 例えば, ある実行列 A と実ベクトル b に対して線形方程式系 $Ax = b$ は解を持つ, という命題を考えてみよう. この命題が真であるならば実際に解の一つ x を証明として提示することができるが, その証明が正しいかどうかを検証するためには検証者は Ax を実際に計算し, その各成分が b と一致するかを確認する必要がある. ところが PCP 定理によれば, 巧妙に構成された証明 π を提示することにより, その証明 π 全ての文字を見ることなく, 99%の確率で正しく検証できるのである (ここでは確率的な検証, つまり検証者はランダムネスを用いた検証を行う設定を考える).

このように, 局所的な情報だけを使って全体の構造を推測できるという PCP 定理の性質は, 単に理論的に興味深いだけでなく, 誤り訂正符号の構成, 確率論的手法の脱乱択化, 最適化問題の近似率限界の導出など, 理論計算機科学において広大な応用を持つ.

Chapter 1

導入

この集中講義では PCP 定理と呼ばれる計算量理論の基本的な結果について解説し, その証明を与える. PCP 定理は 1998 年に Arora and Safra [AS98] and Arora, Lund, Motwani, Sudan, and Szegedy [ALMSS98] によって証明された. この証明は代数的な手法に基づく誤り訂正符号を技巧的に組合せたものであり, 難解なものであったが, その後 Dinur [Din07] によってより簡潔な証明が与えられた. この講義では Dinur [Din07] による比較的簡単な証明を紹介する. ちなみに Dinur はのちにこの業績によりゲーデル賞を受賞している.

1.1 計算量理論の復習

まずは計算量理論のどの教科書にも載っているような基礎的な用語の定義を与える. なお, このノートではアルゴリズムの定義 (チューリング機械の定義) は省略し, アルゴリズムについて述べる際は具体的な計算の手続きを述べる[1].

まずは基本的な記号の定義を与える:

- オーダー記法: 二つの関数 $f, g\colon \mathbb{N} \to \mathbb{N}$ に対し, $f(n) = O(g(n))$ であるとは, ある定数 $c > 0$ が存在して, 十分大きな全ての $n \in \mathbb{N}$ に対して $f(n) \leq cg(n)$ が成り立つことをいう. また, $f(n) = \Omega(g(n)), f(n) = o(g(n)), f(n) = \omega(g(n))$ なども同様に ($n \to \infty$ として) 定義する.
- 自然数 $n \in \mathbb{N}$ に対して $[n] = \{1, \ldots, n\}$ とする.
- $\{0,1\}^* = \bigcup_{n\in\mathbb{N}} \{0,1\}^n$ を有限長の二進文字列全体とする.
- $x \in \{0,1\}^*$ に対して $|x|$ を x の文字数とする.
- アルゴリズム A に対し, $A(x)$ を入力 $x \in \{0,1\}^*$ に対するアルゴリズム A の出力とする. ここで, 単に「アルゴリズム」と言った場合は決定的アルゴリズムを指し, 乱択アルゴリズムについては明示的に言及する.

[1] ひとまず Python や C 言語などで実装されたプログラムを考えれば良い. ただし, 計算機内では全ての数値は有限桁の二進数で表記されており, その読み書きや演算には少なくとも桁数に比例した計算時間がかかる. なお, $\sqrt{2}$ といった無理数は本来は有限桁で打ち切った近似値を扱うが, そのような小数はこの講義では扱わず, 特に断りのない限りは整数値のみを考える. また, 記憶領域へのアクセスは定数時間で行えると仮定する (チューリング機械であればテープの移動にかかる時間も考慮する).

- 関数 $T(n)\colon \mathbb{N} \to \mathbb{N}$ を考える. 十分大きな全ての $n \in \mathbb{N}$ と全ての $x \in \{0,1\}^n$ に対して, A が $A(x)$ を出力するまでにかかる計算ステップ数の最大値が高々 $T(n)$ であるとき, アルゴリズム A の計算量は $T(n)$ であるという. 特に, ある (n に依らない) 定数 $c > 0$ が存在して計算量が $O(n^c)$ で抑えられるアルゴリズムを**多項式時間アルゴリズム**という[2]

- 慣例的な記法だが, 二つの文字列 $x, y \in \{0,1\}^*$ を入力として受け取るアルゴリズムは $A(x,y)$ と表す. 三つ以上の場合も $A(x,y,z)$ などと表す.

計算量理論で最も基本的な問題群として判定問題と呼ばれる問題群がある.

定義 1.1.1 (判定問題)

部分集合 $L \subseteq \{0,1\}^*$ を**判定問題 (または言語)** という. また, 文字列 $x \in \{0,1\}^*$ は $x \in L$ であるとき, 判定問題 L の **Yes インスタンス**といい, そうでない場合は **No インスタンス**という.
文字列 $x \in \{0,1\}^*$ に対し $L(x) \in \{0,1\}$ を, $x \in L$ かどうかの指示関数, すなわち $x \in L$ ならば $L(x) = 1$, そうでなければ $L(x) = 0$ と定義する.
アルゴリズム A は, 任意の $x \in \{0,1\}^*$ に対して $A(x) = L(x)$ が成り立つとき, A は L を解くという.

この講義では「効率的に解ける」といった場合, 多項式時間アルゴリズムによって解けることを指す. そのような判定問題の集合をクラス P という.

定義 1.1.2 (クラス P)

判定問題 L は, それを解く多項式時間アルゴリズムが存在するとき, P に属するという.

次に乱択アルゴリズムについて定義する. 端的に言えばアルゴリズムの内部でコイントスを行うものを乱択アルゴリズムという. ここでは明示的にランダムシードを受け取るアルゴリズムを乱択アルゴリズムと呼ぶことにする.

定義 1.1.3 (乱択アルゴリズム)

入力 $x \in \{0,1\}^*$ とは別にランダムシード (乱数表) と呼ばれる別の文字列 $s \in \{0,1\}^*$ を受け取るアルゴリズムを**乱択アルゴリズム**といい, ランダムシードであることを強調するために $A(x;s)$ などと表す. なお, 任意の $x, s \in \{0,1\}^*$ に対して $A(x;s)$ は有限時間で停止するとし, その計算量は $|x|$ のみに依存する関数で表せるとする. このとき, ランダムシード s の長さを常に A の計算量で上から抑える. すなわち, A の計算量が $T(n)$ であるとき, 十分大きな全ての $n \in \mathbb{N}$ と全ての $x \in \{0,1\}^n$ に対して A が読み込む s の文字数は高々 $T(n)$ であるため, $s \in \{0,1\}^{T(n)}$ であると仮定する. しばし, ランダムシード s を明記する必要が特にない場合は $A(x)$ と表す.
乱択アルゴリズムのランダムシードに関する確率, 期待値, 分散を議論する際は記号と

[2] 計算モデルによって一つ一つの計算ステップの定義は異なるが, 原理的には 1bit の演算や記憶領域への読み書きの回数と思えばよい. 多項式時間で動くかどうかの議論であれば, 多くの古典的な計算モデルは等価である.

して $\Pr_A[\cdot], \mathbb{E}_A[\cdot], \mathbf{Var}_A[\cdot]$ を用いる. 乱択アルゴリズム A が判定問題 L を解くとは,

$$\Pr_A[A(x) = L(x)] \geq 2/3$$

が成り立つことをいう.

また, 入力とは別に文字列へのオラクルアクセスを受け取るアルゴリズムを考える.

定義 1.1.4 （オラクルアルゴリズム）

文字列 $\pi \in \{0,1\}^*$ に対し, π への**オラクルアクセス**を持つアルゴリズム $A^\pi(x)$ とは, 計算途中で π の指定された位置の文字を読むことができるアルゴリズムである. すなわち, π の i 番目の文字を読む操作を $A^\pi(x)$ の計算過程中に $O(\log|\pi|)$ 時間で行うことができるアルゴリズムである.[a] 同様に乱択オラクルアルゴリズムについても定義できる.

[a] 自然数 $i \in [|\pi|]$ を指定するために $O(\log|\pi|)$ ビットを定めなければならないため, $O(\log|\pi|)$ 時間を仮定している.

1.2 検証の計算量

数学全般における検証とは, ある命題が真であると主張する証明が与えられたとき, その証明が実際にその命題を正しく証明しているかどうかを確認することを意味する. 論文や記述試験の証明の査読や採点をイメージしてもらうとわかりやすいだろう. 計算量理論では検証やその計算量の議論は重要な研究テーマであり, その検証に要する計算量が議論される.

1.2.1 効率的な検証とクラス NP

判定問題 L と入力 $x \in \{0,1\}^*$ を与えられたとき, $x \in L$ かどうかを審議したい. ここで $x \in L$ を主張する証明が文字列 $\pi \in \{0,1\}^*$ で与えられたとする. このとき, **検証者**と呼ばれるアルゴリズムは x と π を読み込んで $x \in L$ かどうかを判定する. この判定を多項式時間で行えるとき, その判定問題 L の集合を NP という.

定義 1.2.1 （クラス NP）

判定問題 L は, 以下を満たす多項式時間アルゴリズム V と多項式 $p\colon \mathbb{N} \to \mathbb{N}$ が存在するとき, L は NP に属するという: アルゴリズム V は入力として $x, \pi \in \{0,1\}^*$ を受け取り, 0 または 1 を出力する.

1. もし $x \in L$ ならば, ある $\pi \in \{0,1\}^{p(|x|)}$ が存在して $V(x,\pi) = 1$ となる.
2. もし $x \notin L$ ならば, 全ての $\pi \in \{0,1\}^{p(|x|)}$ に対して $V(x,\pi) = 0$ となる.

また, このようなアルゴリズム V を **NP 検証者**といい, π を **NP 証拠**という.

注釈 1.2.2 （クラス P と NP の関係）

判定問題 L が P に属するならば $L \in \mathsf{NP}$ である．実際, 受け取った $x \in \{0,1\}^*$ に対して $L(x)$ を計算してそれを出力する検証者を考えばよい．すなわち $\mathsf{P} \subseteq \mathsf{NP}$ である．一方, 逆側の包含関係 $\mathsf{NP} \subseteq \mathsf{P}$ が成り立つかどうかは P vs NP 問題と呼ばれる計算量理論における最も重要な未解決問題であり, 多くの研究者は $\mathsf{NP} \subseteq \mathsf{P}$ が成り立たないと信じている．

例 1.2.3 （合成数判定問題）

判定問題 $L = \{x \in \{0,1\}^* : x \text{ は合成数 }\}$ を考える．このとき, 検証者は x と π を読み込んで, $\pi \notin \{1, x\}$ かつ π が x を割り切るかどうかを判定する．もしも $x \in L$ である場合, 合成数なので非自明な約数を証拠 π として与えれば $V(x,\pi) = 1$ となる．そうでない場合, 非自明な約数は存在しないため必ず $V(x,\pi) = 0$ となる．このアルゴリズム V は入力長 (つまり数値の二進表現したときのビット長) に関する多項式時間で動作するため, L は NP に属する．

例 1.2.4 （グラフ彩色問題）

自然数 $k \geq 2$ とグラフ $G = (V, E)$ に対し, 関数 $c\colon V \to [k]$ が全ての辺 $\{u, v\} \in E$ に対して $c(u) \neq c(v)$ を満たすとき, c を G の k-彩色といい, k-彩色が存在するようなグラフは k-彩色可能であるという．任意の $k \geq 2$ に対し, 判定問題

$$L = \{G \in \{0,1\}^* : G \text{ は } k\text{-彩色可能 }\}$$

は NP に属する．検証者は G と π を読み込んで, π が G の k-彩色であるかどうかを判定する．もしも $G \in L$ である場合, G は k-彩色可能であるため, k-彩色の証拠 π を与えれば $V(G,\pi) = 1$ となる．そうでない場合, G は k-彩色可能でないため必ず $V(G,\pi) = 0$ となる．このアルゴリズム V は多項式時間で動作するため, L は NP に属する．

演習問題 1 （素数判定問題）

自然数 $n \in \mathbb{N}$ に対し, 判定問題 $\mathrm{PRIMES} = \{a \in \mathbb{N} : a \text{ は素数 }\}$ を考える．この問題は P に属することが知られている [AKS04] が, その複雑なアルゴリズムを用いずに初等的に $\mathrm{PRIMES} \in \mathsf{NP}$ を示したい．そのために, 以下の事実を用いる:
任意の自然数 $a \in \mathbb{N}$ と $\gamma \in \{1, \ldots, a-1\}$ に対し, $\gamma^0, \gamma^1, \ldots \pmod a$ は周期的である．さらに, 以下が成り立つ:

- a が素数であるならば, ある $\gamma \in \{1, \ldots, a-1\}$ が存在して $\gamma^0, \gamma^1, \ldots \pmod a$ の周期が $a-1$ である[a].
- 一方, a が素数でないならば, 全ての $\gamma \in \{1, \ldots, a-1\}$ に対して $\gamma^0, \gamma^1, \ldots \pmod a$ の周期は $a-1$ 未満である．特に, その周期 L は $a-1$ を割り切る．

これらの事実を用いて, 以下の小問に答えよ．

1. 次の検証者 V_1 を考える: 入力 $a \in \mathbb{N}$ と証拠 $\gamma \in \{1, \ldots, a-1\}$ に対し, $\gamma^0, \gamma^1, \ldots, \gamma^{a-2} \pmod a$ を全て検証し, これら全て相異なるかどうかを判定する. この検証者 V_1 が多項式時間アルゴリズム**でない**理由を簡潔に説明せよ.

2. 入力 $a \in \mathbb{N}$ に対し, a が素数であることの証拠として, 原始元[a] $\gamma \in \{1, \ldots, a-1\}$ および $a-1$ の素因数分解 $a-1 = p_1^{\alpha_1} \cdots p_k^{\alpha_k}$ および各 p_i が素数であることの証拠を再帰的に与える. この証拠を用いて, 検証者 V_2 は a が素数であるかどうかを多項式時間で判定できることを示せ.

[a] このような γ を原始元という.

1.2.2 局所的な検証とクラス PCP

一般に $x \in L$ かどうかの検証では, 証明 π の全ての文字を読む必要がある. しかし, 証明 π のうちの一部の文字を読むだけで $x \in L$ かどうかを**確率的に**判定できる場合がある. そのような性質を持つ証拠を**確率的検証可能な証拠** (Probabilistically Checkable Proof, PCP) という.

定義 1.2.5 (確率的検証可能な証拠)

二つの関数 $r, q \colon \mathbb{N} \to \mathbb{N}$ に対し, $\mathsf{PCP}(r, q)$ を以下の性質を持つ判定集合 L の集合とする: ある多項式時間オラクル乱択アルゴリズム V が存在して, 任意の $x \in \{0,1\}^*$ に対し,

1. もし $x \in L$ ならば, ある $\pi \in \{0,1\}^*$ が存在して, (V の乱択に関して) 確率 1 で $V^\pi(x) = 1$ となる.

2. もし $x \notin L$ ならば, 全ての $\pi \in \{0,1\}^*$ に対して, (V の乱択に関して) 確率 $1/3$ 以上で $V^\pi(x) = 0$ となる.

3. さらに, 入力長が $n = |x|$ のとき, $V^\pi(x)$ はオラクル π のうち高々 $q(n)$ 個の文字を読み, そのランダムシード長は $r(n)$ で抑えられる.

このようなオラクル乱択アルゴリズム V を **PCP 検証者**といい, 証拠 π を **PCP** という.

注釈 1.2.6 (PCP の長さ)

$\mathsf{PCP}(r, q)$ の証拠 π の長さは $q(n)2^{r(n)}$ で抑えられる. 各ランダムシード $s \in \{0,1\}^{r(n)}$ に対して検証者は π のうち高々 $q(n)$ 個の文字を読むため, 全てのランダムシードを列挙すると, アクセスされる可能性のある π の文字数は高々 $q(n)2^{r(n)}$ で抑えられる.

例 1.2.7 (グラフ彩色問題)

...

1.3 PCP 定理

PCP 定理とは, ある $r(n) = O(\log n)$, $q(n) = O(1)$ に対して $\mathsf{PCP}(r, q) = \mathsf{NP}$ が成り立つことを主張する定理である.

定理 1.3.1 （PCP 定理）

…

Chapter 2

制約充足問題

制約充足問題 (Constraint Satisfaction Problem, CSP) は, 論理学や計算複雑性理論において重要な問題の一つであり, PCP 定理の証明においても中心的な役割を果たす.

2.1 制約充足問題の定義

2.1.1 PCP との関係

2.1.2 制約グラフ

2.2 PCP 定理の証明の概要

2.3 エクスパンダーグラフ

2.3.1 エクスパンダーグラフの定義

2.3.2 エクスパンダー混交補題

2.3.3 制約グラフのエクスパンダー化

Chapter 3

ギャップ増幅補題

この章ではPCP定理の証明において重要な役割を果たすギャップ増幅補題について解説する.

3.1　主張と直感

3.2　証明

3.2.1　構成

3.2.2　正当性の証明

Chapter 4

アルファベット削減

4.1 PCP 定理の証明

Bibliography

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. "PRIMES is in P". en. In: **Annals of mathematics** 160 (2 Sept. 1, 2004), pp. 781–793. DOI: 10.4007/annals.2004.160.781 (cit. on p. 10).

[ALMSS98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. "Proof verification and the hardness of approximation problems". In: **Journal of the ACM** 45 (3 May 1, 1998), pp. 501–555. DOI: 10.1145/278298.278306 (cit. on p. 7).

[AS98] S. Arora and S. Safra. "Probabilistic checking of proofs: a new characterization of NP". In: **Journal of the ACM** 45 (1 Jan. 1, 1998), pp. 70–122. DOI: 10.1145/273865.273901 (cit. on p. 7).

[Din07] I. Dinur. "The PCP theorem by gap amplification". In: **Journal of the ACM** 54 (3 June 1, 2007), 12–es. DOI: 10.1145/1236457.1236459 (cit. on p. 7).