数理解析研究所講究録
第436巻 1981年 132-155

計算の難しさについて

電気通信大学　　笠井琢美

1 はじめに

計算の複雑さに関する理論の目的は次の二点にある。第1はより効率のよいアルゴリズムを求めることである。アルゴリズムを設計した場合，それがどのぐらい効率的かということ，即ちそれを実行するのにどのぐらいの時間を必要とするかといった時間量（time complexity）と，どのぐらいの記憶領域を必要とするかといった領域量（space complexity）とを示す必要がある。効率のよいアルゴリズムの研究は，アルゴリズムの概念の歴史と同じ起源を持つものであるが，最近でもグラフとかソーティングなどといった非数値計算における効率的なアルゴリズムが数多く開発されており，活発な活動が続けられている。Knuth[11], Aho, Hopcroft, Ullman[2], Baase[4]にはその広範な解説と発展の歴史が述べられている。第2は問題の"本質的な難しさ"を示すことである。すなわち，ある

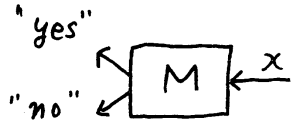
問題を解くために，"どのような"アルゴリズムを作っても，ある時間量（又は領域量）が必要であることを示すこと，いいかえれば，これこれの時間以下では絶体解けないといった下界を求めることであり，計算機の規模と限界を有効に利用するために第1の問題と同様に大切なことである．ここでは第2の問題，"計算量の下界をいかにして求めるか"について論ずる．

原理的計算可能性と実際的計算可能性

計算の理論の基本的概念の一つに計算可能性というのがある．ある問題が計算可能であるとは，その問題を解くアルゴリズムが存在するときをいう．計算可能でない問題は，いまでに数多く発見されている．このような計算可能性という概念をここでは原理的計算可能性と呼ぶことにする．ある問題が原理的に計算可能であるとしても，その問題がただちに"実際的に計算可能"であるとはいえない．その問題を計算機で解こうとする場合，ある限定された時間，限定された記憶容量のもとで解かなければならないからである．こうした計算量を考慮した上での実際的計算可能性についての理論，即ち従来の計算可能性についての議論を，限定された計算量に対し展開したのが計算量の下界についての理論である．

2. 問題と計算機のモデル化

Σをアルファベットとする. Σ^*の部分集合Lのことを問題という. 問題Lを解く機械(又はアルゴリズム, プログラム)とは次のような機械Mのことをいう.



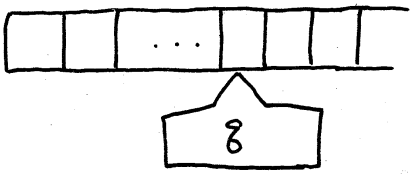
MはΣ^*の元xが入力として与えられたとき, $x \in L$なら"yes"と答え, $x \notin L$なら"no"と答える. このとき, MはLを受理するとかLを計算するなどと言うこともある.

符号化 ある具体的な問題の計算量について論ずるには, その問題の対象がどのように文字列に符号化されるかを明確にしなければならない. ここでは次のような符号化を取ることにする. 自然数は十進数として$\{0,1,\cdots,9\}^*$の元で表わす. (有向)グラフでは, 頂点は$\{0,1,\cdots,9\}^*$の元で表わし, 辺は頂点の組(u,v)で表わし, (有向)グラフ自身は辺の列$(u_1,v_1),(u_2,v_2),\cdots,(u_m,v_m)$で表わす. したがって(有向)グラフは$\{0,1,2,\cdots,9,(,),,\}^*$の元で表わされる.

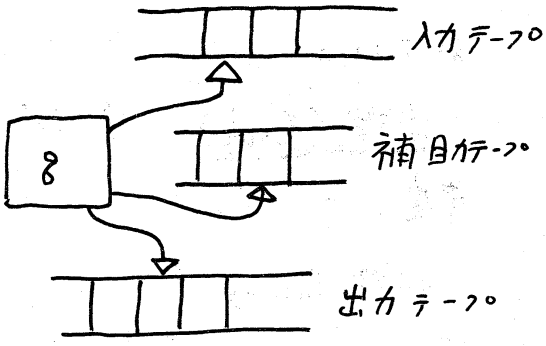
例 1. 素数の判定問題: 与えられた正の整数xが素数であるかどうかを判定する問題. この問題を表わす集合は, $\{0,1,\cdots,9\}^*$の部分集合 $L=\{x \mid x$は素数を表わす$\}$である.

例2. 3彩色問題　与えられたグラフGが3彩色可能かどうかを判定する問題．即ち隣接する2つの頂点に同じ色をぬらないように，Gのすべての頂点を3色でぬれるかどうかを判定する問題．この問題を表わす集合は $L=\{G \mid$ 文字列Gは3彩色可能なグラフを表わす$\}$.

計算量について論ずるには，アルゴリズムを実行する媒体を明確にしなければならない．ここでは，そのような計算機のモデルとしてTuring機械を採ることにする．



単テープTuring機械



オフラインTuring機械

Turing機械は一本（又は複数本）のテープを使用できる．テープはます目に分割されており，各ます目には文字が書き込める．テープには文字を読んだり書いたりするheadがくっついていて，headは左または右に一こますつ移動できる．有限個の状態のうち一つの状態を取ることができ，次の動作は現在の状態と現在headがあるます目の文字によって決定される．状態のうちいくつかが受理状態として指定されてい

て，機械が受理状態に達したなら，入力は受理される．一つの動作（即ち一単位時間で）次が実行される．

(1) 状態を変える．

(2) head があるます目の文字を書き替える．

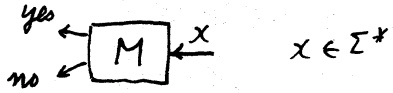
(3) head を左または右に一こま移動させる

RAM (random access machine)　Turing 機械は非常に理想化された単純なモデルである．より現在の計算機に忠実なモデルとして RAM と呼ばれるものがある．しかし次が示されている．

定理 (Cook)　RAM と Turing 機械は計算量のことを考慮してもその能力にあまり差がない．

以後 Turing 機械のことを単に機械と呼ぶ．しかし以下の議論のほとんどの部分は，十分強力な計算モデルであれば (RAM でもその他のモデルでも) なんであってもかまわない．たとえばあるプログラミング言語を固定し，その言語で書かれたプログラムのことを機械と思ってもよい．

定義　L を問題，$L \subset \Sigma^*$，とする．M を L を計算する機械とする．



$time_M(x)$ = 入力 x のもとで M が必要とする時間 (ステップ数)

$space_M(x)$ = 入力 x のもとで M が必要とする領域 (ます目の数)

Nを自然数全体から成る集合とし，$T: N \to N$と$S: N \to N$を関数とする．MがT時間で計算できるとは，任意の$x \in \Sigma^*$に対し

$$time_M(x) \leq T(|x|)$$

となるとき，またMがS領域で計算できるとは，任意のxに対し

$$space_M(x) \leq S(|x|)$$

となるときをいう．ここで$|x|$は語xの長さを示す．

問題LがT時間（S領域）で計算できるとは，LをT時間（S領域）で計算する機械Mが存在するときをいう．

Tは$T(n) \geq n$と仮定してよい．$S(n) < n$なる領域量について議論するときは図のようなオフラインTuring機械を考える．ここで入力テープへの書き込みは禁止する．出力がある場合は，出力テープのheadは右方向にしか移動できないものとする．（即ち出力テープも補助記憶としての使用を禁止する）このとき領域量は使用した補助テープのます目の総数である．

定義　関数$f: \Sigma^* \to \Delta^*$を考える．ここでΣとΔはアルファベット．機械Mがfを計算するとは任意の$x \in \Sigma^*$に対しMの出力が$f(x)$となるときをいう．

関数の時間量と領域量は，問題のときと同様に定義される．

3. 問題の階層

問題を計算量によって分類しようとする場合、まず考えなければならないことは、そもそもそういった分類が可能であるかどうかということである。たとえばn^3時間では計算できるがn^2時間では計算できないような問題は存在するか？与えられた関数Tに対し、T時間以上の時間量を必要とするような問題は存在するか？　ここで述べる階層定理はこれに答えるものである。

現在得られているいろいろな具体的な問題の計算量の下界はすべてこの階層定理を使って得られる。階層定理は対角線論法と呼ばれる論法で証明される。いいかえれば、対角線論法が現在の所、計算量の下界を示すための唯一の手法である。

対角線論法

Σをアルファベットとする。各機械MはΣ上の語として表わされるものとする。(以下の議論では、機械と呼ぶかわりにプログラムと呼んだ方がわかりやすいかも知れない。)

機械Mに対し、

$$L(M) = \{ x \in \Sigma^* \mid M \text{は} x \text{を受理する。即ち} \boxed{M} \leftarrow x \text{が} yes \}$$

と定める。今、集合$A \subset \Sigma^*$を

$$A = \{ M \mid M \notin L(M) \}$$

と定める。

定理　問題 A は計算できない.

証明　A を計算する機械 M が存在したとする. 即ち $A = L(M)$ とする. すると

$$M \in L(M) \Leftrightarrow M \notin L(M)$$

となり矛盾.

階層定理　$T: N \to N$ と $S: N \to N$ を関数とする. Σ^* の部分集合 A_T と A_S を次のように定める.

$A_T = \{M \mid$ 機械 M は M を $T(|M|)$ 時間内で受理(なり$\}$

$A_S = \{M \mid$ 〃 $S(|M|)$ 領域 〃 $\}$

定理　A_T は T 時間では計算できない.

A_S は S 領域では計算できない.

証明　$A_T = L(M)$, M の時間量は T とする.

$$M \in A_T \Leftrightarrow M \notin A_T$$

となり矛盾

A_S についても同様の矛盾が得られる.

T を計算可能な関数とする. A_T は T 時間では計算できないが T 以上の時間をかければ計算できる. A_T を計算する機械 U は次のように構成される. 入力 M が与えられたとする. まず U は入力の長さ $n = |M|$ を測り, $T(n)$ を計算する. 次に M の模倣を始める. 即ち M を解読し M の命令を一つづつ解釈

実行する．Mの命令を一つ実行するごとに，計算したT(n)の値から1づつ減ずる．この値が0になるまでにMがxをMを受理しないならUはMを受理する．それ以外の時はUはMを受理しない．

以上の議論をさらに精密に行うと次の階層定理を得る．

定義 関数 $T: N \to N$ が時間構成可能であるとはTが$O(T)$時間で計算できるときをいう．(即ち2進数nが与えられたとき2進数$T(n)$を高々$c \cdot T(n)$時間で計算する機械が存在するときをいう．) また関数 $S: N \to N$ が領域構成可能であるとは，Sが$O(S)$時間で計算できるときをいう．

時間量に関する階層定理 T_1, T_2 が時間構成可能な関数で，次が成立するとする

$$\inf_{n \to \infty} \frac{T_1(n) \log T_1(n)}{T_2(n)} = 0$$

このとき，T_2時間では計算できるが，T_1時間では計算できない問題が存在する．

領域量に関する階層定理 S_1, S_2 を次を満す領域構成可能な関数とする．このとき S_2 領域では計算できるが，S_1領域では計算できない問題が存在する．

$$\inf_{n \to \infty} \frac{S_1(n)}{S_2(n)} = 0$$

4. いかにして計算量の下界を示すか

還元可能性 A と B を問題とする. $A \subset \Sigma^*$, $B \subset \Sigma^*$ とする. $f: \Sigma^* \to \Delta^*$ を関数とする. 問題 A は f によって問題 B に還元される とは, 任意の $x \in \Sigma^*$ に対し,

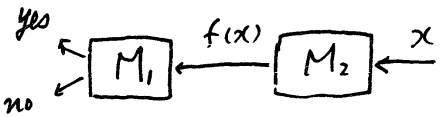
$$x \in A \iff f(x) \in B$$

が成立するときをいう. このとき

$$A <_f B$$

と書く.

$A <_f B$ とする. B を計算する機械 M_1 と f を計算する機械 M_2 があったとする. すると A は M_1 と M_2 を使って次のように計算することができる.



$x \in A$ なら yes
$x \notin A$ なら no

ふつう f が容易に計算できると仮定する. 即ち A とか B の計算量にくらべ, f の計算量は無視できると仮定する. すると, $A <_f B$ は "A より B のほうが難かしい" こと, いいかえれば "A が難かしい問題なら B も難かしい" ことを意味する.

完全問題 $\mathcal{C}$ を問題のクラス, $\mathcal{F}$ を関数のクラスとする. 問題 B が $\mathcal{F}$ に関し $\mathcal{C}$ 困難 ($\mathcal{C}$ hard) であるとは, $\mathcal{C}$ の任意の元 A に対し, $f \in \mathcal{F}$ が存在し

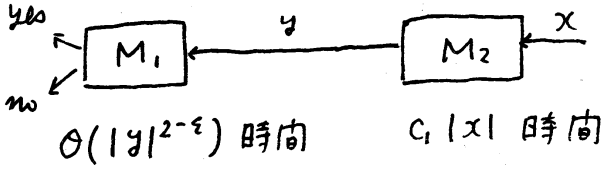
$$A <_f B$$

となるときをいう．さらにBが𝒞の元なら，Bは𝒻に関し𝒞完全（𝒞 complete）であるという．

例　𝒞を$O(n^2)$時間で計算できる問題のクラスとする．𝒻を線形時間で計算できる関数のクラスとする．Bを𝒻に関し𝒞困難な問題とする．次を示そう

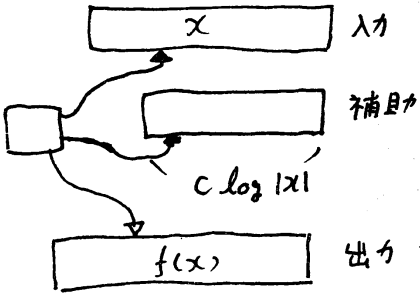
◎ どんな$\varepsilon>0$に対しても，Bは$O(n^{2-\varepsilon})$時間では解けない．

Bを$O(n^{2-\varepsilon})$時間で解く機械Mが存在したと仮定する．Aを𝒞の任意の元とする．$A <_f B$なる$f \in$𝒻が存在する



$|x|=n$とすると$|y| \le c_2|x|$．（但しc_1, c_2はfだけに依存して定まる定数）よってAは$O(n^{2-\varepsilon})$時間で解ける．したがって，$O(n^2)$時間で解けるどんな問題も$O(n^{2-\varepsilon})$時間で解けることになる．これは階層定理に矛盾する．

定義　AとBを問題とする．多項式時間計算可能な関数fが存在し$A <_f B$となるときAはBに多項式時間還元可能であるという．また対数領域計算可能な関数fが存在し$A <_f B$となるとき，AはBに対数領域還元可



能であるという.

f が対数領域計算可能なら（即ち，ある定数 c に対し，f が $c\log n$ 領域で計算できるなら）f は多項式時間計算可能である.

以後特にことわらないかぎり，$\mathcal{F}$ は対数領域計算可能な関数のクラスとする. したがって単に A は B に還元可能であると言ったときは，A は B に対数領域還元可能であることを意味する. また単に $\mathcal{C}$ 完全（$\mathcal{C}$ 困難）といったときは，対数領域計算可能な関数のクラス $\mathcal{F}$ に関し $\mathcal{C}$ 完全（$\mathcal{C}$ 困難）であることを意味する.

以下で次のような問題のクラスについて論ずる.

DLOGSPACE = {対数領域計算可能な問題}

NLOGSPACE = {非決定性の機械で対数領域計算可能な問題}

P = {多項式時間計算可能な問題}

NP = {非決定性の機械で多項式時間計算可能な問題

PSPACE = {多項式領域計算可能な問題}

EXP = {指数関数時間 計算可能な問題，即ち ある
多項式 $p(n)$ が存在し，$2^{p(n)}$ 時間で計算できる問題}

定義より，上のクラスは下のクラスに含まれることがわかる. また次がわかっている.

$$\mathrm{NLOGSPACE} \subsetneq \mathrm{PSPACE}, \qquad \mathrm{P} \subsetneq \mathrm{EXP}$$

5. 石置きゲーム

ここではある具体的な問題が完全問題となることをいかにして示すかについて述べる. まず石置きゲームと呼ばれるゲームを考える. この石置きゲームにいろいろな制限を加えることによって, いろいろな問題のクラスの完全問題が得られる.

定義 石置きゲーム (pebble game) とは $G=(X,R,S,t)$ のことをいう. ここで

(i) X は有限集合で X の元を頂点と呼ぶ.
X の元の個数を G の次数 (order) と呼ぶ

(ii) $R \subseteq \{(x,y,z) \mid x,y,z \in X,\ x\neq y,\ y\neq z,\ z\neq x\}$
R の元を規則と呼ぶ

(iii) $S \subseteq X$. S の元の個数を G の階数 (rank) と呼ぶ.

(iv) $t \in X$. t を最終頂点と呼ぶ.

石置きゲーム G が与えられたとき, X の部分集合上の関係 $\vdash$ を次のように定義する. X の部分集合 A と B に対し, $A \vdash B$ である必要十分条件は, $(x,y,z)\in R$ が存在し,

$$x,y \in A,\quad z \notin A,\quad B=(A-\{x\})\cup\{z\}$$

となることである. X の部分集合 $A_0, A_1, \cdots, A_m$ に対し $A_0 \vdash A_1 \vdash \cdots \vdash A_m$ となるとき $A_0 \vdash^* A_m$ と書く. G が解を持つとは, $S \vdash^* A$ で $t\in A$ となる A が存在することである.

石置きゲーム G は次のようなゲームである．最初に S の頂点に一つずつ石が置いてある．R の元 (x, y, z) は頂点 x と y に石があり，頂点 z に石がないとき，x にある石を z に移すことができることを意味する．G が解を持つのは，S から始めて，R の規則を何回か使って頂点 t に石を置くことができるときである．

例 石置きゲーム $G=(X, R, S, t)$ を次のように定める．

$X=\{a, b, c, d, e, t\}$, $S=\{a, b, c\}$

$R=\{(a,b,d), (b,c,d), (c,d,e), (b,e,a), (a,d,b), (a,e,t)\}$

すると

$S=\{a,b,c\} \vdash \{a,d,c\} \vdash \{a,d,e\} \vdash \{t,d,e\}$

となる．即ち最初 b にある石を d に移し，次に c にある石を e に移し，最後に a にある石を t に移す．よって G は解を持つ．

定義 与えられた石置きゲームが解を持つかどうかを判定する問題を石置きゲーム問題と呼ぶ．

定理 [10] 石置きゲーム問題は PSPACE 完全である．

証明 石置きゲームが多項式領域で計算できることは，実際に石置きゲームを解く機械（プログラム）を作ってみせることによって示される．この構成はアルゴリズムにある程度理解のある人なら容易にできるので，ここでは省略する．

多項式領域で計算できるどんな問題も，石置きゲームに還

元できることを示そう. $A \subset \Sigma^*$ を問題とし, MをAを解く機械とする. Mの領域量を $S(n)$ とする. 但し Sは多項式. Mは単テープ Turing 機械とする. Σ^* の各元 x に対し

$$x \in A \iff G_x \text{ は解を持つ}$$

となるような石置きゲーム G_x を構成すればよい. Mの状態の状態の集合をQ, 初期状態を q_0, 受理状態を q_f, Mが使用するアルファベットを Γ とする. したがって $\Sigma \subseteq \Gamma$ である. Γ は空白記号 b を含み, $Q \cap \Gamma = \phi$ とする. $x = x_1 x_2 \cdots x_n$, $(x_1, \cdots, x_n \in \Sigma)$ とする.

$G_2 = (X, R, S, t)$ を次のように定める.

$$X = X_1 \cup X_2 \cup X_3 \cup \{t\}$$

$$X_1 = \{ (i, a) \mid 1 \le i \le S(n),\ a \in \Gamma \}$$

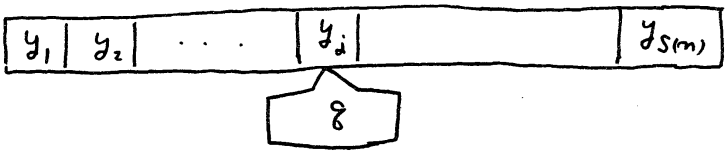
$$X_2 = \{ (j, q) \mid 1 \le j \le S(n),\ q \in Q \}$$

$$X_3 = \{ (j, q, a) \mid 1 \le j \le S(n),\ q \in Q,\ a \in \Gamma \}$$

$$S = \{ (1, q_0), (1, x_1), (2, x_2), \cdots, (n, x_n), (n+1, b), \cdots, (S(n), b) \}$$

X_1 の頂点 (i, a) に石があることは, Mのテープの i 番目のます目に文字 a が書かれていることを示し, X_2 の頂点 (j, q) に石があることは, Mの現在の状態が q で head の位置が j であることを示す. X_3 の頂点はMの動作を模倣するために補助的に使用される. Sの頂点はMの最初の状況を表わす. 即ち状態は q_0 で, head は最初のます目にあり, i 番目のます

目には文字 x_i が書かれており，入力 x の右側のます目には空白が書かれている．今，Mの状況が



であったとする．このとき G_x では次の頂点に石が置かれる．

$$(1, y_1), (2, y_2), \cdots, (j, y_j), \cdots, (S(n), y_{S(n)}),$$
$$(j, q)$$

今Mが次の命令を持ったとする．

「現在の状態が q で，headが読んでいる文字が a なら，その文字を b に書き替え，headを d ($d=-1, 0, 1$) だけ移動し，状態を p に変える」

このとき，Rは各 i ($1 \leq i \leq S(n)$) に対し，次を含む．

$$((i, q), (i, a), (i, q, a))$$
$$((i, a), (i, q, a), (i, b))$$
$$((i, q, a), (i, b), (i+d, \ ))$$

この規則によってMの一つの動作が模倣できる．さらに，各 i, $1 \leq i \leq S(n)$ と各 $a \in \Gamma$ に対し，Rは次を含む．

$$((i, q_f), (i, a), t)$$

この規則は，Mが受理状態に入るなら，G_x で頂点 t に石が置けることを意味する．したがって x がMに受理される必要

十分条件は，G_x が解を持つこととなる．

Σ^* の元 x が与えられたとき，このような G_x が"実際的"な時間で構成できること（正確には対数領域で計算できる，したがって多項式時間で計算できること）は明らかであろう．

6 いろいろなクラスの完全問題

問題Bが©完全であったとする．直観的には，問題Bは問題のクラス©の中で一番難かしい問題，いいかえればクラス©を代表する問題であるということができる．したがって問題Bについて調べることはクラス©全体を調べることに等しい．ここでは §4 であげた問題のクラスについて，それぞれの完全問題を与える．

2人石置きゲーム　与えられた石置きゲーム $G=(X, R, S, t)$ を2人で行う．最初にSの頂点に石が置かれた状態が始め，交互にRの規則を使って石を動かす．最初にtに石を置いたちが勝ち，あるいは途中で石を動かせなくなったちが負けとする．2人石置きゲーム問題とは，Gが与えられたとき，先手が必勝戦略を持つか否かを決定する問題をいう．

線形有界な石置きゲーム　石置きゲーム G が線形有界であるとは $A_0 \vdash A_1 \vdash \cdots \vdash A_t$ ならば，すべての A_i が成り立つこと，即ち n 手以上連続して規則が適用できないときをいう．n は G の次数．

石置きゲームに関し次がわかっている.[ ]

	1人ゲーム	2人ゲーム
階数3の石置きゲーム	NLOGSPACE 完全	P完全
線形有界な石置きゲーム	NP完全	PSPACE 完全
一般の石置きゲーム	PSPACE 完全	EXP 完全

すでに©完全であることがわかっている問題Bを使えば,新しい問題Cが©完全であることを,容易に示すことができる.それには,Cが©に属することと,問題Bが問題Cに還元できることを示せばよい.以下でそのような例を少し見てみよう.

<u>道発見問題</u> 有向グラフ $G=(X,E)$ とGの2つの頂点sとtが与えられたとき,Gにおいて頂点sから頂点tへの道があるかどうかを判定する問題を道発見問題という.

<u>グラフ上の2人ゲーム</u> 有向グラフGとGの頂点sが与えられたとする.2人で次のようなゲームを行う.最初sに石が置いてある.ある時点で頂点xに石があり,xからyに向う辺があるなら,xにある石をyに移せる.最初に石を動かせなくなった方が負けとする.このゲームで,先手が必勝戦略を持つかどうかを判定する問題を,グラフ上の2人ゲーム問題という.

定理 (i) (Savitch) 道発見問題はNLOGSPACE完全である.
(ii) (Jones and Lasser) グラフ上の2人ゲーム問題はP完全である.

証明 道発見問題が非決定性の機械で対数領域で計算できること, およびグラフ上の2人ゲームが多項式時間で計算できることは明らかであろう. したがって (i), (ii) を示すには階数3の石置きゲームが道発見問題に還元できること, 階数3の2人石置きゲームがグラフ上の2人ゲームに還元できることを示せばよい.

$G=(X, R, S, t)$ を階数3の石置きゲームとする. グラフ $G'=(X', E)$ を次のように構成する.

$$X''=\{\{x,y,z\} \mid x,y,z\in X,\ x\neq y,\ y\neq z,\ z\neq x\}$$

$$X'=X''\cup\{t_1,t_2\},\quad t_1 と t_2 は新しい頂点$$

$$E=\{(A,B) \mid A,B\in X'',\ t\notin A かつ G で A\vdash B\}$$
$$\cup\{(A,t_1) \mid A\in X'',\ t\in A\}$$
$$\cup\{(t_1,t_2)\}$$

すると

G が解を持つ $\Leftrightarrow$ G' で S から t_2 への道が存在

G は先手必勝 $\Leftrightarrow$ (G', S) は先手必勝

7 より正確な計算量の下界

いままでは，ある問題が多項式時間で解けるかどうかなどという，た，非常に大ざっぱな分類を行なってきた．しかし，ある問題が多項式時間で解けたとしても，"実際的"な時間で解けるとは限らない．たとえば n^{100} 時間以上かかるような問題は実際的だとはいえない．ここでは，より正確な計算量の下界をいかにしてもとめるか，いいかえれば，ある問題が多項式時間で解けるなら，それはどの程度の多項式となるかを具体的にもとめる方法について論ずる．

より正確な計算量の下界について論じようとすると，いままで問題とならなかったことが問題となってくる．その第1は問題のサイズに関してである．いままでは，問題のサイズは入力の文字列としての長さとした．そして計算量は入力の長さの関数として測った．たとえばグラフを扱う問題を例にとって考えよう．グラフのサイズとしては，頂点の個数を取ることもあるし，辺の個数を取ることもある．頂点の個数を n とすると，辺の個数は n^2 までになりうる．多項式時間で解けるかどうかという議論では，このどれをサイズとしても同じであるが，正確に計算量の下界を求めようとすると何をサイズとするかを明確にしなければならない．

定義　以後問題とは3組(Σ, A, σ)のことをいう．ここでΣはアルファベット，AはΣ^*の部分集合，σはΣ^*からNへの計算可能な関数である．Σとσが明らかなときは，単に問題Aという．σは問題Aのサイズ関数といい，Σ^*の元xに対し，$\sigma(x)$をxのサイズと呼ぶ

$T: N \to N$を関数とする．問題Aがサイズに関しT時間で計算できるとは，Aを計算する機械Mが存在して，Mは任意の入力xに対し，高々$T(\sigma(x))$時間で停止するときをいう．問題AがT時間必要であるとは，任意の関数$T': N \to N$に対し，もし

$$\inf_{n \to \infty} \frac{T'(n)}{T(n)} = 0$$

なら，AはT'時間では計算できないときをいう．

低いレベルの計算量を議論する場合に生ずるもう一つの問題点は，いままでの還元可能性の概念がもはや適用できなくなることである．たとえば多項式時間還元可能という概念はNP以上のクラスには適用できるが，P以下のクラスでは意味を持たない．多項式時間計算可能などんな問題Aも，2つの元$\{0,1\}$だけから成る問題Bに多項式時間還元可能である．したがって正確な下界をおめるためには，より強い還元可能性の概念が必要となる．

◎ 以後議論を簡単にするために，計算量は $c \cdot n^t$ の形の関数だけを考えることにする．n は入力のサイズを表わす変数で，c と t は正の実数とする．

補題 (Σ, A, σ_1) と (Δ, B, σ_2) を問題とする．A がサイズに関し $O(n^k)$ 時間必要とすることがわかっていたとする．次を満す関数 $f: \Sigma^* \to \Delta^*$ が存在したとしよう．x を Σ^* の任意の元とし，$\sigma_1(x) = n$ とする．

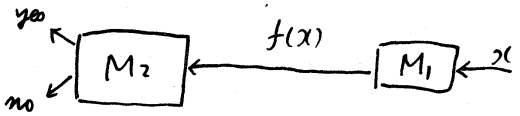
(i) $x \in A \iff f(x) \in B$

(ii) $f(x)$ は $O(n^l)$ 時間で計算できる．但し $l < k$．

(iii) $\sigma_2(f(x)) \leq O(n^\alpha)$

すると B は，サイズに関し，$O(n^{\frac{k}{\alpha}})$ 時間必要とする．

証明 B が $O(n^t)$ 時間で計算できるとしよう．



すると A は $O(n^{\alpha t}) + O(n^l)$ 時間で計算できる．仮定より $l < k$ であり，A は n^k 時間必要とするが，$\alpha t \geq k$ でなければならない．

$O(n^k)$ 時間必要とする問題

使用できる石の数（階数）が固定された石置きゲームを考えよう．石置きゲームのサイズは G の次数，即ち頂点の個数

とする．Gを階数 k，次数 n の石置きゲームとする．k 個の石を n 個の頂点に置く置き方は高々 n^k である．したがって階数 k の石置きゲームが与えられたとき，Gで先手が必勝手を持つかどうかを判定する問題は次数 n の多項式時間で計算できる．(頂点の個数が高々 n^k のゲームの木を作ればよい)

上の補題で述べた還元可能性を使って次が示される．[1]

定理　各 k に対し，階数 $2k+1$ の2人石置きゲーム問題は $O(n^k)$ 時間必要である

k 匹のねずみとねこの問題　ねことねずみゲームとは $G=(X,E,M,c,t)$ のことを言う．ここで (X,E) は有向グラフ，$M \subset X$，$c, t \in X$．M は k 個の元から成るとする．最初 M の頂点に k 匹のねずみがおり，頂点 c にねこがいる．t にはチーズがある．最初ねこが動き，次に k 匹のねずみのうち1匹が動き，以後ねこねずみが交互に動く．ねこねずみが同一の頂点に達したらねこの勝ち，ねずみのうち一匹がチーズに達したらねずみが勝ちとする．問題のサイズは頂点の個数とする．k 匹のねずみとねこの問題とはねことねずみゲーム G が与えられたとき，ねこが必勝手を持つかどうか判定する問題．

定理　$2k+2$ 匹のねずみとねこの問題は $\Theta(n^k)$ 時間必要とする．

文献

[1] A. Adachi, S. Iwata, T. Kasai, Low Level Complexity for Combinatorial Games. Proc. ACM Symposium on the Theory of Computing 1981

[2] A.V. Aho, J.E. Hopcroft, J.D. Ullman, The Design and Analysis of Computer Algorithms. 1974. 野崎・野下他訳 サイエンス社

[3] A.V. Aho. Currents in the Theory of Computing, 1973 宇屋悦朗訳：最近の計算機科学，近代科学社

[4] S. Baase : Computer Algorithms, Introduction to Design and Analysis, 1978

[5] S.A. Cook, R.A. Reckhow, Time bounded random access machines J. Comput. System. Sci. 7, 1973, 354-375

[6] M. Davis, Computability and Unsolvability. 渡辺茂，赤攝也訳 岩波書店

[7] M.R. Garey and D.S. Johnson : Computers and Intractability a guide to the theory of NP-completeness. 1979

[8] J.E. Hopcroft and J.D. Ullman : Formal Language and Their Relation to Automata, 1969. 野崎・木村訳，サイエンス社

[9] T. Kasai, A. Adachi. A Characterization of Time Complexity by Simple Loop Program. J. Comput. System. Sci. 20. 1980, 1-17

[10] T. Kasai, A. Adachi, S. Iwata, Classes of Pebble games and Complete Problems, SIAM J. on Comput. 8, 1979

[11] D. Knuth : The Art of Computer Programming. vol I (1968) Vol II (1969), Vol III (1973).