2 計算効率の評価

計算幾何問題を解くためのアルゴリズムを設計するにあたって，最も注意の払われるのが計算効率の善し悪しです．この計算効率の評価には次の 3 つの方法が用いられます．

経験的解析 (empirical analysis)**:** 問題例の分布を特定してサンプリングを行い，それらに対するアルゴリズムの振る舞いを実際に計測する．

最悪計算量解析 (worst-case analysis)**:** 任意の問題例に対し，アルゴリズムが要する計算量の上限を解析する．

平均計算量解析 (average-case analysis)**:** 問題例の分布に特定の仮定を設け，アルゴリズムが要する計算量の平均を解析する．

この中で平均計算量解析は一般に証明が難しいため，アルゴリズムの性能評価に広く用いられているのは経験的解析と最悪計算量解析です．ここでは計算幾何の理論的な柱となる後者について説明しましょう．

2.1 最悪計算量解析

最悪計算量解析では，四則演算や代入，比較などの基本演算 1 回が単位時間で実行されるものと仮定し，問題サイズを表すいくつかのパラメータ — 例えば，与えられた多角形の個数 m やその頂点総数 n など — を用いてアルゴリズムの実行時間の上限を算定します．

計算幾何問題ではありませんが，次の問題を考えてみましょう:

0-1 ナップサック

入力: 品物の集合 N, 各 $j \in N$ に対する大きさ $a_j \in \mathbb{Z}_+$，価値 $c_j \in Z_+$, およびナップサックの容積 $b \in \mathbb{Z}_+$ と目標値 $z \in \mathbb{Z}_+$．

質問: $\sum_{j \in M} a_j \leq b$ かつ $\sum_{j \in M} c_j \geq z$ を満たす部分集合 $M \subseteq N$ は存在するか？

この問題のサイズは，品物の数 $n = |N|$，その大きさ a_j，価値 c_j，ナップサックの容積 b と目標値 z によって定まり，答は最適化問題

$$\left|\begin{array}{ll} \text{最大化} & \displaystyle\sum_{j\in N} c_j x_j \\ \text{条　件} & \displaystyle\sum_{j\in N} a_j x_j \le b \\ & x_j \in \{0,\ 1\},\quad j \in N, \end{array}\right.$$

を動的計画法で解くことによって求められます．つまり，最適化問題の最適値が z 以上であれば “yes”，そうでなければ “no” が答です．品物 1 から k を容積 y のナップサックに詰めたときの価値の総和を $V_k(y)$ で表すことにすれば，

$$V_k(y) = \max\{V_{k-1}(y),\ V_{k-1}(y - a_k) + c_k\}$$

が成り立つ (何故か？) ので，この漸化式を

$$\begin{cases} V_0(y) = 0, & y \ge 0 \text{ 場合} \\ V_k(y) = -\infty, & y < 0 \text{ の場合}, \end{cases}$$

として $V_1(1)$ から順に $V_1(2), \ldots, V_1(b)$; $V_2(1), \ldots, V_2(b)$; $\ldots$; $V_n(b)$ と計算していけば，$V_n(b)$ が最適化問題の最適値を与えます．

さて，$V_{k-1}(y)$ が $y = 1, \ldots, b$ に対して計算してあれば，$V_k(y)$ の値は，引き算，足し算，比較，代入の 4 回の基本演算で求めることができ，これを $k = 1, \ldots, n$; $y = 1, \ldots, b$ に対して行うわけですから，その実行時間の上限は $4nb$ となります．使用する計算機の違いなどから，この上限は必ずしも実際の計算時間を厳密に予測するものではありません．しかし，最悪の場合に必要となる計算時間が nb に比例することは概ね検討がつきます．次の概念は，このことを正確に表すのに用いられます:

$$\begin{aligned} O(f(n)) &= \{g(n) \mid \exists c, n_0 > 0,\ \forall n > n_0,\ g(n) \le cf(n)\} \\ \Omega(f(n)) &= \{g(n) \mid \exists c, n_0 > 0,\ \forall n > n_0,\ g(n) \ge cf(n)\} \\ \Theta(f(n)) &= O(f(n)) \cap \Omega(f(n)) \\ o(f(n)) &= \{g(n) \mid c > 0 \Rightarrow \exists n_0 > 0,\ \forall n > n_0,\ g(n) \le cf(n)\} \\ \omega(f(n)) &= \{g(n) \mid c > 0 \Rightarrow \exists n_0 > 0,\ \forall n > n_0,\ g(n) \ge cf(n)\} \end{aligned}$$

普通，例えば $g(n) \in O(f(n))$ である場合，$g(n) = O(f(n))$ と書きます．また，$g(n) = \Theta(f(n))$ のときには，$g(n)$ と $f(n)$ は**同じオーダ**，あるいは**漸近的に同じ**であるといい，$g(n) = o(f(n))$ ならば，$f(n)$ は $g(n)$ よりも**オーダが高い**，あるいは $f(n)$ は $g(n)$ よりも**漸近的に早く増加する**といいます．この概念にしたがえば，問題 0-1 ナップサックを解くのに $4nb$ 回の基本演算を必要とする動的計画法ですが，その実行時間は $O(nb)$ と書かれることになります．

多項式時間アルゴリズム

アルゴリズムの実行時間が問題の入力ビット長の多項式関数で押さえられるとき，そのアルゴリズムを**多項式時間アルゴリズム** (polynomial-time algorithm) といいます．問題 0-1 ナップサックであれば，

$$\log b,\ \ \log z,\ \ \log a_j,\ \ \log c_j,\quad j \in N$$

に各データの区切りを示すキャラクタの数を加えたものが入力ビット長となりますが，$A = \max\{a_j \mid j \in N\}$, $C = \max\{c_j \mid j \in N\}$ と表せば，高々

$$\log b + \log z + n(\log A + \log C)$$

程度の入力ビット長をもつと考えられます．したがって，$\log b$, $\log z$, $\log A$, $\log C$ と n の多項式で実行時間が押さえられるなら，問題 0-1 ナップサックに対するそのアルゴリズムは多項式時間アルゴリズムです．さらに，これらのパラメータのうちでデータの個数を表すもの，この場合は n の多項式実行時間の押さえられるアルゴリズムを**強多項式時間アルゴリズム** (strongly polynomial-time algorithm) といいます．

指数時間アルゴリズム

実行時間が問題の入力ビット長の多項式では押さえられないアルゴリズムを**指数時間アルゴリズム** (exponential-time algorithm) といいます．問題 0-1 ナップサックを解く動的計画法の実行時間 $O(nb)$ は n と b との多項式ではありますが，b を入力ビット長 $\log b$ の関数として明示すると $O(n2^{\log b})$ と書ける点に注意してください．これは動的計画法の実行時間が，問題 0-1 ナップサックの入力ビット長を表す $\log b$, $\log z$, $\log A$, $\log C$ と n の多項式にならないことを意味してい

ます．つまり，動的計画法は指数時間アルゴリズムです．後で説明するように，現段階では問題 0-1 ナップサックに対しては多項式時間アルゴリズムの存在は期待できません．

それでは，最悪計算量解析に基づく 2 種類の実行時間のうち，どちらのクラスがアルゴリズムとして好ましいでしょうか？ それは，100MPS(1 秒間に 100 万回の基本演算が可能) の計算機を想定した次の結果から一目瞭然でしょう:

入力	実行時間					
n	$n \log n$	n^2	n^5	2^n	$n^2 2^n$	$n!$
10	3.32×10^{-7} 秒	1×10^{-6} 秒	0.0015 秒	2.1×10^{-5} 秒	0.001 秒	0.036 秒
20	8.64×10^{-7} 秒	4×10^{-6} 秒	0.032 秒	1.05×10^{-2} 秒	4.19 秒	771 年
30	1.47×10^{-6} 秒	9×10^{-6} 秒	0.243 秒	10 秒	2.68 時間	5.61×10^{6} 宇宙年
40	2.13×10^{-6} 秒	1.6×10^{-5} 秒	1.02 秒	3.05 時間	204 日	1.72×10^{22} 宇宙年
50	6.62×10^{-6} 秒	1×10^{-4} 秒	1.67 分	26798 宇宙年	2.68×10^{8} 宇宙年	1.77×10^{132} 宇宙年

「宇宙年」というのは，ここでの造語で，ビックバンが起きてから現在まで約 150 億年を表します．入力ビット長 n の増加とともにいずれの実行時間も増加はしますが，$n \log n$, n^2, n^5 などの多項式時間の基本演算は増加の度合いが大したものでないことがわかります．これに対して，2^n, $n^2 2^n$, $n!$ などの指数時間の演算が行われると，非常に小さな n のときには多項式時間の演算と遜色ないものの，そののち実行時間は急激に増大し，$n = 50$ 程度の入力に対してすら人類の生存中に答が得られるかどうか怪しくなります．

2.2 P, NP, NP 完全

幸いにして計算幾何問題の多くには多項式時間アルゴリズムの存在が知られています．しかし，数理計画問題などさまざまなアルゴリズムによって解決が求められる問題の中で多項式時間に解決可能な問題はむしろ希な存在であり，多くはその可能性すら絶望視されています．そうした問題の代表が，問題 0-1 ナップサックであり，次に示す問題有向ハミルトン閉路です:

有向ハミルトン閉路

入力: 有向グラフ $G = (V,\ E)$

質問: グラフ G に有向ハミルトン閉路は存在するか？

ハミルトン閉路 (Hamiltonian circuit) とは，グラフ G のすべての頂点 $i \in V$ を 1 回だけ通る閉路のことで，**有向ハミルトン閉路** (directed Hamiltonian circuit) は，そこに含まれる枝の向きがすべて同じであるものを指します．問題有向ハミルトン閉路は，そうした閉路が与えられたグラフ G に含まれているかどうかを尋ねるものです．しかし，この問題や先の問題 0-1 ナップサックには，多項式時間アルゴリズムなど存在しないというのが大方の研究者に共通の見解となっています．その理論的な裏付けとなっているのが **NP 完全** (NP-complete) の概念です．

クラス NP

問題 0-1 ナップサックや有向ハミルトン閉路など，次の特徴をもつ**認識問題** (recognition problem) — “yes” か “no” で答えられる問題 — のクラスを Nondeterministic Polynomial-time solvable の頭文字をとって NP とよびます．

(a) 答の証拠を教えてもらえば，それが問題の条件を満たすかどうかのチェックを多項式時間で行える．

例えば，問題有向ハミルトン閉路に対し，答の証拠として $n = |V|$ 個の頂点の順列 $(i_1,\ i_2,\ \ldots,\ i_n)$ が与えられれば，

$$(i_k,\ i_{k+1}) \in E, \quad k = 1,\ 2,\ \ldots,\ n-1;\ \ (i_n,\ i_1) \in E$$

が満たされるかどうかを確かめればよく，明らかに n と $m = |E|$ の多項式時間で処理できます．したがって，問題有向ハミルトン閉路は NP に属します．

NP 完全

NP 問題を「解く」には，しかし，

(b) 答の候補の数が入力ビット長の指数関数だけあり，それらの証拠を 1 つ 1 つチェックするという単純な方法では，結局，指数時間が必要となる

可能性もあります．

実際，問題有向ハミルトン閉路では，答の候補の証拠として単純に頂点の順列 $(i_1,\ i_2,\ \ldots,\ i_n)$ を列挙すると $n!$ 個にもおよび，何らかの方法で候補を絞り込まないかぎり，(b) のような状況が

起こることは避けられません．どんな方法で候補を絞り込もうとも (b) の可能性を否定しきれないとき，計算の複雑さの理論が用いるロジックは，

> 「この問題 A が多項式時間で解けるくらいなら，あの難問 B だって多項式時間で解ける」(したがって，A はたぶん多項式時間では解けない)

といういささか消極的なものです．これを厳密に定義すると次のようになります．

多項式時間還元 (polynomial-time reduction)： 問題 A を 1 回の基本演算と同じ時間で解くアルゴリズム α があれば，α をサブルーチンとして用いることで問題 B を多項式時間で解くことができるとき，B は A に多項式時間還元可能であるという．

この定義を使えば，(b) の可能性のある NP 問題を次のように分類できます．

NP 完全 (NP-complete)： 問題 A が NP で，すべての NP 問題が A へ多項式時間還元可能なとき，A は NP 完全であるという．

言い換えれば，NP 完全な問題はどんな NP 問題と比べても**それ以上に**難しいということです．ところが，NP 問題は問題 0-1 ナップサックや有向ハミルトン閉路だけでなくて星の数ほど存在するので，いま考えてる問題 A が NP 完全かどうかを，**すべての** NP 問題を A に多項式時間還元することによって確かめることなどできるはずありません．そこで役に立つのが次の定理で，これによって他の NP 完全問題を問題 0-1 ナップサックや有向ハミルトン閉路から芋ヅル式に明らかにすることができます．

定理 2.1. 問題 B を任意の NP 完全問題とするとき，B が問題 A へ多項式時間還元可能ならば，A は NP 完全である．

問題有向ハミルトン閉路はすでに NP 完全であることが知られていますが，ここで試しに定理 2.1 を実際に使って，次の問題に有向ハミルトン閉路が多項式時間で還元できることを示し，この問題もまた NP 完全であることを証明してみましょう．

認識巡回セールスマン

入力: 有向グラフ $G = (V,\ E)$，整数 b，および各枝 $(i,\ j) \in E$ の長さ $c_{ij} \in \mathbb{Z}$

質問: グラフ G の巡回路 (有向ハミルトン閉路)H で $\sum_{(i,j)\in H} c_{ij} \leq b$ を満たすものは存在するか?

多項式時間還元の定義にしたがい，認識巡回セールスマンを解く架空のアルゴリズム α をサブルーチンに用いる有向ハミルトン閉路のアルゴリズム β を作成します:

ステップ 1. 問題有向ハミルトン閉路の入力 $G = (V,\ E)$ を読み込む.

ステップ 2. 入力のグラフ $G = (V,\ E)$ をもとに

$$c_{ij} = \begin{cases} 1, & (i,\ j) \in E \text{ の場合} \\ 2, & (i,\ j) \notin E \text{ の場合}, \end{cases}$$

と定義し，各枝 $(i,\ j) \in E'$ の長さが c_{ij} の完全有向グラフ $G' = (V,\ E')$ をつくる.

ステップ 3. アルゴリズム α に，G', $n = |V|$, c_{ij} を入力し，G' の有向ハミルトン閉路 H の中に $\sum_{(i,j)\in H} c_{ij} \leq n$ を満たすものが存在するかどうかを尋ねる.

ステップ 4. アルゴリズム α が “yes” を返せば “yes”，“no” を返せば “no” を出力し，終了.

有向ハミルトン閉路には頂点の数 n と同じ数の枝だけが含まれますので，このアルゴリズム β のステップ 2 で行われる c_{ij} の定義から直ちに

$$G' \text{ の有向ハミルトン閉路 } H \text{ が } \textstyle\sum_{(i,j)\in H} c_{ij} \leq n \text{ を満たす} \iff H \subseteq G$$

であることがわかります. つまり，認識巡回セールスマンを解くアルゴリズム α さえあれば，それを使って問題有向ハミルトン閉路も解けてしまうわけです.

次にアルゴリズム β の最悪計算量を考えてみましょう.

ステップ 1 の手間: グラフ G の各枝がどの頂点から頂点へ接続しているかさえわかればよいので，$m = |E|$ に比例する時間しか掛かりません.

ステップ 2 の手間: すべての頂点対 i, j に対して枝 $(i,\ j)$ と $(j,\ i)$ を定義し，それぞれに 1 か 2 の長さを与えているにすぎません．つまり，頂点対の総数 $n(n-1)\ /\ 2$ に比例する時間が掛かるだけです．

ステップ 3 の手間: 多項式時間還元の仮定により，アルゴリズム α は 1 回の基本演算と同じ時間しか必要としません．

ステップ 4 の手間: “yes” か “no” を出力するだけなので定数時間で済みます．

以上から，アルゴリズム β は全体で $O(n^2+m)$ の計算量しか必要としないことがわかります．つまり，「問題有向ハミルトン閉路は認識巡回セールスマンへ多項式時間還元可能」であり，定理 2.1 から認識巡回セールスマンは有向ハミルトン閉路と同様，NP 完全問題であることが証明されました．さて，残された問題は認識巡回セールスマンが NP に属するかどうかですが，その答の証拠として巡回路 H が与えられれば，そこに含まれる枝の数 $m=|E|$ に比例する手間で $\sum_{(i,j)\in H} c_{ij} \leq b$ をチェックできることは明らかでしょう．

NP 困難

ところで，上で作成した有向ハミルトン閉路を解くアルゴリズム β のサブルーチンとして，認識巡回セールスマンのアルゴリズム α を呼ぶ代わりに次の最適化巡回セールスマンのアルゴリズム α' を呼べばどうなるでしょうか？

最適化巡回セールスマン

入力: 有向グラフ $G=(V,\ E)$，整数 b，および各枝 $(i,\ j)\in E$ の長さ $c_{ij}\in\mathbb{Z}$

出力: グラフ G の巡回路 H で距離 $\sum_{(i,j)\in H} c_{ij}$ が最小のもの．

アルゴリズム α' の返す巡回路 H は最小の距離をもつので，$\sum_{(i,j)\in H} c_{ij} \leq n$ ならば “yes”，そうでなければ “no” をアルゴリズム β の出力とすれば良さそうです．アルゴリズム α' も単位時間しか必要としないことを仮定すれば，この修正によって β が指数時間アルゴリズムとなることもないので，「有向ハミルトン閉路は最適化巡回セールスマンへ多項式時間還元可能」です．それでは，「最適化巡回セールスマンも NP 完全か？」といえば，答は “no” です．

NP 困難 (NP-hard)**:** すべての NP 問題が A へ多項式時間還元可能なとき，A は NP 困難であるという．

NP 完全の定義と比べると，「問題 A が NP で」という部分が抜け落ちています．最適化巡回セールスマンは，そもそも “yes” や “no” で答えられる認識問題ではなく，したがって NP には属しません．同様に，問題 0-1 ナップサックから多項式時間で還元される最適化問題

$$\left|\begin{array}{ll} \text{最大化} & \displaystyle\sum_{j\in N} c_j x_j \\ \text{条　件} & \displaystyle\sum_{j\in N} a_j x_j \le b \\ & x_j \in \{0,\ 1\},\quad j \in N, \end{array}\right. \tag{2.1}$$

も NP ではありません．しかし，これらの問題は任意の NP 完全問題から多項式時間で還元できるので，どんな NP 問題と比べてもそれ以上に難しい．そこで，最適化巡回セールスマンや (2.1) のような問題は NP 困難と総称されているわけです．

クラス P と「P $\neq$ NP」予想

NP 完全に対し，与えられた証拠のチェックだけでなく，答までも多項式時間で求められる認識問題のクラスを，Polynomial-time solvable の頭文字を取って P とよびます．問題 0-1 ナップサックや有向ハミルトン閉路に多項式時間アルゴリズムが絶望視されている理由は，一般に P $\neq$ NP が信じられているからに過ぎません．NP 完全問題がたった 1 題でも多項式時間で解けることが証明できれば，定理 2.1 から直ちにこの不等号は否定されますが，それに成功した研究者はまだ 1 人もいません．

卵が先か，ニワトリが先か

上に示したとおり，認識巡回セールスマンは有向ハミルトン閉路から多項式時間で還元できるので NP 完全です．それならば，有向ハミルトン閉路の NP 完全性はどうやって証明されたのでしょうか？— それは，認識巡回セールスマンの NP 完全性が認められたので，そこから有向ハミルトン閉路に多項式時間で還元される — ことはもちろんできますが，「卵が先か，ニワトリが先か」の議論に陥ってしまいます．実は，有向ハミルトン閉路は 1972 年に R.M. Karp によって

次の NP 完全問題:

頂点被覆

入力: グラフ $G = (V,\ E)$ と整数 $k \leq |V|$

質問: グラフ G の大きさが k 以下の頂点被覆 — $|V'| \leq k$ で各枝 $(i,\ j) \in E$ に対して頂点 i と j の少なくとも一方を含む頂点集合 $V' \subseteq V$ — は存在するか？

から多項式時間還元されています．それでは，この頂点被覆は？といえば，同じく Karp が

3 充足性

入力: 論理変数 $u_1, \ldots, u_n$ と高々3 個の論理変数の和からなる項 $C_1, \ldots, C_m$

質問: $C_1 \wedge \cdots \wedge C_m = 1$ を満たす $u_1, \ldots, u_n$ は存在するか？

から多項式時間還元されています．それでは，3 充足性は？といえば，S.A. Cook が

充足性

入力: 論理変数 $u_1, \ldots, u_n$ と項 $C_1, \ldots, C_m$

質問: $C_1 \wedge \cdots \wedge C_m = 1$ を満たす $u_1, \ldots, u_n$ は存在するか？

から多項式時間還元されています．それでは，充足性は？といえば，— NP 完全問題の記念すべき題 1 号で，1971 年に Cook は定理 2.1 の助けを借りずに (何しろ，充足性問題へ還元すべき NP 完全問題はそれまで発見されていない！) 次の証明をしています:

定理 2.2. [**Cook の定理**]
任意の NP 問題は充足性に多項式時間還元可能である．

宿題

1 題の NP 完全問題が多項式時間で解ければ,「P = NP」が成り立つことを証明せよ.

ヒント : その 1 題の NP 完全問題を A とすれば，任意の NP 完全問題 B は以下の手順で解くことができる:

ステップ 1. 問題 B の入力の読み込み.

ステップ 2. 以下を繰り返す:

(i) 問題 B の入力，アルゴリズム α の出力をもとに問題 A の入力を生成.

(ii) 問題 A を解くアルゴリズム α の呼び出し .

ステップ 3. “yes”，“no” の出力.

アルゴリズム α の必要とする手間が $O(1)$ ならば，これらのステップに掛かる手間は問題 B の入力ビット長 n の多項式 $f(n)$ で押さえられることに注意 (多項式還元の定義). ステップ 2 (i) における問題 A の入力生成の手間も $f(n)$ で押さえられる. つまり，ステップ 2 (ii) で解かれる A の入力ビット長も $f(n)$ で押さえられなければならない. また，多項式時間還元の定義ではアルゴリズム α の呼び出しが繰り返されることは禁じられていないが，アルゴリズム全体が $f(n)$ で押さえれれているので，ステップ 2 の反復回数も $f(n)$ を上限としなければならない.

NP 完全に関する詳細と宿題の答は以下の文献に解説されているので，わからないときにはこれらを参照のこと:

[1] Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W.H. Reeman, 1979).

[2] Papadimitriou, C.H. and Steiglitz, K.S., *Combinatorial Optimization: Algorithms and Complexity* (Prentice Hall, 1982).