計算量理論と暗号

今回のテーマは計算量理論に関するものです．計算量とは，要するに “問題の難しさ” のことですが，数学で “この定理の証明は難しい” と言うときの “難しさ” とは違い，“やれば必ずいつかできることは分かっているが，面倒だ” という意味の作業量のことです．ただし “いつか” には “数十億年後” も含まれます．

ここで，問題が必ずいつかは解ける，というのはこの問題の答を与える処方，すなわち有限ステップで終わるような**解法のアルゴリズム**が知られているということです．解き方には，良いもの (効率的なもの) と悪いもの (非効率なもの) がありますので，答を得るのにどれくらいの時間がかかるかは，アルゴリズムを指定したときの話です．ただし，問題自身の難しさを言う場合は，現在知られている最良のアルゴリズムでどのくらいかかるかという意味ですが，将来もっと良いアルゴリズムが発見されるかもしれないので，こちらについては “取り敢えず” の主張です．これを恒久的な主張にするには，“この問題にはこれ以上効率の良いアルゴリズムは存在しない” ということの証明が必要ですが，これは一般に超難問で，多くの場合未解決です．

計算量は普通，コンピュータを使って解いたときにかかる時間と解釈され，計算時間とも呼ばれます．しかしコンピュータの速さはまちまちで，将来もっと速いコンピュータができるかもしれないので，計算にかかる実時間には理論的な意味はありません．(実用的には重要ですが.) それでこれを理論化するには，一つの固定した問題ではなく，サイズ n でパラメータ付けされた問題のクラスを対象とし，$n \to \infty$ としたときこれを解くのに必要な計算時間が n のどんな関数で増えてゆくかで判定します．このためには整数論や微分方程式など，純粋数学でもお馴染みのランダウの記号 $O(g(n))$ が使われます．例えば，

例 1．ソートの問題 与えられた n 個の数を大きさの順に並べ替えるのに要する時間は，最良のアルゴリズムで $O(n \log n)$.

例 2．行列式の計算 与えられた n 次正方行列の行列式を計算するのに要する時間は，この行列が特殊な形をしているのでなければ $O(n^3)$.

例 3．因数分解 n 桁の整数の素因子を求めるには，現在最良のアルゴリズムで $O(e^{cn^{1/3}(\log n)^{2/3}})$ の時間がかかる．これは $\forall k$ について $O(n^k)$ (多項式) よりは大きいが，$O(e^{cn})$ (指数函数) よりは小さい．

最後に出てきた “$\exists k$ について $O(n^k)$” が，多項式時間計算量と呼ばれ，話の主役となるものの一つです．このように計算量を漸近理論として捉えることにより，コンピュータの速さの違いは単に定数因子の違いとなり，計算量というものが環境に依らない意味を持って，数学の対象となるのです．

計算量理論は，大きく分けて次の二つの全く正反対の意味で実世界と深く関わっています:

アルゴリズムの効率化 頻繁に解く必要がある問題について，既に有る解法よりはもっと高速なものは無いか? というのが，組合せ論，計算幾何学，グラフ理論の実用的な問題で屡々話題となります．例 1 のソートの問題は，子供にやらせても多分バブルソートと呼ばれる $O(n^2)$ の解法に近いものを自分で見つけ出すでしょう．これを上述の $O(n \log n)$ で解くには，大学で情報系の学科に入ると 1 年生で教えられる有名なソートアルゴリズムが必要になります．線形計画法という工学の重要なツールでは，1980 年代に Karmarkar が新しい高速解法を発見し，アメリカで特許を取ったため熱い議論が起きました．良い解法はお金儲けになるかもしれません．

絶対に速くは計算できないことの保証が欲しい こんなことに実用的な意味が有るのか不思議に思われるかもしれませんが，現在インターネットを支えているセキュリティ基盤は，因数分解が速くはできないことを信じて作られた RSA 暗号という公開鍵暗号に依拠しています．これは信じられているだけで，もし因数分解が多項式時間ではできないことをあなたが証明できたら，有名な懸賞問題の一つを解いたことになり，100 万ドルが手に入ります．逆に，多項式時間の因数分解アルゴリズムを発見してしまったら，世の中は大混乱に陥り，お金が儲かるかどうかは分かりませんが，あなたは間違いなく “時の人” になれるでしょう．

この講義では，計算量のこれら二つの面について，いろんな分野に渡る有名な実例を交えながら，なるべく初等的に，しかし単なるお話よりは内容の有る紹介をしてゆきたいと思います．