計算量の解析

1　はじめに

同一の問題に対する様々なアルゴリズムについて考える場合，各アルゴリズムの性能，つまり，計算 (処理) の速さやメモリ使用量について比較する必要が生じる．この比較には以下の二つの代表的な方法がある．

- ベンチマークテスト (benchmarking)
- 計算量の解析 (complexity analysis)

ベンチマークテストでは，特定のプログラミング言語で書かれた，特定のプログラムを，特定のコンパイラを使っている特定のコンピュータ上で，特定のデータに対して動かし，効率を測定する．これは分野によってはよく使われる方法である．例えば，アルゴリズムの性能ではないが，プロセッサやスーパーコンピュータの処理能力を調べる場合などによく用いられる．しかしこの方法は，特定の計算環境で評価されるため，限定的であり，ゆえに結果の一般性に欠け不十分な場合が多い．例えば，ある計算環境での結果から，他の計算環境での効率に関して予測することは難しい．

より理論的な比較の方法として，数学的な手法を用いたアルゴリズムの計算量解析がある．計算量は「計算の複雑さ」(computational complexity) と呼ばれることもある．以下ではこの方法に関して説明する．

2　解析の方法

以下の数列の和を計算するプログラムを例として説明する．
計算量の解析では，まず以下の各解析を行う．

入力の抽象化 ： 入力のサイズを規定するパラメタを見つけ出す．SUMMATION プログラムでは，入力サイズは列 (*sequences*) の長さ (以下ではこれを n とする) である．

```
function SUMMATION(sequence)
        returns a number
    sum ← 0;
    for i ← 1 to LENGTH(sequence) do begin
        sum ← sum + sequence[i];
    end
    exit (sum);
```

実行時間の尺度 : アルゴリズムの実行時間を反映するような尺度 (measure) を見つけ出す. ただしこの尺度は特定の計算機環境 (コンピュータやコンパイラ) に依存しないよう, アルゴリズムの実装とは独立に決める必要がある. SUMMATION プログラムでは, この尺度として, 例えば**実行された行数**を使えばよい. もっと詳しく, SUMMATION プログラムで実行される加算, 代入, 配列の参照, そして分岐などを数えてもよい.

実行時間の尺度としていずれの方法を用いたとしても, 最終的には, アルゴリズムで実行されたステップ総数を入力のサイズの関数として表したものが得られる. この関数を $T(n)$ とする. ちなみに, 上記の実行された行数を計る方法では, $T(n) = 2n + 2$ となる.

2.1 最悪値および平均値

SUMMATION プログラムは比較的解析が簡単なアルゴリズムであり, 計算量を特徴付けるパラメタ n が容易にみつかったが, アルゴリズムによってはそういったパラメタを見つけることが難しい. それらの場合にやれることは, 最悪値 $T_{worst}(n)$ または平均値 $T_{ave}(n)$ を求めることである. ただし, 平均を求める場合には, 入力に対してある特定の分布を仮定する必要がある.

2.2 漸近的解析 (asymptotic analysis)

また, たいていの実用的なアルゴリズムは正確な解析が難しい. よって, そのような場合は近似という次善手段をとらざるをえない. 例えば, SUMMATION プログラムの実行ステップ数を $O(n)$(“order of n” もしくは「オーダ n」と読む) と評価することにする. これは, 実行ステップ数

が，高々nの定数倍に抑えられるという意味である．厳密な定義は以下である．

Definition 1 (計算オーダ) ある k で，すべての $n \geq n_0$ に対して，$T(n) \leq kf(n)$ となるとき，$T(n)$ は $O(f(n))$ である．

この $O()$ 記法 (オーダ記法) による評価を**漸近的解析**という．この方法によるアルゴリズム解析は，

- 計算環境に依存しない一般性を持つ，
- 厳密な実行回数を大まかに数えられる，
- 実際の入力を大まかにとらえられる，

などの理由から一般的に使われている．

3 問題の難しさ

ここまで述べてきたアルゴリズムの解析と $O()$ 記法により，アルゴリズムの効率や問題の難しさについて議論することができる．これは計算量の解析 (complexity analysis) と呼ばれる分野である．では，問題の難しさにはどのようなレベルがあるのだろうか?以下ではそれについて説明する．

3.1 多項式時間

まず大ざっぱに，問題は多項式時間 (polynomial time) 内に解けるかどうか，つまり計算オーダが n の多項式で表現できるかどうかという分け方がある．多項式時間内に解ける問題のクラスは **P** と呼ばれている．P に入る問題は，一般に「やさしい」(easy) 問題と呼ばれる．しかし P は同時に $O(n^{1000})$ 時間かかる問題も含んでいるので,「やさしい」という言葉を額面通りに受け取らないほうがよい．

3.2 至難性

もう一つ，問題の難しさを表現する概念として至難性 (intractability) と呼ばれるものがある．おおざっぱな言い方をすると，至難な問題とは，解を得るのに必要な計算時間が，その大きさに対して少なくとも指数関数的に増加するものである．多項式的増大に較べて指数関数的増大は，普通のサイズの問題ですら適切な時間では計算できないことを意味している．探索を必要とするような問題は，たいてい至難な問題である．