アルゴリズム論（第3回）

2003.10.20
櫻井　彰人
algorithm@ae.keio.ac.jp
http://www.sakurai.comp.ae.keio.ac.jp/

きょうの講義概要

- **計算可能性と計算量（復習）**
- **計算量の意味**
- **計算量算定の基礎**
- **正当性証明のために**
- **アルゴリズムの種類**

理論的な計算可能性と計算量

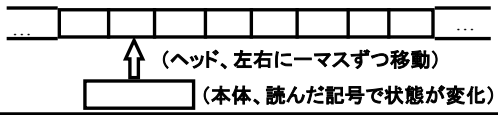
- **個別のハードウェアに依存しない**
  - **原理的にすべてのコンピュータと共通しているもの**
- ◆ 理論的には

チューリングマシンでの計算可能性と計算量

- **厳密な意味での計算可能性(computability)**
  - **チューリング（Turing）マシンで計算可能**
  - **（帰納的関数の存在）**
  - **（λ定義可能）**

2.1 チューリングマシン

- **イギリスのアラン・チューリング（Alan Turing）が１９３６年に考え出したコンピュータの数学モデル**
  - **無限に長いテープ**
  - **テープから記号を一つ読み書きするヘッド**
  - **有限の状態を持つ本体**



チューリングマシンの数学的定義

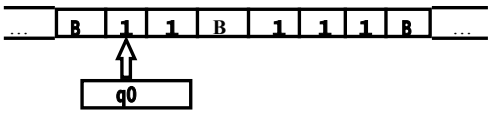
- **Turing Machine (Q, Σ, Γ, L, R, Δ, q_0, H)**
- **Q：本体における状態の有限集合**
- **Σ：テープに書かれる記号の集合**
- **Γ：Σに空白を表わす特別な記号Bなどを要素として加えた集合**
- **L、R：ヘッドの左右への移動**

TMの数学的定義（2）

- **Δ：Q xΣからQ x（Σx {L, R} ）への写像δ**
- **q_0は初期状態を表わすQの要素、Hは最終状態を表わすQの部分集合**
- **結果の表現の仕方(1)**
  - **終了状態＆計算結果をテープ上に表示して**
- **結果の表現の仕方(2)**
  - **初期状態でテープを読み始め、読み終わって最終状態　→　受理（accept）**
  - **最終状態にならない（または、いつまでも読み続ける）　→　拒否（reject）**

演習問題2-1

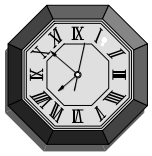
- 記号Bで区切られた二つの負でない整数の加算するチューリングマシンを作りなさい
  - » プログラム可能なコンピュータ

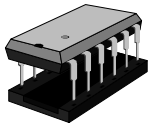


- $a^n b^m$ $(n \geq 0, m \geq 1)$ となる文字列を受理するチューリングマシンを作りなさい

2. 2 計算量(Computational Complexity)

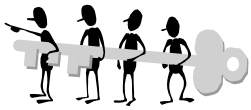
- 計算可能な問題
  - 計算資源（計算時間とメモリ）
- 時間計算量（time complexity）
- 空間計算量（space complexity）





計算量の表示

- 入力の大きさや量に対応した比例式
  - 普通は係数や正確な式は重要でない
    - » 実際の計算機や環境で、変化するから
  - 例：入力がn個のとき … $O(n)$, $O(n^2)$、$O(\log_2 n)$ など
- O
  - オーダー(order)記法



時間計算量の例(探索問題)

- 整数の探索問題
  - n個の整数があるとき、この中に特定のpがあればYESなければNOを返す
- 線形探索(Linear Search)
  - 前から順に１つずつ調べてpが見つかればyes、最後まで調べ尽くしても見つからなければno

例：（n＝10）
整数の組： 5、8、-9、7、-5、-8、3、6、-2、4
特定のp：３、答え：YES

アルゴリズム2.1(線形探索)

- 入力：n 個の整数の組と特定の整数 p
- 出力：整数の組の中に p があれば YES、なければ NO
- 内容
  - （1）整数の組を前から順に p と比較し、等しいものがあれば YES を返して終了
  - （2）整数の組の最後まで p と等しいものがなければ NO を返して終了

線形探索の時間計算量

- 時間計算量 … 比較回数で
- pと等しいものがあるとき
  - 平均 n／２回 の比較回数
- pと等しいものがないとき
  - n回の比較回数

$O(n)$
オーダー n

- オーダー表示では、定数倍は無視
  もう少し注意すべき重要点がある（次回）

2分探索(Binary Search)

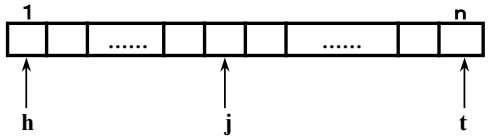
- **線形探索より「効率的」な方法**
  - あらかじめ小さい順に並べ換えておく
  - まず中央にあるものを調べる
    - » p と等しければ yes
    - » p より大きければ左側の中央を
    - » p より小さいときは、右側の中央を
    - » 以上を繰り返し、調べ尽くせば no

⇩

- **２分探索**

アルゴリズム2.2（2分探索）

- **入力：小さい順に並んだ n 個の整数の組と特定の整数 p**
- **出力：整数の組の中に p があれば YES、なければ NO**

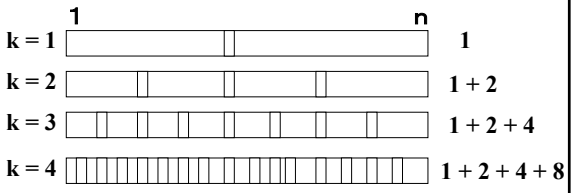


2分探索アルゴリズムの内容

（1） h ← 1; t ← n
（2） h > t なら NO を返して終了
（3） j ← int((h+t)/2)
（4） p = (j番目の数値) なら YES を返して終了
（5） p > (j番目の数値) なら h ← j+1 として（2）へ
（6） p < (j番目の数値) なら t ← j-1 として（2）へ

2分探索アルゴリズムの計算量

- **時間計算量（比較回数）**

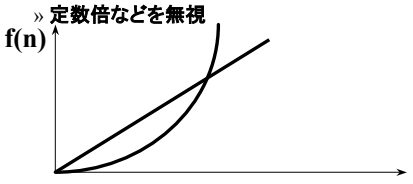


$1 + 2 + 2^2 + \ldots + 2^{k-1} \geq n$ となる最初の k

$2^k - 1 \geq n$ $\quad k = \lceil \log_2 (n+1) \rceil$

2.3 オーダー記法の意味

- **計算量を表わす式**
  - » nが十分に大きくなったときに、漸近的に大小に関わる項だけで
  - » 定数倍などを無視



関数のオーダー

$t(n) = 10n^2 + n + 7$

nが大きい時の主要な項 : $10n^2$

n	$t(n)$	$10n^2$
10	10^3+17	10^3
100	10^5+107	10^5
10^3	10^7+1007	10^7
10^{100}	$10^{201}+10^{100}+7$	10^{201}

$t(n)$ の成長はほぼ $10n^2$

$t(n)$ のオーダーは n^2(定数係数は考慮しない)

定義

$f(n)$: 自然数 $\{0,1,2,...\}$ 上の関数
$f(n)=O(g(n))$ ($f(n)$ のオーダーは高々 $g(n)$)
もしある定数 c が存在して
$|f(n)|\leq c|g(n)|$ が有限個以外の全ての nについて成立する.

$f(n)=\Omega(g(n))$ ($f(n)$ のオーダーは少なくとも $g(n)$)
もしある定数 c が存在して
$|f(n)|\geq c|g(n)|$が有限個以外の全ての nについて成立する.

$f(n)=\Theta(g(n))$ ($f(n)$ のオーダーは $g(n)$)
もし $f(n)=O(g(n))$ かつ $f(n)=\Omega(g(n))$

例

$7n^3+8n+2=O(n^3)$, 何故なら
$7n^3+8n+2\leq 8(n^3+n+1)$
$\leq 8(n^3+n^3+n^3)=24n^3$ ($1\leq n$の時)
$7n^3+8n+2=\Omega(n^3)$, 何故なら
$7n^3+8n+2\geq (n^3+n+1)\geq n^3$ ($0\leq n$の時)
従って
$7n^3+8n+2=\Theta(n^3)$

例

$n^3\neq O(n)$: 何故なら,どんな定数 $c>0$についても,
$n^3-cn=n(n^2-c)>0$ (殆ど全ての nについて).

一般に, $n^i\neq O(n^j)$ ($i>j$の時).

$n=O(2^n)$
数学的帰納法で簡単に証明できる.

定理

次数 k の多項式
$f(n)=a_kn^k+a_{k-1}n^{k-1}+...+a_0$
を考える.但し, $a_k\neq 0$かつ $a_i\geq 0$.
このとき$f(n)=\Theta(n^k)$

証明

$c=a_k+...+a_0$とする. そうすると$n\geq 1$のとき,
$a_kn^k+a_{k-1}n^{k-1}+...+a_0\leq a_kn^k+a_{k-1}n^k+...+a_0n^k$
$\leq (a_k+...+a_0)n^k=cn^k$
従って $f(n)=O(n^k)$.
$f(n)=\Omega(n^k)$は,次のことから明らか.
$a_kn^k+a_{k-1}n^{k-1}+...+a_0\geq a_kn^k$.

以上より, $f(n)=\Theta(n^k)$

例

$\log n=O(n)$

$n\leq 2^n$ であり両辺とも増加関数である故,
両辺の対数をとって,
$\log_2 n\leq n$.

Big-O は推移的

推移的:
$f(n)=O(g(n))$ かつ $g(n)=O(h(n))$ ならば $f(n)=O(h(n))$
$f(n)=O(g(n))$ かつ $g(n)=O(h(n))$ と仮定する
そうすると定数 c と d が存在して
$|f(n)|\le c|g(n)|$ (殆ど全ての n に対して), かつ
$|g(n)|\le d|h(n)|$ (殆ど全ての n に対して).
従って
$|f(n)|\le cd|h(n)|$ (殆ど全ての n に対して).

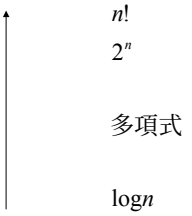
同様に, $\Theta(.)$ も $\Omega(.)$ も推移的.

系

$f(n)$ と $g(n)$ がそれぞれ次数 m と d ($m<d$) の多項式であれば, $g(n)\neq O(f(n))$.

証明: そうでないとする. $g(n)=O(f(n))$.
さて $n^d=O(g(n)), g(n)=O(f(n))$,
かつ $f(n)=O(n^m)$.
推移性から $n^d=O(n^m)$.
$d>m$ であるからこれは矛盾である.

よく見かける関数



$n!$
2^n
多項式
$\log n$

オーダーの上昇順

例

$\log n!=\Theta(n\log n)$

証明:
$$\log n!=\log n+\log(n-1)+\ldots+\log 2+\log 1$$
$$\le \log n+\log n+..+\log n$$
$$\le n\log n,\ (n\ge 1\text{の時}).$$
従って $\log n!=O(n\log n)$.
(次スライドに続く)

一方, $\log n\ge\log(n-1)\ge\ldots\ge\log\lceil n/2\rceil$.
である故
$$\log n!=\log n+\log(n-1)+\cdots+\log 2+\log 1$$
$$\ge\lceil n/2\rceil\log\lceil n/2\rceil$$
$$\ge(n/2)(\log n-1)$$
$$\ge\frac{1}{4}n\log n,\quad (n\ge 2).$$
すなわち, $\log n!=\Omega(n\log n)$.
従って, $\log n!=\Theta(n\log n)$.

例

$n^2+\log n!+3=\Theta(n^2)$
証明
$n^2+\log n!+3\ge n^2$ 故, $n^2+\log n!+3=\Omega(n^2)$.
$\log n!=O(n\log n)$ 故, ある定数 c が存在して
$\log n!\le cn\log n$ (殆ど全ての n について).
$\log n<n$ 故,
$n^2+\log n!+3\le n^2+cn\log n+3\le n^2+cn^2+3\le(c+2)n^2$.
よって, $n^2+\log n!+3=O(n^2)$.
従って, $n^2+\log n!+3=\Theta(n^2)$.

例

1. for $i := 1$ to n do
2. for $j := 1$ to i do
3. $x := x + 1$.

行3が実行される回数 $= 1 + 2 + \cdots + i + \cdots + n$
$= \Theta(n^2)$

例

1. $j := n$
2. while $j \geq 1$ do
3. begin
4. for $i := 1$ to j do
5. $x := x + 1$;
6. $j := \left\lfloor \frac{j}{2} \right\rfloor$;
7. end

$t(n)$ = 行5の実行回数と置く
$t(n) \geq n \quad \therefore t(n) = \Omega(n)$
$t(n) \leq n + \frac{n}{2} + \frac{n}{4} + \cdots + 1$
$\leq n(1 + \frac{1}{2} + \frac{1}{4} + \cdots)$
$\leq 2n$
$\therefore t(n) = O(n)$
$\therefore t(n) = \Theta(n)$

Θ記法を用いる

1. $2n^3 + 67n + 1000$
   $\Theta(n^3)$
2. $(2n+1)^3 + 90$
   $\Theta(n^3)$
3. $(8n-1)^2 \log n + n \log n + 6$
   $\Theta(n^2 \log n)$

老婆心ながら

- Big-O は “オーダーが等しい” ではない
- 例えば、次の式はいずれも正しい
  - $n = O(n^2) = O(n^3) = O(2^n)$

蛇足

- **任意の a, b, c について（→の右側は、有限個を除いた全ての n について）**
  - $a>b>0,\ c>0 \rightarrow n^a > c\, n^b$
  - $a>0,\ b>1,\ c>0 \rightarrow n^a > c \log_b n$
  - $a>1,\ b>0,\ c>0 \rightarrow a^n > c\, n^b$
  - $a>1,\ c>0 \rightarrow n! > c\, a^n$
- **演習：上の関係を証明しなさい。**

通称

オーダー表示	名称
$\Theta(1)$	constant
$\Theta(\log n)$	logarithmic
$\Theta(n)$	linear
$\Theta(n \log n)$	n log n
$\Theta(n^2)$	quadratic
$\Theta(n^3)$	cubic
$\Theta(2^n)$	exponential
$\Theta(n!)$	factorial

増加の速度

n	$\log_2 n$	n^2	n^3	2^n	n!
10	3.3	100	1000	1024	3.6×10^6
100	6.6	10000	10^6	1.27×10^{30}	----
1000	9.9	10^6	10^9	-----	-----
10000	13.2	10^8	10^{12}	-----	-----
100000	16.5	10^{10}	10^{15}	-----	-----
1000000	19.8	10^{12}	10^{18}	-----	-----

- nの多項式オーダー（polynomial order）を超えるnの指数オーダー（exponential order）のものは実質上使えない

2.4 正当性証明のために

- アルゴリズム
  - » 再帰的
  - » 繰り返し（ループ）
- 正当性の証明
  - » 数学的帰納法が中心
    - » 累積帰納法を使うことも

cf. 広瀬健：数学的帰納法、教育出版、1975.

累積帰納法
（course of values induction）

すべての自然数xに対して

P(x)が成立する(C)

（CVIS）任意の自然数kについて、n≦kなるすべての自然数nに対してP(n)が成り立つとすれば、P(k+1)が成り立つ

2分探索の正当性の証明

- P(n)：与えられた整数 n 個。特定の整数 p がこの整数の組の中にあれば YES、なければ NO。
- （CVI）P(1), P(2), …, P(k) が成立していると仮定してP(k+1)を証明
  - » アルゴリズム2.2において
    - » j (= int((h+t)/2))番目の要素と p とを比べて等しければ YES が返るが、この場合に P(k+1) の成立は明らか。
    - » j 番目の要素と p とが等しくない場合は 2.2 の（5）と（6）のステップで対象とする整数の組の要素の個数が k 以下なので、帰納法の仮定から P(k+1) は成立する。
  - » よって、P(n)が成立。

ふつうの数学的帰納法

すべての自然数xに対して

P(x)が成立する(C)

（MI1）P(1)が成り立つ
（MI2）任意の自然数kに対してP(k)が成立すると仮定すればP(k+1)が成り立つ

二つの帰納法の同等性

- ふつうの数学的帰納法で証明できること

- 累積帰納法で証明できること

定理2-2:
ふつうの数学的帰納法で証明できることは、累積帰納法で証明できる。

（証明）普通の数学的帰納法で証明できていると仮定する. 数学的帰納法で必要な仮定 (MI 1)と(MI 2) は、(CVIS)の仮定の一部であり、普通の数学的帰納法の証明手順をそのまま使えば、(C) が得られる

二つの帰納法の同等性（2）

- ふつうの数学的帰納法で証明できること

↑

- 累積帰納法で証明できること

定理2-3:
累積帰納法で証明できることは、ふつうの数学的帰納法で証明できる。

（説明）これは少々やっかい。何しろ、たくさんの仮定を使って証明できるといっているときに、いや～もっと少ない仮定で証明できるよと、言ってあげることなので。

二つの帰納法の同等性（3）

- 補助定理2-1
  - 自然数の全体Nの空でない任意の部分集合には、最小の要素が存在する。
    - » 証明略: これは普通の数学的帰納法で説明される。
- 定理2-3:
  - $\forall n\ P(n)$ の累積帰納法による証明（ (CVIS) ）があったとする。これを書換えて、P(0) の証明と $\forall m\ (P(m) \rightarrow P(m+1))$ の証明を作ることにする
  - P(0) の証明は (CVIS) から 直ちに得られる。
  - $\forall m\ (P(m) \rightarrow P(m+1))$ については次のような証明を書く
    - »（次のスライドに続く）
    - »（次のスライドは、実質的にはもっと簡単になる）

二つの帰納法の同等性（4）

- » P(m) が成立すると仮定する。
- » n<m であって、P(n) が成立しない n があると仮定する
- » そのような n の集合 S は空ではないことになる。補助定理2-1により、最小値が存在する。それを n とする
- » (CVIS) をその対偶を証明する証明に書き直してここにおく
- »（対偶は、$(\exists n \neg P(n)) \rightarrow (\exists k\ k<n$ かつ $\neg P(k))$ ）
- » n が S の最小値であることに反する
- » 従って、「 n<m であって、P(n) が成立しない n 」は存在しない。言換えれば、$\forall n < m\ P(n)$
- »（P(0),…,P(m) が成立するのでそれを仮定した) P(m+1) の証明 (CVIS) をここに書く
- » 証明終

2.5 アルゴリズム各種

- **大まかな分類(網羅的でない)**
  - アルゴリズムの属性/問題解決の属性
  - » 分割統治（divide and conquer）法
    - » より小さな問題の解を組合わせて元の問題の解を得る（2分探索、クイックソートなど）
  - » 貪欲（greedy）法: 最適化問題の近似解法
    - » 複数ステップで最適解を作るとき、各ステップ毎に得る部分最適解を組合わせる
  - » 発見的（heuristic）方法:最適化問題の近似解法
    - » 上記のように、理論的には最適ではないが、経験的によさそうな方法を組合わせる

アルゴリズム各種（2）

- » 確率的（probabilistic）アルゴリズム
  - » 乱数を用いて、確率的に有利なものを探していく
  - » 誤差確率が正確に求まるもの（素数検査法）やそうでないもの
- » 生成検査（generate and test）法: 条件を満足する解
  - » 解の候補をすべて作り出し、確かめる（しらみつぶし）
- » 学習（learning）
  - » 繰り返し実行していると、性能（正解率や速度等）が向上していく
- » その他
  - » 遺伝的アルゴリズム（genetic algorithm）、並列・超並列など

次回から

- **アルゴリズム各論**
  - » 整列（sorting）
  - » 探索（searching）
  - » グラフ（graph）
- **原理、アルゴリズム、計算量、正当性、現実問題への応用**