アルゴリズム論（第2回）

2003.10.6
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

- **アルゴリズムの定義（復習）**
- **計算可能性（チューリングマシン）**
- **計算量（線形探索と２分探索）**

「アルゴリズム」概念の萌芽

- Hilbert の第10問題 (1900)
  - "n 個の未知数を含む整数係数の多項式 $P(x_1,x_2,...,x_n)$ に対し，方程式 $P(x_1,x_2,...,x_n) = 0$（ディオファントス方程式または不定方程式と呼ぶ）が整数解を持つか否かを有限的に判定する方法をみつけよ"
  - Given a diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: *To devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in rational integers.*
  - 実は、この方法(process)は存在しない. それを証明するには、「方法(process)」を厳密に定義しないといけない

「計算」とは（計算のモデル）

- 有限個の（良く定義された）基本的手段がある
  - 四則、比較；文字を書く、写す、消す、進む、
- その手段が忠実に実行できる実行者がいる
- 使用する記号は有限個
- 計算の経過・結果を記す場所がある
- 計算は一歩、一歩行われる
- 何度繰り返しても同一の経過を辿る
- 有限回の実行で終了する

アルゴリズムの定義

- **アルゴリズム =「ある問題のすべての具体例を正しく解くための計算」**
  - **「問題」は「個別・具体的な問題」の集合**
  - **ディオファントス方程式で考えてみよ**
  - **アルゴリズムは「個別・具体的な問題」を入力し、解を出力**
- **「アルゴリズム」定義の簡易版**
  - **明確に定義された個別のステップの集合**
  - **ステップ間の制御のつながりや実行の順序も明確に定義**
  - **正当な入力の範囲が明確（この要請は強すぎる）**
  - **正当な入力に対し有限回ステップの実行で結果を出力**
    - » **計算可能性（Computability）、計算量（Computational Complexity）**
  - **正しい結果を出す**
    - » **正当性（Correctness）**

２　計算可能性と計算量

- **問題**
  - **アルゴリズムで解決可能 … 計算可能**
    - » **解決に必要な計算資源**
      - **時間計算量 … どんな演算を何回**
      - **空間（領域）計算量 … どれくらいの記憶容量を使うか**
  - **アルゴリズムでは解決不可能 … 計算不可能**
- **ユークリッドの互除法の場合（計算可能）**
  - **空間計算量は、M、N、Rのための領域に比例**
  - **時間計算量はループの回数に比例**
    - » **いずれも、それほどでは**

理論的な計算可能性と計算量

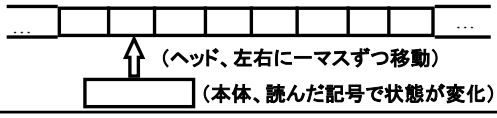
- **個別のハードウェアに依存しない定義**
  - **原理的にすべてのコンピュータに共通**
- ◆ 理論的には

チューリングマシンでの計算可能性と計算量

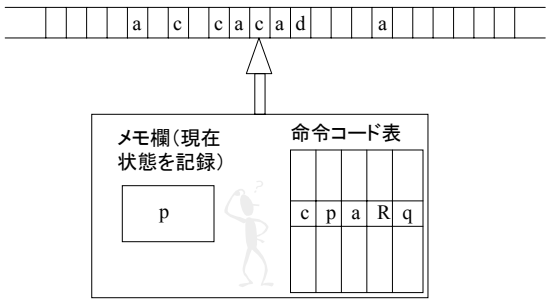
- **厳密な意味での計算可能性(computability)**
  - **チューリング（Turing）マシンで計算可能**
  - **（帰納的関数の存在）**
  - **（λ定義可能）**

2.1 チューリングマシン Turing Machine

- **イギリスのアラン・チューリング（Alan Turing）が1936年に発表した、計算の数学モデル** (“On Computable Numbers, with an Application to the Entscheidungsproblem”)
  - **無限に長いテープ（両端はブランクばかり）**
  - **テープから記号を一つ読み書きするヘッド**
  - **有限個の状態を持ち、明確な状態遷移をする本体**



概念図



自然言語による説明

◆ 理論上、無限の長さの紙とプロセッサで構成される単純なコンピュータの設計思想。プロセッサはその紙の上を滑って回り、紙上のどのシンボルを現在読みとっているか、プロセッサに可能な複数の状態のうちのどれにあるか、ということに応じて、紙上のシンボルを消したり、新しいシンボルをプリントしたりする。手間がかかりすぎて実用にはならないが、過去、現在、未来のあらゆるデジタルコンピュータに演算可能なことであれば、すべて、TMにも演算できると考えられている
（Pinker, 言葉を生み出す本能（下））

チューリングマシンの数学的定義

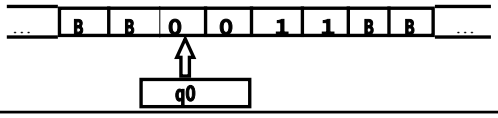
- TM(Q, Σ, Γ, L, R, Δ, q_0, H)
- **Q：本体における状態の有限集合**
- **Σ：テープに書かれる記号の集合**
- **Γ：Σに空白を表わす特別な記号 B などを要素として加えた集合**
- **L、R：ヘッドの左右への移動**
- **Δ：Q x Σ から Q x（Σ x {L, R} ）への写像δの集合**
- **q_0 は初期状態を表わすQの要素、H は最終状態を表わすQの部分集合**

TMの数学的定義（2）

- **初期状態でテープを読み始め、（しばらくして）最終状態になる → 計算終了**
- **最終状態にならない → 発散**

チューリングマシンの例（受理・拒否）

- テープからの入力が 0^n1^n の形のときのみ受理、それ以外のとき拒否（n≧１）
  - 受理(accept)、拒否(reject)：相異なる終了状態
  - 受理：終了状態、拒否：状態遷移が未定義
- Q ＝ {q0, q1, q2, q3, q4, q5}
- Σ ＝ {0, 1} 、H ＝ {q5}
- Γ ＝ {0, 1, B, X, Y}

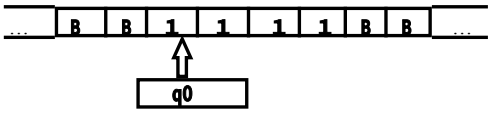


TMの例（つづき）

- {δ(q0,0) = (q1,X,R), δ(q1,0) = (q1,0,R), δ(q1,Y) = (q1,Y,R), δ(q1,1) = (q2,Y,L), δ(q2,Y) = (q2,Y,L), δ(q2,X) = (q3,X,R), δ(q2,0) = (q4,0,L), δ(q4,0) = (q4,0,L), δ(q4,X) = (q0,X,R), δ(q3,Y) = (q3,Y,R), δ(q3,B) = (q5,Y,R)}
- たとえば、0011が受理されて、011が受理されないことを確かめる

チューリングマシンの例（計算）

- 負でない整数m
  - 1をm+1個つなげて表現
    - » 1^{m+1}
- m+1を計算するチューリングマシン



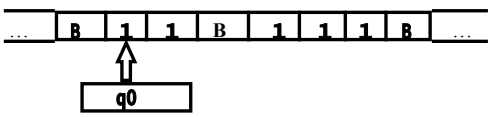
- この例では 3→4
  - 1^4 から 1^5 に

1を加えるチューリングマシン

- Q ＝ {q0, q1}
- Σ ＝ {1} 、H ＝ {q1}
- Γ ＝ {1, B}
- {δ(q0,1) = (q0,1,R), δ(q0,B) = (q1,1,R) }
  - » 1を読んでいる限り右に移動し、Bになったらそれを1に変えて止まる

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を計算するチューリングマシンを作りなさい



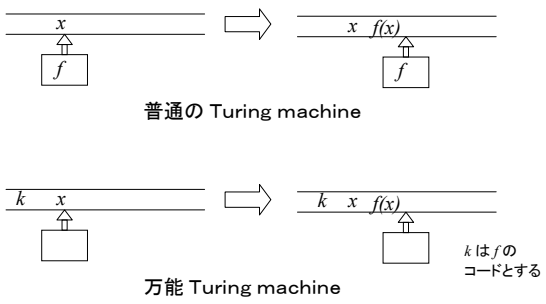
- a^nb^m (n≧0, m≧1) となる文字列を受理するチューリングマシンを作りなさい

補足：チューリングについて

- イギリス人の天才チューリングについては
  - 伊藤智義（作）、久保田眞二（画）『[ブレインズ]コンピュータに賭けた男たち』第１巻、集英社、1996年
  - 第二次世界大戦中はドイツ軍の暗号破り
  - 暗号解読のためのコンピュータ「コロッサス」を設計
  - 「チューリングマシン」の発明
  - 人工知能における「チューリングのテスト」の提唱
  - 米国計算機学会のチューリング賞

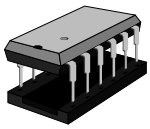
Turing Machine と現実の計算機

- 現実の計算機は
  - 主記憶はアドレスを用いて参照
  - レジスタという特殊なメモリがある
  - スタックがある、機能の高い命令がある、、、、
- TM を用いて模擬できる
  - ランダムアクセスマシンというのもある
  - 「計算可能性」ということに関しては同一
  - プログラム作成という観点からは大きな差
- TM は自分を模擬できる
  - 万能チューリングマシン
- 現実の計算機は自分を模擬できる
  - シミュレータ
  - メタな操作ができる、という意味なら、コンパイラも



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）**

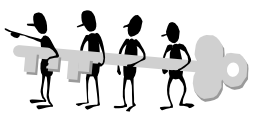


計算量の考え方

- **特定の入力に対する計算資源**
  - **時間計算量**
    - » どんな種類の計算を何回実行すれば答えが出るか（計算時間）
      - アルゴリズムであるからには、有限回であることは自明
  - **空間計算量**
    - » 答えを出す過程で、どのくらいの記憶容量を使うか
- これらの量に応じて
  - 現実的なアルゴリズム
  - 非現実的なアルゴリズム

計算量の表示

- **代表的な入力の大きさや量の関数**
  - **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）

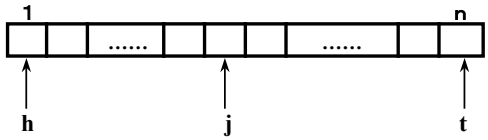
2分探索(Binary Search)

- 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次に調べる
    - » p より小さいときは、右側の中央を次に調べる
    - » 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

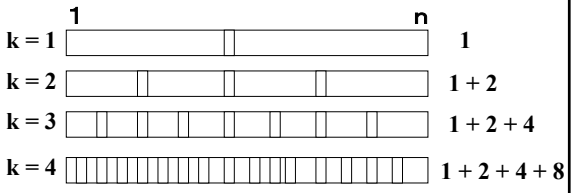


2分探索アルゴリズムの内容

（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

$2^k - 1 \geq n$