ICT リテラシー (情報技術論) A

-- 第 13 回 : アルゴリズム --

栗野 俊一

伝言

私語は慎むように !!

- 席は自由です
  - できるだけ前に詰めよう
  - コロナ対策のために、ソーシャルディスタンスをたもとう
- 色々なお知らせについて
  - 栗野の Web Page に注意する事

    http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino
  - google で「kurino」で検索

ICT リテラシー (情報技術論) A

今後の予定

今後の予定

- 今後の予定(後ろから)
  - 15 回目：試験を行う (試験は年明けになる)
    - オンライン試験を予定している(自宅から受ける)
    - 都合がわるい場合は連絡をすれば別の日時に行う(詳しくは次週説明)
    - 栗野もオンラインで質問対応で待機するが教室には来ない予定
  - 14 回目(次回)：残りの内容をできるだけ..
    - 試験に関する説明行う
  - 13 回目(今回)：アルゴリズム

ICT リテラシー (情報技術論) A

アルゴリズム

アルゴリズム

- □ アルゴリズム とは
  - ○ ある「問題」を解く アルゴリズム の定義
    - ▹ 確定性：明確な手順の有限な列で表現されている
    - ▹ 正当性：その「問題」を解く(解を求める)事ができる
    - ▹ 停止性：有限時間で終了する
  - ○ 問題に対して、アルゴリズムが与えられれば、
    - ▹ アルゴリズム(の示す手順を適用する事)により、答を得る事ができる
    - ▹ 例: 公式(計算の手順が与えられる)、ユークリッドの互除法(最大公約数の求め方)、筆算、10 進 2 進変換
  - ○ 数学的に問題を「解く」事
    - ▹ 問題の「答えを得る」事ではなく、「答えを得る手段(アルゴリズム)を得る」事
    - ▹ アルゴリズムがあれば、(答えを知らない)問題の答えが得られる
- □ 知識の構造
  - ○ 問題は What : 問題の答となるものを定義
  - ○ アルゴリズムは How to: 問題の答となるもの求める手段
- □ コンピュータに答を求めさせるには、アルゴリズムが必要
  - ○ アルゴリズムが明確でない問題をコンピュータにやらせるのは難しい..
  - ○ Deep Learning は、「(ある種の拡張された)アルゴリズムを求める」アルゴリズム
    - ▹ Deep Learning は正当性(問題定義)の点で、課題(正確性)を抱えている

ICT リテラシー (情報技術論) A

プログラム

プログラム

- プログラム (Text p.77 6.2 節)
  - プログラム とは
    - 定義：アルゴリズムをコンピュータに扱える(実行できる)ように表現した物
    - プログラムをコンピュータに与えると、アルゴリズムを実行してくれる
  - ソフトウェア
    - 特定なシステム(CPU/OS上)で、実行可能なプログラム

プログラミング言語

- プログラミング言語とは (Text p.77 6.2.1 節)
  - プログラムを記述するために作られた人工的な言語
    - ハード(電気回路)でも、アルゴリズムが表現可能
    - 柔軟性を高めるために、ソフト(プログラム)で、機能を追加
- プログラミング言語の分類
  - 手続型：何(What)を、どうするか(How to) という処理手順を記述する
    - 例 : C++, Java, Python, etc..
    - 特徴 : CPU の命令に対応する指示を直接指定できるので、効率が良い
  - 非手続型：手続型以外のプログラミング言語
    - 特徴 : 手順の記述が不要なので、プログラム書き易いが、(手順がないので)非効率な事が多い
    - 関数型 : 問題の解を求める関数の定義を行う(例 : Lisp, ML)
    - 論理型 : 問題の解が満す条件を指定し、解を求めさせる (例 : Prolog, SQL)

色々なプログラミング言語

- 低級言語：CPU 依存する
  - 機械語：CPU への命令の並び (CPU 毎に異る / 2 進表現)
    - アセンブリ言語：機械語の命令とほぼ 1 対 1 で表現可能 (文字列表現)
- 高級言語：CPU と独立 (言語処理系 [翻訳/通訳 を行うプログラム] が必要)
  - 手続型
    - Fortan：最初の高級言語 (科学技術計算に利用) / Basic：Fortan 教育用簡易版
    - Cobol：商業計算用
    - Algol (Pascal)：アルゴリズム記述用 (データ構造)
    - C 言語：Unix OS を記述されるために、設計 (free な Unix と一緒に広く利用される)
    - C++：Object 指向型 / Java：仮想 CPU の実装 (OS から独立) / Python：ライブラリが多く、Deep Learning で利用
    - Javascript：Java とは名前が似て居るが、別物 (HTML と併用される)
  - 関数型 Lisp：シンボル処理言語 (人工知能のアセンブラ / ラムダカリキュラス)
  - 論理型 Prolog：論理プログラミング言語 / SQL：関係データベースを操作する言語 (DB 専用言語)
- マークアップ言語：プログラミング言語ではなく、Content を記述する言語
  - HTML：Web Page 記述 (javascript と組み合せる事により、機能を持つページが作

プログラムの内部動作

- プログラムの内部動作 (Text p.78 6.2.2 節)
  - CPU が理解できるプログラムは機械語のみ
    - メモリ上に記録されている命令も機械語の命令 (2 進数)
    - チューリングマイシと同じ(ノイマン型)なので、命令とデータの区別がない
    - CPU にとっては都合がよいが、人間には分かり難い (低級言語)
- 言語処理系
  - (機械語以外の) 言語で記述されたプログラムを実行できるようにするプログラム
    - コンパイラ(翻訳系) : 機械語に変換 (翻訳) する / 一度に変換 / 変換がおわれば不要
    - インタープリター(通訳) : 機械語に変換 (通訳) する / 逐次変換 / いつでも必要

高級言語の基本処理

- 高級言語の基本処理 (Text p.79 6.2.3 節)
  - 手続型言語(C++,Java,Python) によるプログラム記述
    - 変数操作(入力,出力,代入,参照)を命令に基本として、その手順を記述
    - 操作手順の記述(プログラム)がアルゴリズムを表現する
- 基本操作
  - 変数：名前がついた記憶領域(データを記録できる)
    - 名前を指定して、その値を取り出す(参照)や、値を記録させる(代入)が可能
  - 代入：変数に値を記録するように指示する事 (値は計算された結果になる)
  - 参照：変数名を指定して、その変数に記録されている値を取り出す事 (値は何度、参照しても変らない)
  - 式 (四則計算)：代入する値を「計算式」の形で表現すると、その「式」の値が計算される
    - 式で利用可能な演算子は言語によって異るが、四則は (+, -, *, /) が共通で利用できる
- 制御構造：命令の実行の可否や、繰返し等を指示する構文
  - if 文(条件判断)：条件によって、指定した命令を実行したりしなかったりを表現する
  - for 文(繰返し)：条件によって、指定した命令を繰り返すか止めるかを表現する

ICT リテラシー (情報技術論) A

データベース

データベース

- □ データベース (Text p.81 6.3 節)
  - ○ データベース とは : 構造化した情報またはデータの組織的な集合(what)
    - ▹ 大量のデータを保存,管理でき、データの検索,書き換えが容易に行えるもの(利用目的)
    - ▹ 例: (小規模)学籍簿、住所録、(大規模)銀行のオンラインシステム、戸籍
  - ○ データが単に集っているだけではだめ
    - ▹ ビッグデータ (売上情報) : 構造化されていないとデータベースと言えない
    - ▹ 表紙が破れていたり、向きも順も適当に乱雑に本が詰めてある本棚は役に立たない
    - ▹ 整理され、書名順、著者順等に並べられた本棚は有用 (図書館学)
  - ○ データベース に必要な三つの要素
    - ▹ 検索速度 : 欲しいデータが短い時間で、データベースの中から探せる (構造化、索引)
    - ▹ データ量 : 大量のデータが扱える (メモリに入らない量も扱える)
    - ▹ 完備性 : データの一貫性や整合性が保たれている (一部を削除した場合、関連した項目も一緒に削除)

データベースの表現法

- □ データベースの表現法 (Text p.82 6.3.2 節)
  - ○ レコード(記録) : データベース内に記録されているデータの単位
    - ▷ データベースの表現方法 : レコード間の「関係」の表現方法
  - ○ 階層的表現 (木構造)
    - ▷ データを親,子,孫のような階層構造に並べて木の形に表現
    - ▷ 最上位の親からたどることによって検索
    - ▷ cf. 分類、会社組織、住所、ファイルシステム、ドメイン名
    - ▷ 木構造の概念 : 親子[上下]関係, 根(root), 葉, 枝, 子孫, 祖先, 兄弟
    - ▷ 木構造の得失 : 経路が一通り(高速/効率が良い)/全ての状態が表現できない(兼任問題)
  - ○ 網的表現 (ネットワーク構造)
    - ▷ データをノード(普通のレコード)、データ間の二項関係をアーク(関係レコード)としたネットワークで表現
    - ▷ 特定なデータから、関係を手繰る事によって、他のデータを探す(連想ゲーム)
    - ▷ cf. 人間の記憶構造、(AI の)専門化知識、知り合いの関係、WWW, 実体関連モデル
    - ▷ 網構造の得失 : 任意の状態を表現/必要な情報のみ記録/構造がデータに依存(数学的な構造がない)
  - ○ 関係的表現 (表構造 : リレーショナルデータベース)
    - ▷ データ間の(n項)関係を表で表し、表の集合で表現
    - ▷ 表を操作する事により、目的のデータを含む表を作成する
    - ▷ cf. Excel シート、時刻表
    - ▷ 表構造の得失 : 汎用的な表操作だけ/効率が悪い(汎用的過ぎる)

関係的表現のデータ操作

- 関係的表現のデータ操作 (Text p.84, 6.3.3 節)
  - 関係的表現の操作：数学的な集合操作に対応 (数学的な基礎がある)
    - エドガー=コッド「関係的表現の数学的基礎」(1970)
  - 集合演算
    - 合併：和集合を求める(OR)
    - 共通部分：共通集合を求める(AND)
    - 差 (引く)：差集合を求める
    - 直積：直積集合を求める
  - 表操作
    - 射影：表のフィールド名(項目)が与えられたとき、項目に該当するリストを抜き出す操作
    - 選択：条件が与えられたときに、条件に合致するレコードを抜き出す操作
    - 結合：複数の表を結び付ける (自然な結合:重複したレコードを削除)
- SQL (Structured Query Language: 構造化問い合せ言語)
  - 関係的表現のデータベース(RDB)を操作するための言語

ICT リテラシー (情報技術論) A

おしまい