算法设计与分析—入门篇

第一讲 算法概述

哈尔滨工业大学
王宏志
wangzh@hit.edu.cn
http://homepage.hit.edu.cn/pages/wang/

本讲内容

1.1 什么是算法？

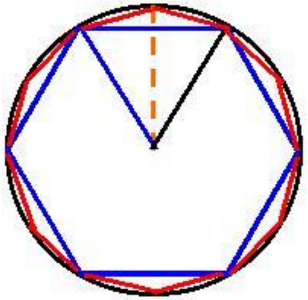
1.2 计算机科学中算法的位置

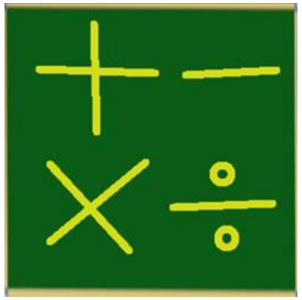
1.3 算法分析引论

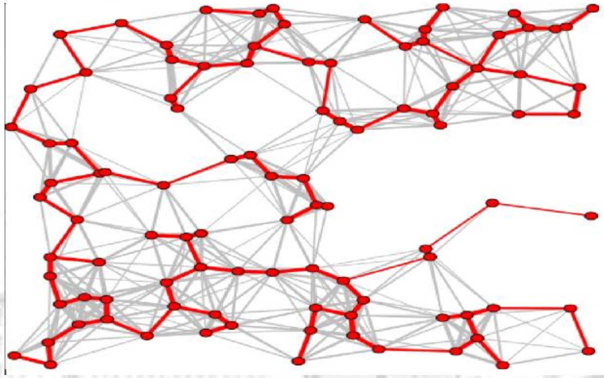
1.4 算法设计引论

什么是算法？

- 在数学和计算机科学之中，算法/算则法（Algorithm）为一个计算的具体步骤，常用于计算、数据处理和自动推理。(Wikipedia)
- 算法的例子
  - 刘徽割圆术
  - 四则运算
  - 最小生成树
  - 快速排序







计算的定义

- 可由一个给定计算模型机械地执行的规则或计算步骤序列称为该计算模型的一个计算
- 注意
  - 一个计算机程序是一个计算（计算模型是计算机）
  - 计算可能永远不停止——不是算法。

算法的定义

算法是一个满足下列条件的计算：

- 有穷性/终止性：有限步内必须停止，
- 确定性：每一步都是严格定义和确定的动作，
- 能行性：每一个动作都能够被精确地机械执行，
- 输入：有一个满足给定约束条件的输入，
- 输出：满足给定约束条件的结果。

关于算法

- “算法”的来源
  - 中文名称：周髀算经
  - 英文名称
    - “Algorithm”来自于9世纪波斯数学家花拉子米（al-Khwarizmi）
    - “算法”原为“algorism”，即“al-Khwarizmi”的音转，意思是“花拉子米”的运算法则
    - 在18世纪演变为“algorithm”
- 最早的算法
    - 欧几里德的“求最大公因子算法”

问题的定义

- 算法的目的是求解问题。什么是问题?
- 问题
  - 设Input和Output是两个集合。一个问题是一个关系$R \subseteq Input \times Output$，Input称为问题R的输入集合，Input的每个元素称为R的一个输入，Output称为问题R的输出或结果集合，Output的每个元素称为R的一个结果。
  - 注意
    - 问题定义了输入和输出的关系。

问题的例子

SORT问题定义如下：

- 输入集合Input={$\langle a_1, \ldots, a_n \rangle \mid a_i$是整数}
- 输出集合Output={$\langle b_1, \ldots, b_n \rangle \mid b_i$是整数，$b_1 \le \ldots \le b_n$}
- 问题SORT={$(\langle a_1, \ldots, a_n \rangle, \langle b_1, \ldots, b_n \rangle) \mid \langle a_1, \ldots, a_n \rangle \in$ Input, $\langle b_1, \ldots, b_n \rangle \in$ Output, $\{a_1, \ldots, a_n\} = \{b_1, \ldots, b_n\}$}

- 问题实例
  - 问题P的一个实例是P的一个二元组。
  - 注意
    - 一个算法面向一个问题，而不是仅求解一个问题的一个或几个实例。

算法示例

- 问题定义
  - $Input=\{\langle a_1, \ldots, a_n\rangle \mid a_i$是整数$\}$
  - $output=\{\langle b_1, \ldots, b_n\rangle \mid b_i$是整数, 且$b_1 \leq \ldots \leq b_n\}$
  - $R=\{(\langle a_1, \ldots, a_n\rangle, \langle b_1, \ldots, b_n\rangle) \mid \langle a_1, \ldots, a_n\rangle \in Input, \langle b_1, \ldots, b_n\rangle \in output, \{a_1, \ldots, a_n\}=\{b_1, \ldots, b_n\}\}$
- 算法的思想—扑克牌游戏

算法演示

A[1, ... , n] = 5, 2, 4, 6, 1, 3

A[1, ... , n] = 5, 2, 4, 6, 1, 3

A[1, ... , n] = 2, 5, 4, 6, 1, 3

A[1, ... , n] = 2, 4, 5, 6, 1, 3

A[1, ... , n] = 2, 4, 5, 6, 1, 3

A[1, ... , n] = 1, 2, 4, 5, 6, 3

A[1, ... , n] = 1, 2, 3, 4, 5, 6

算法描述

```
Insertion-sort(A)
Input:  A[1,......,n]=n个数
output: A[1,......,n]=n个sorted数
FOR j=2  To n  Do
    key←A[j];
    i←j-1
    WHILE i>0 AND A[i]>key  Do
        A[i+1]←A[i];
        i←i-1;
    A[i+1]←key;
```

- 实例：A[1,......,n]=5, 2, 4, 6, 1, 3

本讲内容

1.1 什么是算法？

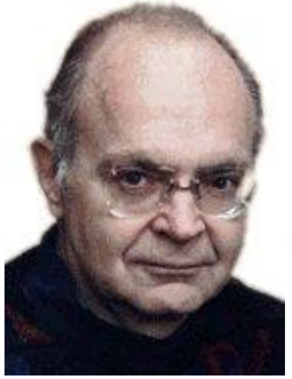
1.2 计算机科学中算法的位置

1.3 算法分析引论

1.4 算法设计引论

算法是计算机科学基础的重要主题

- 70年代前
  - 计算机科学基础的主题没有被清楚地认清。
- 70年代
  - Knuth出版了《The Art of Computer Programming》
  - 以算法研究为主线
  - 确立了算法为计算机科学基础的重要主题
  - 1974年获得图灵奖。



- 70年代后
  - 算法作为计算机科学核心推动了计算机科学技术飞速发展

计算机科学的体系

- **解决一个计算问题的过程**

可计算否

↓

能行可计算否

↓

算法设计与分析

↓

用计算机语言实现算法

↓

软件系统

- 可计算理论
  - 计算模型
  - 可计算问题/不可计算问题
  - 计算模型的等价性--图灵/Church命题
- 计算复杂性理论
  - 在给定的计算模型下研究问题的复杂性
    - 固有复杂性
    - 上界
    - 下界
    - 平均
    - 复杂性问题的分类：P=NP?
    - 抽象复杂性研究

- 算法设计和分析
  - 可计算问题的算法的设计与分析
  - 设计算法的理论、方法和技术
  - 分析算法的理论、方法和技术
- 计算机软件
  - 系统软件
  - 工具软件
  - 应用软件

本讲内容

1.1 什么是算法？
1.2 计算机科学中算法的位置
1.3 算法分析引论
1.4 算法设计引论

算法的正确性分析

- 算法正确性
  - 一个算法是正确的，如果它对于每一个输入都最终停止，而且产生正确的输出。

  - 不正确算法：

    ①不停止(在某个输入上)

    ②对所有输入都停止，但对某输入产生不正确结果
  - 近似算法 근사알고리즘 ：정확한 해를 구하는 「빠른 산법」을 찾아내지 못하든가，또는 존재하지 않을 때 정확한 해를 구하는 대신 근사해를 구하기로 하고，그 대신 충분히 빨리 계산할 수 있는 것으로 한다.
이러한 산법을 근사 산법이라고 한다.
예를 들면，2의 제곱근을 구하는 데 X2＝2를 뉴턴법으로 푸는 것이 이에 해당한다

    ①对所有输入都停止

    ②产生近似正确的解或产生不多的不正确解

- 算法正确性证明
  - 证明算法对所有输入都停止
  - 证明对每个输入都产生正确结果

  - 调试程序≠程序正确性证明：

    程序调试只能证明程序有错，

    不能证明程序无错误！

插入排序的正确性

- 循环不变量
  - 在每次循环的开始，子数组A[1..j-1] 包含原来数组中A[1..j-1] 但是已经有序
- 证明
  - 初始化 : j=2, A[1..j-1]=A[1..1]=A[1], 已经有序.
  - 维护：每一层循环维护循环不变量.
  - 终止: j=n+1, so A[1..j-1]=A[1..n] 有序.

算法的复杂性分析

- 目的：
  - 预测算法对不同输入所需资源量

    yu ce : 예측하다.
- 复杂性测度：
  - 时间，空间， I/O等，是输入大小的函数
- 用途： 용도
  - 为求解一个问题选择最佳算法、最佳设备

    jia : 좋다 she bei : 설비
- 需要的数学基础
  - 离散数学，组合数学，概率论，代数等

    조합론
- 需要的数学能力
  - 建立算法复杂性的数学模型
  - 数学模型化简

算法复杂性分析的度量

- 输入的大小
  - 设Input是问题R的输入集合，R的输入大小是一个函数 F：Input→N，N是正整数集合。

示例：

- 矩阵问题的输入大小=矩阵的维数 차수

wei 차원

- 图论问题的输入大小=图的边数/结点数

算法复杂性分析的度量

- 时间复杂性
  - 一个算法对特定输入的时间复杂性是该算法对该输入产生结果需要的原子操作或“步”数
  - 注意
    - 时间复杂性是输入大小的函数
    - 我们假设每一步的执行需要常数时间，实际上每步需要的时间量可能不同。

算法复杂性分析的度量

- 空间复杂性
  - 一个算法对特定输入的空间复杂性是该算法对该输入产生结果所需要的存储空间大小。

算法复杂性分析的度量

- 最坏复杂性
  - 设Input是问题R的输入集合，Complexity(X)是求解R的算法A的复杂性函数，Size(y)是确定R中输入大小的函数，A的最坏复杂性是

$$Max\{Complexity(size(y)) \mid y \in Input\}$$

- 最小复杂性

$$Min\{Complexity(size(y)) \mid y \in Input\}$$

- 平均复杂性
  - 设y∈Input, y作为算法A的输入出现的概率是p_y，A的平均复杂性为$\sum_{y \in Input} p_y \times complexity(size(y))$

算法分析的模型

- 随机访问模型 (*Random-Access-Model ,RAM*)
  - 单处理机，串行执行，无并发
  - 基本数据类型
  - 基本操作(每个操作常数时间)
- 并行多处理机模型(*PRAM*)

插入排序的分析

INSERTION-SORT(A)	代价	次数
for j = 2 to length[A]	c_1	n
do $key \leftarrow$ A[j]	c_2	n-1
//insert A[j] to sorted sequence A[1..j-1]	0	n-1
$i \leftarrow j$-1	c_4	n-1
while i >0 and A[i]>key	c_5	$\sum_{j=2}^{n} t_j$
do A[i+1] $\leftarrow$ A[i]	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$i \leftarrow i$-1	c_7	$\sum_{j=2}^{n} (t_j - 1)$
A[i+1] $\leftarrow key$	c_8	$n - 1$

(t_j 是对j来说循环中执行的次数)

总时间代价$T(n)$ = 代价的和 × 每行执行次数

$$= c_1 n + c_2(n-1) + c_4(n-1) + c_5\sum_{j=2}^{n} t_j + c_6\sum_{j=2}^{n}(t_j-1) + c_7\sum_{j=2}^{n}(t_j-1) + c_8(n-1)$$

插入排序的分析(续)

- **最好代价**: 有序的数组
  - t_j=1, 且6和7行执行0次
  - $T(n) = c_1 n + c_2(n\text{-}1) + c_4(n\text{-}1) + c_5(n\text{-}1) + c_8(n\text{-}1)$

    $=(c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) = cn + c'$ 0 (n)
- **最坏代价**: 逆序数组
  - $t_j=j$,
  - $\sum_{j=2}^{n} t_j = \sum_{j=2}^{n} j = n(n+1)/2\text{-}1$, 且$\sum_{j=2}^{n}(t_j-1) = \sum_{j=2}^{n}(j-1) = n(n\text{-}1)/2$
  - $T(n) = c_1 n + c_2(n\text{-}1) + c_4(n\text{-}1) + c_5(n(n+1)/2 \text{-}1) + c_6(n(n\text{-}1)/2 \text{-}1) + c_7(n(n\text{-}1)/2) + c_8(n\text{-}1) = ((c_5 + c_6 + c_7)/2)n_2 + (c_1 + c_2 + c_4 + c_5/2\text{-}c_6/2\text{-}c_7/2+c_8)\text{n-}(c_2 + c_4 + c_5 + c_8) = an^2+bn+c$ 0(n^2)
- **平均代价**: 随机数
  - 平均来看, $t_j = j/2$. $T(n)$ 与n^2同阶，和最坏情况相同.

本讲内容

1.1 什么是算法?
1.2 计算机科学中算法的位置
1.3 算法分析引论
1.4 算法设计引论

算法设计模式

- 暴力搜索
- 分治法
- 图搜索与枚举
  – 分支界限
  – 回溯 hui su : 되돌아보다
- 随机化方法

算法实现方法

- 递归与迭代
- 顺序、并行与分布式
- 确定性与非确定性
- 近似求解与精确求解
- 量子算法

最优化算法设计方法

- 线性规划 주어진 계약조건 하에서 두 개 이상의 독립변수와 종속변수와의 관계를 표시하여 주는 이론의 간단한 표현이다. 선형계획기법은 제한된 자원을 어떻게 생산적 용도에 맞게 합리적으로 배분 할 것인가를 결정하는 수리적 기법의 하나이며, 판매량 수익, 자원의 사용 또는 시간을 어떻게 최적화 할 수 있는가에 관해 광범위하게 이용되는 계량적 계획기법이다. 이 기법은 1차 함수로 표시되는 목적 함수와 선형부등식으로 표현되는 제약조건을 갖고 자원할당문제의 최적화를 구하는 것이다.
- 动态规划 - 어떤 문제가 반복적이며 최적 하위구조로 이뤄져 있을 때, 하위구조에 있는 부분 문제의 답을 기반으로 전체의 답을 구하는 방법

  - 최적 하위구조(Optimal Structure) : 전체 문제의 답이 부분 문제들의 답으로 만들어지는 구조.
  - 분할정복과 비슷해보이지만, 동적계획법은 작은부분부터 큰부분으로 올라간다.(Bottom - up)
  - 추가적으로, 동적계획법에서는 이전 단계의 답에 의존적이다
- 贪心法 greedy
- 启发式方法

교사의 설명을 최소한도로 줄이고 교사의 지시 없이도 학생들이 독립적으로 학습목표를 달성할 수 있도록 하는 수업의 형태