HIT
CS&E

第一章

引　　言

张炜
计算机科学与技术学院

HIT
CS&E

1.1 Role of Algorithms in Computer Science

- 算法是计算机科学的主题
- 计算机科学体系与算法设计与分析的地位
- 算法设计与分析的意义

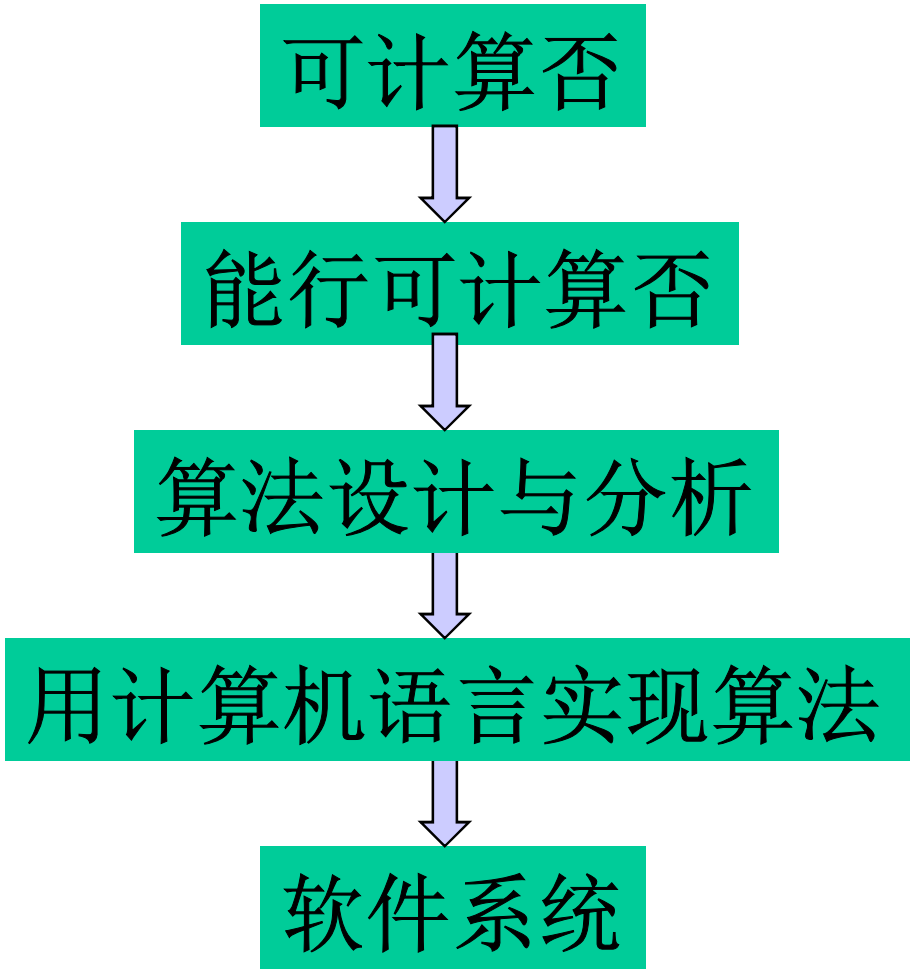
HIT
CS&E

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《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



HIT
CS&E

- 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

HIT
CS&E

- 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

HIT
CS&E

1.2 Concepts of Algorithms

- 算法的定义
- 问题的定义
- 算法的实例

HIT
CS&E

算法的定义

- 定义1.2.1(计算)
  - 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

HIT
CS&E

- 定义1.2.2（算法）
  - 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。
  - 最早的算法是欧几里德的“求最大公因子算法”
  - 直观地说，算法如下图所示：

  - 算法的目的是求解问题。什么是问题？

HIT
CS&E

问题的定义

- 定义1.2.3（问题）
  - 设Input和Output是两个集合。一个问题是一个关系R⊆Input×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

HIT
CS&E

- 例：整数SORT问题定义如下：
  - 输入集合Input={<a_1,…,a_n>|a_i是整数}
  - 输出集合Output={<b_1,…,b_n>|b_i是整数, $b_1 \le \ldots \le b_n$}
  - 问题SORT={(<a_1,…,a_n>,<b_1,…,b_n>)|<a_1,…,a_n>∈Input, <b_1,…, b_n>∈Output,{a_1, …, a_n}={b_1, …, b_n}}

- 定义1.2.4(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

HIT
CS&E

算法示例

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

HIT
CS&E

$$A[1,\ldots\ldots,n] = 5,2,4,6,1,3$$

$$A[1,\ldots\ldots,n] = 5,2,4,6,1,3$$

$$A[1,\ldots\ldots,n] = 2,5,4,6,1,3$$

$$A[1,\ldots\ldots,n] = 2,4,5,6,1,3$$

$$A[1,\ldots\ldots,n] = 2,4,5,6,1,3$$

$$A[1,\ldots\ldots,n] = 1,2,4,5,6,3$$

$$A[1,\ldots\ldots,n] = 1,2,3,4,5,6$$

HIT
CS&E

- 算法描述

Insertion-sort(A)

Input: A[1,.....,n]=n个数

output: A[1,.....,n]=n个sorted数

1. FOR j=2 To n Do

2. key←A[j];

3. i←j-1

4. WHILE i>0 AND A[i]>key Do

5. A[i+1]←A[i];

6. i←i-1;

7. A[i+1]←key;

- 实例：**A[1,.....,n]=5,2,4,6,1,3**

HIT
CS&E

1.3 Analyzing Algorithms

- 算法的正确性分析
- 算法的复杂性分析

HIT
CS&E

算法的正确性分析

- 定义1.3.1(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，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
    - 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!

循环不变量方法

——证明主要结构是循环结构的算法的正确性

循环不变量：数据或数据结构的关键性质

依赖于具体的算法和算法特点

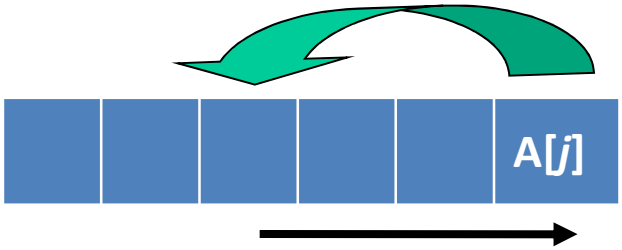
初始　　：循环开始前循环不变量成立

循环步骤：循环体每执行一次,循环不变量成立

终止　：算法结束后，循环不变量保证算法正确

循环不变量方法——实例

```
Insertion-sort(A)
1.  FOR j=2  To n  Do
2.      key←A[j];
3.      i←j-1
4.      WHILE i>0 AND A[i]>key  Do
5.          A[i+1]←A[i];
6.          i←i-1;
7.  A[i+1]←key;
```



循环不变量：循环变量为*j*但循环体未执行前，
A[1,2,...,*j*-1]来自输入且有序

初始 *j*=2：*A*[1]来自输入且有序

循环　　：设*A*[1,...,*j*-1]来自输入且有序
循环体执行一遍，*A*[1,...,*j*-1,*j*]来自输入且有序

终止*j*=*n*+1：*A*[1,2,...,*n*]来自输入且有序

HIT
CS&E

算法的复杂性分析

- 目的
  - 预测算法对不同输入所需资源量
- 复杂性测度
  - 时间，空间， I/O等, 是输入大小的函数
- 用途
  - 为求解一个问题选择最佳算法、最佳设备
- 需要的数学基础
  - 离散数学，组合数学，概率论，代数等
- 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

- 定义1.3.2 (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
    - 图论问题的输入大小=图的边数/结点数

- 定义**1.3.3**（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

- 定义**1.3.4**（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

HIT
CS&E

- 定义1.3.5（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

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

- 定义1.3.5（最小复杂性）

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

- 定义1.3.6（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

$$\sum_{y \in Input} p_y \times \text{complexity}(\text{size}(y))$$

对于一台每秒执行一百万条高级指令的处理器，不同的算法随输入规模增长的（大致）运行时间。在运行时间超过10^{25}年的情况下，记运行时间为非常长

	n	$n\log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
n =10	<1秒	<1秒	<1秒	<1秒	<1秒	<1秒	4秒
n =30	<1秒	<1秒	<1秒	<1秒	<1秒	18分	10^{25}年
n =50	<1秒	<1秒	<1秒	<1秒	11分	36年	非常长
n =100	<1秒	<1秒	<1秒	1秒	12892年	10^{17}年	非常长
n =1000	<1秒	<1秒	1秒	18分	非常长	非常长	非常长
n =10000	<1秒	<1秒	2分	12天	非常长	非常长	非常长
n =100000	<1秒	2秒	3小时	32年	非常长	非常长	非常长
n =1000000	1秒	20秒	12天	31710年	非常长	非常长	非常长

HIT
CS&E

1.4 Designing Algorithms

- 算法的设计方法
- 算法的分析方法

HIT
CS&E

算法的设计方法

- Divide-and-Conquer
- Dynamic Programming
- Greedy Algorithms
- Approximation Algorithms
- Randomlized Algorithms
- Tree Searching Strategies
- Prune-and-Search
- On-Line Algorithms
- Genetic Algorithms
- Parellel Algorithms

HIT
CS&E

算法的分析方法

- 不同的设计方法有不同的分析方法

HIT
CS&E

- 插入排序算法

```
Insertion-sort(A)
1.   FOR j = 2  To n  Do
2.       key←A[j];
3.       i←j-1
4.       WHILE i>0 AND A[i]>key  Do
5.            A[i+1]←A[i];
6.             i←i-1;
7.   A[i+1]←key;
```

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
8 $A[i+1] = key$	c_8	$n-1$

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

Data are sorted (best-case)

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8)\,. \end{aligned}$$

Data are reverse sorted (worst-case)

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\ &\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\ &= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right)n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right)n \\ &\quad - (c_2 + c_4 + c_5 + c_8)\,. \end{aligned}$$

HIT
CS&E

- 欧几里得算法
  - **Input： A,B ∈ Z^+**
  - **Output: r ∈ G且 ∀c ∈ G, c ≤ u ,**
    其中G = {c| c,s,t ∈ Z^+ 且A=c*s, B=c*t}

GCD(A, B)

1. m = A, n = B

2. while(n != 0) {

3. r = m % n;

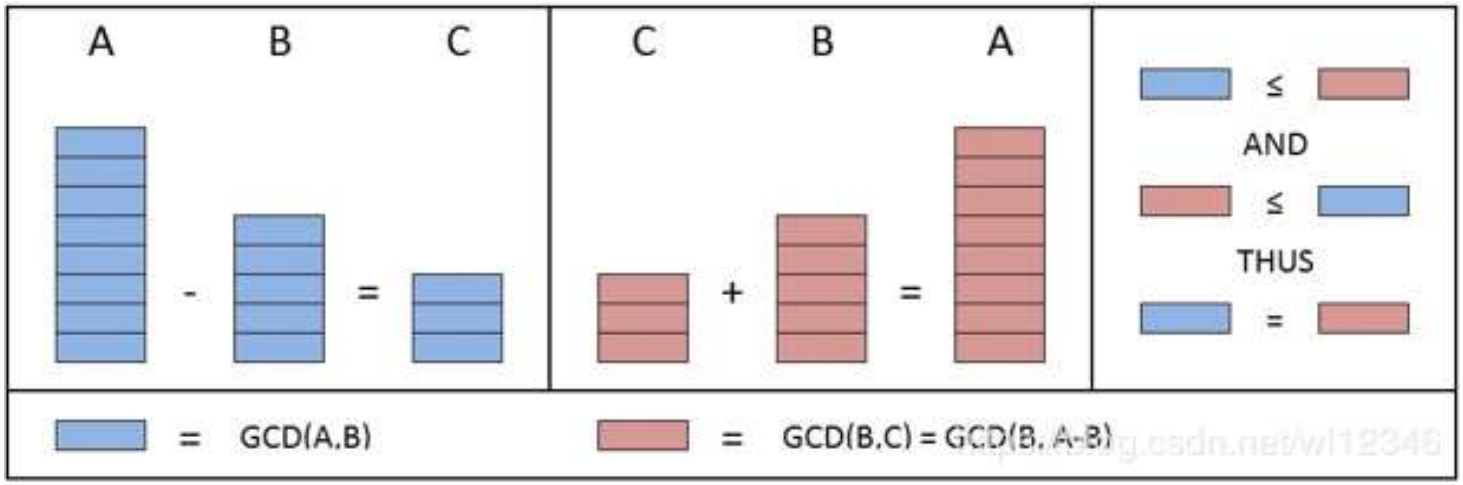
4. m = n;

5. n = r; }

6. Return m

• 欧几里得算法正确性证明

- **https://blog.csdn.net/wl12346/article/details/102624176**
- 设循环变量为n，循环开始时 n ≠0，GCD(A,B) = GCD(B,C)， 即GCD(A,B)=GCD(B,A-B)



- **X*GCD（A，B）=A；Y*GCD（A，B）=B； ==>A-B=X*GCD（A，B）-Y*GCD（A，B）=（X-Y）*GCD（A，B）=C ==>GCD（A，B）也是C的一个约数，GCD(A,B)≤GCD(B,C)**
- **GCD（B，C）是B和C的最大公约数，M*GCD（B，C）=B；N*GCD（B，C）=C； ==>B+C=M*GCD（B，C）+ N*GCD（B，C）=（M+N）*GCD（B，C）=A ==>GCD（B，C）也是A的一个约数, GCD(B,C)≤GCD(A,B)**
- **故GCD(A,B)=GCD(B,A-B)**
- **循环时上述结论成立**
- **循环结束时为 GCD(m, 0) = m**