算法设计与分析
Algorithm Design And Analysis

2021（秋）

哈尔滨工业大学（威海） 计算机科学与技术学院

教师信息

- 教师：王金宝
- 办公室：研究院北524
- 电话：13936192065
- 邮箱：wangjinbao@hit.edu.cn
- 研究方向：大数据分析

课程信息

- 课程消息和交流
  - QQ群、线下答疑（研究院北524）
- 成绩考核
  - 随堂测试、课后作业、实验：30%
  - 期末考试：70%
  - 其他：0%
- 一些课堂规则
  - 作业和考勤

为什么要学习算法

- 算法具有广泛的应用
  - 人类基因工程
  - 互联网
  - 电子商务
  - 大数据

最大公共子串匹配
最短路径
加密解密
亚线性算法
……

- 提升个人竞争力
  - 学术研究与算法关系紧密
  - 工业界中只会编程不会算法的人员价值较低

为什么要学习算法

- 学习过数据结构是否足够应对算法方面的问题？
  - 数据结构是存储和组织数据的方式，以便于访问和修改。
  - 数据结构课程中往往涉及一些算法，但侧重于一些数据结构的应用，并非针对算法本身
- 算法是一种重要的技术
  - 计算机各个方面的应用都直接或间接的应用算法
  - 技术自成体系
  - 算法在各种应用效率方面的影响比硬件还大

Outline of the course

- 绪论
- 数学基础
- 分治算法
- 动态规划
- 贪心算法
- 图算法
- 平摊分析
- 搜索策略
- NP完全性和近似算法

References

1. **Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms,* The MIT Press, Third Edition, 2012.**
2. **R.C.T.Lee, S.S.Tseng, R.C.Chang, and Y.T.Tsai, *Introduction to the Design and Analysis of Algorithms*, McGraw-Hill, 2007.**
3. **D. E. Knuth, *Art of the Computer Programming*, *Vol. 3, Addison-Wesley, 1973*.**
4. **A.V.Aho, J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.**

Important Conferences

- 计算机各个研究领域的Rank One会议
  - STOC、FOCS
  - SIGMOD/PODS、VLDB、ICDE
  - AAAI、IJCAI、KDD、ICCV
  - SIGCOMM、INFOCOM、MOBICOM、USENIX
  - SIGIR、SIGMETRICS
  - ……

Important Journals

- 计算机各个研究领域的Rank One期刊
  - Journal of the ACM、 Algorithmica
  - ACM TODS、TKDE、VLDB J.
  - IEEE/ACM Transactions on Networking、 JSAC、TMC
  - ACM Transactions on Computer Systems
  - IEEE Transactions on Computers
  - IEEE Transactions on Parallel and Distributed Systems
  - IEEE Trans on Pattern Analysis and Machine Intl
  - IEEE Trans on Image Processing
  - ……

第一章
绪 论

提要

1.1 Role of Algorithms in Computer Science
1.2 Concepts of Algorithms
1.3 Analyzing Algorithms
1.4 Designing Algorithms

1.1 Role of Algorithms in Computer Science

- 算法是计算机科学的重要主题
- 影响算法领域的重要学者
- 计算机科学体系及算法设计分析的地位

算法是计算机科学的重要主题

- **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》
    - 1968，1969，1973，2011
  - 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
  - 因具有划时代意义的这本书，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

影响算法世界的重要学者

美国Texas大学
计算机和
数学教授

1972年获得图灵奖
for ALGOL60编译器

Edsger W. Dijkstra

斯坦福大学教授

获1974年图灵奖 for
《The Art of Computer Programming》
and algorithm
design & analysis

Donald Ervin Knuth（高德纳）

Toronto大学教授

1982年获得图灵奖
for a famous paper:
"The Complexity of Theorem Proving Procedures" 奠定了
NP-完全理论基础

Stephen Arthur Cook

UC Berkely("三栖学者")

1985年获得图灵奖for
在算法理论特别是
NP-completeness理论
方面连续不断的贡献

Richard M. Karp

斯坦福大学教授

著名公式“算法+数据结构=程序”

获1984年图灵奖 for 开发了一系列计算机语言

Niklaus Wirth

Cornell University

1986年获得图灵奖 在算法及数据结构设计和分析方面的基础性成就，

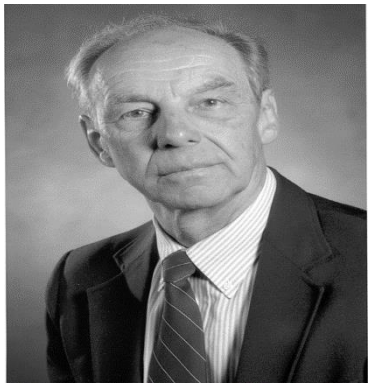
John Hopcroft

Cornell University

1986年获得图灵奖 在算法及数据结构设计和分析方面的基础性成就，

Robert E. Tarjan

Cornell University



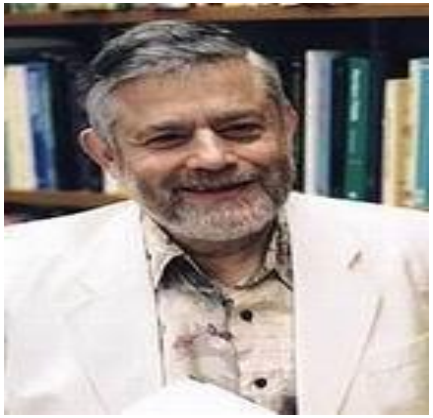
Juris Hartmanis

University at Albany
(奥尔巴尼)

Richard Edwin Stearns

1993年获得图灵奖
for a the paper:
"On the Computational Complexity of Algorithms" published in 1965
奠定了算法计算复杂性的理论基础

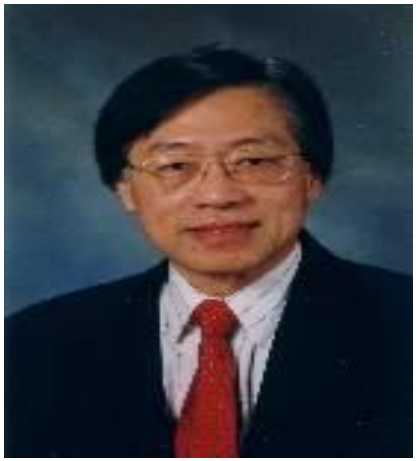
计算复杂性理论的主要奠基人



Manuel Blum

UC Berkely,
卡内基梅隆

1995年获得图灵奖
for在计算复杂性理论及其在密码学方面的应用.



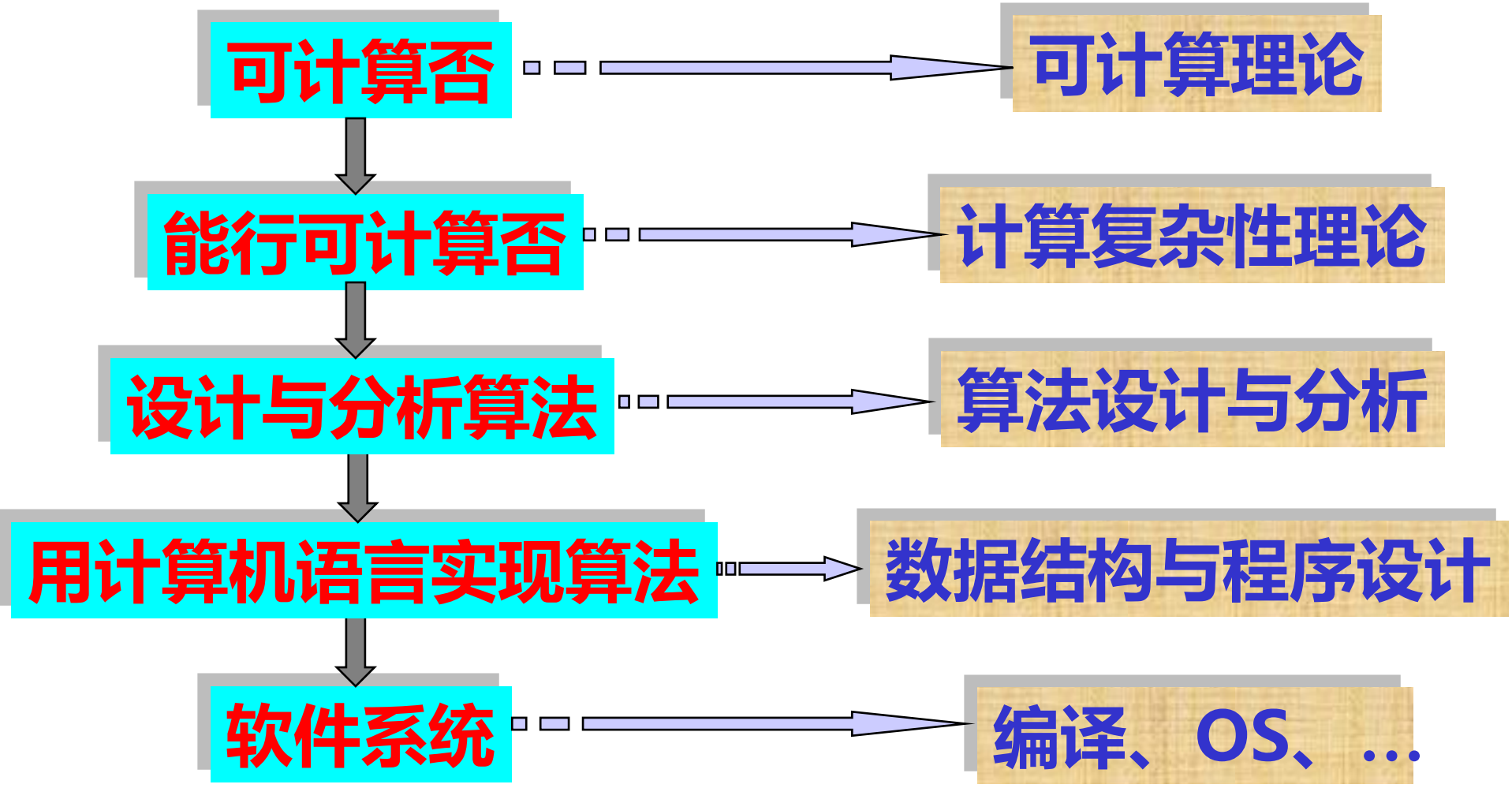
Andrew Yao（姚期智）

普林斯顿大学,
清华大学

2000年获得图灵奖
for在计算理论方面的贡献.

计算机科学技术的体系

- 解决一个计算问题的过程



- 可计算理论
  - 计算模型
    - Turing机、递归函数、λ演算、正则算法、POST系统
  - 可计算问题/不可计算问题
  - 计算模型的等价性--图灵/Church命题
    - 所有合理的计算模型，其计算能力是等价的
- 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
    - 复杂性上、下界
    - 平均复杂性
    - 复杂性问题的分类: P=NP?
    - 公理复杂性理论：抽象复杂性研究

- 算法设计和分析
  - 解决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

1.2 Concepts of Algorithms

- 算法的定义
- 问题的定义
- 算法的示例

算法的定义

定义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

定义2(算法)算法是一个满足下列条件的计算：

终止性：有限步内必须停止（有穷性），

确定性：每步都是严格定义和确定的动作，

能行性：每个动作都能被精确地机械执行，

输　入：具有满足给定约束条件的输入，

输　出：产生满足给定约束条件的结果。

算法的目的是求解问题。

什么是问题？

定义3 (问题) 设$Input$和$Output$是两个集合. 一个问题是一个关系$P \subseteq Input \times Output$, $Input$称为问题P的输入集合，$Input$的每个元素称为P的一个输入，$Output$称为问题P的输出或结果集合，$Output$的每个元素称为P的一个结果.

– 注意

- 问题定义了输入和输出的关系

例. SORT问题定义如下

输入: $Input = \{< a_1, \ldots, a_n > | a_i$是整数$\}$

输出: $Output = \{< b_1, \ldots, b_n > | b_i$是整数

$, b_1 \leq \ldots . \leq b_n\}$

问题: $SORT = \{(< a_1, \ldots, a_n >, < b_1, \ldots, b_n >) |$

$< a_1, \ldots, a_n > \in Input,$

$< b_1, \ldots, b_n > \in Output,$

$\{a_1, \ldots, a_n\} = \{b_1, \ldots, b_n\}\}$

SORT={(<3,2,1,5,4,6>,<1,2,3,4,5,6>), (<6,8,4>,<4,6,8>), (<9,6,7,3>,<3,6,7,9>), ……}

SORT={(<3,2,1,5,4,6>,<1,2,3,4,5,6>), (<6,8,4>,<4,6,8>),
(<9,6,7,3>,<3,6,7,9>), ……}

定义4(问题实例). 问题P的一个实例是P中的一个二元组.

二元组*(<9,6,7,3>,<3,6,7,9>)*是SORT问题的一个实例

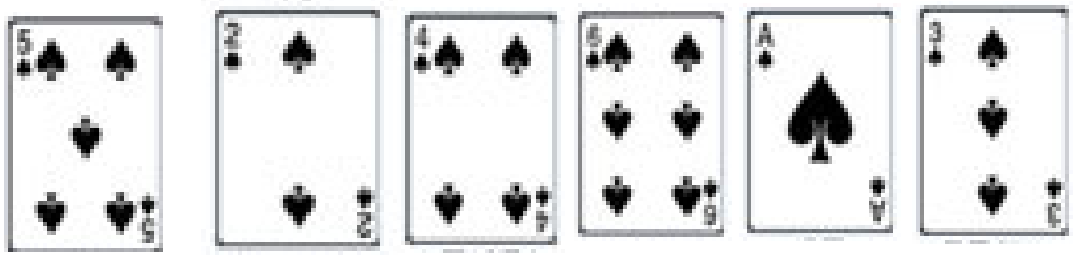
– 注意

- 问题是一个二元组集合，问题实例是一个二元组.
- 一个算法求解一个完整的问题，而不是仅求解一个问题的一个或几个实例.

算法示例

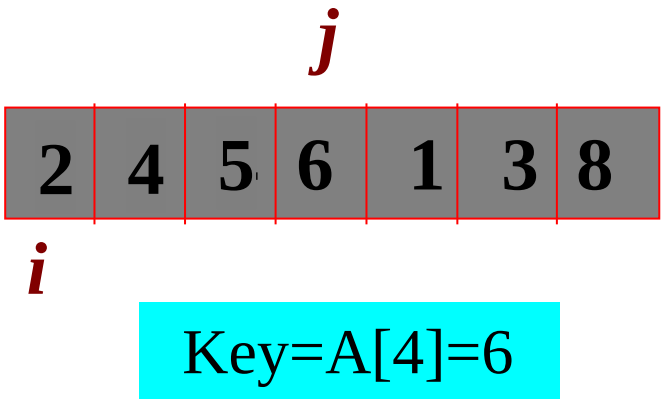
- 问题定义
  - Input=$\{<a_1,....,a_n> \mid a_i$是整数$\}$
  - output=$\{<b_1,....,b_n> \mid b_i$是整数,且$b_1 \le ... \le b_n\}$
  - $P=\{(<a_1,...,a_n>,<b_1,...,b_n>) \mid <a_1,...,a_n> \in \text{Input}, <b_1,...,b_n> \in \text{output}, \{a_1,...,a_n\}= \{b_1,...,b_n\}\}$
- 算法的思想
  - 扑克牌游戏

抓牌过程：



- 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];
//将A[j]插入到已排序的A[1…j-1]中
    i←j-1
    WHILE i>0 AND A[i]>key  Do
       A[i+1]←A[i];
       i←i-1;
    A[i+1]←key;
```



1.3 Analyzing Algorithms

- 算法的正确性分析
- 算法的复杂性分析

算法的正确性分析

定义5(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.

- 什么算法是不正确算法？
  - ①在一个或多个输入上不停止
  - ②对所有输入都停止，但对一个或多个输入产生不正确结果
- 什么是近似算法/随机算法的正确性？
  - ①对所有输入都停止
  - ②产生近似正确的解/产生不多的不正确解

- 为什么要进行正确性证明？
  - 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!!
- 如何证明算法的正确性？
  - 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

算法的复杂性分析

- 目的
  - 分析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6 (输入的大小). 设$Input$是问题P的输入集合，P的输入大小是一个函数

F：$Input \to N$，N是正整数集合.

– $size(\alpha)$表示输入$\alpha \in Input$的大小

– 示例：

- 排序问题的输入大小?
- 矩阵问题的输入大小?
- 图论问题的输入大小?

定义7 (实例时间复杂性). 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数.

– 注意

- 时间复杂性是输入大小的函数$T(n)$
- 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8(实例空间复杂性). 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.

– 注意

- 空间复杂性是输入大小的函数$S(n)$

定义9 (算法最坏复杂性). 设$Input$是问题P的输入集合，$Complexity(Size(x))$是求解P的实例x的算法A的复杂性函数，A的最坏复杂性是

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

定义10 (算法最小复杂性).

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

定义11 (算法平均复杂性). 设$x \in Input$, x作为算法A的输入出现的概率是p_x, A的平均复杂性为

$$\sum_{x \in Input} p_x \times \text{complexity}(\text{size}(x))$$

算法分析模型

- 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：integer、floating point
  - 基本操作：算数、数据移动、控制(每个操作常数时间)
- 并行多处理机模型(PRAM)

算法时间复杂性分析

- 算法执行时间的计算
  - 所有原子操作执行时间的和
  - 每个原子操作假设为常数时间

例如：计算n个数的平均值

S=0;	c_1
for i=1 to n	c_2
输入a值;	c_3
S=S+a;	c_4
S=S/n;	c_5
输出S值;	c_6

$$(n+1)c_2 + n(c_3+c_4) + c_1 + c_5 + c_6$$

- 时间复杂性分析并不是表示一个程序解决问题需要花多少时间，而是当问题规模扩大后，程序需要的时间长度增长得有多快
- 也就是说，对于高速处理数据的计算机来说，处理某一个特定数据的效率不能衡量一个程序的好坏，而应该看当这个数据的规模变大到数百倍后，程序运行时间是否还是一样，或者也慢了数百倍，或者变慢了数万倍
  - 不管数据有多大，程序处理花的时间始终是那么多的，我们就说这个程序很好，具有O(1)的时间复杂度，也称常数级复杂度

算法时间复杂性分析

- 算法描述

Insertion-sort(A) cost times

$$T(n)$$

$$= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) + c_6\left(\frac{n(n-1)}{2}\right)$$

$$+ c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1)$$

$$= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right)n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right)n - (c_2 + c_4 + c_5 + c_8)$$

A[i+1]←key; c_8 $n-1$

t_j：取值j时WHILE循环测试的次数，最坏情况下$t_j = j$

算法时间复杂性分析

- 直接插入排序算法的复杂性分析

输入序列：1，2，3，4，5，6

具有最小时间复杂性：最少比较次数为：$n-1$

输入序列：6，5，4，3，2，1

具有最坏时间复杂性：最多比较次数

$$\sum_{k=2}^{n}(k-1)=\frac{n(n-1)}{2}$$

算法时间复杂性分析

- 直接插入排序算法的复杂性分析

输入序列：5，2，4，6，1，3

排序过程：

5

2, 5

2, 4, 5

2, 4, 5, 6

1, 2, 4, 5, 6

1, 2, 3, 4, 5, 6

当读入第k个数x_k时：

(1). 将x_k与前面的数比较

(2). 若x_k为第j小，比较次数为$k-j+1$

x_k为第j小的概率为$1/k$.

x_k插入后形成有序序列而需要的比较次数：

$$\frac{1}{K}\sum_{j=1}^{K}(K-j+1)=\frac{1}{K}\left[K^2-\frac{K(K+1)}{2}+K\right]=\frac{K+1}{2}$$

总的平均比较次数：

$$\sum_{k=2}^{N}\left(\frac{K+1}{2}\right)=\sum_{i=1}^{N-1}\left(\frac{i+2}{2}\right)=\frac{1}{2}\left(\sum_{i=1}^{N-1}i+\sum_{i=1}^{N-1}2\right)=\frac{1}{4}N^2+\frac{3}{4}N-1$$

算法的正确性分析

- 循环不变量方法
  - 主要结构为循环的算法正确性证明的通用方法
- 主要步骤：
  - 确定循环不变量P(谓词)
    - 算法所操作的数据或数据结构具有的关键性质
  - 三个步骤：
    - 循环初始：即循环开始前，P成立
    - 循环步骤：循环体每执行一次之后P仍然成立
    - 循环终止：循环结束后，P成立，保证算法正确

算法的正确性分析

- 举例：插入排序

首先，定义循环不变量P：

对于任意循环变量j，A[1,.....,j-1]中数据有序

```
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];
//将A[j]插入到已排序的A[1…j-1]中
   i←j-1
   WHILE i>0 AND A[i]>key  Do
     A[i+1]←A[i];
     i←i-1;
   A[i+1]←key;
```

三个步骤：

初始：即循环开始前，j=2，A[1]中数据有序，P成立；

循环：循环体执行前A[1,.....,j-1]有序，第j次循环将A[j]插入到已排序的A[1…j-1]中，循环结束，A[1,.....,j]有序；

终止：j=n+1，循环结束，A[1,.....,n]有序。

1.4 Designing Algorithms

- 算法的设计方法
- 算法的分析方法

算法设计方法

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

算法的分析方法

- 不同的设计方法有不同的分析方法

What do we Learn?