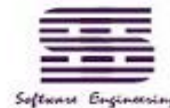


实用算法设计

主讲：余艳玮, ywyu@ustc.edu.cn



2018/9/14



关于课程教学与考核



2018/9/14

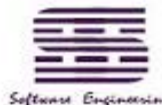


课程简介

- 课程性质
 - 专业基础课，至少**8%**末尾淘汰制
 - 学时：**60/30**； 学分：**3**
- 课程内容：
 - 常用的数据结构：
 - 基本数据结构：线性结构（顺序表，链表，栈和队列）
 - 其他：散列表，树，图
 - 两类典型算法：
 - 排序：
 - 查找：



2018/9/14



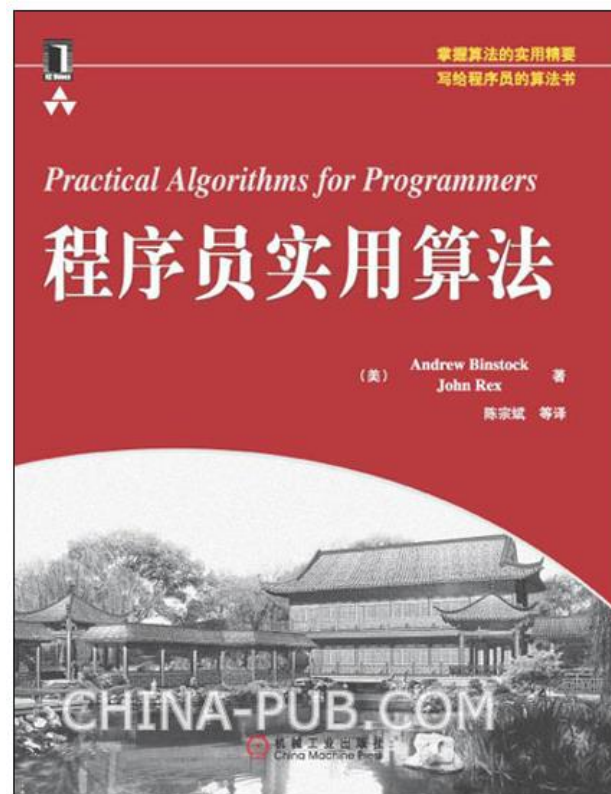
Software Engineering

- 教材

- 《程序员实用算法》 .
Andrew Binstock,
John Rex著, 陈宗斌等译,
机械工业出版社, **2009.**

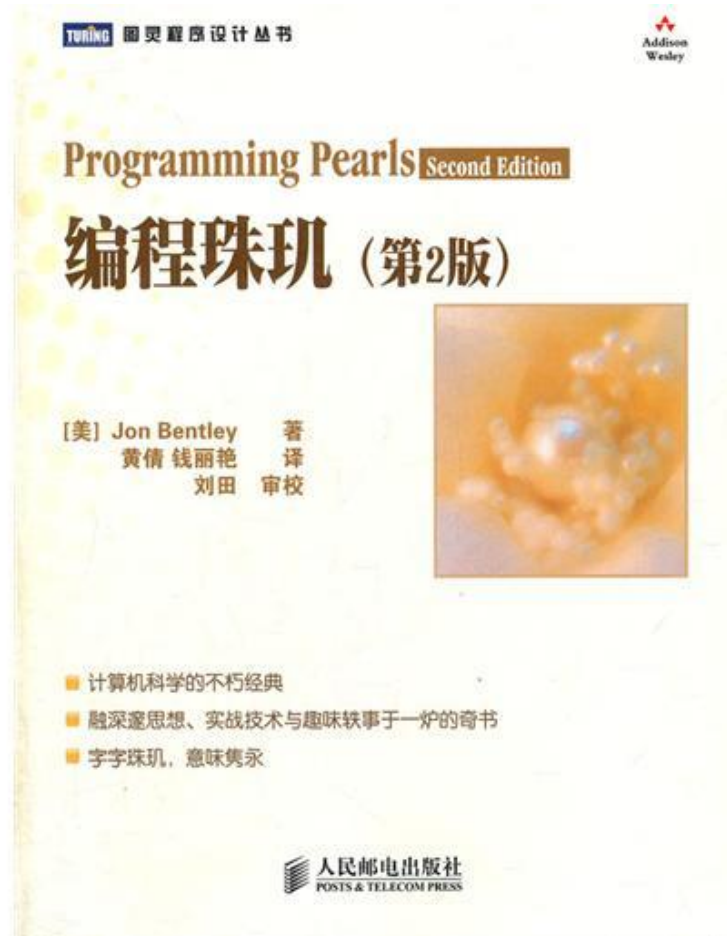
- 《编程珠玑》（第2版）
Jon Bentley著, 黄倩等
译, 人民邮电出版社,
2008.

- 《算法导论》

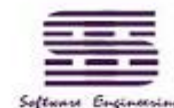


2018/9/14





- **Jon Bentley**，世界计算机科学家，被誉为影响算法发展的十位大师之一。
- 他先后任职于卡内基-梅隆大学（1976~1982）、贝尔实验室（1982~2001）和Avaya实验室（2001年至今）。
- 在卡内基-梅隆大学担任教授期间，他培养了包括Tcl语言设计者**John Ousterhout**、Java语言设计者**James Gosling**、《算法导论》作者之一**Charles Leiserson**在内的许多计算机科学大家。
- 2004年荣获Dr Dobb's程序设计卓越奖。



算法设计与分析

原书第3版



每个算法的分析
既易于理解
又十分有趣

严谨性 全面性 趣味有深度

选材与时俱进，具有实用性且能引起读者的兴趣

2 球与箱子

现在我们来考虑这样一个过程，即把相同的球随机投到 n 个箱子里，箱子编号为 $1, 2, \dots, n$ 。每次投球都是独立的，每一次投球，球等可能落在每一个箱子里，球落在任一箱子里的概率为 $1/n$ 。因此，投球的过程是一组伯努利试验（参见附录 C.4）。每次成功的概率是 $1/n$ ，其中成功是指球落入指定的箱子里。这个模型对分析洗牌（参见第 11 章）特别有用，而且我们可以回答关于该投球过程的各种有趣问题。（思考题 C-1 提出了另外一些问题关于球和箱子的问题。）

有多少球落在指定的箱子里？落在指定箱子里的球数服从二项分布 $B(n, 1/n)$ 。如果投 m 个球，公式 (C.37) 告诉我们，落在指定箱子里的球数期望值是 m/n 。

在平均意义上，我们必须投多少球，才能在指定的箱子里投中一个球？直至指定箱子收到一个球的投球次数服从几何分布，概率为 $1/n$ 。根据等式 (C.32)，成功的投球次数期望值是 $1/(1/n) = n$ 。

我们需要投多少次球，才能使每个箱子里至少有一个球？一次投球落在空箱子里称为一次“命中”。

我们知道为了获得 k 次命中，所需的投球次数期望是 k 。采用命中次数，可以把 m 次投球分为几个阶段。第 i 个阶段包括从第 $i-1$ 次命中到第 i 次命中之间的投球。第 i 阶段共需第 i 次投球，因为我们可以保证一次命中。此时所有的箱子都是空的。对第 i 阶段的每一次投球，有 $i-1$ 个箱子有球， $n-i+1$ 个箱子是空的。因而，对第 i 阶段的每次投球，得到一次命中的概率是 $(n-i+1)/n$ 。

以游戏引入，激发兴趣，让学习不再乏味，具有挑战性趣味性。

2 采用伪代码描述算法

INSERTION-SORT(A)

```
1 for j = 2 to A.length
2   key = A[j]
3   // Insert A[j] into the sorted sequence A[1..j-1].
4   i = j - 1
5   while i > 0 and A[i] > key
6     A[i+1] = A[i]
7     i = i - 1
8   A[i+1] = key
```

采用伪代码描述算法，即简洁易懂又便于抓住本质，再配上丰富的插图来描述和解释算法的执行过程，使得学习内容更加通俗，便于自学

3 分析全面

下面的伪代码实现了上面的思想，但有一个额外的变化，以避免在每个基本步骤 i 是否有牌为空。在每个牌的底部放置一张哨兵牌，它包含一个特殊的值，用于简化代码。我们使用 ∞ 作为哨兵牌，结果每当显示一张值为 ∞ 的牌，它不可能为较小的牌，除非两个牌都显示为哨兵牌。但是，一旦发生这种情况，所有哨兵牌都已被放置到输出堆，因为知道刚好 $r-p+1$ 张牌将被放置到输出堆，所以一旦已执行 $r-p+1$ 个基本步骤，算法就停止。

```
MERGE(A, p, q, r)
1  n1 = q - p + 1
2  n2 = r - q
3  let L[1..n1+1] and R[1..n2+1] be new arrays
4  for i = 1 to n1
5    L[i] = A[p + i - 1]
6  for j = 1 to n2
7    R[j] = A[q + j]
8  L[n1+1] = ∞
9  R[n2+1] = ∞
10 for k = 1 to n1 + n2
```

对算法正确性和复杂度的分析比较全面，即有严密的论证，又有直观的解释。

4 论述详细，有理有据

定理 4.3 令 $a \geq 1$ 和 $b > 1$ 是常数， $f(n)$ 是一个定义在 b 的幂上的非负函数， b 的幂上的函数：

$$g(n) = \sum_{i=0}^{\log_b n-1} a^i f(n/b^i)$$

对 b 的幂， $g(n)$ 有如下渐近界：

1. 若对某个常数 $c > 0$ 有 $f(n) = O(n^{bc-c})$ ，则 $g(n) = O(n^{bc-c})$ 。
2. 若 $f(n) = \Theta(n^{bc-c})$ ，则 $g(n) = \Theta(n^{bc-c} \lg n)$ 。
3. 若对某个常数 $c < 1$ 和所有足够大的 n 有 $a f(n/b) \leq c f(n)$ ，则 $g(n) = \Theta(f(n))$ 。


证明 对情况 1，我们有 $f(n) = O(n^{bc-c})$ ，这意味着 $f(n/b^i) = O((n/b^i)^{bc-c})$ 。式 (4.22) 得

$$g(n) = O\left(\sum_{i=0}^{\log_b n-1} a^i \left(\frac{n}{b^i}\right)^{bc-c}\right)$$

对于 O 符号内的和式，通过提取因子并化简来求它的界，得到一个递增的几何级数

既有结论性知识介绍，也有逐步导出结论的研究过程的展示



 点击看大图

定价：¥ 39.99

普通会员：¥ 32.79

1-3星会员：¥ 31.99

4-5星会员：¥ 30.79

校园优惠价：¥ 31.99(80折) (马上了解)

我要买 件

 加入购物车

 团购

基本信息

原书名：Algorithms

原出版社：McGraw-Hill Science/Engineering/Math

作者：(美) Sanjoy Dasgupta Christos H. Papadimitriou Umesh Vazirani [作译者介绍]

译者：王沛 唐扬斌 刘齐军

丛书名：国外经典教材·计算机科学与技术

出版社：清华大学出版社

ISBN：9787302179399

上架时间：2008-7-15

出版日期：2008 年7月

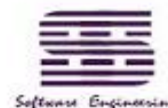
开本：16开

页码：345

版次：1-1

• <http://www.china-pub.com/41426&ref=ps>

2018/9/14



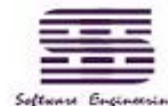
- 教学目标和要求：
 - 掌握常用的数据结构及相关基本操作；
 - 掌握典型的排序和查找算法；
 - 能选择/设计恰当、高效的算法来解决特定的问题；（通过分析实际案例，进行算法设计的思维方法的培养）
 - 能正确实现所设计的算法并进行适当优化。



- 课程考核：最终成绩由以下**4+1**项组成
 - 作业： **20%**;
 - **6次实验**： **40%**;
 - （随堂）期中测试： **10%**；（1小时）（学习“基于树的查找”之前）
 - 期末考试： **30%**；（涵盖整个课程内容）
 - 额外奖励： **2分**；邀请同学（或组）走上讲台，利用**1学时**与同学们分享作业/实验的思考过程或其他课程相关的学习分享。
 - 考勤： 随机的课堂作业



2018/9/14

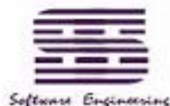


• 理论课程内容安排（60学时）

- 绪论：4学时
- 基本数据结构：12学时
 - 线性表：4学时
 - 算法设计技术：4学时
 - 栈、队列：3学时
 - 散列表：1学时
- 查找：24学时
 - 基于散列表的查找：3学时
 - 蛮力查找（顺序查找）：1学时
 - 基于有序表的二分查找：4学时
 - 字符串的查找：Boyer-Moore算法和KMP算法：4学时
 - 基于树的查找：二叉查找树、多路查找树（B-树和B+树）、Trie树：12学时
- 排序：8学时
 - 简单排序：1学时
 - 复杂排序：3学时
 - 案例分析：4学时
- 综合应用案例分析：4学时（海量数据的处理：统计、查询和排序）
- 图：4学时
 - 图的表示，查找，最小生成树，最短路径（单源和多源）
- 课程总结及答疑：4学时



2018/9/14



• 实验课程（30学时）

– **Lab1** :线性表（顺序表和单链表）的基本操作及其应用，6学时

基础实验

– **Lab2** :双向链表的综合应用，8学时

– **Lab3** :散列(Hash,哈希)表的综合应用，4学时

– **Lab4** :队列的综合应用，4学时

– **Lab5** :栈的综合应用，4学时

– **Lab6** :选择排序的实现，4学时

– 综合实验：Lab2~Lab5

• 利用现有的函数来解决给定问题

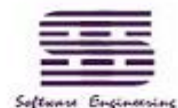
– 基础实验：Lab1, Lab6

• 自主独立编程

综合实验



2018/9/14



- 通过实验，希望学生：
 - 会写简单代码
 - 会修改代码以满足给定的新需求
 - 会利用通用函数/接口实现给定功能



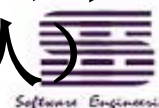
2018/9/14



- 作业：
 - 分为课前（验证型和阅读型）和课后(复习巩固)两类作业。
 - 分为独立完成和分组完成两种类型。
- 通过作业，希望学生：
 - 能理解基本概念，重要算法的原理和设计思想
 - 会简单分析并能验证算法的性能（时间和空间）
 - 通过验证算法（查找、排序）性能，能理解各类算法的适用场景以及优化策略，并能分析总结算法性能的影响因素。
 - 通过阅读已有代码，学习如何设计大型项目中的通用性函数/接口（分组阅读，每组**2**个人）



2018/9/14



Software Engineering

第1章 绪论

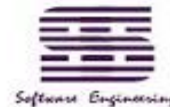
重点： 算法的概念、算法与相关术语的关联

难点： 算法时间复杂度的估算

基础： C语言编程



2018/9/14



1.1 什么是算法

1.2 为什么要学习算法？

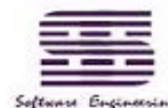
1.3 评估算法

1.4 设计算法的步骤

1.5 最佳算法选择的决定因素



2018/9/14



15

1.1 什么是算法

- 描述了特定问题的**有限**求解步骤;
 - 指为了解决特定问题, 而**操作数据**的方式;
 - 指求解问题的**策略**。
-
- 例子:
 - 问题1: 将大象放到冰箱里。
 - 问题2: 统计学生的数量。
 - 问题3: 计算 $[m+(m+1)+\dots+(m+n-1)]$ 。



- 算法的特征:

- 有穷性: 算法在执行有穷步之后能结束;
- 确定性: 每一条指令有确定的含义;
- 可行性: 每一操作都可以通过已实现的基本运算执行有限次来实现
- 输入: 零个或多个;
- 输出: 一个或多个;

- 例子:

- 问题1: 计算输入的2个数的和。
- 问题2: 计算输入的 n 个数 $\{a_1, a_2, \dots, a_i, \dots, a_j, \dots, a_n\}$ 中的第 i 个数和第 j 个数的和。
- 问题3: 将输入的 n 个数 $\{a_1, a_2, \dots, a_n\}$ 中的第 i 个数和第 j 个数的位置进行交换。



算法 vs. 程序

- 算法是所有程序的核心和灵魂，它一般被设计用于以最小的代价、高效的解决特定的问题。
- 程序：指为计算机处理问题而编制的一组指令。
- 算法 == 程序？
- 算法+数据结构 = 程序



数据结构 vs. 算法

- 数据结构：
 - 性质相同的数据元素的有限集合及其上的关系的有限集合。（数据+结构）
 - 是描述现实世界实体的数据模型及其上的操作在计算机上的表示和实现。
 - 包括逻辑结构和存储结构/物理结构。
- 例子：
 - 问题1：计算输入的2个数的和。
 - 问题2：计算输入的 n 个数 $\{a_1, a_2, \dots, a_n\}$ 中的第 i 个数和第 j 个数的和。
 - 问题3：将输入的 n 个数 $\{a_1, a_2, \dots, a_n\}$ 中的第 i 个数和第 j 个数的位置进行交换。
- 数据结构是影响算法选择的决定性因素之一。比如，链表的插入和顺序表的插入，性能是完全不一样的。



数据决定程序结构

- 数据结构对程序的贡献：可以结构化程序。将大程序缩减为小程序，减少代码量。
- 例：计算输入的 n 个数 $\{a_1, a_2, \dots, a_n\}$ 中的第 i 个数和第 j 个数的和。
- 问题：800电话号码有如下的格式：800-8222657，其中有效的800免费电话不超过800万个，比如不存在以0或1开头的有效免费电话。现要求对这些800免费电话号码进行排序，要求内存不超过1MB。（巧妙选择数据存储方式）
- 提示：可以使用位图法(**BitMap**)存放数据。



数据类型 vs. 变量

- 数据类型:
 - 如：系统定义的int, char等，用户自定义的struct类型
 - 也可采用typedef将类型名重命名，以增加代码的可读性。

- `int Sqlist[100];`

- `struct Node{
 int data;
 struct Node *next
};
Typedef struct Node *Link;
Link head;`

“顺序表”数据结构的实现；
描述了100个整型变量组成的集合，
且隐含着可以利用下标[]来描述两个整型变量之间的联系。

“单链表”数据结构的实现
隐含着可以利用指针next
来描述两个struct Node类
型的变量之间的联系。



1.2 为什么要学习算法？

- 算法无用：
 - “我做了这么多年，根本在实际开发中就没用过算法！”
 - “都已经有了现成的类库了，为啥还要去学算法呢？”
- 怎样才算会算法？
- 当“拿来主义”变成“完全拿来主义”。
- “重新发明轮子”。



- 问题1：从**2.5亿**个整数中找出不重复的数
字的个数。可用的内存限定为**600M**；要求
算法尽量高效，最优；
- 问题2：从**1亿**个整数中找出最大的**1万**个。
-



1.3 评估算法

- 基本要求：
 - 正确性
 - 可读性
 - 健壮性（异常处理机制）
- 更实用的要求：（性能）
 - 高效率（时间复杂度低）
 - 低存储量（内存开销小）



```
if ( argc < 3 || argc > 5 )  
{  
    fprintf ( stderr,  
        "Usage: BUFSIZE infile  
outfile [insize [outsize]]\n" );  
    return ( EXIT_FAILURE );  
}
```

健壮性处理



2018/9/14



- 时间复杂度：衡量算法运行得有多快。（本课程的重点）
 - 1) 如何度量？（估算和实际测量）
 - 通常不依赖于计时，而依赖于性能方程（大O表示法），以显示输入的大小/规模与性能的关系（找频率最高的语句）。
 - $O(1) < O(\lg N) < O(N) < O(N \cdot \lg N) < O(N^2) < O(N^k) < O(2^N)$ （常量阶、对数阶、线性阶、 $N \cdot \lg N$ 阶、平方阶、多项式阶、指数阶）（ $\lg N$ 是以2为底的对数）

例：

```
int m=0;
for (i=1;i<=n;++i)
  for (j=1;j<=n;++j){
    C[i][j]=0;
    for (k=1;k<=n;++k)
      C[i][j]+ =a[i][k]*b[k][j];
    m+=1;
  }
```

时间复杂度 $T(n)=O(n^3)$

实际运行时间 $T1=?$

```
for (i=1;i<=n;++i)
  for (j=1;j<=n;++j){
    for (k=1;k<=n;++k)
      printf("Hello! ");
  }
```

时间复杂度 $T(n)=O(n^3)$

实际运行时间 $T2=?$

$T1 ? T2$



频度最高语句的执行次数，不仅取决于输入规模，还取决于其他输入

//例3：利用二分查找在包含 n 个元素的表 t 上查找指定值 t .

```
int binarysearch2(DataType t)
{   int l, u, m;
    l = 0;
    u = n-1;
    while (l <= u) {
        m = (l + u) / 2;
        if (x[m] < t)
            l = m+1;
        else if (x[m] == t)
            return m;
        else /* x[m] > t */
            u = m-1;
    }

    return -1;}

```

最好的时间复杂度 $T(n)=?$

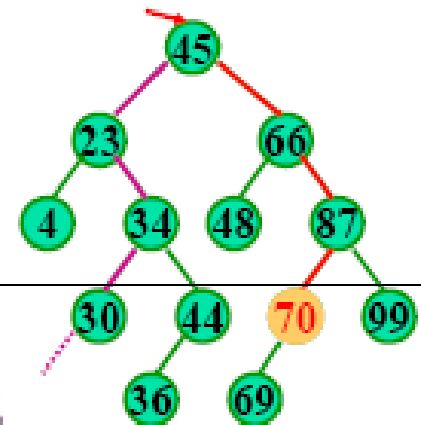
最坏的时间复杂度 $T(n)=?$

平均的时间复杂度 $T(n)=?$

//例4：升序输出二叉查找树 p 中所有结点.

```
void traverse(node *p)
{   if (p == 0)
        return;
    traverse(p->left);
    v[vn++] = p->val;
    traverse(p->right);
}
```

时间复杂度 $T(n)=?$



- 2) 综合考虑三种情况下的性能方程：平均情况，最坏情况和最好情况。
 - a) 考虑到最坏情况是很重要的。有时，最坏情况下的性能有时很差，此时采用一种能自如处理最坏情况的、通常慢一些的算法可能是一个更好的选择。比如，在`qsort()`中考虑到了快速排序的最坏情况。
 - b) 在考虑算法时，应该从可能性和后果两方面研究最坏情况。虽然最坏情况极少发生，但是若后果是灾难性的而你无法改变算法本身，那么必须为最坏情况事先做好准备。如，“哲学家”进餐问题。
 - c) 最好情况虽然没有什么危险，且极少发生，并且也可能极度依赖于输入项，但是，也需分析最好情况下的性能。如，冒泡排序在最好情况下性能为 $O(N)$ ，此时要求输入文件为有序。在某些应用场合中，一个算法的输出是另一个算法的输入，因此可以预见要处理的数据的质量，因而此时可有效使用最好情况下的性能。



- 存储空间：内存开销。
 - 有时候，特定问题的应用背景对可用的内存大小进行了约束。
 - 例：
 - 问题：800电话号码有如下的格式：800-8222657，其中有效的800免费电话不超过800万个，比如不存在以0或1开头的有效免费电话。现要求对这些800免费电话号码进行排序，要求内存不超过1MB。（巧妙选择数据存储方式）
 - 提示：可以使用位图法(BitMap)存放数据。
 - 如何度量？（估算）



1.4 设计算法的步骤

- **Step1:** 问题定义;
- **Step2:** 系统结构的设计。（指将大型系统进行模块分解，是属于程序设计思想的范畴）。（本课程假定所要解决的问题不涉及复杂的大系统，因而此步骤不予考虑）
- **Step3:** **算法设计及数据结构的选择**。依据问题定义、输入数据的特征和要求输出的数据的特征，分析广泛的解决方案（数据结构+算法），并选择最佳的解决方案；（本课程的重点）
 - 解决方案：数据结构+算法
 - 最佳的解决方案的确定：依赖于良好的数据结构和算法的选择。



- **Step4: 代码调优。实现并优化代码。**
 - 效率 vs. 清晰的程序结构/可维护性
 - 原则:
 - 1) 尽量减少输入输出；减少函数调用的次数；限制计算密集型操作（浮点运算，除法运算）；
 - 2) 然后，确定最耗时的操作，并提高其性能；（如，冒泡排序中，应关注比较和交换）；
 - 可以用测量和跟踪工具（如，**Profiler**, **AQTime**）；
 - 也可以用手动测试进行性能监视（打印所花费的时间与输入规模之间的关系，后面在“二分查找”部分会介绍）



- 问题分析：

- 问题1：给定一个英语词典，找出其中的所有变位词集合。例如，” **pots**”、” **stop**”和” **tops**”互为变位词。
- 提示：用**Hash**表来存储变位词（**Hash**表的广泛应用）
- 问题2：给定一个最多包含**40亿个随机排列的32 bits**整数的顺序文件，找出一个不在文件中的**32 bits**整数（在文件中必然缺失一个这样的数——为什么？）。
 - ① 在具有足够内存的情况下，如何解决该问题？
 - ② 如果有几个外部的“临时”文件可用，但是仅有几百字节的内存，又该如何解决？（归并排序+二分搜索）



1.5 最佳算法选择的决定因素

1) 问题的约束:

- 比如, 可用内存空间, 运行时间的上限约束等。

2) 数据的存储方式

- 存储什么信息? (比如, 位图法)
- 选用何种数据结构? (取决于在数据上的常用操作类型(静态? 动态?)、以及内存空间的限制)
- 比如, 顺序表上进行插入/删除比较慢, 所以, 在顺序表上实现交换2个元素的算法要尽量回避这些速度慢的操作。

3) 输入数据的特征: 是否要求有序?

4) 输出数据的特征: 是否要求有序? 是否包含重复记录。

案例分析

- 程序清单1-1

- 问题：测试你自己的系统上最佳的I/O缓冲区大小，它接收四个参数：要复制的测试文件（源文件名）、目标文件的名称、输入缓冲区大小、输出缓冲区大小。

- 解题思路：

- 1) 问题定义；
- 2) 系统结构的设计。
- **Step3: 算法设计及数据结构的选择。**
- **Step4: 代码调优。**

- 算法设计： $T(n)=O(1)$

- **Step1:** 进行参数个数的判断；
- **Step2:** 将输入缓冲区大小和输出缓冲区大小分别转换为整数并储存；
- **Step3:** 复制文件5次，并打印出所花费的最大、最小和平均时间
 - **For (i=1;i<=5,i++)**
 - 计算复制文件1次所花费的时间
 - 保存最大的时间
 - 保存最小的时间
 - 计算所花费的平均时间
 - 打印所花费的最大、最小和平均时间



- 计算复制文件1次所花费的时间

- **Step1:** 按读方式打开源文件;
- **Step2:** 设置输入缓冲区的大小;
- **Step3:** 按写方式打开目标文件;
- **Step4:** 设置输出缓冲区的大小;
- **Step5:** 保存当前时间;
- **Step6:** 逐字符地读取输入缓冲区, 并逐字符地写入到输出缓冲区中, 直至读完源文件。
- **Step7:** 保存当前时间;
- **Step8:** 关闭源文件;
- **Step9:** 关闭目标文件;
- **Step10:** 计算两次时间的差值, 即为复制文件所花费的时间。



作业1(课前)

- 重新生成SourceCodeForTeaching下2个项目文件（Ch1、Ch2_1），再要求：
 - 1) 成功运行项目后，将运行结果（打印出来的信息）截图；
 - 2) 指出项目中使用了哪些数据结构、包含哪几个函数，并添加代码注释；
 - 3) 用算法框图画出main函数中的算法思路。（包含子函数的方框，用下划线标注出来）
 - 4) 总结.c文件中所有子函数（不包含main函数）的代码实现的思路（若有不理解的代码，用红色标记出来）
- **作业命名规则：**“课前01_姓名.学号_姓名.学号”
要求同学**分组完成**，每组2个人，每组最后提交一份实验报告。具体操作为，同一组的组员可以提供同一份报告，但是需在报告中指明包含哪些组员，你负责具体哪些工作。
希望所有同学都能参与到源码阅读过程，积极提高代码的阅读能力，这个将是后面进行独立编程的基础。