# 实用算法设计

主讲：余艳玮, ywyu@ustc.edu.cn
助教：耿洋洋, geng325@mail.ustc.edu.cn
肖文扬, xwy@coolxxy.cn
曹婧, congjia@mail.ustc.edu.cn

# 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

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

# 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)+…+(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{$a_1, a_2, \ldots, a_i, \ldots, a_j, \ldots, a_n$}中的第i个数和第j个数的和。
  - 问题3：将输入的n个数{$a_1, a_2, \ldots a_n$}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{$a_1, a_2, \ldots, a_n$}中的第i个数和第j个数的和。
  - 问题3：将输入的n个数{$a_1, a_2, \ldots, a_n$}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。

# 数据结构 vs. 数据类型

- 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.

# 数据结构 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{a1,a2,…an}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。

# 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

- 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（找频度最高的语句）。
    - $O(1) < O(\lg N) < O(N) < O(N*\lg N) < O(N^2) < O(N^k) < O(2^N)$（常量阶、对数阶、线性阶、N*lgN阶、平方阶、多项式阶、指数阶）(lgN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;}
```

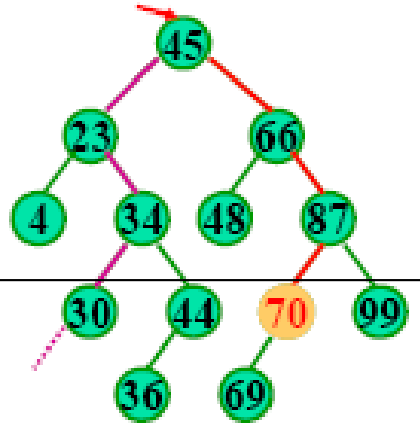
最好的时间复杂度T(n)=?

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又无法改变算法本身，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+栈空间开销。
  - 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。
  - 栈空间开销:
    - 比如，排序时，可能需额外的临时空间来存储一个或更多待排序的记录；
    - 隐式的额外空间需求：递归方式实现的算法对栈空间的需求
  - 示例：
  - 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？（理论估算，实际测量）

# 测量：实际的内存空间开销

```
#define MEASURE(T,text){\
 cout <<text<<"\t"<<sizeof(T) <<"字节\t" <<"实际消耗的内存空间为：
    （字节）";\
        int lastp2=0;\
        T *sentinel;\
        for (int i=0;i<11;i++) {\
                sentinel = new T;\
                int thisp2=(int)sentinel;\
                cout<< " " << thisp2-lastp2;\
                lastp2 = thisp2;\
        }\
        cout <<"\n";\
}
```

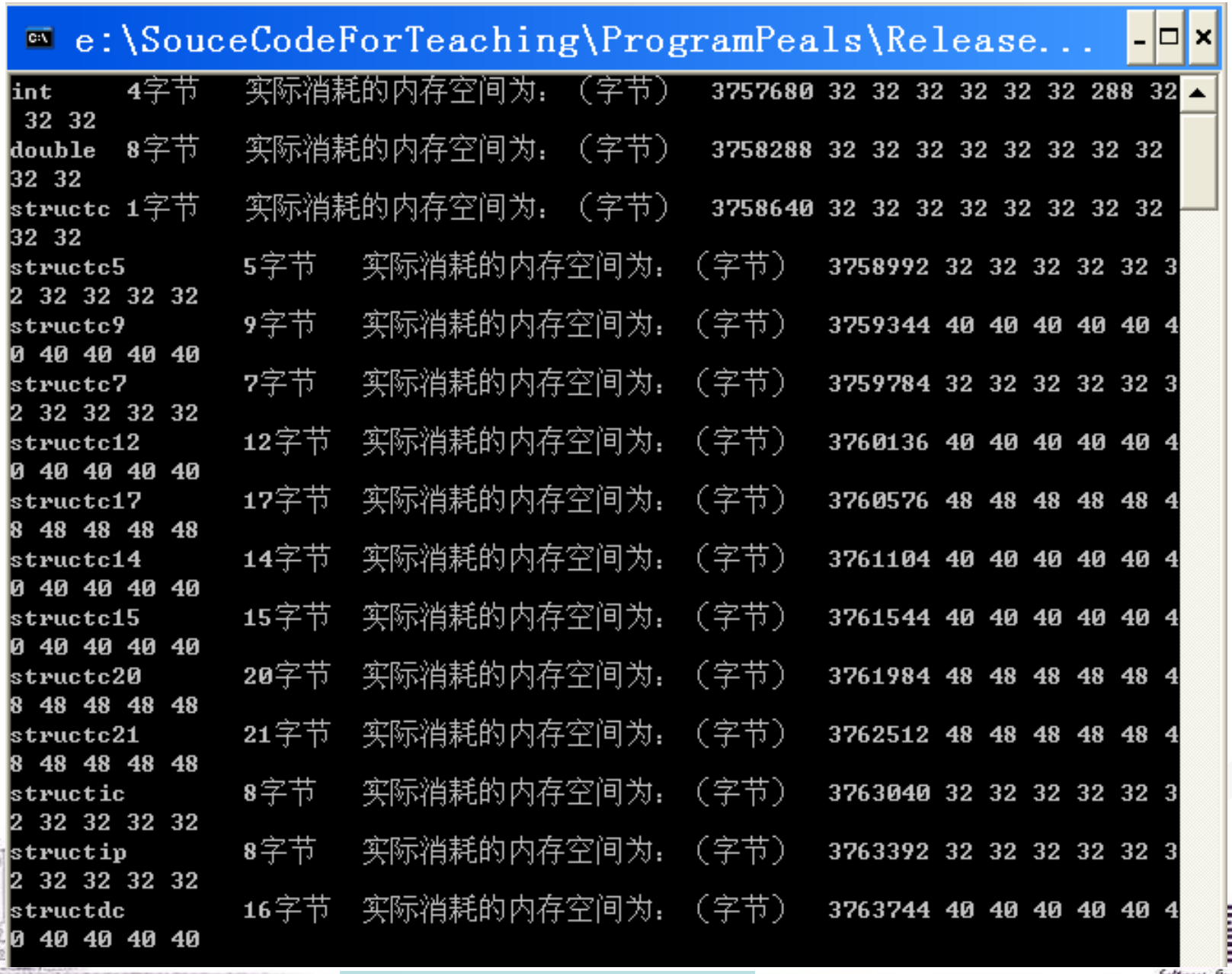
```
int main()
{    MEASURE(int,"int");
     MEASURE(double,"double");
     struct structc {char c;};
     struct structc7 {char c[7];};
     struct structc9 {char c[9];};
     struct structc12 {char c[12];};
     struct structc15 {char c[15];};
     struct structc20 {char c[20];};
     struct structc21 {char c[21];};
     struct structic {int i;char c;};
     struct structip {int i;structip *p;};
     struct structdc {double d;char c;};
     MEASURE(structc,"structc");
     MEASURE(structc9,"structc9");
     MEASURE(structc7,"structc7");
     MEASURE(structc12,"structc12");
     MEASURE(structc15,"structc15");
     MEASURE(structc20,"structc20");
     MEASURE(structc21,"structc21");
     MEASURE(structic,"structic");
     MEASURE(structip,"structip");
     MEASURE(structdc,"structdc");
     return 1;}
```

申请1~8B：实际分配32B；

申请9~16B：实际分配40B；

申请17~24B：实际分配48B；

申请分配内存空间1次，额外开销为：24~31B


e:\SouceCodeForTeaching\ProgramPeals\Release...
int 4字节 实际消耗的内存空间为：（字节） 3757680 32 32 32 32 32 32 288 32 32 32
double 8字节 实际消耗的内存空间为：（字节） 3758288 32 32 32 32 32 32 32 32 32 32
structc 1字节 实际消耗的内存空间为：（字节） 3758640 32 32 32 32 32 32 32 32 32 32
structc5 5字节 实际消耗的内存空间为：（字节） 3758992 32 32 32 32 32 32 32 32 32 32
structc9 9字节 实际消耗的内存空间为：（字节） 3759344 40 40 40 40 40 40 40 40 40 40
structc7 7字节 实际消耗的内存空间为：（字节） 3759784 32 32 32 32 32 32 32 32 32 32
structc12 12字节 实际消耗的内存空间为：（字节） 3760136 40 40 40 40 40 40 40 40 40 40
structc17 17字节 实际消耗的内存空间为：（字节） 3760576 48 48 48 48 48 48 48 48 48 48
structc14 14字节 实际消耗的内存空间为：（字节） 3761104 40 40 40 40 40 40 40 40 40 40
structc15 15字节 实际消耗的内存空间为：（字节） 3761544 40 40 40 40 40 40 40 40 40 40
structc20 20字节 实际消耗的内存空间为：（字节） 3761984 48 48 48 48 48 48 48 48 48 48
structc21 21字节 实际消耗的内存空间为：（字节） 3762512 48 48 48 48 48 48 48 48 48 48
structic 8字节 实际消耗的内存空间为：（字节） 3763040 32 32 32 32 32 32 32 32 32 32
structip 8字节 实际消耗的内存空间为：（字节） 3763392 32 32 32 32 32 32 32 32 32 32
structdc 16字节 实际消耗的内存空间为：（字节） 3763744 40 40 40 40 40 40 40 40 40 40

- 结论：应尽可能减少申请分配内存空间的次数
- 好处：
  - 减少额外开销
  - 可提高运行速度，原因有二：
    - 1）减少额外开销后，Cache中的元素数增加了，可以提高处理速度（Malloc之后的Cache情况）
    - 2）内存的分配和销毁，属于耗时操作。
- 如何减少申请分配内存空间的次数？
  - 一次性手动申请大片的内存空间作为内存池（还可尽量避免产生内存碎片）
  - 按实际需要从内存池中取出相应大小的空间来使用，若内存池中空闲空间不够，则再申请新的大片内存空间。
  - 内存空间用完后不直接回收，而是返回给内存池
  - 注意：内存池不用时，需手动释放。

# 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SouceCodeForTeaching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。