第1章　概论

- 问题求解
- 数据结构及抽象数据类型
- 算法的特性及分类
- 算法的效率度量

算法的渐进分析

$$f(n) = n^2 + 100n + \log_{10} n + 1000$$

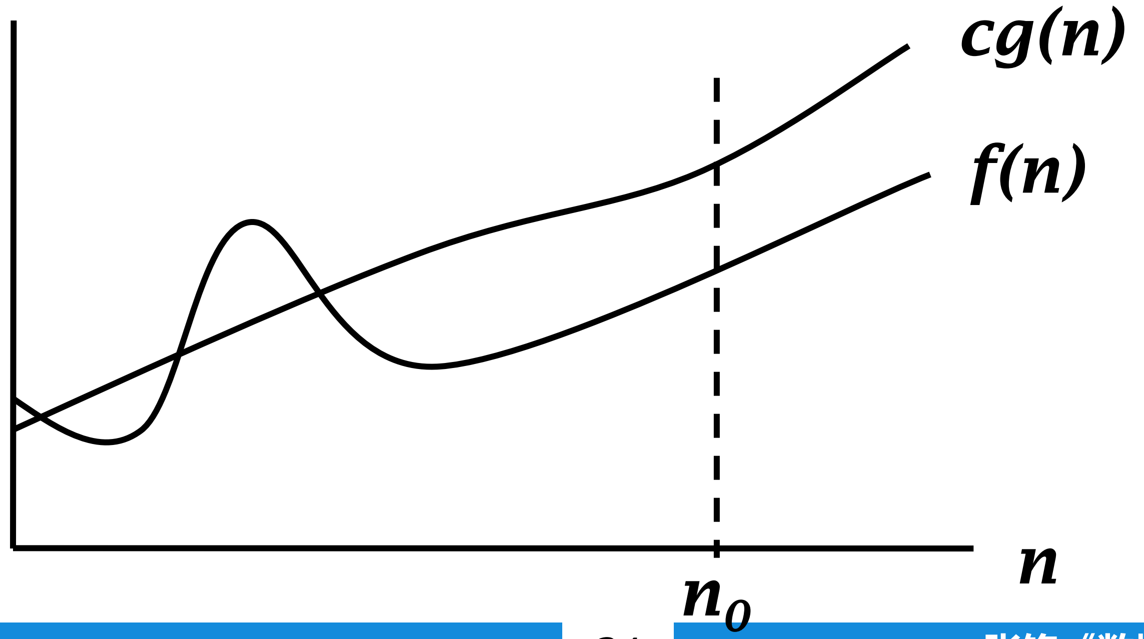
- 数据规模 n 逐步增大时，f(n) 的增长趋势
- 当 n 增大到一定值以后，计算公式中影响最大的就是 n 的幂次最高的项
  - 其他的常数项和低幂次项都可以忽略

算法渐进分析：大O表式法

- 函数 f，g 定义域为自然数，值域非负实数集
- **定义**：如果存在正数 c 和 n_0 ，使得对任意的 $n \geq n_0$， 都有 $f(n) \leq cg(n)$，
- 称f(n)在集合O(g(n)) 中，简称 f(n) 是 O(g(n)) 的，或 f(n) = O(g(n))
- 大 O 表示法：表达函数增长率上限
  - 一个函数增长率的上限可能不止一个
- 当上、下限相同时则可用 Θ 表示法

大 O 表示法

- $f(n) = O(g(n))$，当且仅当
  - 存在两个参数 $c > 0$ ，$n_0 > 0$，对于所有的 $n \geq n_0$ ，都有 $f(n) \leq cg(n)$
- iff $\exists\ c, n_0 > 0$ s.t. $\forall\, n \geq n_0 : 0 \leq f(n) \leq cg(n)$



n足够大
$g(n)$是 $f(n)$ 的上界

大 O 表示法的单位时间

- 简单布尔或算术运算
- 简单 I/O
  - 指函数的输入/输出
    例如，从数组读数据等操作
  - 不包括键盘文件等 I/O
- 函数返回

大 O 表示法的运算法则

- **加法规则:** $f_1(n)+f_2(n)=O(\max(f_1(n), f_2(n)))$
  - 顺序结构，if 结构，switch 结构
- **乘法规则:** $f_1(n) \cdot f_2(n) = O(f_1(n) \cdot f_2(n))$
  - for, while, do-while 结构

```
for (i=0; j<n; i++)
    for (j=i; j<n; j++)
        k++;
```

$$n-i$$

?

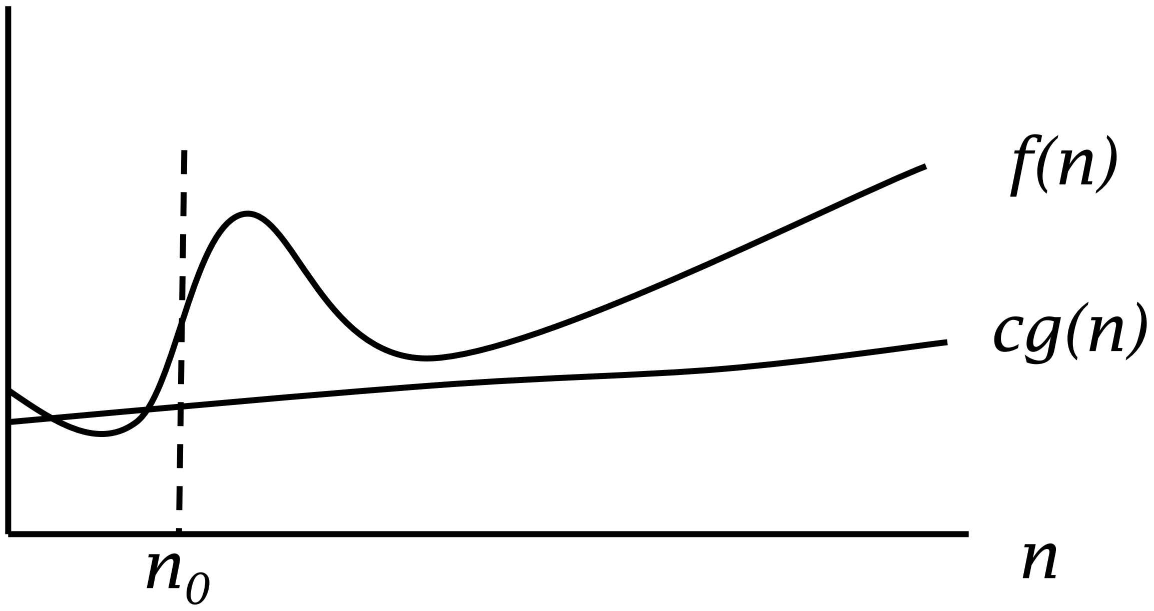
$$\sum_{i=0}^{n-1}(n-i)=\frac{n(n-1)}{2}=\frac{n^2-n}{2}=O(n^2)$$

算法渐进分析：大 Ω 表式法

- **定义** ：如果存在正数 c 和 n_0 ，使得对所有的 $n \geq n_0$ ，都有 $f(n) \geq cg(n)$,
  则称 f(n) 在集合 Ω (g(n)) 中，或简称 f(n) 是 Ω (g(n)) 的， 或 f(n) = Ω (g(n))
- 大 O 表示法和大 Ω 表示法的唯一区别在于不等式的方向而已
- 采用大 Ω 表示法时， 最好找出在函数增值率的所有下限中那个最“紧” （即最大）的下限

大 Ω 表示法

- $f(n) = \Omega(g(n))$
  - iff $\exists\, c, n_0 > 0$ s.t. $\forall\, n \geq n0$, $0 \leq cg(n) \leq f(n)$
- 与大O表示法的唯一区别在于不等式的方向



n足够大
$g(n)$是 $f(n)$ 的下界

算法渐进分析：大 Θ 表示法

- 当上、下限相同时则可用 Θ 表示法
- 定义如下：

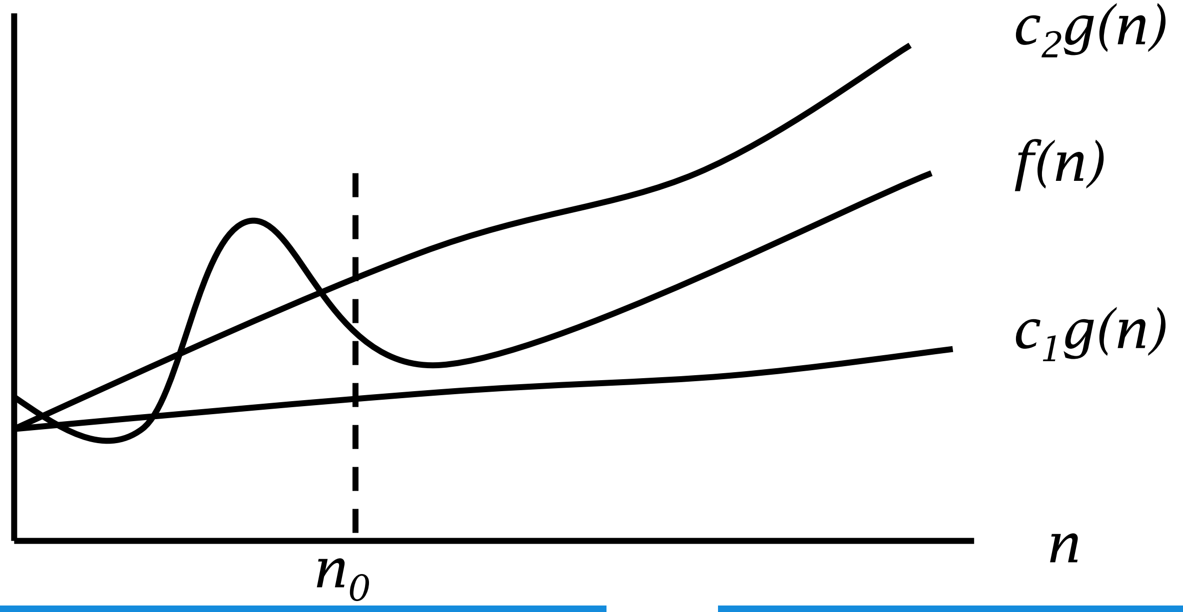
  如果一个函数既在集合 O (g(n)) 中又在集合 Ω (g(n)) 中，则称其为 Θ (g(n))。

- 也即，当上、下限相同时则可用大 Θ 表示法
- 存在正常数 c_1, c_2，以及正整数 n_0，使得对于任意的正整数 $n > n_0$，有下列两不等式同时成立：

$$c_1\, g(n) \le f(n) \le c_2\, g(n)$$

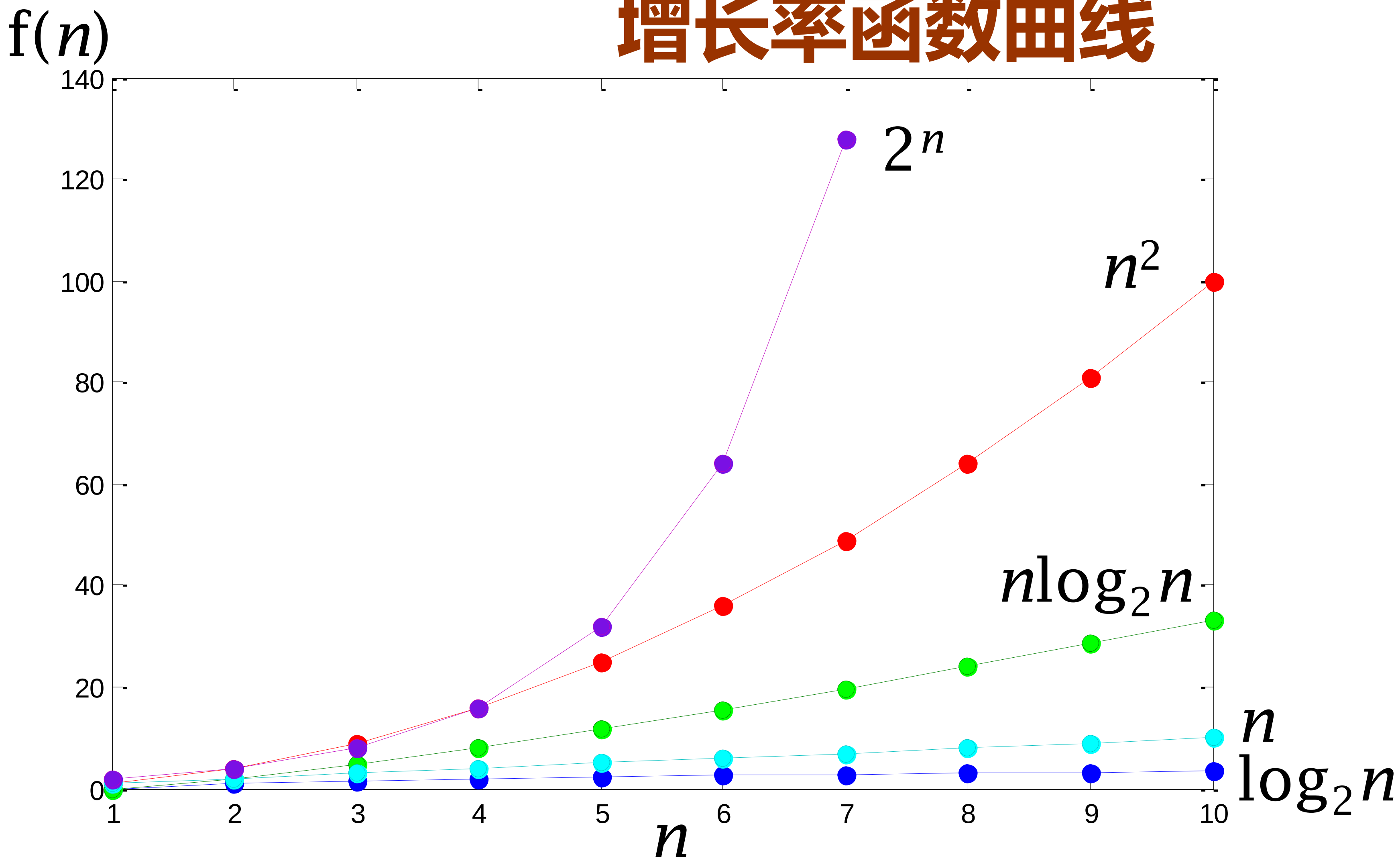
大 Θ 表示法

- $f(n) = \Theta(g(n))$
  - iff $\exists\, c_1, c_2, n_0 > 0$ s.t. $0 \le c_1 g(n) \le f(n) \le c_2 g(n),\ \forall n \ge n_0$
- 上、下限相同，则可用 Θ 表示法

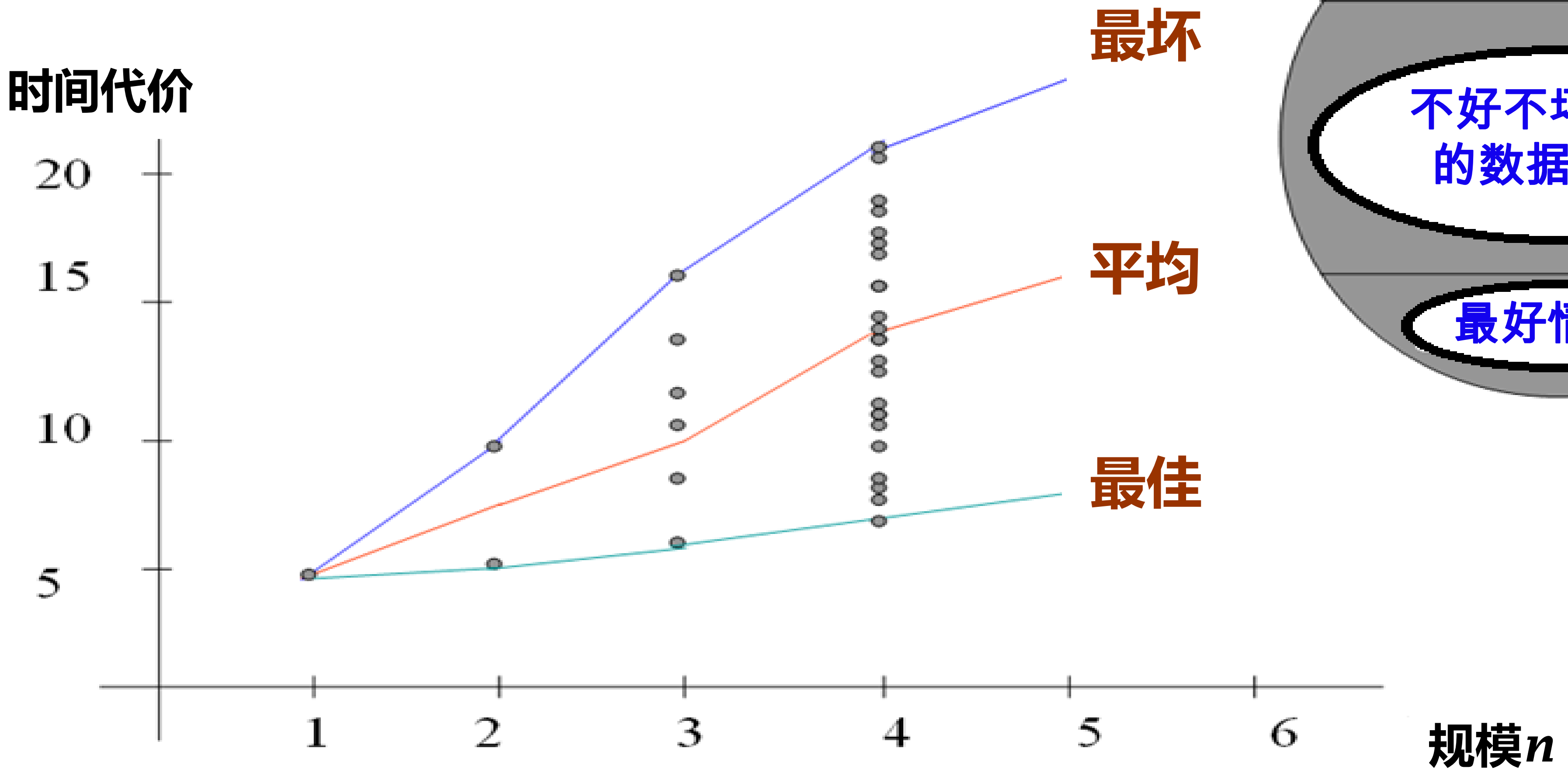


n足够大
$f(n)$ 与 $g(n)$ 增长率一样

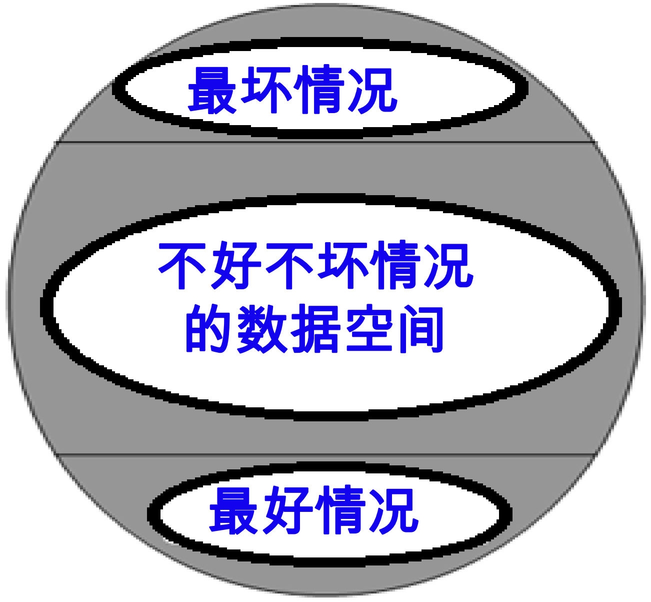
增长率函数曲线



问题空间 vs 时间开销



问题的输入数据空间



顺序找 k 值

- 顺序从一个规模为 n 的一维数组中找出一个给定的 K 值
- 最佳情况
  - 数组中第 1 个元素就是 K
  - 只要检查一个元素
- 最差情况
  - K 是数组的最后一个元素
  - 检查数组中所有的 n 个元素

顺序找 k 值——平均情况

- 如果等概率分布
  - K 值出现在 n 个位置上概率都是 1/n
- 则平均代价为 O(n)

$$\frac{1+2+\ldots+n}{n}=\frac{n+1}{2}$$

顺序找 k 值——平均情况

- 概率不等
  - 出现在第 1 个位置的概率为 1/2
  - 第 2 个位置上的概率为 1/4
  - 出现在其他位置的概率都是

$$\frac{1-1/2-1/4}{n-2}=\frac{1}{4(n-2)}$$

- 平均代价为 O(n)

$$\frac{1}{2}+\frac{2}{4}+\frac{3+\ldots+n}{4(n-2)}=1+\frac{n(n+1)-6}{8(n-2)}=1+\frac{n+3}{8}$$

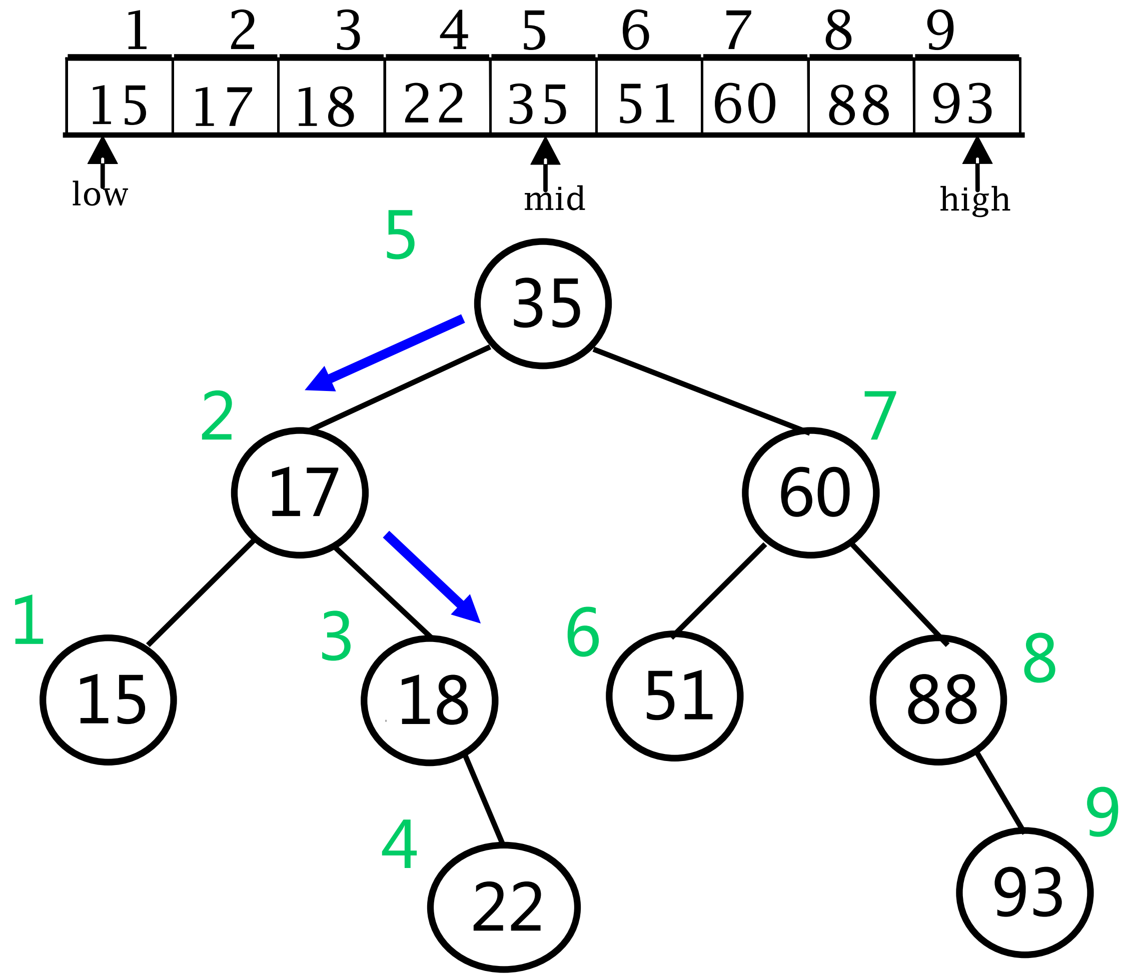
二分法找 k 值

对于已排序顺序线性表

- 数组中间位置的元素值 k_{mid}
  - 如果 $k_{mid} = k$，那么检索工作就完成了
  - 当 $k_{mid} > k$ 时，检索继续在前半部分进行
  - 相反地，若 $k_{mid} < k$，就可以忽略 mid 以前的那部分，检索继续在后半部分进行
- 快速
  - $k_{mid} = k$ 结束
  - $K_{mid} \neq k$ 起码缩小了一半的检索范围

二分法检索性能分析

- 最大检索长度为
  $$\lceil \log_2(n+1) \rceil$$
- 失败的检索长度是
  $$\lceil \log_2(n+1) \rceil$$
  或
  $$\lfloor \log_2(n+1) \rfloor$$
- 平均检索代价为 $O(\log n)$
- 在算法复杂性分析中
  - $\log n$ 是以 2 为底的对数
  - 以其他数值为底，算法量级不变



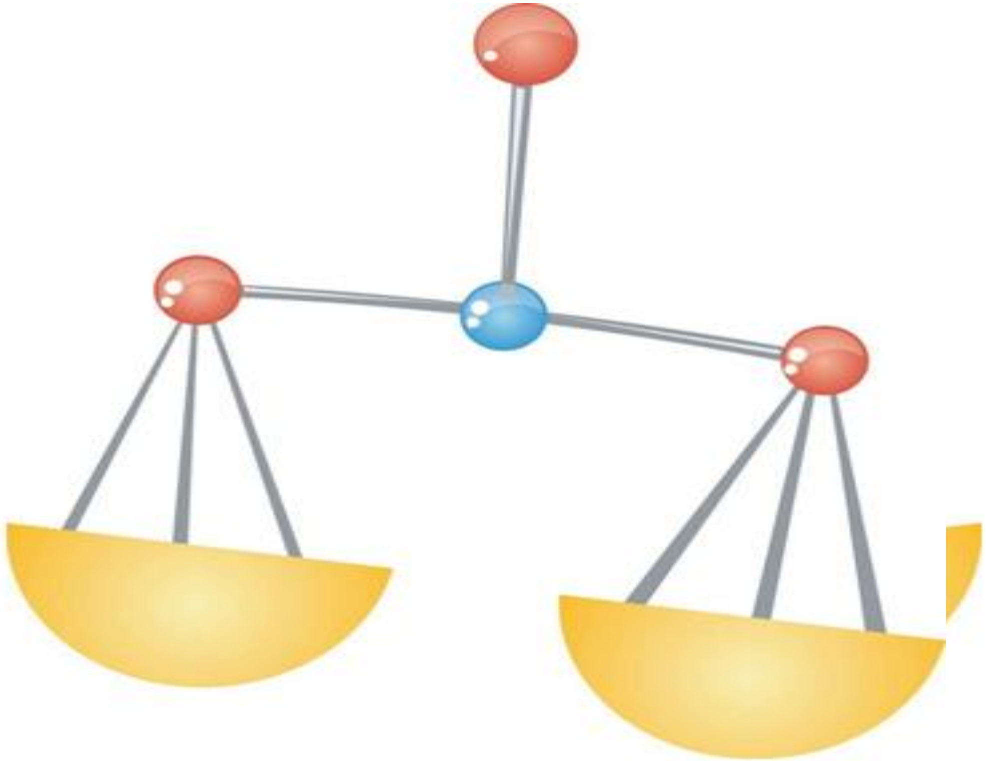
时间/空间权衡

- **数据结构**
  - 一定的空间来存储它的每一个数据项
  - 一定的时间来执行单个基本操作
- **代价和效益**
  - 空间和时间的限制
  - 软件工程

1.4 算法复杂性分析

时空权衡

- 增大空间开销可能改善算法的时间开销
- 可以节省空间，往往需要增大运算时间



数据结构和算法的选择

- 仔细分析所要解决的问题
  - 特别是求解问题所涉及的数据类型和数据间逻辑关系
    —— 问题抽象、数据抽象
  - 数据结构的初步设计往往先于算法设计
- 注意数据结构的可扩展性
  - 考虑当输入数据的规模发生改变时，
    数据结构是否能够适应求解问题的演变和扩展

思考：数据结构和算法的选择

- 问题求解的目标？

- 数据结构与算法选择的过程？

思考：数据结构的三要素

以下哪几种结构是逻辑结构，而与存储和运算无关（ ______ ）。

A. 顺序表　　B. 散列表

C. 线性表　　D. 单链表

下面术语（ ______ ）与数据的存储结构无关。

A. 顺序表　　B. 链表

C. 队列　　D. 循环链表

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕
高等教育出版社，2008. 6。“十一五”国家级规划教材