第8章算法分析与问题的计算复杂度

- 算法的评价指标
- 平凡下界与直接计算最少运算次数
- 决策树

  检索问题的时间复杂度分析

  排序问题的时间复杂度分析
- 构造最坏输入

  选择问题的时间复杂度分析

算法正确性

- 正确性 在给定有效输入后, 算法经过有限时间的计算并产生正确的答案, 就称算法是正确的.
- 正确性证明的内容：
  - 方法的正确性证明——算法思路的正确性. 证明一系列与算法的工作对象有关的引理、定理以及公式.
  - 程序的正确性证明——证明所给出的一系列指令确实做了所要求的工作.

工作量--时间复杂性分析

计量工作量的标准：对于给定问题, 该算法所执行的基本运算的次数.

基本运算的选择：根据问题选择适当的基本运算

问题	基本运算
在表中查找x	比较
实矩阵相乘	实数乘法
排序	比较
遍历二叉树	置指针

两种时间复杂性:

最坏情况下的复杂性$W(n)$

平均情况下的复杂性$A(n)$

占用空间--空间复杂性分析

- 两种占用
  - 存储程序和输入数据的空间
  - 存储中间结果或操作单元所占用空间--额外空间
- 影响空间的主要因素：
  - 存储程序的空间一般是常数(和输入规模无关)
  - 输入数据空间为输入规模 $O(n)$
  - 空间复杂性考虑的是额外空间的大小
- 额外空间相对于输入规模是常数, 称为原地工作的算法.
- 两种空间复杂性:
  - 最坏情况下的复杂性
  - 平均情况下的复杂性.

简单性

- 含义： 算法简单，程序结构简单.
- 好处： 容易验证正确性

  便于程序调试
- 简单的算法效率不一定高. 要在保证一定效率的前提下力求得到简单的算法

基于时间的最优性

- 含义：指求解某问题算法类中效率最高的算法
- 两种最优性

最坏情况下最优：设 *A* 是解某个问题的算法, 如果在解这个问题的算法类中没有其它算法在最坏情况下的时间复杂性比 *A* 在最坏情况下的时间复杂性低, 则称 *A* 是解这个问题在最坏情况下的最优算法.

平均情况下最优：设 *A* 是解某个问题的算法, 如果在解这个问题的算法类中没有其它算法在平均情况下的时间复杂性比 *A* 在平均情况下的时间复杂性低, 则称 *A* 是解这个问题在平均情况下的最优算法

寻找最优算法的途径

(1) 设计算法A, 求$W(n)$，得到算法类最坏情况下时间复杂度的一个上界

(2) 寻找函数$F(n)$, 使得对任何算法都存在一个规模为 n 的输入并且该算法在这个输入下至少要做$F(n)$次基本运算，得到该算法类最坏情况下时间复杂度的一个下界

(3) 如果$W(n)=F(n)$或$W(n)=\Theta(F(n))$, 则A是最优的.

(4) 如果$W(n)>F(n)$, A不是最优的或者$F(n)$的下界过低.

改进A或设计新算法A'使得$W'(n)<W(n)$.

重新证明新下界$F'(n)$使得$F'(n)>F(n)$.

重复以上两步, 最终得到$W'(n) = F'(n)$ 或者$W'(n) = \Theta(F'(n))$

平凡下界

算法的输入规模和输出规模是它的平凡下界

例1

问题：写出所有的n阶置换

求解的时间复杂度下界为$\Omega(n!)$

例2

问题：求n次实系数多项式在给定x的值

求解的时间复杂度下界为$\Omega(n)$

例3

问题：求两个$n \times n$ 矩阵的乘积

求解的时间复杂度下界是$\Omega(n^2)$

直接计数最少运算数

例4 找最大

算法 Findmax

输入 数组L, 项数 $n \geq 1$

输出 L中的最大项MAX

1. $MAX \leftarrow L(1)$; $i \leftarrow 2$;
2. while $i \leq n$ do
3. if $MAX < L(i)$ then $MAX \leftarrow L(i)$;
4. $i \leftarrow i+1$;

$W(n)=n-1$

以比较作为基本运算的算法类的上界：$n-1$

找最大问题的复杂度

下界：在 n 个数的数组中找最大的数, 以比较做基本运算的算法类中的任何算法在最坏情况下至少要做 $n-1$ 次比较.

证 因为*MAX*是唯一的, 其它的 $n-1$ 个数必须在比较后被淘汰. 一次比较至多淘汰一个数, 所以至少需要 $n-1$ 次比较.

结论: Findmax 算法是最优算法.

决策树(Decision Tree)

- 决策树是一棵二叉树，对于给定问题（以比较运算作为基本运算）规定一个决策树的构造规则．求解这个问题的不同算法所构造的决策树结构不一样.
- 给定一个算法的决策树．对于任何输入实例，算法将从树根开始，沿一条路径向下，在每个结点做一次基本操作(比较)．然后根据比较结果(<, =, >)走到某个儿子结点或者在该处停机．对于给定实例的计算恰好对应了一条从树根到树叶或者某个内部结点的路径.
- 给定一个算法，对于不同的输入，算法将在对应决策树的某个结点(树叶或者内部结点)停机．将该结点标记为输入．问题的输入规模对应于决策树的结点总数或者叶结点数.

决策树与问题复杂度

决策树的特点：

- 以比较作基本运算的算法模型
- 一个问题确定了一类决策树，具有相同的构造规则，该决策树类决定了求解该问题的一个算法类
- 结点数(或树叶数)等于输入规模
- 最坏情况下的时间复杂度对应于决策树的深度
- 平均情况下的时间复杂度对应于决策树的平均路径长度

用决策树模型界定确定问题难度

- 给定结点数(或树叶数)的决策树的深度至少是多少？
- 给定结点数(或树叶数)的决策树的平均路径长度至少是多少？

二叉树的性质

命题1 在二叉树的 t 层至多 2^t 个结点

可对 t 进行归纳证明

命题2 深度为 d 的二叉树至多 $2^{d+1}-1$ 个结点

利用命题1的结果

命题3 n个结点的二叉树的深度至少为$\lfloor \log n \rfloor$.

利用命题2

命题4 设 t 为二叉树的树叶个数，d 为树深，如果树的每个内结点都有2个儿子，则 $t \leq 2^d$

对 d 归纳证明

顺序检索的时间复杂度

检索问题：给定按递增顺序排列的数组 L (项数 $n \geq 1$)和数 x，如果 x 在 L 中, 输出 x 的下标; 否则输出0.

算法1 顺序检索

输入：L, x

输出：j

1. $j \leftarrow 1$
2. while $j \leq n$ and $L(j) \neq x$ do $j \leftarrow j+1$
3. if $j > n$ then $j \leftarrow 0$

分析：设 x 在L中每个位置和空隙的概率都是$1/(2n+1)$

$W(n)=n$

$A(n)=[(1+2+...+n)+n(n+1)]/(2n+1) \approx 3n/4.$

二分检索最坏时间复杂度

定理1 $W(n)=\lfloor \log n \rfloor + 1 \qquad n \geq 1$

证 对n归纳

n=1时, 左= $W(1)$=1, 右 =$\lfloor \log 1 \rfloor$+1 = 1.

假设对一切k, $1 \leq k < n$, 命题为真, 则

$$W(n) = 1 + W(\lfloor \frac{n}{2} \rfloor)$$

$$= 1 + \lfloor \log \lfloor \frac{n}{2} \rfloor \rfloor + 1$$

$$= \begin{cases} \lfloor \log n \rfloor + 1 & n\text{为偶数} \\ \lfloor \log(n-1) \rfloor + 1 & n\text{为奇数} \end{cases}$$

$$= \lfloor \log n \rfloor + 1$$

二分检索的平均时间复杂度

令 $n=2^k-1$, S_t 是算法做 t 次比较的输入个数, $1 \le t \le k$

则

$S_1=1=2^0$, $S_2=2=2^1$, $S_3=2^2$, $S_4=2^3$, … ,

$S_t = 2^{t-1}$, $t<k$

$S_k = 2^{k-1} + n + 1$

其中 2^{k-1} 为 x 在表中做 k 次比较的输入个数

$$A(n) = \frac{1}{2n+1}(1S_1 + 2S_2 + \ldots + kS_k)$$

求和

$$A(n)=\frac{1}{2n+1}(1S_1+2S_2+\ldots+kS_k)$$

$$=\frac{1}{2n+1}[\sum_{t=1}^{k}t2^{t-1}+k(n+1)]$$

$$=\frac{1}{2n+1}[(k-1)2^k+1+k(n+1)]$$

$$\approx\frac{k-1}{2}+\frac{k}{2}=k-\frac{1}{2}=\lfloor\log n\rfloor+\frac{1}{2}$$

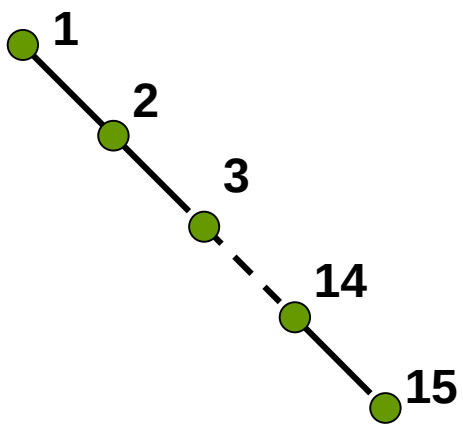
检索问题的决策树

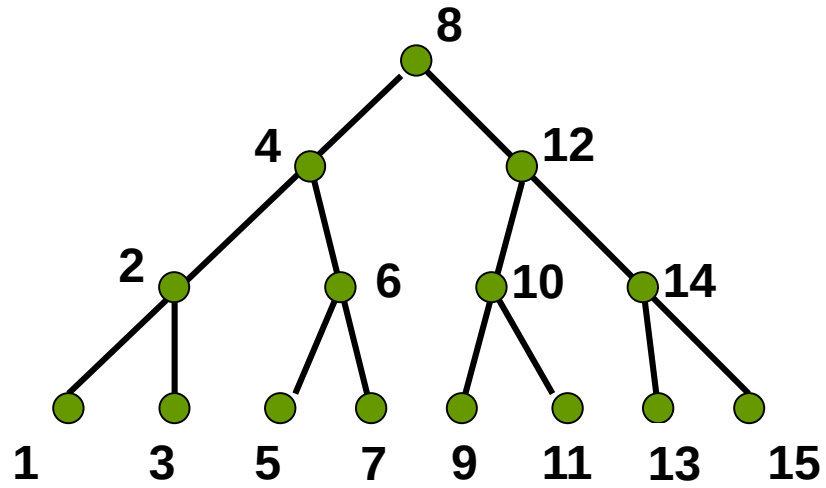
设A是一个检索算法, 对于给定输入规模 n, A的一棵决策树是一棵二叉树, 其结点被标记为1, 2, ... , n, 且标记规则是:

- 根据算法A，首先与 x 比较的 L 的项的下标标记为树根.
- 假设某结点被标记为i,
  - i 的左儿子是：当 $x<L(i)$时，算法A下一步与x比较的项的下标
  - i 的右儿子是：当 $x>L(i)$时，算法A下一步与x比较的项的下标
  - 若 $x<L(i)$ 时算法 A 停止, 则 i 没有左儿子.
  - 若 $x>L(i)$ 时算法 A 停止, 则 i 没有右儿子.

实例

改进顺序检索算法和二分检索算法的决策树，n =15





给定输入, 算法 *A* 将从根开始, 沿一条路径前进, 直到某个结点为止. 所执行的基本运算次数是这条路径的结点个数. 最坏情况下的基本运算次数是树的深度+1.

检索问题的复杂度分析

定理 对于任何一个搜索算法存在某个规模为n 的输入使得该算法至少要做$\lfloor \log n \rfloor +1$ 次比较.

证 由命题3, n 个结点的决策树的深度 d 至少为$\lfloor \log n \rfloor$, 故

$$W(n)=d+1 = \lfloor \log n \rfloor +1.$$

结论: 对于有序表搜索问题, 在以比较作为基本运算的算法类中, 二分法在最坏情况下是最优的.