# Thoughts of Algorithms



随笔 - 54 文章 - 1 评论 - 148

## 【算法16】递归算法的时间复杂度终结篇

开篇前言：为什么写这篇文章？笔者目前在学习各种各样的算法，在这个过程中，频繁地碰到到递归思想和分治思想，惊讶于这两种的思想的伟大与奇妙的同时，经常要面对的一个问题就是，对于一个给定的递归算法或者用分治思想缩小问题规模的算法，如何求解这个算法的时间复杂度呢？在google过很多的博文后，感觉这些博文总结的方法，有很好优秀的地方，但是都不够全面，有感于此，笔者决定总结各家之长，作此博文，总结各种方法于此，有不足之处，欢迎各位批评指证！

在算法的分析中，当一个算法中包含递归调用时，其时间复杂度的分析会转化成为一个递归方程的求解。而对递归方程的求解，方法多种多样，不一而足。本文主要介绍目前主流的方法：代入法，迭代法，公式法，母函数法，差分方程法。

---

【代入法】代入法首先要对这个问题的时间复杂度做出预测，然后将预测带入原来的递归方程，如果没有出现矛盾，则是可能的解，最后用数学归纳法证明。

【举　例】我们有如下的递归问题：T(n)=4T(n/2)+O(n)，我们首先预测时间复杂度为$O(n^2)$,不妨设$T(n)=kn^2$（其中k为常数），将该结果带入方程中可得：左=$kn^2$，右=$4k(n/2)^2+O(n)=kn^2+O(n)$,由于$n^2$的阶高于n的阶，因而左右两边是相等的，接下来用数学归纳法进行验证即可。

---

【迭代法】迭代法就是迭代的展开方程的右边，直到没有可以迭代的项为止，这时通过对右边的和进行估算来估计方程的解。比较适用于分治问题的求解，为方便讨论起见，给出其递归方程的一般形式：

$$T(n) = \begin{cases} O(1) & n = 1 \\ aT(\frac{n}{b}) + f(n) & n > 1 \end{cases}$$

【举　例】下面我们以一个简单的例子来说明：$T(n)=2T(n/2)+n^2$,迭代过程如下：

$$T(n) = 2T(\frac{n}{2}) + n^2$$

$$= n^2 + 2((\frac{n}{2})^2 + 2T(\frac{n}{4}))$$

$$= n^2 + 2((\frac{n}{2})^2 + 2((\frac{n}{4})^2 + 2T(\frac{n}{8})))$$

$$= n^2 + 2((\frac{n}{2})^2 + 2((\frac{n}{4})^2 + \ldots + 2((\frac{n}{2^i})^2 + T(\frac{n}{2^{i+1}}))))$$

容易知道，直到n/2^(i+1)=1时，递归过程结束，这时我们计算如下：

$$T(n) = n^2 + 2\frac{n^2}{2^2} + 2^2\frac{n^2}{4^2} + 2^3\frac{n^2}{8^2} + \ldots$$

$$= n^2(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots)$$

$$= 2n^2$$

到这里我们知道该算法的时间复杂度为$O(n^2)$，上面的计算中，我们可以直接使用无穷等比数列的公式，不用考虑项数i的约束，实际上这两种方法计算的结果是完全等价的，有兴趣的同学可以自行验证。

---

【公式法】这个方法针对形如：T(n) = aT(n/b) + f(n)的递归方程。这种递归方程是分治法的时间复杂性所满足的递归关系，即一个规模为n的问题被分成规模均为n/b的a个子问题，递归地求解这a个子问题，然后通过对这a个子问题的解的综合，得到原问题的解。这种方法是对于分治问题最好的解法，我们先给出如下的公式：

$$T(n)=\begin{cases} O(n^{\log_b a}) & O(n^{\log_b a}) > O(f(n)) \\ O(f(n)*\log n) & O(n^{\log_b a}) = O(f(n)) \\ O(f(n)) & O(n^{\log_b a}) < O(f(n)) \end{cases}$$

(其中a>1, b>1均为常数，f(n)是确定正函数)

公式记忆：我们实际上是比较n^$\log_b$a和f(n)的阶，如果他们不等，那么T(n)取他们中的较大者，如果他们的阶相等，那么我们就将他们的任意一个乘以logn就可以了。按照这个公式，我们可以计算【迭代法】中提到的例子：$O(f(n))=O(n^2)$,容易计算另外一个的阶是O(n),他们不等，所以取较大的阶$O(n^2)$。太简单了，不是吗?

需要注意：上面的公式并不包含所有的情况，比如第一种和第二种情况之间并不包含下面这种情况：f(n)是小于前者，但是并不是多项式的小于前者。同样后两种的情况也并不包含所有的情况。为了好理解与运用的情况下，笔者将公式表述成如上的情况，但是并不是很严谨，关于该公式的严密讨论，请看 这里 。但是公式的不包含的情况，我们很少很少碰到，上面的公式适用范围很广泛的。

特别地，对于我们经常碰到的，当f(n)=0时，我们有：

$$T(n)=\begin{cases} O(n^{\log_b a}) & a \neq 1 \\ O(\log n) & a = 1 \end{cases}$$

---

【母函数法】母函数是用于对应于一个无穷序列的幂级数。这里我们解决的递归问题是形如：$T(n)=c_1T(n-1)+c_2T(n-2)+c_3T(n-3)+...+c_kT(n-k)+f(n)$。为说明问题简便起见，我们选择斐波那契数列的时间复杂度作为例子进行讨论。

【举　例】斐波那契数列递归公式：T(n)=T(n-1)+T(n-2)。这里我们假设F(n)为第n项的运算量。则容易得到：F(n)=F(n-1)+F(n-2),其中F(1)=F(2)=1.我们构造如下的母函数：$G(x)=F(1)x+F(2)x^2+F(3)x^3+.....$,我们可以推导如下：

$$F_3x^3 = F_2x^3 + F_1x^3$$
$$F_4x^4 = F_3x^4 + F_2x^4$$
$$\cdots\cdots$$
$$\Rightarrow G(x) - F_1x - F_2x^2 = x(G(x) - F_1x) + x^2G(x)$$
$$\because F_1 = F_2 = 1$$
$$\therefore G(x) - x - x^2 = x(G(x) - x) + x^2G(x)$$
$$\Rightarrow G(x) = \frac{x}{1 - x - x^2} = \frac{1}{\sqrt{5}}[\frac{1}{1 - \frac{1+\sqrt{5}}{2}x} + \frac{1}{1 - \frac{1-\sqrt{5}}{2}x}]$$
$$= \frac{1}{\sqrt{5}}[(\alpha - \beta)x + (\alpha^2 - \beta^2)x^2 + \ldots]$$

只保留一次项$\alpha-\beta$，令x=1可得$\alpha$，$\beta$的值，从而计算得

$$F_n = \alpha^n - \beta^n = \frac{1}{\sqrt{5}}(\frac{\sqrt{5}+1}{2})^n$$
$$\therefore T(n) = O((\frac{\sqrt{5}+1}{2})^n)$$

上面的方法计算相对来说是比较简单的，关键在于对于母函数的理解，刚开始的时候可能不是很好理解，对于母函数可以参考 这里 和 维基百科这里 。

---

【差分方程法】可以将某些递归方程看成差分方程，通过解差分方程的方法来解递归方程，然后对解作出渐近阶估计。这里我们只考虑最长常见的递归形式，形如：$T(n)=c_1T(n-1)+c_2T(n-2)+c_3T(n-3)+\ldots+c_kT(n-k)+f(n)$，其中c1,c2,...ck为常数且不等于0；我们对该方程的求解如下：

$$T(n) - c_1T(n-1) - c_2T(n-2) - \ldots - c_kT(n-k) = f(n)$$

求对应齐次方程的通解：

$$T(n) - c_1T(n-1) - c_2T(n-2) - \ldots - c_kT(n-k) = 0$$

对应上面的齐次方程的特征方程为：

$$c(t) = t^k - c_1t^{k-1} - c_2t^{k-2} - \ldots - c_k = 0$$

如果解得t=r是该特征方程的m重根，则这m个解的形式为：$\{r^n \quad n*r^n \quad n^2r^n \quad \ldots \quad n^{m-1}r^n\}$,其余的关于复数解的形式和普通的线性方程组的形式类似，不再赘述。接下来，我们要求解该方程的对应非齐次方程组的通解，这里我们针对该方程的特殊形式，不加证明地给出如下的通解形式：

| f(n)的形式 | 条　件 | 特解的形式 |
|---|---|---|
| $a^n$ | C(a)≠0 | $p_0a^n$ |
| | a是C(t)的m重根 | $p_0 \cdot n^m \cdot a^n$ |
| $n^s$ | C(1)≠0 | $p_0 + p_1 \cdot n + p_2 \cdot n^2 + \cdots + p_s \cdot n^s$ |
| | 1是C(t)的m重根 | $n^m \cdot (p_0 + p_1 \cdot n + p_2 \cdot n^2 + \cdots + p_s \cdot n^s)$ |
| $n^sa^n$ | C(a)≠0 | $(p_0 + p_1 \cdot n + p_2 \cdot n^2 + \cdots + p_s \cdot n^s) \cdot a^n$ |
| | a是C(t)的m重根 | $n^m \cdot (p_0 + p_1 \cdot n + p_2 \cdot n^2 + \cdots + p_s \cdot n^s) \cdot a^n$ |

则和线性代数中的解一样，原方程的解等于齐次方程组的通解+特解，即：

$$T(n) = \sum_{i=0}^{k-1} a_i T_i(n) + g(n)$$

其中{$T_i(n)$, i=0, 1, 2, ... n}是基础解系，g(n)是一个特解。

最后由初始条件确定a(i)的值即可。

为了帮助理解，我们举两个例子看看，就明白是怎么回事了。

【举 例1】递归方程如下：

$$F(n) = \begin{cases} 2 & n = 0 \\ 3 & n = 1 \\ 8 + F(n-1) + F(n-2) & n > 1 \end{cases}$$

(1)写出对应齐次方程的特征方程：

$$c(t) = t^2 - t - 1 = 0$$

$$t1 = \frac{1-\sqrt{5}}{2}, t2 = \frac{1+\sqrt{5}}{2}$$

得到基础解系为：{$t_1^n$, $t_2^n$}

(2)计算特解，对于本题，直接观察得特解为：-8

(3)得到原方程解的形式为：$T(n)=a_0t_1^n+a_1t_2^n-8$

(4)代入n=0，n=1的情况，得到a0,a1，最后可得：

$$T(n) = (5 - \frac{6\sqrt{5}}{5})(\frac{1-\sqrt{5}}{2})^n + (5 + \frac{6\sqrt{5}}{5})(\frac{1+\sqrt{5}}{2})^n$$

可以看到该方程形式和上面讨论过的斐波那契数列只差一个常数8，因而两者的时间复杂度是相同的。有兴趣的同学可以按照这个方法再次计算斐波那契数列的时间复杂度验证一下。

【举 例2】递归方程如下：

$$T(n) = 4T(n-1) + 4T(n-2) + n * 2^n$$

其中$T(0) = 0, T(1) = 4/3$

（1）计算对应齐次方程的基础解析：

特征方程为：C(t)=t^2-4t-4=0,得到一个2重根t=2.因而其基础解为：{$2^n$　　$n*2^n$}

（2）由于$f(n)=n*2^n$，对应上面表格的最后一种情况，得到特解形式为：
$T(n)=n^2(p_0+p_1n)2^n$代入原递归方程可得：p0=1/2,p1=1/6

（3）原方程解的形式为：$T(n)=a_0*2^n+a_1*n*2^n+n^2(1/2+n/6)2^n$,代入T(0),T(1)得：
a0=a1=0

（4）综上：$T(n)=n^2(1/2+n/6)2^n$

因而时间复杂度为：$O(n^32^n)$

---

References:

[1]青青的专栏：http://blog.csdn.net/metasearch/article/details/4428865

[2]心灵深处博客：http://blog.csdn.net/metasearch/article/details/4428865

[3]wikipedia中文：母函数

[4]母函数的性质和应用：http://www.doc88.com/p-39037791334.html

[5]关于递归算法时间复杂度分析的讨论：
http://wenku.baidu.com/view/719b053331126edb6f1a1091.html

[6][置顶]递归方程组解的渐进阶的求法一一差分方程法：
http://blog.csdn.net/explore_knight/article/details/1788046

注：

1）本博客所有的代码环境编译均为win7+VC6。所有代码均经过博主上机调试。

2）博主python27对本博客文章享有版权，网络转载请注明出处
http://www.cnblogs.com/python27/ 。对解题思路有任何建议，欢迎在评论中告知。

分类：每天一算法

标签：算法



python27
关注 - 3
粉丝 - 271
+加关注

12 1

« 上一篇：【算法15】字符串的全排列
» 下一篇：【算法17】把数组排成最小的数

posted @ 2011-12-09 23:13 python27 阅读(39457) 评论(11) 编辑 收藏

评论列表

#1楼 2011-12-10 00:58 好坏
这个必须顶了 感谢楼主
支持(0) 反对(0)

#2楼 2011-12-10 08:46 windx
果断顶起，感谢楼主博文！
支持(0) 反对(0)

#3楼 2014-07-14 11:32 阿水
必须顶一记，感谢楼主！
支持(0) 反对(0)

#4楼 2014-08-28 15:10 Alexia(minmin)
比如阶乘T(n)=n*T(n-1)，很容易推出T(n)=n!，按你这么说这个算法的时间复杂度就是O(n!)？实际上它是O(n)啊，这怎么破
支持(0) 反对(1)

#5楼 2015-11-23 08:44 阿水
MARK
支持(0) 反对(0)

#6楼 2016-09-08 21:40 bruce.e.zhao
楼主你好我转载了你的文章，注明了出处，谢谢。

支持(0) 反对(0)

#7楼 2016-11-24 11:34 阿水

顶一记!

支持(0) 反对(0)

#8楼 2017-01-08 16:31 琴鸟

@ Alexia(minmin)
谁说T(n)=n*T(n-1)?
伪代码是f(n)=f(n)*f(n-1)
而复杂度是 T(N)=T(N-1)+1.

支持(0) 反对(0)

#9楼 2017-05-11 13:07 wo4wangle

@ 琴鸟
T(N)=T(N-1)+1 为什么要加一啊

```
int fact（int n）
{if（n＜=1）return 1；
return n*fact（n-1）；}
```

当n>1时候
只执行 n*fact（n-1）没有其他语句了啊
应该是 T(N)=T(N-1) 吧 虽然很荒唐，但是不懂

比如

```
void move(int n, char x, char y, char z)
{
if( 1 == n )
{
printf("%c-->%c\n", x, z);
}
else
{
move(n-1, x, z, y); // 将 n-1 个盘子从 x 借助 z 移到 y 上
printf("%c-->%c\n", x, z); // 将 第 n 个盘子从 x 移到 z 上
move(n-1, y, x, z); // 将 n-1 个盘子从 y 借助 x 移到 z 上
}
}
```

汉诺塔的是 T(N)=2T(N-1)+1
这里的 +1是因为 多了一句 printf("%c-->%c\n", x, z);

支持(0) 反对(0)

#10楼 2017-06-14 18:02 nicholem

楼主你好我转载了你的文章，注明了出处，谢谢。

支持(0) 反对(0)

#11楼 2018-10-19 16:34 _Ljj

@ wo4wangle

```
tmp = f(n-1); // T(n-1)
result = n * tmp; // n很小的时候是 O(1), n很大的时候可能是 O(lg n)
```

T(n) = T(n-1) + O(1)

支持(0) 反对(0)

**相关博文:**
· 递归算法时间复杂度
· 算法时间复杂度分析方法
· 递归算法的时间复杂度分析
· 递归算法的时间复杂度分析
· 递归算法的时间复杂度分析
» 更多推荐...

**最新 IT 新闻**:
· 直接投资20亿，12万台服务器！相隔两天，百度又一个数据中心开工了
· 【物理照片】十张伟大的科学瞬间
· 三星：7nm EUV工艺Q4季度量产 5nm已获得订单
· 你无法看见，但它总在那
· 冲绳首里城大部分建筑被烧毁
» 更多新闻...