

《算法分析与设计》

课程设计报告

学院（系）： 软件工程系

班级： 116030801

学生姓名： 周翔辉

学号： 11603080122

指导老师： 刘祥

时间：从 2018 年 12 月 17 日到 2018 年 12 月 27 日

目录

1	基本的递归算法	1
1.1	二项式的计算	1
1.1.1	问题描述	1
1.1.2	解决问题所用的算法设计方法及基本思路	1
1.1.3	采用的数据结构描述	1
1.1.4	算法描述	1
1.1.5	算法的时间空间复杂度分析	3
1.1.6	算法实例	4
1.2	绘制简单的分形树	5
1.2.1	问题描述	5
1.2.2	解决问题所用的算法设计方法及基本思想	5
1.2.3	采用的数据结构描述	5
1.2.4	算法描述	5
1.2.5	算法的时间空间复杂度分析	5
1.2.6	算法实例	5
2	遍历	6
2.1	8 品脱问题	6
2.1.1	问题描述	6
2.1.2	解决问题所用的算法设计方法及基本思想	6
2.1.3	采用的数据结构描述	6
2.1.4	算法描述	6
2.1.5	算法的时间空间复杂度分析	6
2.1.6	算法实例	6
2.2	24 点问题	7
2.2.1	问题描述	7
2.2.2	解决问题所用的算法设计方法及基本思想	7
2.2.3	采用的数据结构描述	7
2.2.4	算法描述	7
2.2.5	算法的时间空间复杂度分析	7
2.2.6	算法实例	7
3	动态规划	8
3.1	最长回文子序列问题	8
3.1.1	问题描述	8
3.1.2	解决问题所用的算法设计方法及基本思想	8
3.1.3	采用的数据结构描述	8

3.1.4	算法描述	8
3.1.5	算法的时间空间复杂度分析	8
3.1.6	算法实例	8
3.2	小美购物问题	9
3.2.1	问题描述	9
3.2.2	解决问题所用的算法设计方法及基本思想	9
3.2.3	采用的数据结构描述	9
3.2.4	算法描述	9
3.2.5	算法的时间空间复杂度分析	9
3.2.6	算法实例	9
4	分支限界与回溯	10
4.1	n 个处理机和 k 个任务问题	10
4.1.1	问题描述	10
4.1.2	解决问题所用的算法设计方法及基本思想	10
4.1.3	采用的数据结构描述	10
4.1.4	算法描述	10
4.1.5	算法的时间空间复杂度分析	10
4.1.6	算法实例	10
5	附加题目	11
5.1	学习超市选址问题	11
5.1.1	问题描述	11
5.1.2	解决问题所用的算法设计方法及基本思想	11
5.1.3	采用的数据结构描述	11
5.1.4	算法描述	11
5.1.5	算法的时间空间复杂度分析	11
5.1.6	算法实例	11
6	课程设计总结	11

1 基本的递归算法

1.1 二项式的计算

1.1.1 问题描述

完成二项式公式计算, 即 $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$ 公式解释为了从 n 个不同元素中抓取 k 个元素 (C_n^k), 可以这样考虑, 如果第一个元素一定在结果中, 那么就需要从剩下的 $n-1$ 个元素中抓取 $k-1$ 个元素 (C_{n-1}^{k-1}); 如果第一个元素不在结果中, 就需要从剩下的 $n-1$ 个元素中抓取 k 个元素 (C_{n-1}^k)。要求分别采用以下方法计算, 并进行三种方法所需时间的经验分析。

1.1.2 解决问题所用的算法设计方法及基本思路

此问题可以通过下面的算法实现:

1. **递归算法** 考虑计算 C_n^k 的情况, $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$, 则可以递归的调用此方法或者函数, 直到 n 和 k 有一个为 1 就返回 1, $k=1$ 时就返回 n 。
2. **备忘录方法** 需要借助上面的**递归算法**, 当 C_{n-i}^{k-j} 不等于 0 时, 就计算这个值, 然后保存到一个数组中, 其它公式如果需要它的值则直接从数组中调用即可, 不需要再次计算。
3. **迭代算法** 通过一个结果二维数组保存从 C_1^1 到 C_n^k 的所有值, 从小到大计算, 逐行填表。

1.1.3 采用的数据结构描述

在**递归算法**中, 系统底层采用栈存储递归的数据, 然后逐层返回; 在**备忘录方法**中采用了一个额外的二维数组来保存中间过程的值; 在**迭代算法**中, 也是通过一个二维数组实现逐行填表, 计算二项式公式的值。

1.1.4 算法描述

1. 递归算法

算法 CalculateBinomialRecursion (k, n)

```
// 计算二项式公式使用递归
// 输入: 二项式公式的  $k, n$ 
// 输出: 二项式公式的值
if  $k > 0 \ \&\& \ n > 0$ 
    if  $k = 1$  and  $n = 1$ 
        return 1
    else if  $k = 1$ 
        return n
    else
```

```

        return CalculateBinomialRecursion ( $k - 1, n - 1$ )
        + CalculateBinomialRecursion ( $k, n - 1$ )
    return 0

```

2. 备忘录算法

算法 **CalculateBinomialMemo** (k, n)

```

// 计算二项式公式使用备忘录方法
// 输入: 二项式公式的  $k, n$ 
// 输出: 二项式公式的值
// 注意:  $C[n, k]$  用于保存中间过程的值, 减少重复计算
if  $k > 0$  &&  $n > 0$ 
    if  $k = 1$  and  $n = 1$ 
        return 1
    else if  $k = 1$ 
        return n
    else
         $c[k, n] = \text{CalculateBinomialRecursion} (k - 1, n - 1)$ 
        +  $\text{CalculateBinomialRecursion} (k, n - 1)$ 
    return  $c[k, n]$ 
return 0

```

3. 迭代算法

算法 **CalculateBinomialIteration** (k, n)

```

// 计算二项式公式使用迭代方法
// 输入: 二项式公式的  $k, n$ 
// 输出: 二项式公式的值
kns array[][]
for  $i = 0$  to  $n$ 
    for  $j = 0$  to  $k$ 
        if  $i < j$ 
             $kns[i][j] = 0$ 
        else if  $j == 1$ 
             $kns[i][j] = i$ 
        else if  $j == j$ 
             $kns[i][j] = 1$ 
        else
             $kns[i][j] = kns[i - 1][j - 2] + kns[i - 1][j - 1]$ 
return  $kns[n-1][k-1]$ 

```

1.1.5 算法的时间空间复杂度分析

下面是几种算法的运行时间的时间和 k 曲线图

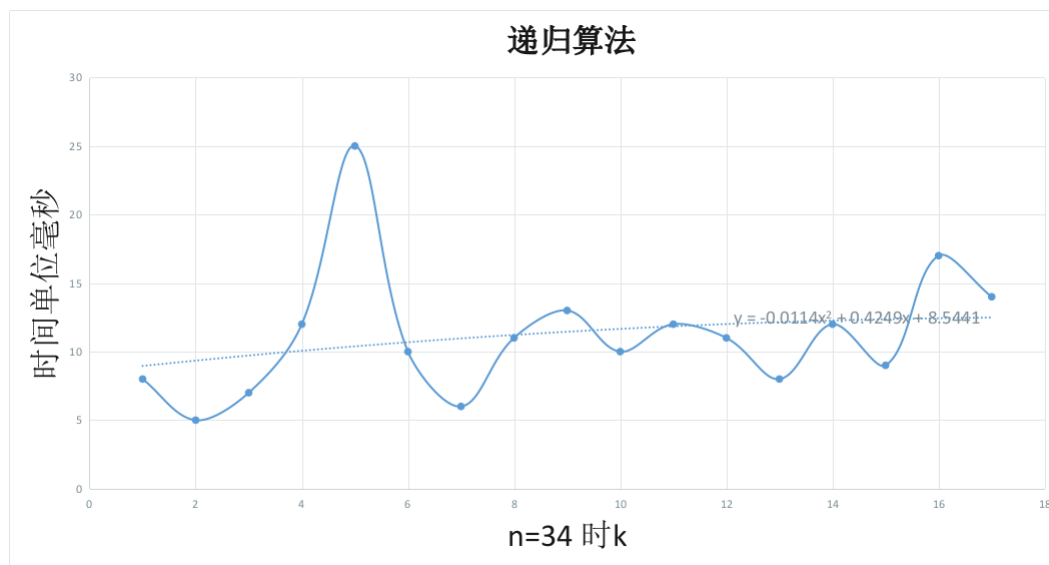


图 1: 递归算法

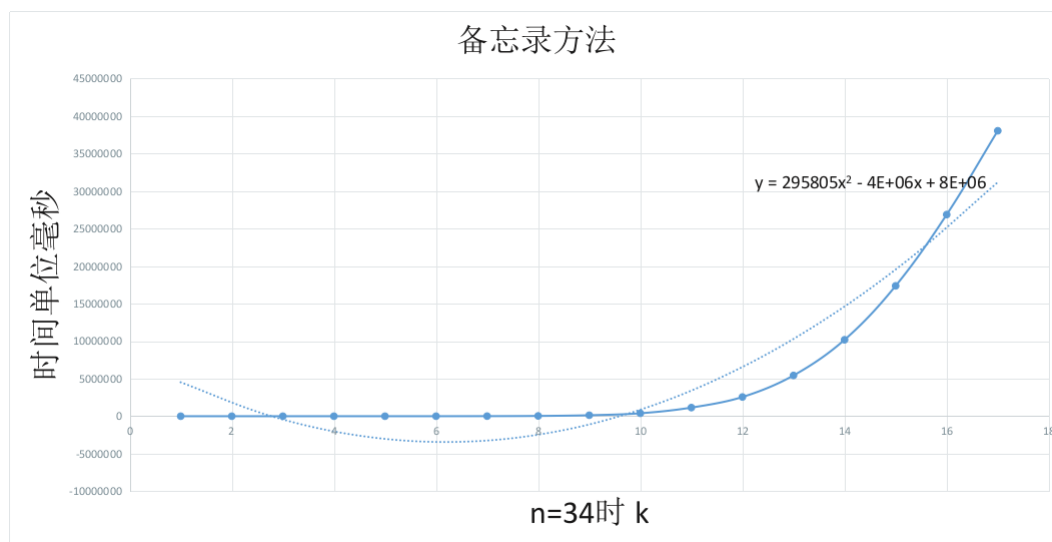


图 2: 备忘录方法

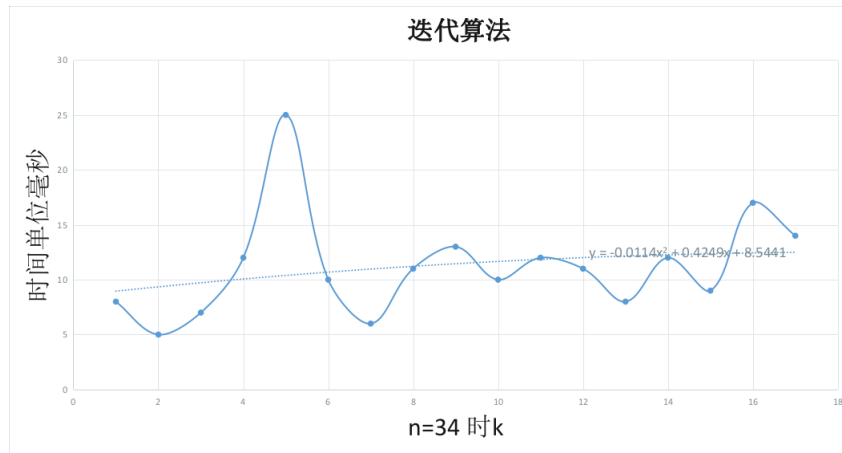


图 3: 迭代算法

通过分析，可以得出递归算法的时间复杂度 $\Theta(n) = n^2$ ，空间复杂度为 n^2 ；备忘录方法的时间复杂度 $\Theta(n) = n^2$ 空间复杂度为 n^2 ；迭代算法的时间复杂度 $\Theta(n) = n^2$ ，空间复杂度为 n^2 。

1.1.6 算法实例

输入： $n = 16, k = 7$

输出：

```

=== RUN    TestCalculateBinomialRecursion
2018/12/25 16:23:54 C(7, 16)=CalculateBinomialBrute=11440, CalculateBinomialRecursion=11440
--- PASS: TestCalculateBinomialRecursion (0.47s)
=== RUN    TestCalculateBinomialMemo
2018/12/25 16:23:54 C(7, 16)=CalculateBinomialBrute=11440, CalculateBinomialMemo=11440
--- PASS: TestCalculateBinomialMemo (0.00s)
=== RUN    TestCalculateBinomialIteration
2018/12/25 16:23:54 C(7, 16)=CalculateBinomialBrute=11440, CalculateBinomialIteration =11440
--- PASS: TestCalculateBinomialIteration (0.00s)

```

图 4: 测试计算二项式的算法

1.2 绘制简单的分形树

1.2.1 问题描述

1.2.2 解决问题所用的算法设计方法及基本思想

1.2.3 采用的数据结构描述

1.2.4 算法描述

1.2.5 算法的时间空间复杂度分析

1.2.6 算法实例

2 遍历

2.1 8 品脱问题

2.1.1 问题描述

2.1.2 解决问题所用的算法设计方法及基本思想

2.1.3 采用的数据结构描述

2.1.4 算法描述

2.1.5 算法的时间空间复杂度分析

2.1.6 算法实例

2.2 24 点问题

2.2.1 问题描述

2.2.2 解决问题所用的算法设计方法及基本思想

2.2.3 采用的数据结构描述

2.2.4 算法描述

2.2.5 算法的时间空间复杂度分析

2.2.6 算法实例

3 动态规划

3.1 最长回文子序列问题

3.1.1 问题描述

3.1.2 解决问题所用的算法设计方法及基本思想

3.1.3 采用的数据结构描述

3.1.4 算法描述

3.1.5 算法的时间空间复杂度分析

3.1.6 算法实例

3.2 小美购物问题

3.2.1 问题描述

3.2.2 解决问题所用的算法设计方法及基本思想

3.2.3 采用的数据结构描述

3.2.4 算法描述

3.2.5 算法的时间空间复杂度分析

3.2.6 算法实例

4 分支限界与回溯

4.1 n 个处理机和 k 个任务问题

4.1.1 问题描述

4.1.2 解决问题所用的算法设计方法及基本思想

4.1.3 采用的数据结构描述

4.1.4 算法描述

4.1.5 算法的时间空间复杂度分析

4.1.6 算法实例

5 附加题目

5.1 学习超市选址问题

5.1.1 问题描述

5.1.2 解决问题所用的算法设计方法及基本思想

5.1.3 采用的数据结构描述

5.1.4 算法描述

5.1.5 算法的时间空间复杂度分析

5.1.6 算法实例

6 课程设计总结