第2-14章习题

Homeworks(1)

习题15

- 分析以下函数执行的乘法次数,该函数计算两个nxn矩阵的乘法.

```
template <class T>
void Mult(T **a, T **b, T **c, int n)
{//Multiply the nxn matrices a and b to get c.
   for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++) {
      T sum = 0;
      for (int k = 0; k < n; k++)
        sum += a[i][k] * b[k][j];
      c[i][j] = sum;
      }
}
```

习题16

- 分析以下函数执行的乘法次数,该函数计算一个mxn矩阵和nxp矩阵的乘法.

```
template <class T>
void Mult(T **a, T **b, T **c, int m,
          int n, int p)
{// Multiply the m x n matrix a and the
// n x p matrix b to get c.
   for (int i = 0; i < m; i++)
    for (int j = 0; j < p; j++) {
      T sum = 0;
      for (int k = 0; k < n; k++)
        sum += a[i][k] * b[k][j];
      c[i][j] = sum;
      }
}
```

习题18

- 函数MinMax找出数组a[0:n-1]中的最大和最小元素,试分析算法所用的元素间比较次数.

```
template <class T>
bool MinMax(T a[], int n, int & Min, int & Max)
{// Locate min and max elements in a[0:n-1].
 // Return false if less than one element.
   if (n < 1) return false;
   Min = Max = 0; // initial guess
   for (int i = 1; i < n; i++) {
     if (a[Min] > a[i]) Min = i;
     if (a[Max] < a[i]) Max = i;
    }
   return true;
}
```

习题18(续)

- 以下算法也计算a[0:n]中最大最小元素,试分析该算法在最好和最坏情形使用的元素比较次数

```
template <class T>
bool MinMax(T a[], int n, int & Min, int & Max)
{// Locate min and max elements in a[0:n-1].
 // Return false if less than one element.
   if (n < 1) return false;
   Min = Max = 0; // initial guess
   for (int i = 1; i < n; i++)
     if (a[Min] > a[i]) Min = i;
     else if (a[Max] < a[i]) Max = i;
   return true;
}
```

习题20

- 以下是另一个顺序查找算法,试分析其最坏情形元素比较次数并和课堂上讲的算法进行比较.

```
template <class T>
int SequentialSearch(T a[], const T& x, int n)
{// Search the unordered list a[0:n-1] for x.
 // Return position if found; return -1 otherwise.
   a[n] = x; // assume extra position available
   int i;
   for (i = 0; a[i] != x; i++);
   if (i == n) return -1;
   return i;
}
```

习题37: 使用计算步数的方法分析下面算法的渐近复杂度

(1)

```
template <class T>
bool MinMax(T a[], int n, int & Min, int & Max)
{// Locate min and max elements in a[0:n-1].
 // Return false if less than one element.
   if (n < 1) return false;
   Min = Max = 0; // initial guess
   for (int i = 1; i < n; i++) {
     if (a[Min] > a[i]) Min = i;
     if (a[Max] < a[i]) Max = i;
    }
   return true;
}
```

(2)

```
template <class T>
void Mult(T **a, T **b, T **c, int n)
{//Multiply the nxn matrices a and b to get c.
   for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++) {
      T sum = 0;
      for (int k = 0; k < n; k++)
        sum += a[i][k] * b[k][j];
      c[i][j] = sum;
      }
}
```

(3)

```
template <class T>
void Rank(T a[], int n, int r[])
{// Rank the n elements a[0:n-1].
   for (int i = 0; i < n; i++)
     r[i] = 0; //initialize
   // compare all element pairs
   for (int i = 1; i < n; i++)
    for (int j = 0; j < i; j++)
      if (a[j] <= a[i]) r[i]++;
      else r[j]++;
}
```

(4)

```
template <class T>
bool MinMax(T a[], int n, int & Min, int & Max)
{// Locate min and max elements in a[0:n-1].
 // Return false if less than one element.
   if (n < 1) return false;
   Min = Max = 0; // initial guess
   for (int i = 1; i < n; i++)
     if (a[Min] > a[i]) Min = i;
     else if (a[Max] < a[i]) Max = i;
   return true;
}
```

Homework(2)

- 1.用归纳法证明

$$T(n) = 0 \quad if\ n = 1$$

$$T(n) \le T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + cn \quad for \quad n > 1$$

- 证明 $T(n) \le cn\lceil \log_2 n \rceil$
- 2. 展开递归树:$T(n)=T(0)+T(n-1)+cn$,并做渐近分析
- 3. 展开T(n)=T(0.1n)+T(0.9n)+**Θ**(n)的递归树并计算递归树的深度和T(n)的渐近值.

Homework(2)

- 4. 试用master方法分析以下递归方程

  （a）$T(n)=10T(n/3)+11n$

  （b）$T(n)= 10T(n/3)+11n^5$

  （c） $T(n)= 27T(n/3)+11n^3$

  （d） $T(n)= 64T(n/4)+10n^3\log_2 n$

- 5 .应用master方法求解$T(n)=2T(n/2)+\Theta(n^{1/2})$

第14章 用分治法设计算法

一、计算2个多项式的乘积

考虑两个1 次多项式$A(x)=a_1x+a_0$ 和 $B(x)=b_1x+b_0$, 要计算A和B的乘积C，可先计算以下4个系数乘法：

$m_1=a_1*b_1$

$m_2=a_1*b_0$

$m_3=a_0*b_1$

$m_4=a_0*b_0$

则 $C(x)=m_1x^2+(m_2+m_3)x+m_4$ ，

1、将n-1（$n=2^k$）次多项式P(x)表示为$P(x)=yP_1(x)+P_2(x)$，$y=x^{n/2}$ ；视P(x)为y 的1次多项式P(y)，系数为多项式$P_1(x)$和$P_2(x)$。试利用上述两个1 次多项式相乘的方法，设计一计算两个n-1 次多项式相乘的分治算法，并分析你设计的算法的时间复杂度。

2、试找到一个用3 次系数乘法的两个1 次多项式相乘的方法。

3、用第2题结果设计一计算两个n-1 次多项式相乘的分治算法，并分析你设计的算法的时间复杂度。

计算排列的逆序数

二、给定自然数1, … , n的一个排列，例如，(1, 3, 4, 2, 5)，如果j > i 但j 排在i 的前面则称(j, i)为该排列的一个逆序。在上例中(3, 2)，(4, 2) 为该排列的逆序。该排列总共有2个逆序。试用分治法设计一个计算给定排列的逆序总数的算法，该算法的时间复杂度为Θ(nlog2n)。

提示：可在O(n)时间内算出2 段排好序的子序列之间的逆序数。例如，(1, 2, 4, 8)为排在左边的子序列，(3, 5, 6, 7)为排在右边的子序列，它们之间的逆序数可用一个线性时间的算法算出。