§9.4 FFT算法的高效实现

DFT变换除用于多项式相乘外，还常用于信号处理，例如：医疗用的CT等。在这些应用中，对算法的速度要求很高。本节介绍一个FFT算法高效的非递归的实现，它比前面所讲的FFT算法快一个常数因子。

一．FFT算法的一个非递归的实现

1.提取公共子表达式

在上一节Recursive\_FFT算法的10-13行，计算了两次，可以将此公共子表达式提出来，只算一次。10-13行的FOR循环改为：

FOR k := 0 TO DO

BEGIN

t := ;

:= ;

:= ; （9.12）

:= ;

END;

2.消除递归

递归调用Recursive\_FFT处理系数向量的过程如图9.3所示的树形结构。每个结点把它的系数向量分成两个子向量，如其左、右孩子所示，然后递归调用Recursive\_FFT过程，处理其左、右孩子中的系数向量。

图9.3

要消除递归，初始时，将系数向量中的元素，按该树的叶子出现的顺序排列，然后顺序地将中两两元素组成对进行蝴蝶操作((9.12)所示)。用算出的对元素代替原来中的元素。再将这对元素顺序地两两组成对(每一对共有4个元素，前两个一组，后两个一组)，用这两组元素进行蝴蝶操作，得到个4个元素一组的向量，再代替上一步的向量。再将4个元素一组的子向量顺序地两两组成对处理。依此下去，进行遍扫描后，最后只剩下个元素的一组，即为DFT的结果。

算法的基本结构如下：

1. FOR s := 1 TO DO
2. FOR k := 0 STEP TO DO

将和的两个

个元素的组结合成个元素的组放在中。

在Recursive\_FFT算法中，即而即。而其中的为，。

1. 算法的基本结构

PROCEDURE FFT\_Base ()

BEGIN

1. n := length[]; /\* n为2的幂,如果不是高位补零 \*/
2. FOR s := 1 TO DO

BEGIN

1. m := ;（跳过m个）
2. := ;
3. FOR k := 0 STEP m TO DO

BEGIN

1. := 1;
2. FOR j := 0 TO DO

BEGIN

1. t := ;
2. u := ;
3. := u + t;
4. := ut;
5. :=

END

END

END

END;

4.实现FFT的高效算法

用过程Bit\_Reverse\_Copy(a, A)将向量a中的元素**按树叶的顺序放入**A数组中。FFT算法如下：

PROCEDURE Iterative\_FFT ();

BEGIN

1. Bit\_Reverse\_Copy (, A);
2. n := length[]; /\* n为2的幂 \*/
3. FOR s := 1 TO DO

BEGIN

1. m := ;
2. := ;
3. := 1;
4. FOR j := 0 TO DO

BEGIN

1. FOR k := j STEP m TO DO

BEGIN

1. t := ;
2. u := ;
3. := u + t;
4. := ut;

END;

1. := ;

END;

END;

1. RETURN(A);

END;

5.Bit\_Reverse\_Copy的计算

考查了图9.3叶结点元素的下标值，它们是0, 4, 2, 6, 1, 5, 3, 7。它们是二进制数：000, 001, 010, 011, 100, 101, 110, 111的逆位串的值：000, 100, 010, 110, 001, 101, 011, 111。

这是因为在树的最顶层，下标最低位为0的元素放在左子树(前面)，下标最低位为1的元素放在右子树(后面)，在树的第2层，对每一个子树的根中的元素，下标第2低位为0的元素放在左子树，下标第2低位为1的元素放在右子树，依此下去，最后形成二进制数逆位串的顺序。

设rev(k)为k的位二进制数的逆位串的值。则有

PROCEDURE Bit\_Reverse\_Copy (, A);

BEGIN

1. n := length[];
2. FOR k := 0 TO DO
3. A[rev(k)] :=

END;

6.算法分析：

Bit\_Reverse\_Copy (, A)时间复杂度为，因为共循环n次，每一次把位逆转需时间。

设是算法9-12行执行的次数。则

故算法Iterative\_FFT的时间复杂度为。

第十章 串匹配

在一个文字串中找某个子串就是串匹配问题。这个工作在文本编辑中经常遇到。某些模式匹配问题也可以化为串匹配问题。例如：在DNA序列中找某些特定的模式。

我们用字符数组T[]代表长度为的串。字符数组代表模式(子串)。

§10.1 朴素的串匹配算法

串匹配的最常用和最简单的方法就是，从1到，试图用匹配。

算法如下：

PROCEDURE Naïve\_String\_Matcher (T, P);

BEGIN

1. n := length(T);
2. m := length(P);
3. FOR s :=1 TO DO
4. IF P[] = T[] THEN

WRITE(“Pattern occurs at”, s);

END;

上述算法在最坏情况下的时间复杂性为。

作业22：

1.用FFT算法计算向量(0, 2, 3,1, 4, 5, 7, 9)的DFT变换，然后对上述计算结果，求逆DFT变换。