4-34. 一个n×n的矩阵T是一个等对角矩阵（Toeplitz matrix）当且仅当对于所有的i 和j 有

T (i, j) =T (i-1, j- 1 )，其中i> 1，j> 1。

1) 证明一个等对角矩阵最多有2n- 1个不同的元素。

2) 给出一种映射模式，把一个等对角矩阵映射到一个大小为2n- 1的一维数组中。

3) 采用2) 中的映射模式设计一个C++ 类To e p l i t z，用一个大小为2n-1 的一维数组来存储等对角矩阵。要求给出两个共享成员函数 Store 和R e t r i e v e，这两个函数的时间复杂性应为( 1 )。

4) 编写一个共享成员函数，用来对两个按2) 中所给方式存储的等对角矩阵进行乘法运算，所得到的结果存储在一个二维数组中。该函数的时间复杂性是多少？

1. 证明：由于对所有的i和j都有T(i,j)=T(i-1,j-1),即对角线上的元素均相等，对下三角(包括最长对角线)，有n条等值线，若是这n条等值线的值互不相等，则有n个不同的元素。对上三角(不包括最长对角线)，有n-1条等值线，若这些等值线的值互不相等，则有n-1个不同的元素。对于此矩阵，若是上三角的元素和下三角的元素均不相等，此时矩阵含有的不同元素最多，即最多有2n-1个不同的元素。
2. 映射关系为：  
   按对角线映射，从低对角线到高对角线：map(i,j) = n+j-i-1  
   按行映射：map(i,j) = n+i-j-1 i>j  
    = j-i i<=j  
   按列映射：map(i,j) = i-j i>j  
    = n+j-i-1 i<=j

(3)

template<class T>

class ToeplitzMatrix {

public :

ToeplitzMatrix(int size)

{n = size; t = new T [2\*n-1];}

~ ToeplitzMatrix() {delete [] t;}

ToeplitzMatrix<T>& Store (const T& x, int i, int j);

T Retrieve(int i, int j) const;

private:

int n; // 矩阵维数

T \*t; // 存储等对角矩阵的一维数组

} ;

template<class T>

ToeplitzMatrix<T>& ToeplitzMatrix<T>:: Store(const T& x, int i, int j)

{// 把x存为T ( i , j )

if ( i < 1 || j < 1 || i > n || j > n)

throw OutOfBounds();

t[n+j-i-1] = x;//按对角线映射

}return \*this;

}

template <class T>

T ToeplitzMatrix<T>::Retrieve(int i, int j) const

{// 返回T ( i , j )

if ( i < 1 || j < 1 || i > n || j > n)

throw OutOfBounds();

return t[n+j-i-1]；//按对角线映射

}

(4)

template<class T>

T ToeplitzMatrix<T>:: operator\*(const ToeplitzMatrix<T>& m) const

{// 矩阵乘法，返回二维数组

if (n != m.n) throw SizeMismatch();

T \*w = new T [n][m.n]; // 结果

// 对所有的i和j计算w

for (int i = 0; i < n; i++) {

for (int j = 0; j <= m.n; j++) {

// 计算w 的第一项

T sum = t[n-i] \* m.t[n+j];

// 累加其余项

for (int k = 1; k < n; k++) {

sum += t[n+k-i] \* m.t[n-k+j];

}

w[i][j] = sum; // 保存w ( i , j )

}

}

return w[n][n];

}