4-32. 令A和B是两个n×n下三角矩阵。两个矩阵的下三角区域中的元素总数为 n ( n + 1 )。设计一种存储模式，用以在一个数组d[n+ 1 ] [n]中同时存储这两个下三角区域。（提示：如果把A的下三角区域与BT 的上三角区域进行合并，就可以得到一个 (n+ 1 )×n的矩阵）。对于A和B，分别给出其相应的存储函数和搜索函数。每个函数的复杂性应为Q ( 1 )。

存储模式为：

b11, b21, b31,……….., , bi1,…..,bn1

a11, b22, b32,……….., , bi2,…..,bn2

a21, a22, b33,…………, , bi3,…..,bn3

………………

ai1, ai2, ai3,…………, , aii,bi+1i+1,…..,bni+1

………………..

an1, an2, an3,…………, , ain,…..,ann

a[i][j]： d[i+1][j] i > = j

a[i][j]：0 i<j

b[i][j]：d[j][i] i > = j

b[i][j]：0 i<j

template<class T>

class LowerMatrix {

public:

LowerMatrix(int size)

{n = size; d = new T [n+1][n];}

~LowerMatrix() {delete [][] d;}

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

T Retrieve(int i, int j) const;

private:

int n; // 矩阵维数

T \*d; // 存储下三角矩阵的两维数组

} ;

存储函数：

template<class T>

LowerMatrix<T>& LowerMatrix<T>:: StoreA(const T& x, int i, int j)

{// 把x 存为L ( i , j ) .

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

throw OutOfBounds();

// 当且仅当i ≥ j 时(i,j) 位于下三角

if (i >= j) d[i+1][j] = x;

else if (x != 0) throw MustBeZero();

return \*this;

}

template<class T>

LowerMatrix<T>& LowerMatrix<T>:: StoreB(const T& x, int i, int j)

{// 把x 存为L ( i , j ) .

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

throw OutOfBounds();

// 当且仅当i ≥ j 时(i,j) 位于下三角

if (i >= j) d[j][i] = x;

else if (x != 0) throw MustBeZero();

return \*this;

}

搜索函数：

template <class T>

T LowerMatrix<T>::RetrieveA(int i, int j) const

{ / /返回L ( i , j ) .

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

throw OutOfBounds();

// 当且仅当i ≥ j 时(i,j) 位于下三角

if (i >= j) return d[i+1][j];

else return 0;

}

template <class T>

T LowerMatrix<T>::RetrieveB(int i, int j) const

{ / /返回L ( i , j ) .

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

throw OutOfBounds();

// 当且仅当i ≥ j 时(i,j) 位于下三角

if (i >= j) return d[j][i];

else return 0;

}