华北电力大学

**实 验 报 告**

|

|

**实验名称** 稀疏矩阵

**课程名称**  数据结构综合实践

|

|

专业班级： 信息2202 学生姓名：徐梓乔

学 号： 22022100327 成 绩：

|  |
| --- |
| 一、实验内容  给出一个 稀疏矩阵 , 编写程序可以显示其三元组表。  可对其做转置运算得, 并显示该矩阵和三元组表。  可对其做加法运算，这里要求做, 并显示该矩阵和三元组表。  可对其做乘法运算，这里要求做, 并显示该矩阵和三元组表。  二、实验方法与步骤  using System;  using System.Collections.Generic;  class Program  {  static void Main()  {  // 创建稀疏矩阵  int[,] matrixData = {  {5, 0, 0, 0},  {1, 0, 2, 0},  {0, 0, 0, 0},  {3, 0, 6, 5}  };  SparseMatrix matrix = new SparseMatrix(matrixData);  // 显示原始矩阵和三元组表  Console.WriteLine("稀疏矩阵A");  matrix.PrintMatrix();  Console.WriteLine("对应三元组");  matrix.PrintSparse();  // 转置运算  SparseMatrix transposedMatrix = matrix.Transpose();  Console.WriteLine("A的转置");  transposedMatrix.PrintMatrix();  Console.WriteLine("A的转置对应三元组");  transposedMatrix.PrintSparse();  // 加法运算  SparseMatrix additionMatrix = matrix.Add(transposedMatrix);  Console.WriteLine("A+A’");  additionMatrix.PrintMatrix();  Console.WriteLine("A+A’ 对应三元组");  additionMatrix.PrintSparse();  // 乘法运算  SparseMatrix multiplicationMatrix = matrix.Multiply(transposedMatrix);  Console.WriteLine("A\*A’");  multiplicationMatrix.PrintMatrix();  Console.WriteLine("A\*A’ 对应三元组");  multiplicationMatrix.PrintSparse();  }  }  class SparseMatrix  {  private int[,] matrix;  private List<Triple> triples;  public SparseMatrix(int[,] data)  {  matrix = data;  triples = GenerateSparse();  }  public void PrintMatrix()  {  for (int i = 0; i < matrix.GetLength(0); i++)  {  for (int j = 0; j < matrix.GetLength(1); j++)  {  Console.Write(matrix[i, j] + " ");  }  Console.WriteLine();  }  }  public void PrintSparse()  {  foreach (var triple in triples)  {  Console.WriteLine($"({triple.Row}, {triple.Column}, {triple.Value})");  }  }  public SparseMatrix Transpose()  {  int[,] transposedData = new int[matrix.GetLength(1), matrix.GetLength(0)];  for (int i = 0; i < matrix.GetLength(0); i++)  {  for (int j = 0; j < matrix.GetLength(1); j++)  {  transposedData[j, i] = matrix[i, j];  }  }  return new SparseMatrix(transposedData);  }  public SparseMatrix Add(SparseMatrix other)  {  if (matrix.GetLength(0) != other.matrix.GetLength(0) || matrix.GetLength(1) != other.matrix.GetLength(1))  {  throw new ArgumentException("Matrices must have the same dimensions for addition.");  }  int[,] resultData = new int[matrix.GetLength(0), matrix.GetLength(1)];  for (int i = 0; i < matrix.GetLength(0); i++)  {  for (int j = 0; j < matrix.GetLength(1); j++)  {  resultData[i, j] = matrix[i, j] + other.matrix[i, j];  }  }  return new SparseMatrix(resultData);  }  public SparseMatrix Multiply(SparseMatrix other)  {  if (matrix.GetLength(1) != other.matrix.GetLength(0))  {  throw new ArgumentException("Number of columns in the first matrix must be equal to the number of rows in the second matrix for multiplication.");  }  int[,] resultData = new int[matrix.GetLength(0), other.matrix.GetLength(1)];  for (int i = 0; i < matrix.GetLength(0); i++)  {  for (int j = 0; j < other.matrix.GetLength(1); j++)  {  int sum = 0;  for (int k = 0; k < matrix.GetLength(1); k++)  {  sum += matrix[i, k] \* other.matrix[k, j];  }  resultData[i, j] = sum;  }  }  return new SparseMatrix(resultData);  }  private List<Triple> GenerateSparse()  {  List<Triple> result = new List<Triple>();  for (int i = 0; i < matrix.GetLength(0); i++)  {  for (int j = 0; j < matrix.GetLength(1); j++)  {  if (matrix[i, j] != 0)  {  result.Add(new Triple(i, j, matrix[i, j]));  }  }  }  return result;  }  private class Triple  {  public int Row { get; }  public int Column { get; }  public int Value { get; }  public Triple(int row, int column, int value)  {  Row = row;  Column = column;  Value = value;  }  }  }  三、实验结果    四、结论与总结  结论:能够利用三元组的方式存储和定位稀疏矩阵中的非零元素,并且可以通过系数矩阵的转置、加法运算、乘法运算后得到的新矩阵仍可以用三元组的方式定位和储存稀疏矩阵中的非零元素.  总结: 对于在实际问题中出现的大型的稀疏矩阵，若用常规分配方法在计算机中储存，将会产生大量的内存浪费，而且在访问和操作的时候也会造成大量时间上的浪费，为了解决这一问题，从而产生了多种解决方案。 由于其自身的稀疏特性，通过压缩可以大大节省稀疏矩阵的内存代价。 |
|  |