山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201705130113 | 姓名： 黄瑞哲 | | 班级： 计科17.3 |
| 实验题目：数组和矩阵 | | | |
| 实验学时：2 | | 实验日期： 2018.10.25 | |
| 实验目的：  掌握稀疏矩阵结构的描述及操作的实现 | | | |
| 软件环境：  G++ 8.1.0  Visual Studio Community 2017 | | | |
| 1. 实验内容（题目内容，输入要求，输出要求） 2. 创建稀疏矩阵类，采用行主顺序把稀疏矩阵非0元素映射到一维数组中，提供操作：两个稀疏矩阵相加相乘、以阵列形式输出矩阵、求转置矩阵。 3. 键盘输入矩阵的行数列数，按行输入矩阵的各元素值，建立矩阵 4. 对建立的矩阵执行相加、相乘的操作，输出操作的结果矩阵      1. 数据结构与算法描述 （整体思路描述，所需要的数据结构与算法） 2. 稀疏矩阵内部存储为线性表，线性表的元素为结构体matrixTerm，其中存储了行主映射后的下标和值。线性表中按下标递增存储保持有序。 3. 使用成员函数get(i, j)获取第i行第j列的值，在线性表中查询时使用二分查找，如果找到则返回否则返回0。 4. 矩阵加法为两个线性表的合并，相同下标的值对应相加。 5. 矩阵乘法为 对a中当前非零元素，找出b中与其列数相同的元素，在答案对应位置上加上两数的乘积。 6. 矩阵转置的方法为：根据原矩阵的下标推算出对应元素所在的行列数，然后以行为列、以列为行插入结果矩阵中。 7. 测试结果（测试输入，测试输出，结果分析）   文件输入输出  运行时间为2.535s   1. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径） 2. 方法get(i, j)采用二分，复杂度为(log(非零元素个数)) 3. 计算矩形A+B的复杂度为O(A中非零元素个数+B中非零元素个数) 4. 计算矩阵A\*B的复杂度为O(A中非零元素个数\*B中非零元素个数\*log(结果非零元素个数)) 5. 矩阵转置的复杂度为O(非零元素个数\*log(非零元素个数)) 6. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）   /\*sparseMatrix.h\*/  #ifndef \_\_sparseMatrix\_H\_  #define \_\_sparseMatrix\_H\_  #include <iostream>  #include <stdexcept>  #include <linearList/arrayList.h>  using namespace std;  template<typename T>  struct matrixTerm  {  int index;  T value;  bool operator!=(const matrixTerm<T>& x) const { return !(index == x.index && value == x.value); }  };  template<typename T>  class sparseMatrix  {  public:  sparseMatrix(int theRow, int theCol) :term(theRow\*theCol/5), row(theRow), col(theCol) {};  T get(int, int) const;  void set(int, int, const T&);  void print(ostream&) const;  sparseMatrix<T> transpose() const;  sparseMatrix<T> operator+(const sparseMatrix<T>&) const;  sparseMatrix<T> operator\*(const sparseMatrix<T>&) const;  protected:  int row, col;  arrayList<matrixTerm<T> > term;  };  template<typename T>  T sparseMatrix<T>::get(int theRow, int theCol) const  {  if (theRow < 1 || theRow > row || theCol < 1 || theCol > col) throw out\_of\_range("the index is out of range");  int index = (theRow - 1)\*col + theCol;  int l = 0, r = term.size(), mid;  while (l < r)  {  mid = (l + r) / 2;  if (index > term[mid].index) l = mid + 1;  else r = mid;  }  if (l == term.size()) return 0;  else if (term[l].index == index) return term[l].value;  else return 0;  }  template<typename T>  void sparseMatrix<T>::set(int theRow, int theCol, const T& theValue)  {  if (theRow < 1 || theRow > row || theCol < 1 || theCol > col) throw out\_of\_range("the index is out of range");  int index = (theRow - 1)\*col + theCol;  int l = 0, r = term.size(), mid;  while (l < r)  {  mid = (l + r) / 2;  if (index > term[mid].index) l = mid + 1;  else r = mid;  }  if (l == term.size())  {  if(theValue != 0) term.insert(l, { index, theValue });  }  else if (term[l].index == index)  {  if (theValue == 0) term.erase(l);  else term[l].value = theValue;  }  else  {  if (theValue != 0) term.insert(l, { index, theValue });  }}  template<typename T>  void sparseMatrix<T>::print(ostream& out) const  {  int n = term.size();  int k = 0;  int index;  T value;  for (int i = 1; i <= row; ++i)  {  for (int j = 1; j <= col; ++j)  {  index = (i - 1)\*col + j;  if (k >= n || index < term[k].index) value = 0;  else value = term[k++].value;  out << value << ' ';  }  out << endl;  }  }  template<typename T>  sparseMatrix<T> sparseMatrix<T>::transpose() const  {  sparseMatrix<T> res(col, row);  for (int i = 0; i < this->term.size(); ++i)  {  int index = term[i].index;  int value = term[i].value;  int c = index % col;  int r = index / col + 1;  if (c == 0)  {  c = col;  r--;  }  res.set(c, r, value);  }  return res;  }  template<typename T>  sparseMatrix<T> sparseMatrix<T>::operator+(const sparseMatrix<T>& a) const  {//矩阵加  if (row != a.row || col != a.col) throw logic\_error("No matching of row and column");  sparseMatrix<T> ans(row, col);  matrixTerm<T> tmp;  int i = 0, j = 0;  while (i < term.size() && j < a.term.size())  {  if (term[i].index < a.term[j].index)  tmp = term[i++];  else if (term[i].index > a.term[j].index)  tmp = a.term[j++];  else  {  tmp = { term[i].index, term[i].value + a.term[j].value };  i++; j++;  }  ans.term.push\_back(tmp);  }  while (i < term.size()) ans.term.push\_back(term[i++]);  while (j < a.term.size()) ans.term.push\_back(a.term[j++]);  return ans;  }  template<typename T>  sparseMatrix<T> sparseMatrix<T>::operator\*(const sparseMatrix<T>& a) const  {  if (col != a.row) throw logic\_error("No matching of row and column");  sparseMatrix<T> tmp(row, a.col);  sparseMatrix<T> ans(row, a.col);  arrayList<T> item(a.col + 1);  int lastR = 0;  for (int i = 0; i < term.size(); ++i)  {  int c = term[i].index % col;  int r = term[i].index / col + 1;  if (c == 0)  {  c = col;  r--;  }  for (int j = 0; j < a.term.size(); ++j)  {  int ac = a.term[j].index % a.col;  int ar = a.term[j].index / a.col + 1;  if (ac == 0)  {  ac = a.col;  ar--;  }  if (c == ar) tmp.set(r, ac, tmp.get(r, ac) + term[i].value\*a.term[j].value);  }  }  for (int i = 0; i < tmp.term.size(); ++i)  if (tmp.term[i].value != 0)  ans.term.push\_back(tmp.term[i]);  return ans;  }  #endif //\_\_sparseMatrix\_H\_  /\*pch.h\*/  #ifndef PCH\_H  #define PCH\_H  // TODO: 添加要在此处预编译的标头  #include "sparsematrix.h"  #endif //PCH\_H  /\*main.cpp\*/  #include "pch.h"  #include <iostream>  #include <fstream>  #include <ctime>  using namespace std;  int main()  {  ifstream fin("1.in");  ofstream fout("ans.txt");  int n, m;  int value;  clock\_t start, end;  start = clock();  fin >> n >> m;  sparseMatrix<int> a(n, m);  for(int i=1;i<=n;++i)  for (int j = 1; j <= n; ++j)  {  fin >> value;  a.set(i, j, value);  }  fin >> n >> m;  sparseMatrix<int> b(n, m);  for (int i = 1; i <= n; ++i)  for (int j = 1; j <= n; ++j)  {  fin >> value;  b.set(i, j, value);  }  (a\*b).print(fout);  end = clock();  cout << (double)(end - start) / CLOCKS\_PER\_SEC << endl;  return 0;  } | | | |