数据结构实践课报告

题目：稀疏矩阵运算器

班级：17052317 姓名：於文卓 学号17061833 完成日期：2018年11月19日

1. **需求分析**
2. 问题描述

稀疏矩阵是指那些多数元素为零的矩阵，利用“稀疏”特点进行存储和计算可以大大节省存储空间，提高计算效率。实现一个能进行稀疏矩阵基本运算的运算器

1. 基本要求

以“带行逻辑连接信息”的三元组顺序表表示系数矩阵，实现两个矩阵相加、相减和想乘的运算。稀疏矩阵的输入形式采用三元组表示，而运算结果的矩阵以通常阵列形式列出

1. 测试数据：

　见课本p136

1. **概要设计**
2. 数据结构

|  |
| --- |
| typedef struct  {  int i, j; //行列  ElemType e;  } Triple; //三元组  typedef struct  {  Triple array[MAXSIZE]; //存的都是三元组 非零元素  int mrow, mcol, num;  } TSMatrix; |

1. 基本操作

|  |
| --- |
| void input(TSMatrix \*m) //输入矩阵  void add(TSMatrix \*m1, TSMatrix \*m2, TSMatrix \*res) //矩阵相加  void sub(TSMatrix \*m1, TSMatrix \*m2, TSMatrix \*res) //矩阵相减  void mult(TSMatrix \*m1, TSMatrix \*m2, TSMatrix \*res) //矩阵相乘  void Printf(TSMatrix \*m) |

1. **本程序包含3个模块：**

(1)主程序模块：

int main(){

输入矩阵；

选择操作；

处理命令；

输出结果；

return 0;

}

(2)矩阵存储模块——实现三元组；

（3）运算单元模块——实现运算操作抽象数据类型；

各模块之间的调用关系如下：

主程序模块 三元组模块 运算单元模块

1. **运算单元模块部分实现代码**

|  |
| --- |
| **void add(TSMatrix \*m1, TSMatrix \*m2, TSMatrix \*res)**  **{**  **if (m1->mrow != m2->mrow || m1->mcol != m2->mcol)**  **{**  **printf("两个矩阵的行数或者列数不相等\n");**  **return;**  **}**  **res->mrow = m1->mrow;//矩阵相加 新的矩阵行列和原来的一样**  **res->mcol = m1->mcol;**  **res->num = 0;//一开始没有三元组**  **for (int i = 1; i <= res->mrow; ++i)**  **{**  **for (int j = 1; j <= res->mcol; ++j)//可以看成res是一个新的矩阵，遍历每一个格子**  **{**  **int flag = 0;**  **for (int k = 0; k < m1->num; ++k)//一开始其实是第一个矩阵的copy (这个等号)**  **{**  **if (m1->array[k].i == i && m1->array[k].j == j)//如果当前这个格子的位置 和 存在m1三元组里的位置是一样的 更新下**  **{**  **res->array[res->num].i = i;**  **res->array[res->num].j = j;**  **res->array[res->num].e = m1->array[k].e;//加入一个三元组**  **flag = 1;**  **break;//同一位置只可能有一个元素**  **}**  **}**  **for (int k = 0; k < m2->num; ++k) //m2同样道理**  **{**  **if (m2->array[k].i == i && m2->array[k].j == j)**  **{**  **res->array[res->num].i = i;**  **res->array[res->num].j = j;**  **if (!flag)**  **{**  **res->array[res->num].e = m2->array[k].e; //如果在m1没有赋值**  **}**  **else**  **{**  **res->array[res->num].e += m2->array[k].e;**  **}**  **flag = 1;**  **}**  **}**  **if (flag)**  **{**  **res->num++; //三元组的个数**  **}**  **}**  **}**  **int cnt = 0;**  **for (int i = 0; i < res->num; ++i)**  **{**  **if (res->array[i].e == 0)//如果一正一负加起来变成零了 把这个节点删掉**  **{**  **for (int j = i + 1 ; j < res->num; ++j)**  **{**  **res->array[j - 1] = res->array[j];**  **}**  **cnt ++;**  **}**  **}**  **res->num -= cnt;//现在还有几个**  **}** |

|  |
| --- |
| **void mult(TSMatrix \*m1, TSMatrix \*m2, TSMatrix \*res)//乘法先把三元组转成普通矩阵**  **{**  **if (m1->mcol != m2->mrow)**  **{**  **printf("矩阵列数不等于行数，错误\n");**  **return;**  **}**  **res->mrow = m1->mrow;//行 是 m1的列**  **printf("%d\n", res->mrow);**  **res->mcol = m2->mcol;//列 是 m2的行**  **printf("%d\n", res->mcol);**  **int arr1[m1->mrow][m1->mcol];**  **int arr2[m2->mrow][m2->mcol];**  **int arr3[res->mrow][res->mcol];**  **for (int i = 0; i < m1->mrow; ++i)**  **{**  **for (int j = 0; j < m1->mcol ; ++j)**  **{**  **arr1[i][j] = 0;**  **}**  **}**  **for (int i = 0; i < m2->mrow; ++i)**  **{**  **for (int j = 0; j < m2->mcol ; ++j)**  **{**  **arr2[i][j] = 0;**  **}**  **}**  **for (int i = 0; i < res->mrow; ++i)**  **{**  **for (int j = 0; j < res->mcol ; ++j)**  **{**  **arr3[i][j] = 0;**  **}**  **}**  **for (int i = 0; i <= m1->num; ++i)//这里的等号忘记了 有m1->num 这么多数据**  **{**  **arr1[m1->array[i].i - 1][m1->array[i].j - 1] = m1->array[i].e; //把三元组的值赋给数组 元组里的行列 是从1开始的 默认加了1 记得要减一**  **}**  **for (int i = 0; i <= m2->num; ++i)**  **{**  **arr2[m2->array[i].i - 1][m2->array[i].j - 1] = m2->array[i].e;**  **}**  **for (int i = 0; i < res->mrow; ++i)**  **{**  **for (int j = 0; j < res->mcol; ++j)**  **{**  **for (int k = 0; k < m1->mcol; ++k)**  **{**  **arr3[i][j] += arr1[i][k] \* arr2[k][j]; //矩阵乘法 三层循环 再看看**  **}**  **if (arr3[i][j] != 0)**  **{**  **res->array[res->num].i = i + 1;**  **res->array[res->num].j = j + 1;**  **res->array[res->num].e = arr3[i][j];**  **res->num++;**  **}**  **}**  **}**  **}** |

1. **算法分析和代码调试**
2. **做加法时，要注意如果m1和m2相加为0，要删去这个节点**
3. **做减法等于m1+（-m2），记得乘完-1后要乘回去**
4. **矩阵乘法时，我把三元组转成了数组，方便操作**
5. **用户手册**
6. **本程序运行环境为win10操作系统，执行文件为稀疏矩阵运算器.exe**
7. **用户界面和测试数据如下**

****

****

****

1. **附件源代码**
2. /\*
3. \* @Author: TD21forever
4. \* @Date: 2018-11-19 16:24:17
5. \* @Last Modified by: TD21forever
6. \* @Last Modified time: 2018-11-19 19:54:08
7. \*/
8. #include<stdio.h>
9. #include<stdlib.h>
10. #define MAXSIZE 12500
11. typedef int ElemType;
12. typedef struct
13. {
14. int i, j;
15. ElemType e;
16. } Triple;
17. typedef struct
18. {
19. Triple array[MAXSIZE];//存的都是三元组 非零元素
20. int mrow, mcol, num;
21. } TSMatrix;
22. void input(TSMatrix \*m)
23. {
24. int row, col;
25. printf("你的矩阵一共有几行几列(行，列)\n");
26. scanf("%d %d", &row, &col);
27. m->mrow = row;
28. m->mcol = col;
29. int i, j, e;
30. m->num = 0;
31. printf("请按照行优先输入三元组，-1 -1 -1 结束\n");
32. while(scanf("%d %d %d", &i, &j, &e) == 3)
33. {
34. if (i == -1 && j == -1 && e == -1)
35. {
36. break;
37. }
38. m->array[m->num].i = i;
39. m->array[m->num].j = j;
40. m->array[m->num].e = e;
41. m->num ++;
42. }
43. }
44. void add(TSMatrix \*m1, TSMatrix \*m2, TSMatrix \*res)
45. {
46. if (m1->mrow != m2->mrow || m1->mcol != m2->mcol)
47. {
48. printf("两个矩阵的行数或者列数不相等\n");
49. return;
50. }
51. res->mrow = m1->mrow;//矩阵相加 新的矩阵行列和原来的一样
52. res->mcol = m1->mcol;
53. res->num = 0;//一开始没有三元组
54. for (int i = 1; i <= res->mrow; ++i)
55. {
56. for (int j = 1; j <= res->mcol; ++j)//可以看成res是一个新的矩阵，遍历每一个格子
57. {
58. int flag = 0;
59. for (int k = 0; k < m1->num; ++k)//一开始其实是第一个矩阵的copy (这个等号)
60. {
61. if (m1->array[k].i == i && m1->array[k].j == j)//如果当前这个格子的位置 和 存在m1三元组里的位置是一样的 更新下
62. {
63. res->array[res->num].i = i;
64. res->array[res->num].j = j;
65. res->array[res->num].e = m1->array[k].e;//加入一个三元组
66. flag = 1;
67. break;//同一位置只可能有一个元素
68. }
69. }
70. for (int k = 0; k < m2->num; ++k)
71. {
72. if (m2->array[k].i == i && m2->array[k].j == j)
73. {
74. res->array[res->num].i = i;
75. res->array[res->num].j = j;
76. if (!flag)
77. {
78. res->array[res->num].e = m2->array[k].e;
79. }
80. else
81. {
82. res->array[res->num].e += m2->array[k].e;
83. }
84. flag = 1;
85. }
86. }
87. if (flag)
88. {
89. res->num++;
90. }
91. }
92. }
93. int cnt = 0;
94. for (int i = 0; i < res->num; ++i)
95. {
96. if (res->array[i].e == 0)//如果一正一负加起来变成零了 把这个节点删掉
97. {
98. for (int j = i + 1 ; j < res->num; ++j)
99. {
100. res->array[j - 1] = res->array[j];
101. }
102. cnt ++;
103. }
104. }
105. res->num -= cnt;//现在还有几个
106. }
107. void sub(TSMatrix \*m1, TSMatrix \*m2, TSMatrix \*res)
108. {
109. for (int i = 0; i < m2->num; ++i)
110. {
111. m2->array[i].e \*= -1;//是m2乘-1 不是m1
112. }
113. add(m1, m2, res);
114. for (int i = 0; i < m2->num; ++i)//记得乘回去
115. {
116. m2->array[i].e \*= -1;
117. }
118. }
119. void mult(TSMatrix \*m1, TSMatrix \*m2, TSMatrix \*res)//乘法先把三元组转成普通矩阵
120. {
121. if (m1->mcol != m2->mrow)
122. {
123. printf("矩阵列数不等于行数，错误\n");
124. return;
125. }
126. res->mrow = m1->mrow;//行 是 m1的列
127. printf("%d\n", res->mrow);
128. res->mcol = m2->mcol;//列 是 m2的行
129. printf("%d\n", res->mcol);
130. int arr1[m1->mrow][m1->mcol];
131. int arr2[m2->mrow][m2->mcol];
132. int arr3[res->mrow][res->mcol];
133. for (int i = 0; i < m1->mrow; ++i)
134. {
135. for (int j = 0; j < m1->mcol ; ++j)
136. {
137. arr1[i][j] = 0;
138. }
139. }
140. for (int i = 0; i < m2->mrow; ++i)
141. {
142. for (int j = 0; j < m2->mcol ; ++j)
143. {
144. arr2[i][j] = 0;
145. }
146. }
147. for (int i = 0; i < res->mrow; ++i)
148. {
149. for (int j = 0; j < res->mcol ; ++j)
150. {
151. arr3[i][j] = 0;
152. }
153. }
154. for (int i = 0; i <= m1->num; ++i)//这里的等号忘记了 有m1->num 这么多数据
155. {
156. arr1[m1->array[i].i - 1][m1->array[i].j - 1] = m1->array[i].e; //把三元组的值赋给数组 元组里的行列 是从1开始的 默认加了1 记得要减一
157. }
158. for (int i = 0; i <= m2->num; ++i)
159. {
160. arr2[m2->array[i].i - 1][m2->array[i].j - 1] = m2->array[i].e;
161. }
162. for (int i = 0; i < res->mrow; ++i)
163. {
164. for (int j = 0; j < res->mcol; ++j)
165. {
166. for (int k = 0; k < m1->mcol; ++k)
167. {
168. arr3[i][j] += arr1[i][k] \* arr2[k][j]; //矩阵乘法 三层循环 再看看
169. }
170. if (arr3[i][j] != 0)
171. {
172. res->array[res->num].i = i + 1;
173. res->array[res->num].j = j + 1;
174. res->array[res->num].e = arr3[i][j];
175. res->num++;
176. }
177. }
178. }
179. }
180. void Printf(TSMatrix \*m) //打印矩阵
181. {
182. int a[m->mrow][m->mcol];
183. for(int i = 0; i < m->mrow; i++)
184. for(int j = 0; j < m->mcol; j++)
185. a[i][j] = 0;
186. for(int k = 0; k <= m->num; k++)//这里有等号，数组从0开始 三元组 从1开始
187. a[m->array[k].i - 1][m->array[k].j - 1] = m->array[k].e;
188. //printf("\n\n矩阵为：\n");
189. for(int i = 0; i < m->mrow; i++)
190. {
191. for(int j = 0; j < m->mcol; j++)
192. printf("%-2d ", a[i][j]);
193. printf("\n");
194. }
195. }
196. int main()
197. {
198. printf("/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*稀疏矩阵运算器 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\n\n");
199. printf("功能选择：\n1.矩阵相加\n2.矩阵相减\n3.矩阵相乘\n");
200. int n;
201. printf("选择功能：");
202. scanf("%d", &n);
203. TSMatrix M1;
204. TSMatrix M2;
205. TSMatrix M3;
206. TSMatrix \*m1 = &M1;
207. TSMatrix \*m2 = &M2;
208. TSMatrix \*res = &M3;
209. // 4 5 1 1 4 1 2 -3 1 5 1 2 4 8 3 3 1 4 5 70 -1 -1 -1 5 3 1 1 3 2 1 4 2 2 2 3 2 1 4 1 1 -1 -1 -1
210. // 1 1 3 2 1 4 2 2 2 3 2 1 4 1 1
211. switch(n)
212. {
213. case 1:
214. printf("\n请输入矩阵1：\n");
215. input(m1);
216. printf("\n请输入矩阵2：\n");
217. input(m2);
218. add(m1, m2, res);
219. printf("\n结果为：\n");
220. Printf(res);
221. break;
222. case 2:
223. printf("\n请输入矩阵1：\n");
224. input(m1);
225. printf("\n请输入矩阵2：\n");
226. input(m2);
227. sub(m1, m2, res);
228. printf("\n结果为：\n");
229. Printf(res);
230. break;
231. case 3:
232. printf("\n请输入矩阵1：\n");
233. input(m1);
234. printf("\n请输入矩阵2：\n");
235. input(m2);
236. mult(m1, m2, res);
237. printf("\n结果为：\n");
238. Printf(res);
239. break;
240. }
241. return 0;
242. }