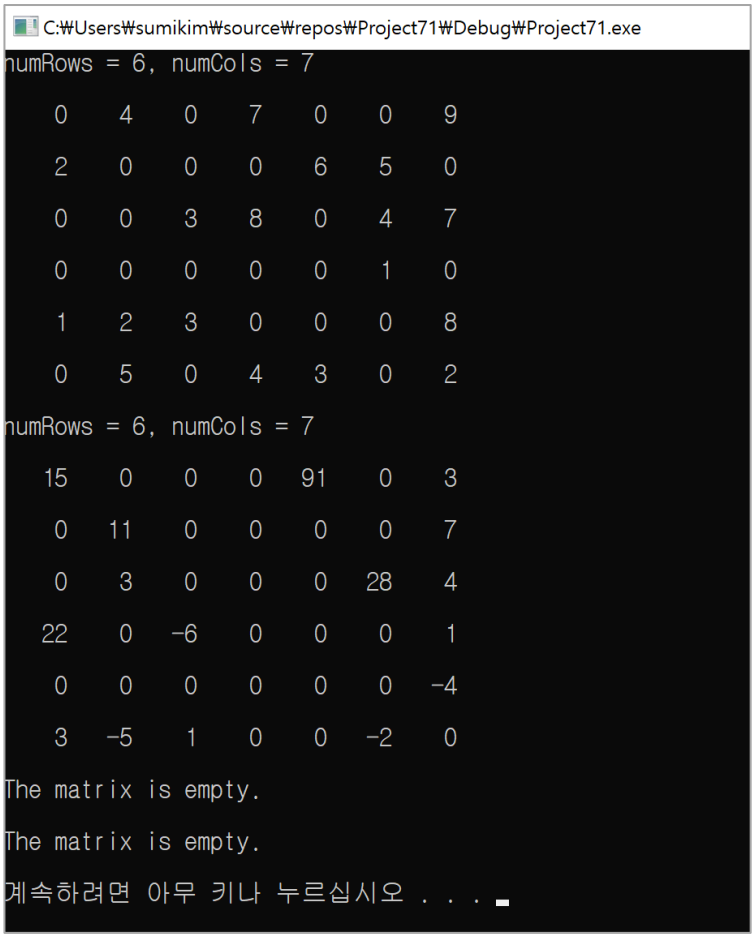
오픈랩 #5 : Sparse Matrix List

2020.11.17.5PM / 20181202 김수미

1. 코드 실행 결과



2. 코드 및 알고리즘 설명

해당 코드는 두개의 sparse matrix 를 입력받아 linked list 에 저장 후 출력하는 코드이다.

sparse matrix 란 행렬의 구성요소에 0 의 개수가 많은 행렬을 뜻하며, 일반적인 배열로 구성할 경우 0 값, 즉 ,NULL 값이 많아 공간의 낭비가 많기 때문에 이를 방지하기 위해 linked list 로 0 값이 아닌 구성요소들의 값 을 저장하되, 행렬 출력 시에는 0 인 요소를 포함하여 출력해 줄 수 있도록 하는 코드이다. 아래는 코드의 내용 으로, 교재의 코드를 전반적으로 참고했다. 가장 먼저 헤더파일들과 상수 값을 지정해주고, 구조체를 선언했다. 아래 코드에서 아래는 entryNode 와 matrixNode 의 정의 역시 확인할 수 있다. 해당 코드에는 두개의 다른 타입의 노드가 등장하므로, union 을 구조체 안에 중첩하여 사용했다.

```c
#include <stdio.h>
#include <stdlib.h>

#define MAX_SIZE  50
typedef enum { head, entry } tagfield;
typedef struct matrixNode *matrixPointer;

typedef struct entryNode {
    int row;
    int col;
    int value;
};

typedef struct matrixNode {
    matrixPointer down;
    matrixPointer right;
    tagfield tag;
    union {
        matrixPointer next;
        struct entryNode entry;
    } u;
};
matrixPointer hdNode[MAX_SIZE];

matrixPointer mread(FILE* fp);
void mwrite(matrixPointer);
void merase(matrixPointer* node);
```

hdNode 의 최대값을 상수를 사용하여 지정해주고, mwrite, merase 함수를 미리 선언해주었다.

```c
void main(void)
{
    matrixPointer a, b;
    FILE *fp1, *fp2;
    fp1 = fopen("A.txt", "r");
    fp2 = fopen("B.txt", "r");

    a = mread(fp1);
    mwrite(a);
    merase(&a);

    b = mread(fp2);
    mwrite(b);
    merase(&b);

    mwrite(a);
    mwrite(b);

    fclose(fp1);
    fclose(fp2);
    system("pause");
    return 0;
}
```

위는 메인함수이다. txt 파일을 읽어 링크드 리스트에 저장하는 mread 함수의 매개변수로 넘기는 과정과 저장된 행렬을 2 차원 행렬의 형태로 출력하는 mwrite 함수의 호출, 사용하고 난 리스트를 초기화시켜주는 merase 함수의 호출이 포함되어 있다.

```c
matrixPointer mread(FILE* fp) {
    int numRows, numCols, numEntries, numHeads, i;
    int row, col, value, CurrentRow;
    matrixPointer temp, last, node;

    fscanf(fp, "%d%d", &numRows, &numCols);
    numEntries = 19;

    numHeads = (numCols > numRows) ? numCols : numRows;
    node = (matrixPointer)malloc(sizeof(struct matrixNode));

    node->tag = entry;
    node->u.entry.row = numRows;
    node->u.entry.col = numCols;

    if (!numHeads) node->right = node;
    else {
        for (i = 0; i < numHeads; i++) {
            temp = (matrixPointer)malloc(sizeof(struct matrixNode));
            hdNode[i] = temp;
            hdNode[i]->tag = head;
            hdNode[i]->right = temp;
            hdNode[i]->u.next = temp;
        }
        CurrentRow = 0;
        last = hdNode[0];

        for (int x = 0; x < numRows; x++) {
            row = x; //row
            for (int y = 0; y < numCols; y++) {
                col = y; //col
                fscanf(fp, "%d", &value);
                if (value == 0) continue;

                if (row > CurrentRow) {
                    last->right = hdNode[CurrentRow];
                    CurrentRow = row;
                    last = hdNode[row];
                }
                temp = (matrixPointer)malloc(sizeof(struct matrixNode));
                temp->tag = entry;
                temp->u.entry.row = row;
                temp->u.entry.col = col;
                temp->u.entry.value = value;
                last->right = temp;          /* link into row list */
                last = temp;
                hdNode[col]->u.next->down = temp;  /* link into column list */
                hdNode[col]->u.next = temp;
            }
        }

        /*close last row */
        last->right = hdNode[CurrentRow];
        /* close all column lists */
        for (i = 0; i < numCols; i++)
            hdNode[i]->u.next->down = hdNode[i];
        /* link all head nodes together */
        for (i = 0; i < numHeads - 1; i++)
            hdNode[i]->u.next = hdNode[i + 1];
        hdNode[numHeads - 1]->u.next = node;
        node->right = hdNode[0];
    }
    return node;
    fclose(fp);
}
```

mread 함수이다. txt 파일에서 행렬의 context 를 입력받아 차례대로 저장한다.

```
119  void merase(matrixPointer* node) {
120      /* erase the matrix, return the pointers to the heap */
121      matrixPointer x, y;
122      int i, numHeads;
123      /* free the entry pointers by row */
124      for (i = 0; i < (*node)->u.entry.row; i++) {
125          y = hdNode[i]->right;
126          while (y != hdNode[i]) {
127              x = y;
128              y = y->right;
129              free(x);
130          }
131      }
132      /* determine the number of head nodes and free these pointers */
133      numHeads = ((*node)->u.entry.row > (*node)->u.entry.col) ?
134          (*node)->u.entry.row : (*node)->u.entry.col;
135      for (i = 0; i < numHeads; i++)
136          free(hdNode[i]);
137      *node = NULL;
138  }
```

mearse 함수이다. 행렬을 저장하는데 사용한 링크드리스트의 노드들을 초기화 시킨다.

```
141  void mwrite(matrixPointer node) {
142      /* print out the matrix in row major form */
143      int i, j;
144      int numRows, numCols;
145      matrixPointer temp, head;
146
147      if (!node) printf("The matrix is empty.₩n₩n");
148      else {
149          head = node->right;
150          /* matrix dimensions */
151          numRows = node->u.entry.row;
152          numCols = node->u.entry.col;
153
154          printf("numRows = %d, numCols = %d ₩n₩n", numRows, numCols);
155
156          temp = head->right;
157          for (i = 0; i < numRows; i++) {
158              if (temp == head) {
159                  head = head->u.next;
160                  temp = head->right;
161              }
162              for (j = 0; j < numCols; j++) {
163                  if (i == temp->u.entry.row && j == temp->u.entry.col) {
164                      printf("%5d", temp->u.entry.value);
165                      temp = temp->right;
166                  }
167                  else
168                      printf("%5d", 0);
169              }
170              printf("₩n₩n");
171          }
172      }
173  }
```

mwrite 함수이다. 저장한 행렬을 화면에 출력한다. 출력 시에는 0 인 요소들까지 출력해 완전한 형태의 행렬을 보여준다.