자료구조 HW2 보고서 20241630 윤찬영

sparse matrix를 node를 통해 head들을 next로 묶고 right down을 통해 나타내고 파일에서 행렬을 받고 출력하고 행렬 뺄셈하는 코드를 visual studio를 통해 cpp 파일로 코드를 작성함.

4개의 함수 mread, mwrite, merase, msub이 있다

mread 함수

```
matrix_pointer mread(FILE* file, int** arr, int r, int c) {
	// file에서 matrix 자료를 읽어 리스트에 저장한다.
	int row, col;
	if (file != NULL) fscanf(file, "%d %d", &row, &col);
	else {
		row = r;
		col = c;
	}
	matrix_pointer ret = (matrix_pointer)malloc(sizeof(matrix_node));
	ret->u.entry.row = row; ret->u.entry.col = col; ret->tag = entry;

	matrix_pointer rowh = (matrix_pointer)malloc(sizeof(matrix_node));
	rowh->tag = head; rowh->right = rowh; ret->down = rowh;
	rowh->down = ret;
	for (int i = 0; i < row-1; i++) {
		matrix_pointer temp = (matrix_pointer)malloc(sizeof(matrix_node));
		temp->tag = head;
		rowh->u.next = temp;
		temp->right = temp;
		rowh = temp;
	}
	rowh->u.next = ret;
	rowh = ret->down;


	matrix_pointer colh = (matrix_pointer)malloc(sizeof(matrix_node));
	colh->tag = head; colh->down = colh; ret->right = colh;
	colh->right = ret;
	for (int i = 0; i < col - 1; i++) {
		matrix_pointer temp = (matrix_pointer)malloc(sizeof(matrix_node));
		temp->tag = head;
		colh->u.next = temp;
		temp->down = temp;
		colh = temp;
	}
	colh->u.next = ret;
	colh = ret->right;


	int a;
	matrix_pointer temp;
	for (int i = 0; i < row; i++) {
```

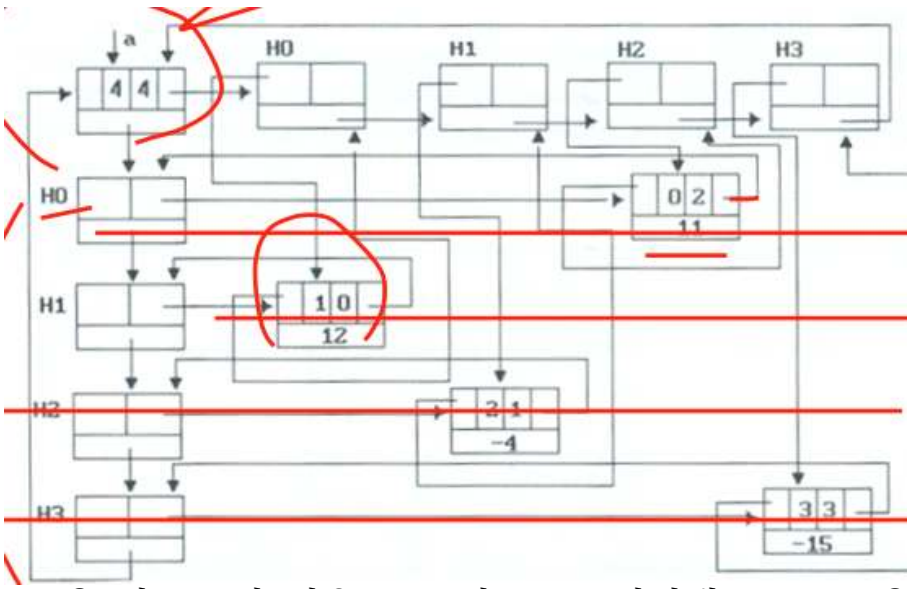
```
                for (int j = 0; j < col; j++) {
                        if (file != NULL) fscanf(file, "%d", &a);
                        else a = arr[i][j];
                        if (a != 0) {
                                matrix_pointer sin = (matrix_pointer)malloc(sizeof(matrix_node));
                                sin->tag = entry; sin->u.entry.row = i; sin->u.entry.col = j; sin->u.entry.value = a;

                                //row
                                for (int k = 0; k < i; k++) rowh = rowh->u.next;
                                temp = rowh;
                                for (; temp->right->tag != head; temp = temp->right);
                                sin->right = temp->right;
                                temp->right = sin;
                                rowh = ret->down;


                                //col
                                for (int k = 0; k < j; k++) colh = colh->u.next;
                                temp = colh;
                                for (; temp->down->tag != head; temp = temp->down);
                                sin->down = temp->down;
                                temp->down = sin;
                                colh = ret->right;
                        }
                }
        }


        return ret;
}
```

우선 인수를 보면 FILE*을 받아 sparse matrix로 만들거나 나중에 msub으로 d를 생성할 때 array로 구현하였기 때문에 int** 값도 받는다, 처음 FILE을 받을 땐 arr를 줄 수 없고 d를 줄때도 FILE*을 줄 수 없기 때문에 부를때 NULL, 0 값으로 넣는다. FILE을 받는 a,b의 경우 fscanf를 통해 txt 파일의 처음 두 int 값인 row, col을 저장하고 최초 node는 tag를 entry로

row, col을 넣고 head node를 row, col만큼 만들어 right, down에 각각 넣은 후 head node 끼리는 next로 연결하고 마지막 head node의 right, down은 각각 맨처음 node로 연결한다. 또한 row head node를 예시로 들면 sparse matrix이므로 그 행에 아무 값이 없을 수 있기에 자기 자신을 right로 받고 col은 자기 자신을 down으로 받는다

head node를 다 만든 후 fscanf를 통해 받은 값이 0이 아니라면 row head node로부터 맨끝 node(없다면 head node)의 right를 연결 받고 새 노드의 right를 head node에 연결시킨다 col이 낮은 곳부터 받는 것이므로 이렇게 하면 col이 작은 것부터 연결되어 0이 아닌 모든 node가 연결된다

col head node도 연결해야하므로 next로 그 줄의 head node를 colh에 설정한 후 col head node로부터 가장 먼 node(없다면 head node)의 down을 연결 받고 새 노드의
down을 head node로 설정한다.



arr을 받는 d일 경우 FILE이 NULL이기에 row col을 받은 인수로 설정하고, fscanf로 a를 받는 대신 arr[i][j]의 값을 받아 구현한다.

mwrite 함수

```
void mwrite(matrix_pointer node)
{
	if (node == NULL) return;


	int i;
	matrix_pointer temp, head = node->down;
	for (i = 0; i < node->u.entry.row; i++) {
		int num = 0;
		for (temp = head->right; temp != head; temp = temp->right) {
			for (; temp->u.entry.col != num; ) {
				printf("0 ");
				num++;
			}
			printf("%d ", temp->u.entry.value);
			num++;
		}
		for (; node->u.entry.col != num; ) {
			printf("0 ");
			num++;
		}
		printf("\n");
		head = head->u.next;
	}

	printf("\n");
}
```

node sparse matrix를 출력하는 코드로 head의 down 즉 0 row head node에서부터 시작해 만약 존재하는 0 row sparse matrix의 col이 2라고 할 때 그 전의 [0][0], [0][1]도 출력해야하므로 부족한 만큼 “0 ”를 출력하고 그 후 num과 같아질 때 sparse matrix의 값을 출력한다. 남은 부분도 num이 col의 크기와 같아 질 때 까지 0를 출력한다
같은 row를 다 출력한 후 row head node의 next인 1 row head node로 넘겨 최초 node인 entry노드를 만날 때까지 반복한다

merase 함수

```
void merase(matrix_pointer* node)
{


        int i, num_heads;
        matrix_pointer x, y, head = (*node)->down;
        for (i = 0; i < (*node)->u.entry.row; i++) {
                y = head->right;
                while (y != head) {
                        x = y; y = y->right; free(x);
                }
                x = head; head = head->u.next; free(x);
        }
        // free remaining head nodes
        y = (*node)->right;
        while (y != *node) {
                x = y; y = y->u.next; free(x);
        }
        free(*node) ; *node = NULL;
}
```

노드를 free 시키는 코드로 cpp로 작성해 delete를 사용하였다 row부터 right로 sparse matrix를 free시키고 next로 넘기고 해당 row의 head node로 free 시켜 마지막엔 처음 entry 노드로 돌아가고 그 후 col node도 똑같이 free시켰다 이는 4장 4.7 pdf를 참조하였다

msub 함수

```
matrix_pointer msub(matrix_pointer a, matrix_pointer b) {
	// matrix subtraction
	matrix_pointer ret = (matrix_pointer)malloc(sizeof(matrix_node));

	matrix_pointer ta = a->down;
	matrix_pointer tb = b->down;

	int** arr = (int**)malloc(a->u.entry.row * sizeof(int*));
	for (int i = 0; i < a->u.entry.row; i++) {
		arr[i] = (int*)calloc(a->u.entry.col, sizeof(int));
	}

	while (ta->tag != entry && tb->tag != entry) {
		ta = ta->right;
		tb = tb->right;
		while (ta->tag != head && tb->tag != head) {
			switch (compare(ta->u.entry.col, tb->u.entry.col)) {
			case 0:
				if (ta->u.entry.value - tb->u.entry.value) {
					//A+B 추가
					arr[ta->u.entry.row][ta->u.entry.col] =
ta->u.entry.value - tb->u.entry.value;
				}
				ta = ta->right;
				tb = tb->right;
				break;
			case 1:
				//B 추가
				arr[tb->u.entry.row][tb->u.entry.col] = -1 *
tb->u.entry.value;
				tb = tb->right;
				break;
			case -1:
				//A 추가
				arr[ta->u.entry.row][ta->u.entry.col] =
ta->u.entry.value;
				ta = ta->right;
			}
		}

		for (; ta->tag != head; ta = ta->right); //A추가
		for (; tb->tag != head; tb = tb->right); //B추가
```

```
                ta = ta->u.next;
                tb = tb->u.next;

        }
        ret = mread(NULL, arr, a->u.entry.row, a->u.entry.col);

        return ret;
}
```

a와 b node sparse matrix를 a-b로 계산하여 d를 return하는 함수로 우선 d를 2차원 배열로 나타낸 후 이를 mread를 통해 sparse matrix로 변환 시켰다

calloc을 통해 기본적으로 0으로 row col크기 만큼 동적할당하였다

각 row로 시작하여 switch, compare를 통해 col이 작은 것을 a는 그대로, b는 -1을 곱해 저장하고 만약 col이 같다면 a-b를 하여 0이 아니라면 저장하게끔하였다

main 함수

```
int main() {
	matrix_pointer a, b, d;

	FILE* file1 = fopen("A.txt", "r");
	FILE* file2 = fopen("B.txt", "r");

	if (file1 == NULL) {
		return 1;
	}

	a = mread(file1, NULL, 0, 0);

	if (file2 == NULL) {
		return 1;
	}

	b = mread(file2, NULL, 0, 0);



	mwrite(a);
	mwrite(b);


	d = msub(a, b);
	mwrite(d);

	merase(&d);
	merase(&a);
	merase(&b);

	mwrite(a);
	mwrite(b);
	mwrite(d);







	return 0;
}
```

FILE*를 통해 A.txt와 B.txt를 받고 각각 a,b에 mread 시켰다

실행 결과