

데이터 구조 설계 프로젝트1 보고서

제출일자: 2022년 10월 11일 (화)

학 과: 컴퓨터정보공학부

담당교수: 이기훈 교수님

학 번: 2021202045

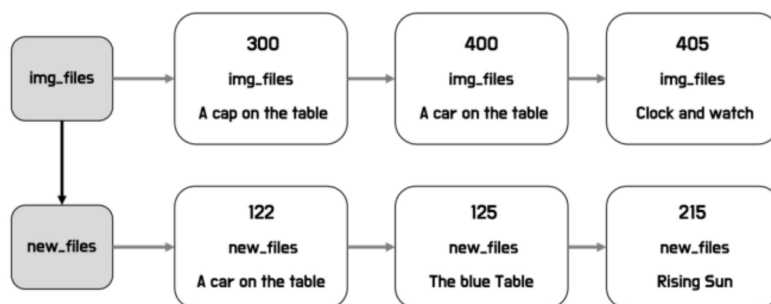
성 명: 김예은

1. Introduction

데이터 구조 설계 1차 프로젝트는 2차원 linked list, queue, stack, tree의 자료구조를 직접 구현해 보면서 자료 구조에 대한 알고리즘을 이해하는 것 뿐만 아니라, csv 파일과 txt파일로부터 데이터와 명령을 입력을 받고 궁극적으로 txt파일에 출력을 하는 파일 입출력에 대해서도 배울 수 있고, 저번 학기까지 배웠던 알고리즘들을 복습 및 정리해 볼 수 있는 종합적인 프로젝트이다. 간단하게 요약하자면 command.txt의 명령어 및 인자에 맞춰 csv파일에서 고유번호와 파일명을 load해 오고, 이를 링크드 리스트에 추가하는 방식으로 add해준다.(맨처음 load외 또 load를 할 경우) 링크드 리스트에서 고유번호를 수정해주거나 전체적으로 데이터가 100개 이하가 되게 유지해주는 버퍼 역할을 끝낸 다음에는 move를 통해 tree구조로 옮겨준다. 이때 옮겨주면서 linked list에 있던 데이터는 모두 지워준다. tree구조에서 stack을 이용한 후위순회를 한 후 파일들의 이름과 고유번호를 큐에 저장해준다. 큐에 저장한 정보들을 dequeue(pop) 해주면서 찾고자 하는 단어가 있는 파일명을 보이며 무어 알고리즘을 통해 찾고, 찾으면 출력을 해준다. EDIT 명령어를 위해 tree에서 파일 고유 번호를 기반으로 전위 순회를 통해 이미지 경로를 찾고, EDIT를 통해 -f, -l, -r은 각각 이미지 점대칭, 이미지 밝기 조정, 이미지 크기 재조정의 기능을 해준 후, 각 기능에 맞게 파일이름을 고쳐준다. EXIT을 만났을 때는 프로그램 상의 모든 메모리를 해제하며, 프로그램을 종료한다. 각 기능마다 ERROR메세지를 뜨는 조건 및 충족해야하는 작은 조건들이 있는데 이는 각 명령어에 대해서 자세히 설명할 때, 적도록 한다.

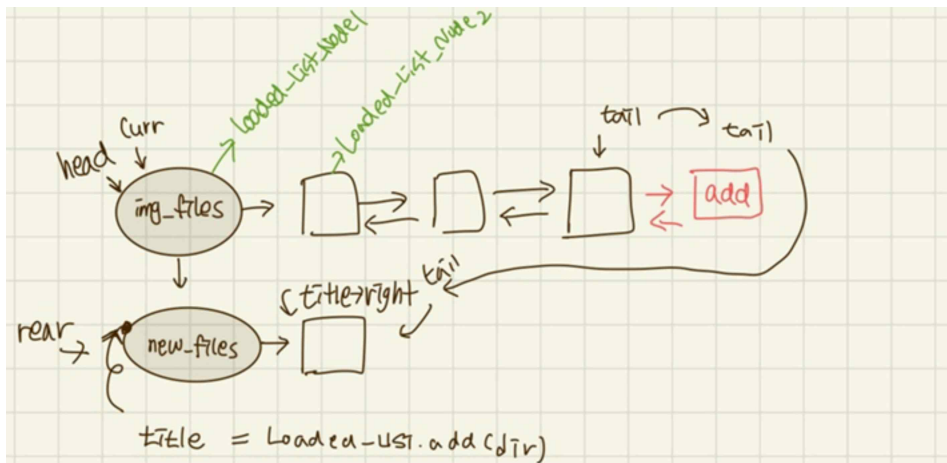
2. Algorithm

(1) 2차원 Linked List



“LOAD”나 “ADD”명령어 때 받은 경로를 통해 csv파일을 찾아 정보를 읽어 linked list에 저장하는 알고리즘으로, 맨 앞에 회색으로 연결되어 있는 부분은 디렉토리명만 저장한 가장 상위 개념을 가진 노드이다. 이 노드는 내 프로젝트 내에서 Loaded_List_Node1으로 선언 및 정의했으며 이들 끼리 linked list를 이름과 동시에 이 디렉토리 내에 있는 csv파일을 line by line으로 읽어서 고유번호, 디렉토리명, 파일명을 저장한 Loaded_List_Node2끼리도 linked list관계를 맺어 결론적으로 2

차원 linked list 구조를 이루게 했다. 위의 그림에서 img_files라고 적혀있는 Loaded_LIST_Node1을 head라고 칭했으며, head가 NULL이라면, Loaded_LIST가 비었다고 가정하여 맨 처음에 LOAD를 하면 head에 동적할당을 한 디렉토리 노드를 할당해주고 그 노드를 title이라 칭했다. 이 디렉토리 노드(회색 노드) 옆에 처음으로 Loaded_LIST_Node2가 이어질 때와 그 외의 경우의 수를 나눠 linked list를 연결해주었다. 맨처음에 삽입되는 경우가 아닐때는 tail이라는 Loaded_LIST_Node2를 가리키는 포인터를 선언해주어 tail->right = add, add->prior = tail, tail = tail->right 순서로 알고리즘을 구성했다. 노드가 하나씩 추가될 때 마다 전역변수 num이 하나씩 증가하고, 이 수가 100이 넘을 경우 가장 먼저 들어온 노드부터(head->right)삭제를 해주고 다시 새 노드를 추가해주었다. ADD의 경우, 먼저 새 디렉토리 노드(회색 노드)를 생성 후 Loaded_LIST_Node1끼리 연결해주고 새로 연결 해 준 디렉토리 노드의 주소를 Load_LIST.insert_node의 인자로 넣어서 title이 그 주솟값을 저장하게 한 후, 그 title->right부터 Loaded_LIST_Node2들을 link 해주었다.



(2) stack 및 queue

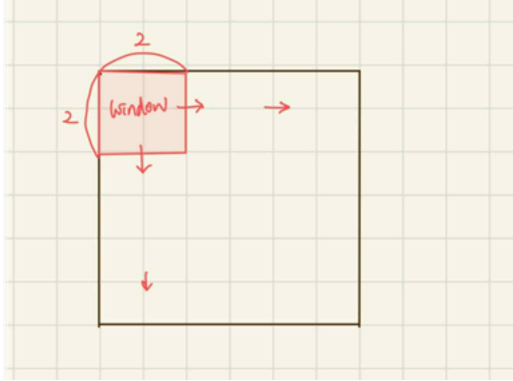
edit와 bst의 후위순회를 할 때 필요한 stack, queue구조는 배열을 통해 구현해주었다. 왜냐하면 배열의 총 크기가 정해져 있기 때문에 굳이 동적할당을 해 줄 필요가 없다고 생각했기 때문이다. 또한 동적할당이 아니기 때문에 나중에 EXIT을 할 때, 자동으로 메모리가 삭제되기 때문에 메모리 누수 부분에서도 실수가 덜 있을 것 같다고 생각했다. 우선 BST 후위순회를 위한 스택구조와 큐 구조는 Database_BST_Node의 주솟값들을 가지고 있고 각 파일명과 고유번호를 가지고 있는 stackNode를 하나 만들어주고 stackNode형 스택 배열, 큐 배열을 만들어주었다. 크기는 트리의 최대 크기인 300으로 정해주었다. 스택의 경우 생성자에서 top = -1로 명시해준 후 스택에 인자가 하나씩 들어 올때마다 ++해주면서 push해주고 pop할때는 top부터 pop을 해줌으로써 last in first out을 구현해주었다. 이렇게 스택에서 pop됨과 동시에 pop된 노드는 queue의 enqueue가 되면서 후위순회의 순서가 저장된다. 큐 구조는 생성자를 만들 때, idx = -1로 초기화 해준 후 큐 구조에 인자가 들어올 때 마다 idx++해주었다. 큐 구조에서 보이어 무어 알고리즘을 통해 search를 할 때, queue의 idx가 0부터 끝까지 돌면서 first in first out을 구현해주었다. 스택을 이용한 후위순회

에서는 데구실 수업의 조교님 설명을 인용해서 구현해보았다. 우선 cur node와 L_node가 필요하다. Root부터 왼쪽 subtree가 leaf node가 될때까지 쪽 내려가면서 훑은 노드들은(각 subtree들의 root같은 존재들) 스택에 push된다. 그리고 pop이 되는 경우는, 총 3가지가 있다. cur이 leaf node 일때, left node를 방문했고 right node가 없는 경우, cur의 right node가 L_node인 경우 pop을 해주면 된다. L_node란 pop을 한 노드가 L_node가 된다. 이 규칙을 이용하여 스택과 큐를 이용한 후위순회 알고리즘을 완성시켰다. 위의 3가지 규칙에서 "leaf node를 방문했고" 를 알기 위해 stackNode형 큐 클래스 내에 int isinQ 함수를 만들어서 인자로 고유번호를 받고 그 고유번호가 이미 큐에 있는지 없는지 큐의 모든 요소를 돌아보고 이미 있으면 return 1이 되게했다. 따라서 isinQ ==1 && right node != NULL일때 pop과 Enqueue가 실행되는 것이다.

edit의 스택과 큐는 각각 클래스 이름 editStack과 editQ이다. 전체적인 알고리즘은 당연히, bst의 후위순회를 위한 스택과 큐와 거의 동일하고, stack배열이 unsigned char형과 queue배열이 unsigned char형이라는 점과 배열 크기들이 256*256+1(혹시 모르는 여유 공간)이라는 점이다. 이들을 이용한 edit 알고리즘은 바로 다음 문단에서 설명을 이어가겠다.

(3) EDIT

우선 이미지 파일(256x256)을 읽어오고 각 픽셀들을 input_data[i][j]로 표현한다. 이미지 점대칭의 경우 stack을 이용해 first in last out을 구현하면 이미지가 점대칭이 된다. 따라서 이중 중첩문을 이용하여 push해준후 pop을 한 것을 output_data로 받아서 파일명을 _flipped.RAW로 바꿔주면 완성이다. 이미지 밝기 조정의 경우 이중 중첩문을 통해(왜냐하면 행과 열로 2차원적으로 각 픽셀들을 다 훑어줘야하기 때문에) enqueue해준다음 순서대로 dequeue해주면서 인자로 받은 밝기 변수를 각 픽셀에 더해주면 된다. 여기서 원래 픽셀 값 + 인자로 받은 밝기 변수 > 255(최댓값)이면, 255으로 고정해주고, 최솟값보다 적어지면 최솟값(0)으로 고정해준다. 그렇게 얻은 output_data의 파일명을 _adjusted.RAW로 바꿔주면 완성이다. 이미지 크기 재조정은 1/4로 크기를 줄여줄 것이기 때문에 인접한 네개의 픽셀들의 평균값을 하나의 픽셀로 넣어주면 된다. 여기서 쓰인게 CNN의 WINDOW같은 개념인데, 2X2정사각형이 2칸씩 가로, 세로로 건너뛰면서 그 WINDOW내의 픽셀들의 값의 평균을 내서 하나의 픽셀로 산출하면 된다. 이렇게 output_data를 받으면 파일명을 _resized.RAW로 바꾸면 완성이다.



위의 그림은 resized 알고리즘을 그림으로 구현한 것이다.

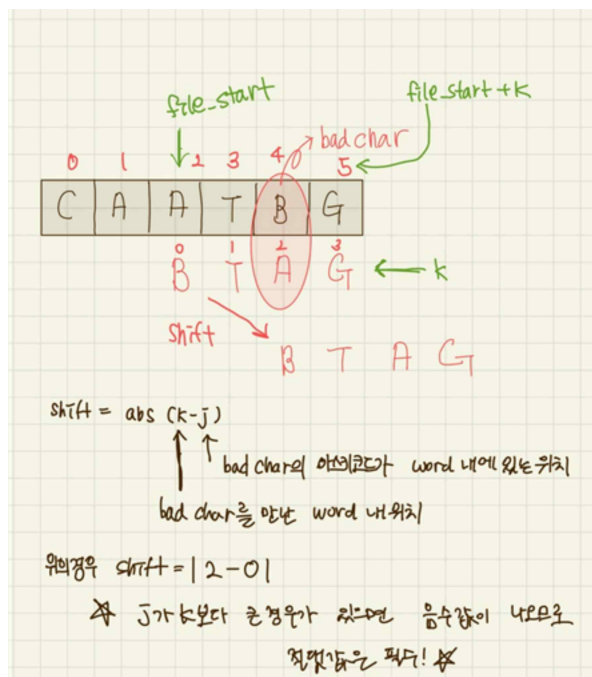
(4) BST

이진탐색트리는 자료를 찾을 때 처리 속도를 빠르게 해주는 장점이 있다. 이번 프로젝트에서 구현하는 트리는 고유번호의 순서로 작으면 부모노드의 왼쪽, 크면 부모노드의 오른쪽으로 가는 규칙과 링크드 리스트의 마지막 노드부터 트리로 옮긴다는 규칙(1)을 바탕으로 한다. Root가 NULL 값이면 해당 트리가 비었음을 알 수 있고, Database_BST_Node를 동적할당해주어 링크드 리스트의 노드의 정보를 복사하여 그 노드들을 트리로 연결해주었다. 이때, 동시에 링크드 리스트의 메모리도 해제해주었다. 위의 (1)은 어떻게 구현했냐면, linked list의 head가 맨처음 directory노드 이므로 즉, 위의 그림을 참고하면 img_files가 head인 것이다. head를 curr로 지정하여 curr->down != NULL인 동안 curr=curr->down해주고 맨 마지막 디렉토리 노드까지 가면 그 디렉토리 노드와 연결된 Loaded_LIST_Node2들의 맨 끝(tail)까지 가서 거기서부터 title(해당 디렉토리 노드)까지 역순하면서 tree로 move해준후 바로바로 노드를 지워준다. 그리고 디렉토리 노드만 남았을 때 그 디렉토리 노드를 가리키는 부모 디렉토리 노드의 down을 NULL로 초기화해주어 LINKED LIST를 끊어주고 그 끊겨서 독립된 노드 즉, rear노드를 delete해준다. 그리고 다시 head부터 curr이 마지막 노드일때 까지 가서(하나씩 줄어들다가 결국 head만 남을 것) 위의 알고리즘을 반복한다. rear노드가 head가 되면 break를 통해 빠져나온다. 트리노드를 삭제해줄 때는 경우의 수를 세 가지로 나누어야한다. 자식노드가 없을때는 그냥 지워주면 되고, 자식노드가 하나만 있을 때는 부모노드를 파악해야지만 부모 노드와 자식노드를 연결해준 후 자신을 지우면 된다. 자식노드가 두 개인 노드를 지울 때는 자식노드의 왼쪽중에서 가장 큰 값을 찾아 그 값과 switch해준 후, 그 위치의 노드를 지워 주면 되는데 그 위치의 노드 또한 자식노드가 0개일지, 1개일지, 2개일지 모르기때문에 재귀함수를 통해 구현해주면 된다. 그렇게 EXIT을 할때 트리의 모든 노드를 삭제한다면, 프로젝트 상의 할당된 메모리는 모두 삭제된 상태로 main함수가 끝나게 되는 구조이다.

(5) 보이어 무어 알고리즘

찾고자하는 단어나 문자의 일부분을 string형 변수 word에 인자로 받는다. 보이어무어 알고리즘은

단어의 뒷부분부터 문자를 비교해주는 알고리즘으로 예를 들어 word의 문자 한개와 queue에 저장된 파일명의 문자 한개가 같지 않다면, 두 가지의 경우에 따라 shift를 다르게 해주어 이동시키면 된다. 우선, 첫번째 경우인 같지 않은 문자를 bad char이라고 칭한다고 가정하면, bad char가 찾고자하는 word변수 내에 있는 문자이면 word내의 문자를 bad char가 있는 위치로 shift해 준 뒤, file과 word의 비교를 그 위치에서부터 다시 해주면 된다. 만약, bad char가 word내에 없는 문자라면, 그냥 word의 크기만큼 shift해준 후 거기서부터 다시 비교를 시작해주면 된다. 이를 더 쉽게 설명하기 위해 밑의 그림을 첨부했다.

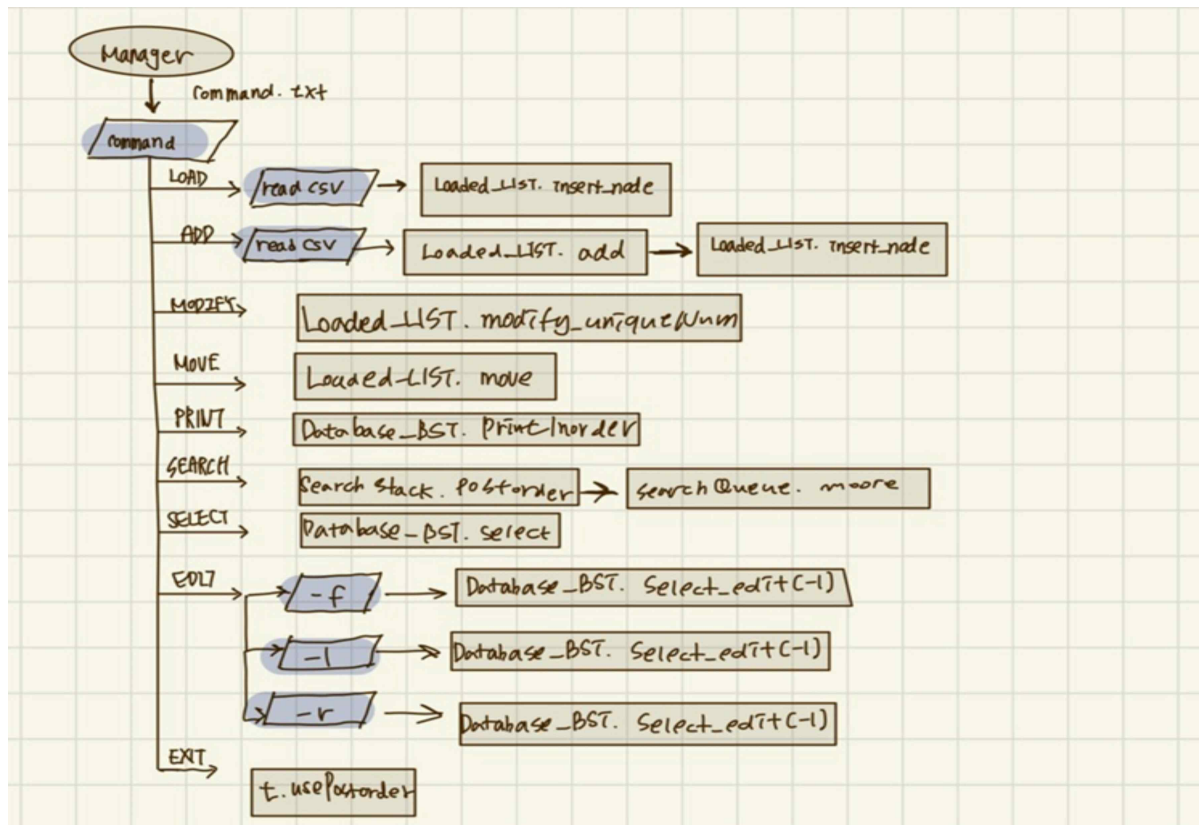


그렇다면 bad char가 word내의 있는지 어떻게 알 수 있는가? 이를 탐색하기 위해 word의 문자를 ascii코드로 바꿔서 int형 memo_ascii라는 배열에 삽입해주었다. file 비교를 시작하는 부분은 file_start, word의 크기는 word_size, word의 패턴이 시작되는 곳(즉, word의 맨뒤부터 차근차근 앞으로 오는 index값)는 k라고 선언해주었다. (파일비교의 시작점+k) (왜냐하면 찾고자하는 단어의 사이즈만큼 뒤부터 비교할 것이므로)와 word의 k 번째 문자가 같다면 k--을 해주어서 index를 앞으로 보내면서 비교해준다. k<0이면, 모든 문자열이 같다는 것이므로(예를 들어 "man"을 찾고하면 n->a->m순서로 비교하므로 k값이 2->1->0(다 찾고 k--이므로 결과적으로 k=-1인상태)이 되기때문에) 해당 파일을 출력해주면 된다.

3. FLOWCHART

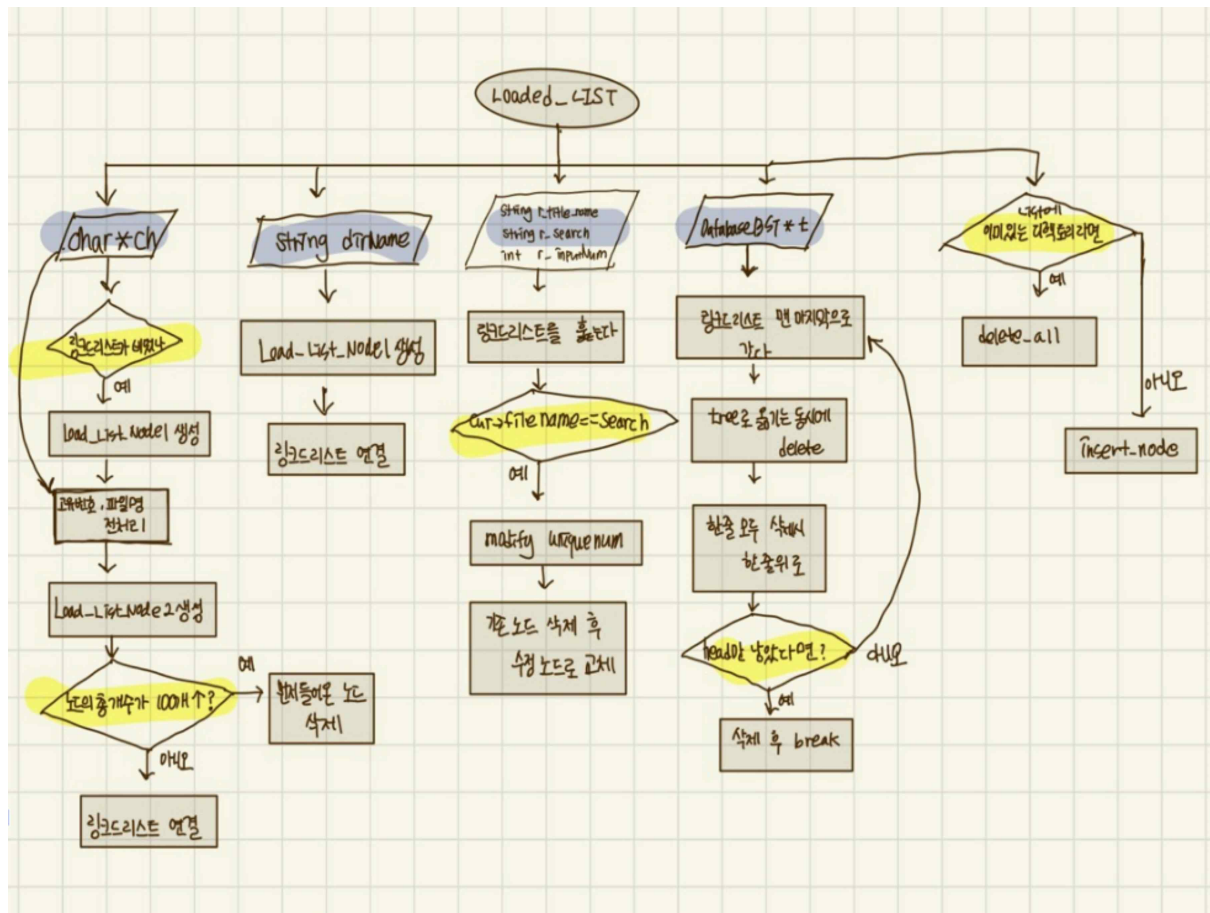
플로우 차트는 각 헤더파일을 기준으로 나누어서 그렸고, 각 클래스 내에서 중요한 역할을 하는 함수를 뽑아 흐름도를 그렸다. 대부분 알고리즘 설명 때 자세한 설명을 했으니, 부연설명할 부분만 플로우차트 그림과 함께 설명하겠다.

(1) Manager.h



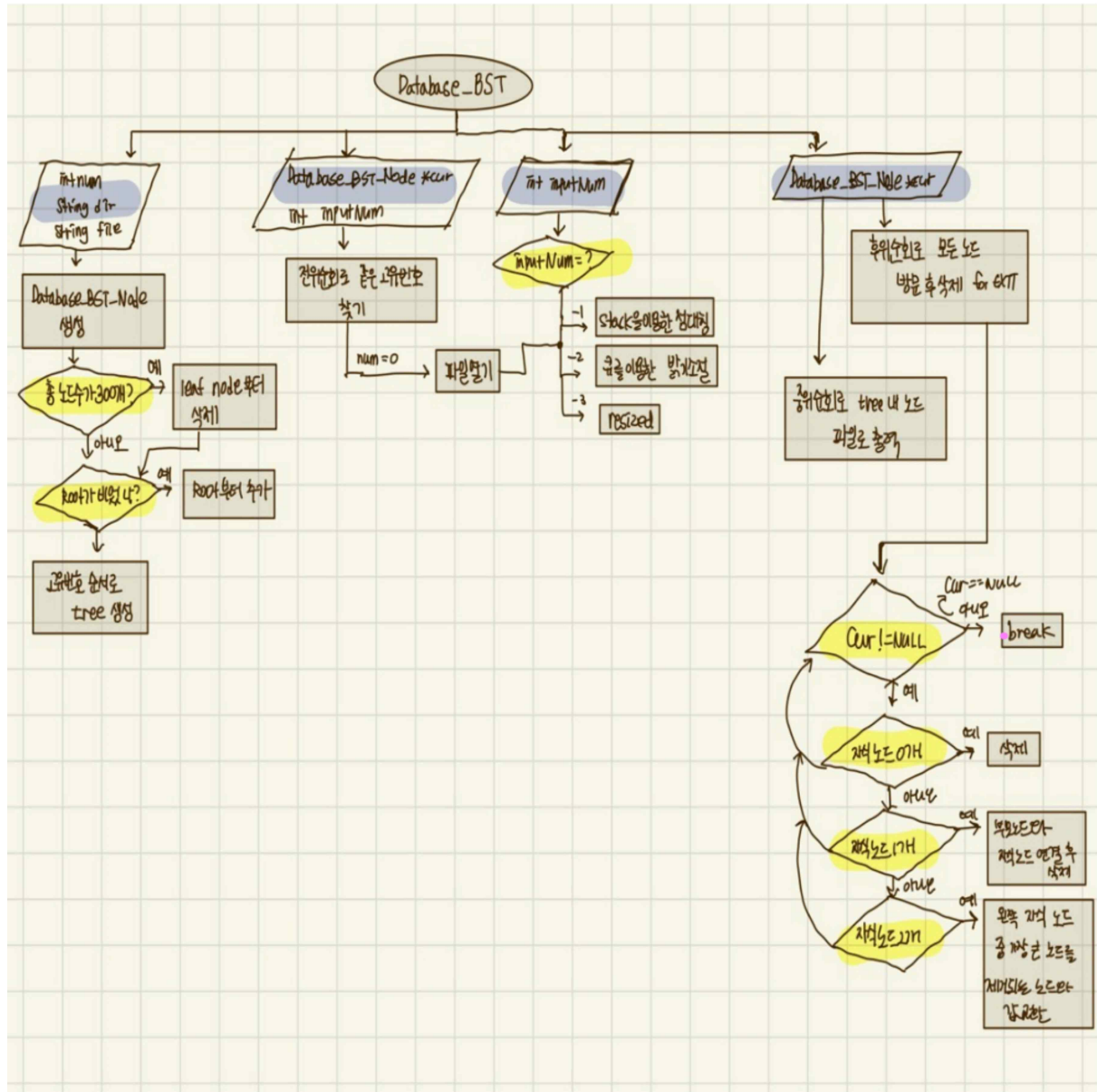
우선 Manager.h에 있는 manager class는 command.txt에서 명령어를 읽어와서 명령어에 따라 각각 클래스 객체 내의 함수에 접근한다. EXIT의 t.usePostorder()의 경우 Database_BST의 객체 t 내의 후위순회 함수를 통해 할당된 모든 메모리를 삭제 후 EXIT을 진행한다.

(2) LoadedCsv.h 내의 Loaded_LIST 클래스



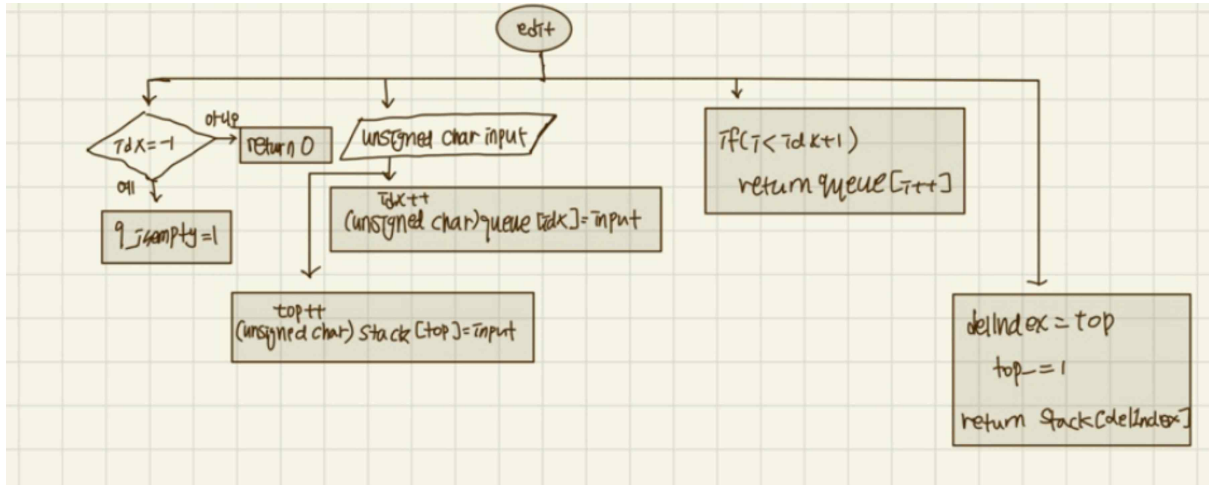
(1번째 흐름도)line by line으로 받은 csv 파일을 char형으로 바꾼 것을 인자로 받은 후 ';'과 '.'(RAW파일을 지우기 위해)을 기준으로 고유번호와 파일명을 전처리 해준 후 노드의 총 개수가 100개 이하일때 링크드 리스트로 연결해준다. (3번째 흐름도)MODIFY의 경우 링크드 리스트를 전체적으로 훑다가 cur(현재노드)의 파일명과 인자로 받은 파일명이 같으면 새로운 고유번호 값을 저장하고 있는 새 노드를 만들어서 기존 노드의 위치와 교체 시킨다. 4번째와 5번째 흐름도는 각각 move와 예외처리에 대한 것(기존에 있는 디렉토리에 대해 load나 add를 실행할 시)으로 위의 algorithm설명에 자세히 적었다.

(3)Tree.h 내의 Database_BST class



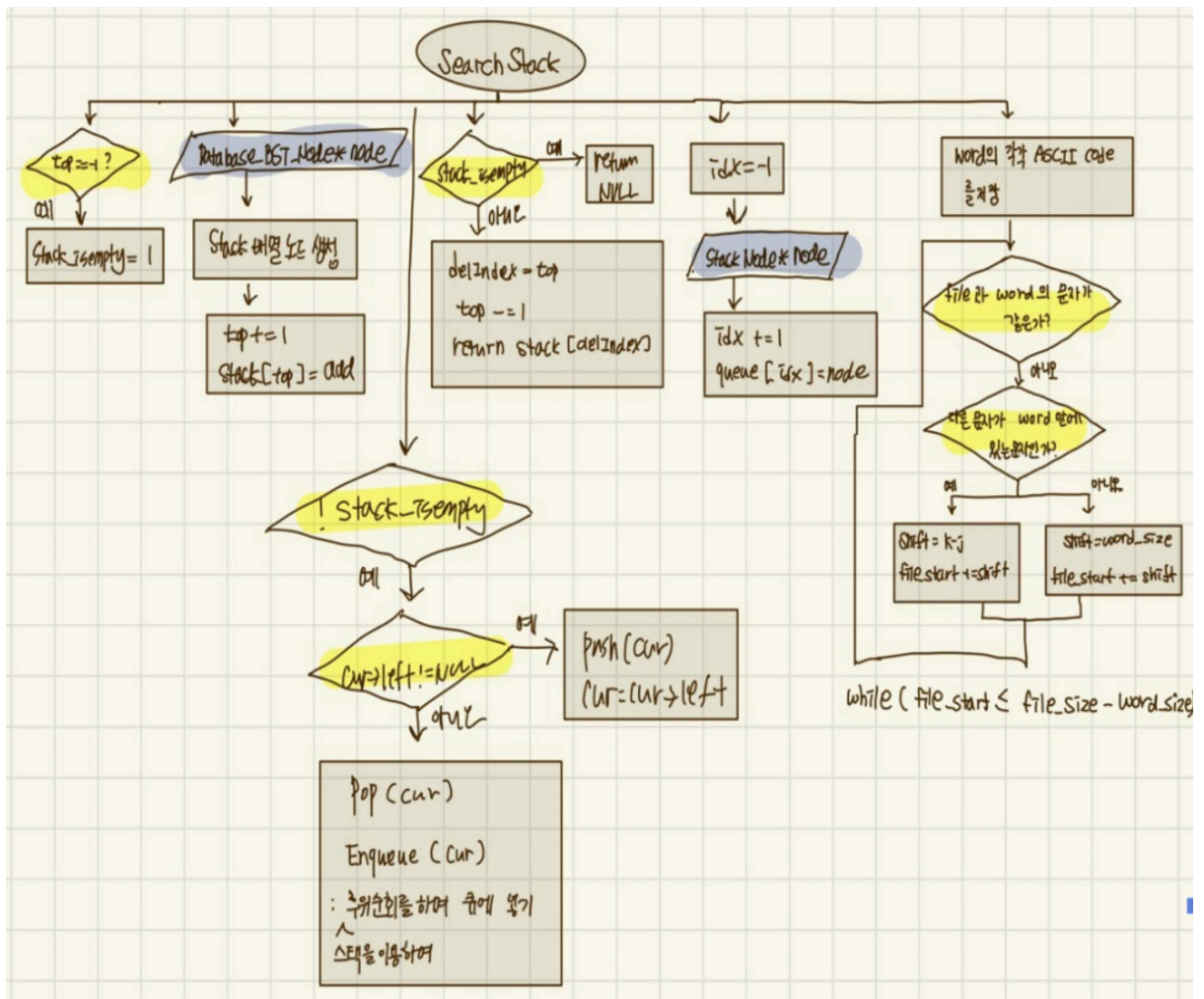
가운데 두개의 흐름도는 select와 edit에 쓰이는 함수를 흐름도로 간단하게 그린 것이다. select 함수의 역할은 찾고자 하는 파일의 고유번호를 인자로 받고 전위 순회로 트리의 노드를 모두 방문하다가 cur의 고유번호와 인자 고유번호값이 같다면 파일을 열어준다. 그리고 파일을 열어준 뒤 return한다. EDIT을 하면 해당 고유번호를 가진 cur의 주솟값을 저장해둔 find node의 주솟값을 인자로 get_img함수 안에 들어가 -1일때는 이미지 점대칭, -2일때는 밝기조절, -3일때는 resized 기능을 구현하게 했다.

(4) edit.h



edit.h내의 editQ와 editStack 클래스의 경우도 위의 알고리즘에서 설명하였으므로 자세한 설명은 생략하겠다.

(5) SearchStack.h




이 헤더파일에서는 후위순회를 할 때 필요한 스택과 큐구조를 클래스로 선언하였고, 보이여 무어

알고리즘을 큐 클래스 안에 선언해주었다. 왜냐하면 큐에 저장된 파일명 정보들을 가져와서 search명령어를 수행해야하기 때문이다. 3번째 흐름도의 경우 stack이 비어 있지 않을 경우 push와 pop기능을 하는 stack을 설명하고 있고, 4번째 흐름도는 enqueue를 하는 흐름도, 마지막의 경우 보이어 무어 알고리즘을 흐름도로 그린 것이다. 보이어 무어 알고리즘의 경우 비교해야하는 word의 크기와 비교시작점인 file_start를 더해서 총 파일명의 크기를 넘지 않는 동안 루프를 돌려줬다.

4. Result Screen

1. 깃허브에 올라온 command.txt에 대한 결과화면(log.txt)

 command.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

LOAD

ADD new_files new_filesnumbers.csv

MODIFY img_files "A cap on the table" 506

MODIFY new_files "A cup on the table" 404

MOVE

PRINT

SEARCH "on the"

SELECT 404

EDIT -f

EDIT -l 16

EDIT -r

EXIT

```

=====LOAD=====
there are lots of people in the park/100
the man is gorgeous/111
the woman is smiling/200
the man takes a picture/222
three peppers are big/300
the building is like a pyramid/333
river is under the bridge/400
The house has windows and doors/444
=====
=====ADD=====
SUCCESS
=====
=====ERROR=====
300
=====
=====ERROR=====
300
=====
=====MOVE=====
SUCCESS
=====
=====PRINT=====
img_files/"there are lots of people in the park/"100
img_files/"the man is gorgeous/"111
img_files/"the woman is smiling/"200
img_files/"the man takes a picture/"222
img_files/"three peppers are big/"300
img_files/"the building is like a pyramid/"333
img_files/"river is under the bridge/"400
img_files/"The house has windows and doors/"444
new_files/"the boat sails a big river /"500
new_files/"The celebrity posed for the picture/"555
new_files/"there are lots of ariplane/"600
new_files/"The woman is looking at the man /"666
new_files/"lena is famous person/"700
new_files/"the woman is wearing a hat/"777
new_files/"there are cars and people/"800
new_files/"airplane fly on the mountain/"888
new_files/"there are numbers and chinese characters/"900
new_files/"fence is near trees/"999
=====
=====SEARCH=====
"airplane fly on the mountain" /888
=====

=====ERROR=====
700
=====

=====ERROR=====
800
=====

=====ERROR=====
800
=====

=====ERROR=====
800
=====

=====EXIT=====
SUCCESS
=====|

```

img_files directory 안에 있는 csv파일 내의 파일들을 잘 LOAD한 것을 볼 수 있다. MODIFY의 명령어의 경우 고치고자 하는 고유번호의 파일명인 "A cap on the table", "A cup on the table"이 linked list에 없는 파일명이었기 때문에 ERROR메세지가 떴다. MOVE의 경우 linked list에서 tree로

데이터들이 잘 옮겨진 것을 볼 수 있고, linked list에 저장한 모든 노드가 출력되었음을 볼 수 있다. "on the"를 SEARCH한 명령어에서 파일명에 "on the"를 가진 파일명과 그 고유번호를 잘 출력해준 것을 볼 수 있다. SELECT의 경우 없는 고유번호를 찾고자 하였기 때문에 ERROR메세지가 나오고 SELECT와 연결되는 프로그램인 EDIT또한 RAW파일을 열 수 없어 세 번의 EDIT 동작에서 ERROR메세지가 잘 나오는 것을 확인 할 수 있다. EXIT을 할 때는 동적할당해준 메모리를 모두 삭제 후 잘 출력하는 것을 볼 수있다.

2. 임의로 기능을 확인하기 위해 만든 command.txt

```
Open - [icon] ~/2021202045_DS_project1
LOAD
ADD new_files new_filesnumbers.csv
MODIFY img_files "the man is gorgeous" 506
MOVE
PRINT
SEARCH "on the"
SELECT 506
EDIT -f
EDIT -l 22|
EDIT -r
EXIT
```

Open ▾



*log.txt

~/2021202045_DS_project1

Save

```
=====LOAD=====
there are lots of people in the park/100
the man is gorgeous/111
the woman is smiling/200
the man takes a picture/222
three peppers are big/300
the building is like a pyramid/333
river is under the bridge/400
The house has windows and doors/444
=====
=====ADD=====
SUCCESS
=====
=====MODIFY=====
SUCCESS
=====
=====MOVE=====
SUCCESS
=====
=====PRINT=====
img_files /"there are lots of people in the park"/ 100
img_files /"the woman is smiling"/ 200
img_files /"the man takes a picture"/ 222
img_files /"three peppers are big"/ 300
img_files /"the building is like a pyramid"/ 333
img_files /"river is under the bridge"/ 400
img_files /"The house has windows and doors"/ 444
new_files /"the boat sails a big river "/ 500
img_files /"the man is gorgeous"/ 506
new_files /"The celebrity posed for the picture"/ 555
new_files /"there are lots of ariplane"/ 600
new_files /"The woman is looking at the man "/ 666
new_files /"lena is famous person"/ 700
new_files /"the woman is wearing a hat"/ 777
new_files /"there are cars and people"/ 800
new_files /"airplane fly on the mountain"/ 888
new_files /"there are numbers and chinese characters"/ 900
new_files /"fence is near trees"/ 999
=====
=====SEARCH=====
"airplane fly on the mountain" /888
=====
=====SELECT=====
SUCCESS
=====

=====EDIT=====
SUCCESS
=====

=====EDIT=====
SUCCESS
=====

=====EDIT=====
SUCCESS
=====

=====EXIT=====
SUCCESS
=====
```

위의 command는 img_files내의 "the man is gorgeous"라는 파일명을 가진 파일의 고유번호를 506으로 바꾼 뒤 move->print했을 때 고유번호가 바뀌어서 잘 출력되는 것을 볼 수 있었고, 바뀐 고유번호를 토대로 select를 했을 때 파일을 잘 발견하였으며 해당 raw file이미지에 대한 edit도 성공적으로 한 것으로 출력된다. EXIT을 할때도 동적메모리 할당 해제가 성공한 것으로 출력된다. 밑의 그림은 "the man is gorgeous"에 대한 edit된 그림들이다.



이미지 점대칭



밝기 조절



크기 1/4만큼 줄인 이미지

5. Consideration(고찰)

우선 LOAD 명령어를 통해 디렉토리 내의 csv파일을 읽을 때, csv파일의 인코딩이 utf-8으로 되어 있어서 파일의 첫 줄 부분에 쓰레기값이 들어가서 제대로 작동이 되지 않았다. utf-8 파일을 읽는 문법이 따로 있었지만(예를 들어, wftread) 보통은 파일 인코딩을 바꾸거나 자세한 코딩 설명이 없었기 때문에 맨 처음에 들어가는 세 개의 쓰레기값을 지워주는 기능을 erase를 통해 구현했다. LOAD를 통해 linked list를 구현 한 후 ADD를 할 때 노드를 추가해줄 때 tail이라는 포인터를 사용하여 노드를 추가해줬는데, 알고보니 계속 img_files디렉토리에 다른 디렉토리에서 읽어온 파일들도 같이 linked 되는 것을 발견했다. ADD를 할 때 tail을 초기화해주지 않아서 생긴 문제였다. 그 다음으로 이 프로젝트하면서 어려웠던 기능이 move였다. linked list자체의 맨 끝에서 부터 tree로 옮겨줘야 했기 때문에 디렉토리 부분에서도 맨 마지막으로 가는 탐색구조가 필요했고, 그 디렉토리랑 연결되어 있는 파일정보가 담긴 노드들의 맨 끝으로 가는 탐색구조 또한 필요했기 때문이다. tree로 값을 옮겨주면서 바로바로 linked list에 동적 할당 된 노드들을 삭제해주는 작업을 해줬는데 Loaded_LIST_Node2에 관한 delete는 쉬웠지만 디렉토리 노드들(Loaded_LIST_Node1)에 대한 삭제를 하기 어려웠다. 왜냐하면 Loaded_LIST_Node2는 노드들끼리 연결해줄 때 이중 링크드 리스트를 사용했기때문에 노드의 전과 뒤로 움직이기 자유로웠는데 디렉토리 노드는 이중 링크드 리스트가 아니었기 때문에 아래로 갈 수는 있어도 순서를 역행하기는 어려웠다. 이를 해결하기 위해 parent node를 도입하여 해결해주었다. 또한 해당 노드를 delete만 해서는 오류가 났기 때문에 해당 노드를 가리키는 부모노드의 포인터 부분도 NULL로 바꿔주어 완전히 연결을 끊어버렸더니 디버깅 오류가 안났다. 그 다음 어려웠던 구현은 스택을 이용한 후위순회였다. 스택이나 큐 구조 자체는 구현하기 어렵지 않았지만, 왼쪽 subtree를 돌고 오른쪽 subtree를 돌고 root를

도는 후위순회를 돌 때 부모노드와 현재노드가 push, pop이후로 어디로 옮겨줘야 어떤 예외사항도 안만들고 규칙적으로 움직이는지 계속 여러 경우의 수를 생각했던 것 같다. pop을 해준 후 top이 가리키는 노드를 cur로 한 후 움직이면 된다는 규칙을 찾아냈고, 이미 left node를 방문했는지 안했는지 알기위해 큐에 해당 고유번호가 있는지 없는지 훑어주는 함수를 만들어 문제를 해결했다. 보이어 무어 알고리즘을 구현할 때는 무작정 문자가 같지 않으면 찾고자하는 문자열의 길이만큼 건너뛰면 되는 줄 알았는데 생각보다 shift를 해주는 방식이 구현하기 어려워서 힘들었다. 특히 bad char가 찾고자하는 문자열 내에 있는 단어인지 알기 위해 찾고자하는 문자열이 가지고 있는 문자들의 아스키코드를 문자에 적힌 순서대로 저장해주는 int형 배열을 만들었고 shift를 얼마만큼 해줄지 일반화해주기 위해 여러 경우의 수를 계산해보았다. 그렇게 나온 결과가 $shift = (bad\ char\ 를\ 만든\ 찾고자\ 하는\ 문자열\ 내의\ 위치) - (bad\ char의\ 아스키\ 코드가\ 있는\ int형\ 배열의\ index)$ 였는데 끝나지 않고 무한 루프가 돌아서 당황스러웠다. 그러다가 위의 공식이 음수가 나오는 경우가 있을 수 있겠다는 생각과 함께 절댓값으로 바꿔주는 abs()함수를 썼더니 해결이 되었다. 이번 프로젝트를 통해 char형을 string형으로 (strtok사용을 위해) string 형을 char형으로, string형을 int형으로 바꾸는 등 형변환에 대한 함수를 많이 써서 다양한 형에 대한 두려움이 많이 없어졌고, 파일 입출력 부분도 항상 헛갈렸는데 이번 프로젝트를 하다보면서 파일 입출력에 대해서도 자신감을 갖게 되었다. 특히 공백을 기준으로 또는 특별한 문자를 기준으로 알아서 파일을 읽을 때 나눠서 변수에 할당해주는 stringstream에 대해 새롭게 알게되었다.