데이터구조 1차 프로젝트 결과 보고서

학과 : 컴퓨터정보공학부

학번 : 2021202071

이름 : 이민형

Introduction

이번 프로젝트는 지난 학기에 배운 Linked List, Binary Search Tree, Stack와 Queue를 이용하여 간단한 사진 파일 편집 프로그램을 구현하는 것을 목표로 한다. 우선 이전에는 Visual Studio를 이용하여 직접 Command를 입력하였지만, 이번 프로젝트에서는 Command.txt에서 한 줄씩 읽어와 동작을 수행하는 방식으로 프로젝트를 구현한다.

프로그램 수행 과정을 간단히 설명하자면 다음과 같다. 우선 LOAD는 프로젝트 폴더에 있는 filesnumbers.csv파일을 읽어와 단방향 Linked List를 생성하고, ADD 명령어는 LOAD로 생성된 Linked List에 2차원으로 연결을 해주어 아래에 ADD List를 생성한다.

LOAD와 ADD가 잘 수행이 되었다면 MODIFY 명령어로 파일 이름을 이용하여 노드의 파일 고유 번호를 수정한다. List에 존재하는 파일 이름에 대한 노드를 삭제하고 새로운 노드를 추가하는 방식으로 노드를 수정한다.

다음은 List에 있는 노드들을 Binary Search Tree로 옮기는 MOVE 명령어를 수행하여 준다. 파일 고유 번호와 파일 이름을 이용하여 BST의 노드를 생성하며 list에 마지막에 있는 노드부터 차례로 지우고 BST에 옮기는 방식으로 설계한다.

MOVE 명령어가 올바르게 수행되었는지를 확인하기 위해서 PRINT 명령어를 수행한다. PRINT는 BST에 있는 모든 노드들을 중위 순회 방식을 이용하여 탐색한다.

다음은 SEARCH 명령어이다. SEARCH 명령어는 특정 단어가 포함된 모든 파일 이름과 고유번호를 출력한다. SEARCH 명령어에서는 보이어 무어 알고리즘을 이용하여 트리 노드들의 파일 이름과 특정 단어를 비교한다. 또한, 파일의 이름기반 검색을 위해서는 반복문을 이용한 전위순회 방식을 이용한다.

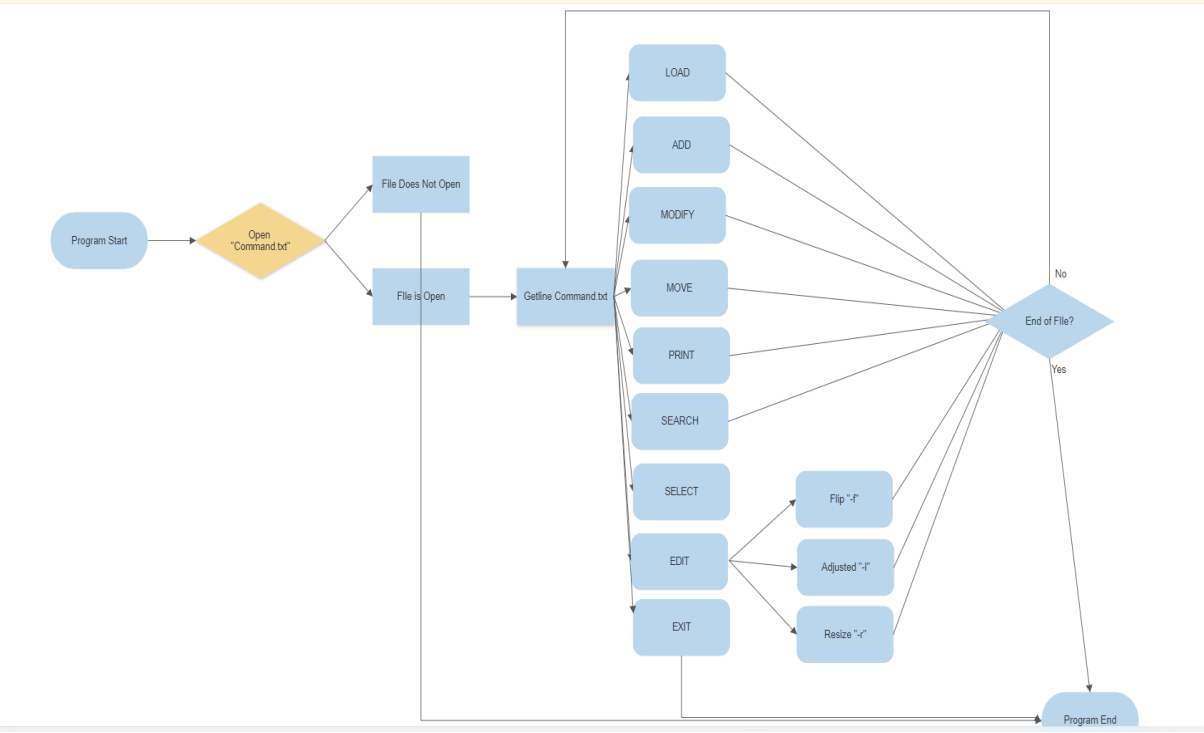
SELECT 명령어는 EDIT 명령어를 구현하기 위해서 .raw파일을 불러오는 명령어로 BST에서 고유번호를 기반으로 전위 순회를 통하여 이미지의 경로를 찾아서 이미지를 불러오는 명령어이다.

마지막으로 사진 편집 프로그램에서 사진 편집을 구현하기 위한 EDIT 명령어이다. EDIT 명령어는 총 3가지로 나뉘는데, 먼저 -f 옵션은 점대칭 동작을 수행한다. 점대칭 동작은 Stack 자료구조를 이용하여 점대칭을 수행한다. 다음으로 -l 옵션은 이미지의 밝기 조정 동작을 수행하며 인자를 추가로 얼만큼의 밝기를 증가시킬 것인지에 대한 인자가 추가적으로 필요하다. 구현할 때 이미지의 픽셀을 Queue 자료구조에 구현하여 선입 선출 방식을 이용하여 이미지의 밝기를 조정한다. 마지막으로 -r 옵션은 이미지의 사이즈를 1/4만큼 감소시켜주는 동작으로 4개 셀의 평균값을 구해 하나의 셀에 입력하는 방식으로 구현한다.

Flow Chart

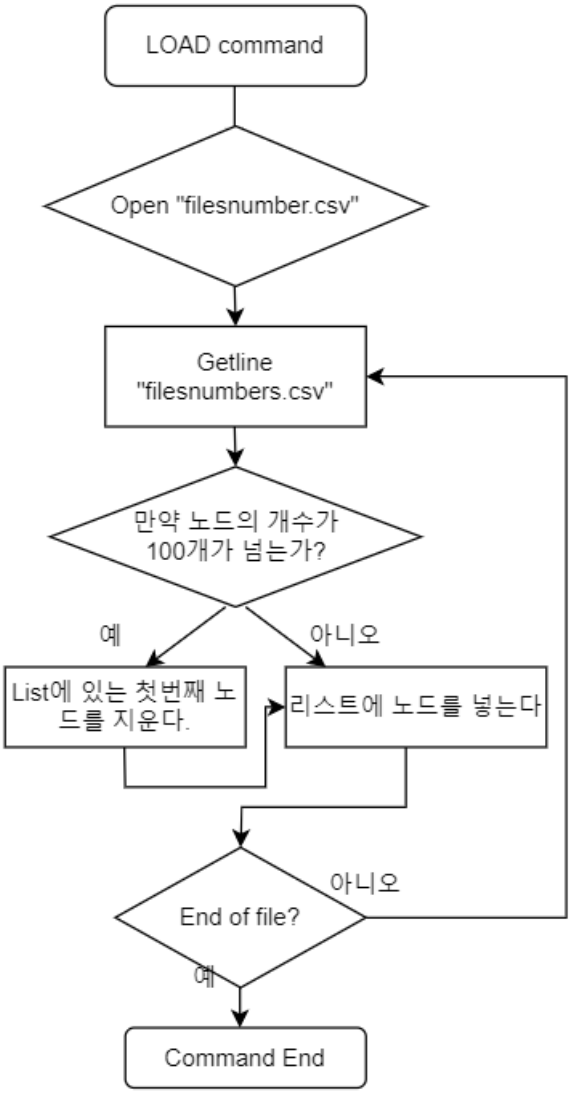
우선 지난 학기의 객체지향프로그래밍에서는 Main함수에 모든 Class를 선언하고 Main함수 내에서만의 동작으로 프로그램을 구현하였다. 반면 이번 프로젝트는 각 Class에 대해서 헤더파일과 함수 구현 파일을 따로 구현하여 코드의 수정을 더 쉽게 해주었다. 또한 Manager Class를 구현하여 프로그램을 전체적으로 관리하는 방식으로 프로젝트를 설계하였다.

<RUN의 Flow Chart>



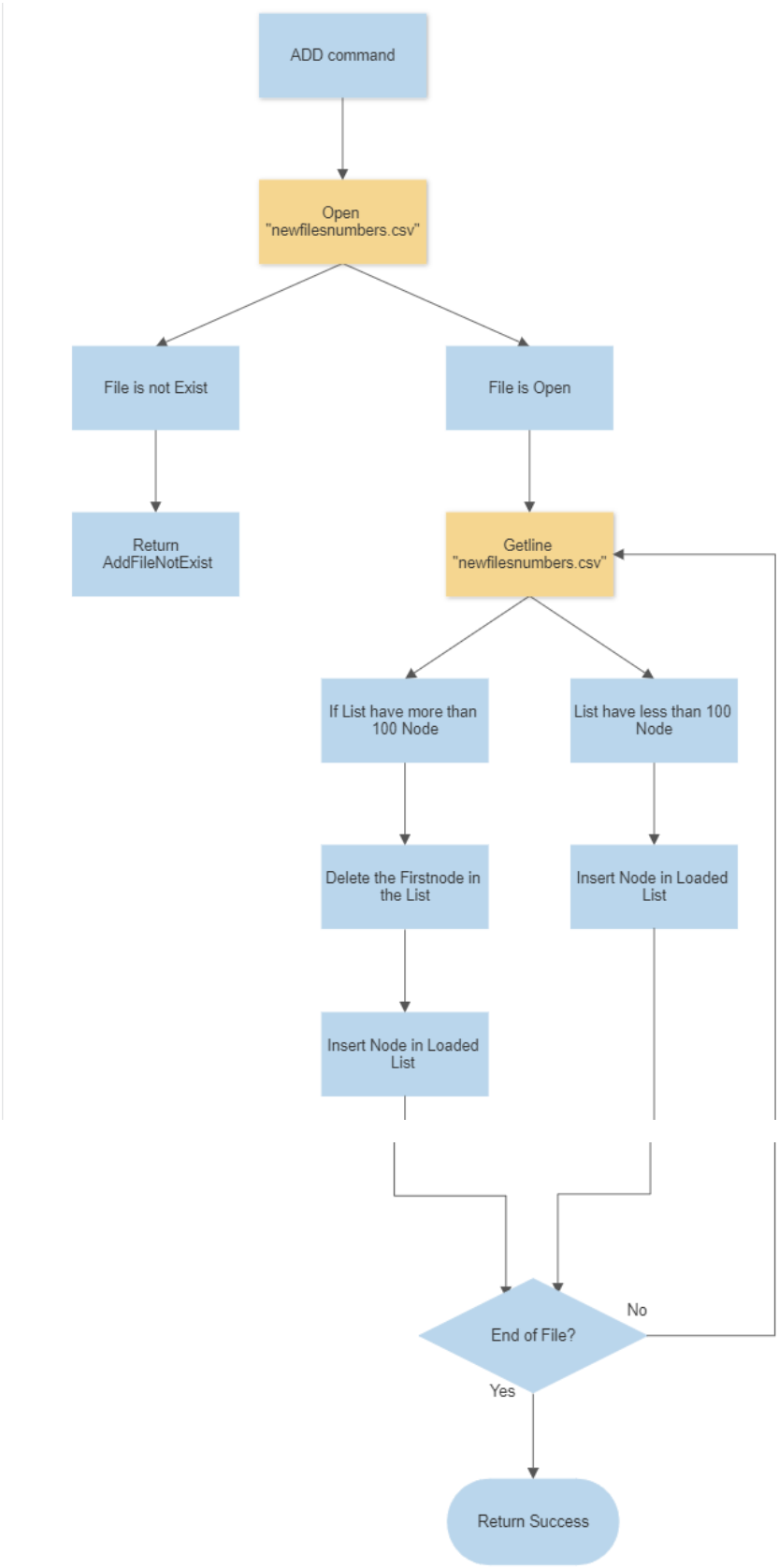
RUN함수는 프로그램을 전체적으로 관리하는 Manager 클래스 내에 존재하며 Command.txt에서 명령을 순차적으로 읽어와 그에 해당하는 동작을 수행한다. Run함수의 Flow chart는 위와 같다.

<LOAD함수의 Flow chart>



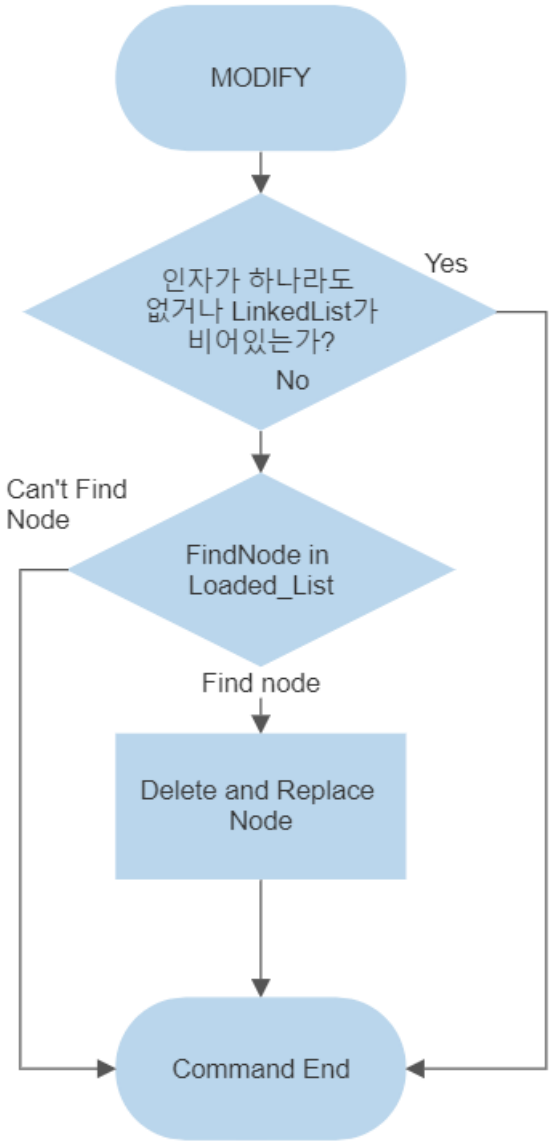
Load함수의 Flow chart는 위와 같다. 먼저 Command에서 LOAD command를 입력 받았다면 Manager에 있는 LOAD 함수로 이동한다. LOAD 함수에서는 "filesnumbers.csv" 파일을 읽어서 만약 파일이 존재하지 않는다면 Result 열거형 중 하나인 LoadFileNotExist를 반환하고 그렇지 않다면 Strtok을 이용하여 디렉토리, 고유번호, 파일 이름을 각각 따로 끊어서 값에 저장하고 Load_List Class에 구현되어 있는 Insert함수를 이용하여 단방향 1차원 Linked List를 구현한다. Insert가 끝났다면 Result 열거형 중 하나인 Success를 반환한다.

<ADD 함수의 Flow Chart>



ADD 함수의 Flow Chart는 다음과 같다. LOAD 함수와 구조가 매우 유사하며, 다른 점은 전체 List 에서 노드의 개수가 100개를 초과한다면 List Class에 존재하는 List에서 제일 먼저 들어온 Node 를 삭제하는 동작을 진행한다. List에서 첫번째로 들어온 노드를 삭제했다면 List Class에 구현된 Insert함수를 수행한다. List Class의 Insert함수는 들어온 인자 중 head에 저장된 디렉토리와 다른 디렉토리가 들어온다면 2차원 Linked List를 생성한다.

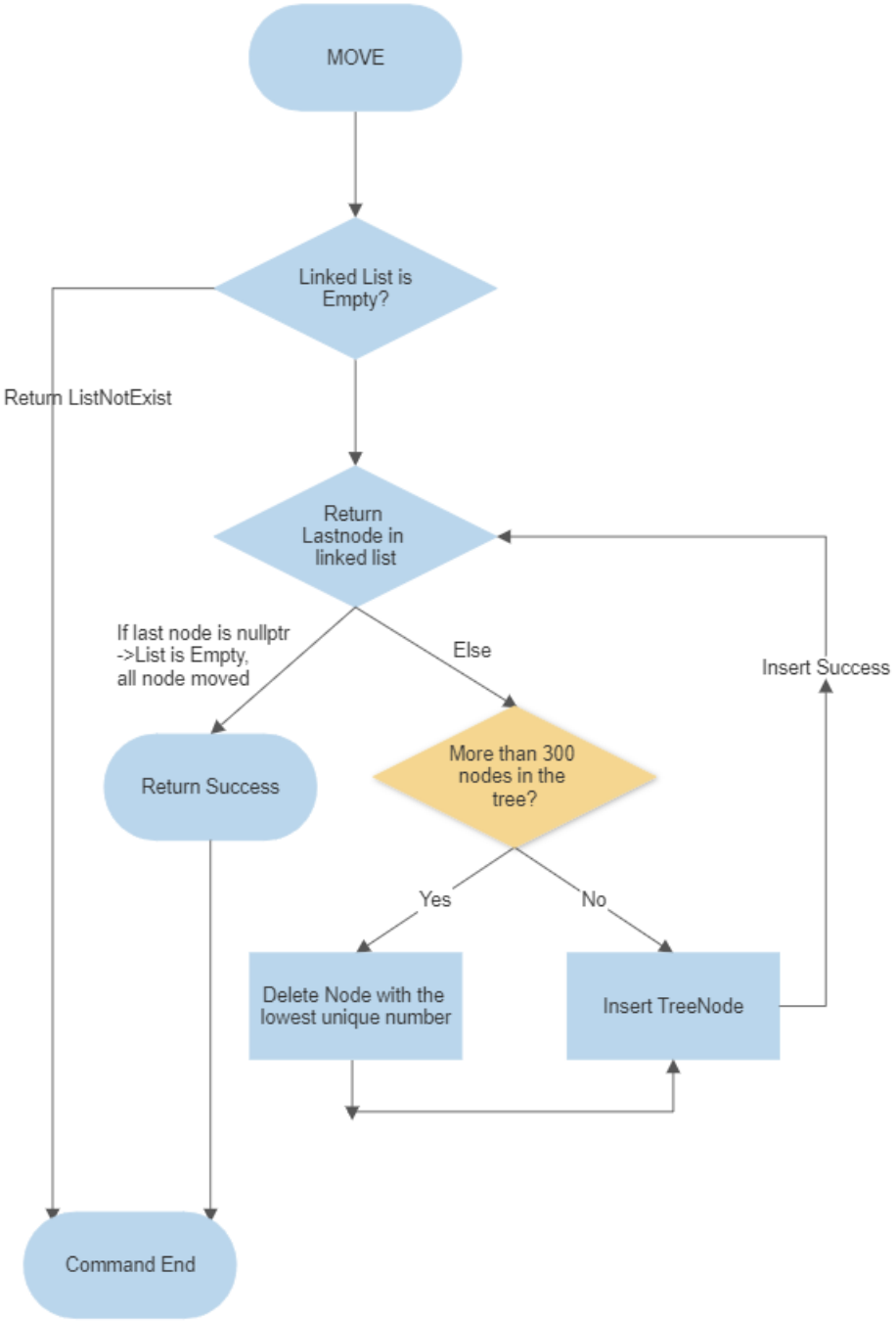
<MODIFY Flow Chart>



MODIFY command의 Flow Chart는 위와 같다. 우선 MODIFY command에는 추가적으로 디렉 이름과 파일 이름, 수정할 고유번호를 인자로 추가로 받는다. 인자가 부족하거나 List가 존재하지 않는다면 커맨드를 종료한다. 위의 인자들을 바탕으로 List를 탐색하면서 노드를 찾는다. FindNode 함수는 bool 값을 return 값으로 가진다. 만약 List에 해당하는 파일 이름이 존재하지 않는다면 ModifyNodeNotExist를 반환하면서 Command를 종료한다. 그렇지 않다면 해당 노드를 삭제하고

고유번호가 수정된 Node를 다시 삽입하는 함수인 DeleteandReplace함수를 수행하고 수행이 완료되었다면 Success를 반환하면서 Command를 종료한다.

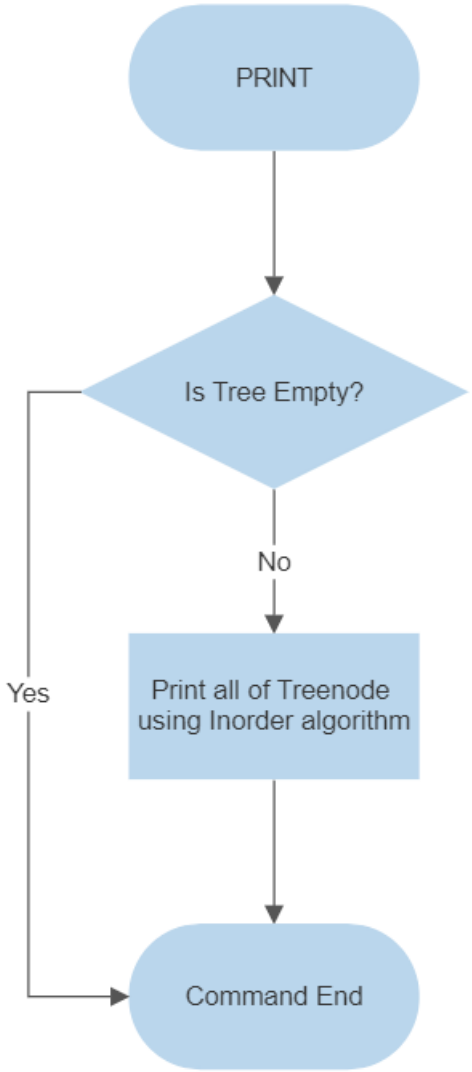
<MOVE Flow Chart>



MOVE command의 Flow chart는 위와 같다. MOVE 명령어는 List의 마지막 노드를 지우고 그 노드를 BST에 옮기는 구조를 띤다. 이를 구현하기 위해서 List Class에 마지막 노드를 지우고 반환해주는 함수인 retlastNode를 구현하여 List에 있는 모든 노드들을 지우고 BST에 옮기는 과정을 진

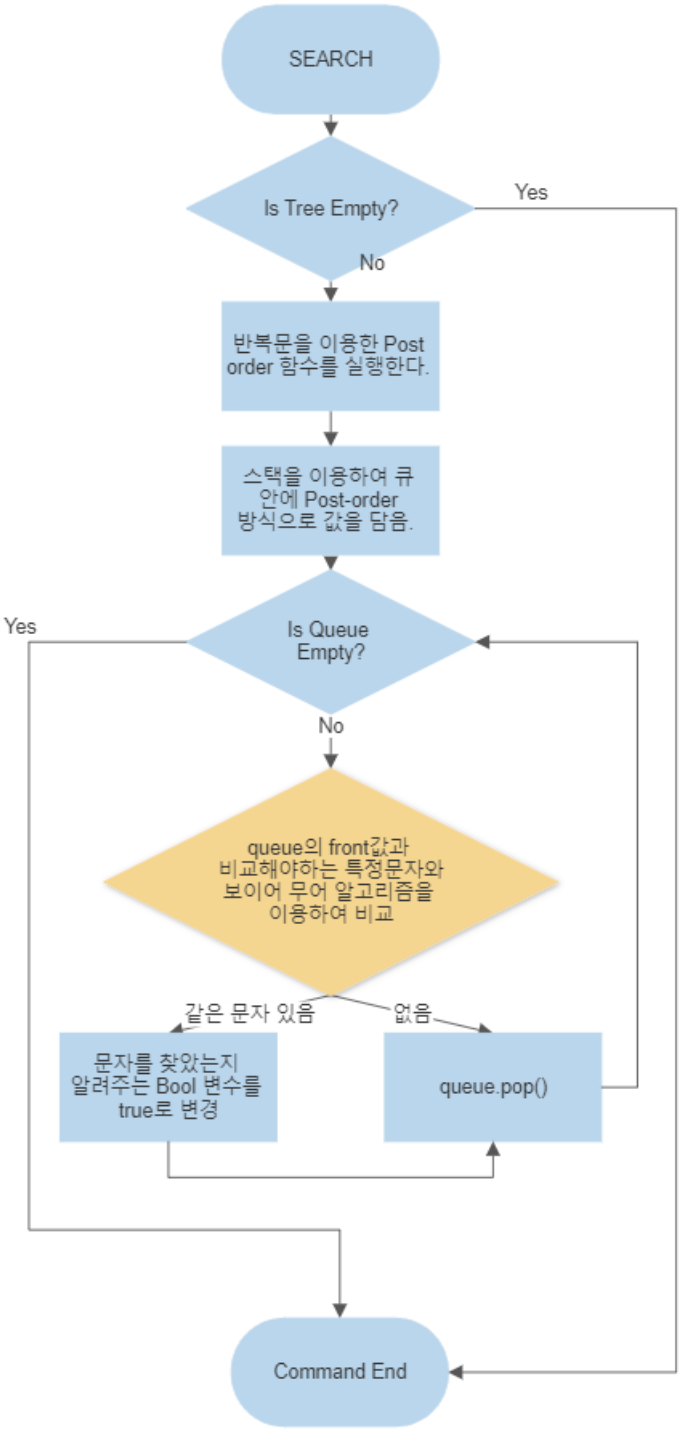
행해 주었다. BST에서 옮기는 과정에서 BST의 최대 노드의 개수가 300개라고 하였으므로 BST에 Insert해주기 전에 300개의 Node가 넘는지를 확인하여 만약 넘는다면 고유번호가 가장 작은 노드를 지우고 Insert를 수행하도록 설계하였다.

<PRINT Flow Chart>



PRINT command의 Flow Chart는 다음과 같다. 우선 PRINT함수에서 Tree가 존재하지 않는다, 즉 루트가 nullptr이라면 에러코드 500을 출력한다. 그렇지 않다면 중위순회 방법을 이용하여 Tree내에 존재하는 모든 노드들을 출력한다. 중위순회를 구현하기 위해서 재귀함수를 이용하였다.

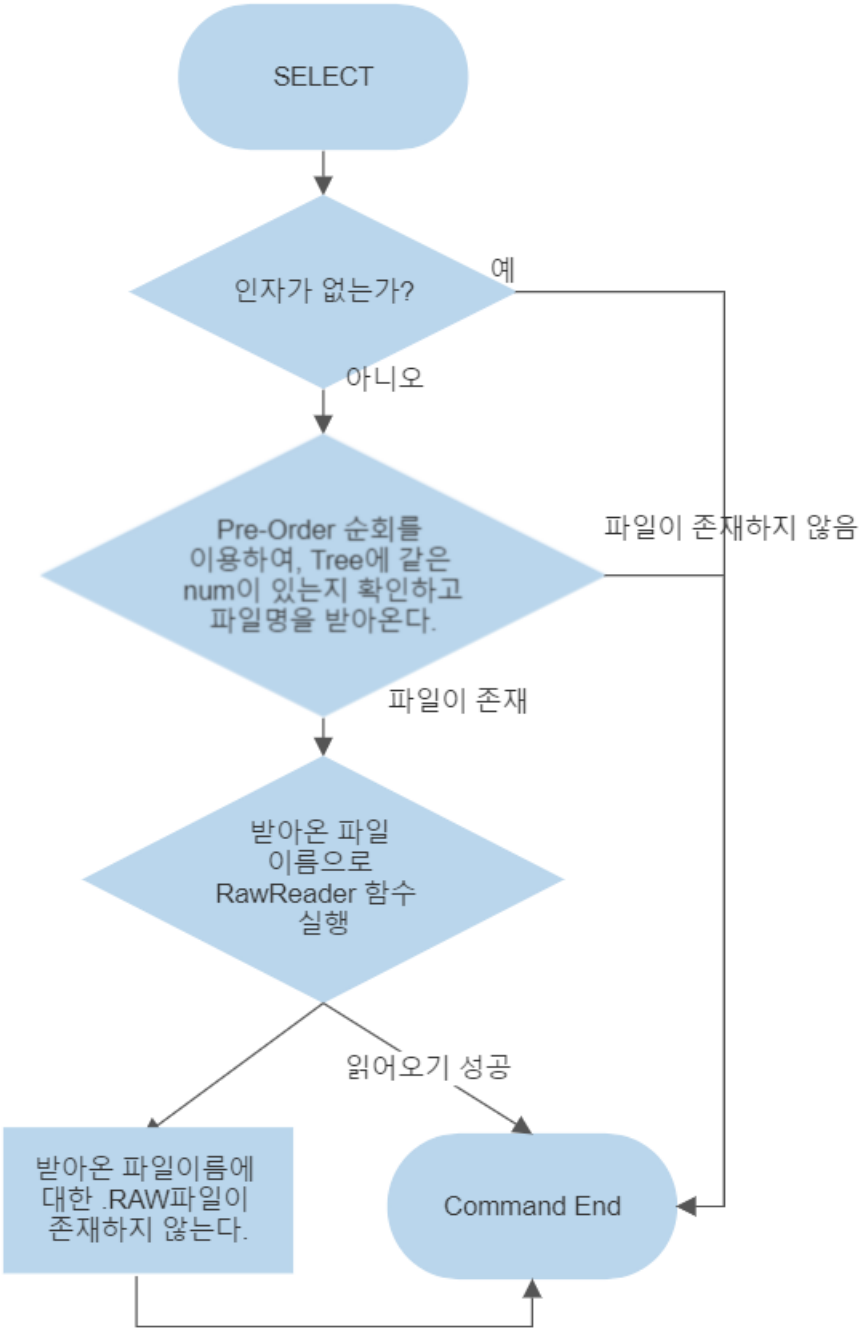
<SEARCH Flow chart>



SEARCH의 Flow Chart는 위와 같다. 만약 Tree가 비어있다면 Command를 종료하며 에러코드를 출력한다. 그렇지 않다면 반복문을 이용한 Post-order 방식을 이용하여 큐에 Post-order방식으로 고유번호와 파일 이름을 담는다. Queue에 BST내에 모든 정보가 Post-order 방식으로 저장되었다면 큐가 비어 있을 때까지 Queue의 front값과 비교해야 하는 특정 문자와 보이어 무어 알고리즘을 이용하여 비교하고 만약 같은 문자를 찾았다면 문자를 찾았는지를 알려주는 Bool 변수를 true

값으로 변경하고 찾은 값의 고유번호와 파일 이름을 출력한다. 큐가 비어 있을 때까지 pop을 해주면서 Queue에 있는 파일이름에 특정문자가 존재한다면, 고유번호와 파일이름을 출력하고, 큐가 비었다면 명령어를 종료한다.

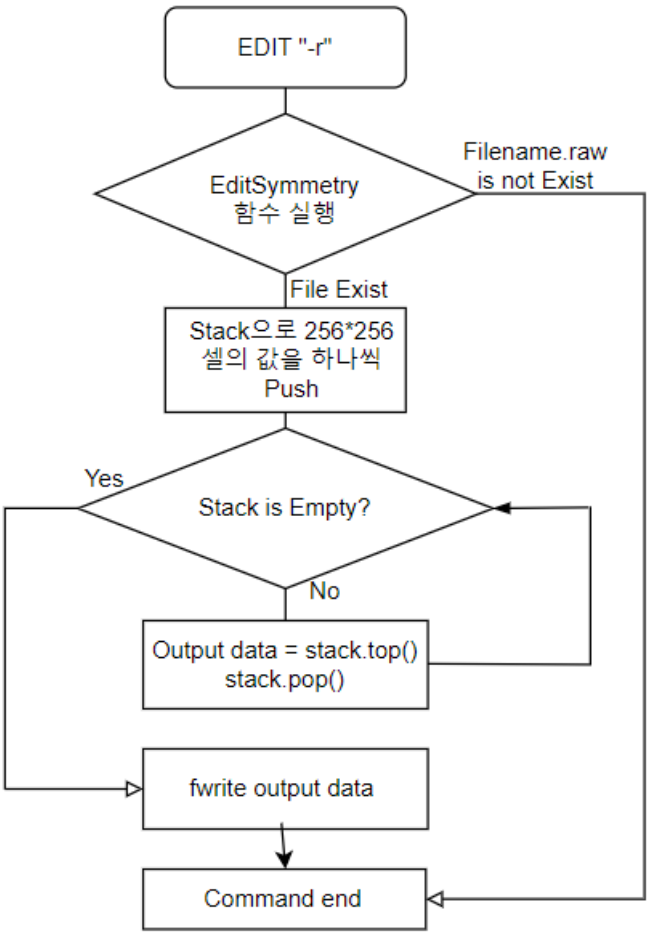
<SELECT의 Flow Chart>



Select 함수의 Flow Chart는 위 사진과 같다. 우선 인자가 없다면 에러코드를 출력하고 명령어를 종료하고 그렇지 않다면, Pre-order 순회를 이용하여 Tree를 탐색하여 같은 숫자가 있는지를 확인하고 파일명을 받아온다. 탐색을 마쳤는데 파일이 존재하지 않는다면 에러코드를 출력하고 명령

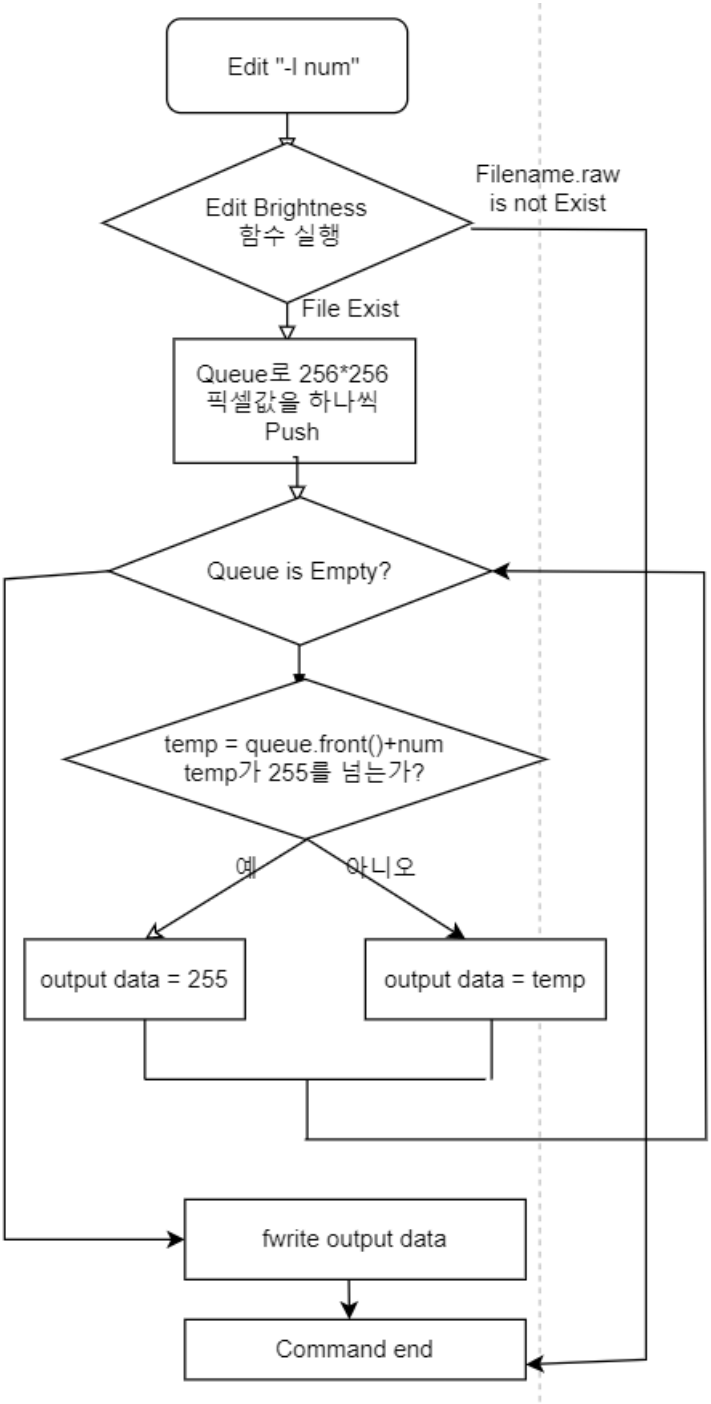
어를 종료한다. 받아온 파일 이름으로 Raw파일을 읽는 RawReader함수를 실행하여 images 폴더에 존재하는 .raw파일을 읽어온 뒤 종료하고, 받아온 파일 이름에 대한 .raw파일이 존재하지 않는다면 파일이 존재하지 않는다는 에러 메시지를 출력 후 종료한다.

<EDIT "-f" Flow Chart>



EDIT Command는 3가지 옵션으로 수행되기 때문에 Flow Chart또한 가독성을 위하여 세가지 따로 작성하도록 하겠다. 우선 이미지의 점대칭을 시켜주는 "-f" 옵션은 Manager에 구현을 하여준 EditSymmetry함수를 실행한다. 만약 Select에서 읽어준 파일이 없다면 오류 메시지를 출력하고 명령어를 종료시킨다. 반면 읽어준 파일을 올바르게 열었다면, 스택에 256*256 셀을 차례로 입력시키고, 스택의 LIFO 구조를 이용하여 output data에 stack.top()의 값을 넣어주고 stack은 pop을 해주는 방식으로 점대칭을 구현하였다. 배열에 점대칭이 완료된 Cell이 담겨있다면 배열을 fwrite로 읽어서 "파일이름_flipped.raw"으로 저장하였다.

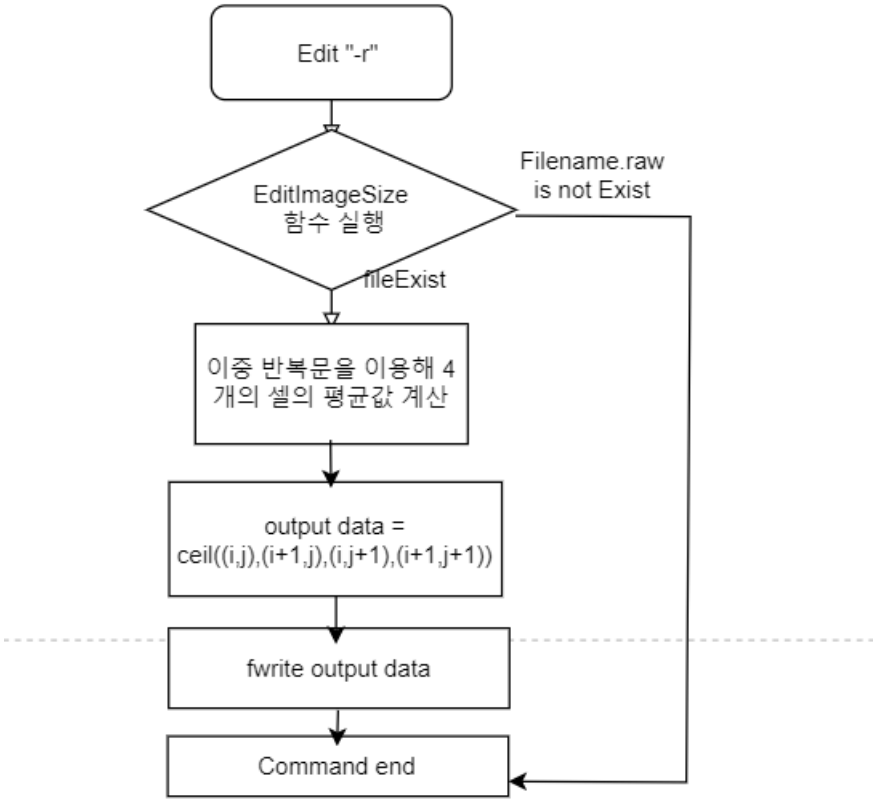
<EDIT "-l num" Flow Chart>



이미지의 밝기를 조정하는 EDIT "-l"의 Flow Chart는 위와 같다. 이미지 밝기는 0부터 255로 고정하였다. 먼저, Command가 EDIT "-l"을 입력받았으면, 추가로 얼만큼의 밝기를 높일 것인지를 알기 위해서 num의 인자를 command.txt에서 읽는다. 그 후 EditBrightness함수를 실행하여 만약 Select에서 읽어온 파일을 올바르게 열지 못하였다면 에러코드를 출력하고 Command를 종료한다.

반면 파일을 올바르게 읽어왔다면 Queue에 256*256 모든 픽셀을 하나씩 입력하고 Queue가 빌 때까지 Pop을 해주며, 반복문을 돌아 temp 변수에 Queue에 Front에 num을 더해서 255가 넘는다면 output data 배열 값에 최대값 255를 저장하고, 그렇지 않다면 Queue의 front에 num을 더해 output data 배열 값에 256*256 픽셀값을 저장한다. 모든 픽셀값을 저장하였다면, fwrite를 이용하여 새로 저장해줄 파일 "파일이름_adjusted.raw"에 저장한 후 Command를 끝낸다.

<EDIT "-r" Flow Chart>



EDIT "-r"의 Flow Chart는 위와 같다. 이미지 크기를 조정하는 EDIT "-r" 명령어는 입력된 이미지의 크기를 1/4로 축소하는 동작을 진행한다. 우선 EditImageSize함수를 실행하여 SELECT해온 파일이 존재하지 않는다면, Command를 끝내주고, 올바르게 파일을 읽어왔다면 이중 반복문을 돌아 index를 2씩 증가시켜 준 뒤 사각형의 셀의 평균값을 내주어 반올림하여 output data에 저장하였다. 반올림해주는 함수는 cmath 라이브러리의 ceil함수를 이용해주었다. 256*256 셀의 값이 올바르게 저장되었다면 "파일이름_resized.raw"의 파일이름으로 output data를 fwrite해준 뒤 Command를 끝낸다.

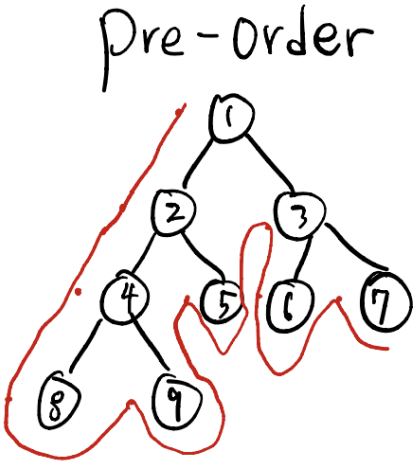
Algorithm

이번 프로젝트에서 사용한 알고리즘은 크게 2가지로 나눌 수 있다. 순회알고리즘과 문자열에서 특정 문자열을 검색하는 보이어 무어 알고리즘이 그것이다.

그 중 순회알고리즘에 대해서 알아보도록 하겠다. 순회알고리즘에는 전위 순회(Pre-Order), 중위 순회(In-Order), 후위 순회(Post-Order)로 나누어진다.

먼저 전위 순회는 부모노드->왼쪽 자식노드->오른쪽 자식 노드로 탐색하는 순회 방법이다. 트리를 생각할 때 우리는 Root와 왼쪽 Subtree, 오른쪽 Subtree로도 생각 할 수 있다. 따라서 Root 탐색, 왼쪽 Subtree의 root 탐색, 왼쪽 Subtree의 Subtree의 root 탐색, 그리고 난 뒤 왼쪽 자식이 하나인 Subtree, 즉 Leafnode를 가지는 Subtree를 만나면 왼쪽 자식 노드를 탐색하고, 그 왼쪽 자식 노드가 leafnode인 Subtree의 오른쪽 Subtree를 같은 방식으로 탐색하는 방법이다.

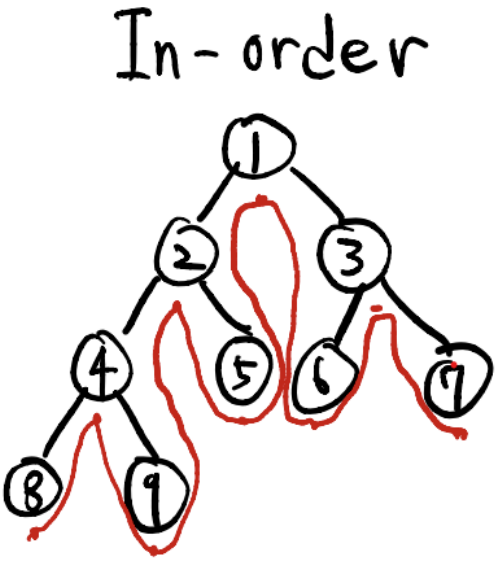
부모노드, 즉 Root 노드부터 탐색을 시작하여 Root노드를 먼저 탐색하고, 왼쪽 SubTree의 왼쪽 자식 노드를 순차적으로 방문하고, Root노드의 오른쪽 SubTree의 왼쪽 자식 노드를 순차적으로 방문하는 알고리즘이다. 전체적으로 생각하자면, 루트노드->왼쪽 자식노드->오른쪽 자식노드로 방문하는 것으로 생각할 수 있다. 이해하기 쉽게 그림을 그려 설명해보도록 하겠다.



위의 사진은 전위 순회의 탐색 경로이다. 따라서 위의 사진에서와 같은 Tree에서는 1->2->4->8->9->5->3->6->7 순서로 순회가 이루어진다.

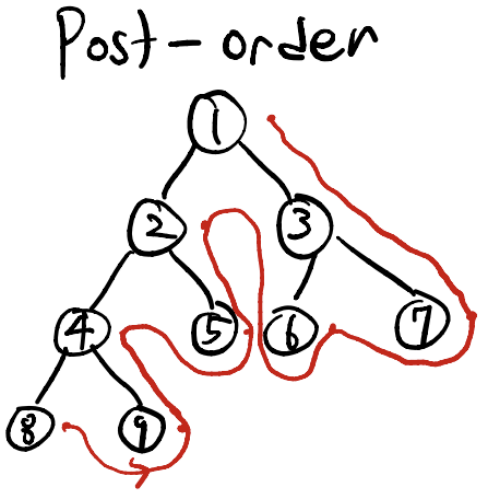
다음은 중위순회 방법이다. 중위순회는 왼쪽 자식 노드->부모노드->오른쪽 자식 노드로 탐색하는 알고리즘이다. 쉽게 설명하기 위해서 풀어서 설명하자면, Root에서 먼저 왼쪽 자식 node가

leafnode인 SubTree로 이동을 한 뒤 그 Subtree의 왼쪽 자식 노드, Subtree의 부모노드를 방문하고 왼쪽 자식 node가 leafnode인 Subtree의 오른쪽 자식 노드가 leafnode, 즉 Subtree가 아니라면 오른쪽 자식 노드도 방문을 하고, 그렇지 않다면 오른쪽 Subtree로 이동하여 위와 같은 방법을 똑같이 진행하면서 순회하는 구조이다. 그림을 통해서 중위 순회의 예시를 알아보도록 하겠다.



위의 사진과 같은 예시에서는 8->4->9->2->5->1->6->3->7 순서로 순회가 이루어진다.

마지막으로 후위순회(Post-Order) 방식이다. 후위 순회 방식은 큰 틀에서 보면 왼쪽 자식 노드-> 오른쪽 자식 노드->부모노드로 순회가 이루어진다. 앞서 설명한 두가지 방식으로 설명하자면 왼쪽 자식 노드가 Leaf node인 Subtree로 이동하여 왼쪽 leaf node를 방문하고, 오른쪽 자식 node가 leaf node라면 오른쪽 leaf node를 방문한다. 그리고 Subtree의 root 노드를 방문하고 난 뒤 다시 Subtree의 부모노드의 오른쪽 Subtree 또한 같은 방식으로 순회하는 방식이다. 방문 순서는 아래 그림과 같다.

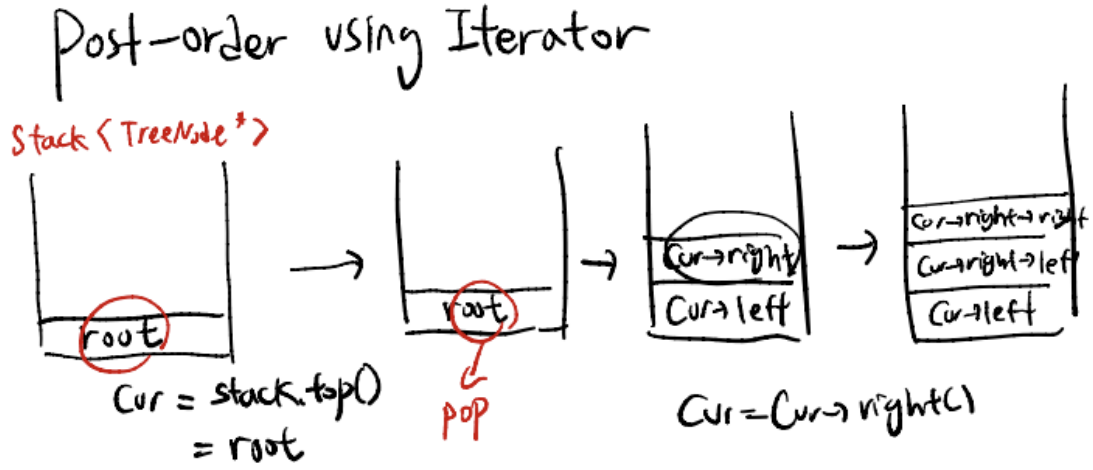


위의 사진과 같은 예시에서는 중위 순회로 순회를 하면 8->9->4->5->2->6->7->3->1 순서로 방문이 이루어진다.

이번 프로젝트에서는 순회 알고리즘을 반복문을 이용한 방법, 재귀함수를 이용한 방법 2가지 방법으로 순회 알고리즘을 구현한다. 먼저 재귀함수를 이용한 알고리즘은 인자로 node를 받아온다. 여기서 인자는 Tree나 Subtree의 루트를 받아온다. Node가 nullptr이 아니라면 Preorder(node->left), Preorder(node->right)로 왼쪽 Subtree와 오른쪽 Subtree를 재귀적으로 실행시켜준다.

재귀함수를 이용한 순회 방식의 장점은 우선 코드가 직관적으로 이해하기 쉽다는 장점이 있다. 반면 단점으로는 재귀함수를 사용하기 때문에, 재귀함수는 node의 개수가 많아질수록 시작 복잡도가 빠른 상승폭으로 증가하는 단점이있다.

반면 반복문을 이용한 순회 알고리즘은 스택과 큐를 이용한다. 스택에 자료형을 Treenode로 지정하고 스택에 먼저 root를 넣는다. 반복문을 돌면서 stack이 빌 때까지 반복을 한다. 반복문 안에서는 stack의 top을 반환하여 그 값을 큐에 넣고, 후위순회 방식으로 stack에 값을 집어넣는다. 스택이 선입선출(LIFO) 구조라는 점을 이용한 것이다. 모든 반복문을 지나고 나면 Queue에 Post-order 형태로 값이 저장된 것을 확인할 수 있다. 스택에 저장되는 구조는 다음 사진과 같다.

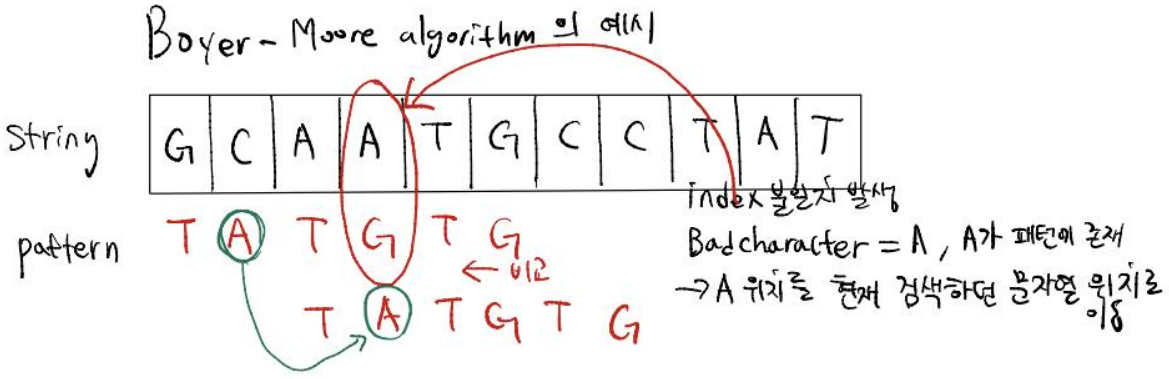


반복문을 이용한 순회 알고리즘의 장점은 시간복잡도가 재귀함수를 이용한 순회방식보다 상대적으로 낮은 장점이 있다.

다음은 문자열에서 특정 문자열을 검색하는 보이어 무어 알고리즘에 대해서 알아보도록 하겠다. 보이어 무어 알고리즘은 불필요한 검색은 건너뛰고, 특정 문자열의 검색을 빠르게 하는 것을 목표로 한다. 보통 상황에서 문자열은 앞부분보다는 뒷부분에서 불일치가 일어날 확률이 높다는 성질때문에 오른쪽 끝에서부터 비교를 한다.

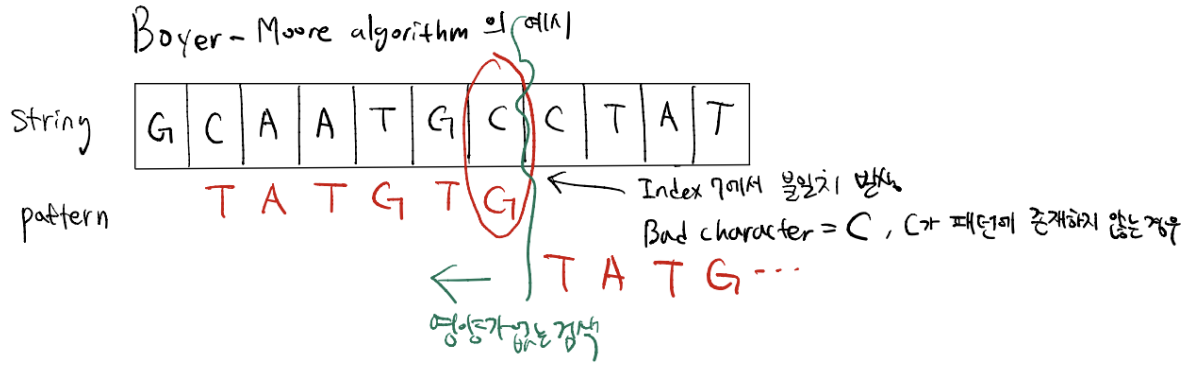
보이어 무어 알고리즘에서 Bad Character란 검색패턴의 현재 문자와 일치하지 않는 텍스트의 문자이다. 탐색 과정에서 크게 Bad Character가 패턴에 존재하는 경우와 Bad Character가 패턴에 존재하지 않는 경우로 나눌 수 있다. 그림을 통해서 설명하면 다음과 같다.

먼저 Bad Character가 패턴에 존재하는 경우이다.



위 그림에서는 문자열의 위치를 2만큼 이동시켜서 다시 비교해준 것을 확인할 수 있다.

다음은 Bad Character가 패턴에 존재하지 않는 경우이다.



위 그림은 BadCharacter가 존재하지 않은 case이므로 인덱스 7까지는 영양가 없는 검색이기 때문에 Index를 6만큼 증가시킨 것을 확인할 수 있다.

Result Screen

먼저 LOAD명령어일 때 filesnumbers.csv파일이 존재하지 않을 때 오류를 출력하고, LOAD가 성공적으로 되었다면, Load된 List를 Print해주는 명령어이다. 또한, 노드의 개수가 100개가 넘을 경우 먼저 들어왔던 노드를 삭제 후 새로운 데이터를 추가한다.

먼저 LOAD가 성공적으로 되었다면, Load된 List를 Print해주는 결과화면이다.

```
=======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
the building is like a pyramid/333
river is under the bridge/400
The house has windows and doors/444
the boat sails a big river/500
The celebrity posed for the picture/555
there are lots of ariplane/600
The woman is looking at the man/666
lena is famous person/700
the woman is wearing a hat/777
there are cars and people/800
airplane fly on the mountain/888
there are numbers and chinese characters/900
fence is near trees/999
====================
```

다음은 filesnumbers.csv파일이 존재하지 않을 때 오류를 출력하는 화면이다.

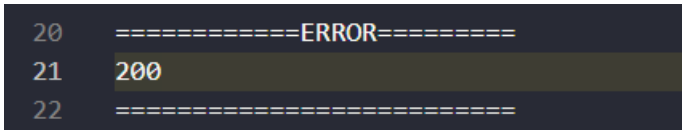
```
1   ============ERROR=========
2   100
3   ==========================
```

다음은 ADD 명령어의 결과화면이다. ADD 명령어는 Loaded_List에 새로운 디렉토리의 파일 데이터들을 추가한다.

먼저, ADD가 올바르게 되었을 때 결과화면은 다음과 같다.

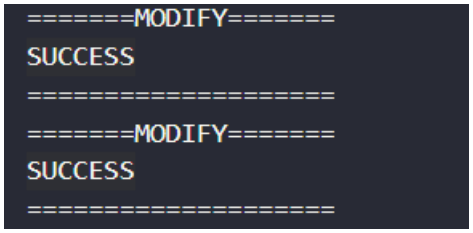
```
20   ==========ADD==========
21   SUCCESS
22   ======================
23   ==========ADD==========
24   SUCCESS
25   ======================
```

다음은 인자가 하나라도 존재하지 않거나 Loaded_List가 존재하지 않는 경우 에러코드를 출력하는 결과화면이다.

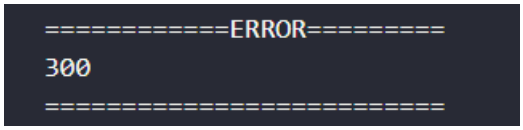


다음은 MODIFY 명령어의 결과화면이다.

우선 Modify가 성공적으로 수행되었을 때의 결과화면이다.

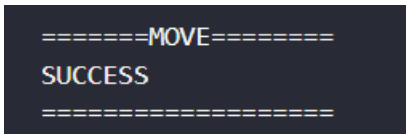


다음은 인자가 하나라도 없거나 파일 이름에 대한 노드가 존재하지 않거나 중복되는 고유번호로 할당할 경우 에러코드를 출력하는 결과화면이다.

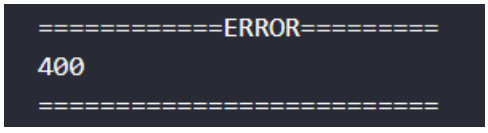


다음은 MOVE 명령어의 결과화면이다. MOVE는 Loaded_List의 정보들을 BST에 옮기는 명령어이다.

먼저, MOVE 명령어가 성공적으로 수행되었을 때의 결과화면이다.



다음은 Loaded_List에 노드가 존재하지 않을 경우 에러코드를 출력한 결과화면이다.



다음은 Print명령어의 결과화면이다.

먼저, 중위 순회 방식의 탐색을 이용하여 정보를 출력하는 결과화면이다.

```
========PRINT========
new_new_files / there are lots of people in the car / 80
img_files / there are lots of people in the park / 100
new_new_files / the man is nice / 102
new_files / the man is nice / 105
img_files / the man is gorgeous / 111
img_files / the woman is smiling / 200
new_new_files / the man takes a car / 202
new_files / there are lots of people in the car / 210
img_files / the man takes a picture / 222
new_new_files / three peppers are small / 303
img_files / the building is like a pyramid / 333
new_new_files / the building is like a 63 building / 401
img_files / river is under the bridge / 430
img_files / The house has windows and doors / 444
img_files / the boat sails a big river / 500
new_files / the man takes a car / 501
img_files / The celebrity posed for the picture / 555
img_files / there are lots of ariplane / 600
new_files / three peppers are small / 601
img_files / The woman is looking at the man / 666
img_files / lena is famous person / 700
new_files / the building is like a 63 building / 701
img_files / the woman is wearing a hat / 777
img_files / there are cars and people / 800
img_files / airplane fly on the mountain / 888
img_files / there are numbers and chinese characters / 900
img_files / fence is near trees / 999
=====================
```

다음은 BST가 존재하지 않을 때 에러코드를 출력하는 결과화면이다.

```
 ============ERROR=========
 500
 ==========================
```

다음은 SEARCH 명령어의 결과화면이다.

Search가 성공하였을 때의 결과화면이다. 가독성을 좋게 하기 위해서 출력 form을 조금 변경하였다.

```
=======SEARCH========
Search word is "on the"

"airplane fly on the mountain"  888
======================
```

```
=======SEARCH========
Search word is "is"

"the building is like a 63 building"  401
"the building is like a 63 building"  701
"fence is near trees"  999
"the woman is wearing a hat"  777
"lena is famous person"  700
"The woman is looking at the man"  666
"river is under the bridge"  430
"the building is like a pyramid"  333
"the man is nice"  102
"the man is nice"  105
"the woman is smiling"  200
"the man is gorgeous"  111
======================
```

```
=======SEARCH========
Search word is "asdk"

======================
```

위의 사진은 검색을 실패했을 때의 결과화면이다.

다음은 인자가 없거나 BST가 비어있는 경우 에러를 출력하는 결과화면이다.

```
============ERROR=========
600
==========================
```

다음은 Select 명령어이다. 먼저, 올바르게 Select 되었을 때 결과화면이다.

```
=======SELECT========
SUCCESS
====================
```

다음은 에러코드 출력이다. 인자가 없거나 파일 고유번호가 존재하지 않을 경우 에러를 출력한다.

```
============ERROR=========
700
==========================
```

마지막으로 Edit 명령어이다. 올바르게 Edit 되었을 때 결과화면은 다음과 같다.

```
=========EDIT==========
SUCCESS
=======================
=========EDIT==========
SUCCESS
=======================
=========EDIT==========
SUCCESS
=======================
```

다음은 인자가 존재하지 않는 경우 에러코드를 출력하는 것이다.

```
============ERROR=========
800
==========================
```

다음 사진은 이미지가 올바르게 편집되었는 지 확인하는 것이다.

처음 사진부터 images 폴더에서 불러온 파일, 밝기 조정, 점대칭, 크기조정을 한 화면이다.

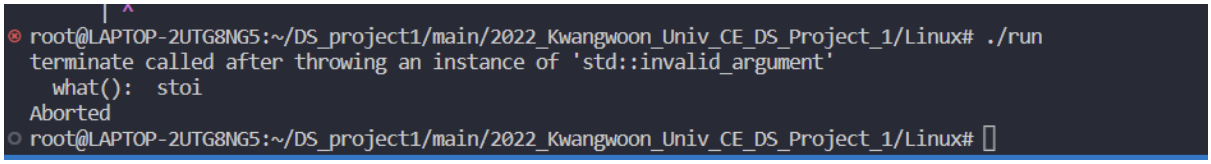
마지막으로 EXIT 명령어의 결과화면이다.

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

Consideration

먼저, 객체지향프로그래밍에서는 Visual studio를 사용하며 컴파일이 상대적으로 간편하였지만, 데이터구조 프로젝트에서는 makefile을 이용하여 컴파일을 하였기 때문에 이전에 구현하던 것보다 초반에 감을 잡는 것이 힘들었다. 우선 메인 함수에서는 매니저 클래스에 객체를 선언하여 run함수를 통해서 command.txt를 읽어서 명령어를 읽어서 프로그램을 수행하는 구조로 설계하였다. 우선 LOAD명령어에서 처음으로 LOAD를 구현할 때에는 string으로 getline을 통해서 substr, find 함수 등을 이용하여 파일을 한 줄씩 읽어오는 구조로 설계를 하였다. 이러한 방식으로 코드를 구현하니 string에서 int형으로 바꾸어 줄 때 stoi 함수를 사용하였는데, make를 통해서 compile을

했더니 아래와 같은 문제가 발생하였다.



위의 문제를 해결하기 위해서 command.txt를 char배열로 getline을 하는 방식으로 코드를 수정하였다. 수정한 뒤, make를 통해서 컴파일을 진행하보니, 이번엔 segmentation fault가 발생하는 문제가 있었다. 보통 줄 단위를 읽어 strtok을 활용하는데, 읽어온 줄이 공백이면 null값을 읽어와 segmentation fault가 발생하는 문제였다. 이 문제를 해결하기 위해서 파일이 끝났으나 한번 더 읽으려고 할 때와 파일에 공백줄이 존재할 때인 경우를 예외처리를 해주어 문제를 해결하였다.

다음은 MODIFY Command를 구현할 때의 Segmentation fault가 발생하는 오류가 있었다. 이는 MODIFY를 해주는 과정에서 노드를 지우고 다시 Insert해주는 과정이 있는데 여기서 메모리 연결이 잘못되어서 발생하는 오류로 보여졌다. 오류처럼 보여지는 코드에 cout을 통해서 어느 지점에서 Segmentation fault가 발생하는지를 확인하는 방식으로 오류를 잡았다. 문제에 발생 원인은 지운 노드의 이전 노드의 next를 nullptr로 설정해 주지 않아서 생기는 문제였다. Deletenode의 이전 node의 next를 nullptr로 설정해주고 나니 문제를 해결하였다.

다음은 SELECT 명령어의 경로 설정 문제이다. 우선 프로젝트에서 경로 설정을 어디에 해주는지가 애매하게 작성이 되어있어서 어디에 Images file폴더를 .cpp, .h 파일이 있는 폴더에 넣어서 경로 설정을 해주었다. 따라서 상대경로로 ./images/파일이름.raw로 설정해주니 경로 설정을 올바르게 할 수 있었다. 또한 EDIT 명령어에서 파일을 쓸 때에는 경로를 main폴더에 넣어주었다.

다음은 EDIT 명령어에서 raw파일이 올바르게 수정되었는지를 확인하기 위해서 Window에서의 raw extension를 설치하였는데, raw파일이 열리지 않았다. 구글링을 통해서 knRaw 프로그램을 통해서 raw파일을 열어주었다.

다음은 #include해주는 과정에서 헤더파일이 정의되지 않았다는 에러가 발생하였다. 구글링을 통해서 문제점이 무엇인지를 확인해보니 헤더파일이 여러 번 정의가 되어서 컴파일러가 헤더파일을 인식하지 못하는 것이었다. 이 문제를 해결하기 위해서 #ifndef를 사용하여 문제를 해결하였다. #ifndef란 만약 헤더파일이 정의되지 않았다면 #define을 통해서 정의한 뒤 #endif로 마무리해주는 것이다. 만약 헤더파일이 정의되었다면 #endif로 이동하여 여러 번 정의되는 문제가 발생하지 않게 해주는 것이다.

마지막으로, 수업시간에는 VMware, 가상머신을 통해서 우분투를 설치하였다. 그러나 가상머신으로 Ubuntu를 작동하려니 가상머신에서 편집기를 통해서 코딩을 해야하는데, 프로그램에 버퍼링이 심하여 code를 작성하는 것이 쉽지 않았다. 교수님께서 수업시간에 Window 내에서도 우분투 설치가 가능하다는 말을 듣고, Window in Linux를 설치하여 Window 내에서 우분투를 설치하고, Vscode와 우분투를 연결해주어서 Linux환경으로 코딩을 하였다. 그런데, 설치한 wsl 우분투가

20.04버전을 설치하였는데, 데이터구조실습에서는 18.04버전을 이용하여 작동을 확인해야 했다. 따라서 VMware, 가상머신에서 우분투 18.04버전을 설치하여 terminal에서 작동이 되는지를 확인하였다. 확인한 결과화면은 다음과 같다. 다음 프로젝트를 진행할 때에는 18.04 환경에서 프로젝트를 구현해야 겠다고 생각하였다.

```
=======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
the building is like a pyramid/333
river is under the bridge/400
The house has windows and doors/444
the boat sails a big river/500
The celebrity posed for the picture/555
there are lots of ariplane/600
The woman is looking at the man/666
lena is famous person/700
the woman is wearing a hat/777
there are cars and people/800
airplane fly on the mountain/888
there are numbers and chinese characters/900
fence is near trees/999
====================
==========ADD==========
SUCCESS
======================
==========ADD==========
SUCCESS
======================
```

```
=======MODIFY=======
SUCCESS
====================
=======MODIFY=======
SUCCESS
====================
=======MOVE========
SUCCESS
==================
========PRINT========
new_new_files / there are lots of people in the car / 80
img_files / there are lots of people in the park / 100
new_new_files / the man is nice / 102
new_files / the man is nice / 105
img_files / the man is gorgeous / 111
img_files / the woman is smiling / 200
new_new_files / the man takes a car / 202
new_files / there are lots of people in the car / 210
img_files / the man takes a picture / 222
new_new_files / three peppers are small / 303
img_files / the building is like a pyramid / 333
new_new_files / the building is like a 63 building / 401
img_files / river is under the bridge / 430
img_files / The house has windows and doors / 444
img_files / the boat sails a big river / 500
new_files / the man takes a car / 501
img_files / The celebrity posed for the picture / 555
img_files / there are lots of ariplane / 600
new_files / three peppers are small / 601
img_files / The woman is looking at the man / 666
img_files / lena is famous person / 700
new_files / the building is like a 63 building / 701
img_files / the woman is wearing a hat / 777
img_files / there are cars and people / 800
img_files / airplane fly on the mountain / 888
img_files / there are numbers and chinese characters / 900
img_files / fence is near trees / 999
====================
=======SEARCH========
Search word is "on the"

"airplane fly on the mountain"  888
======================
```

```
=======SEARCH========
Search word is "is"

"the building is like a 63 building"  401
"the building is like a 63 building"  701
"fence is near trees"  999
"the woman is wearing a hat"  777
"lena is famous person"  700
"The woman is looking at the man"  666
"river is under the bridge"  430
"the building is like a pyramid"  333
"the man is nice"  102
"the man is nice"  105
"the woman is smiling"  200
"the man is gorgeous"  111
=======================
=======SEARCH========
Search word is "there"

"there are numbers and chinese characters"  900
"there are cars and people"  800
"there are lots of ariplane"  600
"there are lots of people in the car"  210
"there are lots of people in the car"  80
"there are lots of people in the park"  100
=======================
=======SELECT========
SelectFile is "fence is near trees"

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