

## Introduction

이번 프로젝트는 크게 보자면 `c++` 언어의 전반적인 내용을 복습하며 기초를 다지고 기본적인 자료 구조를 구현하여 잘 구동 되는지 확인하는 프로젝트이다. 자막 관리 프로그램을 설계 하는 데에 있어서 큐, 이진 트리, 연결 리스트를 사용한다.

자료 구조를 구현하는 데에 가장 기본이 되는 연결 리스트의 원리와 연결 리스트가 `return` 하는 값이 무엇인지 파악하고 그 값을 잘 이용할 줄 알아야 하고, 큐의 데이터가 정수 값이 아닌 문자 값으로 이루어져 있어 좀 더 까다로운 구현이 될 것이다. 또한 그 데이터를 커맨드를 통해 삽입하는 것이 아닌 텍스트 파일에서 읽어서 큐를 구현하므로 파일 입출력에 대한 지식도 있어야 한다.

트리 중 가장 기본이라고 할 수 있는 이진 트리를 구현할 때에는 큐에서 `pop`한 값을 이용하여 그 값을 이진 트리로 만드는데, 이때 이진 트리의 구조와 만들어지는 원리를 파악해야 하고, 스트링 타입으로 이루어진 데이터를 잘 분리하여 시간을 비교해야 하므로 이것에 관한 로직 또한 잘 생각을 하여 구현하는 게 첫 번째 중요 포인트라고 생각한다.

`Section` 함수는 구현할 때 단순 연결리스트로 만들지만, `Section`을 구분하는 노드와 `Subtitle`을 구분하는 노드가 있으니 이 둘을 잘 엮어서 어떻게 `Section`을 만들 것인지가 두 번째 중요 포인트이다. 그리고 `프린트` 함수가 인자를 받을 때와 안 받을 때로 구분이 되어 작동을 하니, 함수 오버로딩에 관한 지식도 있어야 할 것이다.

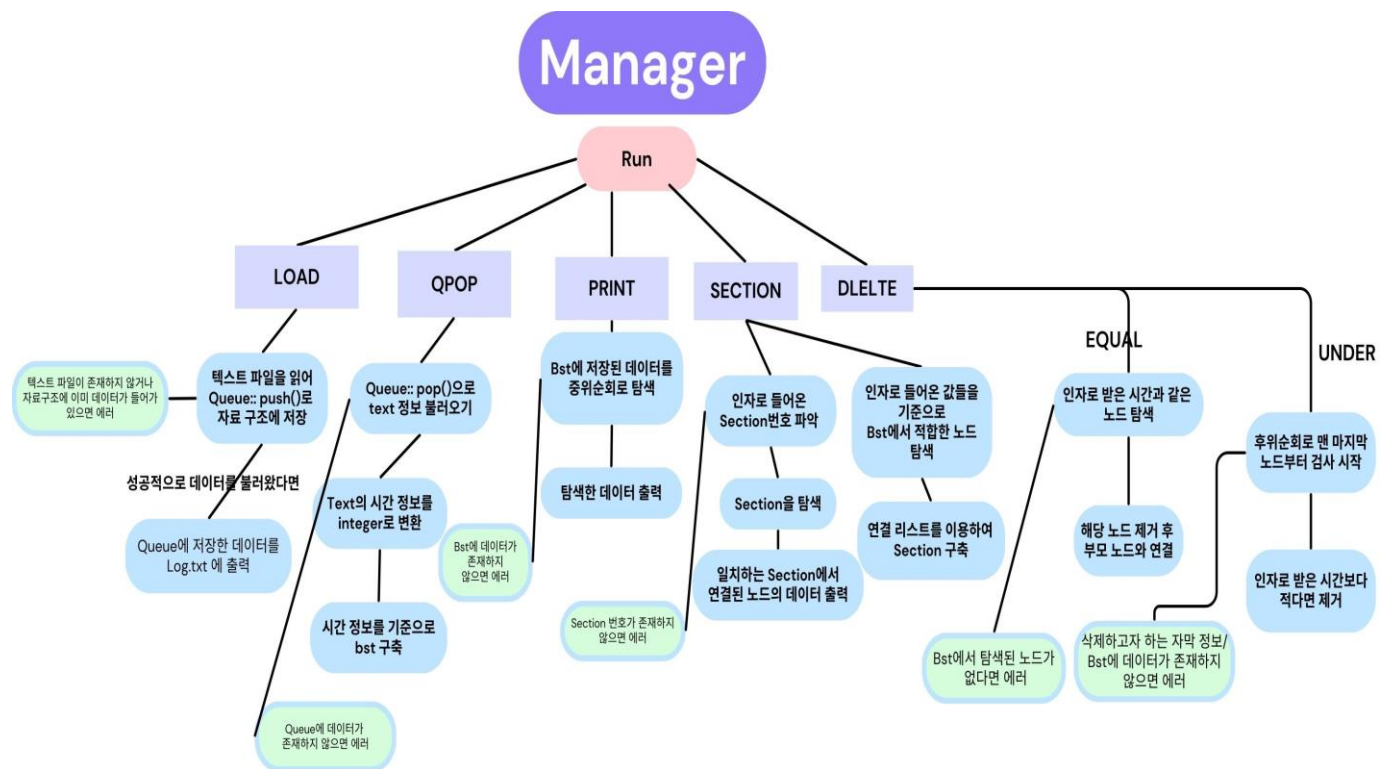
`Delete`를 구현할 때에는 특정 노드 삭제와 특정 시간 미만 노드 삭제. 두 가지 경우가 있는데, 특정 노드 삭제는 비교적 단순하게 구현할 수 있지만, 특정 시간 미만 노드 삭제는 어떤 순서로 순회를 하며 삭제를 해 모든 경우에 대비할 수 있는지, 오류가 나지 않게 만드는 것이 중요하다.

프로그램을 관리하는 `Manager` 클래스를 통해 이 자료구조 클래스와 명령어, 함수들을 한 번에 다루는데, `Manager` 클래스에서 어떻게 이 클래스들을 유기적으로 엮어서 프로그램을 돌아가게 하는 지가 세 번째로 중요한 포인트이다.

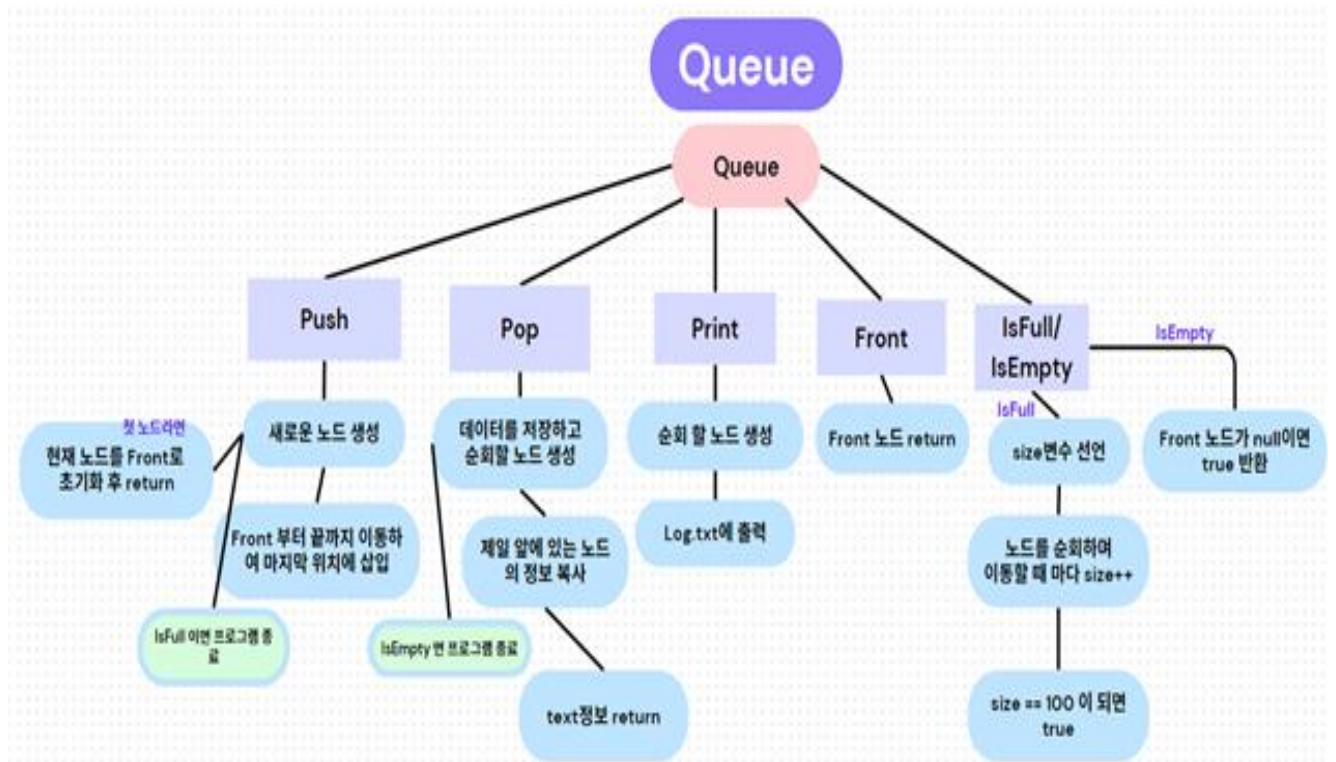
마지막으로 자료구조를 구현함에 있어 제일 중요하고 요긴하게 쓰이는 재귀함수를 사용할 줄 아는 것이 마지막 중요한 포인트이다. 결국 구현만 잘 할 줄 안다고 되는 것이 아닌 데이터의 흐름을 알고 프로그램의 데이터 구조를 파악하여 구현하는 것이 중요한 프로젝트라고 말 할 수 있다.

## FlowChart

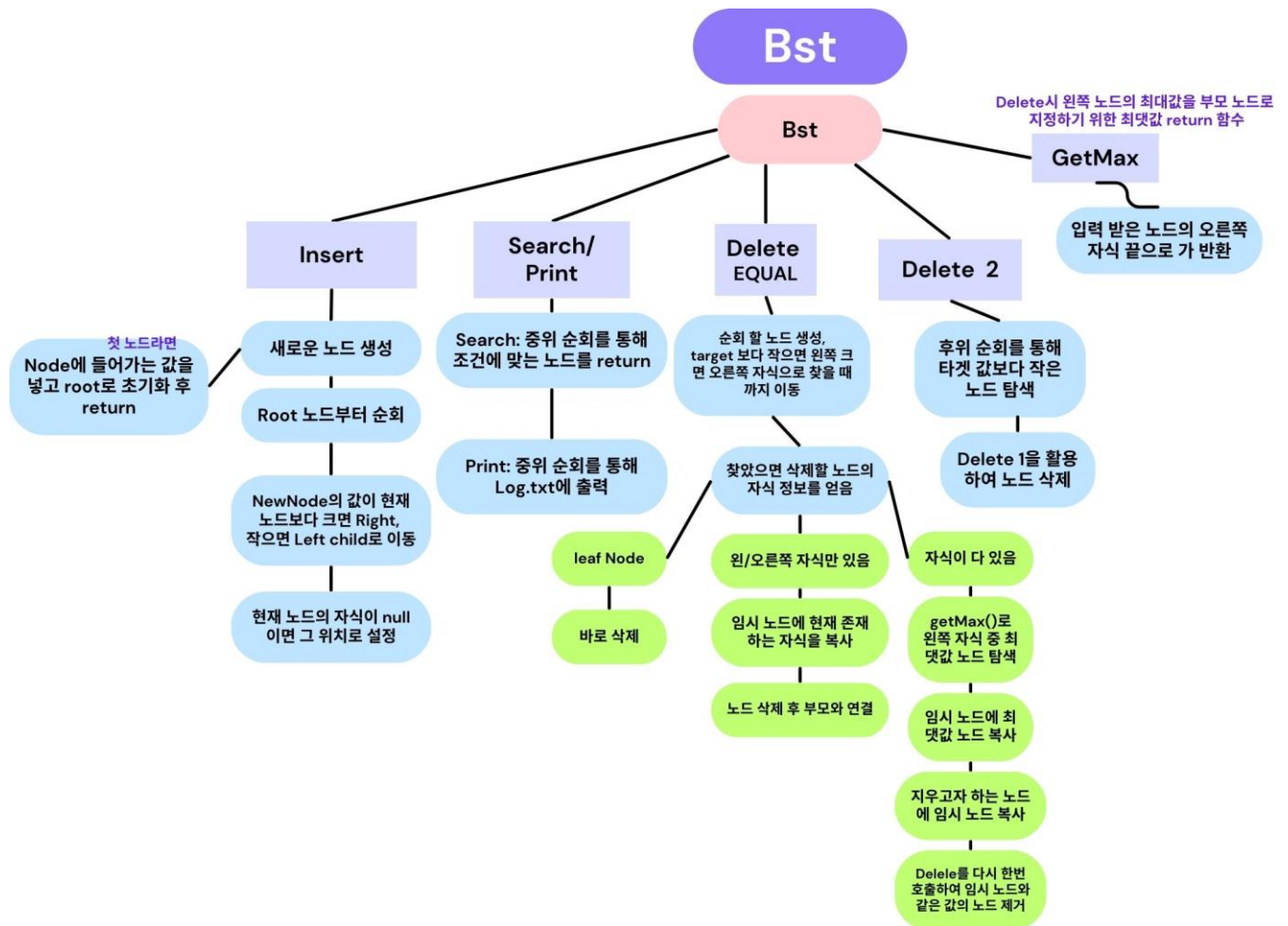
Manager:



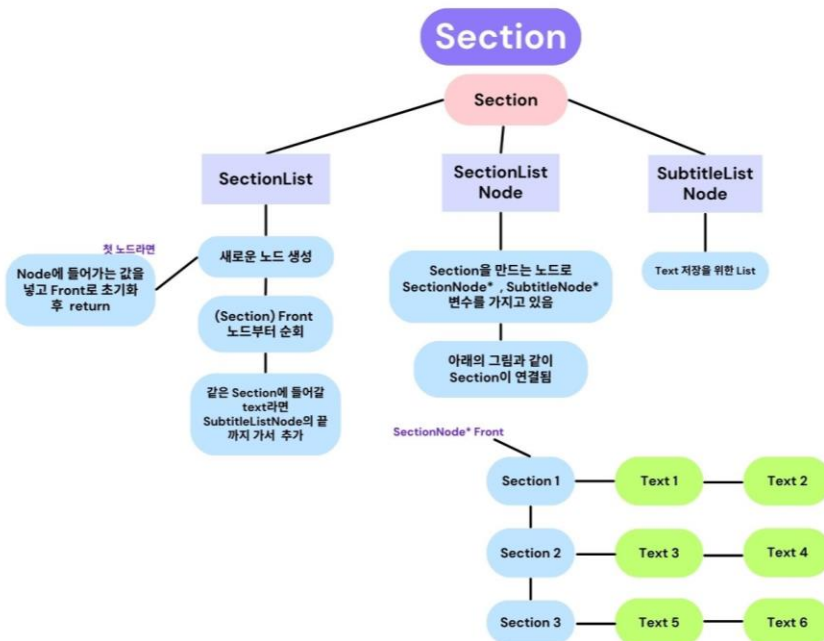
Queue:



Bst:



Section:



## Algorithm:

자막 관리 프로그램을 구현하기 위해서는 먼저 1. Queue, Bst, Subtitle/Section Node를 구현한다. 2. 이를 관리하는 Manager class를 만들어 명령어에 따른 동작을 구현한다. 라는 순서로 구현을 하면 복잡하지 않게 구현을 할 수 있다.

### 1. Queue

Queue의 구조는 FIFO으로, 먼저 들어온 노드가 제일 먼저 삭제(pop)되는 구조를 가지고 있다. 이를 관리하기 쉽게 배열이 아닌 연결리스트로 구현을 하여 쉽게 유지.보수가 되도록 구현하였다. 헤더 파일에 Queue의 맨 처음과 맨 끝을 가르키게 하는 Front, Back을 선언하여 Push와 Pop을 할 때에 어느 노드를 가르키고 있는지 알 수 있게 하였다.

Push 를 할 때에는 새로운 노드를 선언하고, Front가 null이면 새로 만든 노드를 front로 초기화하고, 텍스트 파일을 열어 subtitle을 삽입해준다 후에 return을 해주고, null이 아닐 시 Queue를 순환하는 노드 포인터를 만들어 queue의 제일 끝으로 가게 만들어준다. Queue의 끝에 도달할 시 텍스트 파일을 열어 subtitle을 삽입해준다. 그렇게 하면 연결리스트의 맨 마지막에 삽입하는 꼴이므로 FIFO의 구조를 지킬 수 있게 된다. 또한 Push함수는 Manager에서 관리 할 수 있게 SubtitleQueueNode\*을 인자로 받게 해주었다.

Pop함수는 삭제할 때에 그 노드의 정보를 복사할 노드 포인터 (\* deleteNode) 를 선언해 front로 초기화를 해준다. 그렇게 하면 delete가 front부터 순회를 하며 하나씩 삭제를 한다. 그리고 Queue 내부에 있는 text 정보를 return해주기 위해 text를 저장하는 노드를 만들어주고, delete를 하기 전 text 정보를 복사해 pop이 끝나면 return을 해주어 이를 이용하여 bst를 구축할 수 있게 하였다.

Queue의 Print 함수는 queue를 순회하며 하나씩 출력하면 되므로 queue를 순회하는 노드 포인터를 만들어 front로 초기화를 해준 후 next로 이동하고 log.txt에 출력하고 next가 null이 될 때까지 반복하면 queue의 모든 값을 출력할 수 있게 된다.

IsEmpty 함수는 front가 null이면 참을 반환하고, IsFull 함수는 이 문제에서 queue의 크기가 100으로 초기화 된 후 바뀌지 않는다고 했으므로 size변수와 순회할 노드를 만들어 next로 이동 할때 마다 size가 1씩 증가하게 하여 현재 노드가 back에 도달하기 전까지 순회했을때 size가 총 98 (size = 0 부터 backNode 의 전인 98까지 돌면 100개이다.)이면 참을 반환한다.

### 2. Bst

Bst는 이진 트리의 약자로, root노드부터 시작하여 삽입할 노드가 root노드의 data값 보다 작으면 왼쪽 자식으로, 더 크다면 오른쪽 자식으로 위치해 구조를 이루는 tree이다.

헤더파일에 left,right를 가르키는 포인터 노드, text를 저장할 string 변수, text 안의 시간을 기준으로 왼,오른쪽 자식을 가르므로 string을 int로 바꾸어 저장할 변수를 선언해 주었다.

Insert를 할 때는 data의 크기를 비교할 변수와 삽입할 text를 인자로 받는다. Root가 null일 때에는 인자로 입력 받은 값들을 노드에 삽입을 해주고 return 해주었다. Root가 null이 아닐 경우 인자로 받은 data가 현재 있는 노드보다 크면 오른쪽 자식, 작으면 왼쪽 자식으로 가게 한다. 자식의 끝에 연결을 해야하므로 오른쪽, 왼쪽 자식으로 이동하는 것을 재귀함수를 이용하고, 새로 만든 노드를 return 하여 부모 노드와 연결을 시켜준다.

Print 함수는 중위 순회를 이용하는데, 중위 순회란 왼쪽 자식의 끝으로 갔다가 자식의 끝을 만나면 return 하여 출력하고, 오른쪽 자식의 끝으로 가서 끝을 만나면 다시 retrn 하고 출력함을 반복하는 순회 방식을 말한다. 이렇게 출력을 하게되면 왼쪽 자식의 제일 끝으로 가서 출력하고 부모 노드로 return 했다가 오른쪽 끝으로 가게 되므로 최종적으로 오름차순으로 출력을 하게 된다.

Search함수는 함수 인자로 들어온 값을 가지고 있는 노드를 찾는 함수로, 모든 노드를 순회하여 조건에 맞는 값을 가진 노드를 반환하는 함수이다. 이 문제에서는 Section을 만들 때에 활용이 되므로 탐색 시작 값과 탐색 종료 값을 함수 인자로 받게 하였다. 또한 string 타입의 text를 반환해야 하는데, 탐색 시작 값  $\leq$  노드의 시간 값  $\leq$  탐색 종료 값 에 맞는 모든 노드의 text를 반환해야해서 string배열을 선언해 이 배열에 조건에 맞는 값을 전부 저장하고 활용할 수 있게 string 배열과 배열의 크기를 알려주는 변수를 함수의 인자로 받게 하였다. Search를 할 때에는 print 와 마찬가지로 중위 순회 방식을 선택하여 왼쪽 자식 끝으로 가서 탐색 시작, 탐색 종료 값 사이에 있는 노드의 text를 string 배열에 저장하고, 배열의 크기를 알려주는 변수를 +1 해주고, 오른쪽 자식의 끝으로 가서 다시 반복하게 한다.

Delete 함수는 Equal, Under 두 가지 경우로 나누어 구현을 해주었다. Equal은 target 값을 인자로 받아 이 값과 같은 값을 가지고 있는 노드를 삭제하는 것이고, Under은 target값 보다 작은 값을 가지고 있는 노드를 전부 삭제하는 것이다. 또한 Manager class 안에서 만든 bst를 삭제해야 하므로 SubtitleBstNode를 함수의 인자로 받게 하였다. 삭제 할 때에는 자식의 경우에 따라 3가지로 나뉘어 구현을 해주었는데, 삭제하고자하는 노드가 1) 자식이 없는 경우 2) 왼/오른쪽 자식만 가지고 있는 경우, 3) 자식을 모두 가지고 있는 경우로 나누었다. 먼저 삭제할 노드의 정보를 복사할 노드를 선언하고, root부터 시작하여 현재 노드의 값이 삭제할 값보다 작으면 왼쪽 자식으로, 그렇지 않으면 오른쪽 자식으로 이동하며 target값과 맞는 노드를 탐색에 성공했을때 위의 경우로 나누어 삭제를 진행한다. 1번 경우는 자식노드가 없으므로 부모 노드와의 연결을 끊고 삭제를 해준다. 2번 경우는 왼/오른쪽 자식만 가지고 있으므로 왼/오른쪽 자식의 정보를 복사하여 임시 노드에 넣어주고, 삭제할 노드를 삭제한 다음, 복사한 임시 노드를 다시 부모 노드와

연결을 시켜준다. 3번의 경우 자식을 다 가지고 있으므로 왼쪽 자식 중의 최댓값 or 오른쪽 자식 중의 최솟값을 찾아 부모 노드와 연결을 시켜주는데, 여기서는 왼쪽 자식 중의 최댓값을 구해 부모 노드와 연결을 시켜주었다. 여기서 중요한것은 자식 노드를 전부 가지고 있을 때에는 해당 노드를 물리적으로 삭제하면 그 자식들의 정보를 모두 복사해서 다시 붙여야 하는 작업이 소요되므로, 논리적으로 삭제하는 방법을 택했다. 먼저 왼쪽 자식 중의 최댓값을 찾아 이를 복사하고 삭제할노드에 넣어준다. 그렇게 되면 같은 값을 가진 노드가 2개가 되므로 그 값을 target으로 하여 다시 Delete함수를 호출해 target값을 가진 노드를 삭제한다. 그 후에 지우고 연결시킨 노드를 return하여 부모 노드와 연결을 시켜주면 삭제가 끝나게 된다. Under는 후위순회 방식을 활용했는데, 후위 순회를 하면 왼쪽과 오른쪽의 맨 끝 자식부터 순회를 하며 동작하는데, Delete Under 라는 특성을 생각했을때 제일 작은값 부터 탐색하며 삭제하는 것이 유리하다고 생각하여 이 방법을 채택하였다. 순회 후 target 값보다 작다면 삭제를 한다. 이때 하나씩 돌며 삭제를 하니 Delete Equal 함수를 호출하여 간단하게 구현하였다.

GetMax 함수는 노드를 삭제할 때에 왼쪽 자식 중의 최댓값을 찾는 함수이므로 삭제할 노드의 정보를 함수 인자로 받고, 그 노드의 오른쪽 끝이 최댓값이니, 오른쪽 자식 끝으로 가서 그 값을 반환하게 만들었다.

### 3. SectionList

SectionList는 탐색 시작 시간  $\leq$  노드가 가진 값  $\leq$  탐색 종료 시간에 해당하는 모든 노드를 한 섹션으로 묶는 것이다. Section을 만들 때에는 Section number, SectionList의 연결을 위한 포인터, Subtitle를 연결하기 위한 SubtitleNode 포인터를 멤버 변수로 선언해주었다. 내가 생각한 Section을 구현하는 쉬운 방법은 새로운 노드가 들어왔을 때 같은 Section Number를 가지고 있다면 subtitleNode에 연결하고, 번호가 같지 않다면 SectionNode 와 연결하여 새로운 Section을 만들고 그 안에서 subtitleNode를 연결하여 Section을 출력할 때에 간편하게 할 수 있게 구현하였다. (Flowchart의 Section부분 그림 참고.)

Insert 함수는 연결리스트를 맨 끝에 연결시키는 코드로 짰다. Head포인터가 null이면 새로운 노드에 정보를 다 넣어주고 return한다. Null이 아니라면 순회 노드를 만들어서 Section 노드를 순회하며 섹션 번호가 같으면 SubtitleNode 순회를 시작하고, 끝에 다다르면 새로운 SubtitleNode를 연결하고 그 안에 데이터를 삽입한다. 만약 같은 섹션 번호가 아니라면 SectionNode의 끝까지 이동한 다음 새로운 노드를 연결하고 그 노드에 데이터를 넣어준다.

Search함수는 섹션 번호가 일치하는 섹션을 찾는 함수로 SectionNode를 순회하는 포인터 노드를 선언하여 Section을 순회한다. 같은 섹션 값을 가진 노드가 나오면 그 노드를 return한다.

#### 4. Manager

Manager class는 자료구조들을 이용하여 전체적인 프로그램을 관리하는 class이다. 먼저 Run함수는 프로그램을 실행시키는 함수로, command.txt에서 한줄씩 읽어와 명령어를 실행한다. 한줄씩 읽을때는 getline을 이용하여 string에 넣어줬고, stringstream을 이용해 이 한줄을 띄어쓰기를 기준으로 나눠주어 string배열에 저장해 명령어를 분석하고 명령어가 부족하지 않은지, 더 많지 않은지를 검사한다.

##### LOAD:

LOAD 명령어는 subtitle.txt에서 한줄씩 읽어와 queue에 저장하고 잘 저장이 되었으면 log.txt에 출력하는 함수이다. SubtitleQueue 타입의 변수를 참조형으로 하여 queue를 만든 후 값이 저장될 수 있게 하였다. txt 파일에서 한 줄씩 읽어와 Queue에 push해준다. Queue의 front가 null이 아니라면 이미 queue에 데이터가 들어가있는 것 이므로 에러처리를 해준다. 그리고 잘 동작했다면 log.txt에 출력을 해주고 return한다.

##### QPOP:

QPOP 명령어는 queue에 있는 데이터를 pop하여 return된 데이터로 bst를 구축하는 명령어이다. 먼저 LOAD에서 저장한 queue변수를 불러와 front를 가져오고, front부터 다음으로 넘어가며 text를 불러오고, 하나씩 pop을 하여 queue를 비운다. bst에 구축하기 전에 queue에서 받은 text는 string 타입이니 bst에서 비교할 수 있게 integer 타입으로 변환해주는 함수가 필요하여 TimetoSec 이라는 함수를 선언하였다. TimetoSec 함수에서는 text를 인수로 받아 istringstream에 넣어 분리를 해준다. 이때 콜론(:) 을을 기준으로 데이터를 나누고 정수형 배열에 저장한다. 예를 들어 12:03:34 “~~d” 라는 텍스트를 받았다면 istringstream으로 12:03:34 만 저장을 하고, getline을 콜론(:)기준으로 나누어 배열에 저장하면 12 / 03 / 34 이런식으로 배열에 저장이 되는 것이다. 저장된 배열 값을 초로 바꿔주어 시간을 비교할 수 있게 하였다. 다시 QPOP으로 돌아와서 시간을 구하고 pop을 완료했다면, bst에 삽입을 하여 이진트리를 구축해준다. 이때 모든 pop한 노드를 통해 bst를 구축하므로 queue가 빌때까지 삽입을 해주면 된다.

##### PRINT

PRINT 함수는 인자를 받을 때와 받지 않을 때로 나눌 수 있는데, 매개변수로 함수를 구분할 수 있는 오버로딩을 이용하여 구현한다. 그냥 PRINT 명령어만 시행됐을 때는 bst의 root를 불러와 log.txt에 출력해주면 된다. PRINT 3 이라는 인자를 받으면 Section을 출력하는 명령어로 Section을 탐색하는 포인터 노드를 선언하고, Search로 Section을 탐색하여 같은 섹션 번호가 있는 노드를 탐색한다. 만약 같은 번호인 섹션을 찾았으면 그 노드의 SubtitleNode 를 순회하며 출력해준다.



## SECTION

SECTION 명령어는 섹션을 구축하는 명령어로, 탐색 시작시간, 종료시간, 구축하는 섹션 번호를 함수 인자로 넣어 섹션을 구축한다. 명령어로 들어오는 탐색 시작 시간, 종료 시간은 string 타입이니 TImetoSec 함수를 호출하여 이를 integer 타입으로 변환해준다. 그리고 Bst를 기반으로 섹션을 구축하므로, bst를 불러와 Search 함수를 호출하여 탐색 시작 시간과 종료 시간 사이에 있는 노드들을 전부 구하고, return 된 string 배열, 섹션 번호를 Section에 삽입해주면 Section 구축이 완료된다.

## DELETE

DELETE 명령어는 EQUAL, UNDER 두 개가 있는데, EQUAL은 특정 시간과 같은 노드를 삭제하는 것이고, UNDER는 특정 시간 미만의 노드들을 전부 삭제하는 것이다. SubtitleBST에서 DELETE를 두 개의 함수로 나누어 구현했으므로. 여기서도 두 개로 나누어 만들어준다.

첫번째로 DELETE EQUAL은 원하는 특정 시간을 입력 받고, TImetoSec 함수를 호출하여 integer로 바꾼다음, bst에서 Delete 함수를 호출하여 삭제를 해준다. 두 번째 DELETE UNDER는 특정 시간을 입력받고, Delete 2를 호출하여 삭제를 해주면 끝이다.

결과화면:

|===== LOAD =====

01:03:45 You're not listening to me!  
00:52:36 You're welcome.  
00:54:16 Hey, you! Where's the other mother?  
00:54:18 I wanna go home.  
00:53:50 Mom! Dad!  
00:53:56 Oh, God. I'm still here?  
00:54:19 All will be swell, soon as Mother's refreshed.  
00:54:23 Her strength is our strength.  
00:52:21 Bed? Before dinner?  
00:52:20 Right now!  
00:52:19 I'm going to bed.  
00:21:37 See you soon.  
00:52:23 I'm really, really tired. Yeah.  
01:09:34 Ok.  
01:09:24 Challenge her, then.  
01:09:29 She's got a thing for games.  
01:09:26 She may not play fair, but she won't refuse.  
01:09:17 I have to go back. They are my parents.  
01:11:36 A finding things game.  
01:11:27 Everybody likes games.  
01:11:51 What if you don't find them?  
01:11:59 And I'll let you sew buttons into my eyes.  
01:11:54 If I lose, I'll stay here with you forever  
01:11:31 What kind of game would it be?  
01:11:38 And what is it you'd be finding?  
01:11:41 My real parents. And... the eyes of the children.  
01:11:26 I know you like them.  
01:11:22 Why don't we play a game?

=====

===== QPOP =====

Success

=====

===== PRINT =====

Subtitle\_BST

00:21:37 See you soon.

00:52:19 I'm going to bed.

00:52:20 Right now!

00:52:21 Bed? Before dinner?

00:52:23 I'm really, really tired. Yeah.

00:52:36 You're welcome.

00:53:50 Mom! Dad!

00:53:56 Oh, God. I'm still here?

00:54:16 Hey, you! Where's the other mother?

00:54:18 I wanna go home.

00:54:19 All will be swell, soon as Mother's refreshed.

00:54:23 Her strength is our strength.

01:03:45 You're not listening to me!

01:09:17 I have to go back. They are my parents.

01:09:24 Challenge her, then.

01:09:26 She may not play fair, but she won't refuse.

01:09:29 She's got a thing for games.

01:09:34 Ok.

01:11:22 Why don't we play a game?

01:11:26 I know you like them.

01:11:27 Everybody likes games.

01:11:31 What kind of game would it be?

01:11:36 A finding things game.

01:11:38 And what is it you'd be finding?

01:11:41 My real parents. And... the eyes of the children.

01:11:51 What if you don't find them?

01:11:54 If I lose, I'll stay here with you forever

01:11:59 And I'll let you sew buttons into my eyes.

=====

===== SECTION =====

Success

=====

===== SECTION =====

Success

=====

===== SECTION =====

Success

=====

===== PRINT =====

Section 3

00:52:19 I'm going to bed.

00:52:20 Right now!

00:52:21 Bed? Before dinner?

00:52:23 I'm really, really tired. Yeah.

=====

===== PRINT =====

Section 2

00:53:50 Mom! Dad!

00:53:56 Oh, God. I'm still here?

00:54:16 Hey, you! Where's the other mother?

00:54:18 I wanna go home.

00:54:19 All will be swell, soon as Mother's refreshed.

00:54:23 Her strength is our strength.

=====

===== DELETE =====

Success

=====

===== PRINT =====

Subtitle\_BST

00:21:37 See you soon.

00:52:19 I'm going to bed.

00:52:20 Right now!

00:52:21 Bed? Before dinner?

00:52:23 I'm really, really tired. Yeah.

00:53:50 Mom! Dad!

00:53:56 Oh, God. I'm still here?

00:54:16 Hey, you! Where's the other mother?

00:54:18 I wanna go home.

00:54:19 All will be swell, soon as Mother's refreshed.

00:54:23 Her strength is our strength.

01:03:45 You're not listening to me!

01:09:17 I have to go back. They are my parents.

01:09:24 Challenge her, then.

01:09:26 She may not play fair, but she won't refuse.

01:09:29 She's got a thing for games.

01:09:34 Ok.

01:11:22 Why don't we play a game?

01:11:26 I know you like them.

01:11:27 Everybody likes games.

01:11:31 What kind of game would it be?

01:11:36 A finding things game.

01:11:38 And what is it you'd be finding?

01:11:41 My real parents. And... the eyes of the children.

01:11:51 What if you don't find them?

01:11:54 If I lose, I'll stay here with you forever

01:11:59 And I'll let you sew buttons into my eyes.

=====

===== DELETE =====

Success

=====

===== PRINT =====

Subtitle\_BST

00:52:23 I'm really, really tired. Yeah.

00:53:50 Mom! Dad!

00:53:56 Oh, God. I'm still here?

00:54:16 Hey, you! Where's the other mother?

00:54:18 I wanna go home.

00:54:19 All will be swell, soon as Mother's refreshed.

00:54:23 Her strength is our strength.

01:03:45 You're not listening to me!

01:09:17 I have to go back. They are my parents.

01:09:24 Challenge her, then.

01:09:26 She may not play fair, but she won't refuse.

01:09:29 She's got a thing for games.

01:09:34 Ok.

01:11:22 Why don't we play a game?

01:11:26 I know you like them.

01:11:27 Everybody likes games.

01:11:31 What kind of game would it be?

01:11:36 A finding things game.

01:11:38 And what is it you'd be finding?

01:11:41 My real parents. And... the eyes of the children.

01:11:51 What if you don't find them?

01:11:54 If I lose, I'll stay here with you forever

01:11:59 And I'll let you sew buttons into my eyes.

=====

=====

===== PRINT =====

Section 3

00:52:19 I'm going to bed.

00:52:20 Right now!

00:52:21 Bed? Before dinner?

00:52:23 I'm really, really tired. Yeah.

=====

결과화면 설명:

#### 1. LOAD

Subtitle.txt에서 자막 정보를 불러와 queue에 저장을 하고, 잘 저장이 되었으면 log.txt에

출력을 하는 명령어인데, subtitle.txt 와 log.txt를 비교하였을때 빠진 자막이나 중복된 자막없이 잘 출력이 된 것으로 보아 queue가 성공적으로 만들어졌음을 알 수 있다.

## 2. QPOP

LOAD에서 만든 queue를 pop 하여 bst를 구축하는 명령어로, 코드가 잘 실행되었을 때 success 를 출력하는데, log를 보아 잘 구축되었음을 알 수 있다.

## 3. PRINT

인자가 들어오지 않았으니 BST에서 순회를 하며 Print하는 명령어이다. Bst에서 중위순회를 하며 출력한 값은 오름차순으로 정렬되는데, PRINT를 보면 오름차순으로 출력됐음을 알 수 있다.

### 4.5. SECTION 3 00:52:10 00:52:30 / SECTION 2 00:53:50 00:54:23

SECTION을 구축하고, 섹션 번호는 3,2 로, 00:52:10 ~ 00:52:30 / 00:53:50 ~ 00:54:23 에 해당되는 노드들을 섹션으로 구축하는 명령어이다. Success가 출력된 것으로 보아 잘 구축되었다.

## 6. PRINT 3 / PRINT 2

PRINT에 인자가 들어왔으므로 Section을 출력하는 명령어이다. PRINT 3은 00:52:10 ~ 00:52:30 안에 있는 값들만 출력이 되어야 하고, PRINT 2는 00:53:50 ~ 00:54:23 안에 있는 값들만 출력이 되어야하는데, log를 보니 잘 출력이 됐음을 알 수 있다. 이는 즉 Section이 올바르게 구축되었음을 알 수 있다.

## 7. DELETE EQUAL 00:52:36 , PRINT

Delete Equal 명령어는 특정 시간을 가진 노드만 삭제하는 명령어로 success가 출력됨을 보아 잘 제거 된 것을 알 수 있고, PRINT명령어를 했을때 00:52:36 시간을 가진text만 사라져야하는데, log를 보면 잘 되었음을 볼 수 있다.

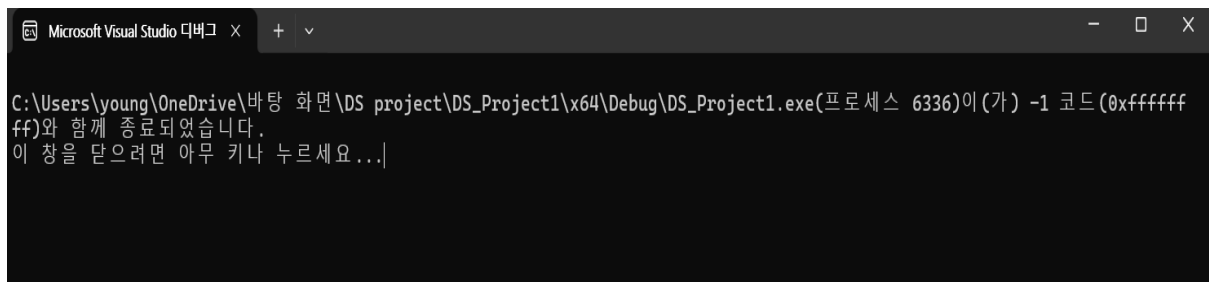
## 8. DELETE UNDER 00:52:23 / PRINT



Delete Under 명령어이므로 00:52:23 미만의 시간을 가진 text는 전부 삭제가 되어야하는데, success가 출력됐으므로 잘 작동했음을 알 수 있다. PRINT 명령어를 시행했을 때 DELETE 00:52:23 미만의 시간들이 제거되어 출력 되지 않으면 잘 작동한 것인데, log를 보면 성공적으로 작동했음을 알 수 있다.

#### 9. PRINT / PRINT 3 / EXIT

PRINT는 DELETE를 시행하고 나서 PRINT를 했으므로 00:52:23 미만의 시간은 출력되면 안되고, 00:52:36 의 시간도 출력되면 안되는데, log를 보면 잘 수행되었음을 알 수 있다. 마지막 EXIT 명령어는 프로그램을 종료하는 명령어로, cmd를 봤을때 잘 수행되었음을 알 수 있다.



```
Microsoft Visual Studio 디버그 x + v
C:\Users\young\OneDrive\바탕 화면\DS project\DS_Project1\x64\Debug\DS_Project1.exe(프로세스 6336)이(가) -1 코드(0xffffffff)와 함께 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요...
```

## 고찰

이번 프로젝트를 진행하면서 Queue, Bst, 연결 리스트 등 1학기때 C++ 에서 배웠던 내용들을 복습하며 다시 만들어보고, 이를 구현하는 것뿐만 아니라 자료구조를 활용하여 관리 프로그램을 만드는 것을 해보아 참 좋았던 것 같다. 저번 학기 때 구현을 했을 때에는 처음 해보는 것이다 보니 많이 버벅이면서 오래걸렸는데, 한번 해본 것을 하니 구현하고 문제를 파악하고 푸는 속도가 전보다 빨라진 것 같아 뿌듯하기도 했다. 그래도 class를 활용하는 부분에서는 아직 모르는 것들이 많아 조금 어려웠는데, 책을 참고하고 전에 내가 풀었던 문제들을 참고하고 풀어보니 나아지는 것 같아 다행이라고 생각했다. 클래스 중에서 특히 어려웠던 것은 Manager와 section 클래스를 다루는 것에서 조금 애를 먹었는데, Manager에서 자료구조를 만들고 이를 저장하는 변수와 return 한 값을 활용하는 방안 등 깔끔한 코드를 만들기 위해 어떤 식으로 하면 좋을지 고민을 많이 했던 것 같다. 그리고 section 클래스에서는 SubtitleList와 SectionNode, Section 이렇게 세개의 파일을 다루어야 해서 이 파일들의 각각의 역할을 이해하고, 구현하는데에 시간을 좀 소요하였다. 특히 SubtitleNode와 SectionNode 를 어떤식으로 연결하여 구현해야 내가 원

하는 그림으로 Section을 구현할 수 있을지 고민을 많이 하였다. 코드를 구현 하다가 중위, 후위 순회를 구현할 때에 반복문을 쓰지, 재귀적으로 함수를 쓰지 고민을 했는데, 재귀 함수로 구현을 하고 나서 검색을 해보니 반복문은 함수 호출 없이 실행되고, 추가적인 메모리도 쓰지 않으니 반복문을 쓰는 것이 메모리를 더 적게 쓴다. 하지만 재귀 함수가 더 직관적으로 표현을 할 수 있다고 배웠다. 함수들을 호출 할 때에 직관적으로 성공했는지 아닌지를 보기 위해 bool 변수들을 선언하여 쓴 함수들도 있고, 정확한 에러처리를 하기 위해 코드의 순서도 잘 지켜 작성하기 위해 노력했다. 이번 프로젝트를 설계하며 Ubuntu를 처음 써보게 되었는데, 우분투를 쓰기위해 리눅스라는 환경을 공부하고, 리눅스에서 git을 쓰기 위해 git 관련된 것들도 열심히 찾아보아 이번 프로젝트에서는 코딩만 한 것이 아닌 여러 기술들을 배운 것 같다. 그리고 아예 처음 들어보는 Makefile의 사용법과 용도, 왜 써야 하는지 등을 공부해 지식을 넓힌 것 같다. 그리고 다른 어려움을 겪은 것은 항상 겪는 문제인 디버깅이었다. 코드가 수행이 안될 때에는 항상 디버깅을 하는데, 이번에는 함수를 호출하는 부분도 많고 메모리를 쓰는 부분도 많아 이를 파악하고 쫓아가며 문제를 파악하기가 수월하지 않았다. class에서 소멸자를 구현할 때에는 동적으로 할당 해 놓은 메모리들을 전부 해제해야 하는데, 어떻게 해야 메모리에 누수가 생기지 않고 잘 삭제할 수 있을지 디버깅을 하며 보는 것도 생각보다 많은 소요가 있었다.

이번에 내가 설계한 bst와 queue, 연결 리스트의 탐색 시간 복잡도를 생각해 보자면, 연결리스트는 하나씩 연결할때마다 시간복잡도가 늘어나는데, 연결 리스트의 경우 원하는 값을 찾을 때 까지 이동을 해서 탐색을 하는데, worst의 경우 n개의 리스트를 모두 돌아야 하므로  $O(n)$ 의 시간 복잡도이고, queue의 경우도 n개의 리스트가 연결이 되어 있으니  $O(n)$ 의 시간복잡도를 가진다. Bst의 경우는 조금 다른데, 트리 구조를 가지고 있기 때문에  $O(h)$ 의 시간 복잡도를 가지고 (h : 트리의 높이), 최악의 경우  $O(n)$ 의 시간복잡도를 가진다. 따라서 평균적으로는  $\log(n)$ 의 시간복잡도를 가지게 된다.

메모리를 적게 쓰는 것이 무조건적으로 좋다는 걸 알았지만, 대부분의 경우 다 구현을 하고 나서 어떻게 메모리를 적게 쓸 수 있을까? 라는 고민을 하기에는 늦어 그냥 제출하는 경우가 허다 했는데, 이제 자료 구조를 어떤식으로 구현해야 할 지 경험을 했으니 앞으로는 프로젝트를 해결 하는 방법을 조금 바꿔야겠다고 생각을 했다. 먼저 머릿속으로 어떻게 구현을 할 지 플로우 차트를 먼저 이용하여 대강으로 구조를 잡고, 그 속에서 메모리를 어떻게 적게 쓸 수 있을지 고민하는 것이다. 예를 들어 이번에 반복문이 메모리를 더 적게 쓴다는 것을 알았으니 재귀나 반복문 둘 다로 구현 할 수 있을 때에는 최대한 반복문을 사용하는 쪽으로 가는 방법을 선택하는 것이다. 이렇게 이번 학기 동안 프로젝트를 진행하면 더 나은 실력을 가진 사람으로써 목표에 한발짝 다가갈 것 같아 마음이 좋다.