Data Structure Project #1

2024년 09월 16일

Due date: 2024년 10월 13일 일요일 23:59:59 까지

본 프로젝트에서는 큐(Queue) , 이진 탐색 트리(Binary Search Tree), 연결 리스트(Linked List) 자료구조를 이용하여 간단한 자막 관리 프로그램을 구현한다. 해당 프로그램은 LOAD 명령어를 실행하여 자막 데이터 파일로부터 자막시간, 자막내용 정보를 읽어 Queue를 구축하며, 해당 Queue를 Subtitle_Queue라 부른다. Queue가 구축된 후 QPOP 명령어를 실행하면 데이터를 방출하여 BST 자료구조(Subtitle_BST)에 저장한다.

Subtitle_BST는 자막시간, 자막내용 정보를 갖는 노드로 구성되며, 자막시간 정보를 기준으로 정렬된다. BST가 구축된 후 SECTION 명령어를 실행하면 특정 시간 구간에 포함되는 노드들을 탐색하여 List 자료구조(Section_List)에 저장한다.

Section_List는 헤더 노드와 내용 노드로 구성된다. 헤더 노드는 섹션 번호, 내용 노드 포인터 정보를 가지며, 내용 노드는 자막시간, 자막내용 정보를 갖는다. SECTION 명령어 실행 시 헤더 노드가 생성되며, 해당 SECTION 명령어에서 탐색된 BST 노드들의 정보를 통해 내용 노드들이 생성되어 연결된다.

각 자료구조의 구축 방법과 조건에 대한 자세한 설명은 **Program Implementation**에서 설명한다.

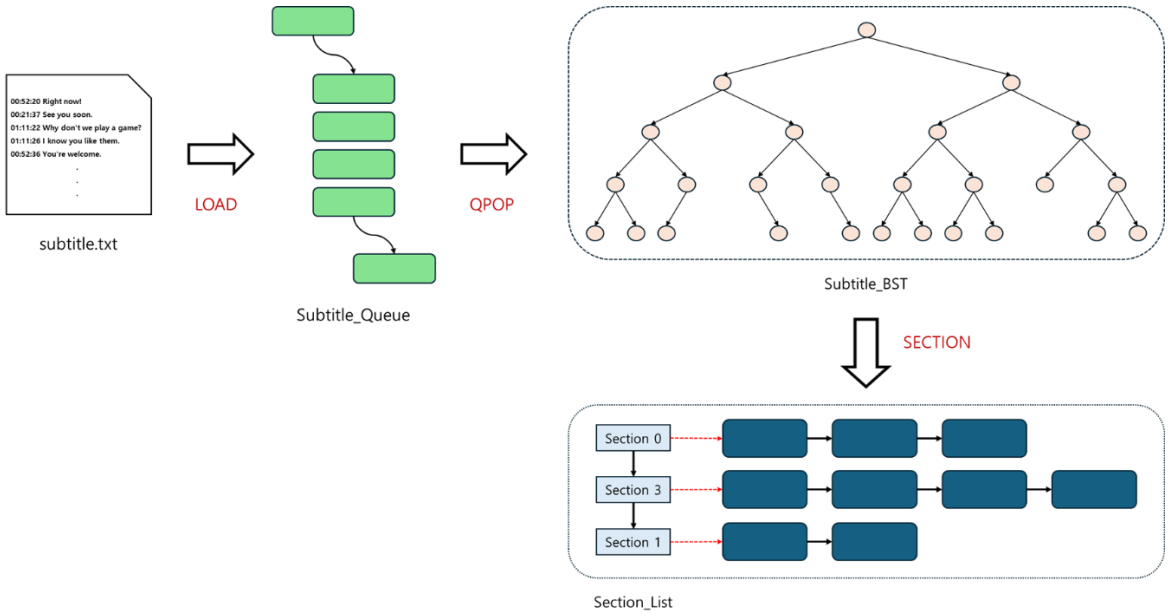

그림 1. 프로그램 구조

☐ Program Implementation

1) Subtitle_Queue

- 주어진 subtitle.txt에 저장된 데이터를 이용하여 구축된다. subtitle.txt의 줄 단위로 데이터를 읽고 노드를 구성하여 PUSH 되며, Queue의 자료구조에 따라 입력된 순서대로 POP 된다.
- 입력 데이터는 자막시간 자막내용 (ex. 00:52:20 Right now!)의 형식을 갖는다. 이때, 자막시간은 중복되지 않으며, 자막내용은 공백문자를 포함하고 200자 이하의 길이를 갖는다고 가정한다.
- Queue의 노드는 SubtitleQueueNode class로 구현되며, 자막시간, 자막내용 정보를 갖는다
- Queue의 크기는 100으로 초기화되고 이후 크기가 변하지 않으며, Queue가 비어 있을 때 POP 또는 전부 차 있을 때 PUSH가 수행되는 경우 프로그램이 종료된다.
- Queue에서 QPOP 명령어를 통해 방출되는 데이터는 BST의 입력으로 사용된다.

그림 2. Subtitle_Queue 예시

2) Subtitle_BST

- Queue에서 방출된 데이터를 이용하여 구축된다.
- BST의 노드는 SubtitleBSTNode class로 구현되며, 자막시간, 자막내용 정보를 갖는다.
- BST에서 지원하는 연산은 삽입, 삭제, 탐색, 출력(중위 순회)이며, 각 연산에 대해서는 **Functional Requirements**에서 자세히 설명한다.
- BST 연결 규칙은 다음과 같다.
  ① 부모 노드보다 자막시간이 시간적으로 이전인 노드는 왼쪽, 같거나 이후인 노드는 오른쪽 서브 트리에 위치한다.
  ② 노드를 제거할 때, 양쪽 자식 노드가 모두 존재할 경우에는 오른쪽 자식 노드 중 가장 작은 노드를 제거되는 노드 위치로 이동한다.

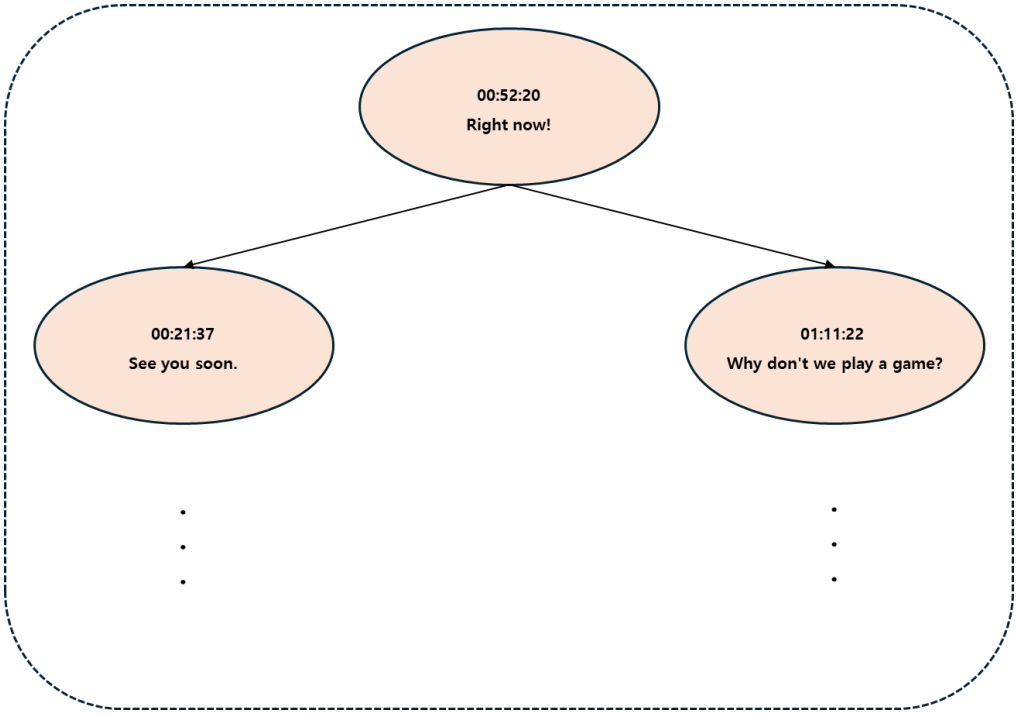

그림 3. Subtitle_BST 예시

3) Section_List

- Subtitle_BST이 구축된 후, SECTION 명령어 실행 시 구축된다.
- List의 노드는 헤더 노드와 내용 노드로 구성된다. 헤더 노드는 SectionListNode class로 구현되며, 섹션 번호, 내용 노드 포인터 정보를 갖는다. 내용 노드는 SubtitleListNode class로 구현되며, 자막시간, 자막내용 정보를 갖는다.
- SECTION 명령어 실행 시, SECTION 명령어의 첫번째 인자로 입력된 숫자를 섹션 번호로 갖는 헤더 노드가 생성되어 List에 추가된다. 이때, 섹션 번호는 중복되지 않는다고 가정한다. 이후 헤더 노드의 내용 노드 포인터를 시작으로 SECTION 명령어에서 탐색된 각 BST 노드의 정보를 갖는 내용 노드가 생성되어 연결된다.
- DELETE 명령어를 통한 삭제 연산은 BST에만 적용되며, List에는 적용되지 않는다고 가정한다. 즉, BST에서 삭제된 노드의 정보가 List에 남아있을 수 있다고 가정한다.

그림 4. Section_List 예시

☐ Functional Requirements

* 출력 포맷은 포맷에 대한 예시일 뿐 실제 출력되는 데이터들과는 차이가 있을 수 있습니다.

명령어	명령어 사용 예시 및 기능 설명
LOAD	사용 예시) LOAD 텍스트 파일의 데이터 정보를 불러오는 명령어로, 텍스트 파일에 데이터 정보가 존재할 경우 텍스트 파일을 읽어 Subtitle_Queue 자료구조에 모두 저장한다. 성공적으로 데이터를 불러온 경우 읽은 데이터를 출력하며, 출력 형태는 "자막시간 - 자막내용"이다. 만약 텍스트 파일이 존재하지 않거나 자료구조에 이미 데이터가 들어가 있으면 에러 코드를 출력한다. 사용되는 텍스트 파일의 이름은 아래와 같으며 파일 이름을 수정하지 않는다. 텍스트 파일: subtitle.txt 데이터 조건 - 자막시간은 중복되지 않으며, XX:XX:XX의 형식으로 입력된다. - 자막내용은 공백문자를 포함하고 200자 이하의 길이를 갖는다. - 텍스트 파일에서 자막시간과 자막내용은 공백문자로 구분된다 - 각 자막 정보는 라인 별로 작성되며, 개행문자로 구분된다. 출력 포맷 예시) ===== 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. . . . =============== ===== ERROR ===== 100 ===============
QPOP	사용 예시) QPOP Subtitle_Queue에서 데이터를 POP하여 Subtitle_BST를 구축하는 명령어이다. Subtitle_Queue의 front부터 모든 노드를 POP하여 해당 노드의 데이터로 자료구조를 구축한다. Subtitle_Queue에 데이터가 존재하지 않을 시 에러 코드를 출력한다.

	출력 포맷 예시) ===== QPOP ===== Success ========== ===== ERROR ===== 200 ==========
PRINT	사용 예시1) PRINT 사용 예시2) PRINT 3 Subtitle_BST 또는 Section_List에 저장된 데이터들을 출력하는 명령어이다. 인자가 입력되지 않는 경우, 중위 순회(in-order) 방식으로 Subtitle_BST를 탐색하여 데이터를 출력하며, 출력 형태는 "자막시간 - 자막내용"이다. Subtitle_BST에 데이터가 존재하지 않을 시 에러 코드를 출력한다. 인자로 특정 숫자가 입력되는 경우, Section_List에서 해당 섹션 번호를 갖는 헤더 노드와 연결된 내용 노드의 데이터를 순차적으로 출력하며, 출력 형태는 "자막시간 - 자막내용"이다. 존재하지 않는 섹션 번호가 인자로 입력된 경우 에러 코드를 출력한다. 출력 포맷 예시) 1) PRINT ===== PRINT ===== Subtitle_BST . . . 01:09:29 - She's got a thing for games. 01:09:34 - Ok. 01:11:22 - Why don't we play a game? . . . ========== 2) PRINT 3 ===== 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. ========== ===== ERROR ===== 300 ==========
SECTION	사용 예시) SECTION 3 00:52:10 00:52:30 Subtitle_BST에서 특정 시간 구간에 포함되는 노드들을 탐색하여 Section_List에 데이터를 추가하는 명령어이다. 인자로 "섹션번호 탐색시작시간 탐색종료시간"을 입력 받는다. 첫번째 인자로 입력 받은 섹션번호 정보를 갖는 헤더 노드를 생성하여 Section_List에 추가한다. Subtitle_BST에서 두번째와 세번째 인자로 입력 받은 탐색시작시간과 탐색종료시간 사이에 포함되는 BST 노드들을 탐색하고 (탐색시작시간 ≤ 자막시간 ≤ 탐색종료시간), 탐색된 BST 노드들의 데이터로 내용 노드들을 생성하여 추가된 헤더 노드에서부터 순차적으로 연결한다. Subtitle_BST에서 탐색된 노드가 없을 경우 에러 코드를 출력한다. 출력 포맷 예시) ===== SECTION ===== Success ========== ===== ERROR ===== 400 ==========
DELETE	사용 예시1) DELETE EQUAL 01:09:29 사용 예시2) DELETE UNDER 00:52:36 Subtitle_BST에 저장된 데이터를 제거하는 명령어이다. 인자로 EQUAL "특정시간"이 입력되는 경우 Subtitle_BST에서 입력된 특정시간과 동일한 자막시간을 가진 노드를 제거한다. 인자로 UNDER "특정시간"이 입력되는 경우 Subtitle_BST에서 입력된 특정시간보다 이전인 자막시간을 가진 모든 노드들을 제거한다. (자막시간<특정시간) 삭제하고자 하는 자막정보가 존재하지 않거나, Subtitle_BST에 데이터가 존재하지 않을 시 에러 코드를 출력한다. 출력 포맷 예시)

	===== DELETE ===== Success ========== ===== ERROR ===== 500 ==========
EXIT	사용 예시) EXIT 프로그램 상의 메모리를 해제하며, 프로그램을 종료한다. 출력 포맷 예시) ===== EXIT ===== Success ==========

□ 명령어 별 에러 코드

명령어	에러 코드
LOAD	100
QPOP	200
PRINT	300
SECTION	400
DELETE	500
잘못된 명령어	1000

□ Requirements in Implementation

- ✓ 모든 명령어는 command.txt에 작성되며 순차적으로 읽고 처리한다.
- ✓ 모든 명령어는 반드시 대문자로 입력한다.
- ✓ 명령어에 인자가 모자라거나 필요 이상으로 입력 받을 경우 에러 코드를 출력한다.
- ✓ 예외처리에 대해 반드시 에러 코드를 출력한다.
- ✓ 출력은 "출력 포맷"을 반드시 따라한다.
- ✓ log.txt 파일에 출력 결과를 반드시 저장한다.

□ 구현 시 반드시 정의해야 하는 Class

- ✓ SubtitleQueue : 자막 정보 Queue 클래스
- ✓ SubtitleQueueNode : 자막 정보 Queue의 노드 클래스
- ✓ SubtitleBST : 자막시간을 기준으로 정렬되는 BST 클래스
- ✓ SubtitleBSTNode : 자막시간을 기준으로 정렬되는 BST의 노드 클래스
- ✓ SectionList : 섹션 List 클래스
- ✓ SectionListNode : 섹션 List의 헤더 노드 클래스
- ✓ SubtitleListNode : 섹션 List의 내용 노드 클래스
- ✓ Manager : Manager 클래스
  - 다른 클래스들의 동작을 관리하여 프로그램을 전체적으로 조정하는 역할을 수행

□ Files

- ✓ subtitle.txt : 프로그램에 추가할 자막 데이터가 저장되어 있는 파일
- ✓ command.txt : 프로그램을 동작시키는 명령어들이 저장되어 있는 파일
- ✓ log.txt : 프로그램 출력 결과를 모두 저장하고 있는 파일

□ 채점 기준

✓ 코드

채점 기준	점수
LOAD 명령어가 정상 동작 하는가?	1
QPOP 명령어가 정상 동작 하는가?	2
PRINT 명령어가 정상 동작 하는가?	2
SECTION 명령어가 정상 동작 하는가?	2
DELETE 명령어가 정상 동작 하는가?	3
총합	**10**

– 채점 기준 이외에도 조건 미달 시 감점 (linux 컴파일 에러, 파일 입출력 X, 주석 미흡 등)

✓ 보고서

채점 기준	점수
Introduction을 잘 작성하였는가?	1
Flowchart을 잘 작성하였는가?	2
Algorithm을 잘 작성하였는가?	3
Result Screen을 잘 작성하였는가?	2
Consideration을 잘 작성하였는가?	2
총합	**10**

✓ 최종 점수는 (코드 점수 × 보고서 점수) 로 계산됩니다.

□ 제한사항 및 구현 시 유의사항

✓ 제공되는 코드(github 주소 참고)를 이용하여 구현하며 작성된 소스 파일과 클래스, 함수의 이름을 임의로 변경하지 않는다.
✓ 클래스의 함수 및 변수는 자유롭게 추가 구현이 가능하다.
✓ 제시된 Class를 각 기능에 알맞게 모두 사용한다.
✓ 프로그램 구조에 대한 디자인이 최대한 간결하도록 고려하여 설계한다.
✓ 채점 시 코드를 수정해야 하는 일이 없도록 한다.
✓ 주석은 반드시 영어로 작성한다. (한글로 작성하거나 없으면 감점)
✓ 프로그램은 반드시 리눅스(Ubuntu 18.04)에서 동작해야한다. (컴파일 에러 발생 시 감점)
– 제공되는 Makefile을 사용하여 테스트하도록 한다.
✓ 인터넷에서 공유된 코드나 다른 학생이 작성한 코드를 절대 카피하지 않도록 하며 적발 시 전체 프로젝트 0점 처리됨을 명시

□ 제출기한 및 제출방법

✓ 제출기한

- 2024년 10월 13일 일요일 23:59:59 까지 제출
  (추가 제출: 2023년 10월 14일 월요일 00:59:59 까지, 10% 감점)

✓ 제출방법

- 소스코드와 보고서 파일(**학번_DS_project1.pdf**)을 함께 압축하여 제출
- 확장자가 .cpp, .h, .pdf가 아닌 파일은 제출하지 않음 (Makefile과 텍스트 파일 제외)
- 보고서 파일 확장자가 pdf가 아닐 시 감점
- KLAS -> 과제 제출 -> tar.gz로 과제 제출

✓ 제출형식

- **학번_DS_project1.tar.gz** (ex. 2024123456_DS_project1.tar.gz)
- 제출 형식 미 준수 시 감점 (보고서 및 압축파일 학번, 프로젝트 이름 반드시 준수)

✓ 보고서 작성 형식 및 제출방법

- Introduction : 프로젝트 내용에 대한 설명
- Flowchart : 설계한 프로젝트의 플로우 차트를 그리고 설명
- Algorithm : 프로젝트에서 사용한 알고리즘의 동작을 설명
- Result Screen : 모든 명령어에 대해 결과화면을 캡처하고 동작을 설명
- Consideration : 고찰 작성

보고서는 위의 각 항목을 모두 포함하여 작성하며 보고서에는 소스코드를 포함하지 않음. Introduction과 Consideration 외의 모든 각 항목은 최소 2페이지 이상 작성하여 제출함. Consideration는 본 프로젝트 설계에 있어 힘들었던 점이나 성능 및 코드 구조를 개선하기 위해 작업한 내용, 프로젝트 및 코드관리를 위해 작업한 내용, 프로젝트를 설계하며 새로이 알게 되거나 참고한 내용이 될 수 있음.