Data Structure Lab. Project #2

2021 년 10 월 4 일

Due date: 2021 년 11 월 05 일 금요일 23:59:59 까지

본 프로젝트에서는 B+ tree, AVL tree, STL vector 를 이용하여 코로나 19 예방접종 관리 프로그램을 구현한다. 예방접종 관리 프로그램은 이름, 백신 명, 접종 횟수, 나이, 지역명을 관리하며, 이를 이용하여 접종 대상자 및 접종 완료자에 대한 정보를 제공할 수 있다. B+ tree 를 이용하여 접종 관리하며, AVL tree 를 이용하여 접종 완료자를 관리한다. 그리고 Print_vector 를 이용하여 접종 완료자를 다양한 정렬 방법으로 출력할 수 있다. [그림 1]은 코로나 19 예방접종 관리 프로그램의 구조이다. 자료구조의 구축 방법과 조건에 대한 자세한 설명은 **program implementation** 에서 설명한다.

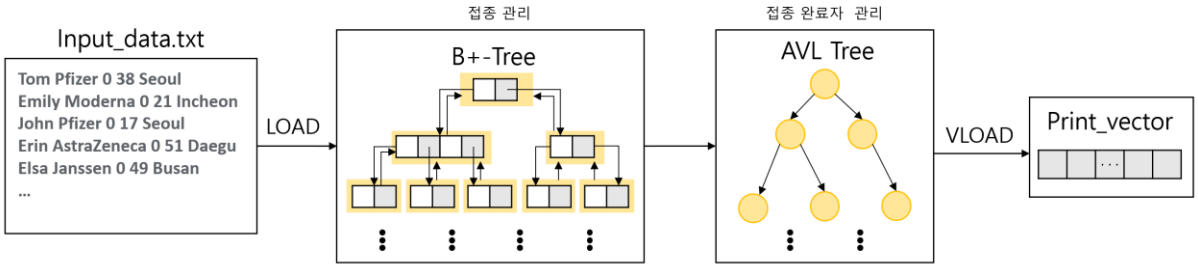

그림 1. 코로나 19 예방접종 관리 프로그램의 구조

□ Program implementation

1. 백신 접종 정보 데이터

- 프로그램은 이름, 백신 명, 접종 횟수, 나이, 지역명 정보가 저장된 파일 input_data.txt 을 LOAD 명령어를 통해 읽어 해당 정보를 클래스에 저장한다. input_data.txt 의 예시는 그림 2 와 같이 접종 대상자에 대한 정보가 저장되어 있다.

  * Pfizer, Moderna, AstraZeneca 접종 대상자는 백신 접종 횟수가 1 이하, 접종 완료자는 접종 횟수가 2 인 사람을 의미한다.

  * Janssen 접종 대상자는 백신 접종 횟수 0, 접종 완료자는 접종 횟수가 1 인 사람을 의미한다.

  * input_data.txt 에는 접종 대상자만 포함될 수 있다.

* 이름 정보는 항상 고유하며, 중복인 경우는 없다고 가정한다.
* 데이터는 ' '(space bar 한 번)로 구분한다.

```
Tom Pfizer 0 38 Seoul
Emily Moderna 0 21 Incheon
John Pfizer 0 17 Seoul
Erin AstraZeneca 1 51 Daegu
Elsa Janssen 0 49 Busan
…
…
```

그림 2. 데이터가 저장되어 있는 텍스트 파일의 예

표 1. 백신별 접종 횟수

백신 명	접종 횟수
Pfizer	2
Moderna	2
AstraZeneca	2
Janssen	1

2. B+ tree

- B+ tree 의 차수는 3 으로 구현한다.
- 주어진 input_data.txt 에 저장된 데이터를 읽은 후, 접종 대상자는 B+ tree 에 저장한다.
- ADD 명령어로 추가되는 데이터를 읽은 후, B+ tree 에 저장한다. B+ tree 에 없으면 node 를 새로 추가하며, 존재하는 경우 접종 횟수만 증가시킨다. ADD 로 추가된 데이터가 이미 접종 횟수가 2 인 접종 완료자인 경우 예외처리를 한다.
- B+-tree 에 저장되는 데이터는 VaccinationData class 로 선언되어 있으며, 멤버 변수로는 이름, 백신 명, 접종 횟수, 나이, 지역명이 있다.
- 이름 정보는 항상 고유하며, 중복으로 입력되는 경우는 없다고 가정한다.
- B+-tree 는 그림 3 의 예시와 같이 이름을 기준으로 구성하며, 대소문자를 구별한다. (아스키 문자 코드 기준으로 구별, 대문자 < 소문자)
- B+-Tree 는 인덱스 노드(BpTreeIndexNode)와 데이터 노드 (BpTreeDataNode)로 구성되며, 각 노드 클래스는 B+-tree 노드 클래스(BpTreeNode)를 상속받는다.
- 데이터 노드는 이름, 백신 명, 접종 횟수, 나이, 지역명을 저장한 VaccinationData 를 map 컨테이너 형태로 가지고 있다.

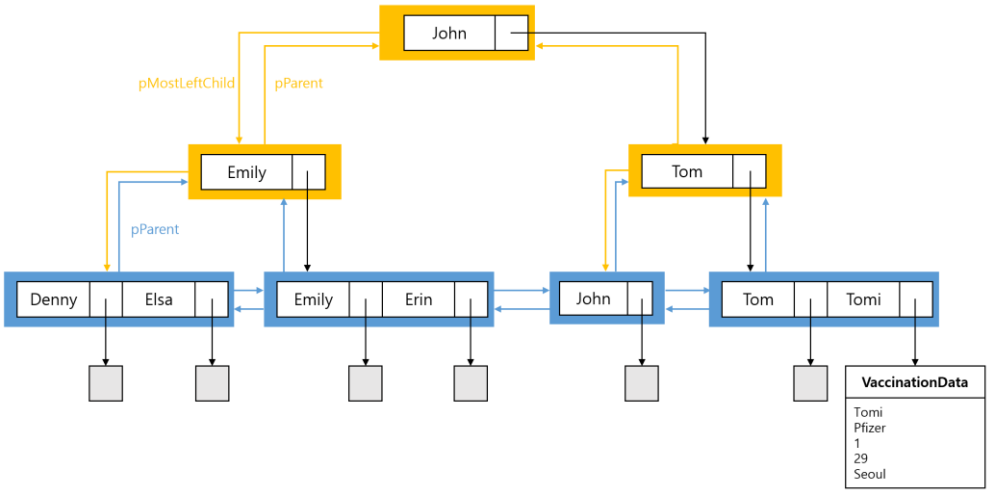

그림 3. B+ tree 의 예

3. AVL Tree

- ADD 명령어로 B+tree 에 저장된 데이터의 접종 횟수를 업데이트 후에, 접종 완료자는 AVL tree 에 저장한다. 이때, 해당 데이터는 B+ tree 에서 삭제하지 않고 그대로 유지한다.
- AVL tree 에 저장되는 데이터는 VaccinationData class 로 선언되어 있으며, 멤버 변수로는 이름, 백신 명, 접종 횟수, 나이, 지역명이 있다.
- AVL tree 는 그림 4 예시와 같이 이름을 기준으로 정렬하며, 대소문자를 구별하지 않는다.
- AVL tree 는 각 노드마다 balance factor 를 가지고 있으며, 이를 활용하여 트리의 균형을 유지한다.

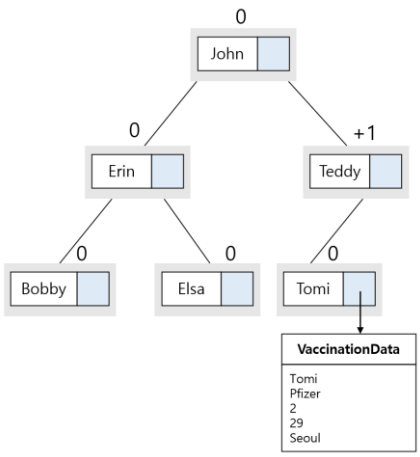

그림 4. AVL tree 의 예

☐ **Program implementation**

프로그램은 명령어를 통해 동작하며, 실행할 명령어가 작성되어있는 커맨드 파일(command.txt)을 읽고 명령에 따라 순차적으로 동작한다.

명령어	명령어 사용 예 및 기능
LOAD	사용 예) LOAD 텍스트 파일 input_data.txt 의 데이터 정보를 불러오는 명령어로, 텍스트 파일에 데이터 정보가 존재할 경우 텍스트 파일을 읽어 B+ tree 에 데이터를 저장한다. LOAD 는 command.txt 의 첫번째 줄에 한 번만 사용된다. [에러 코드 출력] • 텍스트 파일이 존재하지 않는 경우 • 텍스트 파일 안에 데이터가 존재하지 않는 경우 • B+ tree 가 존재하는 경우 [출력 포맷 예시] ========LOAD======== Success ==================== ========ERROR======== 100 ====================
VLOAD	사용 예) VLOAD AVL tree 에 저장된 모든 데이터 정보를 Print_vector 에 불러오는 명령어이다. queue, stack STL 을 이용하여 재귀적인 방법을 사용하지 않고 데이터를 불러온다. 기존에 Print_vector 가 존재할 경우, 비우고 새롭게 데이터를 불러온다. [에러 코드 출력] • AVL Tree 가 비어있는 경우

	[출력 포맷 예시] ========VLOAD======== Success ==================== ========ERROR======== 200 ======================
ADD	사용 예) ADD kevin Pfizer 32 Seoul B+ tree 에 데이터를 직접 추가하기 위한 명령어로, 4 개의 인자를 추가로 입력한다. 첫 번째 인자부터 이름, 백신 명, 나이, 지역명을 입력한다. 추가된 데이터의 이름이 B+ tree 에 존재하는 경우, 접종 횟수만 1 추가한다. B+ tree 에 이름이 존재하지 않는 경우, B+ tree 에 데이터를 새로 추가한다. 접종 완료자의 경우, AVL tree 에 추가한다. [에러 코드 출력] • 인자 4 개를 입력하지 않은 경우 • B+ tree 에 추가하려는 대상이 이미 백신 완료자인 경우 [출력 포맷 예시] ======== ADD ======== kevin Pfizer 32 Seoul ==================== ========ERROR======== 300 ======================
SEARCH_BP	사용 예) SEARCH_BP John SEARCH_BP b e SEARCH_BP 명령어의 인자로 이름을 입력하는 경우, B+ tree 에 저장되어 있는 데이터만을 출력한다. SEARCH_BP 의 인자로 2 개 (시작 알파벳, 끝 알파벳)를 입력한다면 시작 알파벳을 포함하는 단어부터 끝 알파벳을 포함하는 단어까지 범위 검색의 결과를

	출력한다. 대소문자는 구별하여 진행한다. (헷갈리면 밑의 예시를 참고해주세요.) [에러 코드 출력] • 인자 1 개 또는 2 개를 입력하지 않은 경우 • B+ tree 에 데이터가 없는 경우 • 해당하는 이름에 대한 데이터가 없는 경우 [출력 포맷 예시] 1) SEARCH_BP John ======= SEARCH_BP ======= John Pfizer 0 17 Seoul ========== 2) SEARCH_BP b e (b~e 사이의 데이터가 없는 경우) ========ERROR======== 400 ========== 3) SEARCH_BP B E → B 를 포함하는 단어와 E 를 포함하는 단어까지 출력입니다. ======= SEARCH_BP ======= Denny Pfizer 0 32 Gyeonggi Elsa Janssen 1 49 Busan Emily Moderna 0 21 Incheon Erin AstraZeneca 0 51 Daegu ========== ========ERROR======== 400 ==========
SEARCH_AVL	사용 예) SEARCH_AVL Tom SEARCH_AVL 명령어에 인자로 이름을 입력하여 AVL tree 에 저장되어 있는 이름에 대한 데이터만을 출력한다. [에러 코드 출력] • 인자 1 개를 입력하지 않은 경우 • AVL tree 에 데이터가 없는 경우

	• 해당하는 이름이 없는 경우 [출력 포맷 예시] Command : SEARCH_ AVL Jacob ====== SEARCH_AVL ====== Jacob Pfizer 2 32 Gyeonggi ========== ========ERROR======== 500 ==========
VPRINT	사용 예) VPRINT A VPRINT B Print_vector 에 저장된 데이터를 조건 A, B 에 따라 정렬을 하고 출력한다. 조건은 다음과 같다. 조건 A) - 백신 명 오름차순, 같을 경우 나이 오름차순, 같을 경우 이름 오름차순 조건 B) 지역 명 오름차순, 같을 경우 나이 내림차순, 같을 경우 이름 오름차순 조건 A, B 의 정렬은 반드시 C++ STL 에서 제공하는 sort() 함수를 이용해서 구현하며, 대소문자는 구별하지 않는다. [에러 코드 출력] • AVL tree 가 비어 있는 경우 [출력 포맷 예시] 1) VPRINT A ======= VPRINT A ======= Bob Janssen 1 33 Incheon Emma Pfizer 2 24 Seoul Rose Pfizer 2 24 Seoul Jacob Pfizer 2 32 Gyeonggi

	========== 2) VPRINT B ======= VPRINT B ======= Jacob Pfizer 2 32 Gyeonggi Bob Janssen 1 33 Incheon Emma Pfizer 2 24 Seoul Rose Pfizer 2 24 Seoul ========== 3) VPRINT B (Print_vector 가 비어있는 경우) ========ERROR======== 600 ==========
PRINT_BP	사용 예) PRINT_BP B+ tree 에 저장된 데이터를 이름 순서로 전부 출력한다. [에러 코드 출력] • B+ tree 가 비어 있는 경우 [출력 포맷 예시] ======= PRINT_BP ======= Denny Pfizer 0 32 Gyeonggi Elsa Janssen 1 49 Busan Emily Moderna 0 21 Incheon Erin AstraZeneca 0 51 Daegu John Pfizer 1 17 Seoul Tom Pfizer 1 38 Seoul Tommy Pfizer 1 38 Seoul ========== ========ERROR======== 700 ==========
EXIT	사용 예) EXIT 프로그램 상의 메모리를 해제하며, 프로그램을 종료한다.

	출력 포맷 예시) =======EXIT======== Success ==================

☐ **Requirements in implementation**

- 모든 명령어는 command.txt 에 저장하여 순차적으로 읽고 처리한다.
- 모든 명령어는 반드시 대문자로 입력한다.
- 명령어에 인자(Parameter)가 모자라거나 필요 이상으로 입력받으면 에러 코드를 출력한다.
- 로그 파일(log.txt)에 출력 결과를 반드시 저장한다.
- 로그 파일에 에러 결과를 반드시 저장한다.
- 프로그램이 종료될 때 메모리 누수가 발생하지 않도록 한다.

☐ **Error Code**

명령어	에러 코드
LOAD	100
VLOAD	200
ADD	300
SEARCH_BP	400
SEARCH_AVL	500
VPRINT	600
PRINT_BP	700
잘못된 명령어	800

☐ **동작 예시**

input_data.txt
Denny Pfizer 0 32 Gyeonggi Tom Pfizer 1 38 Seoul Emily Moderna 0 21 Incheon John Pfizer 1 17 Seoul

Erin AstraZeneca 0 51 Daegu

command.txt

```
LOAD
ADD Elsa Janssen 49 Busan
ADD Tommy Pfizer 38 Seoul
VPRINT
PRINT_BP
SEARCH_BP Tom
ADD John Pfizer 17 Seoul
ADD Emily Moderna 21 Incheon
ADD Emily Moderna 21 Incheon
ADD Emily Moderna 21 Incheon
SEARCH_AVL John
VLOAD
VPRINT A
ADD Tom Pfizer 38 Seoul
VLOAD
VPRINT B
```

log.txt

```
========LOAD========
Success
====================

======== ADD ========
Elsa Janssen 49 Busan
====================

======== ADD ========
Tommy Pfizer 38 Seoul
====================

========ERROR========
600
====================

======= PRINT_BP =======
Denny Pfizer 0 32 Gyeonggi
Elsa Janssen 1 49 Busan
Emily Moderna 0 21 Incheon
```

```
Erin AstraZeneca 0 51 Daegu
John Pfizer 1 17 Seoul
Tom Pfizer 1 38 Seoul
Tommy Pfizer 1 38 Seoul
======================

====== SEARCH_BP =======
Tom Pfizer 1 38 Seoul
======================

======== ADD ========
John Pfizer 17 Seoul
====================

======== ADD ========
Emily Moderna 21 Incheon
====================

======== ADD ========
Emily Moderna 21 Incheon
====================

========ERROR========
300
======================

====== SEARCH_AVL =====
John Pfizer 2 17 Seoul
====================

========VLOAD========
Success
====================

======= VPRINT A =======
Elsa Janssen 1 49 Busan
Emily Moderna 2 21 Incheon
John Pfizer 2 17 Seoul
====================
```

```
======== ADD ========
Tom Pfizer 38 Seoul
====================

========VLOAD========
Success
====================

======= VPRINT B =======
Elsa Janssen 1 49 Busan
Emily Moderna 2 21 Incheon
Tom Pfizer 2 38 Seoul
John Pfizer 2 17 Seoul
====================
```

□ 채점 기준

채점 기준	인자	점수
VLOAD	-	1
SEARCH_BP	이름	1
	시작 단어, 끝 단어	2
SEARCH_AVL	이름	2
VPRINT	조건 A	1
	조건 B	1
PRINT_BP	-	3
에러 코드	-	2
주석	-	2
총합		15

□ 제한사항 및 구현 시 유의사항

✓ 반드시 제공되는 코드(github 주소 참고)를 이용하여 구현하며, 작성된 소스 코드의 이름과 클래스, 함수 이름 및 형태를 임의로 변경하지 않는다.

✓ 클래스의 함수 및 변수는 자유롭게 추가 구현이 가능하다.

✓ 제시된 클래스를 각 기능에 알맞게 모두 사용한다.

✓ 프로그램 구조에 대한 디자인이 최대한 간결하도록 고려하여 설계한다.

✓ 채점 시 코드를 수정해야하는 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 Makefile 을 사용하여 테스트하도록 한다.

□ 제출기한 및 제출방법

✓ 제출기한

- 2021 년 11 월 5 일 금요일 23:59:58 까지 제출

✓ 제출 방법

- 소스코드(Makefile과 텍스트 파일 제외)와 보고서 파일(pdf)을 함께 압축하여 제출
    - 확장자가 .cpp, .h, .pdf가 아닌 파일은 제출하지 않음(.txt 파일 제출 X)
    - 보고서 파일 확장자가 pdf 가 아닐 시 감점

- KLAS -> 과제 제출 -> tar.gz 로 과제 제출

✓ 제출 형식

- 학번_DS_project2.tar.gz (ex. 2020202001_DS_project2.tar.gz)

✓ 보고서 작성 형식 및 제출 방법

- 보고서 내용은 한글로 작성
- 보고서에는 소스코드를 포함하지 않음
- 아래 각 항목을 모두 포함하여 작성
  - Introduction : 프로젝트 내용에 대한 설명
  - Flowchart : 설계한 프로젝트의 플로우 차트를 그리고 설명
  - Algorithm : 프로젝트에서 사용한 알고리즘의 동작을 설명
  - Result : 모든 명령어에 대해 결과 화면을 캡처하고 동작을 설명
  - Consideration : 고찰 작성