Data Structure Lab. Project #2

2024년 10월 11일

Due date: 2024년 11월 17일 일요일 23:59:59 까지

항공권 정보 검색 시스템은 항공권 정보를 저장하고, 다양한 기준을 적용해 효율적으로 정렬 및 검색할 수 있는 시스템이다. 이번 프로젝트는 B^+ Tree, AVL Tree, STL vector를 이용해 항공권 정보 검색 시스템을 구현한다. 항공권 정보 시스템은 [항공사명, 항공편명, 도착지, 좌석 수, 상태]로 구성되어 있으며, 이를 이용해 특정 도착지에 대한 항공권 정보를 빠르게 파악할 수 있다. B^+ Tree를 이용해 항공권을 관리하며, AVL Tree를 이용해 탑승 완료한 항공권(좌석 수가 0인 항공권)을 관리한다. 그리고 Print_vector를 이용해 탑승 완료된 항공권을 다양한 정렬 방법으로 출력할 수 있다. 그림 1은 항공권 정보 검색 시스템의 구조이다. 자료구조의 구축 방법 및 조건에 대한 설명은 Program Implementation에서 자세히 설명한다.

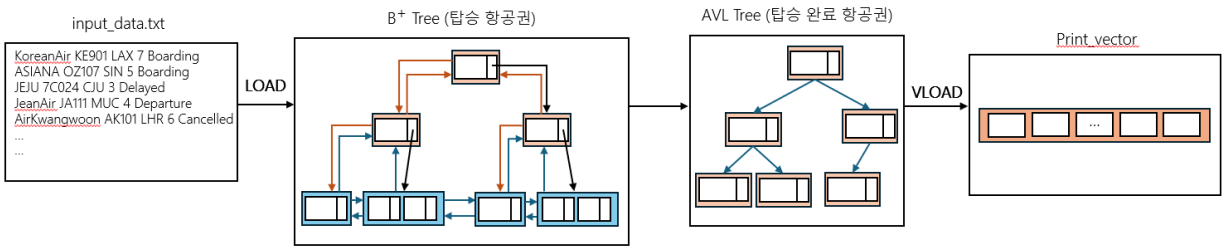

그림 1. 항공권 정보 검색 시스템의 구조

☐ Program Implementation

1. 항공권 정보 데이터

항공사명, 항공편명, 도착지, 좌석 수, 상태 정보가 포함된 데이터셋 input_data.txt을 LOAD 명령어를 사용해 해당 정보를 '항공권 정보 클래스(FlightData)'에 저장한다. input_data.txt의 예시는 그림2와 같이 항공권 정보가 저장되어 있다. 데이터는 ' '(Space Bar 한 번)으로 구분한다.

항공사명

a. 항공사명은 5개("KoreanAir", "ASIANA", "JEJU", "JeanAir", "AirKwangwoon") 로 구성되어 있다.

항공편명

a. 항공편명은 항상 5자리(숫자와 대문자 알파벳의 조합)로 구성되어 있다.
b. 항공편명은 항상 고유하며, 중복인 경우는 없다고 가정한다.

도착지명

a. 도착지명은 항상 (대문자 알파벳 3자리)로 구성되어 있다.
b. 항공사명이 "JEJU"의 경우 도착지는 항상 제주도(CJU)로 고정되어 있다.
c. 나머지 항공사의 경우 도착지는 자유이다.

좌석 수

a. 항공사명이 "KoreanAir"와 "ASIANA"의 경우 탑승 가능 좌석 수는 7이 최대이다.
b. 항공사명이 "JEJU"와 "JeanAir"의 경우 탑승 가능 좌석 수는 5가 최대이다.
c. 항공사명이 "AirKwangwoon'인 경우 탑승 가능 좌석 수가 6이 최대이다.

상태 정보

a. 운항 상태 정보는 4개(“Departure”, “Boarding”, “Delayed”, “Cancelled”)로 구성되어 있다.
b. 상태 정보가 “Boarding”, “Delayed”인 경우 승객을 받는다.
   → 좌석 수 감소 O
c. 상태 정보가 “Departure” 또는 “Cancelled”인 경우 더 이상 승객을 받지 않는다.
   → 좌석 수 감소 X
d. 상태 정보는 업데이트가 가능하다.

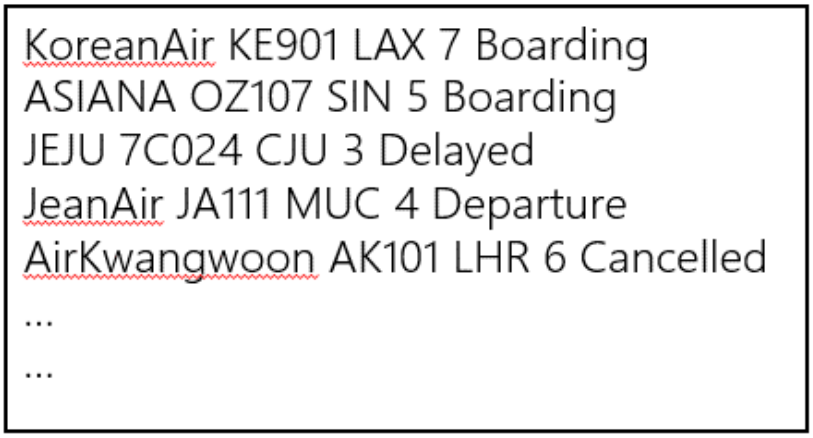

그림 2. 데이터가 저장되어 있는 텍스트 파일 예시

2. B^+ Tree

a. B^+ Tree의 차수는 3으로 구현한다.
b. ADD 명령어로 추가되는 데이터를 읽은 후, 상태가 Delayed, Boarding이며, 좌석 수가 남아 있는 항공권에 한해 B^+ Tree에 저장한다. B^+ Tree에 존재하지 않는 경우 Node를 새로 추가하며, 존재하는 경우 좌석 수를 감소시킨다. 만약 ADD로 추가된 데이터의 좌석 수가 0인 경우 더 이상 감소하지 않도록 예외처리를 한다.
c. B^+ Tree에 저장되는 데이터는 FlightData Class로 선언되어 있고, 멤버 변수로는 항공사명, 항공편명, 도착지명, 좌석 수, 상태 정보가 있다.
d. B^+ Tree는 그림 3의 예시와 같이 항공편명을 기준으로 구성한다.
e. B^+ Tree는 Index Node(BpTreeIndexNode)와 Data Node(BpTreeDataNode)로 구성되며, 각 Node Class들은 B^+ Tree Node Class(BpTreeNode)를 상속받는다.
f. Data Node는 항공사명, 항공편명, 도착지, 좌석 수, 상태 정보를 저장한 FlightData를 map 컨테이너 형태로 가지고 있다.

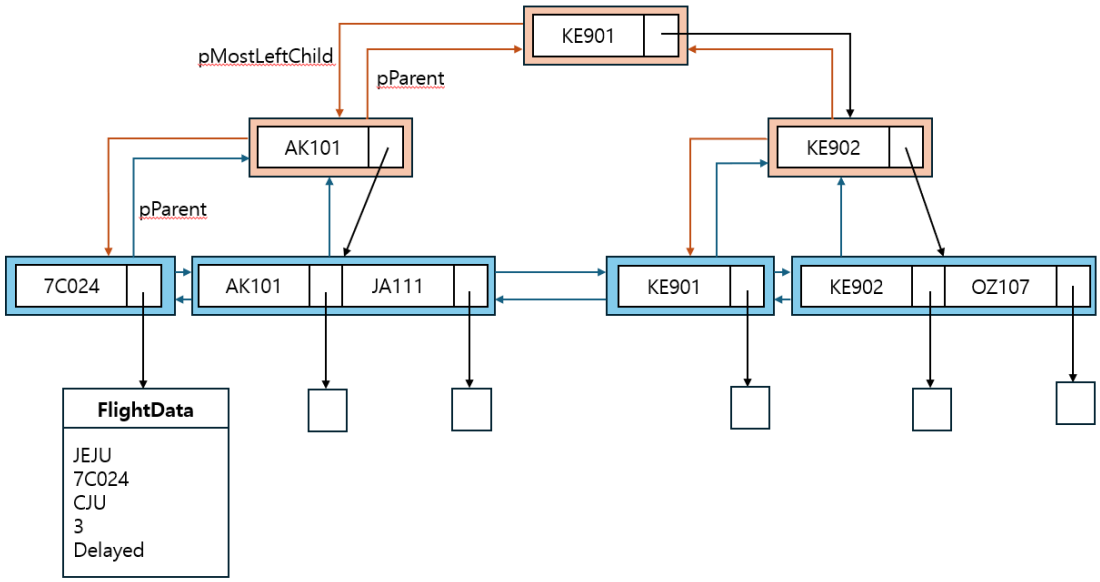

그림 3. B^+ Tree 예시

3. AVL Tree

a. ADD 명령어로 B^+ Tree에 저장된 데이터의 좌석 수를 업데이트 한 후에, 좌석 수가 0인 항공권의 경우 AVL Tree에 저장한다.

b. AVL Tree에 저장되는 데이터는 FlightData Class로 선언되어 있으며, 멤버 변수로는 항공사명, 항공편명, 도착지명, 좌석 수, 상태 정보가 있다.

c. AVL Tree는 그림 4 예시와 같이 항공편명을 기준으로 정렬하며, 대소문자를 구별하지 않는다.

d. AVL Tree는 각 Node마다 Balance Factor를 가지고 있으며, 이를 활용해 Tree의 균형을 유지한다.

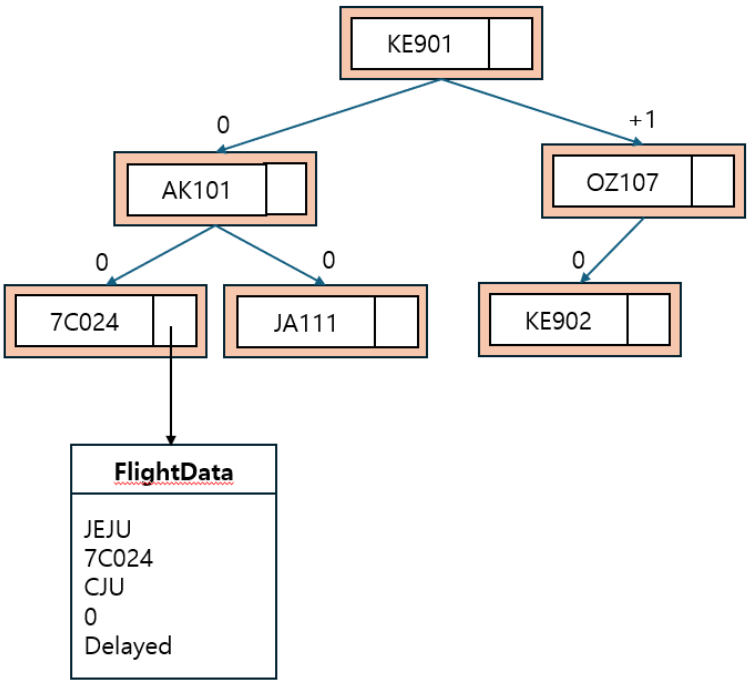

그림 4. AVL Tree 예시

☐ Functional Requirements

출력 포맷은 예시일 뿐, 실제 출력되는 데이터들과 차이가 있을 수 있습니다. 명령어에 인자가 존재하는 경우 "₩t"(탭)으로 구분합니다. **(예시에서는 스페이스로 사용했으니 주의할 것)**

명령어	명령어 사용 예시 및 기능
LOAD	사용 예시) LOAD input_data.txt 의 데이터 정보를 불러오는 명령어로, 텍스트 파일에 데이터 정보가 존재할 경우 텍스트 파일을 읽어 B+ Tree에 데이터를 저장한다. LOAD는 command.txt 의 첫번째 줄에 한 번만 사용된다. [에러 코드 출력] 텍스트 파일이 존재하지 않는 경우 텍스트 파일 안에 데이터가 존재하지 않는 경우 B+ Tree가 이미 존재하는 경우 [출력 포맷 예시] ========LOAD======== Success ==================== ========ERROR======== 100 ======================
VLOAD	사용 예시) VLOAD AVL Tree에 저장된 모든 데이터 정보를 Print_vector에 불러오는 명령어로, queue, stack STL을 이용해 재귀적인 방법을 사용하지 않고 데이터를 불러온다. 기존에 Print_vector 가 존재할 경우, 비우고 새롭게 데이터를 불러온다. [에러 코드 출력] - AVL Tree가 비어 있는 경우 [출력 포맷 예시] ========VLOAD======== Success ====================

	========ERROR======== 200 ==========
ADD	사용 예시) ADD ASIANA OZ999 MUC Boarding B+ Tree에 데이터를 직접 추가하기 위한 명령어로, 4개의 인자를 추가로 입력한다. 첫 번째 인자부터 항공사명, 항공편명, 도착지명, 상태를 입력한다. 추가된 데이터의 이름이 B+Tree에 존재하는 경우, 상태에 따라, 좌석 수를 1 감소시킬지 여부를 결정한다. 존재하지 않는 경우, B+Tree에 데이터를 새로 추가하며, 좌석 수의 경우 최대치로 고정한다. 좌석 수가 0인 항공편의 경우, AVL Tree에 추가하고, 더 이상 동일한 항공편을 받지 않는다. (상태 정보 또한 업데이트 하지 않는다.) 상태 정보 역시 업데이트 할 수 있다. 1) Cancelled → Boarding 좌석 수를 1 감소시킨다. 2) Boarding → Cancelled 좌석 수를 감소시키지 않는다. [에러 코드 출력] - 인자 4개를 입력하지 않은 경우 - B+Tree에 추가하려는 항공편의 자리가 없는 경우 - Cancelled → Boarding, Boarding → Boarding, Delayed → Delayed, Boarding → Cancelled를 제외한 모든 경우 [출력 포맷 예시] 1) ADD ASIANA OZ999 MUC Boarding ======== ADD ======== OZ999 | ASIANA | MUC | 7 | Boarding ========== ========ERROR======== 300 ==========

<table>
<tr><td>SEARCH_BP</td><td>사용 예시 1) SEARCH_BP 7C024
사용 예시 2) SEARCH_BP C K

SEARCH_BP 명령어는 B+ Tree에 저장된 데이터를 검색하는 명령어로 1개 또는 2개의 인자를 추가로 입력한다.
SEARCH_BP 명령어의 인자로 1개가 입력되는 경우, 해당 항공편의 검색 결과를 출력한다.
SEARCH_BP 명령어의 인자로 2개(시작 단어, 끝 단어)가 입력되는 경우, 범위 검색의 결과를 출력한다.
예를 들어, 인자로 C와 K가 입력되는 경우 C로 시작되는 항공편부터 K로 시작하는 항공편까지 모두 출력한다.

[에러 코드 출력]
- 인자에 1개 또는 2개를 입력하지 않은 경우
- B+ Tree에 데이터가 없는 경우
- 해당하는 항공편이 없는 경우

[출력 포맷 예시]
1) SEARCH_BP 7C024
========SEARCH _BP========
7C024 | JEJU | CJU | 2 | Delayed
==========

2) SEARCH_BP C K
========SEARCH _BP========
JA111 | JeanAir | MUC | 4 | Departure
KE901 | KoreanAir | LAX | 3 | Boarding
KE902 | KoreanAir | CJU| 1 | Delayed
==========

========ERROR========
400
==========</td></tr>
<tr><td>SEARCH_AVL</td><td>사용 예시) SEARCH_AVL JA115
SEARCH_AVL 명령어에 인자로 항공편명을 입력해 AVL Tree에 저장되어 있는 이름에 대한 데이터만을 출력한다.

[에러 코드 출력]
- 인자 1개를 입력하지 않은 경우
- AVL Tree에 데이터가 없는 경우</td></tr>
</table>

	- 해당하는 이름이 없는 경우 [출력 포맷 예시] 1) SEARCH_ AVL JA115 ====== SEARCH_AVL ====== JA115 | JeanAir | LAX | 0 | Delayed ========== ========ERROR======== 500 ==========
VPRINT	사용 예시 1) VPRINT A 사용 예시 2) VPRINT B Print_vector에 저장된 데이터를 조건 A, B에 따라 정렬하고 출력한다. 조건은 다음과 같다. 조건 A) 1. 항공사명 오름차순 2. (같을 경우) 도착지명 오름차순 3. (같을 경우) 상태 정보 내림차순 조건 B) 1. 도착지명 오름차순 2. (같을 경우) 상태 오름차순 3. (같을 경우) 항공사명 내림차순 [에러 코드 출력] - AVL Tree가 비어 있는 경우 [출력 포맷 예시] 1) VPRINT A ======= VPRINT A ======= AirKwangwoon | AK109 | LHR | Delayed JeanAir | JA003 | LHR | Delayed JeanAir | JA001 | LHR | Boarding JeanAir | JA002 | MUC | Boarding ========== 2) VPRINT B

	======= VPRINT B ======= AirKwangwoon | AK001 | CJU | Boarding ASIANA | AK109 | CJU | Boarding JeanAir | JA002 | CJU | Delayed JEJU | 7D003 | CJU | Delayed KoreanAir | JA001 | SIN | Boarding ========== 3) (Print_vector가 비어 있는 경우) ========ERROR======== 600 ==========
PRINT_BP	사용 예시) PRINT_BP B+ Tree에 저장된 데이터를 항공편명을 기준으로 오름차순으로 전부 출력한다. [에러 코드 출력] - B+Tree에 저장된 데이터가 없는 경우 [출력 포맷 예시] ========PRINT_BP======== 7C024 | JEJU | CJU | 2 | Delayed AK101 | AirKwangwoon | LHR | 6 | Cancelled JA111 | JeanAir | MUC | 4 | Departure KE901 | KoreanAir | LAX | 3 | Boarding KE902 | KoreanAir | CJU| 1 | Delayed OZ107 | ASIANA | SIN | 2 | Boarding ========== ========ERROR======== 700 ==========
EXIT	사용 예시) EXIT 프로그램 상의 메모리를 해제하며, 프로그램을 종료한다. [출력 포맷 예시] ========EXIT======== Success

	==========

☐ 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

☐ Requirements in Implementation

1. 모든 명령어는 명령어 파일(command.txt)에 저장해 순차적으로 읽고 처리한다.
2. 출력 형식은 예시로, 실제 출력되는 데이터들과는 차이가 있을 수 있다.
3. 모든 명령어는 반드시 대문자로 입력한다.
1. 명령어에 인자가 존재하는 경우 Tab("₩t")으로 구분한다.
2. 명령어에 인자(Parameter)가 모자라거나 필요 이상으로 입력 받을 시 에러 코드를 출력한다.
3. 로그 파일(log.txt)에 출력 결과와 에러 결과를 반드시 저장한다.
   - log.txt가 이미 존재할 경우 텍스트 파일 가장 뒤에 이어서 추가로 저장한다.
4. 프로그램 종료 시 메모리 누수(Memory Leak)가 발생하지 않도록 한다.

☐ 구현 시 반드시 정의해야 하는 Class

- ✓ **AVLTree**
  - AVL Tree Class
- ✓ **AVLNode**
  - AVL Tree의 Node Class
- ✓ **BpTree**
  - B^+ tree Class
- ✓ **BpTreeNode**
  - B^+ tree의 Node Class
- ✓ **BpTreeIndexNode**
  - B^+ tree의 Index node Class
- ✓ **BpTreeDataNode**
  - B^+ tree의 Data node Class
- ✓ **Manager**
  - 다른 Class 들의 동작을 관리해 프로그램 조정하는 역할 수행

☐ Files

1. command.txt
   프로그램을 동작시키는 명령어들을 저장하고 있는 파일
2. log.txt
   프로그램 출력 결과를 모두 저장하고 있는 파일
3. input_data.txt
   프로그램에 추가할 항공권 정보 데이터가 저장된 파일

☐ 제한사항 및 구현 시 유의사항

- ✓ 반드시 제공되는 코드(github 주소 참고)를 이용하여 구현하며 작성된 소스 파일의 이름과 클래스와 함수 이름 및 형태를 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 동작해야 한다. (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 명시한다.**

☐ 채점 기준

✓ 코드

채점 기준	점수
LOAD 명령어가 정상 동작하는가?	1
VLOAD 명령어가 정상 동작하는가?	2
ADD 명령어가 정상 동작하는가?	3
SEARCH_BP 명령어가 정상 동작하는가?	3
SEARCH_AVL 명령어가 정상 동작하는가?	3
VPRINT 명령어가 정상 동작하는가?	2
PRINT_BP 명령어가 정상 동작하는가?	1
총합	**15**

✓ 보고서

채점 기준	점수
Introduction을 잘 작성하였는가?	1
Flowchart를 잘 작성하였는가?	2
Algorithm을 잘 작성하였는가?	3

Result Screen을 잘 작성하였는가?	3
Consideration을 잘 작성하였는가?	1
총합	**10**

□ 제출기한 및 제출방법

✓ 제출기한
- 2024년 11월 17일 일요일 23:59:59까지 클라스(KLAS)에 제출

✓ 제출방법
- 소스코드와 보고서 파일(pdf)을 함께 압축하여 제출
  - •확장자가 .cpp, .h, .pdf가 아닌 파일은 제출하지 않음 (제출시 감점)
  - •보고서 파일 확장자가 pdf가 아닐 시 감점

✓ 제출 형식
- 학번_DS_project2.tar.gz (ex. 2023XXXXXX_DS_project2.tar.gz)
  - •제출 형식을 지키지 않은 경우 감점

✓ 보고서 작성 형식 및 제출 방법
- 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
  - 위의 각 항목을 모두 포함하여 작성
  - 보고서 내용은 한글로 작성
  - 보고서에는 소스코드를 포함하지 않음