데이터구조설계 프로젝트 2 보고서

2023202027 정정윤

**Introduction**

2차 프로젝트에서 만드는 것은 항공권 정보 검색 시스템으로, 공항 등지에서 항공권 정보를 저장하고, 관리하며, 필요할 때 빠르게 불러올 수 있도록 돕는 시스템이다. DB 액세스가 언제 이루어질지 모르고, 수많은 데이터들 중에서 빠르게 찾는 것이 중요하기 때문에, 주로 사전 정렬된 형태의 자료를 사용하는 경우가 많다. 따라서, 이번 프로젝트는 B+ Tree와 AVL Tree를 이용해 사전 정렬된, 이진 탐색을 통한 빠른 접근이 가능한 자료구조를 프로그래밍 하는 것이 중요한 프로젝트이다.

1차 프로젝트와 같이 명령어를 입력받고, 이에 따른 처리를 하여 log 파일에 남기는 구조이며, DB에서 자주 보이는 쿼리 처리와 유사한 처리 과정이다. 출력 시에는 원하는 데이터들을 찾아 사용자가 원하는 형태로 표현해줄 필요가 있어 print\_vector에 다양한 정렬 순서를 바탕으로 집어넣은 후 출력하도록 한다.

시스템은 우선 다양한 항공권 정보를 가진 데이터를 입력 파일에서 받아 B+ Tree에 저장한다. 입력되는 정보는 항공사명, 항공편명, 도착지, 좌석 수, 상태 정보가 포함되어 있으며, 이후 몇몇은 정렬 기준으로, 몇몇은 시스템 처리의 기준이 된다.

항공권 마다 줄바꿈이 이루어지며, 항공권 정보 사이에는 스페이스 바를 통한 공백이 하나 있어 이를 split하는 능력을 요구하기도 한다. 각 항공편에는 몇몇 조건들이 있다. 항공사에 따른 좌석 수 제한이 있어 좌석 수가 제공되지 않고, 항공사만 제공되는 다른 명령어가 사용되면, 이 수치들을 바탕으로 좌석 수를 추가해야 한다. 또, 항공편의 상태 정보를 통해 항공편에 대한 요구가 왔을 때, 좌석 수를 감소 시키거나 감소시키지 않는 처리를 진행해야 한다.

구현하는 B+ Tree의 차수는 3으로 노드는 최대 2개의 값을 가지고 있을 수 있다. 일반적으로 B Tree는 [주소, 인덱스, 주소, 인덱스, 주소]로 양끝에 주소를 두고 이를 기준으로 불러오는 방식을 취하는 경우가 있으나, 본 프로젝트에서는 스켈레톤 코드에서 사용한 방식인 가장 왼쪽 자식 주소를 저장한 후, [인덱스, 주소] 벡터를 다수 저장하는 구조를 가지고 있다. 이 저장에는 map을 사용하여 알아서 내부에서 인덱스를 기준으로 정렬되도록 한다.

B+ Tree의 정렬은 항공편명을 기준으로 이루어지므로, 즉 인덱스는 항공편명, 데이터는 항공편 정보가 된다.

이후 명령에 따라 B+ Tree에 있는 항공편 데이터들을 관리하며, 항공편의 좌석 수가 0이 될 경우, AVL Tree로 옮기도록 한다. AVL Tree의 데이터는 [인덱스, 항공편 정보]의 구조를 가지고 있으며, 인덱스는 여전히 항공편명이다. AVL Tree에 데이터를 넣을 때는 깊이를 기록하는 방식이 아닌 각 Node에 Balance Factor를 기록해두고 사용하는 방식으로 구현한다.

명령어는 LOAD를 통해 입력 파일로부터 항공편 정보들을 가져오는 것으로 시작하여, ADD 명령어를 통해 입력 파일이 아닌 명령어로부터 직접적으로 추가하거나, 항공편을 관리할 수 있고, SEARCH\_BP 명령어를 통해 B+ Tree에 저장된 항공편의 데이터들을 확인할 수 있다. 단일 데이터를 확인하거나 여러 데이터들을 모아 확인하는 기능 또한 구현해야 한다. SEARCH\_AVL 명령어를 통해 AVL Tree에 저장된 데이터를 확인할 수도 있다. 이후에는 VLOAD를 통해 AVL Tree에 저장된 값들을 Print\_Vector로 옮긴 후, VPRINT를 통해 특수하게 정렬된 기준에 따라 정렬하고 출력한다.

LOAD 명령을 제외한 다른 명령들은 여러번 수행되어도 큰 문제가 발생하지 않는다. 명령을 모두 수행한 후에는 프로그램을 종료하는 EXIT명령어를 사용할 수 있다.

**Flowchart**

d

**Algorithm**

d

**Result Screen**

d

**Consideration**

d