2019202005_DS_project2

수업 명 : 데이터구조실습
과제 이름 : DS project2
담당 교수님 : 이형근 교수님
학 번 : 2019202005
이 름 : 남종식

Introduction

이번 과제에서는 대출 도서 관리 프로그램을 구현을 진행합니다. 이때, 사용되는 알고리즘은 크게 b+ tree, heap 그리고 selection tree가 있습니다. 먼저 도서 관련 정보 데이터가 들어있는 loan_book.txt파일에서 데이터를 불러와 이를 바탕으로 b+트리로 대출 중인 도서를 관리합니다. 각 도서 관련 정보 데이터에는 도서 분류 코드와 대출 권수정보가 포함되어 있는데 분류 코드에 따라 대출 가능 권수가 다릅니다. 만약 대출 가능 권수가 넘어간다면 이는 selection tree로 해당 도서를 전달하고 b+tree에서 삭제합니다. 이때 selection tree는 min winner tree로 구현해야 하며 각 run은 각 분류 코드에 따라 heap으로 구성되어 있습니다. 이때 heap또한 min heap으로 구성되어 있습니다. 여기서 heap이나 selection tree중 한 쪽에서 insert나 delete를 통해 재정렬이 되는 경우에는 다른 한 쪽에서도 재정렬이 이루어져야 합니다. 이들의 데이터 정보로는 도서명, 도서분류코드, 저자, 발행 연도, 대출 권수에 대한 정보들이 저장되어 있습니다.

LOAD

LOAD명령어를 통해 loan_book.txt파일을 불러올 수 있으며 이를 b+ tree에 저장합니다. 오류가 없을 시에는 불러온 data가 잘 저장되었다는 success를 출력합니다. 만약 텍스트 파일이 존재하지 않거나 자료구조에 이미 데이터가 들어가 있으면 에러 코드 100을 출력합니다.
이때 데이터 조건으로는 도서명은 50자 미만의 소문자로 되어있으며 각각의 정보는 tab으로 구분되어 저장되어 있습니다.

ADD

ADD명령어를 통해 b+ tree에 직접 데이터를 추가할 수 있습니다. 총 4개의 정보 도서명, 분류 코드, 도서 저자, 발행 연도 중 하나라도 없으면 에러 코드 200을 출력합니다. 오류가 없을 시에는 추가하는 data를 출력합니다.

SEARCH_BP

SEARCH_BP명령어를 통해 b+ tree에 저장된 회원정보를 검색할 수 있습니다. 이때 인자로 1개 또는 2개가 입력되는 경우가 존재하는데 먼저 인자가 1개인 경우에는 인자로 도서명이 입력됩니다. 이 도서명이 b+ tree에 존재하면 도서명에 해당하는 도서정보를 출력하고 존재하지 않는다면 에로 코드 300을

출력합니다.
다음으로 인자가 2개가 입력되는 경우에는 첫번째 인자가 시작 단어 두번째 인자가 끝 단어입니다. 만약 a e 가 입력된다면 a로 시작하는 도서명부터 e로 시작하는 도서명까지 출력하면 됩니다.

PRINT_BP

PRINT_BP명령어를 통해 b+ tree에 저장된 도서정보를 출력할 수 있습니다. 이때 저장된 데이터들을 도서명을 기준으로 오름차순으로 출력하며 b+ tree에 저장된 데이터가 없으면 에러 코드 400을 출력합니다.

PRINT_ST

PRINT_ST명령어를 통해 selection tree에서 인자로 받은 분류코드에 해당하는 도서정보를 출력할 수 있습니다. 이때 저장된 데이터들을 도서명을 기준으로 오름차순으로 출력하며 selection tree에 저장된 데이터가 없거나 해당 코드의 heap자료구조가 없는 경우, 잘못된 분류코드를 입력하는 경우에는 에러 코드 500을 출력합니다.

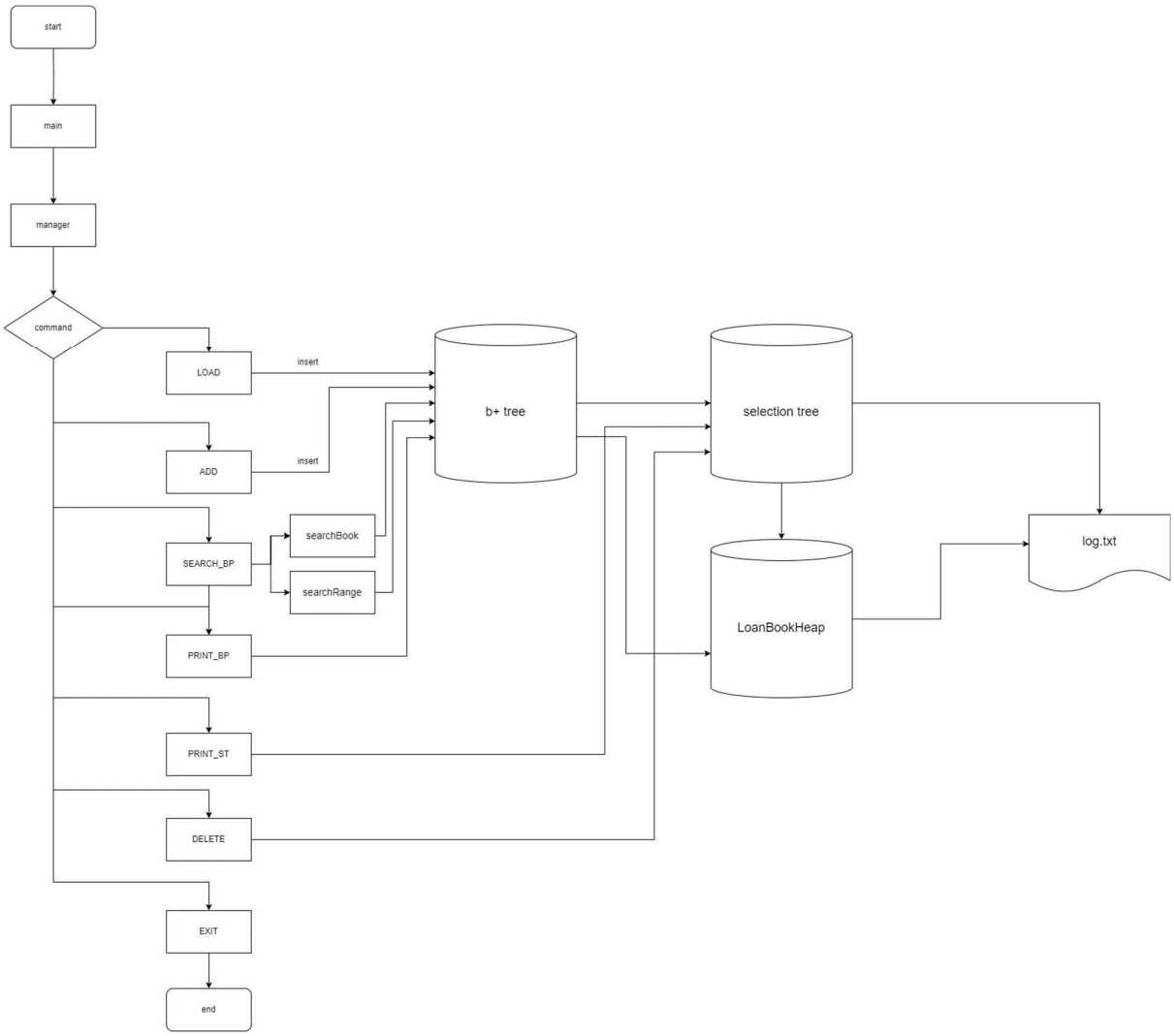
DELETE

DELETE명령어를 통해 selection tree의 root노드에 저장된 데이터를 삭제할 수 있습니다. 이때 도서가 제거된 후 heap에 저장된 도서들의 대소관계에 변경이 있을 시에는 heap에서도 재정렬이 이루어져야 합니다. 자료구조에 데이터가 존재하지 않을 시 에러 코드 600을 출력합니다.

EXIT

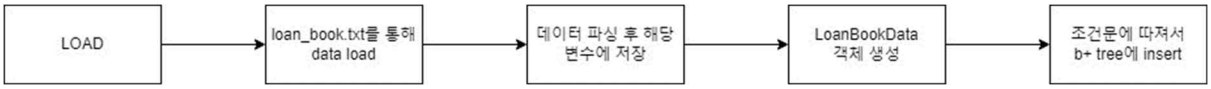
EXIT명령어를 통해 프로그램 상의 메모리를 해제하며, 프로그램을 종료합니다. 잘못된 명령어가 들어오면 에러 코드 700을 출력합니다.

Flowchart



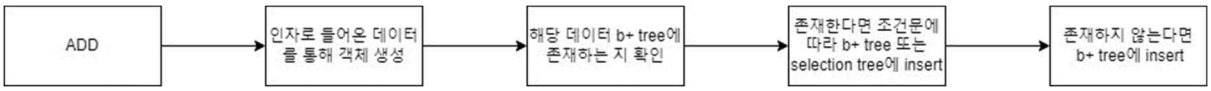
이번 프로젝트의 전체적인 흐름은 나타내는 manager의 순서도입니다. command.txt 파일에서 getline으로 한줄씩 읽어온 다음 이를 바탕으로 어떤 명령어가 실행될지 알 수 있습니다. 각 command가 들어오면 그에 해당하는 함수를 호출하게 됩니다. Command에 해당하는 동작을 정상 작동할 시에는 success문을 출력하며 실패 시에는 각 command에 해당하는 error code를 출력합니다. 마지막으로 command가 exit가 들어오면 프로그램은 종료하게 됩니다.

LOAD



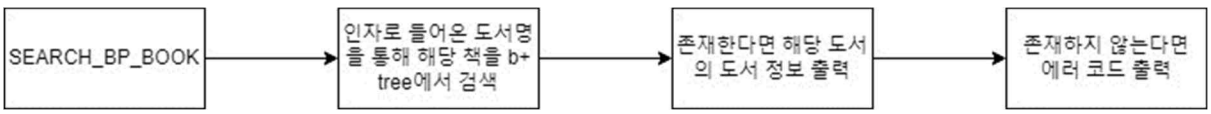
다음으로 LOAD command일 때 실행되는 순서도입니다. 만약 텍스트 파일이 존재하지 않거나 b+ tree에 데이터가 들어가 있는 상태라면 오류코드 100을 출력합니다. 그렇지 않으면 loan_book.txt에 저장되어 있는 데이터들을 읽어오고 데이터 파싱을 통해 도서 정보를 얻은 다음에 이를 바탕으로 객체를 생성하여 조건문에 따라 b+ tree에 insert합니다.

ADD



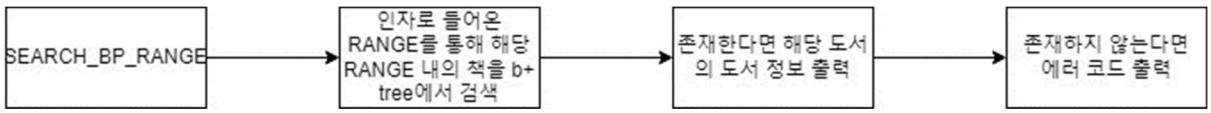
다음으로 ADD command일 때 실행되는 순서도입니다. 이 명령어를 통해 b+ tree에 직접 데이터를 추가할 수 있습니다. 인자들은 순서대로 도서명, 분류 코드, 저자, 발행연도이며 이를 바탕으로 객체를 생성하여 insert합니다. 이때, 조건문에 따라 분류코드와 대출 권수를 따져서 대출 가능 권수를 넘으면 selection tree에 insert, 넘지 않으면 b+ tree에 insert하게 됩니다. 이때, 이름으로 해당 도서가 b+ tree에 존재하는 지에 대해 먼저 확인하며 만약 존재한다면 대출 권수 정보를 얻어와 조건문에서 판별합니다. 만약 존재하지 않는다면 바로 b+ tree에 insert하면 됩니다. 인자가 하나라도 존재하지 않을 시에는 에러 코드 200을 출력합니다.

SEARCH_BP_BOOK



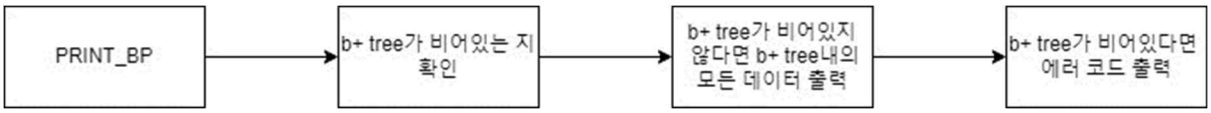
다음은 SEARCH_BP에 인자가 1개 들어온 경우입니다. 이때, 인자로써 도서명이 들어올 수 있으며 도서명으로 해당 책이 b+ tree에 존재하는 지에 대해 검색합니다. 만약 존재한다면 이에 해당하는 도서 정보를 출력하며, 존재하지 않는다면 에러 코드 300을 출력합니다.

SEARCH_BP_RANGE



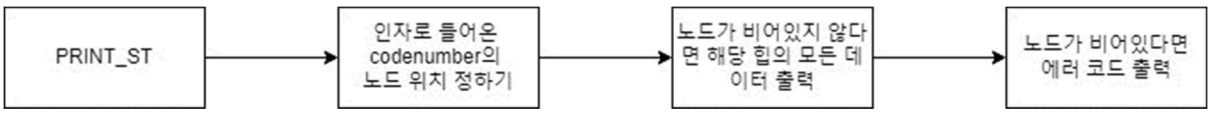
다음은 SEARCH_BP에 인자가 2개 들어온 경우입니다. 이때, 인자로써 range를 나타내는 두개의 알파벳이 들어올 수 있으며 첫번째 단어로 시작하는 도서명부터 두번째 단어로 시작하는 도서명까지 존재하는 모든 책에 대하여 오름차순으로 출력하면 됩니다. 만약 도서가 존재하지 않는다면 에러 코드 300을 출력합니다.

PRINT_BP



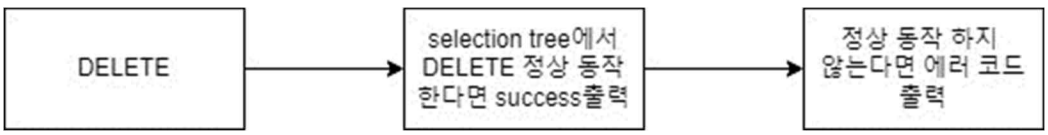
PRINT_BP를 통해 b+ tree에 저장되어 있는 모든 데이터들을 도서명을 기준으로 오름차순으로 출력할 수 있습니다. b+ tree에 저장된 데이터가 없는 경우 에러 코드 400을 출력합니다.

PRINT_ST



PRINT_ST를 통해 인자로 받은 분류 코드를 가지고 해당 분류코드의 힙에 저장되어 있는 모든 데이터들을 오름차순으로 출력할 수 있습니다.
Selection Tree에 저장된 데이터가 없는 경우, 도서분류코드에 대응되는 Heap 자료구조가 없는 경우, 잘못된 도서분류코드를 입력하는 경우에 에러 코드 500을 출력합니다.

DELETE



DELETE를 통해 selection tree에서 루트 노드에 저장되어 있는 데이터를 삭제할 수 있으며 삭제 후 selection tree와 heap을 모두 재정렬합니다. 이때 정상동작

하면 success를 출력하고 정상 동작 하지 않는다면 에러 코드 600을 출력합니다.

Algorithm

파일 읽기 후 명령어 및 인자들을 구분하는 Algorithm

일단 파일로부터 getline을 통해 데이터를 한 줄씩 읽어서 string으로 저장하고 명령어와 인자들을 구분하기 위해서는 stringstream을 사용하였습니다.

LOAD와 ADD를 통한 b+ tree에 데이터 삽입하는 Algorithm

LOAD명령어를 통해 loan_book.txt에 읽는 데이터를 읽어와 b+ tree 자료구조에 삽입할 수 있습니다. 이때 또한 인자들을 파싱 해서 도서명을 name, 도서분류코드를 code, 저자를 author, 발행 연도를 year, 대출 권수를 loan_count에 저장 후 b+ tree에 삽입합니다. 이때, b+ tree에는 데이터가 존재하지 않아야 하며 만약 존재한다면 에러 코드 100을 출력합니다.
ADD명령어 또한 거의 동일합니다. 이 또한 b+ tree에 데이터를 추가하는 명령어이지만 LOAD와 다르게 직접 b+ tree에 추가할 수 있습니다. 대출 권수를 제외한 4개의 인자 중 하나라도 존재하지 않으면 에러 코드 200을 출력합니다.
두 명령어 모두 BpTree.cpp에 구현되어 있는 Insert함수를 통해 b+ tree에 데이터 삽입이 이루어집니다.

인자로 들어온 이름에 해당하는 도서가 b+ tree에 저장되어 있는 도서인지 확인하는 Algorithm

searchDataNode를 통해 B+ 트리에서 특정 도서명을 가진 도서를 찾아 해당 데이터 노드를 반환하는 함수입니다. 루트 노드부터 시작하여 가장 왼쪽 자식 노드로 이동하면서 탐색을 시작합니다. 이는 B+ 트리의 특성 중 하나로, 모든 리프 노드가 연결 리스트로 구성되어 있기 때문입니다. 다음, 현재 노드에서 데이터 맵을 순회하면서 코드 값과 일치하는 도서를 찾습니다. 도서명과 일치하는 도서를 찾은 경우, 현재 노드를 반환합니다.
현재 노드에서 찾지 못한 경우, 다음 데이터 노드로 이동하면서 계속해서 탐색을 진행합니다. 만약에 모든 노드를 순회한 후에도 도서를 찾지 못했다면, NULL을 반환합니다. 이러한 함수는 B+ 트리에서 데이터를 검색하는 데 사용될 수 있으며, 도서명으로 데이터를 빠르게 찾을 수 있도록 도와줍니다.

b+ tree에서 데이터 노드를 분할하는 작업을 하는 Algorithm

먼저 B+ 트리에서는 중간 지점에서 분할하며 반복문을 통해 데이터 노드의 맵을 순회하면서 데이터를 적절한 위치에 삽입합니다. 분할 지점에 도달하면 새로운 데이터 노드에 데이터를 삽입하고, 필요한 경우 부모 노드에도 인덱스를 삽입합니다. 데이터 노드의 중간 이후의 데이터들은 새로운 데이터 노드로 옮겨지고, 기존 데이터 노드에서 삭제됩니다. 새로운 데이터 노드를 기존 데이터 노드와 연결하고, 필요한 경우에는 부모와 인덱스 노드들도 연결합니다. 이 함수의 주요 목적은 B+ 트리에서 데이터 노드를 분할하고, 필요 한 경우에는 새로운 인덱스 노드를 생성하여 트리의 구조를 유지하는 것입니다.

SEARCH_BP를 통한 b+ tree에 저장된 도서정보를 출력하는 Algorithm

SEARCH_BP명령어가 실행된다면 먼저 인자를 한 개 또는 두개를 받을 수 있습니다. 먼저 인자가 1개가 들어오는 경우에는 도서의 제목이 들어오며 이때 해당 도서의 정보가 b+ tree에 저장되어 있다며 이에 해당하는 정보를 출력합니다. 인자가 두개가 들어오는 경우에는 도서 제목 첫 알파벳의 범위가 들어오며 첫번째 인자로 들어오는 알파벳으로 시작하는 도서부터 두번째 인자로 들어오는 알파벳으로 시작하는 도서까지 모두 출력합니다. 이때 Bptree에 구현되어 있는 searchBook, searchRange함수가 호출됩니다.

B+ tree에서 모든 리프 노드는 연결되어 있고 데이터들이 저장되어 있기 때문에 가장 왼쪽의 리프 노드부터 탐색을 시작합니다. 그 후 데이터 맵을 순회하면서 주어진 이름과 일치하는 도서를 찾습니다.

찾고자 하는 도서정보가 b+ tree에 존재하지 않는 경우와 인자가 1개 또는 2개를 입력하지 않는 경우에는 에러 코드 300을 출력합니다.

PRINT_BP를 통한 오름차순으로 노드의 정보들을 출력하는 Algorithm

PRINT_BP명령어를 통해 b+ tree에 저장되어 있는 노드들을 오름차순으로 출력할 수 있습니다. B+ tree에는 모든 데이터가 리프 노드에 저장되어 있기 때문에 일단 가장 왼쪽의 리프 노드로 이동합니다. 이동 후 데이터 맵을 순회하면서 모든 노드를 순서대로 출력하면 됩니다. 이미 insert를 할 때 데이터들이 도서명의 오름차순으로 저장되어 있기 때문입니다.

만약 b+ tree에 데이터가 존재하지 않는다면 에러 코드 400을 출력합니다.

PRINT_ST를 통한 오름차순으로 노드의 정보들을 출력하는 Algorithm

PRINT_ST명령어를 통해 selection tree에 저장되어 있는 노드들을 오름차순으로 출력할 수 있습니다. 인자로 분류 코드가 들어오면 해당 분류코드에 해당하는 도서의 정보들을 오름차순으로 모두 출력하면 됩니다.
만약 selection tree에 데이터가 존재하지 않거나 해당 분류코드의 heap이 없는 경우, 도서 분류코드가 잘못된 입력된 경우에는 에러 코드 500을 출력합니다.

DELETE를 통한 selection tree의 루트 노드에 저장된 데이터를 삭제하는 Algorithm

Selection tree에서 루트 노드를 삭제하는 알고리즘입니다. 루트 노드를 삭제하기 위해서 힙의 마지막 노드와 루트 노드의 자리를 바꾸고 마지막 자리에 있는 기존의 루트 노드를 삭제합니다. 그 후 heapifyDown함수를 통해 루트 노드와 자식 노드의 크기를 비교해서 힙을 재정렬 해줍니다.

B+ tree

먼저 b+ tree에 insert하는 과정입니다. 트리가 비어있을 때 새로운 BpTreeDataNode 객체를 생성합니다. 해당 노드에 도서 데이터를 삽입합니다. 루트를 새로 생성한 노드로 설정합니다. 연산을 성공적으로 완료하고 true를 반환합니다. 루트 노드가 이미 존재할 때 도서 데이터가 트리에 존재하는지 확인합니다. 데이터가 트리에 없을 경우에는 트리를 순회하면서 적절한 위치를 찾아 도서 데이터를 삽입합니다. 데이터 노드가 너무 많은 데이터를 포함하고 있다면 데이터 노드를 분할합니다. 데이터가 이미 트리에 존재할 경우에는 해당 데이터가 속한 노드를 찾아 도서의 대출 횟수를 증가시킵니다.
다음으로 B+ 트리에서 특정 도서명을 찾아 해당 도서의 정보를 가지고 있는 노드를 찾는 과정입니다. 현재 노드를 루트 노드로 초기화합니다. 루트부터 가장 왼쪽 자식으로 이동하여 최하위 레벨의 가장 왼쪽 데이터 노드에 도달합니다. 현재 노드가 NULL이 아닌 동안 다음을 반복합니다. 현재 데이터 노드에서 도서 데이터를 검색하기 위해 dataIter를 사용합니다. 현재 데이터 노드에 속한 도서들을 차례로 확인하며 입력된 도서명과 일치하는 도서를 찾습니다.
일치하는 도서를 찾으면 현재 데이터 노드를 반환합니다. 현재 데이터 노드에서 검색이 끝나면 다음 데이터 노드로 이동합니다

.

LoanBookHeap

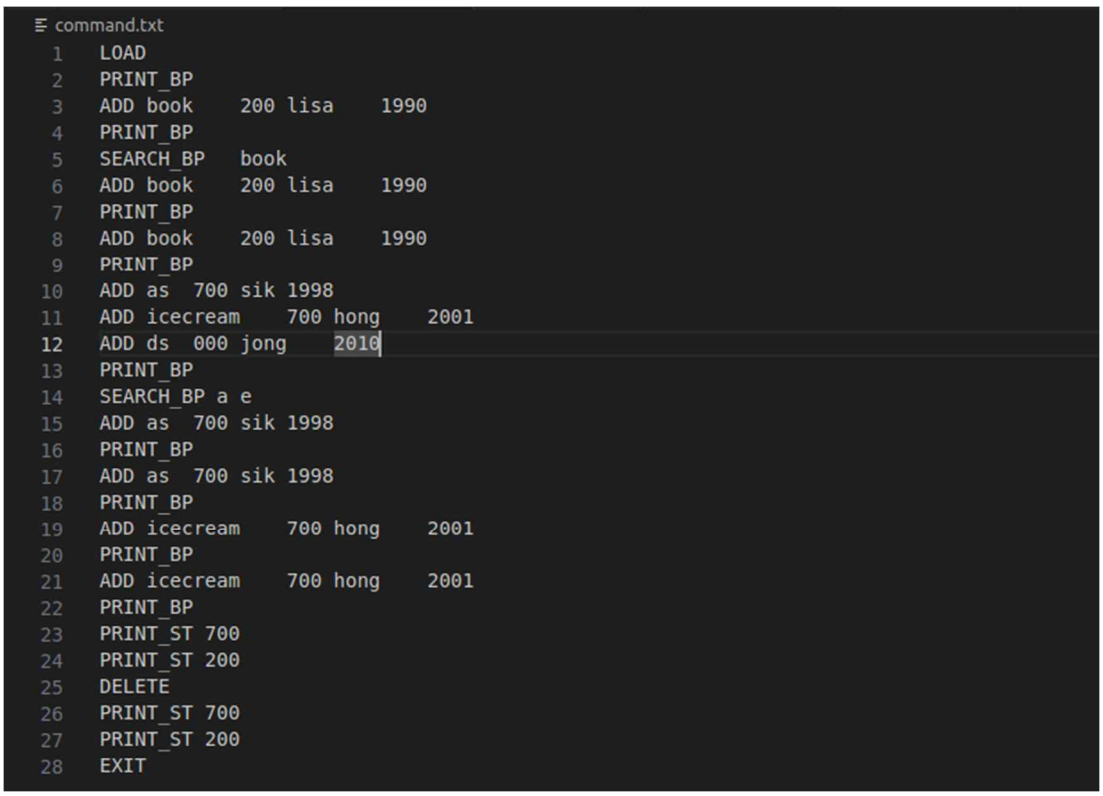
먼저 힙에 insert를 진행하는 방법으로는 level order순회 방식으로 insert하였습니다. 힙이 비어있으면, 새로운 노드를 루트로 설정하고 함수를 종료합니다. 힙이 비어있지 않은 경우, level order순회 방식으로 트리를 탐색하면서 새로운 노드를 삽입할 위치를 찾습니다. 현재 노드의 왼쪽 자식이 비어 있으면 새로운 노드를 왼쪽 자식으로 삽입하고, 그렇지 않으면 오른쪽 자식으로 삽입합니다. 힙을 재정렬하기 위해 heapifyUp 함수를 호출합니다. 이를 통해 새로 삽입된 노드를 부모 노드와 비교하여 필요에 따라 위치를 교환합니다. 다음으로 힙에 있는 데이터를 출력하는 함수입니다. 힙에 있는 데이터를 오름차순으로 출력하기 위해 우선순위 큐에 데이터를 담은 후, 우선순위 큐에서 데이터를 빼내면서 출력하도록 했습니다. 힙이 비어있으면 오류 메시지를 출력하고 함수를 종료합니다. 그 외의 경우에는, 우선순위 큐를 선언합니다. 재귀적으로 힙을 순회하면서 각 노드의 책 데이터를 우선순위 큐에 삽입합니다. 이때, 비교 함수를 이용하여 이름을 기준으로 최소 힙이 되도록 설정합니다. 우선순위 큐에서 데이터를 하나씩 빼내면서, 각 데이터의 코드에 따라 출력합니다. 다음은 힙의 delete하는 과정입니다. 힙이 비어있으면 오류 메시지를 출력하고 함수를 종료합니다. 비어 있지 않다면, 힙에서 가장 마지막에 있는 노드를 찾습니다. 노드와 마지막 노드의 데이터를 서로 교환합니다. 이렇게 하면 힙의 구조가 유지됩니다. 마지막 노드를 제거합니다. 루트에서부터 Heapify Down 연산을 수행하여 힙을 재정렬합니다. 노드의 값을 자식 노드들과 비교하면서 올바른 위치로 이동시킵니다.

Selection tree

먼저 selection tree에 insert하는 과정입니다. 트리의 루트 노드부터 시작하여, 주어진 newData의 코드값에 따라 트리의 경로를 따라 내려갑니다. newData의 코드값에 따라 왼쪽 또는 오른쪽 자식 노드로 이동하면서 경로를 설정합니다. 코드값이 0, 100, 200, ..., 700인 경우에 따라 서브 트리로 이동합니다. 원하는 노드에 해당하는 선택 트리 노드에 대해 해당 노드의 힙에 newData를 삽입하거나 갱신합니다. 코드에서는 해당 노드의 힙이 비어 있는 경우에 새로운 LoanBookHeap을 만들어 newData를 삽입하고, 이미 힙이 있는 경우에는 해당 힙에 newData를 추가합니다. 새로운 newData가 삽입되면 그것을 포함하는 상위 노드들 중에서 최대 3개의 레벨까지 거슬러 올라가면서, 해당 노드의

BookData를 갱신합니다. 새로운 데이터의 도서 이름이 해당 노드의 BookData의 이름보다 사전적으로 앞서거나 해당 노드의 BookData가 비어 있는 경우에만 갱신이 수행됩니다. 다음은 루트 노드를 삭제하는 과정입니다. 주어진 도서 데이터의 코드값에 따라서 선택 트리의 루트에서 시작하여 해당 데이터가 위치한 노드 및 그 하위 노드들(node1, node2, node3)을 설정합니다. 삭제될 노드(node, node1, node2, node3)의 BookData를 가상의 빈 데이터로 초기화합니다. node3에 연결된 힙(Heap)에서 Delete 연산을 수행합니다. node2, node1, node 순서로 각 노드에 대해 reorder 함수를 호출하여 노드의 BookData를 재조정합니다. 이 작업은 해당 노드의 하위 힙에서 가장 큰 값을 루트로 하는 데이터를 선택하여 해당 노드의 BookData로 설정하는 것입니다.

Result Screen



위 사진은 command.txt이며 여기에 작성된 순서대로 명령어에 대해 결과화면을 캡쳐하고 동작을 설명하겠습니다.

LOAD

loan_book.txt

```
book    200 lisa    1990    1
to kill a monchingbird  400 Harper Lee  1960    2
educated    300 Tara Westover   2018    1
the power of now    600 Echhart tolle   1997    0
```

log.txt

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

```

프로그램 동작을 확인하기 위해 임의로 loan_book.txt를 만들었습니다.
LOAD명령어를 통해 첫번째 사진에서 볼 수 있는 loan_book.txt의 데이터들을 가져와 b+ tree에 저장한 후 success를 출력한 모습을 확인할 수 있습니다.
Success가 출력은 되었지만 b+ tree에 제대로 저장되었는 지는 아직 모르기 때문에 뒤에서 PRINT_BP를 통해 출력 후 확인하겠습니다.

PRINT_BP

```
========PRINT_BP========
book/200/lisa/1990/1
educated/300/Tara Westover/2018/1
the power of now/600/Echhart tolle/1997/0
to kill a monchingbird/400/Harper Lee/1960/2
========================
```

LOAD후 b+ tree에 데이터들이 잘 저장되었는 지 확인하기 위해서 PRINT_BP명령어를 통해 b+ tree에 저장된 데이터들을 모두 출력하였습니다. 이때 출력은 도서명을 기준으로 오름차순으로 출력하며 도서명, 분류코드, 저자, 발행연도, 대출 권수 총 5가지의 정보들이 잘 출력된 모습을 확인할 수 있습니다.

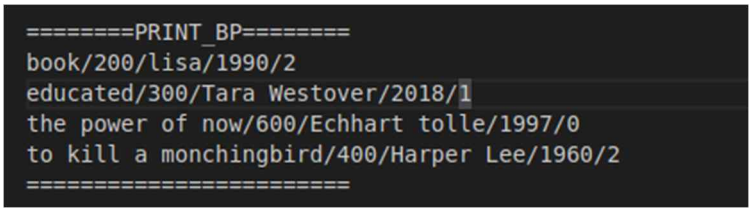
ADD

```
========ADD========
book/200/lisa/1990
===================
```

ADD book 200 lisa 1990
위 command를 진행한 결과이며 ADD명령어를 통해 b+ tree에 직접 데이터를 추가하는 동작을 하고 추가된 데이터의 정보가 잘 출력되는 모습을 확인할 수 있습니다. 이때 ADD명령어의 인자로는 도서명, 도서분류코드, 저자, 발행연도가 있습니다. 각각의 정보들은 tab으로 구분합니다.
b+ tree에 추가되었는 지에 대해서는 아직 확인할 수 없으므로 다음

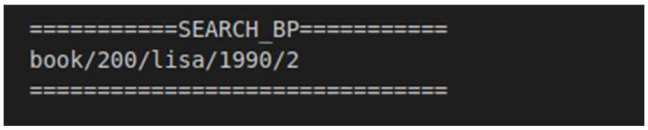
PRINT문에서 확인하겠습니다.

PRINT_BP



이전의 PRINT_BP명령어를 실행했을 때 출력문과 비교해봤을 때 도서명이 book인 책의 대출 권수가 1이 증가해 2가 된 모습을 확인할 수 있습니다. 이를 통해 ADD명령어가 잘 이루어진 모습 또한 확인할 수 있습니다.

SEARCH_BP 인자 1개



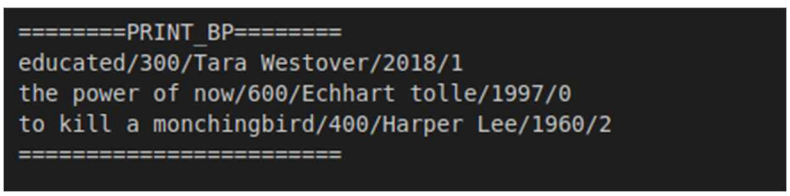
SEARCH_BP book

위 command를 진행한 결과이며 SEARCH_BP명령어와 뒤에 인자로 들어오는 도서의 이름을 통해 b+ tree에서 해당 이름의 도서명을 가지고 있는 노드 정보를 출력하는 모습을 확인할 수 있습니다. 이를 통해 loan_book.txt에서 LOAD가 잘 이루어지고 ADD를 통해 b+ tree에 데이터들이 잘 insert된 것 또한 알 수 있습니다.

ADD

다음으로 도서 분류 코드가 200인 도서가 대출 권수가 3이 되었을 때 selection tree로 넘어가고 b+ tree에서 삭제가 되는 지 확인하기 위해 도서명인 book인 책을 한번 더 ADD해주었습니다.

PRINT_BP



분류 코드가 000부터 200까지의 도서들은 대출 가능 권수가 3개이기 때문에 기존에 대출 권수가 2였던 book은 한번 더 ADD되어 대출 권수가 3이 되었고 selection tree로 전달되고 b+ tree에서 삭제되어 출력되지 않는 모습을 확인할 수 있습니다. Selection tree에 잘 전달되었는 지에 대해서는 PRINT_ST명령어를 통해 뒤에서 확인해보겠습니다.

ADD

```
=========ERROR=========
200
=======================
```

ADD book 200 lisa 1990

위 command를 진행한 결과이며 더 이상 대출이 불가능해진 책을 ADD하려고 할 때 에러 코드가 잘 출력되는 지 확인하기 위해 진행하였습니다. 에러 코드 200이 잘 출력되는 모습을 확인할 수 있습니다.

PRINT_BP

```
========PRINT_BP========
educated/300/Tara Westover/2018/1
the power of now/600/Echhart tolle/1997/0
to kill a monchingbird/400/Harper Lee/1960/2
========================
```

더 이상 ADD할 수 없어 이전에 에러 코드를 출력하고 b+ tree에도 추가되지 않은 모습을 확인할 수 있습니다.

ADD

```
========ADD========
as/700/sik/1998
===================

========ADD========
icecream/700/hong/2001
===================

========ADD========
ds/000/jong/2010
===================
```

ADD as 700 sik 1998

ADD icecream 700 hong 2001

ADD ds 000 jong 2010

위 command를 진행한 결과이며 다양한 데이터들을 ADD하였습니다.

PRINT_BP

```
========PRINT_BP========
as/700/sik/1998/1
ds/000/jong/2010/1
educated/300/Tara Westover/2018/1
icecream/700/hong/2001/1
the power of now/600/Echhart tolle/1997/0
to kill a monchingbird/400/Harper Lee/1960/2
========================
```

위에서 ADD한 데이터들이 모두 b+ tree에 잘 저장된 모습을 확인할 수 있습니다. 그리고 모두 오름차순으로 잘 출력되는 모습 또한 확인할 수 있습니다.

SEARCH_BP 인자 2개

```
===========SEARCH_BP===========
as/700/sik/1998/1
ds/000/jong/2010/1
educated/300/Tara Westover/2018/1
==============================
```

SEARCH a e

위 command를 진행한 결과이며 SEARCH_BP명령어와 뒤에 인자로 들어오는 첫번째 단어와 두번째 단어를 통해 b+ tree에 저장되어 있는 첫번째 단어로 시작하는 도서부터 두번째 단어로 시작하는 도서 사이에 있는 모든 도서의 노드 정보를 출력하는 모습을 확인할 수 있습니다. 인자가 1개가 들어왔을 때와 마찬가지로 loan_book.txt에서 LOAD가 잘 이루어지고 ADD를 통해 b+ tree에 데이터들이 잘 insert된 것 또한 알 수 있습니다.

ADD

```
========ADD========
as/700/sik/1998
===================
```

다음은 분류코드가 700인 도서가 대출 권수가 2가 되었을 때 selection tree에 잘 전달되고 b+ tree에서 삭제되는 지 확인하기 위해 실행한 결과입니다.

PRINT_BP

```
========PRINT_BP========
ds/000/jong/2010/1
educated/300/Tara Westover/2018/1
icecream/700/hong/2001/1
the power of now/600/Echhart tolle/1997/0
to kill a monchingbird/400/Harper Lee/1960/2
========================
```

도서명이 as인 책이 b+ tree에서 삭제된 모습을 확인할 수 있습니다. Selection tree에 잘 전달되었는 지에 대해서는 PRINT_ST명령어에서 확인하겠습니다.

ADD

```
=========ERROR=========
200
=======================
```

ADD as 700 sik 1998

위 command를 진행한 결과입니다. 분류코드가 700인 도서 as가 대출 가능 권수를 넘어 더 이상 ADD할 수 없어 에러 코드 200을 출력하는 모습을 확인할 수 있습니다.

PRINT_BP

```
========PRINT_BP========
ds/000/jong/2010/1
educated/300/Tara Westover/2018/1
icecream/700/hong/2001/1
the power of now/600/Echhart tolle/1997/0
to kill a monchingbird/400/Harper Lee/1960/2
========================
```

도서명이 as인 책을 더 이상 ADD할 수 없어 PRINT_BP후 출력 결과가 이전의 출력문과 같은 것을 확인할 수 있습니다.

```
========ADD========
icecream/700/hong/2001
====================

========PRINT_BP========
ds/000/jong/2010/1
educated/300/Tara Westover/2018/1
the power of now/600/Echhart tolle/1997/0
to kill a monchingbird/400/Harper Lee/1960/2
========================

=========ERROR=========
200
======================

========PRINT_BP========
ds/000/jong/2010/1
educated/300/Tara Westover/2018/1
the power of now/600/Echhart tolle/1997/0
to kill a monchingbird/400/Harper Lee/1960/2
========================
```

위 결과 또한 도서명이 icecream인 책을 ADD해 대출 가능 권수를 모두 만족해 b+ tree에서 삭제되어 PRINT_BP명령어를 통해 b+ tree에서 확인할 수 없으며 한번 더 ADD했을 때 에러 코드 200을 출력하는 모습을 확인할 수 있습니다.

PRINT_ST 인자 1개

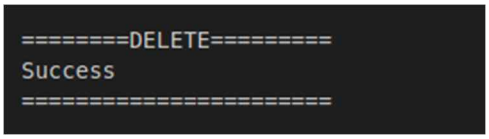
```
========PRINT_ST========
as/700/sik/1998/2
icecream/700/hong/2001/2
========================

========PRINT_ST========
book/200/lisa/1990/3
========================
```

PRINT_ST 700

PRINT_ST 200

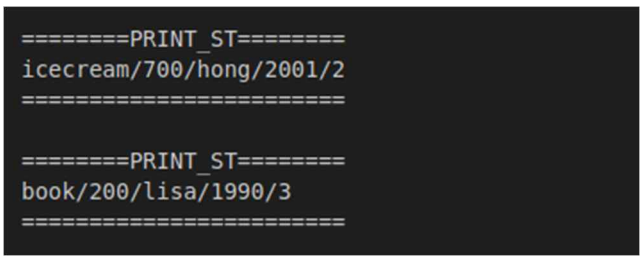
위 command를 진행한 결과이며 이전에 대출 가능 권수를 모두 만족한 책들이 selection tree로 넘어가 분류 코드 별로 출력되는 모습을 확인할 수 있습니다. 이때 또한, 모두 오름차순으로 출력되는 모습을 확인할 수 있습니다.

DELETE



다음은 selection tree의 루트 노드에 저장되어 있는 도서 정보를 지우는 명령어이며 루트 노드가 잘 지워지고 힙이 재정렬되어 success를 출력하는 모습을 볼 수 있습니다.

PRINT_ST 인자 1개



PRINT_ST 700

PRINT_ST 200

위 command를 진행한 결과이며 min winner tree로 이루어진 selection tree에는 알파벳 순이 가장 낮은 도서명인 as가 selection tree의 루트 노드에 저장되어 있었을 것입니다. 따라서 도서명이 as인 책이 삭제되어 분류 코드가 700인 힙을 출력하였을 때 출력되지 않는 모습을 확인할 수 있습니다. 이를 통해, 삭제가 잘 이루어진 것을 알 수 있습니다.

EXIT

마지막으로 프로그램이 성공적으로 종료된 모습을 확인할 수 있습니다.

Consideration

일단 저번 1차 과제에서는 각 자료구조들을 구현하고 그 안의 함수들을 확인할 때 마다 임의로 main함수를 수정하며 진행하였는데 이 방법이 처음에 자료구조가 하나일 때는 편했는데 자료구조들이 많아질수록 많이 헷갈리고 굉장히 불편함을 느꼈습니다. 그래서 이번 과제부터는 처음부터 manager파일을 구성하여 함수들의 동작을 확인하며 진행했습니다. manager파일에서 다른 알고리즘의 함수를 호출하기 위해 객체를 생성하는 과정에서 인스턴스화를 통해 스택에 할당했는데 이 부분에서 계속 오류가 발생했었습니다. 그래서 이 부분을 동적할당을 이용해 힙 메모리에 할당하여 문제를 해결할 수 있었습니다. 과제를 진행하면서 노드의 들어가 있는 데이터들을 계속 읽지 못하는 상황이 발생했는데 아마 스택에 할당 후 함수 종료 시 자동으로 소멸자가 호출되어 노드의 데이터를 읽지 못했던 것으로 예상됩니다. 그래도 이번에 시행착오를 겪어 앞으로 힙 메모리에 동적으로 할당하는 방법을 주로 이용할 것 같습니다.

이번 프로젝트에서 가장 어려웠던 점은 B+ tree의 구축이었던 거 같습니다. 처음에 구현을 시작할 때, 수업 시간에 배운 내용을 그대로 적용하여 구현을 하려 했지만 접근 자체가 굉장히 어려웠고 다양한 내용들을 찾아볼 수밖에 없었습니다. 그래도 이 과제를 통해 기말고사 시험범위에 해당하는 b+ tree에 대해서는 완벽하게 이해할 수 있었으며 이를 기말고사에 적용해 좋은 결과를 얻을 수 있을 것 같습니다. 힙과 selection tree는 중간고사 범위에 해당한 내용이어서 b+ tree보다는 더 쉽게 접근할 수 있었습니다. 이미 이해하고 있던 내용이라서 접근 자체가 어렵지는 않았지만 구현에 대해서 헷갈리는 부분이 많아서 이 점 또한 많이 찾아보고 생각하면서 구현했습니다. 그래도 수업 시간에 배운 내용들이 과제와 이어지는 부분이 대다수이다 보니까 배웠던 내용을 다시 공부하고 직접 실습할 수 있었던 점은 굉장히 좋았습니다. 특히, 이번 과제를 진행하면서 vector나 map을 유용하게 잘 사용하였고 사용법에 대해서 완벽하게 익히고 배울 수 있었습니다.