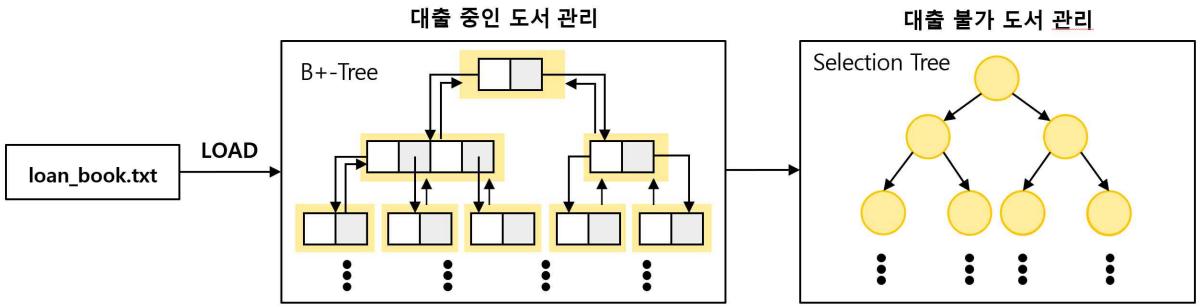
Data Structure Lab. Project #2

2023년 10월 13일

Due date: 2023년 11월 16일 목요일 23:59:59 까지

본 프로젝트에서는 B+-Tree와 선택 트리(Selection Tree), Heap을 이용하여 도서 대출 관리 프로그램을 구현한다. 대출 관리 프로그램은 도서명, 도서 분류 코드, 저자, 발행 연도, 대출 권수를 관리하며, 이를 이용해 대출 중인 도서와 대출 불가 도서에 대한 정보를 제공할 수 있다. B+-Tree를 이용하여 대출 중인 도서를 관리하며, 선택 트리를 이용하여 대출 불가 도서를 관리한다. 그림 1은 도서 대출 관리 프로그램의 구조이다. 자료구조의 구축 방법과 조건에 대한 자세한 설명은 **program implementation**에서 설명한다.



[그림 1] 도서 대출 관리 프로그램 구조

□ Program implementation

1) 도서 관련 정보 데이터

- 프로그램은 도서명(name), 도서 분류 코드(code), 저자(author), 발행 연도(year), 대출 권수(loan count) 정보가 저장된 파일 loan_book.txt를 LOAD 명령어를 통해 읽어 해당 정보를 클래스에 저장한다. loan_book.txt는 그림 2와 같이 도서에 대한 정보가 '\t'(탭)로 구분되어 저장되어 있다. 이때 도서명은 항상 고유하며, 중복인 경우는 없으며, 50자 이하로 가정한다.

```
1984	100	George Orwell	1949	0
To Kill a Monchingbird	400	Harper Lee	1960	0
Educated	300	Tara Westover	2018	0
The Power of Now	600	Echhart Tolle	1997	0
....
```

[그림 2] 데이터가 저장되어 있는 텍스트 파일의 예

- 도서 분류 코드는 000~600번대까지 있으며, 각 분류 코드에 따른 대출 가능 권수는 표 1과 같다.
  - 분류 코드가 000~200번대인 도서는 대출 권수가 2 이하, 대출 불가 도서는 대출 권수가 3

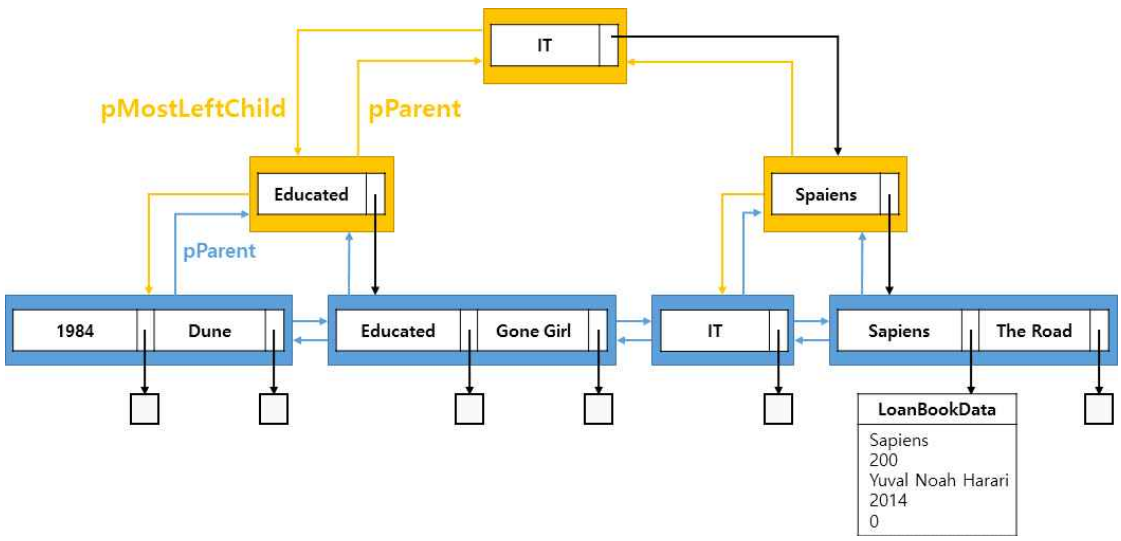
인 도서를 의미한다. 분류 코드가 300~400번대인 도서는 대출 권수가 3 이하, 대출 불가 도서는 대출 권수가 4인 도서를 의미한다. 분류 코드가 500~700번대인 도서는 대출 권수가 1 이하, 대출 불가 도서는 대출 권수가 2인 도서를 나타낸다.

[표 1] 도서 분류 코드 별 대출 가능 권수

도서 분류 코드	대출 가능 권수
000	3
100	3
200	3
300	4
400	4
500	2
600	2
700	2

2) B+-Tree

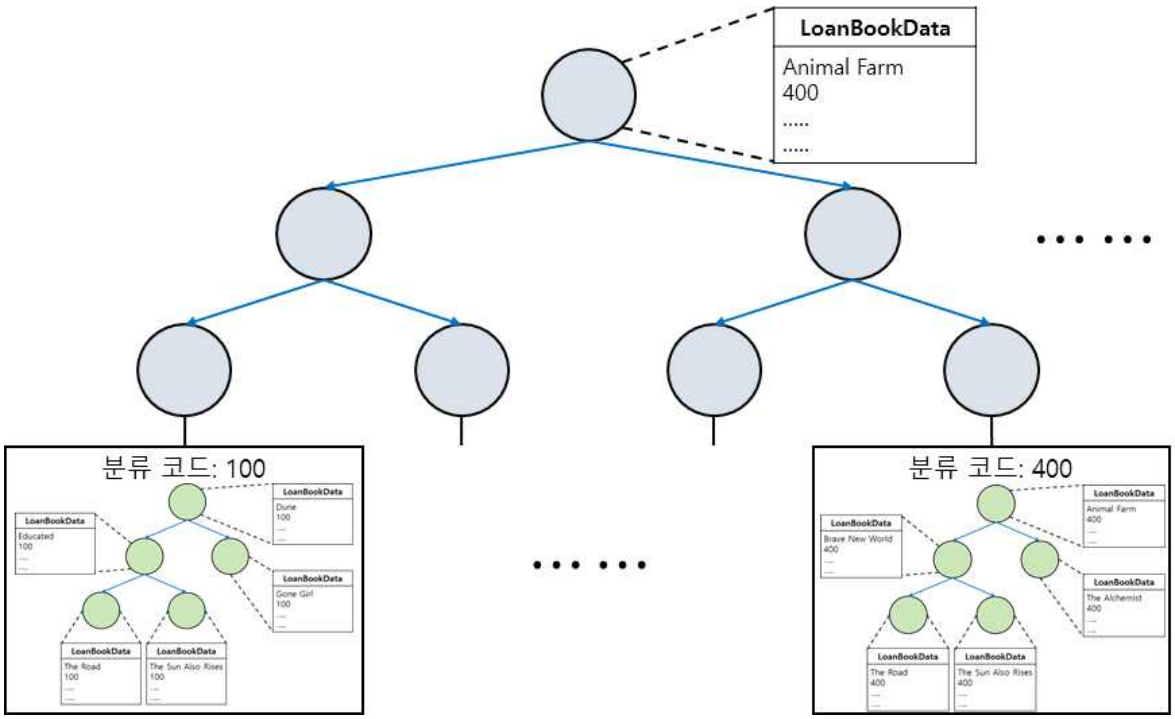
- B+-tree의 차수는 3으로 구현한다.
- 주어진 loan_book.txt에 저장된 데이터를 읽은 후, 대출 중인 도서는 B+-tree에 저장한다.
- ADD 명령어로 추가되는 데이터를 읽은 후, B+-tree에 저장한다. B+-tree에 없으면 node를 새로 추가하며, 존재하는 경우 대출 권수만 증가시킨다. 이후 대출 불가 도서(즉, 도서가 모두 대출된 경우)는 Selection Tree로 해당 도서를 전달하고, B+-tree에서 제거한다.
- B+-tree에 저장되는 데이터는 LoanBookData class로 선언되어 있으며, 멤버 변수로는 도서명, 도서 분류 코드, 저자, 발행 연도, 대출 권수가 있다.
- B+-tree는 그림 3의 예시와 같이 도서명을 기준으로 하며, 대소문자는 구별하지 않는다.
- B+-tree는 인덱스 노드(BpTreeIndexNode)와 데이터 노드(BpTreeDataNode)로 구성되며, 각 노드 클래스는 B+-tree 노드 클래스(BpTreeNode)를 상속받는다.
- B+-tree의 데이터 노드는 도서명, 도서 분류 코드, 저자, 발행 연도, 대출 권수를 저장한 LoanBookData를 map 컨테이너 형태로 가지고 있다.



[그림 3] B+-tree의 예시

3) Selection Tree

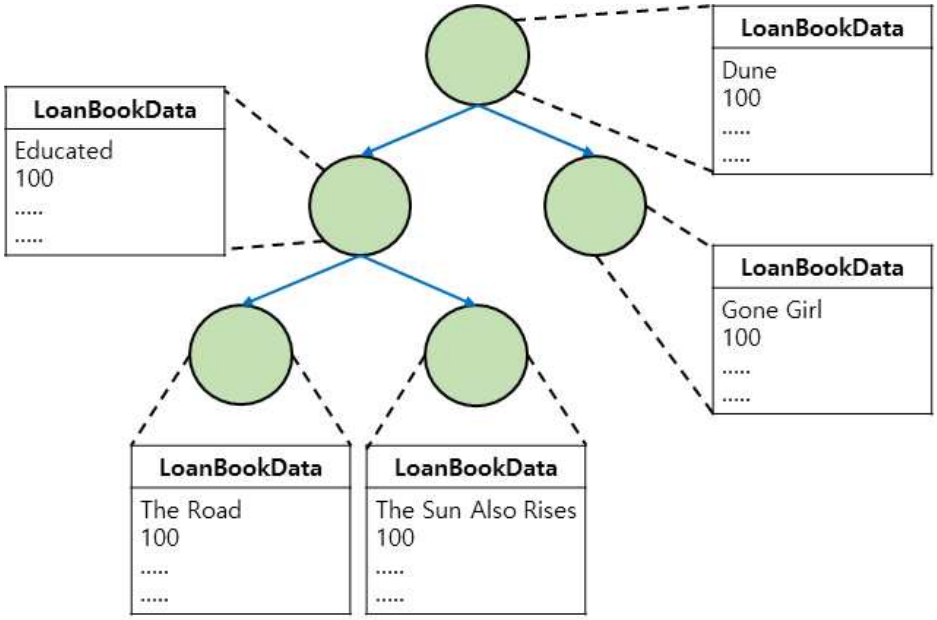
- Selection Tree는 그림 4와 같이 도서명을 기준으로 Min Winner Tree를 구성하며, 대소문자는 구별하지 않는다.
- Selection Tree의 run의 개수는 도서 분류 코드 개수와 동일하다.
- ADD 명령어로 B+-tree에 저장된 데이터의 대출 권수를 업데이트 후에, 대출 불가 도서는 도서 분류 코드에 따라 Min Heap으로 구현된 Selection Tree의 각 runs에 저장한다. 이때, 대출 불가가 된 도서는 B+-tree에서 삭제된다.
- Selection Tree의 내부 node에 저장되는 데이터는 LoanBookData class로 선언되어 있으며, 멤버 변수로는 도서명, 도서 분류 코드, 저자, 발행 연도, 대출 권수를 갖고 있다.
- Selection Tree의 각 run은 LoanBookHeap class로 선언되며, Heap이 정렬되는 경우 Selection tree로 같이 재정렬된다. LoanBookHeap에 대해서는 다음 절에서 자세히 설명한다.



[그림 4] Selection Tree의 예시

4) LoanBookHeap

- LoanBookHeap은 Min Heap으로 그림 5와 같이 도서명을 기준으로 정렬된다.
- ADD 명령어를 통해 B+-tree에 저장된 데이터 중 대출 불가 도서를 받아와 Heap을 구축하며, 기존에 Heap이 존재하지 않는 경우 새로 구축한다.
  - Heap에 새로운 데이터가 추가할 때는 왼쪽 자식 노드부터 생성한다.
- ADD 명령어를 통해 대출 불가 도서가 나타날 때마다 Heap을 재정렬한다.
- Heap을 재정렬할 때, 모든 부모 노드의 도서명이 자식 노드의 도서명보다 작거나 같은 경우에 정렬을 완료한다.
  - Heap이 재정렬되는 경우 Selection Tree도 재정렬한다.
- LoanBookHeap에 저장되는 데이터는 LoanBookData class로 선언되어 있으며, 멤버 변수로는 도서명, 도서 분류 코드, 저자, 발행 연도, 대출 권수를 갖고 있다.



[그림 5] Min Heap의 예시

□ Functional Requirements

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

표 3. 명령어 사용 예 및 기능 설명

명령어	명령어 사용 예 및 기능
LOAD	사용 예) LOAD 텍스트 파일의 데이터 정보를 불러오는 명령어로, 텍스트 파일에 데이터가 존재하는 경우 텍스트 파일을 읽어 B+-tree에 데이터를 저장한다. 만약 텍스트 파일이 존재하지 않거나 자료구조에 이미 데이터가 들어가 있으면 알맞은 에러 코드를 출력한다. 사용되는 텍스트 파일의 이름은 아래와 같으며 파일 이름을 수정하지 않는다. 텍스트 파일: loan_book.txt ***데이터 조건** - 도서명은 50자 미만으로 소문자로 주어지며, 그 이외의 입력은 없다고 가정한다. - “\t”를 구분자로 하여 정보가 저장되어 있다. 출력 포맷 예시) ========LOAD======== Success

	======================== ========ERROR======== 100 ========================
ADD	사용 예) ADD book 200 risa 1990 B+-tree에 도서를 직접 추가하기 위한 명령어로, 총 4개의 인자를 추가로 입력한다. 첫 번째 인자부터 도서의 이름, 분류 코드, 저자, 발행 연도를 나타내며 하나라도 존재하지 않을 시 에러 코드를 출력한다. 주어진 분류 코드 이외의 코드를 입력하는 경우는 없다. 출력 포맷 예시) ========ADD======== book/200/risa/1990 ==================== ========ERROR======== 200 ========================
SEARCH_BP	사용 예1) SEARCH_BP book 사용 예2) SEARCH_BP a c SEARCH_BP 명령어는 B+-tree에 저장된 데이터를 검색하는 명령어로 1개 또는 2개의 인자를 추가로 입력한다. SEARCH_BP의 인자로 1개가 입력되는 경우는 해당 도서의 검색 결과를 출력한다. 인자가 (시작 단어, 끝 단어)로 2개가 입력되는 경우 범위 검색의 결과를 출력한다. 이때 대소문자는 구별하지 않는다. 인자가 1개 또는 2개를 입력하지 않은 경우, B+-tree에 데이터가 없는 경우, 해당하는 도서명에 대한 도서가 없는 경우 에러 코드를 출력한다. 출력 포맷 예시) 1) SEARCH_BP book ========SEARCH_BP======== book/200/risa/1990/1 ==========================

	2) SEARCH_BP a e ========SEARCH_BP======== asia/300/john/1990/1 book/300/risa/2000/2 educated/300/chase/2005/1 ========== ========ERROR======== 300 ==========
PRINT_BP	사용 예) PRINT_BP B+-tree에 저장된 데이터를 도서명을 기준으로 오름차순으로 전부 출력한다. B+-tree에 저장된 데이터가 없는 경우 에러 코드를 출력한다. 출력 포맷 예시) ========PRINT_BP======== asia/300/john/1990/1 book/300/risa/2000/2 educated/300/chase/2005/1 ========== ========ERROR======== 400 ==========
PRINT_ST	사용 예) PRINT_ST 100 PRINT_ST 명령어는 Selection tree에서 인자로 받은 도서 분류 코드에 해당하는 데이터를 출력하는 명령어로 1개의 인자를 입력한다. 인자는 도서 분류 코드이다. 출력은 도서명을 기준으로 오름차순으로 출력한다. Selection Tree에 저장된 데이터가 없는 경우, 도서 분류 코드에 대응되는 Heap 자료구조가 없는 경우, 잘못된 도서 분류 코드를 입력하는 경우에 에러 코드를 출력한다. 출력 포맷 예시) ========PRINT_ST======== asia/100/john/1990/3 book/100/risa/2000/3 educated/100/chase/2005/3 ==========

	========ERROR======== 500 ======================
DELETE	사용 예) DELETE DELETE 명령어는 Selection Tree에서 root node에 저장된 도서를 제거하는 명령어이다. 이때 도서가 제거되어 Heap에 저장된 도서명의 대소관계가 변경되면 Heap을 재정렬한다. 자료구조에 데이터가 존재하지 않을 시 에러 코드를 출력한다. 출력 포맷 예시) ========DELETE======== Success ====================== ========ERROR======== 600 ======================
EXIT	사용 예) EXIT 프로그램상의 메모리를 해제하며, 프로그램을 종료한다. 출력 포맷 예시) =======EXIT======== Success ==================

□ 동작 별 에러 코드

동작	에러 코드
LOAD	100
ADD	200
SEARCH_BP	300
PRINT_BP	400
PRINT_ST	500
DELETE	600
잘못된 명령어	700

□ Requirements in implementation

- ✓ 모든 명령어는 command.txt에 저장하여 순차적으로 읽고 처리한다.
- ✓ 모든 명령어는 반드시 대문자로 입력한다.
- ✓ 명령어에 인자(Parameter)가 모자라거나 필요 이상으로 입력받으면 에러 코드를 출력한다.
- ✓ 예외처리에 대해 반드시 에러 코드를 출력한다.
- ✓ 출력은 "출력 포맷"을 반드시 따라 한다.
- ✓ log.txt 파일에 출력 결과를 반드시 저장한다.
  - log.txt가 이미 존재할 경우 텍스트 파일 가장 뒤에 이어서 추가로 저장한다.

□ 구현 시 반드시 정의해야하는 Class

- ✓ BpTreeNode - B+-tree의 노드 클래스
- ✓ BpTreeIndexNode - B+-tree의 인덱스 노드 클래스
- ✓ BpTreeDataNode - B+-tree의 데이터 노드 클래스
- ✓ BpTree - B+-tree 클래스
- ✓ SelectionTreeNode - Selection tree의 노드 클래스
- ✓ SelectionTree - Selection tree 클래스
- ✓ LoanBookHeapNode - 도서 min heap 노드 클래스
- ✓ LoanBookHeap - 도서 min heap 클래스
- ✓ Manager - Manager 클래스
  - 다른 클래스들의 동작을 관리하여 프로그램을 전체적으로 조정하는 역할을 수행

□ Files

- ✓ loan_book.txt : 프로그램에 추가할 대출 도서 데이터가 저장된 파일
- ✓ command.txt : 프로그램을 동작시키는 명령어들을 저장하고 있는 파일
- ✓ log.txt : 프로그램 출력 결과를 모두 저장하고 있는 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, 적발시 전체 프로젝트 0점 처리됨을 명시한다.**

□ 채점 기준

✓ 코드

채점 기준	점수
LOAD 명령어가 정상 동작하는가?	1
ADD 명령어가 정상 동작하는가?	2
SEARCH_BP 명령어가 정상 동작하는가?	3
PRINT_BP 명령어가 정상 동작하는가?	3
PRINT_ST 명령어가 정상 동작하는가?	3
DELETE 명령어가 정상 동작하는가?	3
총합	**15**

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

✓ 보고서

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

✓ 최종 점수는 (코드 점수 x 보고서 점수)로 계산됩니다.

□ 제출기한 및 제출방법

✓ 제출기한

- 2023년 11월 16일 목요일 23:59:59까지 클라스(KLAS)에 제출

✓ 제출방법

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

✓ 제출 형식

- 학번_DS_project2.tar.gz (ex. 2020XXXXXX_DS_project2.tar.gz)

✓ 보고서 작성 형식 및 제출 방법

- Introduction : 프로젝트 내용에 대한 설명
- Flowchart : 설계한 프로젝트의 플로우차트를 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