Data Structure Lab. Project #1

2021년 9월 13일

Due date: 2021년 10월 11일 월요일 23:59:58 까지

본 프로젝트에서는 이진 탐색 트리(Binary Search Tree, BST)와 연결 리스트(Linked List), 큐(Queue), 힙(Heap)을 이용하여 계정 관리 프로그램을 구현한다. 이 프로그램은 파일로부터 사용자 이름과 나이, 계정 ID를 읽어 Queue를 구축하며, 해당 Queue를 Account_Queue라 부른다. pop 명령을 실행하면 Queue에서 데이터를 방출하여 Account_BST와 User_List에 저장한다.

Account_BST는 계정 ID와 사용자 이름으로 노드를 구성하며, 계정 ID를 기준으로 BST를 연결한다. User_List는 사용자 이름과 나이, 사용자가 보유한 계정 수로 노드를 구성하며, 노드가 입력된 순서대로 List를 연결한다. User_List에 존재하지 않는 사용자의 계정 정보가 입력될 경우 List에 노드를 추가하고, 이미 존재하는 사용자의 계정 정보가 입력될 경우 해당 노드를 수정한다. User_List의 노드는 Account_BST 노드를 가리키는 포인터를 추가로 가지며, 해당 User_List 노드의 사용자가 보유한 계정들을 Linked List로 연결한다. HeapLoad 명령을 실행하면 User_List의 정보들을 순서대로 읽어 연령대별 노드를 생성하고 User_Heap에 저장한다.

User_Heap은 연령대와 연령대별 사용자 수로 노드를 구성하며, 연령대별 사용자 수를 기준으로 정렬된다. User_Heap에 존재하지 않는 연령대의 사용자 정보가 입력될 경우 Heap에 노드를 추가하고, 이미 존재하는 연령대의 사용자 정보가 입력될 경우 해당 노드의 사용자 수를 증가시킨다. 자료구조의 구축 방법과 조건에 대한 자세한 설명은 program implementation에서 설명한다.

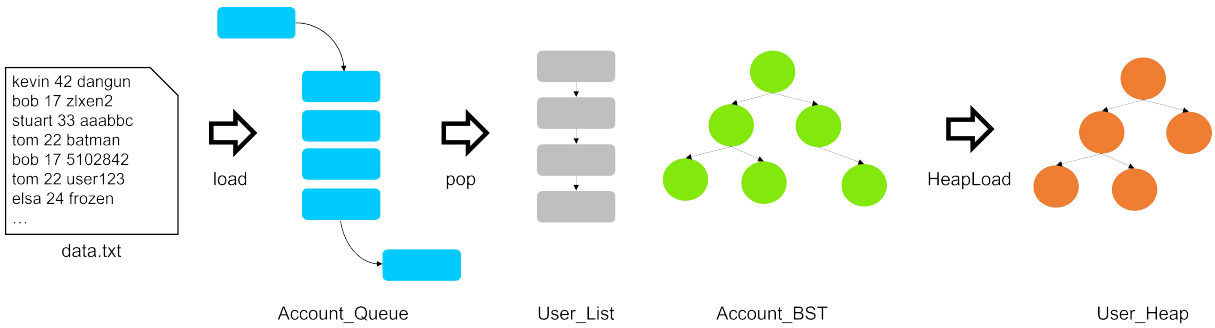

그림 1. 계정 관리 프로그램

□ Program implementation

1) Account_Queue

- data.txt에 저장된 데이터를 이용하여 구축한다. Queue의 자료구조에 따라 저장되는 순서대로 방출된다. Queue에는 data.txt의 첫 번째 줄부터 마지막 줄까지 순서대로 저장된다.
- Queue에 저장되는 데이터는 AccountNode class로 선언되어 있으며 멤버 변수로는 사용자 이름과 나이, 계정 ID를 갖는다.
- Queue에서 pop 연산을 수행하여 방출되는 데이터는 BST와 List의 입력으로 사용된다.
- 동명이인은 없다고 가정하며, 동일 사용자에 다른 나이가 입력되는 경우는 없다고 가정한다.
- 사용자 이름은 대소문자, 계정 ID는 대소문자와 숫자로 구성되며, 대소문자는 구분하지 않는다. 이외의 입력은 없다고 가정한다.
- 나이의 범위는 10~69세로 제한한다.

그림 2. Account_Queue의 예

2) Account_BST

- Queue에서 방출된 데이터를 삽입하며, BST의 노드는 사용자 이름과 계정 ID로 구성된다.
- BST에서 지원하는 연산은 삽입, 삭제, 검색, 출력(중위 순회, 후위 순회, 전위 순회, 레벨 순회)이다. 각 연산에 대해서는 Functional Requirements에서 자세히 설명한다.
- 계정 ID는 고유한 문자열로 이루어져야 하며, 기존에 존재하는 계정 ID와 동일한 계정 ID가 입력으로 들어오면 입력되지 않고, 에러를 출력한다.
- 한 명의 사용자 당 최대 3개의 계정 ID만 보유할 수 있으며, 4번째 계정 ID가 입력으로 들어오면 입력되지 않고, 에러를 출력한다.
- BST 연결 규칙은 다음과 같다.

※ BST 연결 규칙

① 부모 노드보다 계정 ID의 사전적 순서가 작은 노드는 왼쪽, 큰 노드는 오른쪽 서브 트리에 반드시 위치한다(숫자는 알파벳보다 왼쪽에 존재한다).

② 노드를 제거할 때, **양쪽 자식 노드가 모두 존재할 경우에는 오른쪽 자식 노드 중 가장 작은 노드를 제거되는 노드 위치로 이동한**다.

3) User_List

- Queue에서 방출된 데이터를 삽입하며, List의 노드는 사용자 이름과 나이, 보유한 계정 수로 구성된다.
- Queue에서 방출된 데이터가 BST에 추가되지 않으면 List에도 입력하지 않는다.
- 입력된 데이터의 사용자 이름을 확인하여 List에 없는 경우 List에 노드를 추가하고, List에 존재하는 경우 해당 노드의 보유한 계정 수를 증가시키며, List는 입력된 순서로 정렬된다.
- User_List의 노드는 BST 노드 포인터를 가지며 Account_BST에서 해당 노드의 사용자가 보유한 계정 노드들을 Linked List로 연결한다.
- Account_BST에서 삭제 연산이 수행된 경우 User_List에서 해당하는 노드의 보유한 계정 수를 감소시키며, 보유한 계정 수가 0이 되는 노드는 삭제한다.

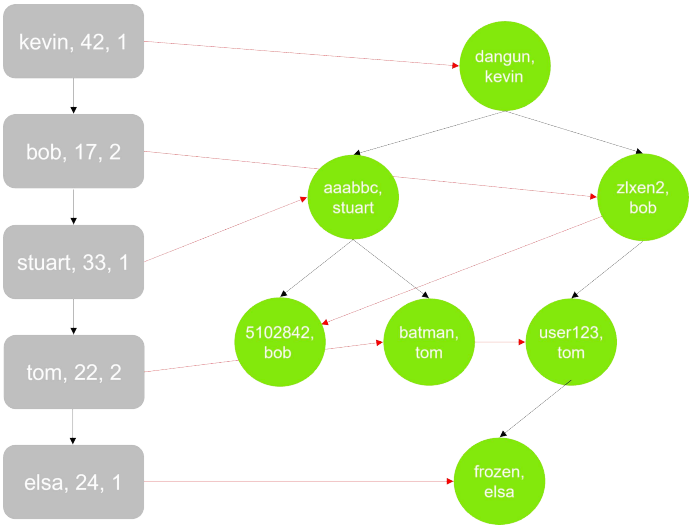

그림 3. User_List(좌)와 Account_BST(우)의 예

4) User_Heap

- User_Heap은 그림 4와 같이 연령대별 사용자의 수를 기준으로 정렬된다.
- HeapLoad 연산을 통해 User_List의 정보를 받아와 Heap을 구축하며, 기존에 Heap이 존재할 경우 제거하고 새로 구축한다.
- User_List의 노드를 1개씩 읽어 Heap에 입력하고, 매번 Heap을 재정렬한다.
- 입력된 데이터의 사용자 나이를 확인하여 Heap에 해당 연령대의 노드가 존재하지 않는 경우 Heap에 노드를 추가하고, 노드가 존재할 경우 해당 노드의 사용자 수를 증가시킨다.
- Heap을 재정렬할 때, 비교하는 노드의 사용자 수가 추가된 노드보다 크거나 같은 경우에 정렬을 완료한다.
- Heap에서는 삽입과 출력 연산(레벨 순회)만 이뤄진다.
- MaxHeap에 저장되는 데이터는 UserHeapNode class로 선언되어 있으며 멤버 변수로는 연령대에 속한 사용자 수와 연령대를 갖고 있다.
- 연령대는 10세부터 10단위로 구분하며, 연령대 이름은 10, 20, …, 60으로 지정한다.

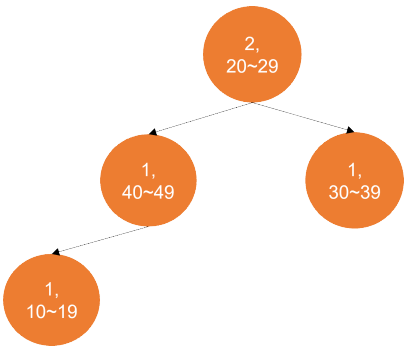

그림 4. User_Heap의 예

□ Functional Requirements

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

표 1. 명령어 사용 예 및 기능 설명

명령어	명령어 사용 예 및 기능
QLOAD	사용 예) QLOAD 텍스트 파일의 데이터 정보를 불러오는 명령어로, 텍스트 파일에 데이터 정보가 존재할 경우 텍스트 파일을 읽어 Queue 자료구조에 모두 저장하고, 그 정보들을 출력한다. 만약 텍스트 파일이 존재하지 않거나 자료구조에 이미 데이터가 들어가 있으면 에러 코드를 출력한다. 사용되는 텍스트 파일의 이름은 아래와 같으며 파일 이름을 수정하지 않는다. 텍스트 파일: data.txt ***데이터 조건** - 계정 ID는 대소문자와 숫자로 구성되며, 대소문자를 구별하지 않는다. - 사용자 이름은 대소문자로 구성되며, 동명이인은 없다고 가정한다. - 나이는 자연수 형태로 표현되며, 10~69세 범위로 한정한다. - 각 데이터의 인자들은 모두 스페이스로 구분된다. 출력 포맷 예시) =======QLOAD======== kevin/42/dangun bob/17/zlxen2 … ========== ======== ERROR ======== 100 ==========
ADD	사용 예) ADD tom 22 user222 Queue에 데이터를 직접 추가하기 위한 명령어로, 총 3개의 인자를 추가로 입력한다. 첫 번째 인자부터 사용자의 이름, 나이, 계정 ID를 나타내며 하나라도 존재하지 않을 시 에러 코드를 출력한다. 출력 포맷 예시) ======== ADD ======== tom/22/user222 ========== ======== ERROR ======== 200 ==========

QPOP	사용 예) QPOP 10 Queue의 데이터를 BST와 List로 옮기는 명령어이다. Queue의 front부터 첫 번째 인자의 숫자만큼 데이터가 방출되며 Queue에 저장된 데이터보다 높은 숫자가 입력될 경우 에러코드를 출력하며, 명령을 실행하지 않는다. 출력 포맷 예시) ======== QPOP======== Success ===================== ======== ERROR ======== 300 ======================
SEARCH	사용 예) SEARCH user tom 사용 예) SEARCH id frozen BST와 List에 저장된 정보를 찾아 출력하는 명령어로, 첫 번째 인자로 “user”나 “id”를 입력으로 받고, 두 번째 인자로 검색할 정보를 입력으로 받는다. 첫 번째 인자로 “user”를 사용할 경우 찾고자 하는 사용자의 정보와 해당 사용자가 보유한 모든 계정을 출력하고, “id”를 사용할 경우 찾고자 하는 계정을 보유한 사용자 이름을 출력한다. 인자가 부족하거나 잘못된 경우, 찾고자 하는 정보가 BST나 List에 존재하지 않는 경우 모두 에러 코드를 출력한다. 출력 포맷 예시) ======== SEARCH ======== User tom/22 batman user123 ======================= ======== SEARCH ======== ID frozen/elsa ======================= ======== ERROR ======== 400 ======================

PRINT	사용 예) PRINT L 사용 예) PRINT B PRE 사용 예) PRINT H 자료구조에 저장된 데이터들을 출력하는 명령어로, 첫 번째 인자로 L을 입력할 경우, User_List에 저장된 사용자들을 출력한다. 출력형태는 (사용자 이름)/(사용자 나이)/(보유한 계정 수)이다. User_List에 데이터가 존재하지 않는 경우 에러 코드를 출력한다. B를 입력할 경우, Account_BST에 있는 계정 정보들을 전부 출력한다. 탐색 방법은 총 4가지로 전위 순회(pre-order), 중위 순회(in order), 후위 순회(post-order), 레벨 순회(level-order)를 지원한다. 두 번째 인자로 순회 방법을 선택한다. “PRE”, “IN”, “POST”, “LEVEL”을 입력받아 아래의 의미를 참조하여 각각의 트리순회(Tree Traversal) 방법에 따른 알맞은 순서대로 데이터들을 전부 출력한다. 이때, level-order 기능은 위에서부터 level 순으로, 왼쪽에서부터 오른쪽으로 순서대로 출력한다. 출력형태는 (계정 ID)/(사용자 이름)이다. H를 입력할 경우, User_Heap에 저장된 지역들을 level-order 순서대로 출력한다. 출력형태는 (연령대별 사용자 수)/(연령대)이다. User_Heap에 데이터가 존재하지 않는 경우 에러 코드를 출력한다. 출력 포맷 예시) 1) PRINT L인 경우 ======== PRINT ======== LIST kevin/42/1 bob/17/2 … ====================== 2) PRINT B인 경우 ======== PRINT ======== BST LEVEL dangun/kevin aaabbc/stuart … ====================== 3) PRINT H인 경우 ======== PRINT ======== Heap 2/20 1/40 …

	========== ======== ERROR ======== 500 ==========
DELETE	사용 예) DELETE batman BST와 List에 저장된 계정 정보를 제거하는 명령어로 계정 ID를 인자로 입력받는다. 인자로 받은 계정의 BST 노드를 Account_BST에서 제거하고, 해당 계정을 보유한 User_List의 사용자 노드는 보유한 계정 수를 감소시키며, 보유한 계정 수가 0이 되는 User_List의 노드는 제거한다. 출력 포맷 예시) ======== DELETE ======== Success ========== ======== ERROR ======== 600 ==========
HLOAD	사용 예) HLOAD User_List의 정보를 이용하여 User_Heap을 구축하는 명령어이다. 기존에 User_Heap이 존재할 경우 제거하고 새로운 User_Heap을 생성한다. User_List에서 노드를 1개씩 읽어 Heap에 입력하며, 사용자의 나이를 연령대별로 분류하여 Heap Node를 구성한다. Heap에 사용자의 연령대에 해당하는 노드가 없으면 노드를 새로 생성하고, 노드가 존재하면 해당 노드의 사용자 수를 증가시키며 Heap을 재정렬한다. MaxHeap을 정렬할 때 부모 노드의 크기가 자식 노드의 크기보다 작은 경우에만 노드를 교환하며, 노드를 교환할 때에는 노드 자체를 교환한다. List가 비어있는 경우 에러 코드를 출력한다. 출력 포맷 예시) ====== HLOAD ====== Success ========== ======== ERROR ======== 700 ==========

EXIT	사용 예) EXIT 프로그램 상의 메모리를 해제하며, 프로그램을 종료한다. 출력 포맷 예시) =======EXIT======== Success ==================

□ Requirements in implementation

- ✓ 모든 명령어는 command.txt에 저장하여 순차적으로 읽고 처리한다.
- ✓ 모든 명령어는 반드시 대문자로 입력한다.
- ✓ 명령어에 인자(Parameter)가 모자라거나 필요 이상으로 입력받을 경우 에러 코드를 출력한다.
- ✓ 동명이인은 존재하지 않는다.
- ✓ 예외처리에 대해 반드시 에러 코드를 출력한다.
- ✓ 출력은 “출력 포맷”을 반드시 따라한다.
- ✓ log.txt 파일에 출력 결과를 반드시 저장한다.
  - log.txt가 이미 존재할 경우 텍스트 파일 가장 뒤에 이어서 추가로 저장한다.
- ✓ 읽어야할 텍스트 파일이 존재하지 않을 경우 해당 텍스트 파일을 생성한 뒤 진행하도록 한다.

□ 명령어별 에러 코드

명령어	에러 코드
QLOAD	100
ADD	200
QPOP	300
SEARCH	400
PRINT	500
DELETE	600
HLOAD	700
잘못된 명령어	800

□ 동작 예시

data.txt

```
kevin 42 dangun
bob 17 zlxen2
stuart 33 aaabbc
tom 22 batman
bob 17 5102842
tom 22 user123
elsa 24 frozen
```

command.txt

```
LOAD
QLOAD
QLOAD
QPOP 4
QPOP 5
QPOP 3
PRINT L
PRINT B PRE
SEARCH id batman
SEARCH bob
SEARCH user bob
PRINT H
HLOAD
PRINT H
DELETE stuart
DELETE aaabbc
PRINT L
HLOAD
PRINT H
EXIT
```

실행 결과 화면 및 log.txt

```
=======ERROR========
800
====================
=======QLOAD========
kevin/42/dangun
bob/17/zlxen2
stuart/33/aaabbc
tom/22/batman
bob/17/5102842
tom/22/user123
elsa/24/frozen
====================
```

```
=======ERROR========
100
====================

=======QPOP========
Success
====================

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

=======QPOP========
Success
====================

======== PRINT =======
LIST
kevin/42/1
bob/17/2
stuart/33/1
tom/22/2
elsa/24/1
======================

======== PRINT =======
BST PRE
dangun/kevin
aaabbc/stuart
5102842/bob
batman/tom
zlxen2/bob
user123/tom
frozen/elsa
======================

======== SEARCH ========
ID
batman/tom
========================

=======ERROR========
400
========================
```

```
======== SEARCH ========
User
bob/17
zlxen2
5102842
=======================

=======ERROR========
500
=======================

======== HLOAD ========
Success
=======================

======== PRINT ========
Heap
2/20
1/40
1/30
1/10
=======================

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

======== DELETE ========
Success
=======================

======== PRINT =======
LIST
kevin/42/1
bob/17/2
tom/22/2
elsa/24/1
=======================

======== HLOAD ========
Success
=======================

======== PRINT ========
Heap
2/20
1/10
```

```
1/40
======================

=======EXIT========
Success
===================
```

□ 채점 기준

채점 기준	점수
AccountQueue가 정상적으로 동작하는지?	2
AccountBST가 정상적으로 동작하는지?	3
UserList가 정상적으로 동작하는지?	2
UserHeap이 잘 동작하는지?	2
SEARCH user가 잘 동작하는지? (UserListNode와 AccoutBSTNode를 연결하는 linked list 관련)	1
총합	10

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

□ 구현 시 반드시 정의해야하는 Class

- ✓ AccountQueue - 계정 Queue 클래스
- ✓ AccountQueueNode - 계정 Queue 노드 클래스
- ✓ AccountBST - 계정 BST 클래스
- ✓ AccountBSTNode - 계정 BST 노드 클래스
- ✓ UserList - 사용자 List 클래스
- ✓ UserListNode - 사용자 List 노드 클래스
- ✓ UserHeap - 사용자 max heap 클래스
- ✓ UserHeapNode - 사용자 max heap 노드 클래스
- ✓ Manager - Manager 클래스
  - 다른 클래스들의 동작을 관리하여 프로그램을 전체적으로 조정하는 역할을 수행

□ Files

- ✓ data.txt : 프로그램에 추가할 환자 정보들이 저장되어 있는 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 절대 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 동작해야한다. (컴파일 에러 발생 시 감점)
- 제공되는 Makefile을 사용하여 테스트 하도록 한다.

□ 제출기한 및 제출방법

✓ 제출기한
- 2021년 10월 11일 월요일 23:59:58 까지 제출

✓ 제출방법
- 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 pdf가 아닐 시 감점
- KLAS -> 과제 제출 -> **tar.gz로 과제 제출**

✓ 제출 형식
- 학번_DS_project1.tar.gz (ex. 2020202001_DS_project1.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