

# Data Structure Lab. Project #1

2020년 9월 14일

Due date: 2020년 10월 9일 금요일 23:59:59 까지

본 프로젝트에서는 이진 탐색 트리(Binary Search Tree, BST)와 큐(Queue), 힙(Heap)을 이용하여 질병 관리 프로그램을 구현한다. 이 프로그램은 질병 의심 환자 데이터 이름, 체온, 기침 여부, 지역을 받아 Queue에 구축하며, 해당 Queue를 Patient\_Queue라 부른다. Queue에서 pop 명령어를 실행하면 BST 구조에 저장된다. BST는 Location\_BST, Patient\_BST 두 종류로 구성된다. Location\_BST는 지역 정보를 담고 있는 BST이고, Patient\_BST는 각 Location\_BST의 노드마다 하나씩 연결된 BST 이다. Patient\_Queue에서 pop된 환자 데이터는 환자의 지역에 해당하는 노드에 속한 Patient\_BST에 삽입되며, 이때, 질병 여부를 검증하여 양성 혹은 음성 여부를 '+', '-'로 표기하여 저장한다. BST에서 pop 명령어를 실행하면 Location\_MaxHeap 이라는 Heap 구조에 저장되며, 각 노드는 지역과 지역에 해당하는 환자 수를 기준으로 정렬된다. 자료구조의 구축 방법과 조건에 대한 자세한 설명은 **program implementation**에서 설명한다.

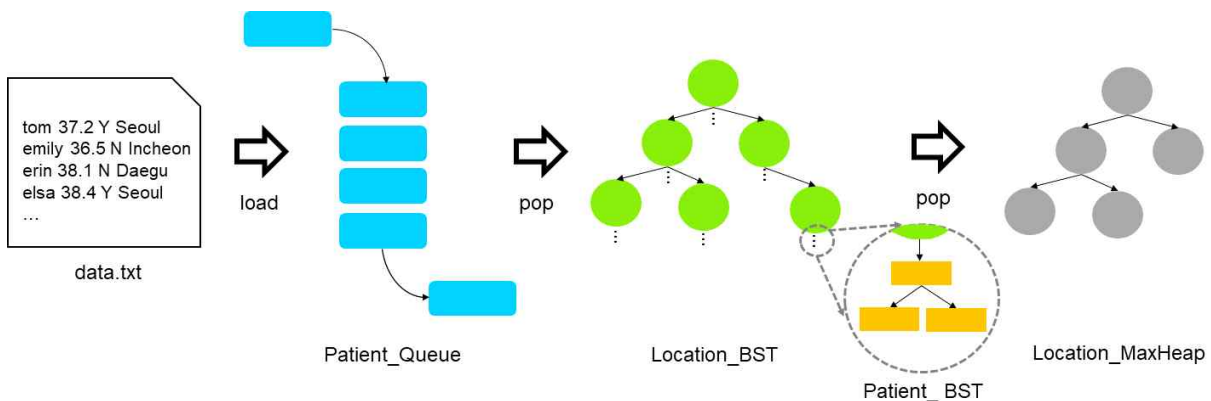


그림 1. 질병 관리 프로그램 구조

## □ Program implementation

### 1) Patient\_Queue

- 주어진 data.txt에 저장된 데이터를 이용하여 구축한다. Queue의 자료구조에 따라 저장되는 순서대로 방출된다. Queue에는 data.txt에 첫 번째 줄부터 마지막 줄까지 순서대로 저장된다.
- 프로그램 구현 시 Queue는 <queue> 라이브러리에서 제공하는 STL을 이용한다.
- Queue에 저장되는 데이터는 PatientNode class로 선언되어 있으며 멤버 변수로는 환자이름, 체온, 기침 여부, 지역을 갖는다.
- 이름 정보는 항상 고유하며, 중복으로 입력되는 경우는 없다고 가정한다.



그림 2. Patient\_Queue의 예

### 2) Location\_BST

- Location\_BST는 초기에 한 번 구축되며, 그 후로 노드 추가 혹은 삭제가 일어나지 않는다.
- Location\_BST의 구조는 그림 3과 동일하게 구성되어야 한다. 즉 반드시 Gwangju, Daegu, Seoul, Busan, Daejeon, Incheon, Ulsan 순으로 추가한다.
- 지역 이름은 항상 첫 글자만 대문자로 표기하며 그렇지 않은 경우의 입력은 없다고 가정한다.
- Patient\_Queue에서 방출된 데이터의 지역 정보를 확인하여 해당 지역의 노드에 속한 Patient\_BST에 삽입한다.
  - 예를 들어, {tom, 37.2, Y, Seoul}와 같이 구성된 환자 데이터는 Location\_BST의 Seoul 노드에 속한 Patient\_BST에 삽입한다.
- Location\_BST에 저장되는 데이터는 LocationNode class로 선언되어 있으며 멤버 변수로는 지역, Patient\_BST 포인터를 갖는다.

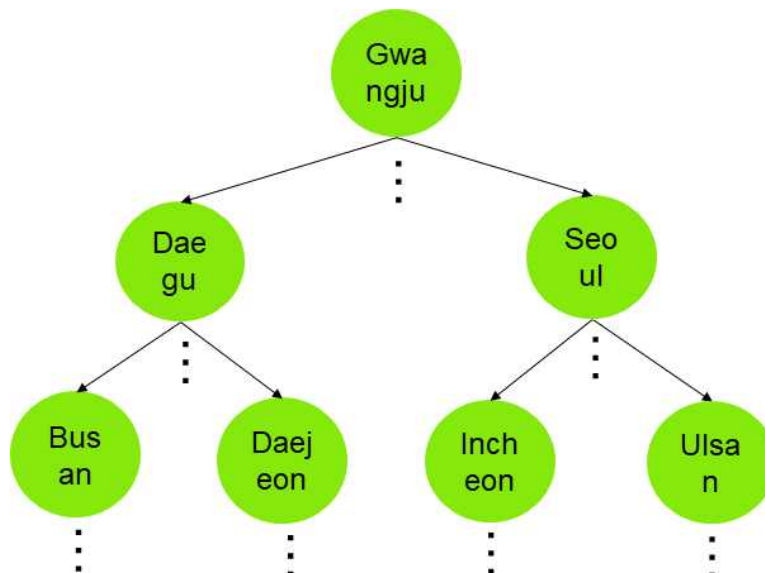


그림 3. Location\_BST 구조

### 3) Patient\_BST

- Patient\_BST는 그림 4와 같이 Locaton\_BST의 각 노드에 속한 BST이다.
- Location\_BST에는 Patient\_BST의 root 노드의 주소값만 저장하고 있으며, 초기값은 null이다.

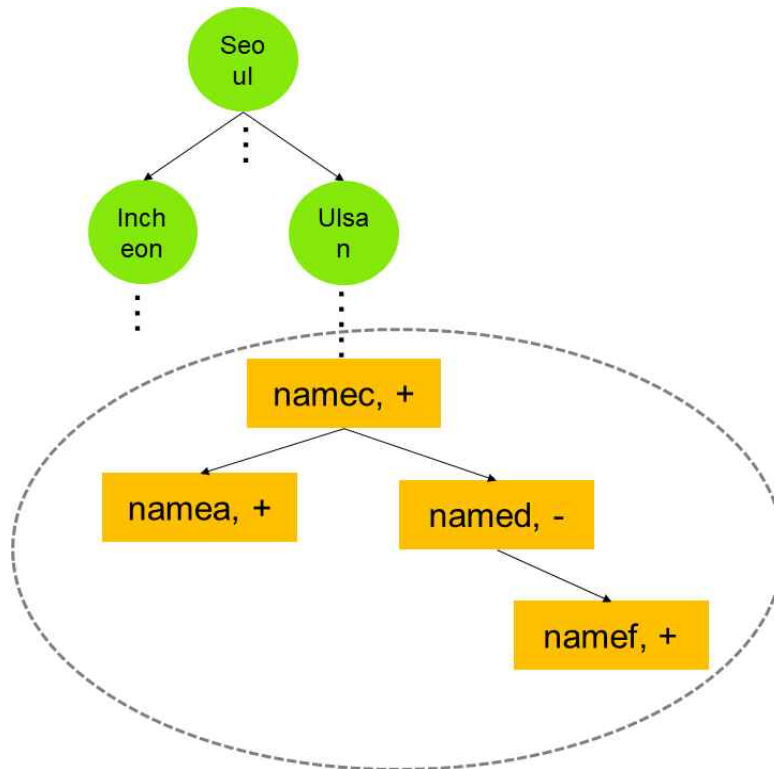


그림 4. Patient\_BST의 예

- Patient\_Queue에서 방출된 데이터를 삽입하며, 체온과 기침 여부 데이터를 통하여 양성, 음성을 판단한다.
  - \* 체온은 37.5도 이상이며 기침 여부가 Y일 경우 양성으로 판단하여 '+'를 표기한다. 그 이외의 경우는 모두 음성으로 판단하여 '-'를 표기한다.
- 저장되는 데이터는 이름, 양성/음성 정보이다. (온도, 기침 여부 등 데이터 표기 제외)
- Patient\_BST에서 지원하는 연산은 삽입, 삭제, 탐색(중위 순회, 후위 순회, 전위 순회, 레벨 순회)이다. 각 연산에 대해서는 **Functional Requirements**에서 자세히 설명한다.
- Patient\_BST에 저장되는 데이터는 PatientBSTNode class에 선언되며, 멤버 변수로 환자이름, 양성 음성 여부를 갖고 있다.
- Patient\_BST의 자료구조 정보는 PatientBST class에 선언되며, 멤버 변수로 root 정보를 가지고 있으며 멤버 함수로 삽입, 삭제, 탐색 등 요구되는 연산을 위한 함수를 갖고 있다.
- Patient\_BST에서 사용되는 BST를 연결하는 규칙은 다음과 같다.

#### ※ BST 연결 규칙

- ① 부모 노드보다 환자 이름의 사전적 순서가 작은 노드는 왼쪽, 큰 노드는 오른쪽 서브 트리에 반드시 위치한다.
- ② 노드를 제거할 때, 양쪽 자식 노드가 모두 존재할 경우에는 왼쪽 자식 노드 중 가장 큰 노드를 제거되는 노드 위치로 이동시킨다.

#### 4) Location\_MaxHeap

- Location\_MaxHeap의 Heap는 그림 5와 같이 지역에 해당하는 환자 수를 기준으로 정렬된다.
- BST에서 방출된 데이터를 삽입하며, max를 결정하는 기준은 지역에 속한 환자의 수이다. 다시 말해, BST에서 가장 많이 방출된 지역이 root에 오게 된다.
- BST에서 방출 연산이 일어날 때마다 Heap을 재정렬한다.
- Heap을 재정렬할 때, 비교하는 노드의 환자 수가 추가된 노드보다 크거나 같은 경우에 정렬을 완료한다.
- Location\_MaxHeap에서는 삽입 연산만 이뤄진다.
- Location\_MaxHeap에 저장되는 데이터는 LocationHeapNode class로 선언되어 있으며 멤버 변수로는 지역에 속한 환자 수, 지역 이름을 갖고 있다.

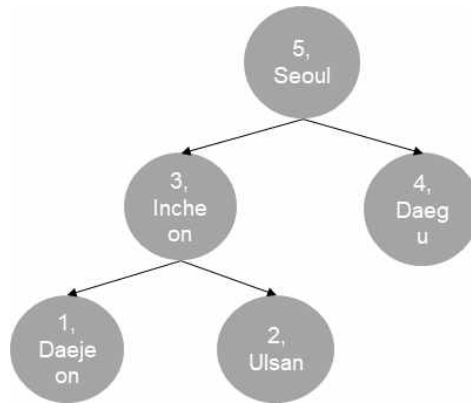


그림 5. Location\_MaxHeap의 예

## □ 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
LOAD	<p>사용 예) LOAD</p> <p>텍스트 파일의 데이터 정보를 불러오는 명령어로, 텍스트 파일에 데이터 정보가 존재할 경우 텍스트 파일을 읽어 Patient_Queue 자료구조에 모두 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.</p> <p>텍스트 파일: data.txt</p>

	<p><b>*데이터 조건</b></p> <ul style="list-style-type: none"> <li>- 환자 이름은 무조건 소문자로 주어지며 그 이외의 입력은 없다고 가정한다.</li> <li>- 체온은 실수형태로 표현된다.</li> <li>- 기침 여부는 대문자 Y 혹은 N으로만 입력되며 그 이외는 없다고 가정한다.</li> <li>- 지역은 첫글자만 대문자로 표기하며 이외 문자는 모두 소문자로 표시한다. 주어진 7개의 지역 외의 다른 지역의 입력은 없다고 가정한다.</li> <li>- 각 데이터의 인자들은 모두 스페이스로 구분된다.</li> </ul> <p>출력 포맷 예시)</p> <pre> ===== LOAD ===== Success =====  ===== ERROR ===== 100 ===== </pre>
ADD	<p>사용 예) ADD risa 38.1 Y Incheon</p> <p>Patient_Queue에 직접 추가하기 위한 명령어로, 총 4개의 인자를 추가로 입력한다. 첫 번째 인자부터 환자의 이름, 체온, 기침 여부, 지역을 나타내며 하나라도 존재하지 않을 시 에러 코드를 출력한다. 주어진 7개 지역 이외의 지역을 입력하는 경우는 없다.</p> <p>출력 포맷 예시)</p> <pre> ===== ADD ===== risa/38.1/Y/Incheon =====  ===== ERROR ===== 200 ===== </pre>
QPOP	<p>사용 예) QPOP 10</p> <p>Patient_Queue의 환자 데이터를 BST로 옮기는 명령어이다. Queue의 front 부터 첫 번째 인자의 숫자 만큼 데이터가 이동되며 Queue에 저장된 데이터보다 높은 숫자가 입력될 경우 에러코드를 출력하며, 명령을 실행하지 않는다.</p> <p>출력 포맷 예시)</p> <pre> ===== QPOP===== </pre>

	<p>Success</p> <p>=====</p> <p>===== ERROR =====</p> <p>300</p> <p>=====</p>
SEARCH	<p>사용 예) SEARCH tom</p> <p>BST에 저장된 정보를 찾아 출력하는 명령어로, 첫 번째 인자로 반드시 환자 이름을 입력받는다. 인자를 입력하지 않은 경우, 해당 이름이 BST에 존재하지 않는 경우 모두 에러 코드를 출력한다.</p> <p>출력 포맷 예시)</p> <p>===== SEARCH =====</p> <p>tom/+</p> <p>=====</p> <p>===== ERROR =====</p> <p>400</p> <p>=====</p>
PRINT	<p>사용 예) PRINT B PRE</p> <p>사용 예) PRINT H</p> <p>자료구조에 저장된 데이터들을 출력하는 명령어로, 첫 번째 인자로 B를 입력할 경우, Patient_BST에 있는 환자 정보들을 전부 출력한다. 탐색 방법은 총 4가지 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.</p> <p>H를 입력할 경우, Location_MaxHeap에 저장된 지역들을 level-order 순서대로 출력한다. 출력할 때는 (지역에 속한 환자 수)/(지역이름)의 형태로 출력한다. Location_MaxHeap에 데이터가 존재하지 않는 경우 에러 코드를 출력한다.</p> <p>출력 포맷 예시)</p> <p>1) PRINT B인 경우</p>

	<pre> ===== PRINT ===== BST tom/- erin/+ ... =====  2) PRINT H인 경우 ===== PRINT ===== Heap 5/Seoul 4/Daegu ... =====  ===== ERROR ===== 500 ===== </pre>
BPOP	<p>사용 예) BPOP tom</p> <p>BST에 저장된 환자 정보를 지우고, Location_MaxHeap 의 지역별 환자 수를 업데이트한다. Heap에 저장된 데이터의 부모 자식간 대소관계가 변경되었다면, Heap을 재정렬한다. 첫 번째 인자는 반드시 이름 정보이며, 인자가 입력되지 않은 경우 에러 코드를 출력하고, 해당 이름 정보가 BST에 없을 시 에러 코드를 출력한다.</p> <p>출력 포맷 예시)</p> <pre> ===== BPOP ===== Success =====  ===== ERROR ===== 600 ===== </pre>

EXIT	<p>사용 예) EXIT</p> <p>프로그램 상의 메모리를 해제하며, 프로그램을 종료한다.</p> <p>출력 포맷 예시)</p> <pre>=====EXIT===== Success =====</pre>

#### □ 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.
- ✓ 질병 관리 프로그램에는 중복된 환자 이름이 존재하지 않는다.
- ✓ 예외처리에 대해 반드시 에러 코드를 출력한다.
- ✓ 출력은 “출력 포맷”을 반드시 따른다.
- ✓ log.txt 파일에 출력 결과를 반드시 저장한다.
  - log.txt가 이미 존재할 경우 텍스트 파일 가장 뒤에 이어서 추가로 저장한다.
- ✓ 읽어야 할 텍스트 파일이 존재하지 않을 경우 해당 텍스트 파일을 생성한 뒤 진행하도록 한다.

#### □ 동작별 에러 코드

동작	에러 코드
LOAD	100
ADD	200
QPOP	300
SEARCH	400
PRINT	500
BPOP	600



## □ 동작 예시

data.txt
tom 37.2 Y Seoul emily 36.5 N Incheon erin 38.1 Y Daegu elsa 38.4 Y Seoul mia 36.4 Y Ulsan
command.txt
LOAD LOAD QPOP 4 QPOP 5 PRINT B PRE BPOP tom SEARCH erin SEARCH mia BPOP erin BPOP elsa PRINT H EXIT
실행 결과 화면 및 log.txt
<pre> =====LOAD===== Success =====  =====ERROR===== 100 =====  =====QPOP===== Success =====  =====ERROR===== 300 =====  ===== PRINT ===== BST erin/+ tom/- </pre>

```

elsa/+
emily/-
=====

===== BPOP =====
Success
=====

===== SEARCH =====
erin/+
=====

=====ERROR=====
400
=====

===== BPOP =====
Success
=====

===== BPOP =====
Success
=====

===== PRINT =====
Heap
2/Seoul
1/Daegu
=====

=====EXIT=====
Success
=====

```

## □ 구현 시 반드시 정의해야하는 Class

- ✓ PatientNode                      - 환자 정보 노드 클래스
- ✓ LocationNode                    - 지역 노드 클래스
- ✓ LocationBST                     - 지역 BST 클래스
- ✓ PatientBST                      - 환자 정보 BST 클래스
- ✓ PatientBSTNode                 - 환자 정보 BST 노드 클래스
- ✓ LocationHeap                    - 지역 max heap 클래스
- ✓ LocationHeapNode               - 지역 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
  - 2020년 10월 9일 금요일 23:59:59 까지 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 사이트는 추후 공지
- ✓ 제출 형식
  - 학번\_DS\_project1.tar.gz (ex. 2020202\_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
    - 보고서는 한글로 작성
    - 보고서에는 소스코드를 포함하지 않음