

# B-Tree\_Assignment

데이터베이스시스템및응용

김상욱 교수님

제출일 : 2020.09.27

컴퓨터소프트웨어학부

2019000982

이지원

## 1. 알고리즘 요약

### (1) class의 구조

BPlusTree, FileIO, Main, 3개의 class로 구성되어있다.

#### <BPlusTree>

BPlusTree 안에는 Node, LeafNode, NonLeafNode, Pair, KeyPair, NodePair의 class를 갖고 있다. LeafNode와 NonLeafNode는 각각 Node를 extends한다. KeyPair와 NodePair는 각각 Pair를 extends하고 LeafNode와 NonLeafNode에서 key와 value, 혹은 child와 key의 pair를 묶어주는데 사용된다.

#### <FileIO>

file 입출력과 관련된 함수들이 존재한다.

file을 받아서 리스트에 저장하고, 트리를 저장하거나 불러오는 함수들이 구현되어있다. 인자로써 받아온 input file과 delete file을 저장하기 위한 inputList와 deleteList가 있다.

#### <Main>

전체적인 코드의 흐름에 대한 class이다.

명령어를 입력받고 받은 명령어에 따른 동작을 수행한다.

### (2) 각 명령어들의 수행 알고리즘

#### <Insertion>

1. tree가 삽입 전이라면, tree를 새로 만들고 첫 노드를 삽입한다.
2. root부터 차례로 움직여 삽입할 leafNode를 찾는다.
3. 그 노드안의 배열 p에서 pair를 삽입 할 위치를 찾고 삽입한다.
4. 삽입 후 node의 m이 degree보다 크거나 같다면 split을 한다.
5. nonLeafNode에 대한 삽입도 진행한다.

#### <NonLeafNodeInsertion>

1. nonLeafRoot가 null이라면 새로운 nonLeafNode를 만들고 첫 노드를 삽입한다.
2. nonLeafRoot부터 차례로 움직여 삽입할 NonLeafNode를 찾는다.
3. 새로운 pair를 만들고 key와 leftChild의 정보를 저장한다.
4. 그 노드의 배열 p에서 어떤 위치에 pair를 저장할지 찾고 pair를 그 위치에 삽입한다.
5. 삽입 후 m이 degree보다 크거나 같다면 split을 진행한다.

#### <Deletion>

1. tree에 해당 key가 존재하는지 찾고 없으면 없다는 메시지를 출력한다.
2. 있다면 그 위치를 찾아 nonLeafNode에서 삭제한다.
3. 삭제 후 node의 m이  $\text{degree} / 2$  보다 작다면 borrow와 merge 중 더 적합한 것을 택하여 진행한다.

##### I) Borrow

1. 같은 parent를 가지고 있는 node들을 살펴본다.
2. 이중 pair 개수가 borrow 해와도  $\text{degree} / 2$  보다 클 때만 진행한다.
3. 한 개를 빌려오고 nonLeafNode의 key 값이 바뀔 시 이를 반영한다.
4. nonLeafNode에서도 underflow가 발생하면 다시 merge 혹은 parent로부터 borrow를 진행한다.

##### II) Merge

1. 한쪽으로 pair들을 모은 뒤 nonLeafNode들을 정리한다.

#### <SingleSearch>

1. LeafNode를 따라 가며 key들을 출력하고 찾아야하는 key가 있는 node에서 멈춘다.
2. 그 노드에서 값을 비교하며 원하는 key를 찾아 value를 출력한다.

#### <RangeSearch>

1. LeafNode를 따라가며 입력한 두 개의 key 사이에 있는 key와 value들을 차례로 출력한다.

## 2. 함수에 대한 자세한 설명

### (1) FileIO 관련 함수

#### <readFile>, <readDeleteFile>

파일의 경로를 이용해 파일안의 내용을 읽고 이를 list에 저장

#### <creation>

-c 명령어 사용시 파일을 만들고 이에 degree와 tree를 저장

#### <makePair>

입력받은 리스트의 key와 value 값들을 pair로 묶어서 pairList에 저장

## (2) BPlusTree 함수

<insertion>

입력받은 pair를 tree에 삽입한다.

<insertNonLeafNode>

split이 발생할 시에 발생하는 nonLeafNode들을 생성하고 삽입한다.

<deletion>

입력 받은 key를 갖고 있는 pair를 삭제한다.

<singleSearch>

입력받은 key가 tree에 있는지 판별하고 key를 찾는 path에 있는 key들을 출력한다.  
또한 같은 node에 있는 pair들의 key와 value쌍을 출력하고 key가 tree에 있으면  
그 key의 value 값을 출력한다.

<rangeSearch>

입력받은 두 key 사이의 존재하는 key값들의 key와 value를 출력한다.

<findLocation>

입력 받은 key의 값이 어디에 insert될지 찾기 위한 함수이다.  
key가 들어갈 적당한 위치를 찾고 그 노드를 반환한다.

<findNonLeafLocation>

입력 받은 key의 값이 어떤 NonLeafNode에 insert될지 찾기 위한 함수이다.  
key가 들어갈 적당한 위치를 찾고 그 노드를 반환한다.

<findDeleteLocation>

입력 받은 key를 삭제하기 위해 그 key가 존재하는 노드를 찾아 반환한다.

<printNonLeafNode>

NonLeafNode들을 가장 위인 root부터 preorder로 출력한다. index.dat에 저장하고  
불러오기 위해 만든 함수이다.

<findPrevious>

입력받은 node의 이전 node를 반환한다.

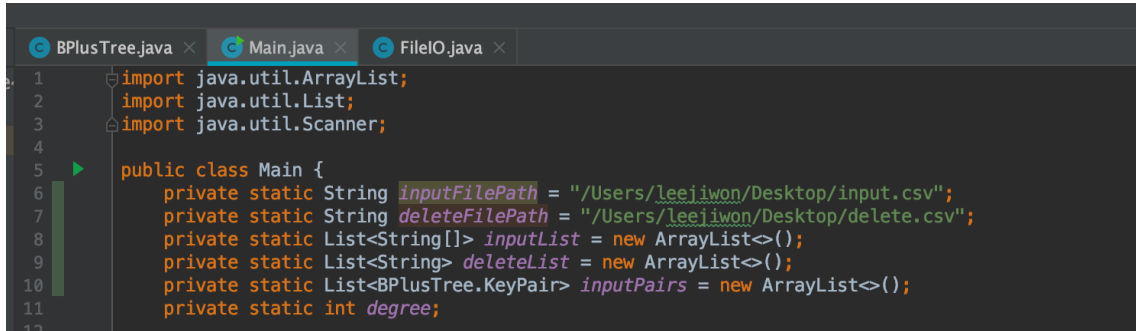
<balanceNonLeafNode>

삭제 시 nonLeafNode의 균형이 깨지면 균형을 다시 맞추기 위한 함수이다.

### 3. 컴파일 방법

#### (1) 조교님들께 양해를 구하는 부분(...ㅠㅠ)

1. dat 파일에 저장하고 csv 파일에서 불러오는 부분을 제대로 구현하지 못했습니다.



```
1 import java.util.ArrayList;
2 import java.util.List;
3 import java.util.Scanner;
4
5 public class Main {
6     private static String inputFilePath = "/Users/leejiwon/Desktop/input.csv";
7     private static String deleteFilePath = "/Users/leejiwon/Desktop/delete.csv";
8     private static List<String[]> inputList = new ArrayList<>();
9     private static List<String> deleteList = new ArrayList<>();
10    private static List<BPlusTree.KeyPair> inputPairs = new ArrayList<>();
11    private static int degree;
12 }
```

-> main 함수의 가장 위에 선언된 상수 inputFilePath와 deleteFilePath 부분에 경로를 작성 후 컴파일 해주세요.

-> degree를 파일에 저장하고 불러올 수 없습니다. -c 명령어 사용시 마지막에 있는 degree를 변수에 저장 후 사용하니, 그 부분먼저 test 후 insertion을 해주세요.

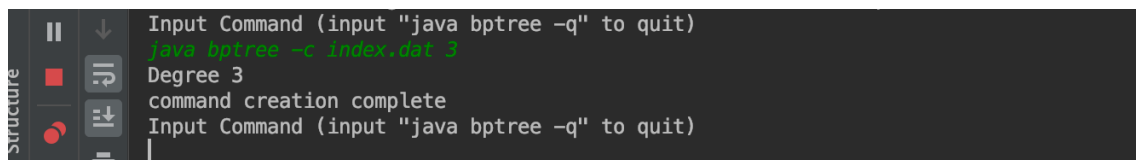
2. Delete 함수를 완벽하게 구현하지 못했습니다.

-> singleSearch와 rangeSearch를 먼저 test 해주세요.

#### (2) 컴파일 방법

##### 1. Creation

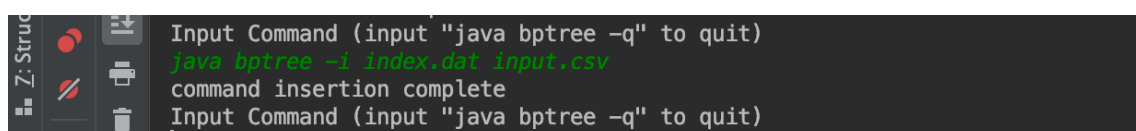
java bptree -c indexFile degree



```
Input Command (input "java bptree -q" to quit)
java bptree -c index.dat 3
Degree 3
command creation complete
Input Command (input "java bptree -q" to quit)
```

##### 2. Insertion

java bptree -i indexFile inputFile



```
Input Command (input "java bptree -q" to quit)
java bptree -i index.dat input.csv
command insertion complete
Input Command (input "java bptree -q" to quit)
```

### 3. SingleSearch

java bptree -s indexFile key

```
Input Command (input "java bptree -q" to quit)
java bptree -s index.dat 26
9 10 20
26의 value: 1290832
Input Command (input "java bptree -q" to quit)
```

### 4. RangeSearch

java bptree -r indexFile startKey endKey

```
Input Command (input "java bptree -q" to quit)
java bptree -r index.dat 10 80
10, 84382
20, 57455
26, 1290832
37, 2132
68, 97321
```

### 5. Deletion

java bptree -d indexFile deleteFile

```
Input Command (input "java bptree -q" to quit)
java bptree -d index.dat delete.csv
command deletion complete
```

### 6. Quit

java bptree -q

```
Input Command (input "java bptree -q" to quit)
java bptree -q
Disconnected from the target VM, address: '127.0.0.1:51275', transport: 'socket'

Process finished with exit code 0
```