

# <<B+ tree\_Assignment 보고서>>

2017029452 김용성

## 1. Index.dat 저장 방법

Index.dat에는 첫줄에 m 을 저장(B+tree의 차수)

다음 줄 부터는 key,value를 순서대로 저장합니다.

Ex) 5차 B+tree의 index.dat

5

1,100

2,200

3,250

## 2. 알고리즘 요약 및 함수 설명

### - MainTest.java

프로그램의 main함수가 위치하여 command 를 처리해주는 역할을 합니다.

Insertion, Deletion, Search, RangeSearch command가 실행되었을 때, index.dat에 저장된 key, value 값을 한줄 씩 읽어 B+ tree를 생성합니다. 단, Insertion, Deletion을 수행하고 나서는 다시 Index.dat에 저장하는 방식입니다.

### 1) public static void dataInsert(Bplustree b,Scanner inputStream)

index.dat 혹은 input.csv에 저장되어 있는 key, value를 이용하여 B+tree 를 만들어주는 함수입니다.

### - Bplustree.java

Bplustree를 관리하는 역할을 합니다. 구조를 설명하면 Node class를 상속받는 Bplustree class 의 내부클래스 InternalNode class와 LeafNode class를 만들었고 LeafNode class안에 key와 value를 저장하는 내부클래스인 Data class를 만들었습니다.

모든 함수는 기본적으로 Insert함수와 Delete함수에서 도움을 주기 위해 사용하기 때문에 Insert와 Delete함수를 중심으로 작성하겠습니다.

### 1) insert(int key, int value) -> Bplustree class

insert함수는 key와 value를 매개변수로 받고 LeafNode를 찾아 key와 value를 저장합니다.

Insert 를 처음 실행할 때는(firstLeaf == null) LeafNode를 생성하여 값을 저장합니다.

그 후 Insert에서는 삼항연산자를 사용하여 LeafNode를 찾아 Insert를 실행합니다. Insert 함수 (LeafNode의 Insert 함수)의 반환 값이 false라면, 즉 넣고자 하는 LeafNode에 key값이 들어가지 못한다면 조건문의 아래를 실행합니다.

중간 값을 기준으로 key value 를 나누어서 부모 노드에 중간 값을 넣어주거나 부모를 새로 만들어 주는 과정을 실행합니다. 그 후 root 값이 존재하지 않는다면 루트를 설정해주거나 부모 노드를 탐색하며 split과정을 반복합니다.

### 2) delete(int key) -> Bplustree class

delete함수는 key를 매개변수로 받고 LeafNode의 key와 value를 삭제합니다. LeafNode Delete의 경우 Case를 나누어서 진행하였습니다.

**Case 1 삭제하려는 값이 LeafNode에만 존재할때.**

- 1-1 삭제 후 최소 key의 개수를 만족할때
- 1-2 삭제 후 최소 key의 개수보다 적을때.
  - 1-2-1 왼쪽에서 빌리기.
  - 1-2-2 오른쪽에서 빌리기
  - 1-2-3 왼쪽으로 합치기
  - 1-2-4 오른쪽으로 합치기

**Case 2 삭제하려는 값이 LeafNode와 InternalNode 모두 존재할때.**

- 2-1 삭제 후 최소 key의 개수를 만족할 때.
- 2-2 삭제 후 최소 key의 개수보다 적을 때.
  - 2-2-1 왼쪽에서 빌리기
  - 2-2-2 오른쪽에서 빌리기
  - 2-2-3 왼쪽으로 합치기
  - 2-2-4 오른쪽으로 합치기

왼쪽 혹은 오른쪽으로 LeafNode를 합치는 과정에서 InternalNode가 B+tree의 조건을 만족하지 않을 수 있기 때문에 changeInternalNode 함수를 이용하여 반복적으로 root까지 조건을 만족하도록 코드를 구현했습니다.

**3) changeInternalNode(LeafNode ln) -> Bplustree class**

함수의 구현은 아래의 조건에 따라 수행하도록 진행하였습니다.

- 1. InternalNode의 왼쪽에서 key를 빌려오기
- 2. InternalNode의 오른쪽에서 key를 빌려오기
- 3. 빌려올 수 없다면 부모노드에서 key를 빌려오기
- 4. root가 되었을 때, InternalNode가 B+tree의 조건을 만족하지 않는다면 Height를 줄여주기.

**4) searchKey(int key) -> Bplustree class**

LeafNode에 존재하는 key를 찾아가면서 지나가는 InternalNode의 key를 출력해주는 역할을 합니다. 만약 LeafNode가 아니라면 재귀적으로 overloading 한 searchKey(int key) 함수를 이용합니다.

**5) public void rangedSearch(int start, int end) -> Bplustree class**

매개변수의 start를 기준으로 start가 존재하는 LeafNode를 탐색합니다. 그 후 rightSibling을 이용하여 start와 end 범위 사이의 key,value를 출력합니다.

**6) leftMostChild() -> Bplustree class**

가장 왼쪽의 LeafNode를 찾는 함수입니다.

**7) saveBplustree(String index\_file) -> Bplustree class**

Index\_file에 B+tree를 저장하는 함수입니다.

8) splitInternalNode(InternalNode n) -> Bplustree class

Insert 할 때 InternalNode의 split에 이용되는 함수입니다.

9) splitChild(InternalNode n, int mid) -> Bplustree class

Insert 할 때 매개변수 n의 child\_pointer를 split할 때 이용되는 함수입니다.

10) splitKey(Integer[] key, int mid) -> Bplustree class

Insert 할 때 매개변수 key의 child\_pointer를 split할 때 이용되는 함수입니다.

11) findLeaf(int key) -> Bplustree class

Key값이 존재하는 LeafNode를 찾아주는 함수입니다.

12) nullSearch(Data[] data) 와 nullSearch(Node[] n) -> Bplustree class

매개변수에서 첫번째의 null이 존재하는 인덱스를 반환하는 함수입니다.

13) getMidpoint() -> Bplustree class

중간값을 반환하는 함수입니다.

14) sortData(Data[] data) -> Bplustree class

매개변수를 오름차순으로 정렬해주는 함수입니다.

15) splitData(LeafNode l, int mid) -> Bplustree class

mid값을 기준으로 data를 split해주는 함수입니다.

16) aboveInternalNode(InternalNode in, int key) -> Bplustree class

InternalNode에 지우려는 key값이 존재하는지를 확인하는 함수입니다.

17) addChild(Node n) -> InternalNode class

InternalNode에 child\_pointer를 추가해주는 역할입니다.

18) findIdxPointer(Node n) -> InternalNode class

Child\_pointer에서 매개변수 n을 찾아주는 함수입니다.

19) insertChild (Node n, int idx) -> InternalNode class

idx위치에 child\_pointers를 추가해주는 함수입니다.

20) removePointer(int idx) -> InternalNode class

Idx의 위치에 child\_pointers를 삭제해주는 함수입니다.

21) shrinkHeight(InternalNode ancestor, InternalNode in) -> InternalNode class

Delete할때 height가 줄어들도록 하는 역할입니다.

22) rootKeyLeftBorrow(InternalNode ancestor, InternalNode parentLeft, int idx) -> InternalNode class

InternalNode에서 parent의 key를 빌려오는 역할입니다.

23) rootKeyRightBorrow(InternalNode ancestor, InternalNode parentRight, int borrowLeftPointer)

-> InternalNode class

InternalNode에서 parent의 key를 빌려오는 역할입니다.

24) borrowLeftInternalNode(InternalNode ancestor, InternalNode parentLeft, int borrowLeftPointer) -> InternalNode class

왼쪽 InternalNode의 key를 빌려오는 역할입니다.

25) borrowRightInternalNode(InternalNode ancestor, InternalNode parentRight, int borrowLeftPointer) -> InternalNode class

오른쪽 InternalNode의 key를 빌려오는 역할입니다.

26) deleteLeafNode(int idx)

Idx위치에 data를 지우는 역할입니다.

27) isFull()

노드가 가득차있는지 확인하는 함수입니다.

28) leftSibling(LeafNode ln)

매개변수로 주어진 LeafNode의 leftSibling을 찾아주는 함수입니다.

28) insert(Data d)

LeafNode에 data를 추가해주는 함수입니다.

29) borrowLeftLeafNode(LeafNode leftSibling, int deletIdx)

현재 LeafNode의 왼쪽 LeafNode에서 data를 빌려오는 함수입니다.

30) borrowRightLeafNode(LeafNode rightSibling, int deletIdx)

현재 LeafNode의 오른쪽 LeafNode에서 data를 빌려오는 함수입니다.

31) mergeLeftLeafNode(LeafNode leftSibling, int deletIdx)

현재 LeafNode와 왼쪽 LeafNode를 합쳐주는 함수입니다.

32) mergeRightLeafNode(int deletIdx)

현재 LeafNode와 오른쪽 LeafNode를 합쳐주는 함수입니다.

### 3. 컴퓨터에서 컴파일 방법

실행파일을 실행시키기 위해 아래와 같은 명령어를 입력합니다.

**java -cp Bplustree.jar workspace.MainTest -c index.dat 5**

```
test — -zsh — 80x24
[yongseongkim@Yongui-MacBookAir test % ls
Bplustree.jar  delete.csv      input.csv
[yongseongkim@Yongui-MacBookAir test % java -cp Bplustree.jar workspace.MainTest
-c index.dat 5
[yongseongkim@Yongui-MacBookAir test % ls
Bplustree.jar  delete.csv      index.dat      input.csv
yongseongkim@Yongui-MacBookAir test % █
```

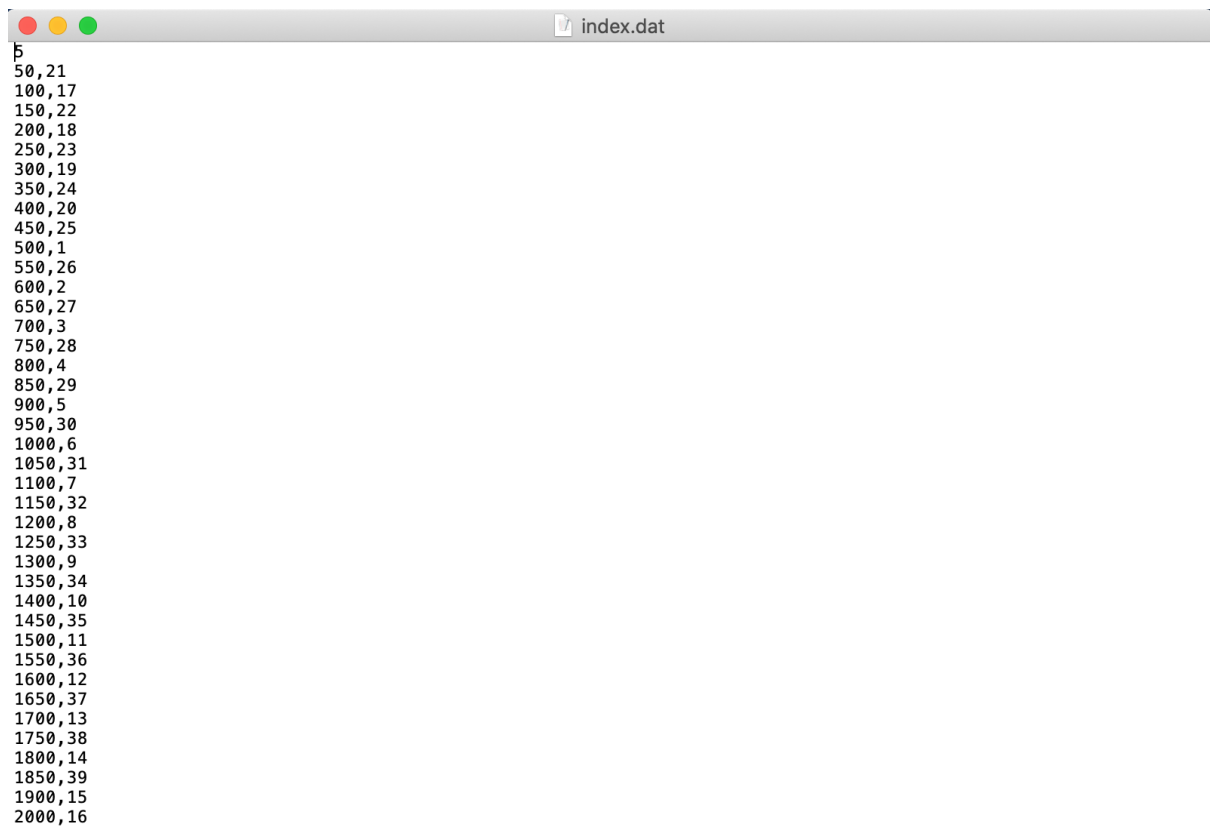
그 후 insert, delete, rangeSearch, search를 실행합니다.

- insert

java -cp Bplustree.jar workspace.MainTest -i index.dat input.csv

```
test — -zsh — 80x24
[yongseongkim@Yongui-MacBookAir test % ls
Bplustree.jar  delete.csv      index.dat      input.csv
[yongseongkim@Yongui-MacBookAir test % java -cp Bplustree.jar workspace.MainTest
-i index.dat input.csv
yongseongkim@Yongui-MacBookAir test % █

input.csv
500,1
600,2
700,3
800,4
900,5
1000,6
1100,7
1200,8
1300,9
1400,10
1500,11
1600,12
1700,13
1800,14
1900,15
2000,16
100,17
200,18
300,19
400,20
50,21
150,22
250,23
350,24
450,25
550,26
650,27
750,28
850,29
950,30
1050,31
1150,32
1250,33
1350,34
1450,35
1550,36
1650,37
1750,38
1850,39
```

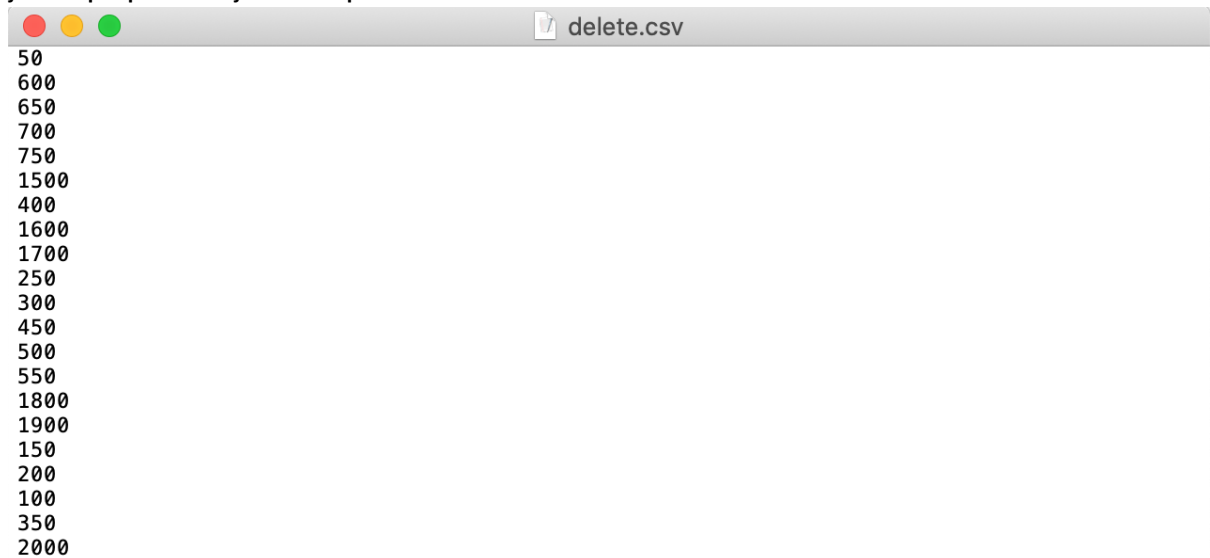


index.dat

50,21  
100,17  
150,22  
200,18  
250,23  
300,19  
350,24  
400,20  
450,25  
500,1  
550,26  
600,2  
650,27  
700,3  
750,28  
800,4  
850,29  
900,5  
950,30  
1000,6  
1050,31  
1100,7  
1150,32  
1200,8  
1250,33  
1300,9  
1350,34  
1400,10  
1450,35  
1500,11  
1550,36  
1600,12  
1650,37  
1700,13  
1750,38  
1800,14  
1850,39  
1900,15  
2000,16

- delete

java -cp Bplustree.jar workspace.MainTest -d index.dat delete.csv



delete.csv

50  
600  
650  
700  
750  
1500  
400  
1600  
1700  
250  
300  
450  
500  
550  
1800  
1900  
150  
200  
100  
350  
2000

```
index.dat
5
800,4
850,29
900,5
950,30
1000,6
1050,31
1100,7
1150,32
1200,8
1250,33
1300,9
1350,34
1400,10
1450,35
1550,36
1650,37
1750,38
1850,39
```

#### - search

```
java -cp Bplustree.jar workspace.MainTest -s index.dat 1
```

```
test — -zsh — 80x24
[yongseongkim@Yongui-MacBookAir test % java -cp Bplustree.jar workspace.MainTest ]
-s index.dat 1
1100
900,1000
NOT FOUND
yongseongkim@Yongui-MacBookAir test %
```

#### - rangeSearch

```
java -cp Bplustree.jar workspace.MainTest -r index.dat 999 1549
```

```
test — -zsh — 80x24
[yongseongkim@Yongui-MacBookAir test % java -cp Bplustree.jar workspace.MainTest ]
-r index.dat 999 1549
1000,6
1050,31
1100,7
1150,32
1200,8
1250,33
1300,9
1350,34
1400,10
1450,35
yongseongkim@Yongui-MacBookAir test %
```