데이터베이스시스템 및 응용

B+ Tree 구현

2015004957 임현택

1. Algorithm Summary

- Insertion : 인풋의 맞는 자리를 찾아 넣어 준 후 노드 안의 키의 개수 가 설정해준 degree를 넘어가면 split 해준다.

- Split : 노드 안의 중간 위치의 키를 parent로 올리고 중간 키를 기 준으로 두 개의 노드로 쪼개준다.

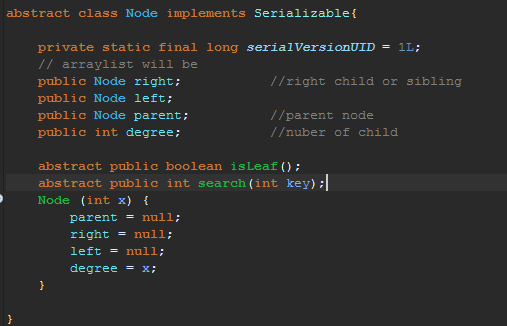
- Deletion : 인풋의 키를 찾아 삭제해 준 후 노드 안의 키의 개수가 (degree-1)/2을 만족하지 못할 경우 redistribution과 merge 를 수행한다.

- Redistribution : 왼쪽 또는 오른쪽의 형제 노드에서 키를 가져오고 부모 의 값을 업데이트 해준다.

- Merge : 키의 개수를 만족하지 못하는 양쪽의 형제 노드 중 하나의 노드와 노드를 합친다.

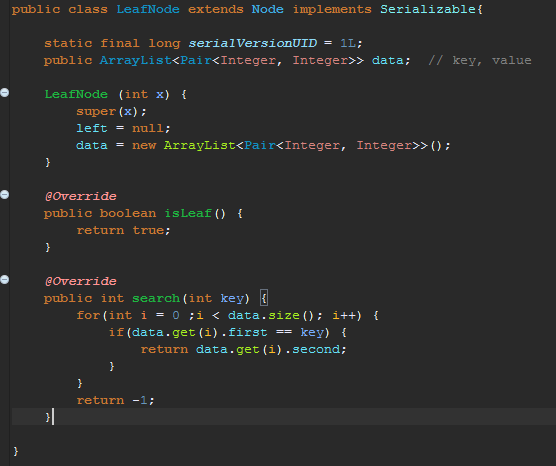
2. Detailed Description

1 ) Node Class



- right, left, parent, degree를 가진다.

1-1 ) LeafNode Class



- Node Class를 상속받은 뒤 key와 value를 저장할 Pair형 ArrayList인 data를 선언

1-2 ) TreeNode Class



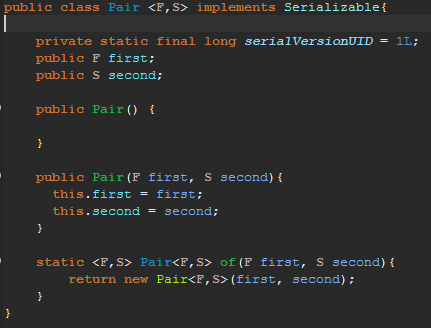
- Node Class를 상속받은 뒤 key와 자식노드를 저장할 Pair형 ArrayLis인

data를 선언

- TreeNode의 오른쪽 형제 노드를 가리킬 rightnode 선언

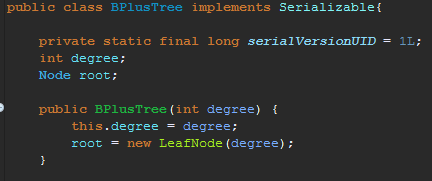
- TreeNode merge에서 사용할 temp 선언

2 ) Pair Class



- Java에는 없는 pair를 구현하기 위한 class

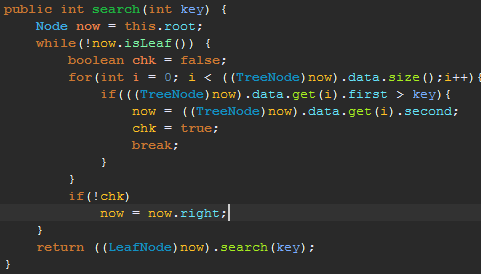
3 ) BPlusTree Class



- B+Tree의 차수를 저장할 degree 선언

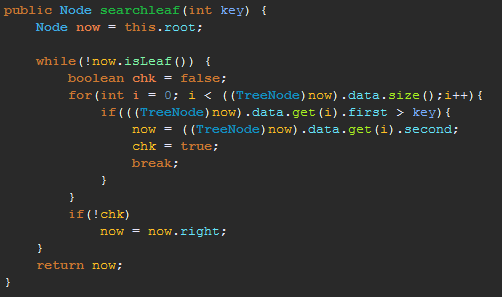
- B+Tree의 루트를 저장할 root 선언

3-1 ) search



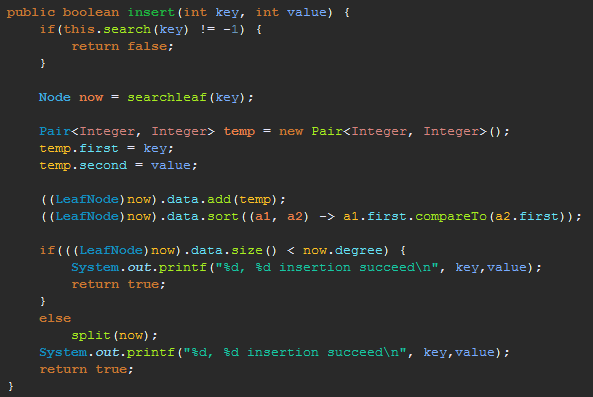
- 트리를 순회해 인풋 key에 해당하는 value를 반환

3-2 ) searchleaf



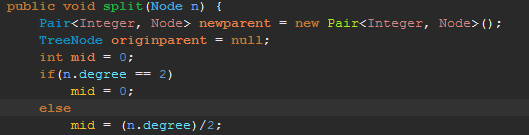
- 트리를 순회해 인풋의 키가 존재하거나 가장 가까운 리프노드 반환

3-3 ) insert



- key와 value를 받아 트리에 저장한 후 노드의 개수가 degree를 넘으면 split 호출

3-4 ) split



- split할 중간 위치 설정



- 대상이 LeafNode일때의 split

- 중간 위치를 기준, 왼쪽의 값을 newnode에 옮겨준다.

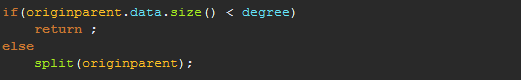
- 만약 부모노드가 없었으면 새로 만들어주고 아니면 split된 노드들을

연결할 인자를 넣어준다.

- 왼쪽, 오른쪽 형제노드에 대한 연결을 갱신해준다.

- 부모노드의 인자가 degree보다 크면 다시 split을 호출한다.





- 논리프노드(TreeNode)일 때의 split

- 중간 위치를 기준, 왼쪽의 값을 newnode에 옮겨준다.

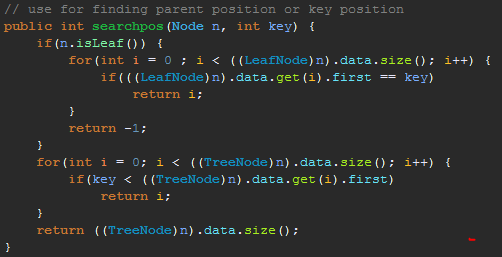
- 만약 부모노드가 없었으면 새로 만들어주고 아니면 split된 노드들을

연결할 인자를 넣어준다.

- 왼쪽, 오른쪽 형제노드에 대한 연결을 갱신해준다.

- 부모노드의 인자가 degree보다 크면 다시 split을 호출한다.

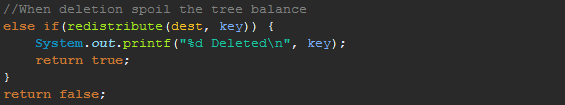
3-5 ) searchpos



- 인풋으로 받은 키의 위치 또는 키를 갖는 부모노드에서의 인자의 위 치를 반환한다.

3-6 ) delete





- 인풋으로 받은 키 값의 위치를 찾아 리프노드에서 지운다.

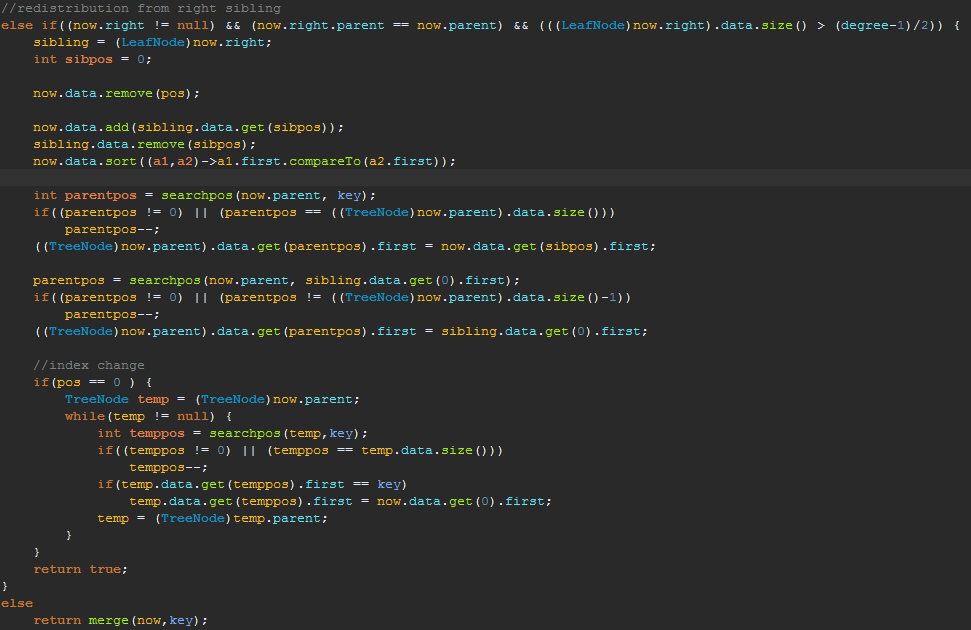
- 만약 리프노드가 루트이거나 트리의 balance에 영향을 주지 않으면 그냥 지운다.

- 아니면 redistribution을 진행한다.

3-7 )redistribution



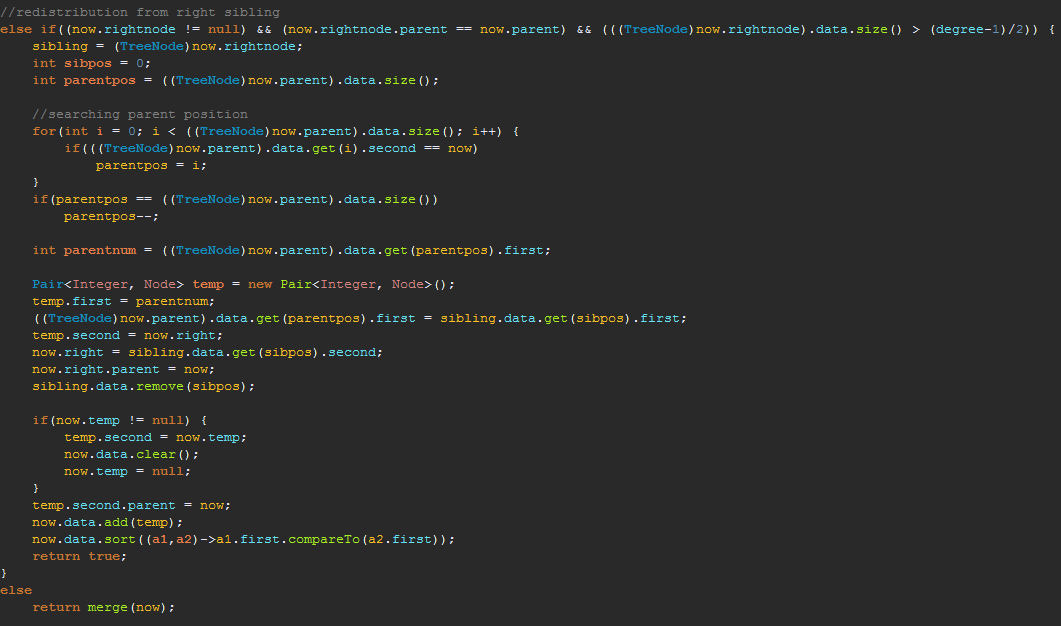
- 목표하는 리프노드의 왼쪽 노드와 redistribution을 진행하는 경우



- 목표하는 리프노드의 오른쪽 노드와 redistribution을 진행하는 경우

- 양쪽의 형제노드 둘다 redistribution을 못하면 merge를 실행한다





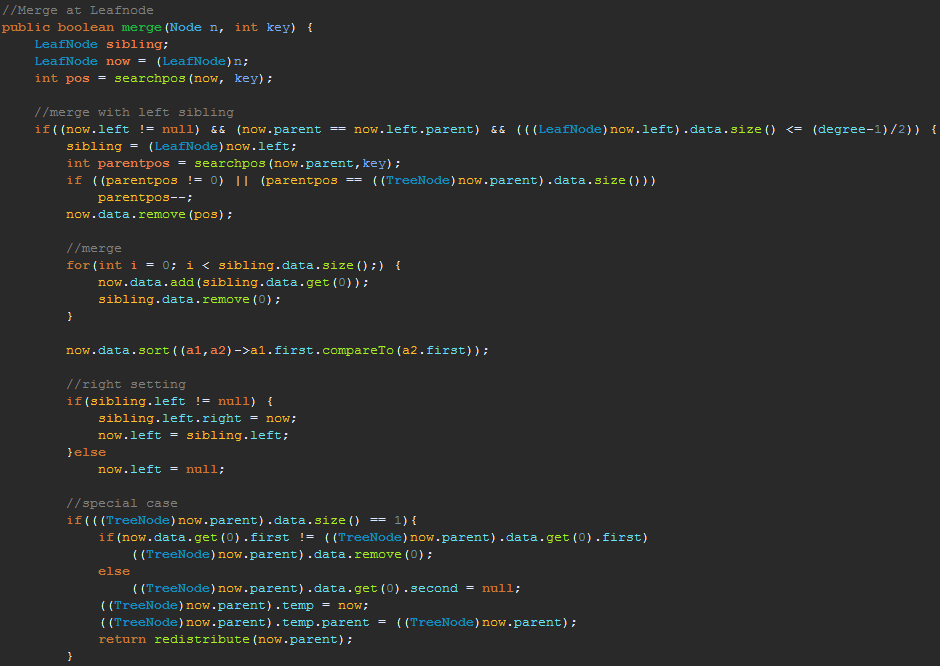
- 논리프 노드에서 redistribution을 실행하는 경우

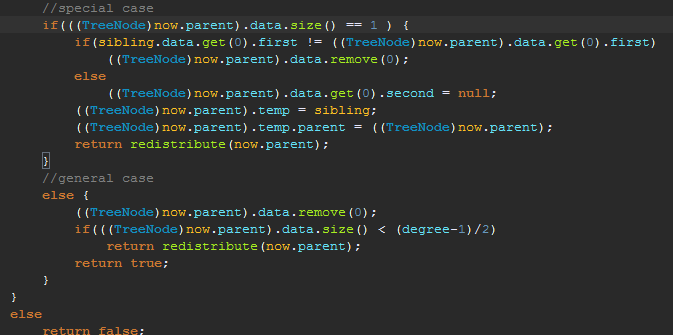
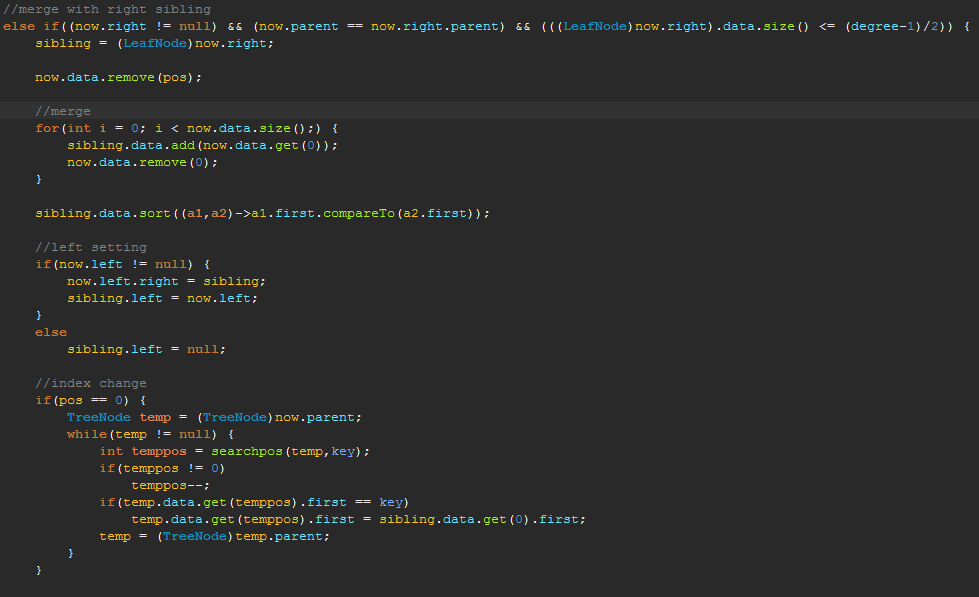
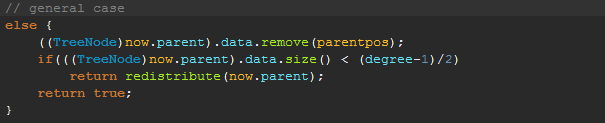
- redistribution은 부모의 인자 수에 영향을 안 미치므로 다른 함수를 불러오지 않는다.

- 왼쪽, 오른쪽 형제 노드에 대해 redistribution이 이루어 지지 않으면

merge를 호출한다.

3-8 ) merge



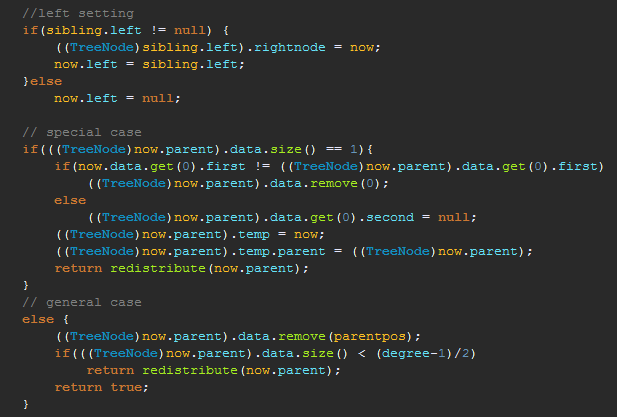


- 리프 노드에서의 merge, 받은 key의 위치를 찾아 삭제한다.

- special case는 merge로 인해 부모 노드의 링크가 하나 밖에 남지

않을때다.

- merge를 수행하지 못한다면 false를 반환한다.

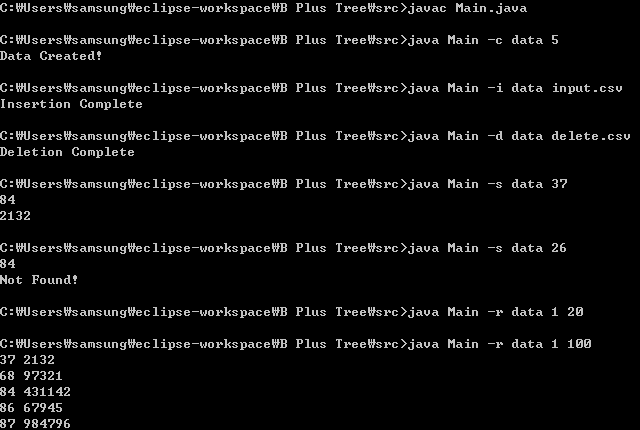




- 논리프 노드에서의 merge

- 마지막의 root setting은 merge로 인해 노드가 다 합쳐졌을 때를 의미 한다.

3. Command Guide



Compile : javac Main.java

Create : java Main -c 'filename' 'degree(integer)'

Insert : java Main -i 'filename' 'input filename'

Delete : java Main -d 'filename' 'delete filename'

Search : java Main -s 'filename' 'key(integer)'

Ranged Search : java Main -r 'filename' 'start(integer)' 'end(integer)'