2016025469 서건식

Database – B+ Tree 구현 프로젝트

요약

클래스는 총 5개로 이뤄져 있습니다.

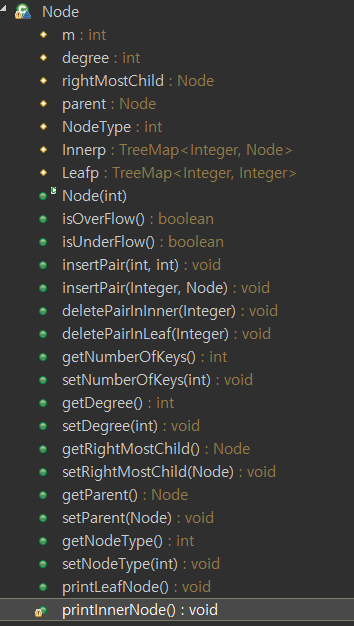
1. Node
2. InnerNode
3. LeafNode
4. BplusTree
5. Main

InnerNode는 non-leaf를 의미합니다. LeafNode와 InnerNode는 Node를 상속합니다.

Bplustree 인스턴스에 Node 객체를 만들어, Tree를 형성하였습니다.

밑으로는 클래스별로 description을 기술하겠습니다. 주요 함수만 설명드리겠습니다.

1. Node 클래스



Node 클래스의 instance와 함수들입니다.

P는 자료구조중 TreeMap을 사용하였고, Innerp와 Leafp를 구분하여 리프 일때와 아닐 때 다른 인스턴스에 저장하도록 하였습니다. (자료형이 다르기 때문)

NodeType은 리프인지 아닌지 식별하기 위한 instance입니다.

R은 rightMostChild로 구현하였습니다.

그 외 setter 와 getter등이 있고, 각 인스턴스에 대한 함수입니다.

설명드릴 함수는 아래와 같습니다.



1. insertPair 함수입니다. 오버로딩 함수로 구현했습니다.

리프일때와 아닐 때 구분해서 leafp와 innerp에 따로 넣어주고,

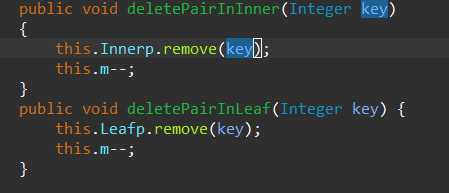
논리프 노드에서 넣을때는 3개의 조건이있습니다.

첫번째 조건은 처음 노드에 값을 집어넣을 때, p만 집어넣으면 r이 비게되므로 r을 설정해주기 위함입니다.

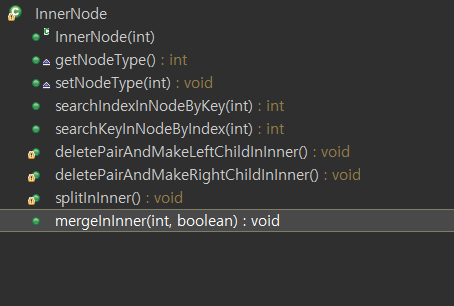
두번째 조건은 첫번째조건 다음에 들어오는 값을 제대로 집어 넣어주기 위함입니다.

세번째 조건은 일반적인 경우입니다.

모두 m을 1씩 올려줍니다.

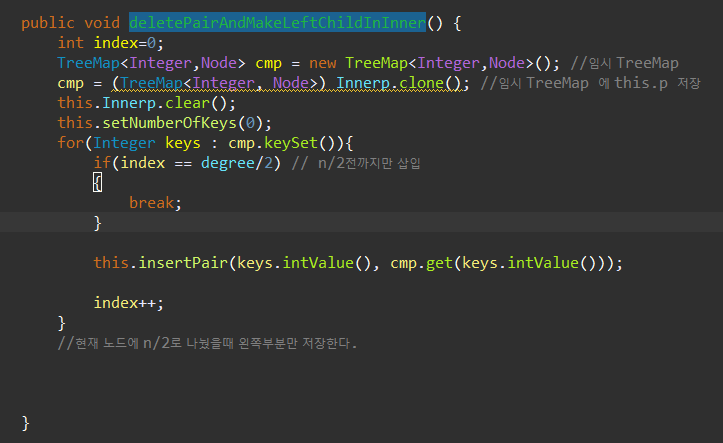


1. deletePairInInner, Leaf 함수입니다. Insert와 거의 동일합니다.
2. InnerNode



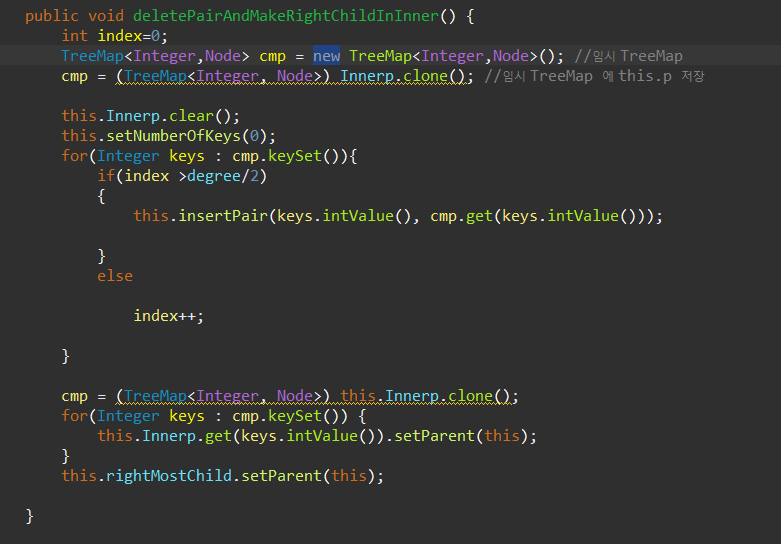
Split 과 Merge 파트로 나눠서 설명하겠습니다.

1. Split



This가 search를 통해 해당 키가 존재할 가능성이 있는 노드라고 했을 때, 그 노드의 midindex를 기준으로 왼쪽의 부분만 남겨놓고 나머지를 초기화합니다.

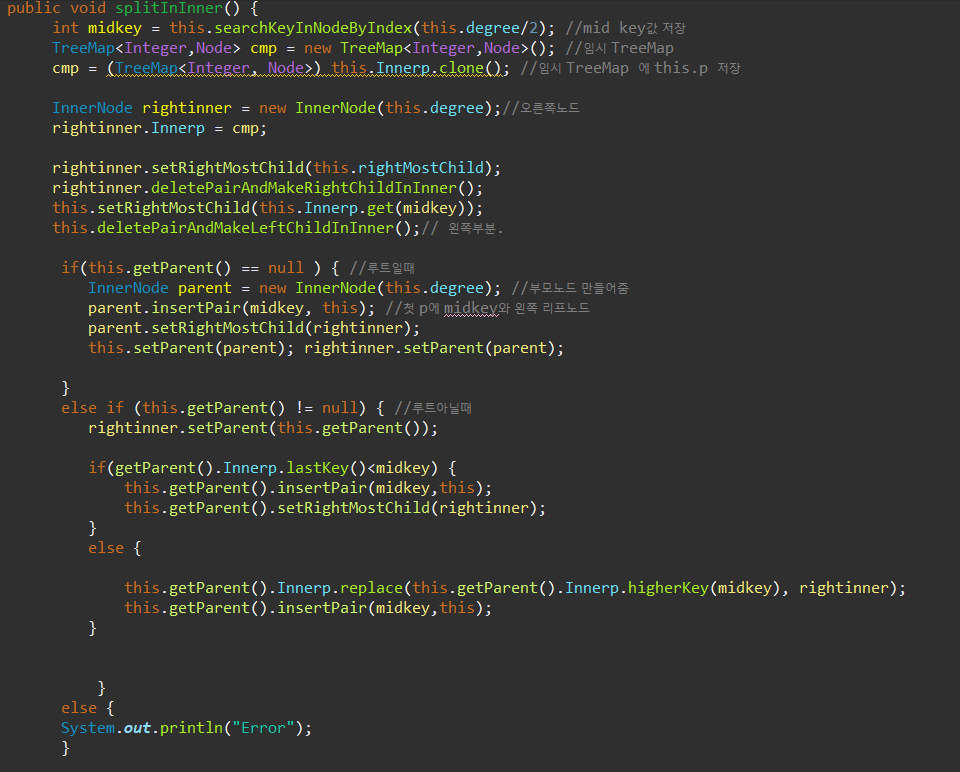
Cmp는 TreeMap으로 Innerp.clone()함수를 사용하면 그대로 복사가 됩니다. Innerp를 초기화한후에 왼쪽의 부분만 삽입합니다.



This를 기준으로 오른쪽을 만드는 함수입니다.

여기서 다른점은, 오른쪽 부분은 split함수에서 새로 변수를 할당한 후에 만들어지기 때문에 (ex: rightleaf = this.deletePairAndMakeRightChildInInner();)

마지막부분에 부모설정을 다시 해주는 부분을 제외하고는 동일합니다.



split함수입니다.

함수를 간략하게 설명드리자면, 부모가 있고 부모의 왼쪽자식이 left, 오른쪽 자식이 right라고 했을때 left가 this입니다.

이 기준으로 함수를 이해하면, 먼저 조건문에 들어가기 전에 오른쪽부분을 만들어주고, 왼쪽과 오른쪽을 이어줍니다.



이어줄 때 이렇게 코드를 구현한 것은, 이렇게 구현해야 트리가 커질 때 중간에 LinkedList가 끊기지 않고 이어지기 때문입니다. 단순히 rightinner를 rightMostChild로 설정해버리면 split이 여러 번 일어났을 때 이어지지 않는 경우가 생깁니다.

첫번째 조건문입니다.

루트일때는 부모를 생성해주고 포인터를 설정해줍니다. 이때 부모에 넣는 값은 2/degree의 키값입니다.

두번째 조건문입니다.

루트가 아닐때는 rightinner의 부모만 원래있던 부모로 설정을 해주면 됩니다.

이 조건문에서도 두가지 경우로 나뉘는데, 첫번째는 midkey가 부모의 마지막키보다 클 때 입니다. 이 경우에는 rightMostChild를 따로 설정 해줘야 하기 때문에 나눴습니다.

나머지 경우에는 midkey보다 큰 key에 rightinner로 replace해줍니다.

1. Merge

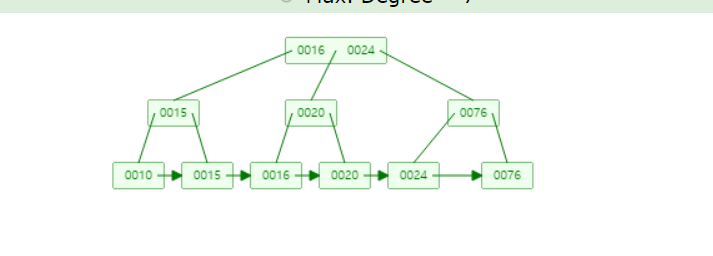


merge같은 경우는, re\_distribute가 leaf에서 일어난 후에 그 leaf의 부모또한 underflow일 때, 병합을 해주는 함수입니다.

경우는 3가지입니다.

제일 왼쪽에서 병합이 일어날 때, 제일 오른쪽에서 병합이 일어날 때, 중간에서 병합이 일어날 때 입니다.

제일 왼쪽에서 병합이 일어날 때는 부모의 제일 첫번째 값보다 this의 마지막 값이 더 작을 때 제일 왼쪽에서 병합이 일어납니다.



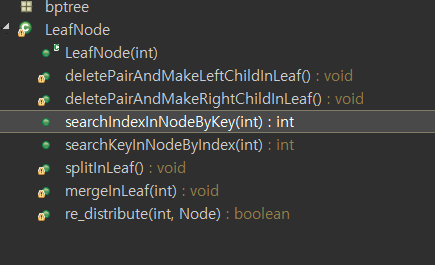
예시를 들면 15라는 innernode 가 this일때를 의미합니다. This의 마지막 값이 부모의 첫번째 값보다 작으므로 왼쪽에서 병합이 일어나는 것입니다.

합칠 노드를 20, 그니까 24의 value값으로 설정한 후에 this를 모두 지웁니다. 그리고 this가 가리키던 r을 16이 가지도록 포인터를 설정합니다. 빌린값은 (16) 부모에서 지워줍니다.

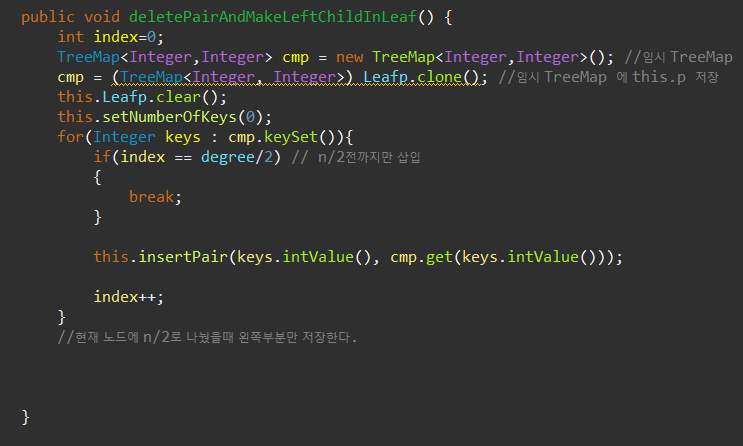
제일 오른쪽일때는 76이라는 innernode가 this일때를 의미합니다. 이 또한 구분해준 이유는 포인터 설정 때문입니다. 76은 parent에서 rightMostChild에 해당하는 부분이기 때문입니다.

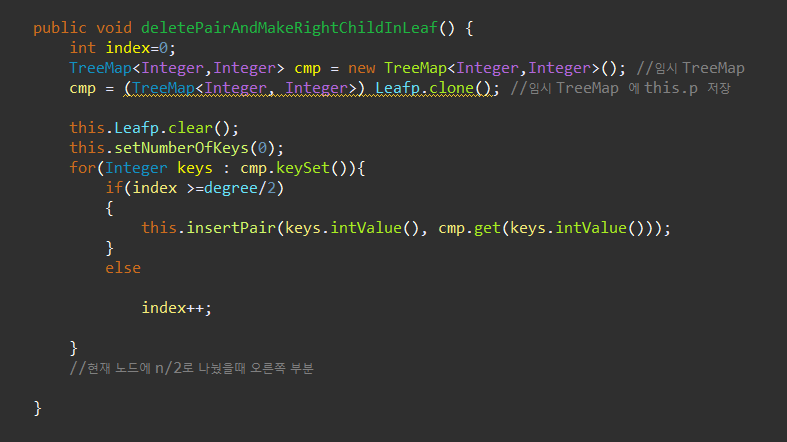
중간에서 병합이 일어날땐 root가 지워질 가능성도 있습니다. 예를들어 16이 지워질 때 입니다. 이 경우에는 rootChange라는 인자를 함수에서 불러와서 루트가 변할때와 안변할때를 나눠주었습니다.

1. LeafNode



1. Split





InnerNode와 동일합니다.



Leaf에서의 Split입니다. 왼쪽과 오른쪽을 생성하고,루트일땐 부모를 만들어서 포인터를 설정합니다. This는 righleaf를 가리키게 하면서 Linked List를 만들어줍니다.



부모가 루트가 아닐때는, 경우가 나눠집니다.

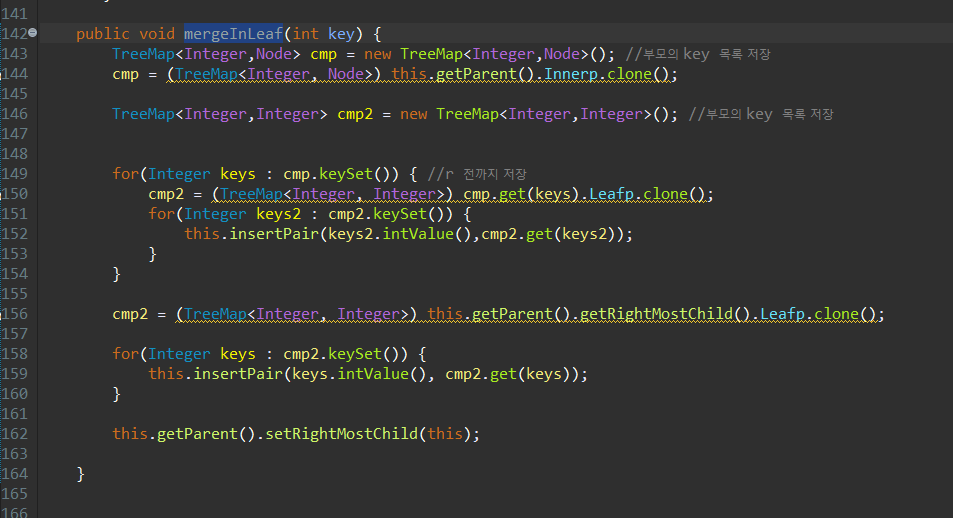
노드는 degree가 3이라고 할 때 \*ㅁ\*ㅁ\* (\*는 포인터 , ㅁ은 키값) 인데, 이렇게 되면 Treemap instance 가 2개, rightMost라는 변수 1개로 저장됩니다.

따라서 제일 오른쪽에 midkey가 들어갈땐 다르게 설정해야합니다.

중간, 왼쪽에 들어갈땐 higherkey를 사용해 midkey 오른쪽 값의 포인터에 rightleaf를 저장합니다.

제일 오른쪽일때는 , 부모노드의 제일 끝 (pdf파일에서 r을 의미)에 rightleaf를 넣어줍니다

1. Merge

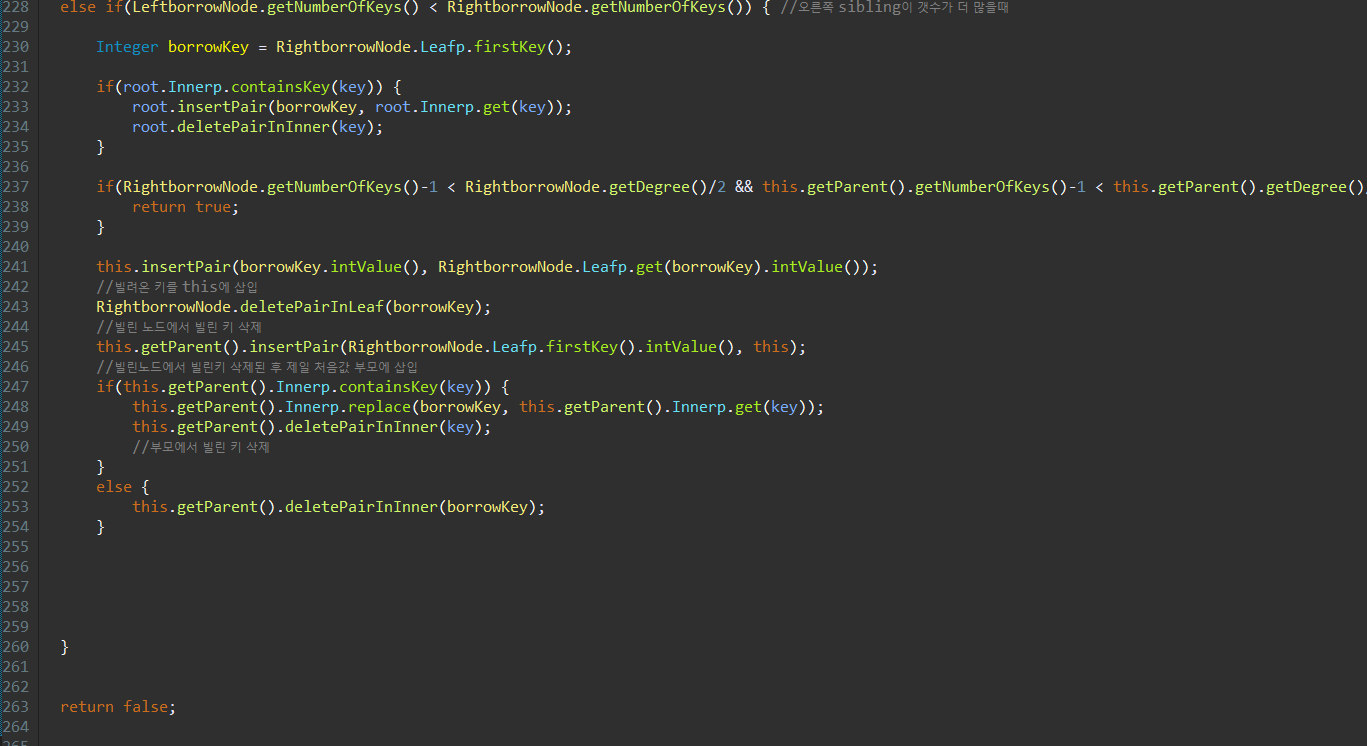
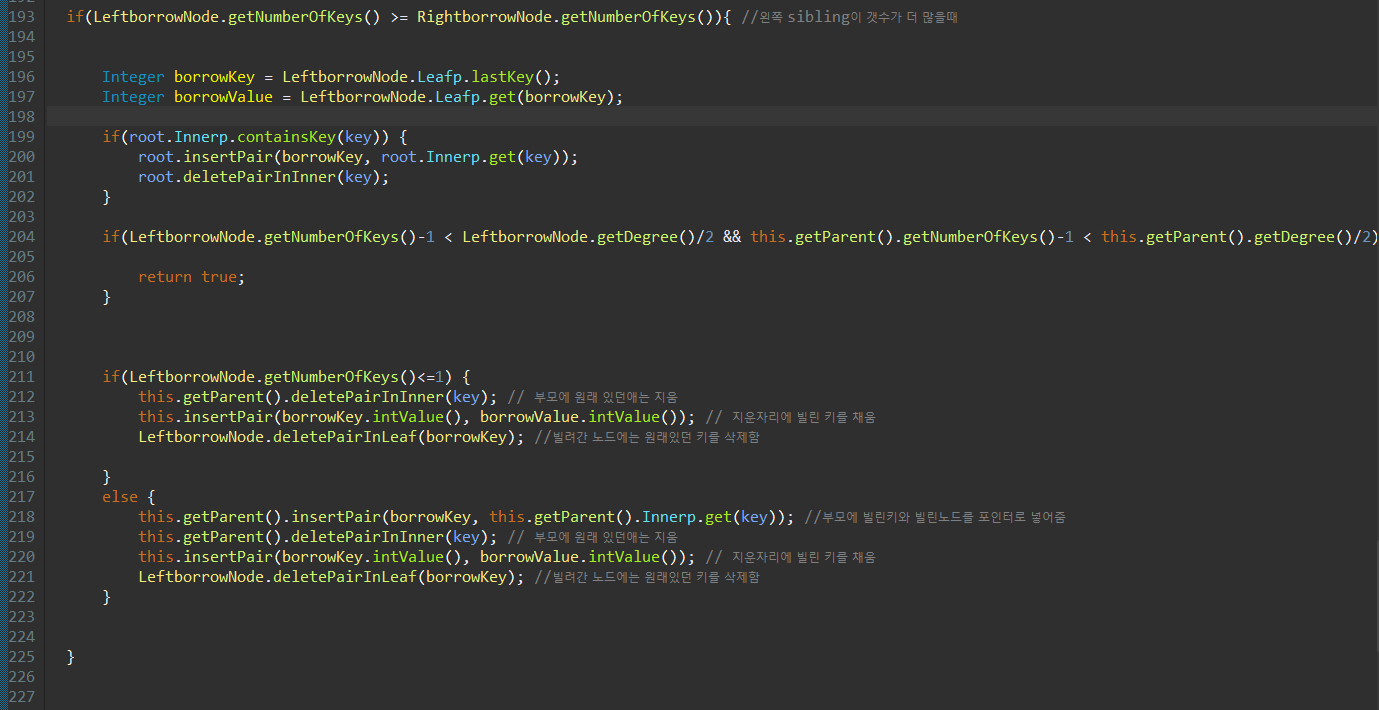


Merge함수입니다. 부모의 key와 부모의 자식의 모든 값들을 this에 저장합니다.

부모의 r을 this로 설정합니다. 이는 후에 mergeInInner에서 쓰이게 됩니다.

r전까지 저장할 때 2중반복문을 선언한 것은 부모의 자식에 접근하기 위해서 입니다.



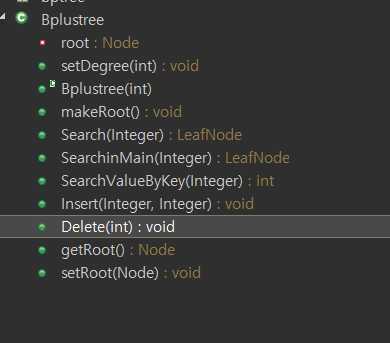


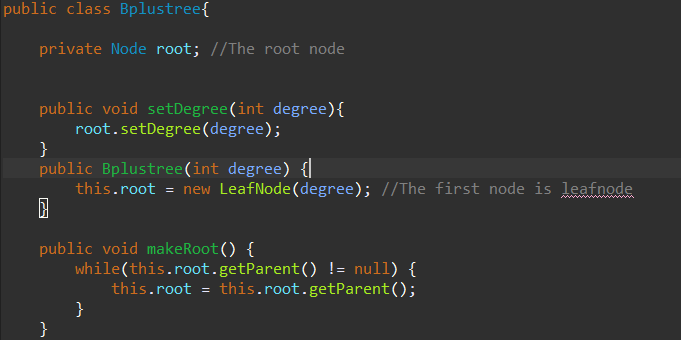
조건문이 굉장히 많지만 크게는 왼쪽에서 빌릴때와 오른쪽에서 빌릴때로 나뉩니다.

위쪽의 조건문들은 왼쪽에서 빌릴노드나 오른쪽에서 빌릴노드 둘 중 하나가 자기 자신일때를 골라주기 위한 조건문입니다.

이 함수는 return 값이 boolean 자료형입니다. 잘 분배되었으면 false, 아니면 true를 반환합니다. 이를 조건문에 넣어주었고, 루트에 해당 값이 있을때도 다루었습니다.

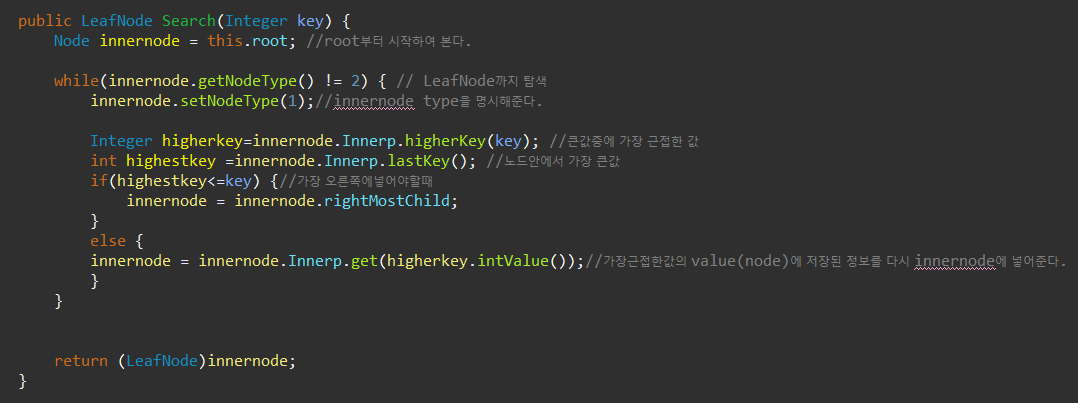
1. Bpluetree Class





Tree를 생성할 클래스입니다.

makeRoot함수는 root를 초기화 하기 위함 입니다. 부모가 없을 때 까지 올라가며 root를 설정합니다.



1. Search함수입니다.

TreeMap의 higherkey함수를 통해 bst를 구현했습니다.

첫번째 조건문은 들어간값이 해당 노드의 가장 큰값보다 클 때, r로이동하고

아닐땐 get함수(get 함수는 Treemap에서 키값에 해당하는 value를 가져오는 함수입니다.)를 통해 key값의 바로 오른쪽 값의 포인터로 들어갑니다.

SearchInMain함수는 print만 추가되었고, 나머지는 동일합니다.



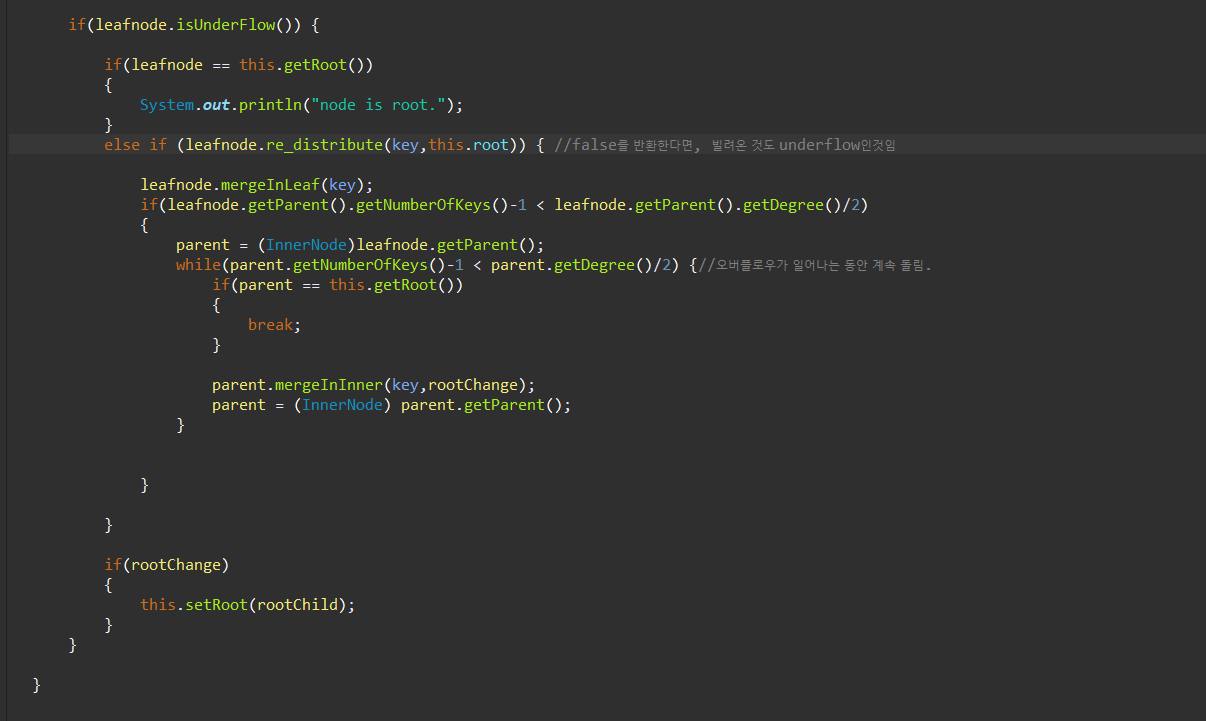
1. Insert함수입니다.

처음에 들어갈 LeafNode를 찾은다음에, 삽입을 합니다. 삽입후에 overflow를 체크한후에, leafnode를 split합니다.

만약 부모도 오버플로우가 일어났다면, 부모도 split해주면서 오버플로우가 일어나지않을때 까지 split하면서 Balance를 조정합니다.

1. Delete함수입니다.





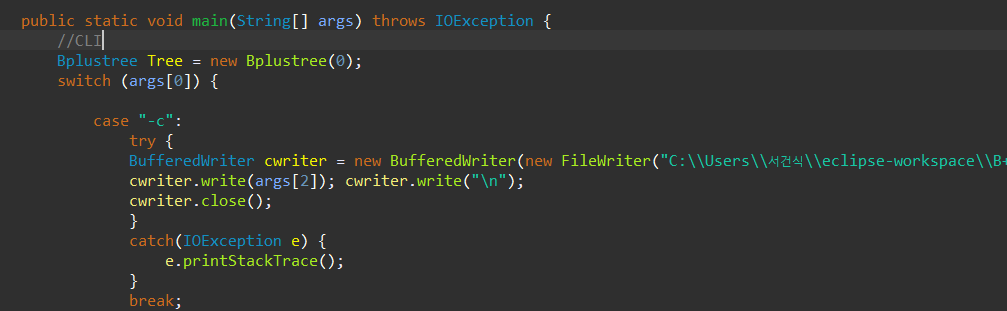
rootChange는 root가 변했는지 안변했는지에 따라 트리의 높이가 바뀌기 때문에 설정한 변수입니다. 첫번째 조건문에서 지우는 키값이 root에 있는지 확인한 후에 그 자식 노드를 미리 저장해둡니다. Root를 변경하기 위함입니다.

노드에서 키값을 지운후에, 부모가 그 키 값을 가지고 있고 해당 leafnode가 underflow가 나지 않은 경우에는 leafnode의 제일 첫번째 값을 부모의 제일 마지막값으로 Tree의 Balance를 위해 설정해야 하게 하였습니다.

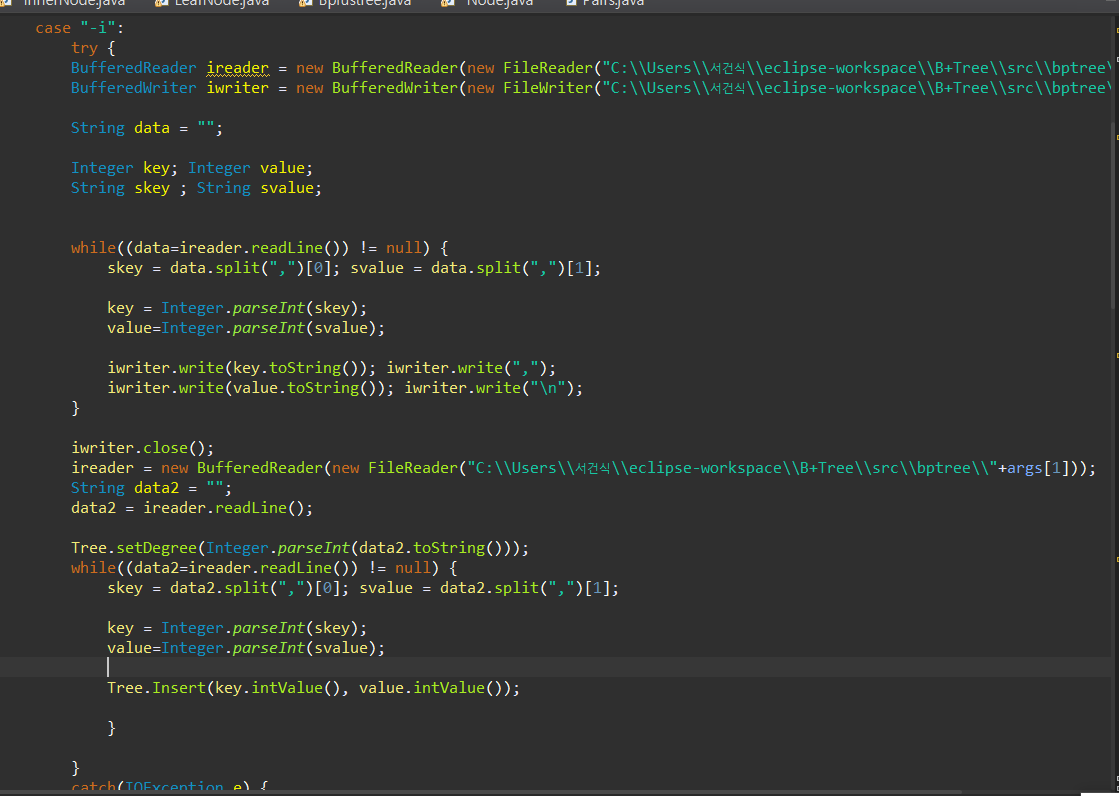
그 후 leafnode가 underflow가 발생할시에는 redistribute함수를 통해 재분배를 해줍니다. 그렇게 재분배를 해서 false가 반환되었다면 빌린 것 또한 underflow가 된 것입니다. 그 경우에는 먼저 leaf에서 merge를 해준후에 그 노드의 부모 또한 underflow라면 루트까지 검사하면서 merge합니다.

마지막에 root가 바뀌었다면, 아까 저장한 child를 루트로 설정합니다.

1. Main



1. -c: input.dat에 degree만 write합니다. 여기서 계속 초기화를 해주면 데이터가 누적되지않기 때문에 모두 -i에서 다시불러와 다시 쓰는방식을 택했습니다.



1. -i:

Insert 입니다.

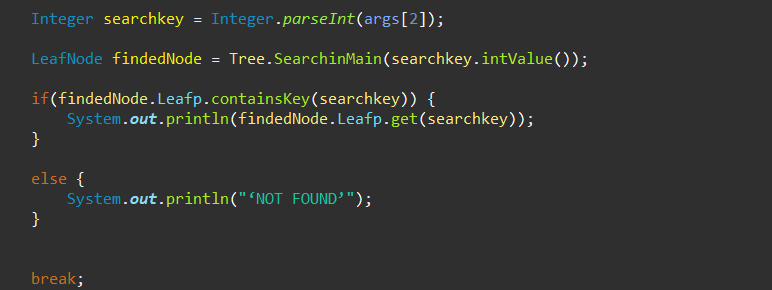
이때 tree의 degree를 설정해줍니다. 먼저 들어온 input.csv를 index.dat파일에 덮어쓰지않고 그대로 넣어준 후에, 처음부터 읽어가면서 파싱합니다.

while문에서 볼수있는데, “,”으로 문자열을 구분하면서 받고 key와 value값을 parseInt 함수를 통해서 형변환하여 저장한후에 Tree.Insert함수를 실행해 넣어줍니다.



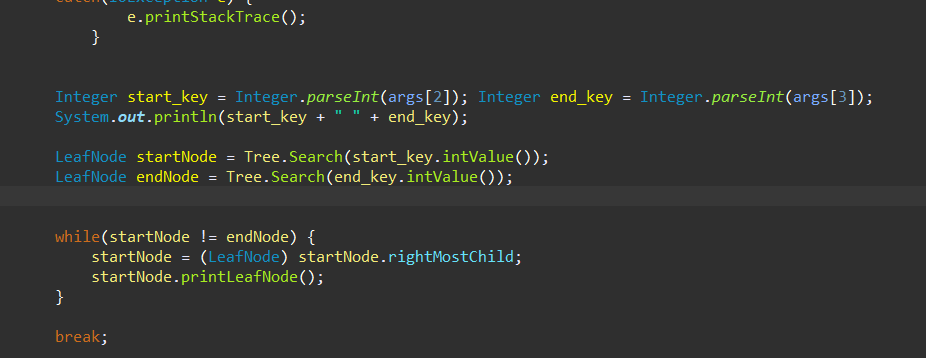
1. -s:

Search입니다. Index.dat파일을 불러온후에 -i에서 했던것과 동일한 방식으로 파싱을 통해 데이터를 삽입합니다. 여기서 또다시 Insert를 하는 이유는, 프로그램이 실행될때마다 데이터를 index.dat파일에 유지시키면서 하기 때문에, -s에서는 트리가 비었다고 가정하고 dat파일에 쓰여진 내용들을 기반으로 트리를 새로 구현합니다.



Search는 이렇게 이뤄집니다. LeafNode에 해당 key값이 존재할만한 노드를 찾아준 후에 그 노드에 key값이 있다면 찾아주고, 없다면 NOT FOUND를 출력합니다.

leaf까지 가는 경로에 InnerNode를 출력하는 것은, SearchinMain함수에서 실행됩니다.



1. -r:

범위연산입니다. 위 try - catch부분은 -s의 부분과 같으니 생략합니다.

Start key와 end key를 콘솔로 입력받은후에, start키가 존재할것 같은 노드와 end키가 존재할 것 같은 노드를 찾아준 후에,

startNode 가 endNode에 도달하기 전까지 오른쪽으로 포인터를 이동하면서 print해줍니다.

1. -d:



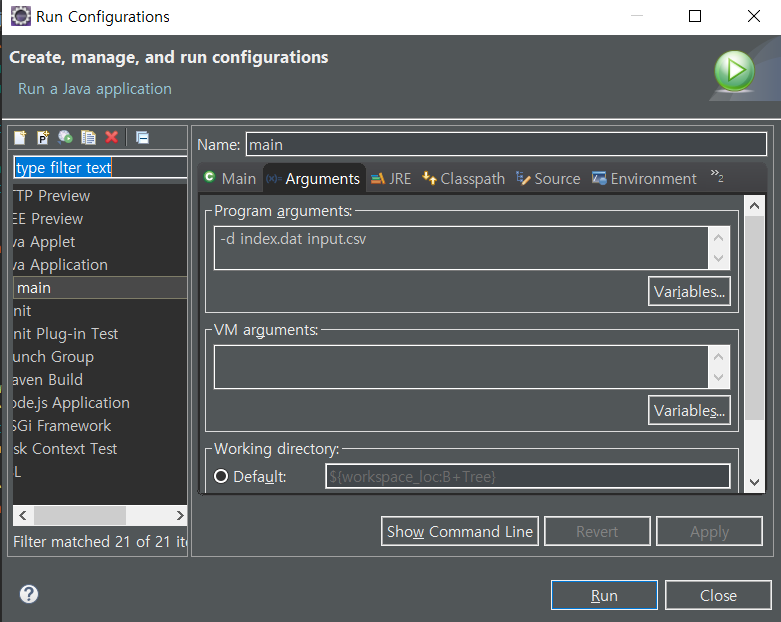
Insert와 비슷하나 다른점은, 앞서 설명했듯 index.dat파일을 처음부터 쭉 다시 읽어가면서 새로 트리를 초기화하여 구성하기 때문에 중간에 -d를 통해서 “,”와 같이 파싱되지 않는 데이터가 들어가게 되기 때문에, data2에 저장할 때 예외를 넣었습니다. Data2는 한 라인을 읽는데, 그 라인에 “,”가 포함되어 있으면 insert를 하고, 아니라면 delete를 합니다.

\*\*Instructions

:코드는 degree가 3,4,5,6모두 테스트 하였으나 디버깅은 degree가 3일 때 진행하였습니다.

cmd창에서 제 컴퓨터에서는 컴파일이 되었으나, 만약 컴파일이 되지 않는다면

eclipse상에서



Argument를 설정하여 테스트해주시면 감사드리겠습니다.

또한 파일의 위치를 제 컴퓨터 기준 src폴더에 (코드가 있는 폴더) 위치시켰는데, 이 때문에 컴파일이 안될수도 있습니다. 이 경우에는 파일의 위치를 변경해주시면 컴파일이 될 것입니다.