Data Structure Week 5

김재용 | 2024.08.07

Index

1. Multiway Search Tree
2. B Tree
3. B+ Tree

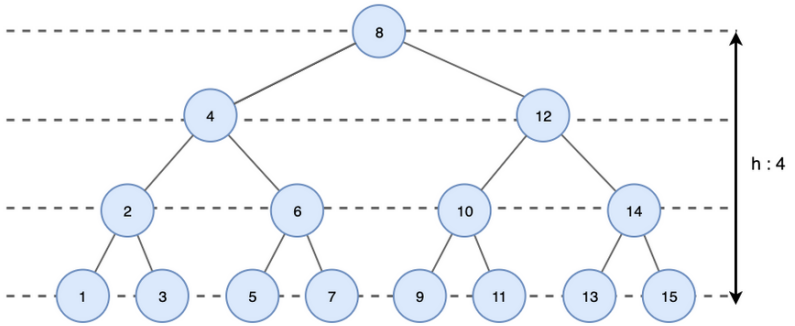
Multiway Search Tree

Binary Search Tree와의 차이점

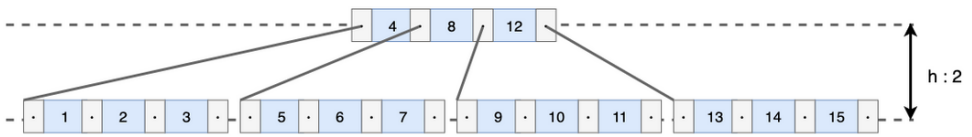
Binary Search Tree는 한 node 당 최대 2개의 child node를 갖게 됨
Tree의 높이가 최소로 잡아도 $log_2 n$이 될 수밖에 없음

-> 한 node에서 여러 갈래로 Child Node를 가질 수 있는 Tree를 제안

주로 C++ STL <map>을 사용해서 구현



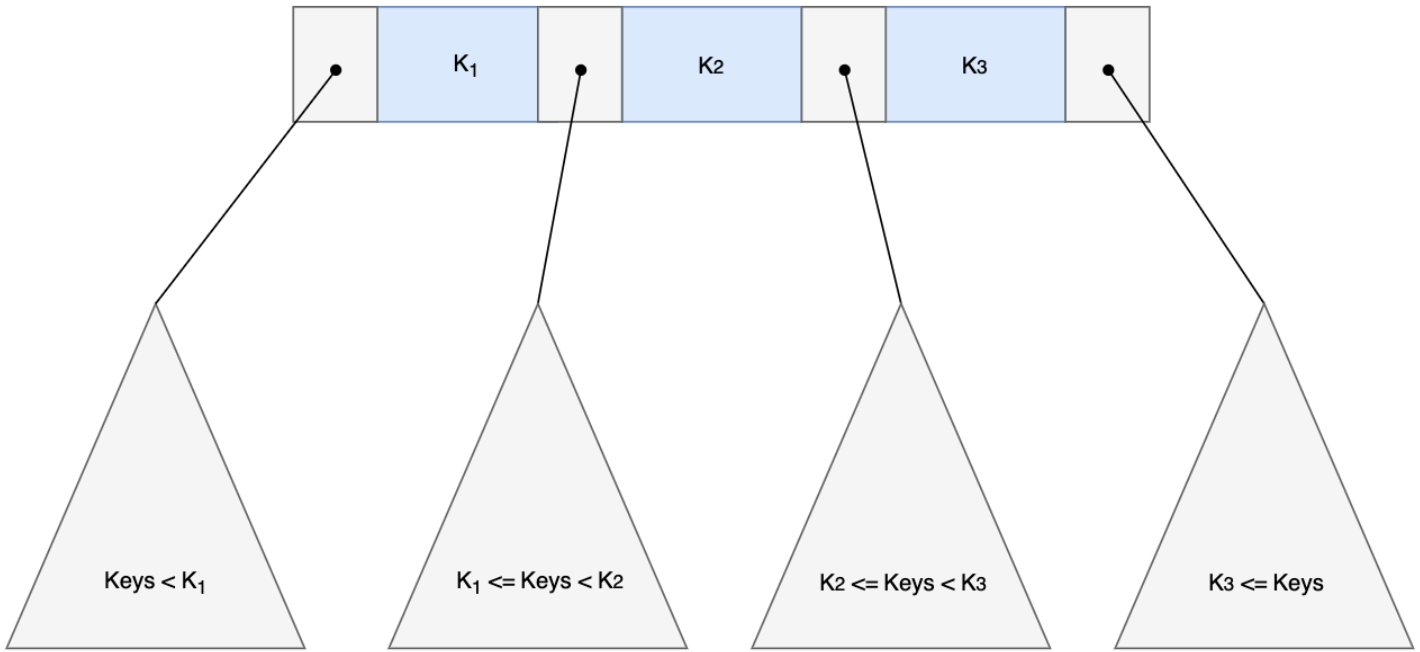
이진 탐색 트리 (Binary Search Tree)



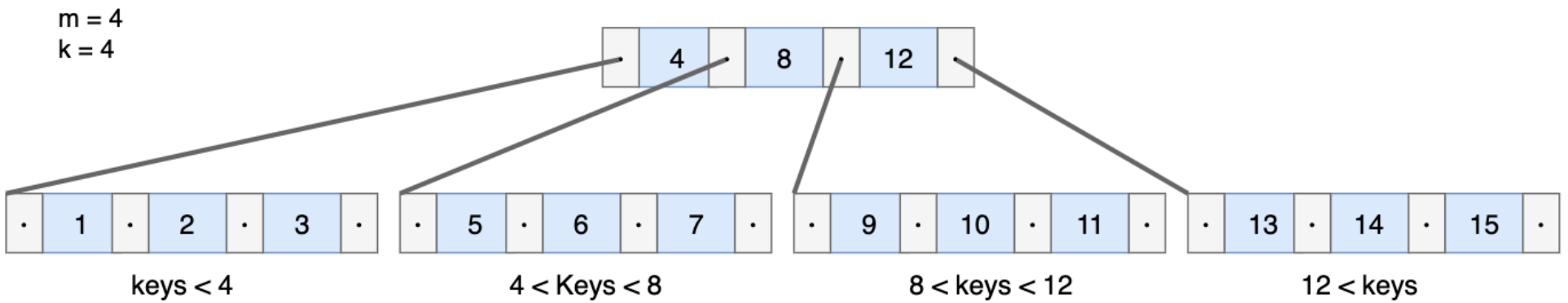
다원 탐색 트리 (m-way Search Tree)

Multiway Search Tree

K: key
m = 4



Multiway Search Tree

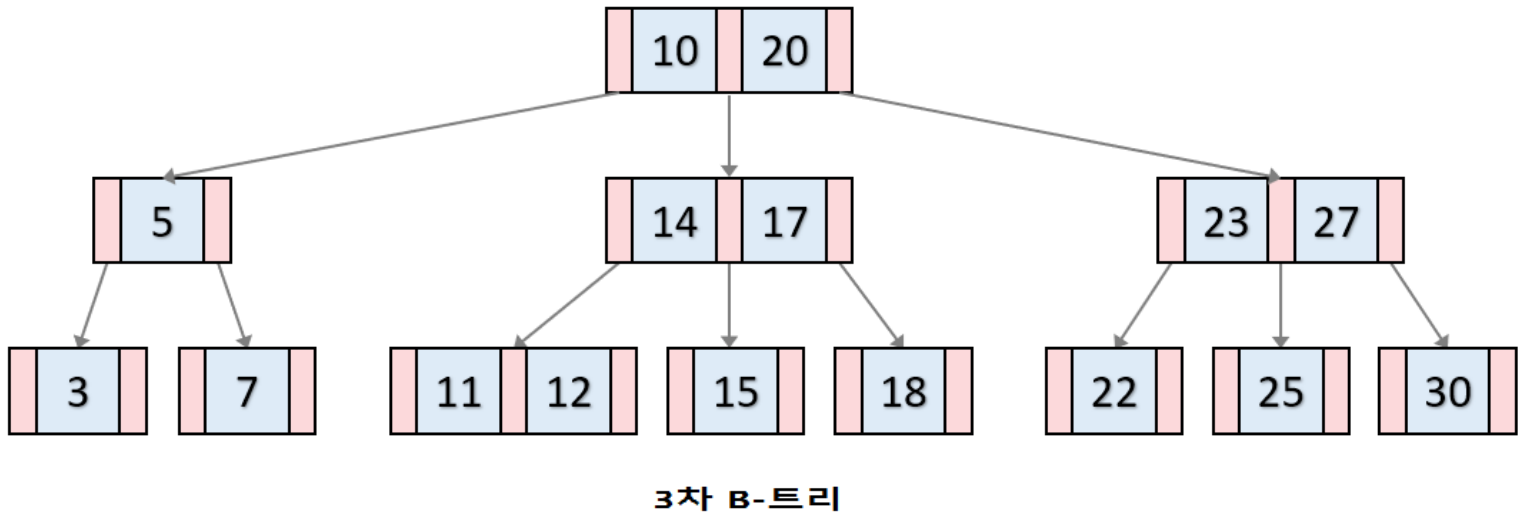


B Tree

Multiway Search Tree의 한 종류

최대 M개의 자식 노드를 가질 수 있는 B Tree를 M차 B Tree라고 함

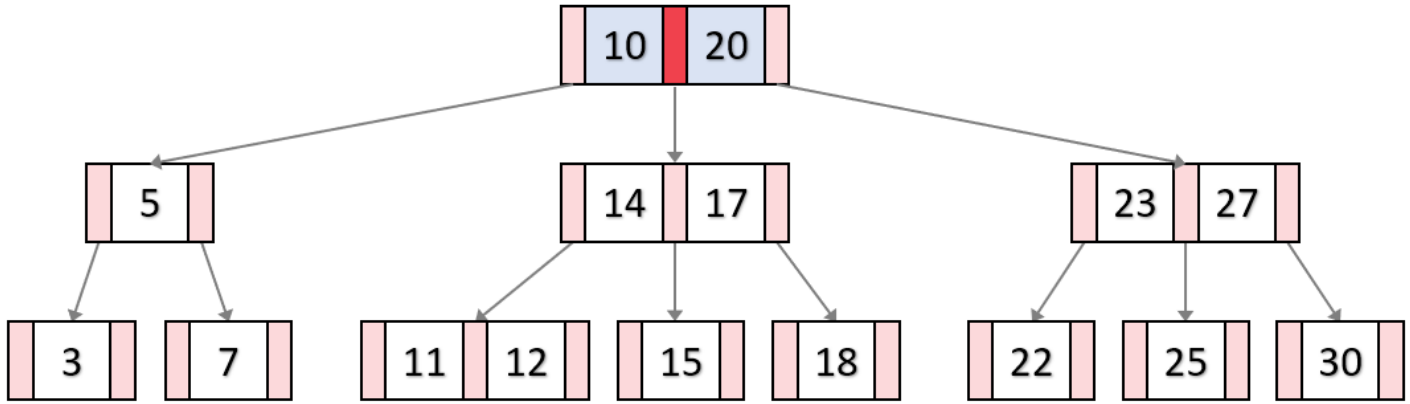
M차 B Tree는 M-1개의 Data를 갖고 있을 수 있음



3차 B-트리

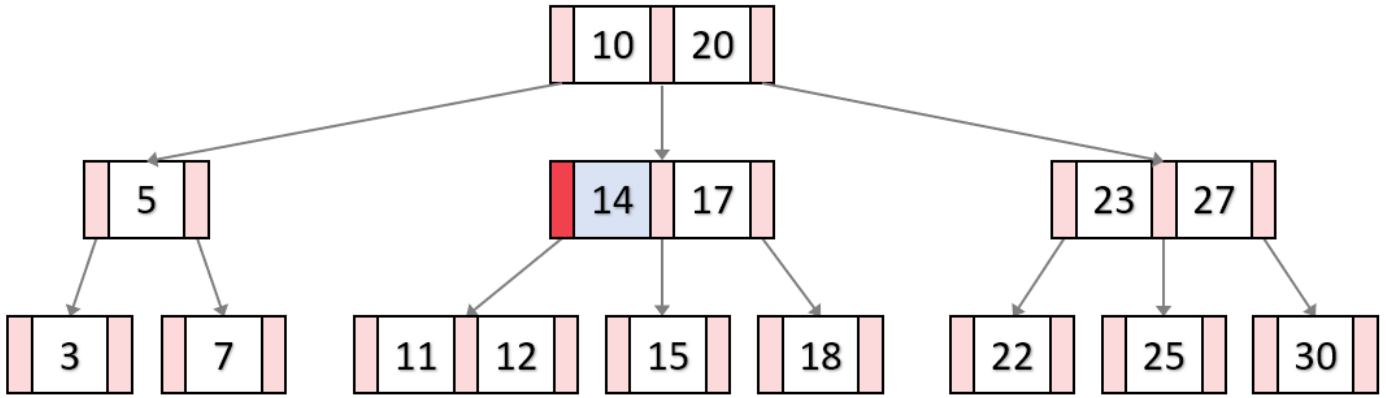
B Tree

1 13 삽입 시작, 루트노드의 key 중 10보다 크고 20보다 작기 때문에 중간 자식노드로 이동



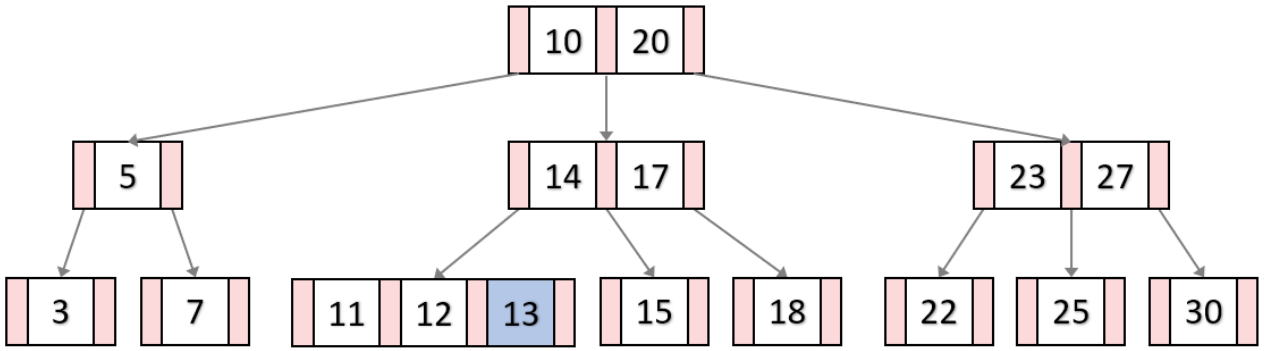
B Tree

2 13은 14보다 작기 때문에, 가장 왼쪽 자식노드로 이동



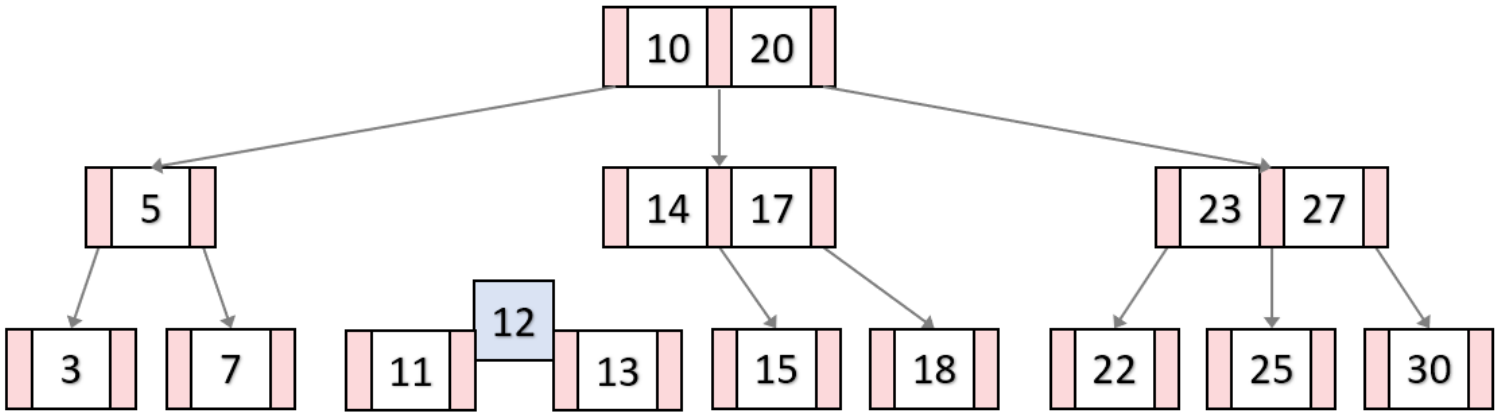
B Tree

3 13은 12보다 크기 때문에 가장 마지막에 삽입. 해당 노드가 최대로 가질 수 있는 key의 개수를 초과했기 때문에 분할을 수행



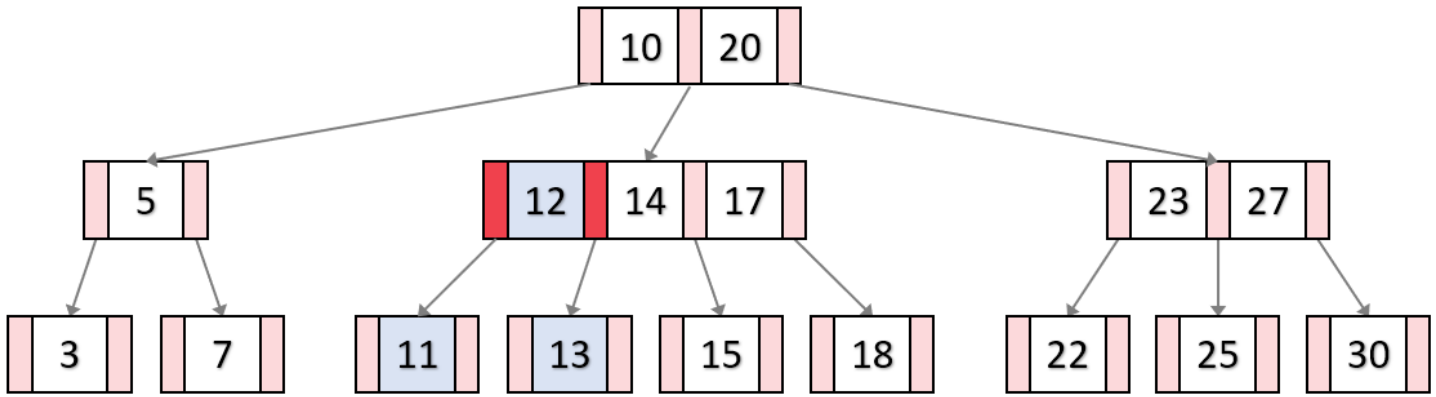
B Tree

4 중앙값 12를 기준으로 분할을 수행



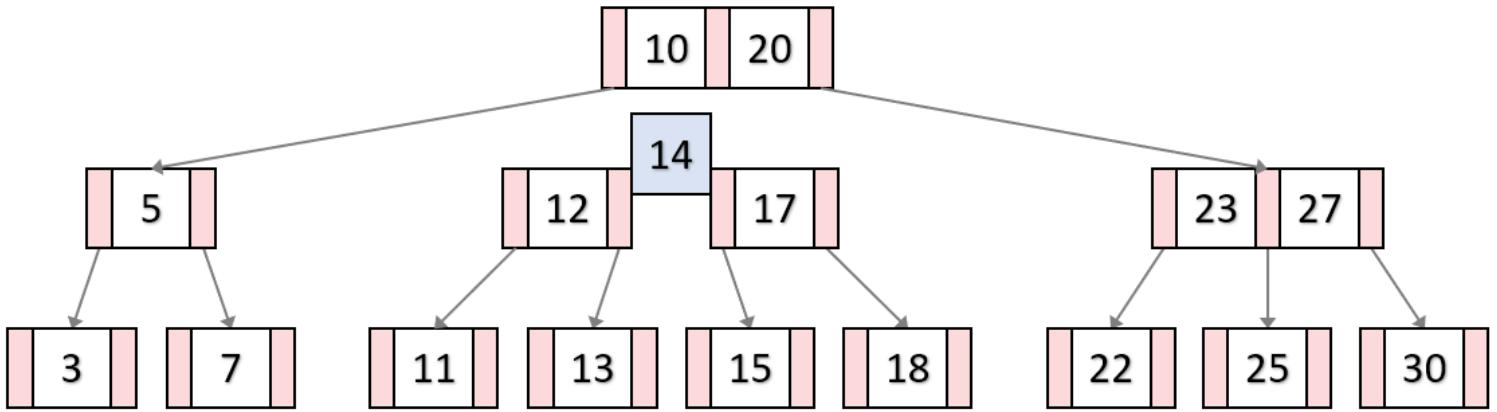
B Tree

5 12는 부모노드에 오름차순으로 삽입, 11과 13은 12의 왼쪽자식, 오른쪽 자식으로 설정



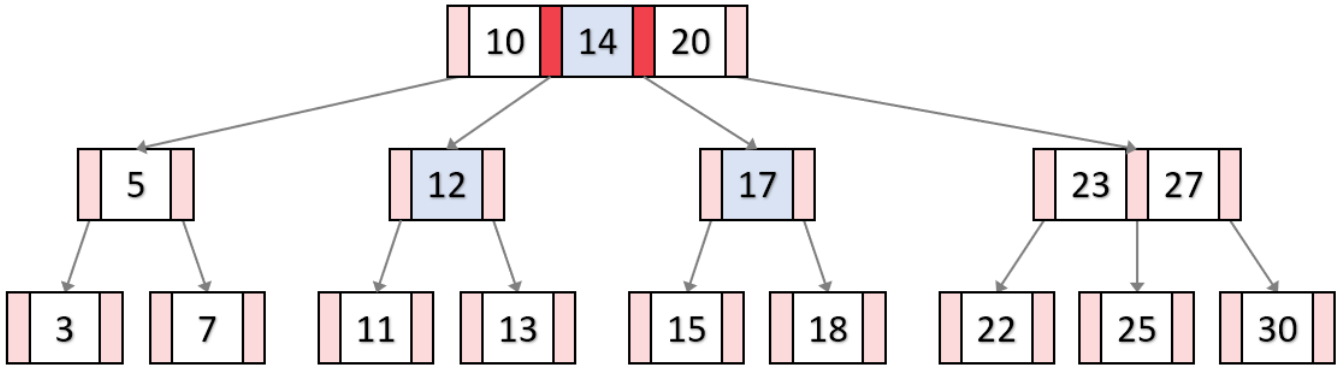
B Tree

6 12가 병합된 노드가 최대로 가질 수 있는 key의 개수를 초과, 중앙값 14를 기준으로 분할을 수행



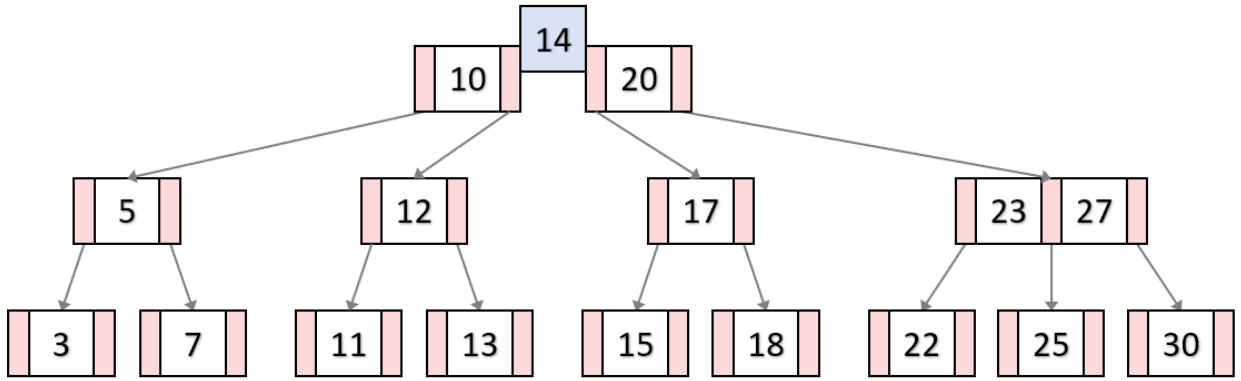
B Tree

7 14는 부모노드에 오름차순으로 삽입, 12과 17은 14의 왼쪽자식, 오른쪽 자식으로 설정



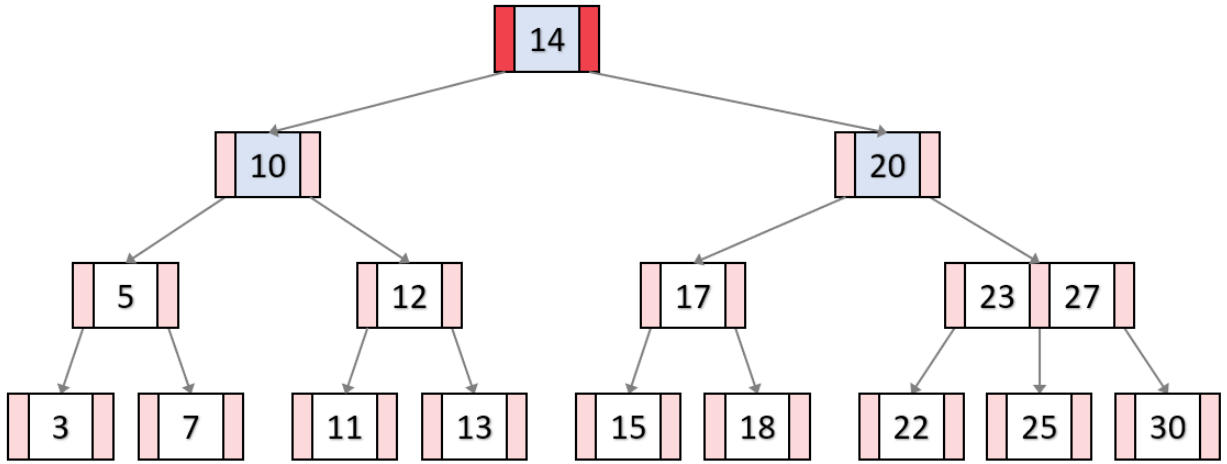
B Tree

8 14가 병합된 노드가 최대로 가질 수 있는 key의 개수를 초과, 중앙값 14를 기준으로 분할을 수행



B Tree

9 14가 새로운 루트노드가 되고, 10과 20은 14의 왼쪽자식, 오른쪽 자식으로 설정. 삽입과정 완료

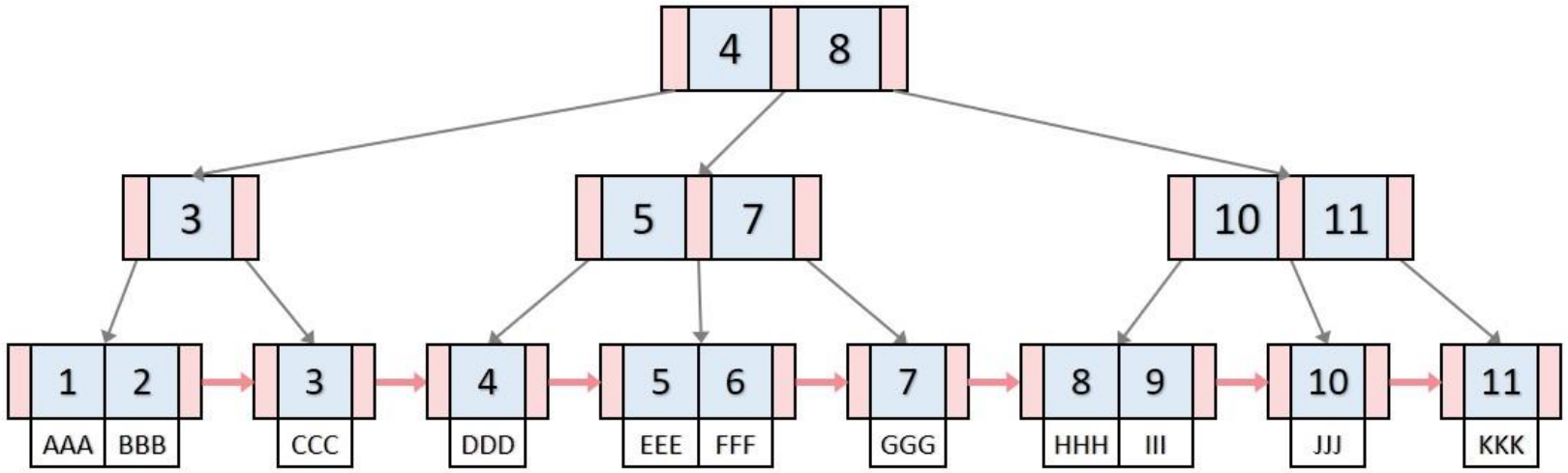


B Tree & B+ Tree

데이터를 저장하는 방식이 다른 Tree들과 다른 점이 있음

한 노드의 1개의 데이터를 저장하는 이전의 방식들과 달리 여러 개의 데이터를 저장해야 하므로 노드에 key-value 쌍으로 데이터를 저장하는 map을 갖고 있음

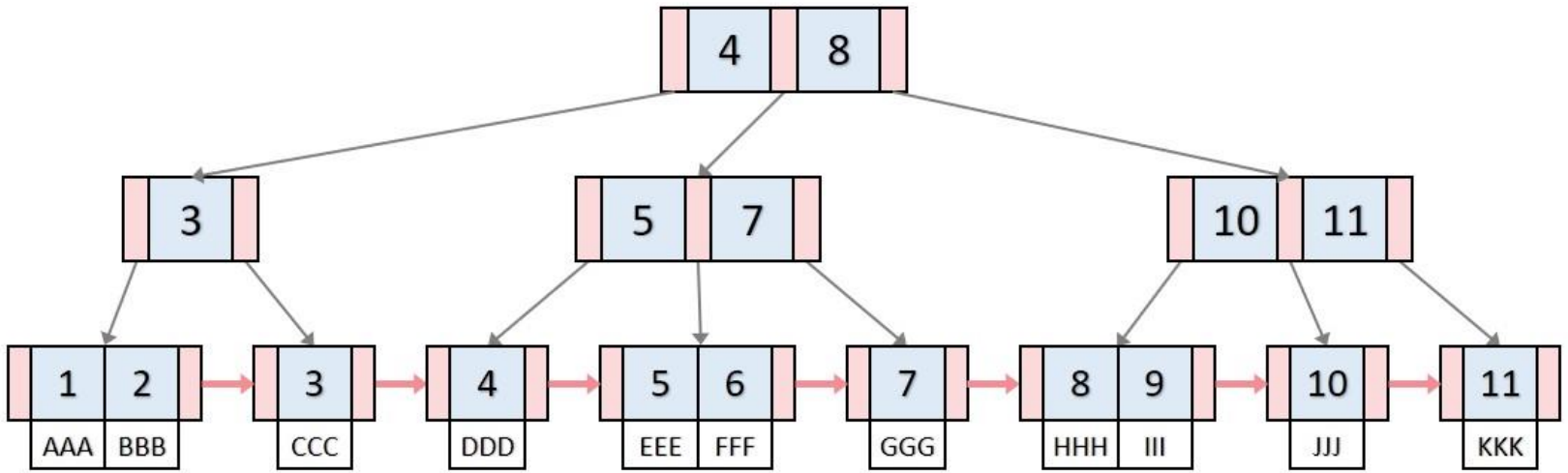
Map: {key, value} 형태로 데이터를 저장하는 컨테이너



B+ Tree

B Tree에서 데이터의 구간 Search를 용이하게 만든 트리

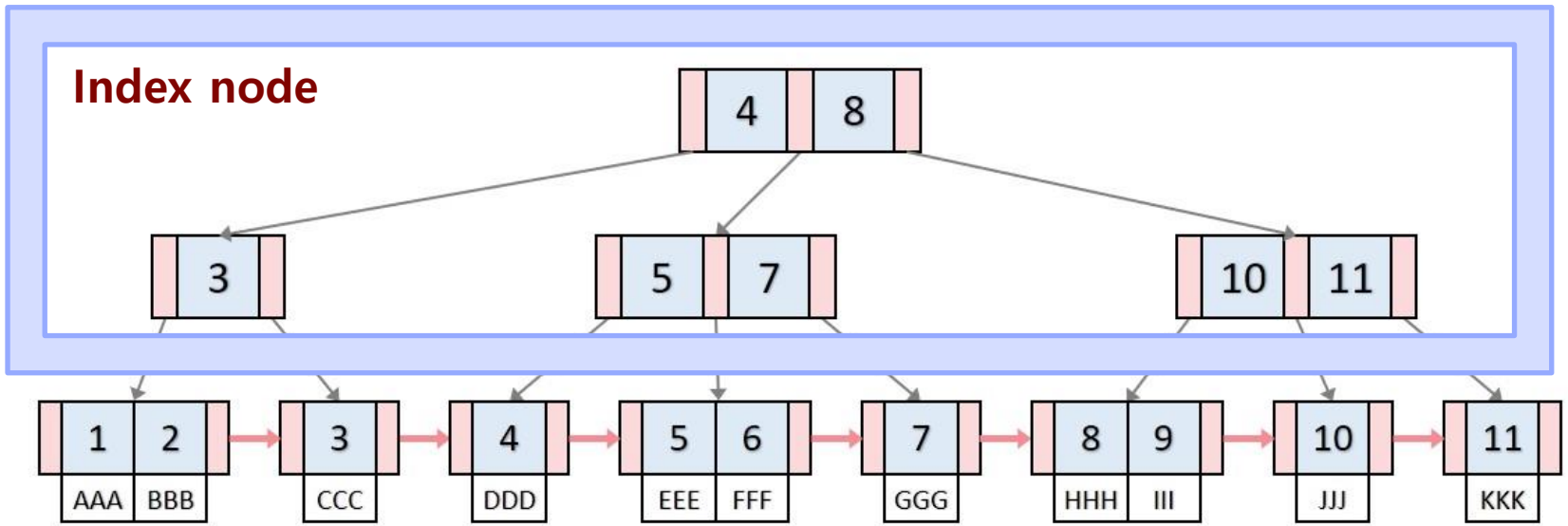
노드가 index node와 data node로 나뉨



B+ Tree

B Tree에서 데이터의 구간 Search를 용이하게 만든 트리

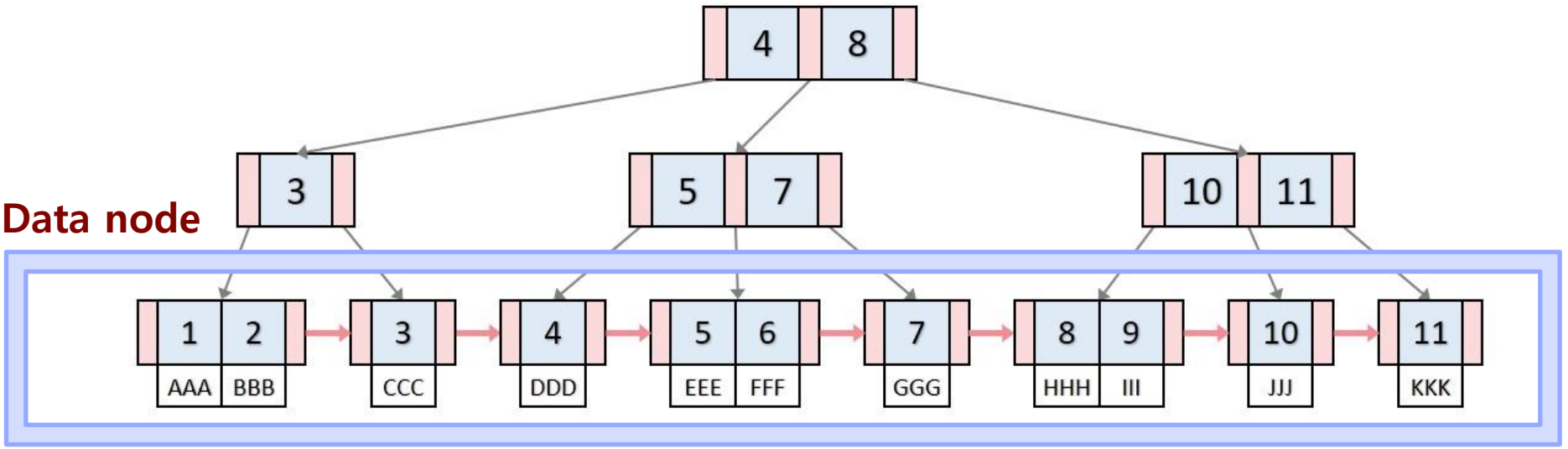
노드가 index node와 data node로 나뉨



B+ Tree

B Tree에서 데이터의 구간 Search를 용이하게 만든 트리

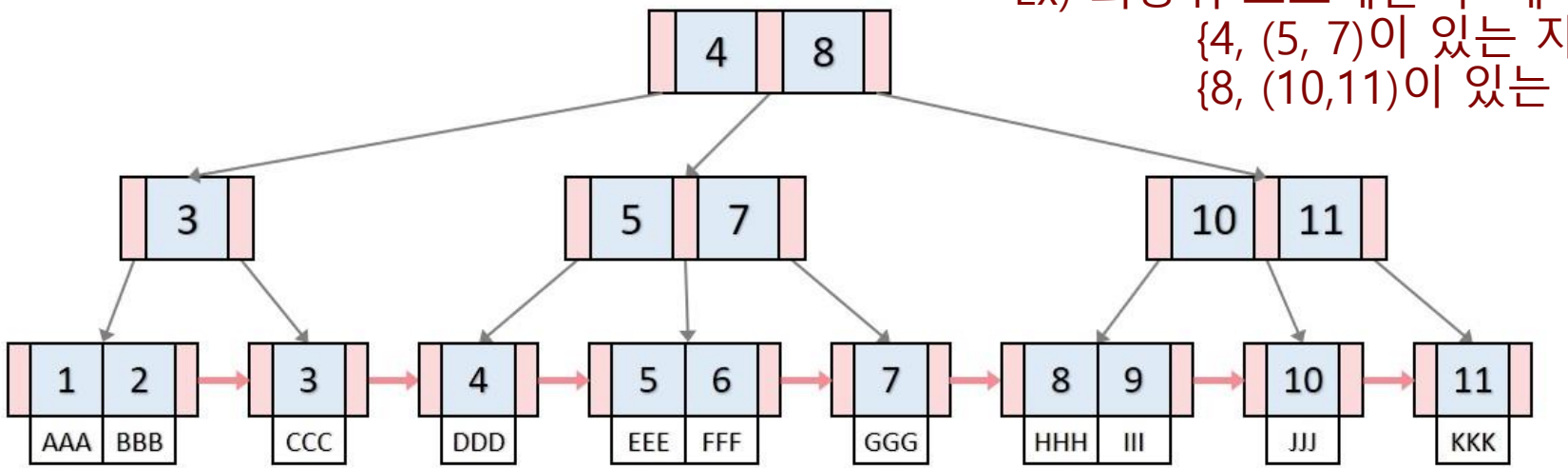
노드가 index node와 data node로 나뉨



B+ Tree

Index Node
리프 노드가 아닌 모든 노드, 실질적으로 데이터를 저장하고 있는 것이 아니고 그 데이터에 접근할 수 있는 주소를 갖고 있음.
Map에 들어갈 데이터의 Key값은 크기 비교를 할 숫자, Value 값은 자식 노드의 주소

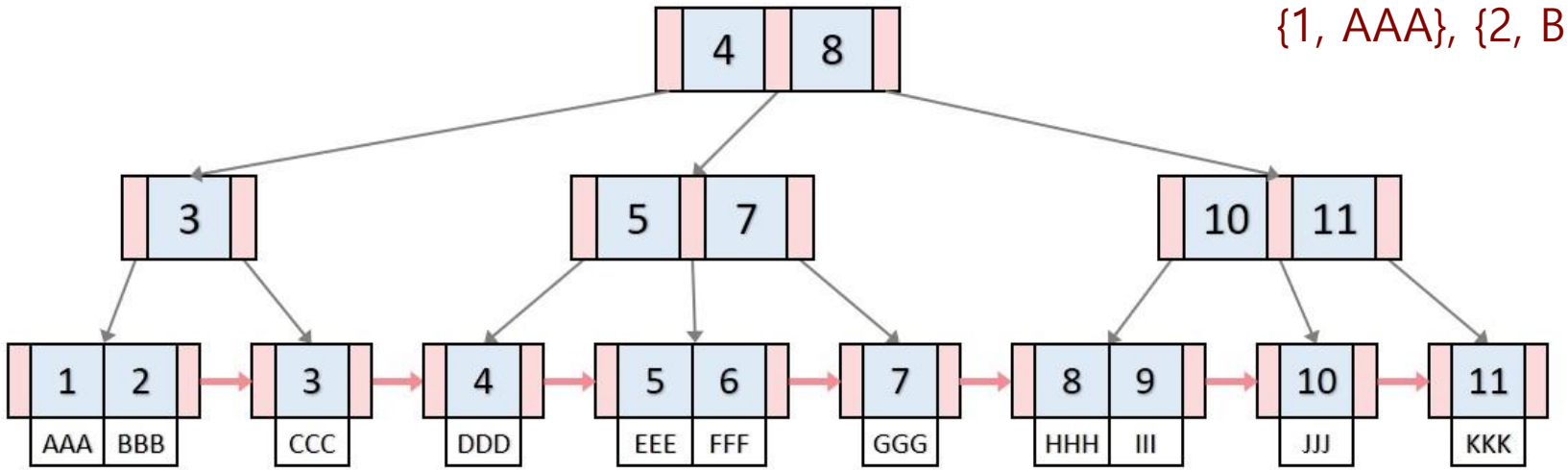
Ex) 최상위 노드에는 두 데이터 존재
{4, (5, 7)이 있는 자식 노드}
{8, (10,11)이 있는 자식 노드}



B+ Tree

Data Node
리프 노드에 있는 노드, 리프 노드에 있는 모든 데이터는 Linked List로 연결되어 있어야 함
Map에 들어갈 데이터의 Key값은 크기 비교를 할 숫자, Value 값은 실제 데이터

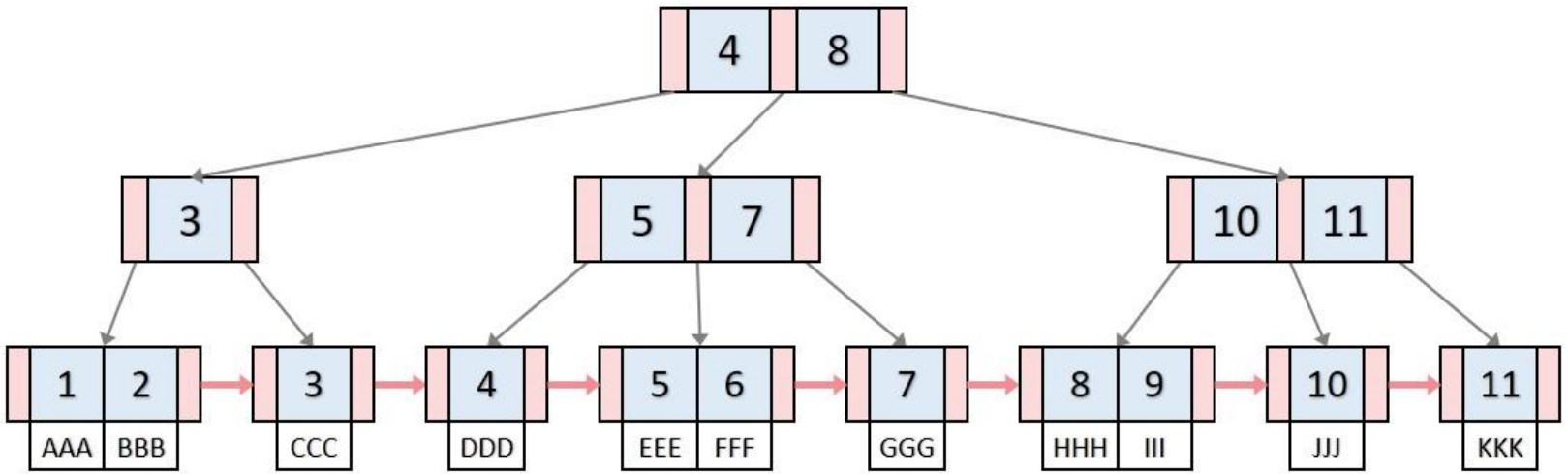
Ex) 맨 왼쪽 Leaf 노드에 두 데이터 존재
{1, AAA}, {2, BBB}



B+ Tree

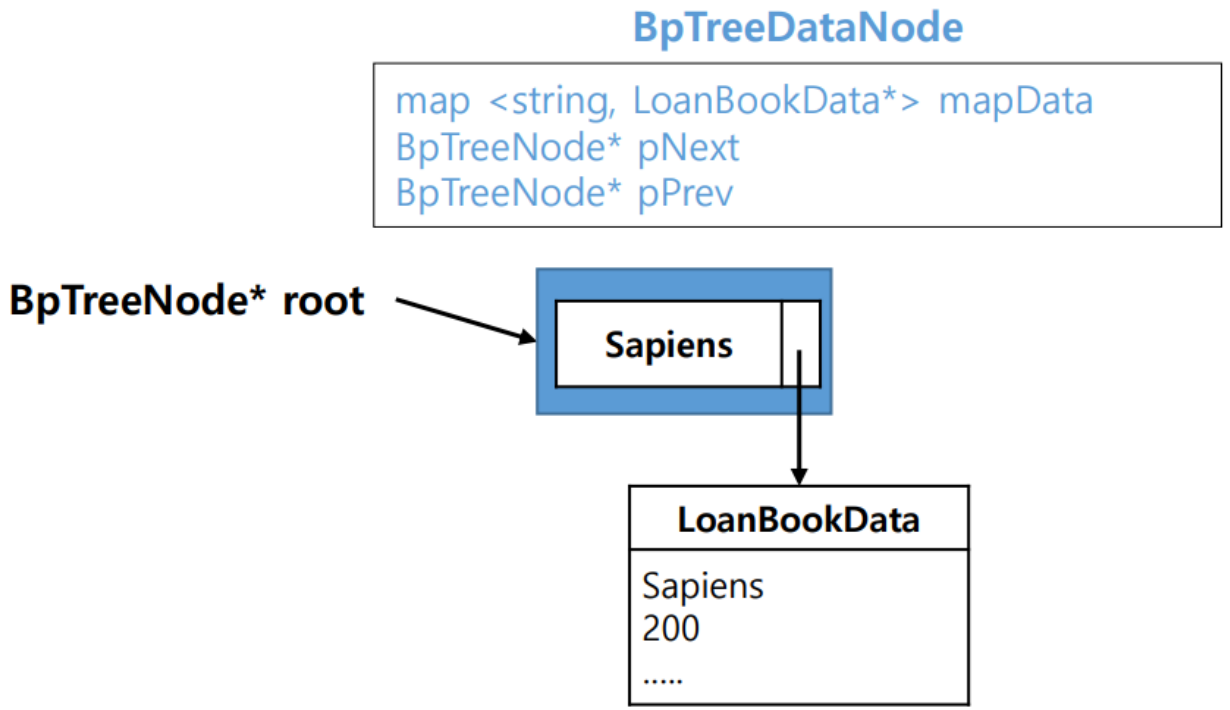
B+ Tree의 장점
B Tree는 한 데이터만 검색할 때는 효율적이지만, 모든 데이터를 순회할 때는 트리 안의 모든 노드를 순회해야 하므로 비효율적임

B+ Tree는 Data Node만 순회하면 모든 데이터를 찾을 수 있으므로 효율적임
특히, 특정 구간을 탐색할 때 효율적임

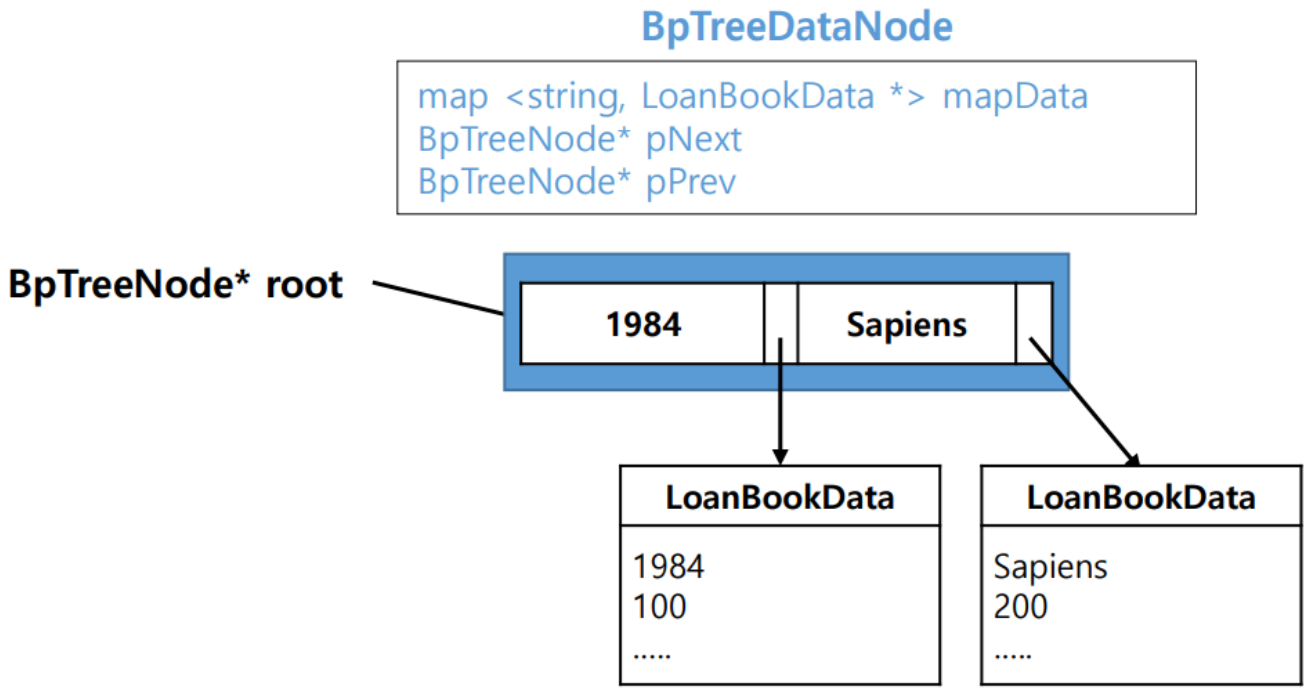


Ex) 4~8까지 데이터를 탐색하고 싶으면
4를 탐색한 후 Linked List로 이어진 데이터를 따라가서 8까지 탐색

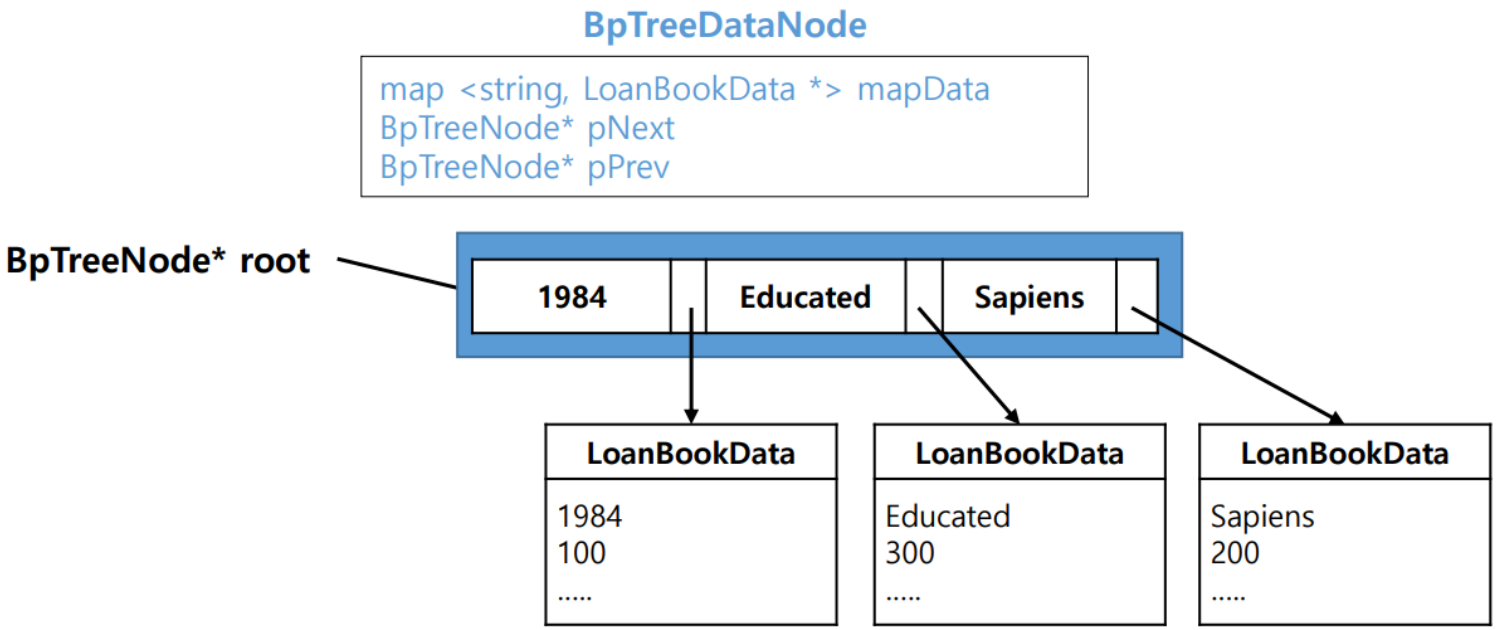
B+ Tree


BpTreeDataNode
map <string, LoanBookData*> mapData
BpTreeNode* pNext
BpTreeNode* pPrev
BpTreeNode* root
Sapiens
LoanBookData
Sapiens
200
.....

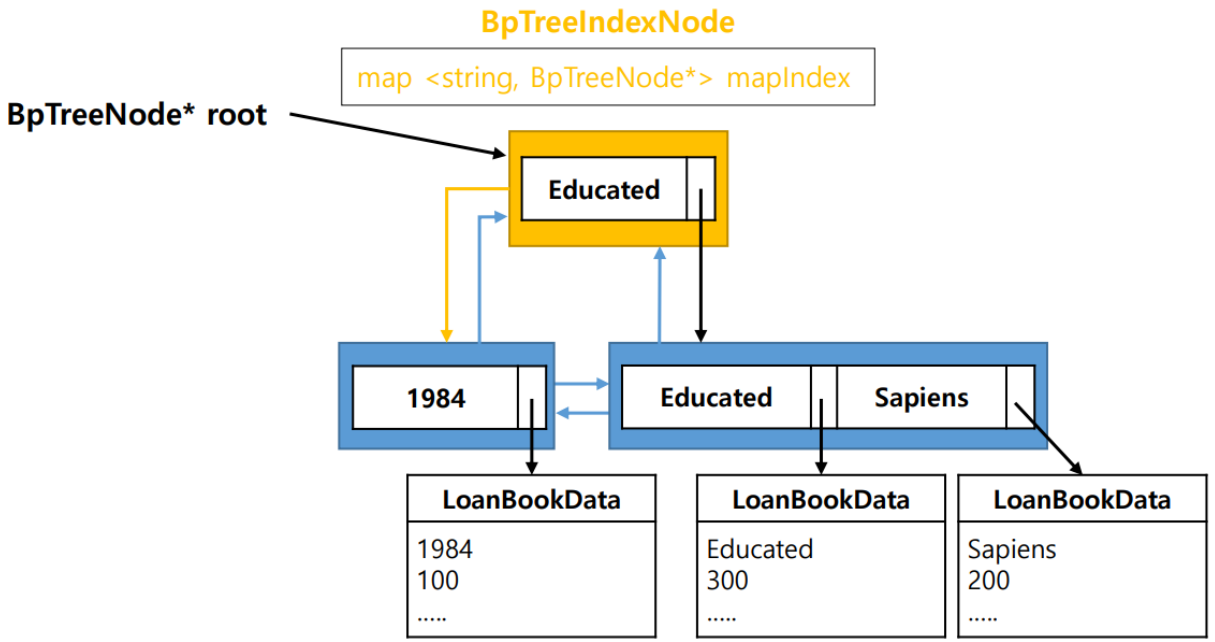
B+ Tree


BpTreeDataNode
map <string, LoanBookData *> mapData
BpTreeNode* pNext
BpTreeNode* pPrev
BpTreeNode* root
1984
Sapiens
LoanBookData
1984
100
.....
LoanBookData
Sapiens
200
.....

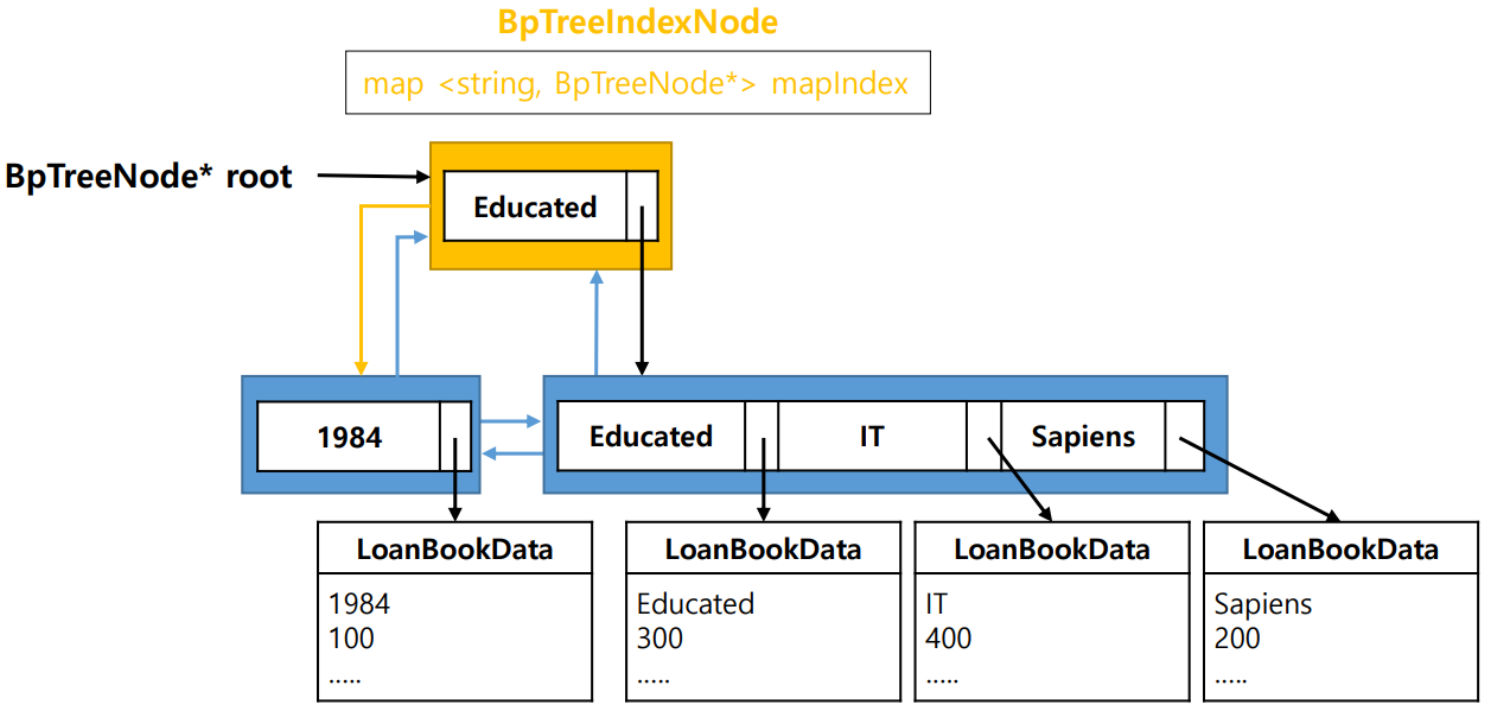
B+ Tree


BpTreeDataNode
map <string, LoanBookData *> mapData
BpTreeNode* pNext
BpTreeNode* pPrev
BpTreeNode* root
1984
Educated
Sapiens
LoanBookData
1984
100
.....
LoanBookData
Educated
300
.....
LoanBookData
Sapiens
200
.....

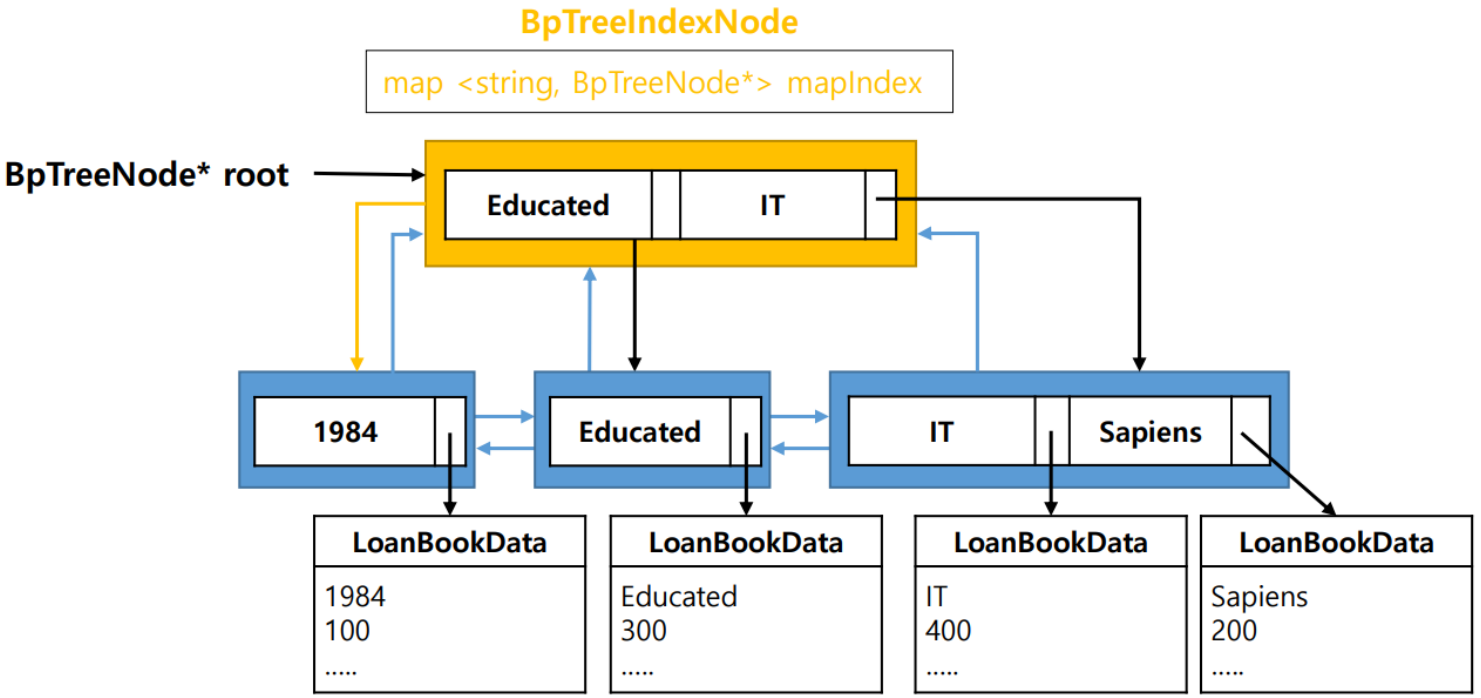
B+ Tree


BpTreeIndexNode
map <string, BpTreeNode*> mapIndex
BpTreeNode* root
Educated
1984
Educated
Sapiens
LoanBookData
1984
100
.....
LoanBookData
Educated
300
.....
LoanBookData
Sapiens
200
.....

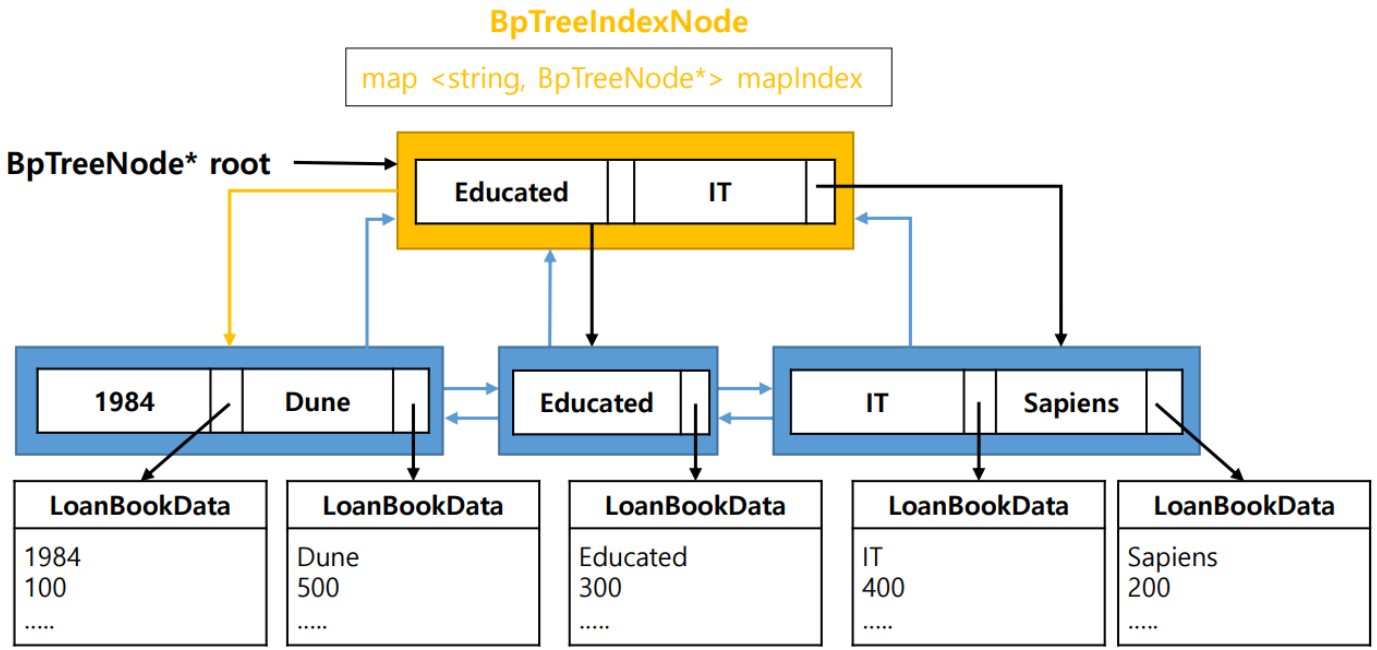
B+ Tree


BpTreeIndexNode
map <string, BpTreeNode*> mapIndex
BpTreeNode* root
Educated
1984
Educated
IT
Sapiens
LoanBookData
1984
100
.....
LoanBookData
Educated
300
.....
LoanBookData
IT
400
.....
LoanBookData
Sapiens
200
.....

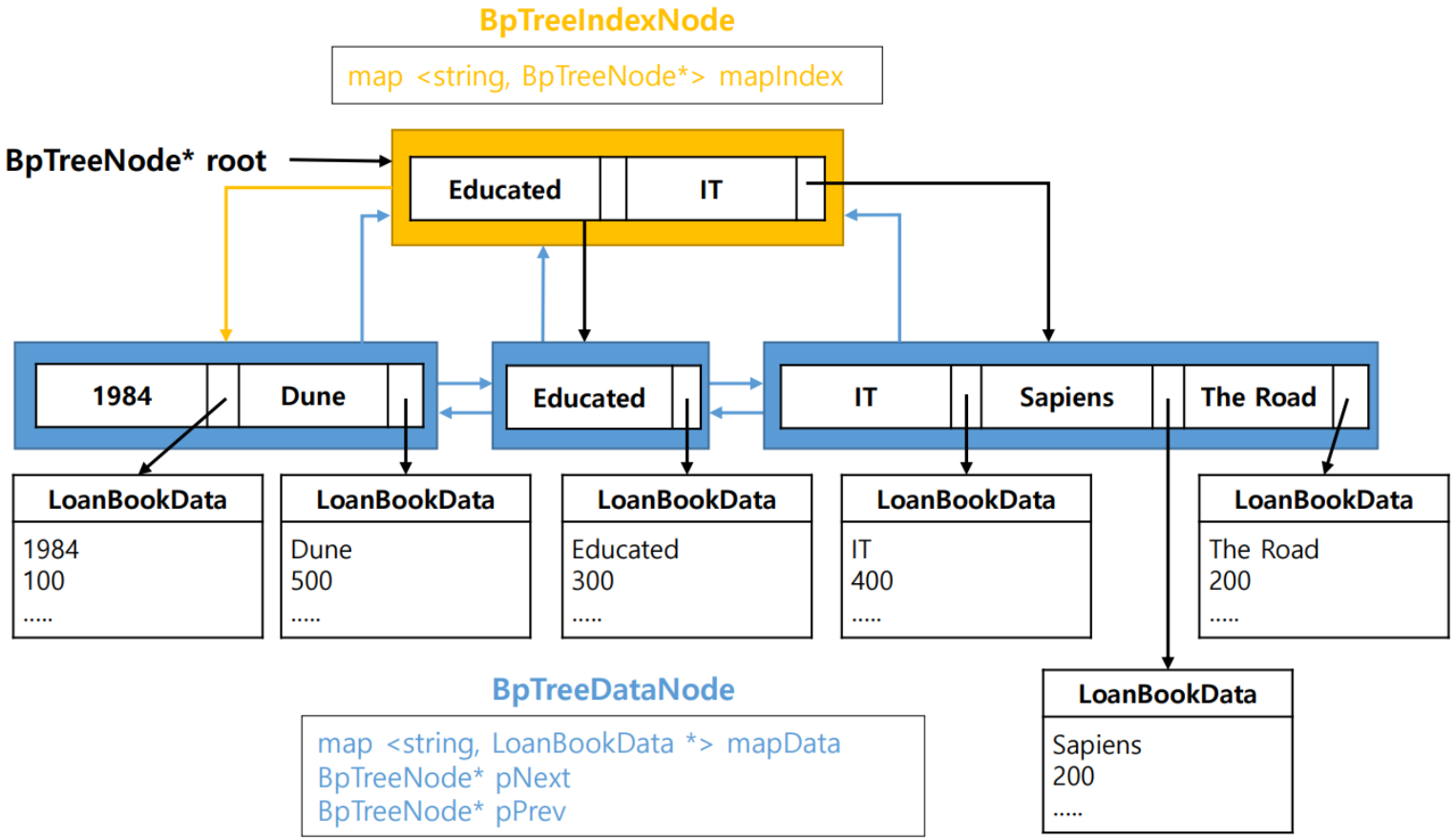
B+ Tree


BpTreeIndexNode
map <string, BpTreeNode*> mapIndex
BpTreeNode* root
Educated
IT
1984
Educated
IT
Sapiens
LoanBookData
1984
100
.....
LoanBookData
Educated
300
.....
LoanBookData
IT
400
.....
LoanBookData
Sapiens
200
.....

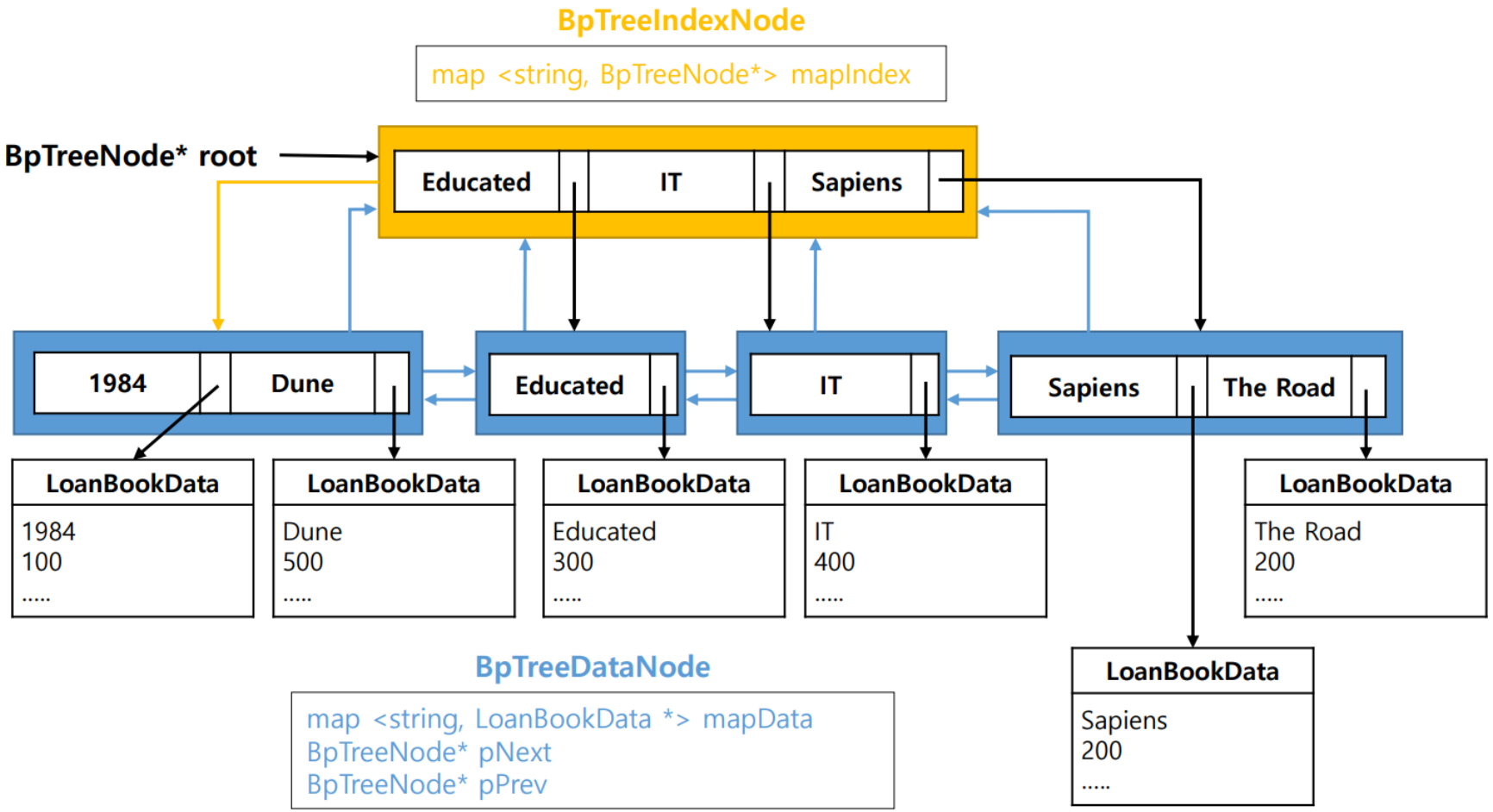
B+ Tree


BpTreeIndexNode
map <string, BpTreeNode*> mapIndex
BpTreeNode* root
Educated
IT
1984
Dune
Educated
IT
Sapiens
LoanBookData
1984
100
.....
LoanBookData
Dune
500
.....
LoanBookData
Educated
300
.....
LoanBookData
IT
400
.....
LoanBookData
Sapiens
200
.....

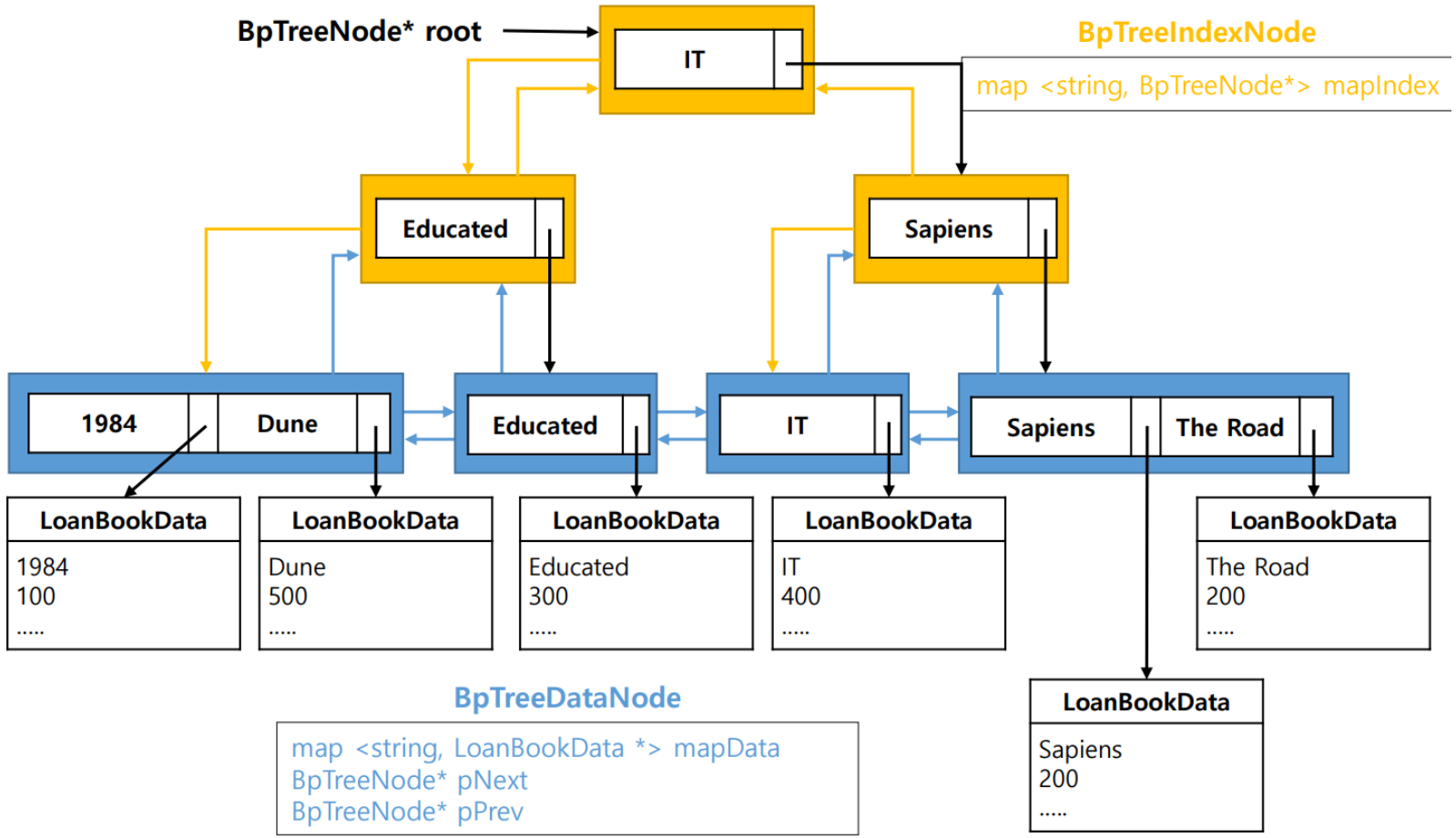
B+ Tree


BpTreeIndexNode
map <string, BpTreeNode*> mapIndex
BpTreeNode* root
Educated
IT
1984
Dune
Educated
IT
Sapiens
The Road
LoanBookData
1984
100
.....
LoanBookData
Dune
500
.....
LoanBookData
Educated
300
.....
LoanBookData
IT
400
.....
LoanBookData
The Road
200
.....
BpTreeDataNode
map <string, LoanBookData *> mapData
BpTreeNode* pNext
BpTreeNode* pPrev
LoanBookData
Sapiens
200
.....

B+ Tree


BpTreeIndexNode
map <string, BpTreeNode*> mapIndex
BpTreeNode* root
Educated
IT
Sapiens
1984
Dune
Educated
IT
Sapiens
The Road
LoanBookData
1984
100
.....
LoanBookData
Dune
500
.....
LoanBookData
Educated
300
.....
LoanBookData
IT
400
.....
LoanBookData
The Road
200
.....
LoanBookData
Sapiens
200
.....
BpTreeDataNode
map <string, LoanBookData *> mapData
BpTreeNode* pNext
BpTreeNode* pPrev

B+ Tree


BpTreeNode* root
IT
BpTreeIndexNode
map <string, BpTreeNode*> mapIndex
Educated
Sapiens
1984
Dune
Educated
IT
Sapiens
The Road
LoanBookData
1984
100
.....
LoanBookData
Dune
500
.....
LoanBookData
Educated
300
.....
LoanBookData
IT
400
.....
LoanBookData
The Road
200
.....
BpTreeDataNode
map <string, LoanBookData *> mapData
BpTreeNode* pNext
BpTreeNode* pPrev
LoanBookData
Sapiens
200
.....

End