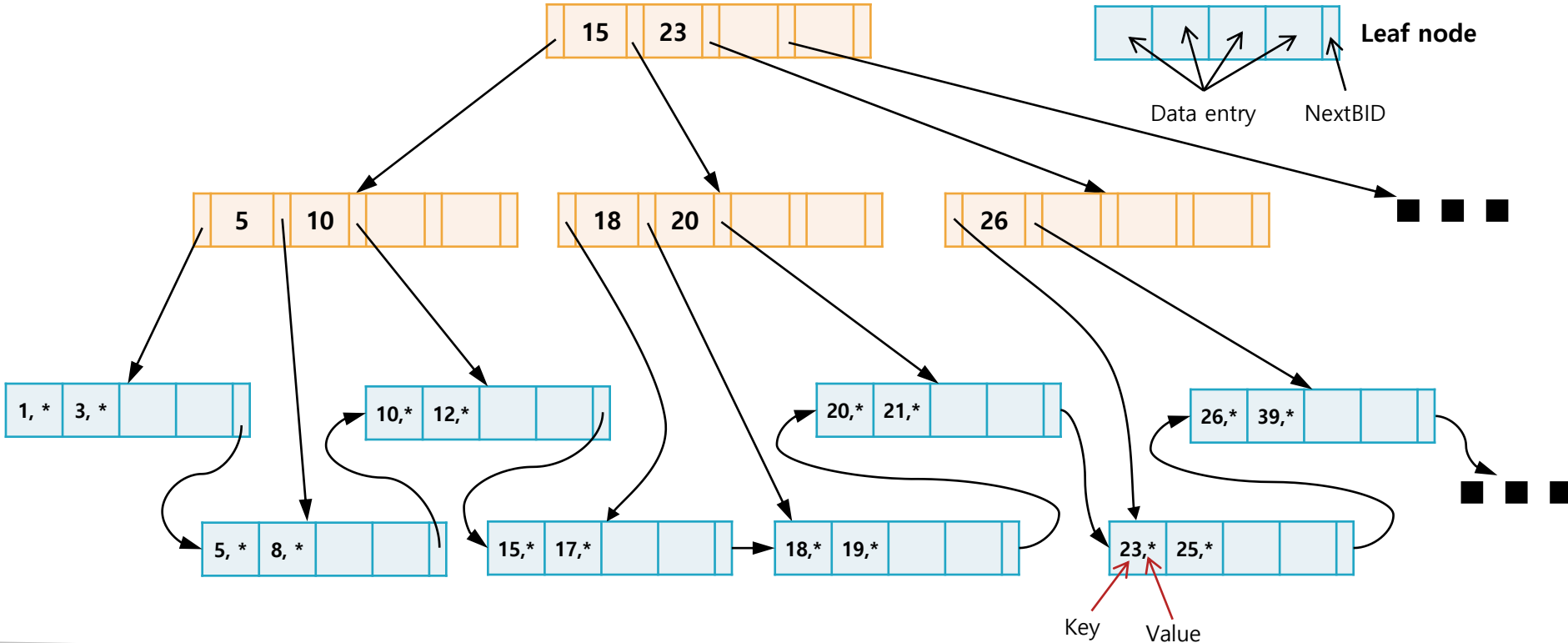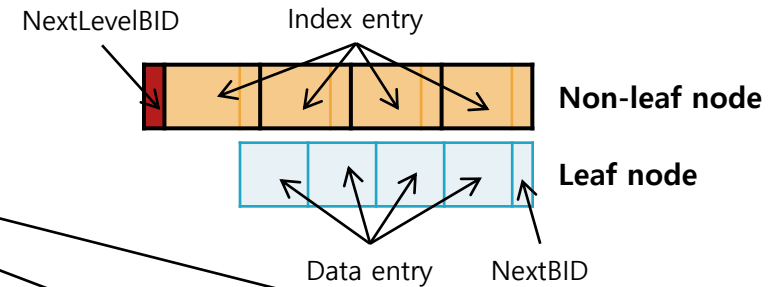# Implementation of a Disk Based B⁺-Tree

CSE3207 Database Project #2
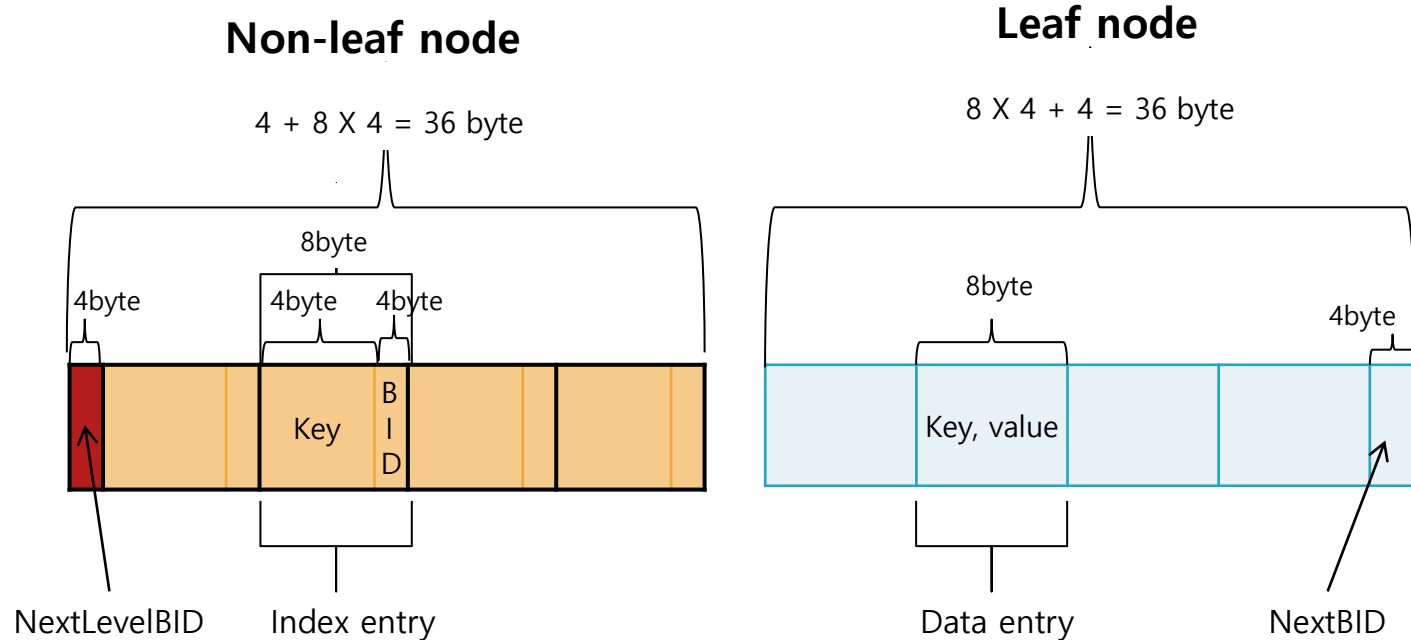
**Assignment Date : May 28th, 2020
Due Date : July 5th, 2020**

B⁺-Tree Structure

NextLevelBID   Index entry

Non-leaf node

Leaf node

Data entry   NextBID

15   23

5   10   18   20   26

1, *   3, *   10,*   12,*   20,*   21,*   26,*   39,*

5, *   8, *   15,*   17,*   18,*   19,*   23,*   25,*

Key   Value

Details of Nodes and Entries

Non-leaf node

4 + 8 X 4 = 36 byte

8byte

4byte   4byte   4byte

Key   B I D

NextLevelBID   Index entry

Leaf node

8 X 4 + 4 = 36 byte

8byte

4byte

Key, value

Data entry   NextBID

* page size = node size = 36 byte

Index entry
Key
NextLevelBID

Key   B I D
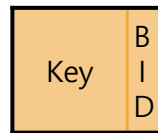
Data entry
Key
Value

Key, value
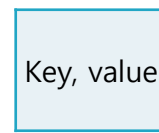
B⁺-Tree Data File Structure

Insert data (key, ID)

1	5
6	5
4	5
7	5
9	5

12 bytes

File header
`BlockSize, RootBID, Depth`

```
1   6   2

1,5  4,5        2 →  6,5  7,5  9,5
```
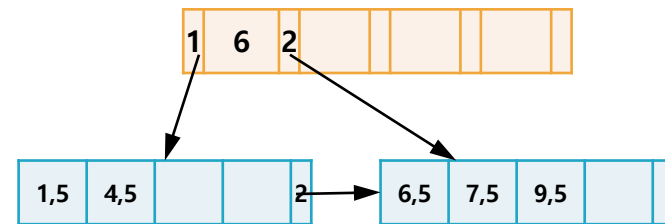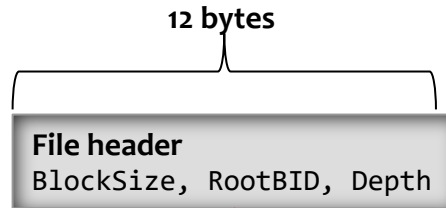
*36 bytes for each node

```
              0   1   2   3   4   5   6   7   8   9   a   b   c   d   e   f
00000000h:   24  00  00  00  03  00  00  00  01  00  00  00  01  00  00  00  ;
00000010h:   05  00  00  00  04  00  00  00  05  00  00  00  00  00  00  00  ;
00000020h:   00  00  00  00  00  00  00  00  00  00  00  00  02  00  00  00  ;
00000030h:   06  00  00  00  05  00  00  00  07  00  00  00  05  00  00  00  ;
00000040h:   09  00  00  00  05  00  00  00  00  00  00  00  00  00  00  00  ;
00000050h:   00  00  00  00  01  00  00  00  06  00  00  00  02  00  00  00  ;
00000060h:   00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  ;
00000070h:   00  00  00  00  00  00  00                                     ;
```

84

*Physical offset of a PageID = 12 + ((BID-1) * BlockSize)*

Test & UI

▸ Index creation
  ▸ *bptree.exe c [bptree binary file] [page_size]* *e.g., bptree.exe c bptree.bin 36*
    ▸ Generates [bptree binary file] with only header
▸ Insertion
  ▸ *bptree.exe i [bptree binary file] [records text file]* *e.g., bptree.exe i bptree.bin insert.txt*
    ▸ Inserts nodes(entries) to [bptree binary file] using [records text file]
▸ Point(exact) search
  ▸ *bptree.exe s [bptree binary file] [input text file] [output text file]*
                        e.g., bptree.exe s bptree.bin search.txt output.txt
    ▸ Output searched keys and IDs to [output text file] using [bptree binary file]
▸ Range search
  ▸ *bptree.exe r [bptree binary file] [input text file] [output text file]*
                        e.g., bptree.exe r bptree.bin rangesearch.txt output.txt
    ▸ Output searched keys and IDs to [output text file] using [bptree binary file]

▸ MUST follow input and output file formats in the document

Test & UI

- Print B⁺-Tree structure
  - *bptree.exe p [bptree binary file] [output text file]*
    - Output node structure of [bptree binary file] to [output text file]
    - Output only root node<level 0> and next level <level1>
    - Example
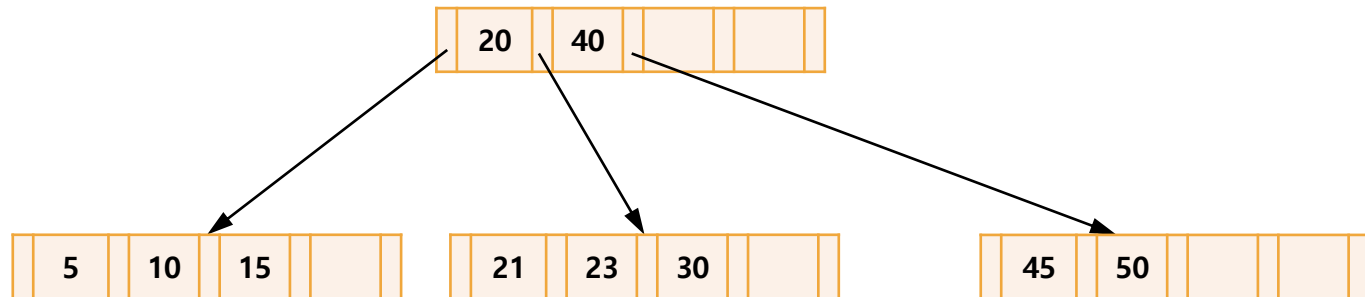

```
<0>
20, 40
<1>
5, 10, 15, 21, 23, 30, 45, 50
```

Submission

▸ To the I-Class website

▸ Upload file containing the followings:

   ▸ A single source file, named as **"bptree.cpp or bptree.c"**

   ▸ README.doc explaining:

      ▸ What you've implemented and what you've NOT

      ▸ Brief explanation of your implementation (Do not make it look fancy, Less than 0.5 page)

      ▸ How to compile and run

      ▸ Talk about your experience of doing this project

      ▸ Contact information (just in case)