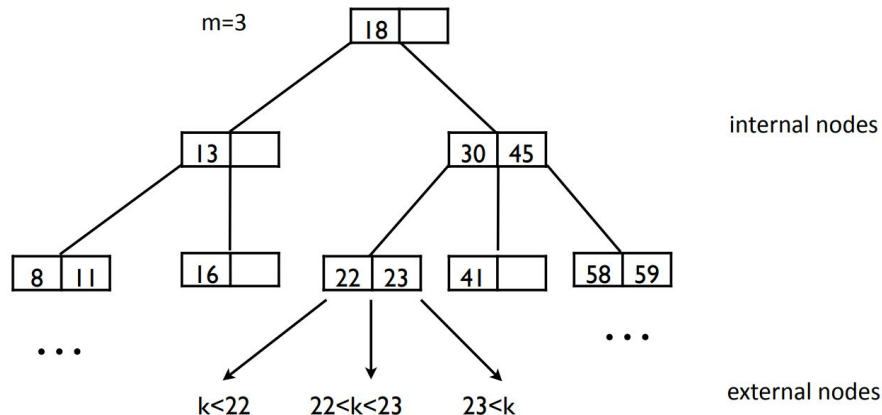
The seal of Hanyang University is a circular emblem. It features a central shield-like shape with the Korean characters '한양' (Hanyang) inside. The shield is surrounded by a wreath of leaves and flowers. The outer ring of the seal contains the text 'HANYANG UNIVERSITY' at the top and '1939' at the bottom.

Lab 09: B-Tree

Data Structure

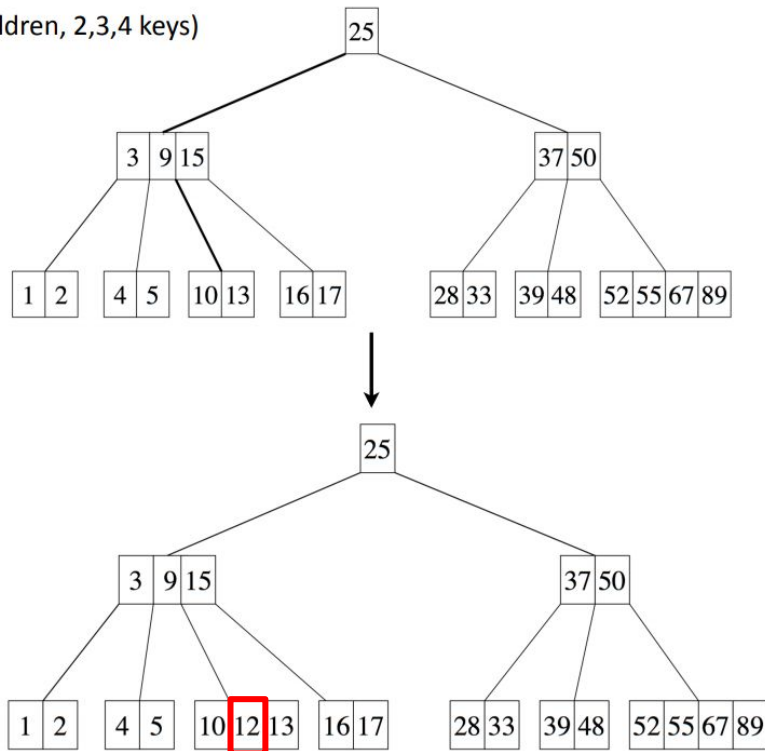
B-Tree

- A B-tree of **order m** is an m-way search tree with the following properties
 - the root has **at least 2** children
 - each node has **upto m-1** keys
 - all external nodes are at the same level (perfectly balanced)
 - all internal nodes (except the root) have **between $\lceil m/2 \rceil$ and m** children
 - when $m=3$, all internal nodes of B-tree have a degree of either 2 or 3 (2-3 tree)
 - when $m=4$, all internal nodes



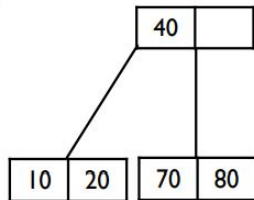
B-Tree - Insertion

m= 5 (3,4,5 children, 2,3,4 keys)
insert 12

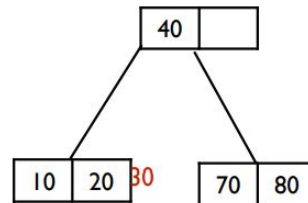


B-Tree – Node Split

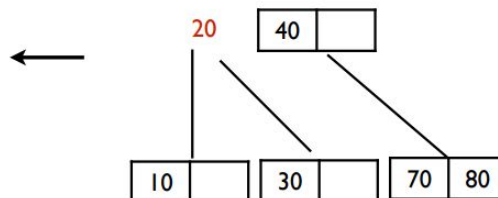
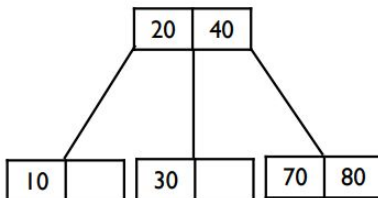
m= 3 (2,3 children, 1,2 keys)
insert 30



insert 30

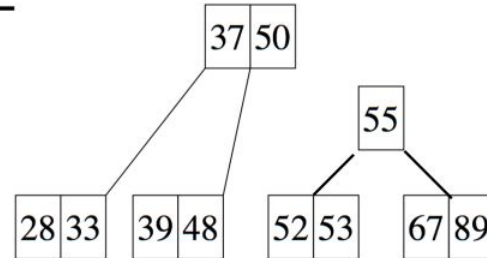
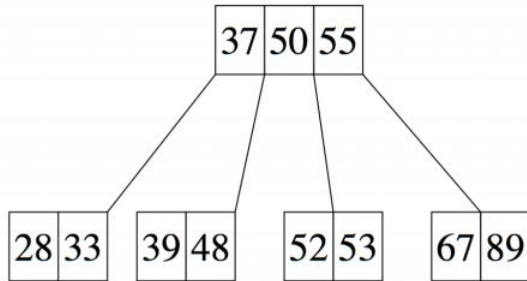
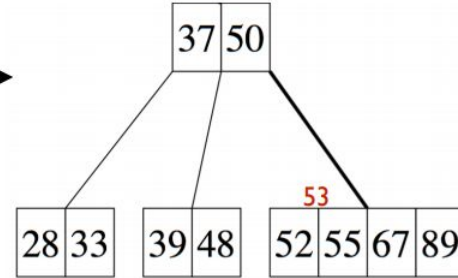
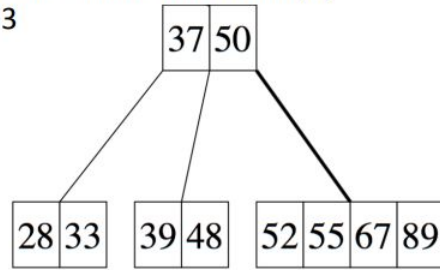


find the middle one and
push it to the parent node



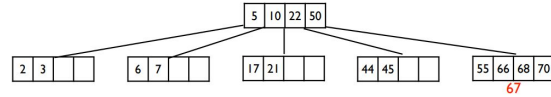
B-Tree – Node Split

m= 5 (3, 4, 5 children, 2,3,4 keys)
insert 53

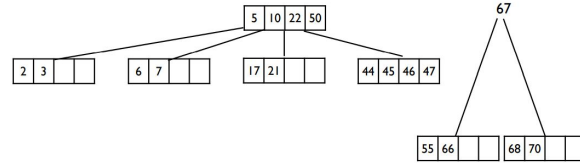


B-Tree – Node Split

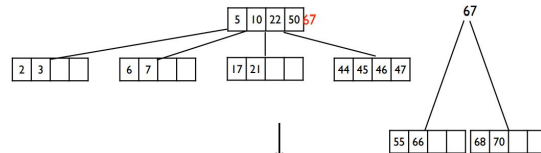
m= 5 (3, 4, 5 children, 2,3,4 keys)



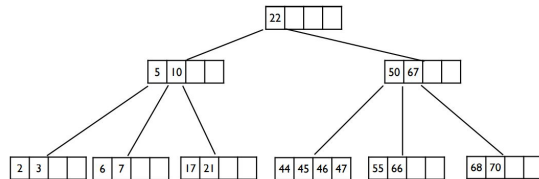
↓
insert 67



m= 5 (3, 4, 5 children, 2,3,4 keys)



↓



B-Tree ADT

- **BNodePtr CreateTree(int order)**
 - Create a B-Tree with the given order.
- **void Insert(BNodePtr root, int key)**
 - Insert a new key to the b-tree. You should find the right position for the new key to maintain the B-Tree. Print what key you inserted. You can split the node when node is full.
- **int Find(BNodePtr root, int key)**
 - Find the key in the B-Tree. Return 1 if the value exists. Otherwise, return 0.
- **void PrintTree(BNodePtr root)**
 - Print the entire tree by inorder traversal.
- **void DeleteTree(BNodePtr root)**
 - Delete the entire tree.

B-Tree ADT

- **i x**

- Insert a new key “x” into the B-Tree. **Print what key you inserted.**

- **f x**

- Find the given key to check whether the key exists in the B-Tree, and **print whether the key exists or not.**

- **p**

- **Print the entire B-Tree in inorder traversal.**

B-Tree ADT

Structure

```
typedef struct BNode* BNodePtr;
struct BNode{
    int order;
    int size      /* number of children */
    BNodePtr *child; /* children pointers */
    int *key;      /* keys */
    int is_leaf;
}BNode;
```

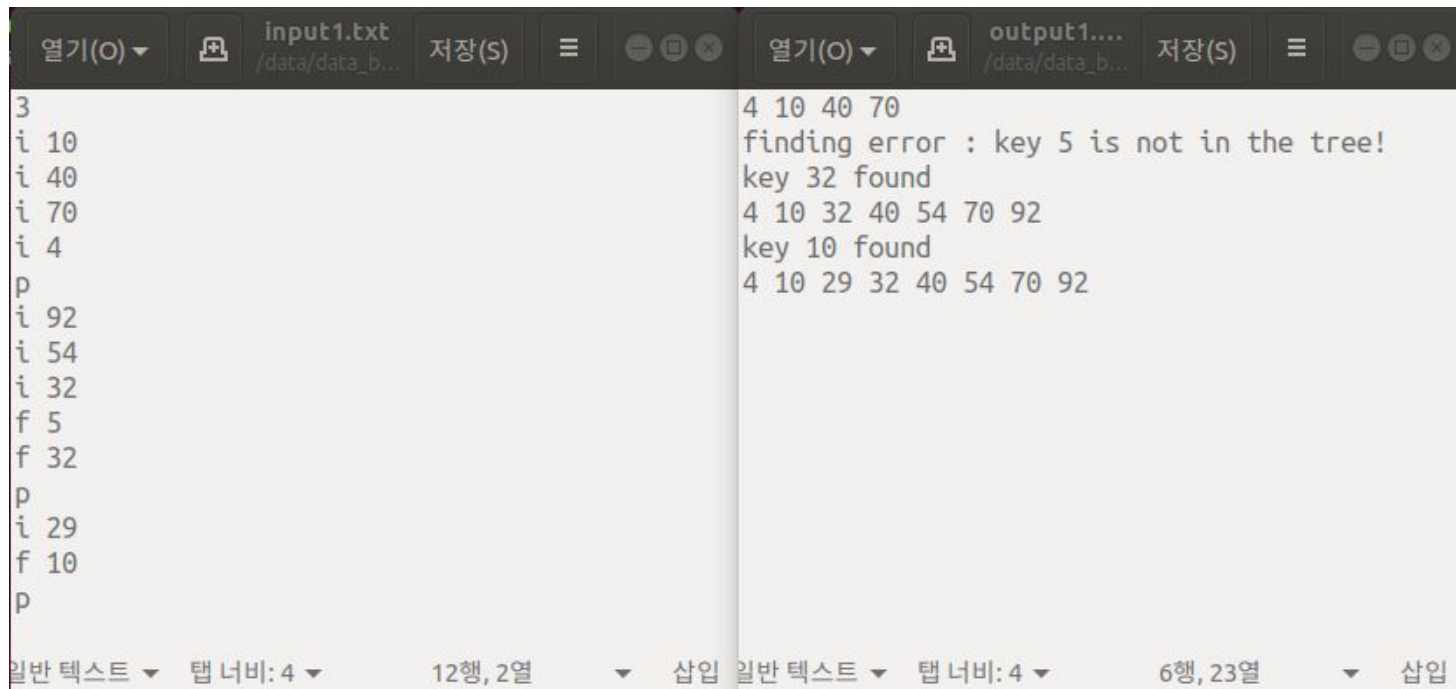
Function

```
BNodePtr CreateTree(int order);
void Insert(BNodePtr root, int key);
int Find(BNodePtr root, int key);
void PrintTree(BNodePtr root);
void DeleteTree(BNodePtr root);
```

B-Tree ADT – Skeleton Code

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  FILE *fin;
5  FILE *fout;
6
7  typedef struct BNode* BNodePtr;
8
9  struct BNode{
10     int order;
11     int size;      /* number of children */
12     BNodePtr *child; /* children pointers */
13     int *key;      /* keys */
14     int is_leaf;
15 }BNode;
16
17 BNodePtr CreateTree(int order);
18 void Insert(BNodePtr *root, int key);
19 int Find(BNodePtr root, int key);
20 void PrintTree(BNodePtr root);
21 void DeleteTree(BNodePtr root);
22
23 int main(int argc, char* argv[]){
24     fin = fopen(argv[1], "r");
25     fout = fopen(argv[2], "w");
26
27     int order;
28     fscanf(fin, "%d", &order);
29     BNodePtr root = CreateTree(order);
30
31     char cv;
32     int key;
33     while(!feof(fin)){
34         fscanf(fin, "%c", &cv);
35         switch(cv){
36             case 'i':
37                 fscanf(fin, "%d", &key);
38                 if(Find(root, key))
39                     fprintf(fout, "insert error: key %d is already in the tree!\n", key);
39                 else
40                     Insert(&root, key);
41                 break;
42             case 'f':
43                 fscanf(fin, "%d", &key);
44                 if(Find(root, key))
45                     fprintf(fout, "key %d found!\n", key);
46                 else
47                     fprintf(fout, "finding error: key %d is not in the tree!\n", key);
48                 break;
49             case 'p':
50                 if (root->size == 1)
51                     fprintf(fout, "print error: tree is empty!");
52                 else
53                     PrintTree(root);
54                 fprintf(fout, "\n");
55                 break;
56             }
57     }
58
59     DeleteTree(root);
60     fclose(fin);
61     fclose(fout);
62
63     return 0;
64 }
```

Input & Output Example



The image shows a code editor with two side-by-side windows. The left window is titled 'input1.txt' and contains the following text:

```
3
i 10
i 40
i 70
i 4
p
i 92
i 54
i 32
f 5
f 32
p
i 29
f 10
p
```

The right window is titled 'output1....' and contains the following text:

```
4 10 40 70
finding error : key 5 is not in the tree!
key 32 found
4 10 32 40 54 70 92
key 10 found
4 10 29 32 40 54 70 92
```

At the bottom of the editor, there are status bars for both windows. The left status bar shows '일반 텍스트' (Plain Text), '탭 너비: 4' (Tab width: 4), '12행, 2열' (12 lines, 2 columns), and '삽입' (Insert). The right status bar shows '일반 텍스트' (Plain Text), '탭 너비: 4' (Tab width: 4), '6행, 23열' (6 lines, 23 columns), and '삽입' (Insert).

Assignment

- Due
 - ~ **2023.05.17(수) 23:59**
 - Last Commit 기준

- 자세한 내용은 과제 명세 PDF 파일 참고