# COSC2436 Hw4: B-Trees

Created by Kunpeng Zhang (kzhang21@uh.edu)

## 1. Introduction

You will create a C++ program that can build a B-Tree. You need to use the given integers to make a B-Tree and traverse or retrieve the data required by the commands. Step by step, build your B-Tree; refer to here.

https://www.cs.usfca.edu/~galles/visualization/BTree.html

## 2. Input and Output

The inputs are regular text files.

1. Two files will be given.
    a) An input data file contains all the nodes of the tree. You need to create the B-Tree by adding the nodes to the tree.
    b) A command file contains the commands, including "Degree=n"(n is an integer greater than or equal to 3, refer to the textbook for the meaning of this "Degree"), "Inorder Traversal", "Level m" (commands are case sensitive). The m in "Level m" is a number representing the tree's level; the level starts from 1, which is the tree's root.
2. Your output file contains the results of finished commands. If no result is available for specific commands, you should output "empty".

All files should be processed sequentially from beginning to end. So, the tree should be built according to the sequence of the input values.

## 3. Program specification and Examples

Your source code will be compiled and tested by the TAs. The results should be written to another text file (output file), provided with the command line. Notice the input and output files are specified in the command line, not inside the C++ code. When calling your program, all the necessary parameters will be in the command line, so you don't need to worry about the missing parameter problem. When calling your program, the command format would be one of the two standard types as below. Also, notice the quotes in the program call.

The general call to the executable is as follows:

bitree "input=input41.txt;command=command41.txt;output=output41.txt"

Call example with another command line type.

bitree input=input41.txt command=command41.txt output=output41.txt

both types may be used simultaneously.

Assumptions:

All data are integers for the input data file; no characters other than digits will be given. No empty file will be provided. All integers will be separated with space characters. All integers are positive, and without sign and leading zero. When you build the B-Tree, no-repeat integers appear in the B-Tree (so if a value is already

inserted, it should be omitted for the repeat appearance, notice this part is different from the link https://www.cs.usfca.edu/~galles/visualization/BTree.html, your code should follow the requirements in this assumptions first). The data may appear in more than one line. Also, you need to eliminate empty line(s).

- The total node number of the B-Tree will not exceed 1000.
- For the command file, you need to process commands one by one. One line only has one command. No empty file will be given. All command files contain the "Degree=n" command. There will be no error command, but "Level m" may request a non-existed level; for example, the "Level 5" command applied to a 4-level tree, the result should be "empty" (case sensitive). There may have empty lines between commands.

**Example 1 of input and output**

**input41.txt**
**2 1 3**

**command41.txt**
**Degree=3**
**Inorder Traversal**
**Level 2**

**Command-line:**
**calculate input=input41.txt command=command41.txt output=output41.txt**

**output41.txt**
**1 2 3**
**1 3**

**Example 2 of input and output**

**input42.txt**
**3 10 15 23 65 85 235 457 51 9 2 1**
**235 457 51 9 2 1**

**command42.txt**
**Degree=4**
**Level 3**

**Command-line:**
**calculate input=input42.txt command=command42.txt output=output42.txt**

**output42.txt**
**1 3 9 15 51 65 235 457**

**Example 3 of input and output**

**input43.txt**
**2 6 8 45 21 63 85 55 14 16 9 3 4 7 2 55 11 13 654 214 9**

**command43.txt**
**Degree=3**
**Level 20**
**Level 1**
**Level 2**
**Level 3**
**Level 4**
**Level 5**

**Command-line:**
**calculate input=input43.txt command=command43.txt output=output43.txt**

**output43.txt**
**empty**
**11**
**6 21**
**3 8 14 63 214**
**2 4 7 9 13 16 45 55 85 654**
**empty**

## 4. Turn in your homework

Your program should output results within 2 seconds after will be killed by the

system.

Make sure to create a folder under your root directory, name it hw4 (name need to be lower case), only copy your .cpp and .h files to this folder, no test case.

Homework is individual. Your homework will be automatically screened for code plagiarism against code from the other students and code from external sources. Code that is copied from another student (for instance, renaming variables, changing for and while loops, changing indentation, etc.) will be detected and result in "0" in this homework. The limit is 50% similarity. Here is some previous homework that has been found to copy each other (the main function has been deleted).

ps. This document may have typos, if you think something is illogical, please email TAs for confirmation.