ECE 309 Spring 2024

Program 2: Binary search tree

Due Date: March 22th 2024 (11:59PM)

1- Assignment Learning Objectives:

- Binary search tree (BST) implementation.
- Understanding tree traversal and using it to complete the task.
- Solving problems in an object-oriented manner.

2- Associated Files:

- Assignment description (this document)
- Test input files (e.g., request.txt)
- Tree file (tree.txt)

Both files will be on Moodle under (Programming 2 Assignment)

3- Description:

In this program, first you need to create a class for binary search tree and use it to store data tree.txt, which includes 9 distinct digit numbers in a given order (4,2,1,3,7,6,5,9,8 see sample file). As a start, you will read the tree.txt file and take the first item, which is 4, as the root node. By calling insert method to the next item in the file, you add its children nodes and create a binary search tree. Your tree structure should be the same as in Figure 1.

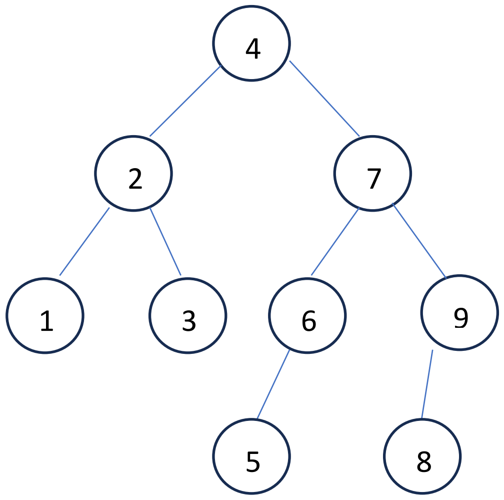


Figure 1 binary search tree for storing data in tree.txt.

As you can see, Figure 1 shows the tree structure vertically, with the root on the top, each child is lower than their parent, and the leaf at the bottom. Let's say we'd like to display it in a horizontal way i.e. the root is on the left, each child is on their parent's right, and leaf is on the right, as shown in Figure 2.

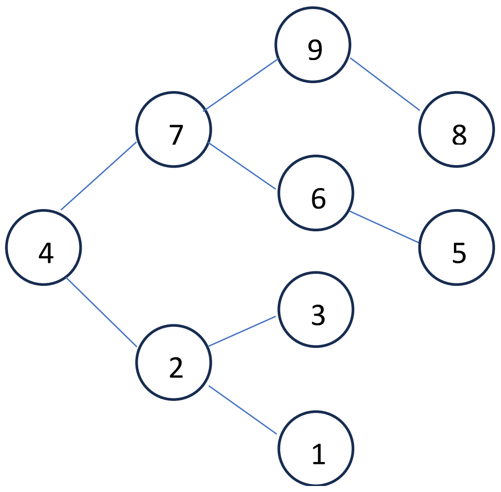


Figure 2: demonstration for representing a binary search tree horizontally.

Therefore, for second task, you are asked to implement a method in your binary search tree class called "horizon_display" that will print out the structure of the BST horizontally. Specifically, since the tree contains 9 numbers, it is expected to print out 9 line and each line contains only 1 number. Moreover, since the tree has height 3, it is expected to have 4 columns and every node with the same level should belong to same column. To space out the tree properly, please set up 5 spaces between tree levels. You can refer the left-hand side of Figure 3 as the correct format of the output when running your "horizon_display".

```
                    9
                              8
          7
                    6
                              5
4
                    3
          2
                    1
```

Figure 3: demonstration for output of horizon_display.

To complete this task, you might want to consider a similar in-ordering tree traversal that visits in the following way: right-subtree, node, left-subtree, and refer Appendix to get the idea to implement it .

Third, like Prog 1, request.txt is the file containing requests that need to be searched in BST, and you need to count the total number of nodes it has visited after searching all requests in this file.

Lastly, you are asked to create another BST with a different structure that is based on count the frequency for each request in request.txt. To do that, you need to sort these 9 digit numbers in a decreasing order with respect to their frequency in the request file, i.e. if 6 has more frequency than 2 in the request file, then 6 will come before 2. In other word, the root node contains the digit that has the highest frequency in request file. After that, please compare their total number of nodes they have visited after searching all the request in the request file and present in the report.

To test your implementation, your program should expect two input parameters, the first input is the filename including tree data, which your program will initially create a BST to store those data, and the second input is the name of the input file including the requests to search in the binary search tree you just created. Thus, to execute your program it takes the following form:

./program <tree file> <input filename>

For example:

./program tree.txt request.txt

The program that you will implement shall do the following:

1- Parse the input parameters for tree file and request file
2- The program create BST based the tree file.
3- Implement horizon_display to show its structure.
4- Derive the frequency of each number in the request file. Sort these 9 numbers based on their frequency in a descending order, and create another BST based on this order.
5- Iterate over each item in request.txt to process the requests; each line is an independent request for a particular item in BST, to be identified through the item value.
6- After processing each request in (4), you will search each value from request file in the BST you just created and count the number of nodes you have visited. Unlike program 1, you do not change any internal structure in BST.
7- Move to the next request.
8- Repeat 5-6 until the end of the request.txt file.

To understand the impact of a certain heuristic, your program should print out 2 integer value, which represent the total number of nodes you have visited for searching all item in each BST structure. In your report, you will compare the number of steps for these 2 BST.

4- <u>Report Format:</u>

The report should include:

- Print out the frequency for each item in request file.

	1	2	3	4	5	6	7	8	9
Frequency									

- The output of the horizon_display for these 2 BST.
- comparison of total number of steps for each request file

BST structure	request.txt
tree.txt	
Frequency based	

Please submit your report in PDF format with file name "report.pdf".
Kindly remind: you can assume TA will run your code in the same directory where the file is stored, so don't include absolute path to read files in the main.cpp you submitted.

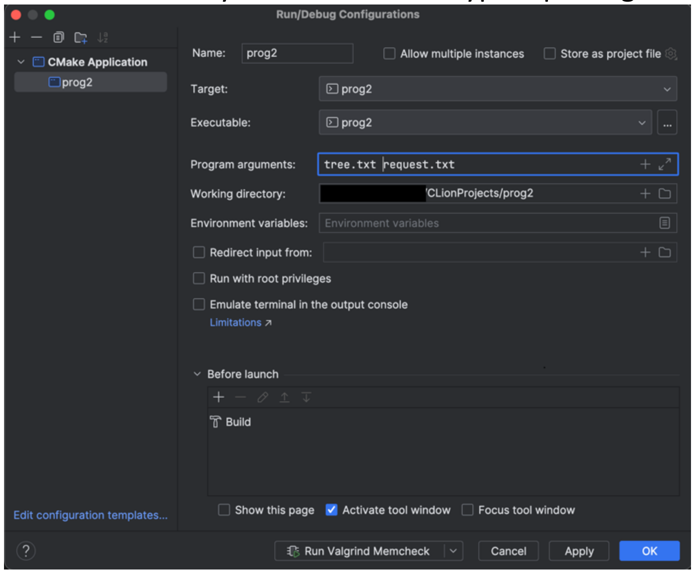
5- <u>Submission Format:</u>

main.cpp and report.pdf.

6- <u>Tips for developing program in CLion:</u>

1. For students who are not familiar with CLion, you can first create the project. (For example: prog2)

2. Since our program will take 2 inputs, you can set up the inputs (also the working directory) whenever you run it in CLion so that you don't have to type input argument all the time.

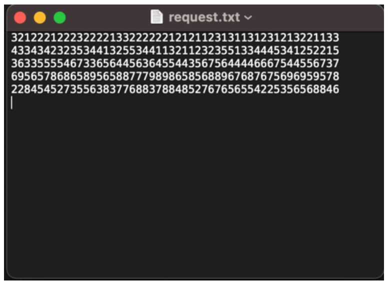


Name: prog2
Allow multiple instances
Store as project file
Target: prog2
Executable: prog2
Program arguments: tree.txt request.txt
Working directory: [redacted]CLionProjects/prog2
Environment variables: Environment variables
Redirect input from:
Run with root privileges
Emulate terminal in the output console
Limitations
Show this page
[✓ Activate tool window]
Focus tool window

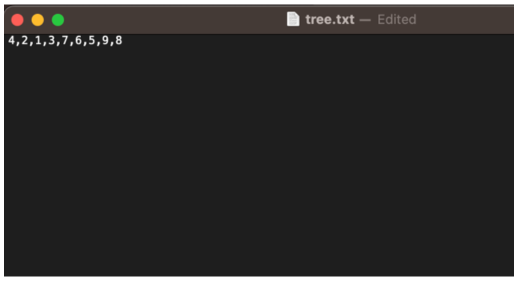
Go to "Run" menu and select "Edit Configuration."

3. By setting this configuration, you can easily test your code by clicking RUN in the toolbar.

7- Sample file for input:



```
3212221222322221332222221212112313113123121322 1133
```



```
4,2,1,3,7,6,5,9,8
```

Sample file for request.txt *Sample file for tree.txt*

8- Grading:

10 points: Files submitted with proper name.
10 points: Program is complete. The program successfully takes 2 inputs and returns the integer as output.
10 points: Proper coding style and comments.
60 points: the program performs all functions correctly:
(15 pts) Create a class named BST and implement insert method to create BST based on tree.txt.
(15 pts) Correctly implement for horizon_display and print out the each BST structure in your program
(20 pts) Correct frequency for each item in request file and create another BST based on the frequency.
(10 pts) Correct results for request files.
10 points: Report.

Good luck