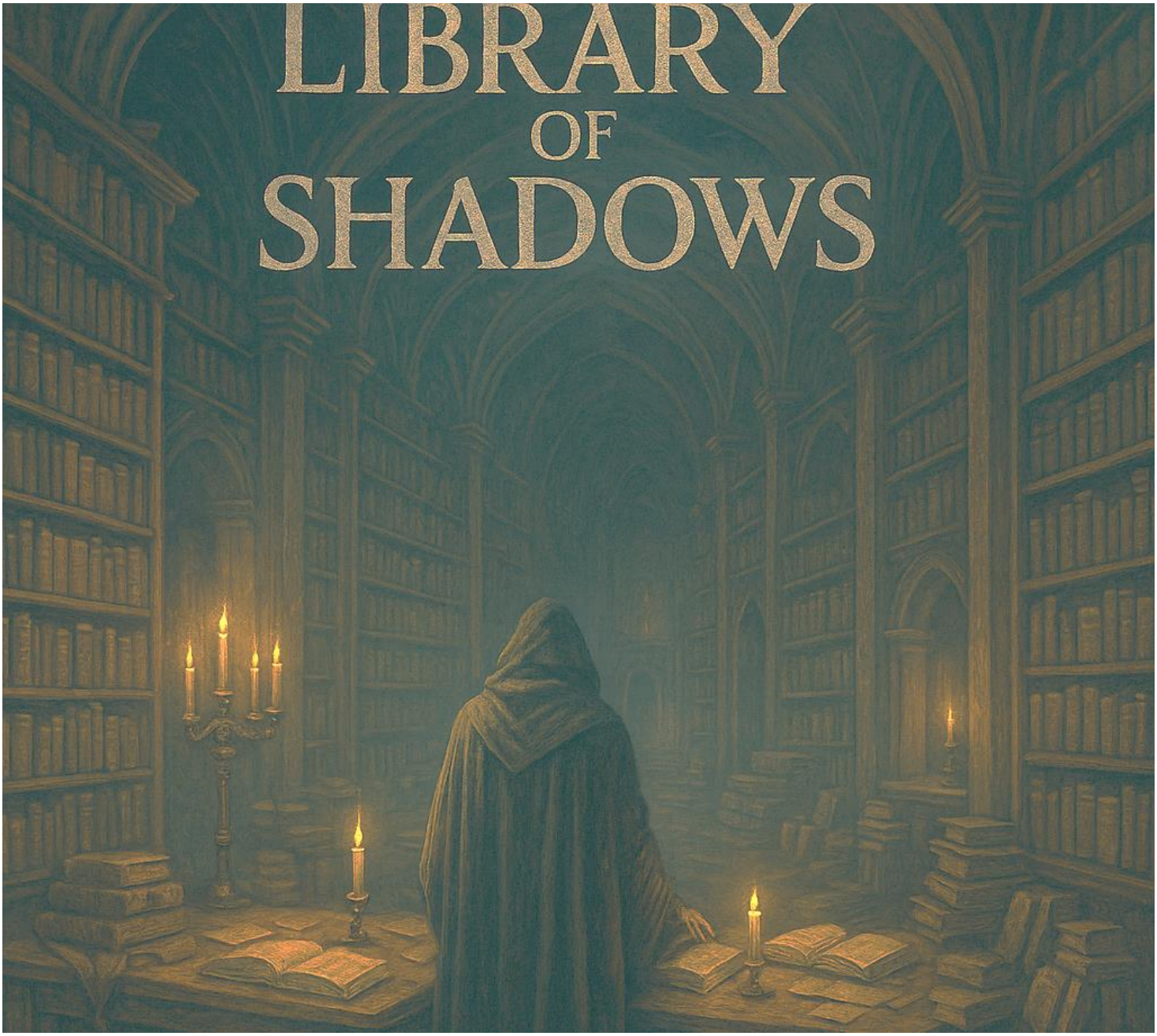


A course project for

CS210 Data Structures and Algorithms with Dr. Basit Qureshi

Posted: Tuesday, 22 April 2025

Due: Saturday, 3 May 2025

Weight: 15 %

Test site: https://www.hackerrank.com/242cs210project

Submission on: https://lms.psu.edu.sa/

In a mysterious digital world lies the Library of Shadows — an ancient, intelligent archive that stores knowledge from countless forgotten civilizations. You, the Keeper of the Index, have been tasked by the Secret Council with building and maintaining the library's powerful search system: the "Eye of Insight."

You must create the **Eye of Insight**, a magical search engine with the following powers:

1. Index the Scrolls: When new *scrolls* (text files) are added, you must read and store every *token* (word) and the *scrolls* they appear in. The **index** is your lifeblood — make it fast. Use a Binary Search Tree (BST) with an average runtime of O(log n).
2. Search the Lore: Allow visiting scholars to search for *tokens* (words or phrases). For each token query, list which scrolls contain the word and on which line.
3. Purge the Dangerous tokens: Occasionally, the Council will ban a token. You must remove all traces of it from the index — as if it never existed.
4. List all tokens: *Traverse* through the Scrolls to list the frequency of all the tokens. The scholars have decided to traverse using only the In-order method.

Your program reads input from the console. It reads the text file token by token. A token can be any word having removed any spaces or punctuation marks etc.

Your index is made of a *BST*. Each BST node (*BSTNode*) stores a *token*, an integer value called the *frequency*, this is the count of the number of times this token appears in all the scrolls (files); and a singly linked list (S*List*). The SList consists of Linked list nodes (*ListNode*) where each node contains the *filename* and *line-number* where the *token* is found. The following figure illustrates the data structure:

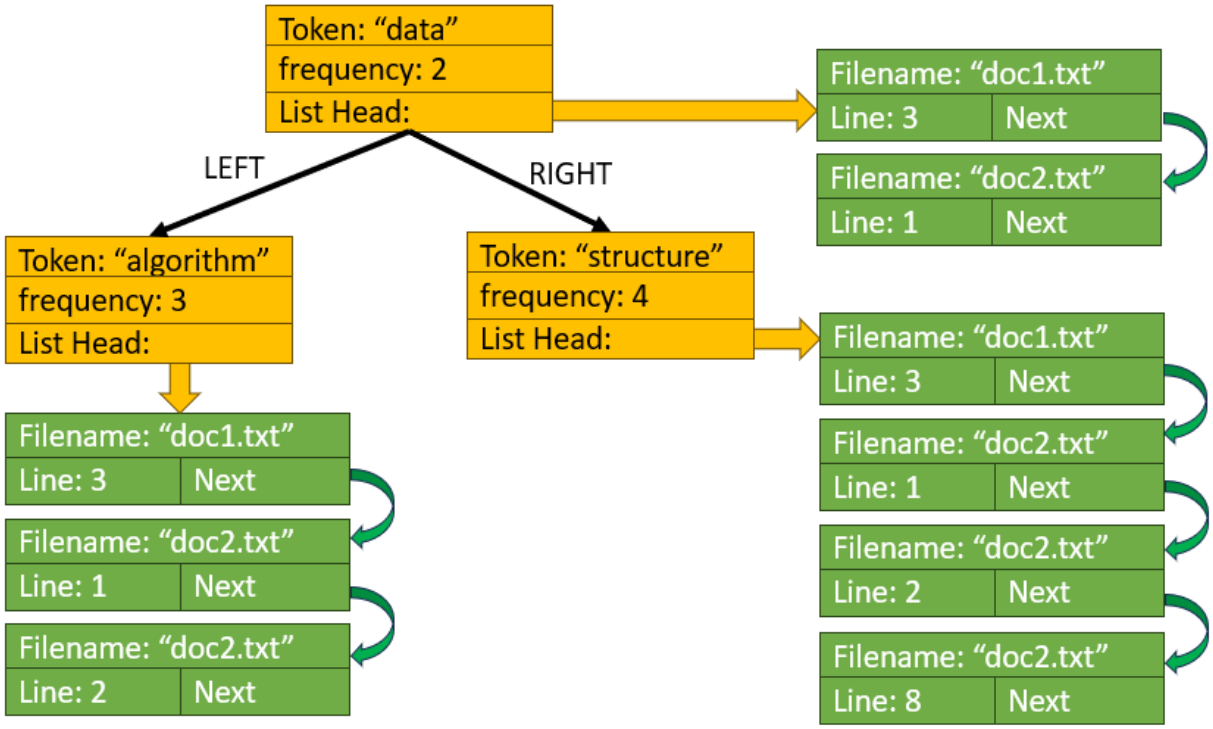


Figure 1: The "Eye of Insight Index" data structure is a BST with embedded singly linked list.

Warning! The council requires that the Eye of Insight must be written in the scared Java language, however they do not trust the Collections Framework. You must avoid the use of Java Collections Framework at any cost.

Students are expected to use the following API to develop their software.

Index
BSTNode root int size
Index() Insert(String) Remove(String) Search(String) Traverse() String toString() //appropriate methods

.

BSTNode
String token int frequency SList List BSTNode left BSTNode right
BSTNode() //appropriate methods

.

SList
int size Node head
Insert(Node) Remove(Node) Search(String s) String toString() //appropriate methods

.

Node
String filename int lineNumber Node next
//appropriate methods

.

Solution
main method //appropriate methods for reading and writing

Students are allowed to add/modify the API to suit their needs such as inclusion of setters and getters, however the attributes and methods names in the above classes must persist. In addition to these classes, students will also write a tester program that reads from the console, processes the input and produces the correct output for this software considering the following conditions.

Input Format

- A line starts with a integer Command followed by instructions.

- If the Command is 1, it is followed by an integer S which represents the number of scrolls / files to read. For each Si, the line reads the filename and a integer value N, which represents the number of lines in the scroll Si. Your program reads all this information in the data structure.
- If the Command is 2, it is followed by a token T. You program searchers for token T in the data structure and prints a count P i.e. how many times did the token T appear in the data structure followed by P number of lines. Each Pj line consists of the filename and the line number where the token was found.
- If the Command is 3, it is followed by a token T. You are required to remove all instances of T from the data structure.
- If the Command is 4, you program prints the In-order traversal of the binary tree.
- For any invalid input, the program prints -1.
- 1 <= Command <= 4
- 1 <= S <= 100000
- 1 <= P <= 100

Running your program

The following is a test run with explanation

Standard Input: Consider the following input:

```
1 3
File1.txt 3
data structures are cool
algorithms are fun
this is a cool project
File2.txt 2
search the data
time is out
File3.txt 1
data in data is fun
2 data
3 are
4
```

Standard Output: The above generates the following output

```
4
File1.txt 1
File2.txt 1
File3.txt 1
File3.txt 1
a algorithms cool data fun is out project search structures the this time
```

Explanation
1 3

```
File1.txt 3
data structures are cool
algorithms are fun
this is a cool project
File2.txt 2
search the data
time is out
File3.txt 1
data in data is fun
```

The first line reads 1 3. 1 means insert. 3 means there are 3 files/scrolls to read.
The first file name is File1.txt and it has 3 lines of text.

```
data structures are cool
algorithms are fun
this is a cool project
```

The next file name is File2.txt and it has 2 lines of text.

```
search the data
time is out
```

The next file name is File3.txt and it has 1 line of text.

```
data in data is fun
```

2 data

2 data -> requires you to print all the occurrences of data. In the next line, you print 4 because data appears 4 times; followed by details for each occurrence (filename and line number).

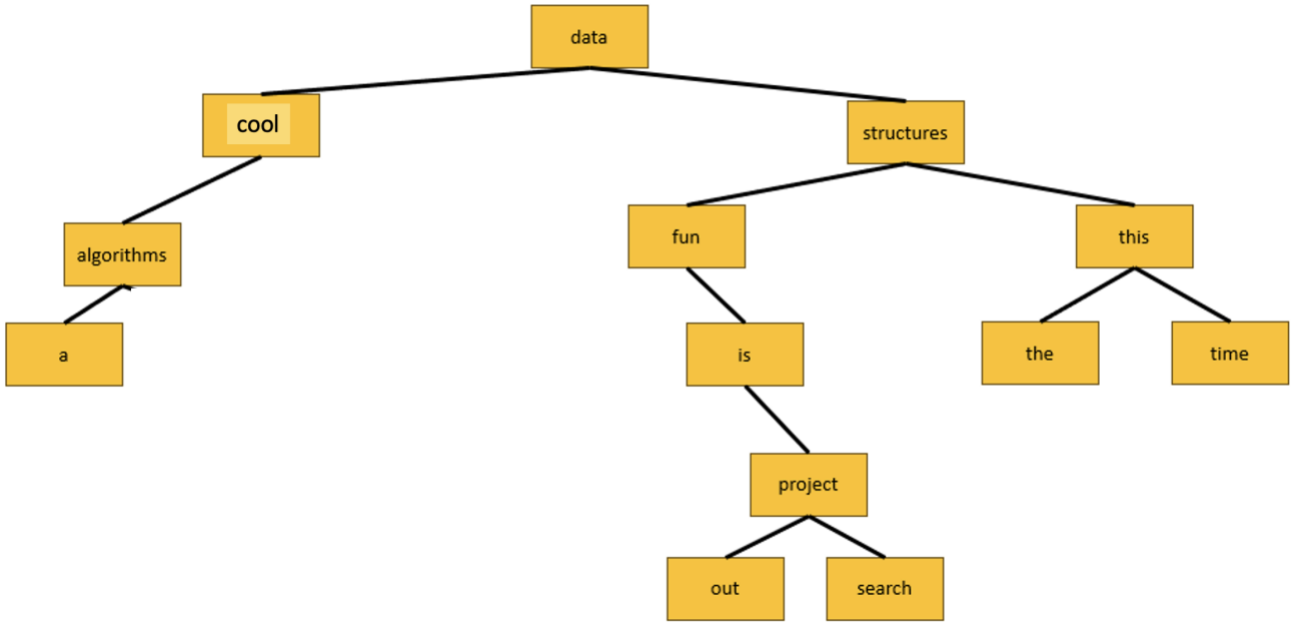
```
4
File1.txt 1
File2.txt 1
File3.txt 1
File3.txt 1
```

3 are

3 are -> requires you to search are and remove it from the data-structure.

4

4 -> requires you to print the Inorder traversal of this tree.
At this moment the tree looks like this:



Inorder traversal of this tree is:

```
a algorithms cool data fun is out project search structures the this time
```

Evaluation:

You are allowed to work as a group with maximum 2 members in a group. Your work's evaluation would be based the following criterion.

Step 1: Verify correctness of your program on Hackerrank

You have unlimited chances to submit your code to hackerrank before the deadline. When you submit your code, the hackerrank platform tests your program against pre-built (hidden) testcases and grades your work. You need to fix all the mistakes in the project and try to earn a full score. For this project, there are no limitations on time and memory usage.

Step 2: Writing Report

Complete a report consisting of the following sections:

1. **Cover:** Use the coversheet. Add personal details and a screenshot from Hackerrank showing your score.
2. **Introduction**: Explain why you think the data structure you implemented is a good choice for implementing the Index.
3. **Performance**:
    a. Give the runtime of your program using big Oh notation.
    b. Give the best case and worst case scenario using big Oh notation.
    c. Compare *Index* data structure to a Doubly Linked list.
    d. Compare *Index* data structure to a AVL Tree.

4. **Conclusion**: Give reasons why or why not the *Index* data structure is better than a doubly linked list or a AVL implementation.
5. **Code**: Copy paste all of you code

Step 3: Upload your final submission to LMS. The submission will consist of

- One Microsoft word file consisting of your report.
- One java file consisting of all your code.

The java file must consist of the following structure

```
public class Solution {
}
class Index {…}
class BSTNode {…}
class SList {…}
class Node {…}
```

Interview:

Student would be requested to demonstrate their work. The instructor may ask the student to modify the code to satisfy any test-case(s) there in.

Evaluation and Grading:

- Correctness of program verified on hackerrank - 5 points
- Completeness of report – 5 points
- Interview – 5 points
- Total = 15 points

Warning: Not allowed to use java collections framework or any of its classes such as ArrayList etc.

The instructor reserves the right to determine the scores of each test case. Test-cases will be posted on hackerrank, students will have unlimited number of opportunities to post and test their project until the due date. **The system will not take any submissions after the due date.**

Code Inspection and plagiarism:
The code would be inspected by the instructor. The instructor would use the MOSS tool (https://theory.stanford.edu/~aiken/moss/) to determine the originality of submission.

If the code similarity is above 40%, the students would earn ZERO score on the project. This includes *all* group members for all teams involved as well (i.e. all other groups with similar code)

Submission Dead-Line:

The submission deadline is final. Late Submissions will be awarded ZERO points.

Important Notes:

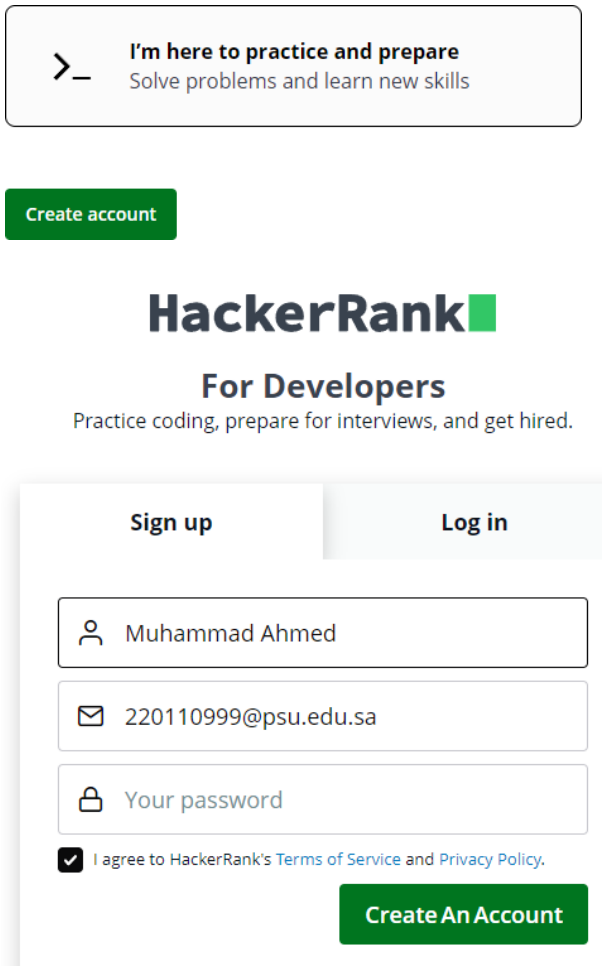
- It is the student's responsibility to check/test/verify/debug the code before submission.
- It is the student's responsibility to check/test/verify all submitted work (including jar files)
- It is the student's responsibility to verify that all files have been uploaded to the LMS.
- Incomplete or wrong file types that do not execute will NOT be graded.
- For each project, instructor will provide a few sample test-cases to verify the execution of your program.
- After an assignment/project has been graded, re-submission with an intention to improve an assignments score **will not be allowed**.

TUTORIAL:

Submission on Hacker Rank

Step 1. Register on https://www.hackerrank.com/

Make sure your username as your PSU ID number.



2. Join contest "242CS210project"

Start working on your project.