Lab Topic: Optimized Word Search Using BST Variants

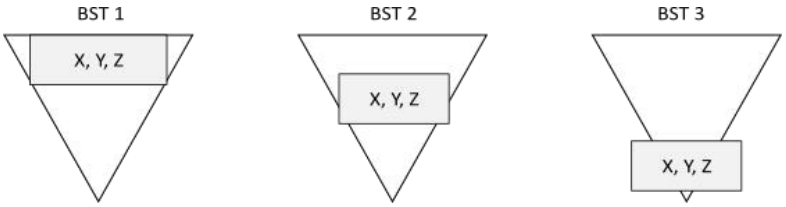
Course: Data Structures
Level: 1st Year Computer Science
Year: 2025

Objective

To optimize the search for words starting with specific alphabetic letters, a triplet of Binary Search Trees (**BST1, BST2, BST3**) is used, following these rules:

- **BST1**: Each inserted word that starts with **X = 'Y', Y = 'Z',** or **Z = 'a'** is moved to the root using rotations.
- **BST2**: Each inserted word that starts with **X, Y,** or **Z** is moved to the middle of the search path using rotations.
- **BST3**: Each inserted word that does not start with **X, Y,** or **Z** is moved to the root using rotations.

The following figure illustrates this triplet:



Additional Considerations:

- The BSTs can be designed to include a **Parent** pointer for each node.
- Alphabetic letters are **ordered** as follows: 'A' to 'Z' (uppercase) followed by 'a' to 'z' (lowercase).

Section 1: Implementation in Z

1. Initial Tree Construction & Verification

1. Generate a file **F** containing **N randomly generated words** (N ≥ 100).
2. Construct BST1, BST2, and BST3 from **F**.
3. To verify the correctness of the constructed trees, perform the following operations for each BST:

    - Count the number of words starting with X, Y, and Z.
    - Compute the depth of the tree.
    - Perform an inorder traversal.

- For each level, compute and display the number of nodes starting with X, Y, and Z.

2. Word Search Operations

Single Word Search

Given a word **Word**, the search process follows this algorithm:

```
If Charact(Word, 1) is in [X, Y, Z]
   Search in BST1
Else if (Charact(Word, 1) > X) or (Charact(Word, 1) > Y) or
        (Charact(Word, 1) > Z)
   Search in BST2
Else
   Search in BST3
```

The Z function `Charact(Word, 1)[Caract in French]` returns the first letter of the word **Word**.

Note: If the search element is not in the tree, the search may terminate before reaching a **Null** node.

Write this algorithm.

Range Search: [Word1, Word2]

Write an algorithm using the triplet (BST1, BST2, BST3) to search for all words within the range **[Word1, Word2]**.

3. Test Algorithm

Design a main algorithm that tests all the modules discussed in this section.

Note: Z is not designed for high-quality result presentation. Focus on correct implementation.

Section 2: Implementation in C

(a) Automatic Translation to C and Program Testing

- Convert the Z program to C.
- Ensure that the translated program functions correctly.

(b) Enhancing the Output Presentation

- Improve the visual presentation of results.

(c) Additional Statistical Modules

Module 1: Implement a module to construct a standard Binary Search Tree (**BST0**).

Module 2: Implement a simulation algorithm to evaluate word search efficiency:

For **i = 1 to M** (M ≥10**)**:

- Generate a file **F** with **N** random words (N≥10,000).
- Build BST0, BST1, BST2, and BST3 from **F**.
- Generate a file **F2** with **N** random words.
- Search for all elements of **F** and **F2** using BST0 and the triplet (BST1, BST2, BST3).
- Compute and save the total length of search paths traversed separately for successful and unsuccessful searches.

Module 3: Implement a simulation algorithm to evaluate the efficiency of word range search:

For **i = 1 to M** (M ≥10):

- Generate a file **F** with **N** random words (N≥10,000) .
- Generate a file **F2** containing **N/2** random word pairs.
- Build BST0, BST1, BST2, and BST3 from **F**.
- Perform range searches for all pairs in **F2** using BST0 and the triplet (BST1, BST2, BST3).
- Compute and save the total number of nodes traversed.

If the BSTs are built with a parent field, the total number of nodes corresponds to the number of **Parent**, **Left Child**, and **Right Child** operations performed.

If the BSTs are built without a parent field, the total number of nodes corresponds to the number of **Pop**, **Left Child**, and **Right Child** operations performed.

Submission Requirements

Submit a **1-page PDF report** including:

1. Table and Graphs comparing search efficiency for a single word **Word** and conclusions. Consider both successful and unsuccessful search cases.
2. Table and Graphs comparing search efficiency for the range **[Word,1 Word2]** and conclusions.

Grading Criteria

- **Z Implementation:** 15 points (Presentation of the program, results, and result verification)
- **C Implementation:** 5 points (Output Presentation: 2pts; Simulation & Report: 3pts)

Handling Errors in C Translation

- If the translated C program does not work, students should attempt to debug it.
- Any identified and corrected errors will be rewarded with a **bonus**. In this case, submit both the faulty **Z** code and the corrected **C** code with a list of modifications.
- If the error remains unsolvable, submit the **Z** code for correction.

Software Requirements

- **Standard C language using Dev-C++ or Code::Blocks.**
- **Khawarizm (French Version):** Download
- **Khawarizm (Multi-language Version):** Download

Plagiarism Warning

Strict measures will be taken against plagiarism.

Submission Deadline

- **Deadline:** Saturday, May 31, 2025, before midnight.
- **Submission Format:** A specific form will be provided.
- **Late Submission Penalty:** 2 points deducted per day (maximum 2 days tolerated).

This lab topic has been reviewed and enhanced, both in content and formatting, by ChatGPT.