Binary Search Tree in C++ with *Qt*

Benoît Piranda

1 Introduction

We will define a binary search tree to store and search for words in the database provided (in file "*words.txt*"). To do this, two classes are provided:

- the `Word` class, which contains a character string and a rule for comparing two element keys,
- and the `BST` class, which stores the binary tree.

There is also an interface for displaying the search term and the list of results obtained.

2 Download and basic exercises

Download the "*BSTword.zip*" code from Moodle, unzip it into a working directory, and test the code. Try searching for the word *"apple"*. What do you notice? Study the code provided for the two classes (`Word` and `BST`), particularly the `insert(...)` method of the `BST` class.

Please note that we want to store words in uppercase, removing duplicates.

Exercise 1** : Code analysis

Question 1: In the `Word` class, when is it possible to assign the `str` attribute?

Question 2: In the `Word` class, which operators are overloaded?

Question 3: If `w` is a pointer to `Word`, how do you evaluate whether the attribute `str` of `w` is less than "bridge" and not equal to "false"?

Question 4: Study the `BST::insert()` function. How is the data structure constructed? Draw an example after inserting the elements {"`apple`","`bear`","`cat`", "`dog`"}.

Question 5: Study the `BST::search()` method and explain how the while loop works. What is the complexity of this algorithm?

Question 6: If we consider that the words in the data file are sorted in ascending order, what optimization can be applied to this algorithm? What is the effect on complexity?

Exercise 2* : Insertion and search method in a BST

Question 1: Create a method `bool search(const QString &src,int &n)` that returns if a node that constains *src* exists.

Question 2: The insertion procedure (`BST *insert(const QString &word)`) is quite similar to the previous searching method. If the node is empty it fill it with `word` and return the node pointer. In general cases, it starts at the root and recursively goes down the tree searching for a location in a BST to insert a new node.
If the element to be inserted is already in the tree, we are done (we do not insert duplicates, and the method returns the node pointer). Otherwise create a new node and insert it at the good branch.

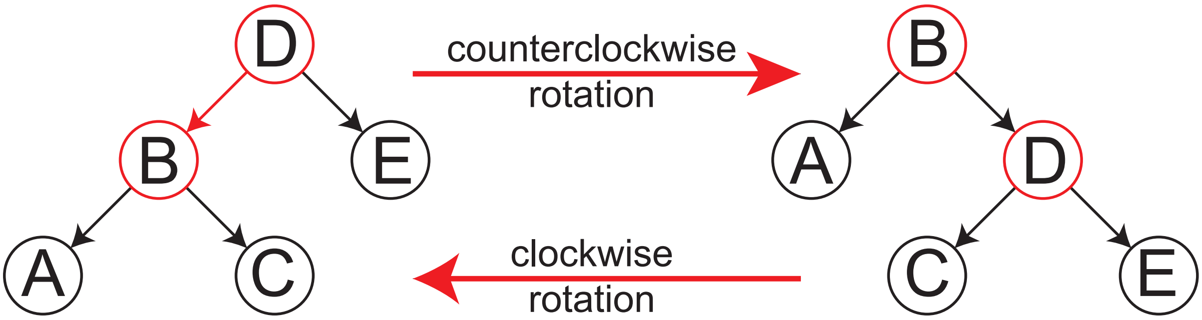

Figure 1: The two rotations to equilibrate a subtree.

Question 3: Insert a list of ordered elements, for example: {"`apple`","`bear`","`cat`", "`dog`, "`elephant`"}. What is the shape of the generated BST? Is is efficient for the searching algorithm?

Exercise 3** : Equilibrate the BST

In order to balance binary search trees during their construction, Georgii Adelson-Velsky and Evguenii Landis proposed the following method:

1. Each node stores a balancing coefficient $b = depth(right) - depth(left)$.
2. If this coefficient verifies $|b| > 1$ for a node P, balancing consists of applying a rotation between P and the child of the longest branch. The action will therefore be a clockwise rotation between P and its right child if $b > 1$, or a counterclockwise rotation between P and its left child if $b < -1$.

The figure 1 shows the effects of a counterclockwise rotation between D and its left child B and the opposite rotation.

Question 1: Write the method `int BST::depth()` that returns the depth of a branch.

Question 2: Complete the previous method to compute the balance coefficient at each node and run the appropriate rotation if necessary.

Question 3: Insert a list of ordered elements to test your new code, for example: {"`apple`","`bear`","`cat`", "`dog`, "`elephant`"}. What is the new shape of the generated BST?

Exercise 4* : Complexity

Question 1: How complex was the algorithm provided?

Question 2: What is the maximum number of nodes you would need to examine to perform any search? Considering a complete tree, what are the probabilities of crossing $1, 2, ..H$ node?

Question 3: Deduce the complexity of this new solution?

Question 4: Considering the file contains 45373 different words, what is the theoretical size of the BST?

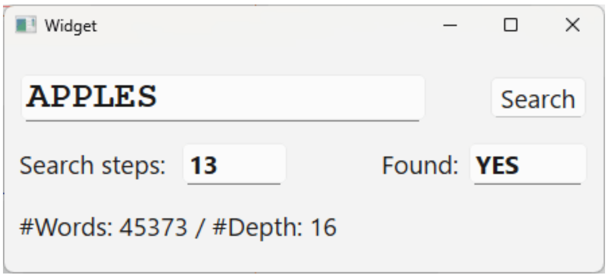

Figure 2: Snapshot of the application.