Name: Morad E.

How to run the code: Load the project into Eclipse and select run.

Part 1

Note: For (a) and (b) comments are provided in the code, explaining implementations in depth, along with providing the time complexities.

c) Considering the IS-ELEMENT operation, complexity is O(n) whether sorted or unsorted. As for ADD, the complexity for unsorted is O(1): insertion after a particular element is independent of the length of the list; the complexity for inserting a value in a sorted way in sorted doubly linked list is O(n): if the location to look in the list isn't given, linear traversal of the list is necessary.

d) Regarding the naïve approach, taking into account tree1 having n1 as its number of nodes and another tree2 which n2 as its number of nodes, taking each of the elements in the one tree and inserting it into the other would result in complexity O(n1 x log(n2)). By the end of the operation there will be one binary search tree with n1+n2 nodes.

Alternative solution with code provided: Create a third tree and run a recursive function. In the recursive function, consider if the tree is empty, just return because there's nothing to add. Otherwise, recur through the left subtree and then the right, making recursive calls to the function, finally adding to the new tree key values of the nodes representing the union of trees1&2. Once there are no more additions to make to tree3 (base case) return tree3 with the unified elements. Complexity: [O(m*n) where m and n are number of nodes in given two trees] with (additional comments in code).

[Previous solution before announcement to include a version with implementation]

A potential solution may be:

First, apply in-order iteration of the whole trees, converting the BSTs into sorted linked lists. The flattening is O(n) and so flatting both trees has complexity O(n1+n2). Then, merge the two sorted linked lists into a single sorted linked list, with pointers to the heads of the two lists, picking the smaller head and advancing its pointer: complexity here is O(n1+n2). Finally, convert the sorted linked list (of n1+n2) to a balanced BST: complexity would thus be O(n1+n2) ultimately. Considering n1 and n2 of same magnitude order, O(n1+n2) is better than O(n1 x log(n2)).

Part 2

a) Average time taken by BST: 302,220ns; Average time taken by DLL’s linear search: 73,870ns
For the BST, searching through the data in an unstructured manner makes the search less effective as it’s typically better optimised for searching ordered elements; the result is an unbalanced tree with height proportional to n items.

b) Output of Set-size(S): 16,536

c) Height of the BST implementing set S: 34