Assessed Exercise 2

This exercise is for submission using Moodle and counts for 10% of the total assessment mark for this course.

This exercise is worth a total of 30 points.

The deadline for submission is Friday 26 March 2021 at 4:30pm.

Exercise

This exercise has two parts. The first involves implementing in Java the Dynamic Set abstract data type using two different data structures. The second involves running an empirical study to compare the performance of each implementation.

Submission

Submit the Java sources of your implementations **and** a short (max 3 pages) report describing what you have done in each part of the exercise. Your report should include a heading stating your full name and matriculation number and clear instructions on how to run your code.

Please make sure the report is in pdf format and your sources are not password protected.

Part 1

The *Dynamic Set* is an abstract data type (ADT) that can store **distinct** elements, without any particular order. There are five main operations in the ADT:

- ADD(S,x): add element x to S, if it is not present already
- REMOVE(S,x): remove element x from S, if it is present
- IS-ELEMENT(S,x): check whether element x is in set S
- SET-EMPTY(S): check whether set S has no elements
- SET-SIZE(S): return the number of elements of set S

Additionally, the Dynamic Set ADT defines the following set-theoretical operations:

- UNION(S,T): return the union of sets S and T
- INTERSECTION (S,T): return the intersection of sets S and T
- DIFFERENCE(S,T): returns the difference of sets S and T
- SUBSET(S,T): check whether set S is a subset of set T

Implement in Java the Dynamic Set ADT defined above using

a) a doubly linked list and [9]
b) a binary search tree. [9]

Observe that the ADT implementation should use Java Generics (see Lab 3) and operations should be in the form `s.add(x)`, `s.remove(x)`, etc. Explain in the report your implementation, noting the running time (using big Oh notation) of each operation in both implementations. Note you can use a self-balancing binary tree but no extra marks will be awarded. Also, you are not allowed to rely on Java library classes in your implementation.

c) Suppose your implementation based on a doubly linked list maintains the list sorted. Explain in the report what are the implications of such implementation choice on the complexity of operations ADD and IS-ELEMENT? [2]

d) A naive implementation of operation UNION(S,T) in the implementation based on BST consists in taking all elements of BST S one by one, and insert them into BST T. Describe in the report an implementation with a better running time. Use big Oh notation to indicate running times. [5]

Part 2

a) Compare the two implementations of the Dynamic Set ADT by carrying out the following empirical study. First, populate (an initially empty) set S with all the elements from dataset `int20k.txt` provided on Moodle under Lab/Files. Then, generate 100 random numbers in the interval [0, 49999]. Finally, for each random number x record the time taken to execute IS-ELEMENT(S,x). What is the average running time of IS-ELEMENT over 100 calls in the two implementations of the ADT? Comment and explain your findings. [3]

b) What is the output of SET-SIZE(S)? [1]

c) What is the height of the BST implementing set S [1]