Department of Computer Science
Trent University

COIS 3020H: Data Structures and Algorithms II

Assignment 3: Being Lazy

Total grade: 60
Due date: Monday, April 5, 2021 at 23:59 EST
Late assignments will be deducted 10% per day up to five days (including weekends)

Part A: Ternary Trees (25 marks)

Using the ternary tree implementation of the Trie given in class, implement (12 marks) and test (6 marks) the public and private methods below to remove a key (value) from a ternary tree. Implement a Print method as well to show the ternary tree (5 marks). Main program/documentation (2 marks).

```
// Public Remove
// Calls the private Remove which carries out the actual removal
// Returns true if successful; false otherwise
public bool Remove (string key) { ... }

// Remove
// Remove the given key (value) from the Trie
// Returns true if the removal was successful; false otherwise
private bool Remove (ref Node p, string key, int i) { ... }
```

Part B: Lazy Binomial Heaps (35 marks)

Each public method (Insert, Remove and Front) of the Binomial Heap seen in class takes O(log n) time. However, if we relax the condition that at most one instance of each B_k can be part of the root list then the time to perform Insert can be reduced to constant time since no Merge is required. On the other hand, the Remove method is now entrusted to "clean up" the mess left behind by Insert. It does so by coalescing (combining) binomial trees so that the binomial heap is returned to having at most one instance of each B_k.

Re-implement the methods Insert (2 marks), Front (2 marks), and Remove (4 marks) of the class Binomial Heap, removing the methods Degrees, FindHighest, Merge, Union, and Consolidate and introducing the method Coalesce (9 marks) as described in the reference below.

https://web.stanford.edu/class/cs166/lectures/08/Small08.pdf

To support the Coalesce method, redefine the data structure of binomial heap as an array $B[0..\log_2 n]$ where B[k] is a reference to a root list of binomial tree(s) B_k. Finally, include a data member that always maintains a reference to the item with the overall highest priority (3 marks) so that Front can execute in constant time as well. Testing (8 marks). Implement a Print method as well to show the lazy binomial heap (5 marks). Main program/documentation (2 marks). **Note:** My re-implementing the Binomial Heap in this way, the amortized cost of Remove is O(log n) and the worst case behaviour of Insert and Front is reduced to O(1). Just by being lazy!

Submit your project (all files including an executable) as well as your test document at Blackboard.