COMP90077 Assignment 1 (10% out of the total Subject mark)

Due Date: **09:30AM March 28 (Monday), 2022**

Prepared by **Junhao Gan** and **Tony Wirth**

Problem 1. (Amortized Analysis.) [5 Marks] For every node u in a rooted tree, T, let $T(u)$ be the sub-tree rooted at u; denote the number of nodes in $T(u)$ by $size(u)$. Consider the following variant of the binary search tree (BST).

A binary search tree is a *weight-balanced binary search tree* (WB-BST) if, for every node u in the tree, the following *weight balance* holds:

$$size(u\text{'s left child}) \leq \tfrac{1}{\beta} \cdot size(u) \text{ and } size(u\text{'s right child}) \leq \tfrac{1}{\beta} \cdot size(u),$$

where β is a constant with $\frac{3}{2} < \beta < 2$.

Moreover, a WB-BST is *perfect*, if, for every node u in the tree,

$$0 \leq size(u\text{'s left child}) - size(u\text{'s right child}) \leq 1.$$

Insertions, deletions and searches on a WB-BST are each implemented by standard BST-style algorithms. The only difference is that for an insertion (or a deletion), it needs to maintain $size(u)$ of all the nodes u on the path from the root to the target node in the operation (i.e., the leaf inserted in the insertion, or the node removed in the deletion).

Suppose the weight balance is violated at some node: let u be the *highest* node (i.e., the node with smallest depth) that has a weight-balance violation. The WB-BST T invokes a *rebuild* operation at u, denoted by $rebuild(u)$:

- (i) destroy $T(u)$, the sub-tree rooted at u, and
- (ii) rebuild a *perfect* WB-BST on all the elements in the destroyed $T(u)$.

Solve the following problems.

- (a) [1.5 Marks] Prove that the $rebuild(u)$ can be performed in $O(size(u))$ worst-case time.
- (b) [0.5 Mark] Prove that the height of a WB-BST on n elements is at most $\lfloor \log_\beta n \rfloor + 1$.
- (c) [1 Mark] Show that after the rebuild operation on the highest node u that has violation on the weight balance, the entire tree T is restored to a WB-BST.
- (d) [2 Marks] Design an appropriate potential function and then show that the amortized cost for each insertion and deletion is bounded by $O(\log_\beta n)$.

Problem 2. (The Quake Heap.) [5 Marks] In class, we discussed the quake heap feature of maintaining a node-number-at-height invariant:

$$n_{i+1} \leq \alpha \cdot n_i, \text{ for all } i = 1, 2, \ldots, h'_{\max} - 1\,, \tag{1}$$

where $\alpha \in (\frac{1}{2}, 1)$ is a fixed constant and $h'_{\max}$ is the maximum tree height in the forest. If we write the dependence on α explicitly, the amortized cost for *decrease-key* is $O(\frac{1}{2 \cdot \alpha - 1})$ and the amortized cost for *delete-min* is $O(\log_{1/\alpha} n)$. The closer α's value is to $\frac{1}{2}$, the more expensive *decrease-key* is. To remedy this, we consider the following invariant, instead of invariant (1):

$$n_{i+c} \leq \alpha \cdot n_i, \text{ for all } i = 1, 2, \ldots, h'_{max} - c\,, \tag{2}$$

where $c \geq 1$ is an integer constant. Clearly, when $c = 1$, invariant (2) is exactly the same as invariant (1). Now to solve the following problems, with respect to invariant (2).

- (a) [1 Mark] Prove that the maximum possible height $h_{\max}$ is less than $c \cdot \log_{1/\alpha} n + 2 \cdot c$;
- (b) [0.5 Mark] Prove that the space consumption bound of the quake heap now becomes $O(c \cdot n)$;
- (c) [1 Mark] Define a potential function related to the integer constant c, which can be used to perform amortized analysis for the subsequent questions;
- (d) [0.5 Mark] Show that the amortized cost of *insert* is bounded by $O(1)$;
- (e) [0.5 Mark] Show that the amortized cost of *decrease-key* is bounded by $O(\frac{1}{c} \cdot \frac{1}{2 \cdot \alpha - 1})$; and
- (f) [1.5 Marks] Prove that the amortized cost of *delete-min* is bounded by $O(c \cdot \log_{1/\alpha} n)$.