CSCB63 Assignment 1

Worst Case Complexity, BSTs and Augmented Data Structures

Due 11pm, Feb 5, 2019

Warning: *Your electronic submission of a PDF to MarkUs (https://markus.utsc.utoronto.ca/cscb63w19) affirms that this assignment is your own work and no one else's, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters, the Code of Student Conduct, and the guidelines for avoiding plagiarism in CSCB63. Note that using Google or any other online resource is considered plagiarism. Late assignments will not be accepted.*

1. (10 marks) For the following questions prove the bounds using the definitions of $\mathcal{O}, \Omega$ and Θ and not the limit laws.

   (a) (5 marks) Prove that for any $k \in N, 1^k + 2^k + \ + n^k \in \Theta(n^{k+1})$.

   (b) (5 marks) Choose an appropriate $g(n)$ and prove that $\frac{n+1}{2n^2+2} \in \Theta(g(n))$.

2. (10 marks) Consider a binary tree T. Let $|T|$ be the number of nodes in T and $w(T) = |T| + 1$. Let x be a node in T, let L_x be the left subtree of x and let R_x be the right subtree of x. We say that x has the "approximately balanced property", $ABP(x)$, if $w(R_x) \leq 2w(L_x)$ and $w(L_x) \leq 2(R_x)$.

   Update. In this question you may choose whether you define height to be 0 for a single node or 1 for a single node. If you define height to be 1 for a single node (as we do in class) then in part (b) you want to prove $height(T) \leq \log_2(n+1)/\log_2 \frac{3}{2}$.

   (a) (4 marks) What is the maximum height of a binary tree T on n nodes where $ABP(root)$ holds?

   (b) (6 marks) We call T an ABP-tree if $ABP(x)$ holds for every node x in T. Prove that if T is an ABP-tree (and we define height of a single node to 0) then the height of T is $O(\log n)$. More precisely, show that

   $$height(T) \leq \log_2 n / \log_2 \frac{3}{2}$$

   .

3. (10 marks) Consider an ADT consisting of a set S of distinct integers and the following operations:

   INSERT(S,x): Insert the element x into the set S. This operation has no effect if x is already in S.

   DELETE(S,x): Delete the element x from the set S. This operation has no effect if x is not in S.

   QUERY(S,x): Return true if x is in S and return false if x is not in S.

CLOSEST-PAIR(S): Return two integers in S which are closest together in value. In other words, if CLOSEST-PAIR(S) returns the integers a and b, then they must satisfy the condition

$$\forall x \forall y (x \neq y \rightarrow |a - b| \leq |x - y|).$$

It is an error if S contains fewer than two elements.

Describe a data structure to implement this ADT. All operations must be performed in $O(\log n)$ time, where $n = |S|$. Describe any new information that will be stored. Justify why your algorithms are correct and why they achieve the required time bound.

4. [10 marks] Implement an algorithm for determining whether two AVL trees (storing ordered sets — just keys) have no keys in common. In other words, in the set-theory sense their intersection is empty. The starter code and the test program are on MarkUs: AVL.py, Node.py, and Test.py. You will add your code to only AVL.py and submit it. Do not submit any other Python files—they will be ignored. (To add your own classes, make them inner classes of AVL.) All other instructions (such as how to test your code) are documented in the source files.

Marking scheme If your code looks like a general algorithm (as opposed to customizing for the test cases) that takes $\mathcal{O}(m + n)$ worst-case time, where m and n are the tree sizes with $m \leq n$, you get 2 marks to start. If your algorithm takes $\mathcal{O}(m \lg(\frac{n}{m} + 1))$ worst-case time, you get 3 more marks. In addition:

- If your code compiles: additional marks for each test passed to a maximum of 5.

If your code does not look like a general algorithm that takes $\mathcal{O}(m+n)$ worst-case time, 0 to 3 marks depending on how far off it is. Examples: $m \lg n \notin \mathcal{O}(m+n)$; a hash-table approach takes $\Omega(m \times n)$ time.