Computer Science B63
University of Toronto Scarborough

Homework Exercise # 1

Handing in and marking

For this exercise, you need to submit your solutions to the pencil-and-paper exercises on crowdmark and your solutions to the programming question on MarkUs. Your pencil-and-paper solutions will be marked with respect to correctness, clarity, brevity, and readability. Your code will be marked with respect to correctness, efficiency, program design and coding style, clarity, and readability. This exercise counts for 10% of the course grade.

Question 1. Asymptotic Bounds [12 MARKS]

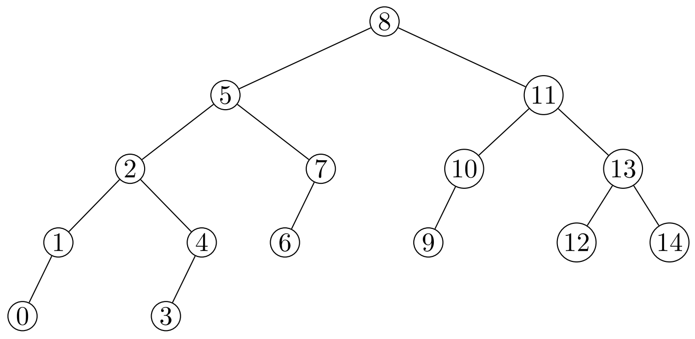
Prove each of the following *using the definitions of* Big-Oh / Big-Omega / Big-Theta.

- (3 marks) If $f \in \mathcal{O}(g)$ and $g \in \mathcal{O}(h)$ then $f \in \mathcal{O}(h)$, for all functions f, g, h in $\mathbb{N} \to \mathbb{R}^+$.
- (3 marks) If $f_1 \in \mathcal{O}(g_1)$ and $f_2 \in \mathcal{O}(g_2)$ then $f \in \mathcal{O}(g)$, where $f(n) = f_1(n) \cdot f_2(n)$, $g(n) = g_1(n) \cdot g_2(n)$, for all functions f_1, f_2, g_1, g_2 in $\mathbb{N} \to \mathbb{R}^+$.
- (3 marks) $2^{2n} \notin \mathcal{O}(2^n)$.
- (3 marks) If $f_1 \in \mathcal{O}(g)$ and $f_2 \in \mathcal{O}(g)$ then $f_{\max} \in \mathcal{O}(g)$, where $f_{\max}$ is defined by $f_{\max}(n) = \max(f_1(n), f_2(n))$, for all functions f_1, f_2, g in $\mathbb{N} \to \mathbb{R}^+$.

Question 2. AVL Trees Basic Operations [16 MARKS]

Show the AVL trees resulting from performing the requested sequence of operations. Whenever a rotation is required, indicate the type of rotation used and what node it applies to. Whenever a double rotation is required, show the intermediate step (i.e., show the result after each of the single rotations).

- On an initially empty tree, show each step of inserting the keys `9, 10, 12, 14, 3, 34, 19, 37, 20`, in this order.
- On the tree shown below, show each step of deleting the keys `2`, `11`, `12`, `13`, in this order.



Question 3. Implementing RAVL Trees [50 MARKS]

We provided you with the starter code for implementing augmented (with Rank) AVL Trees. Your task is to implement the functions declared in `RAVL_tree.h` that are not already implemented in `RAVL_tree.c`. **Each function must be implemented in either $\mathcal{O}(1)$ or $\mathcal{O}(\log n)$ running time**, as we learned in class. Note that `RAVL_tree.c` is the only file you will submit, so make sure your implementation works with the original provided `RAVL_tree.h`.

Make sure to **carefully study the starter code**, including the file `sample_session.txt`, before you begin to add your own. Making an erroneous assumption can lead to your function(s) earning 0 correctness marks, as your code will be autotested, so please study all requirements stated in the starter code and in this handout carefully.

The marking scheme for this question is as follows:

- Correctness (includes proper memory management):
  - `RAVL_Node* search(RAVL_Node* node, int key)`: 4 marks
  - `RAVL_Node* insert(RAVL_Node* node, int key, void* value)`: 8 marks
  - `RAVL_Node* delete(RAVL_Node* node, int key)`: 18 marks
  - `int rank(RAVL_Node* node, int key)`: 4 marks
  - `RAVL_Node* findRank(RAVL_Node* node, int rank)`: 4 marks
- Program design (modular implementation, self-explanatory code, clear logic, no repeated code, no unnecessary code): 8 marks
- Readability and coding style (good use of white space, good naming, etc.): 4 marks
  - You are strongly encouraged to use a `linter` to help check and/or auto-format your code. See C-programming resources linked from the course website.

Question 4. Augmenting AVL Trees [22 MARKS]

Mary is planning a party and wondering if every friend is available to drop by. Mary has a start and end time in mind, and every friend has their own start and end of free time. It is OK if a friend has to arrive late or leave early. This calls for an ADT consisting of a set S of distinct closed intervals and the following operations:

- `insert(S, x, y)`: Insert $[x, y]$ into the set S. This operation has no effect if $[x, y]$ is already in S. This inserts a friend's free time.
- `delete(S, x, y)`: Delete $[x, y]$ from the set S. This operation has no effect if $[x, y]$ is not in S.
- `meetAll(S, l, h)`: Return true if every interval in S overlaps with $[l, h]$, and an interval $[x, y] \in S$ if it does not overlap with $[l, h]$. This checks whether $[l, h]$ is an acceptable time for the party.

For focus, we assume $x < y$ and $l < h$.

Your task is to design a data structure to implement this ADT, such that all operations are performed in $\mathcal{O}(\log n)$ time, where $n = |S|$. You will do so by augmenting our familiar AVL tree.

1. Describe all information that will be stored in the nodes.
2. Provide pseudo-code for each required operation.
3. Justify why your algorithms are correct and why they achieve the required time bound.