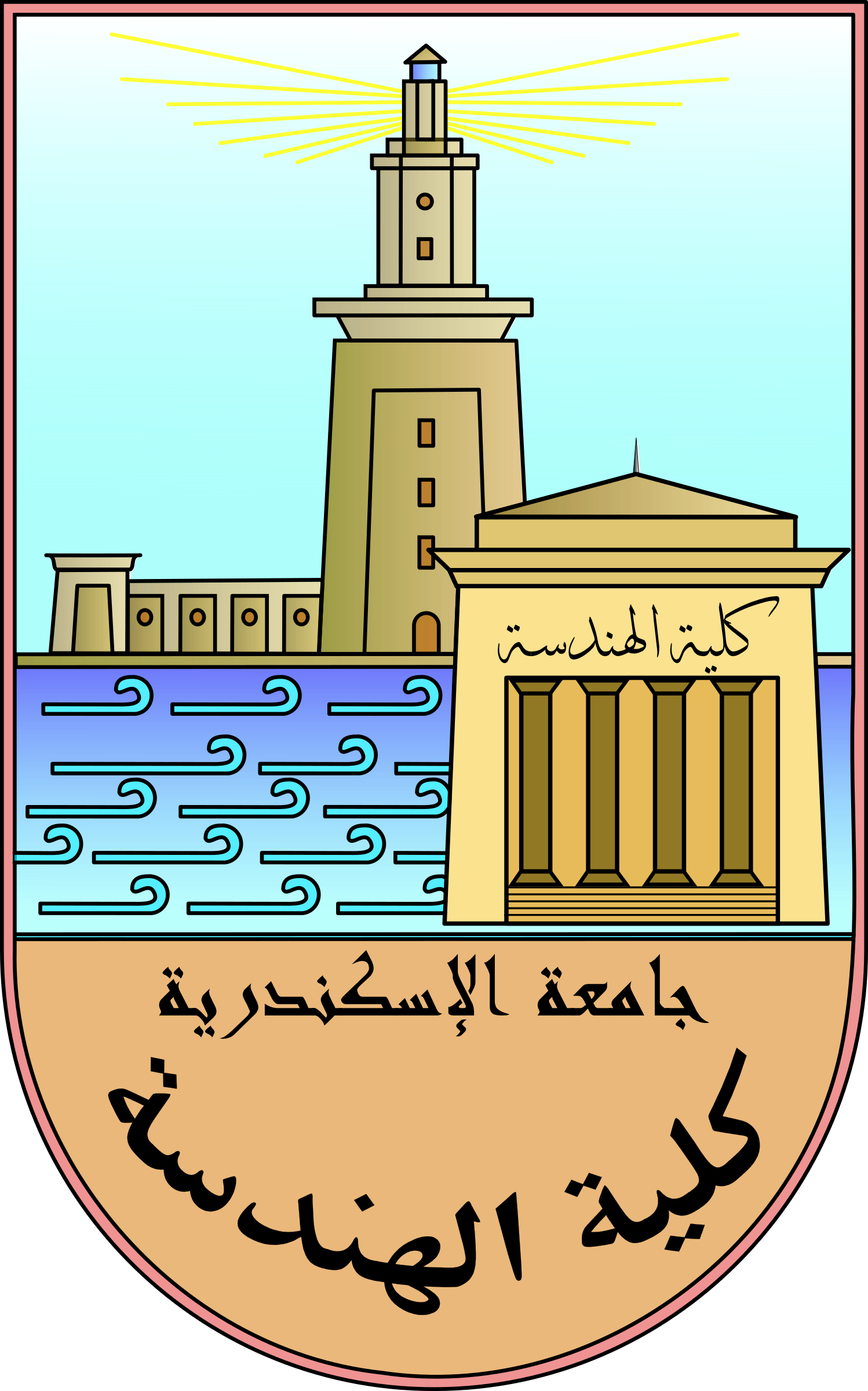
DSA: 1



Karim Tarek Ibrahim

Yahya Azzam

Yousef Elmedany

Abdullah Taman

1. Time Analysis

Peace be upon you,

First of all, we’ll discuss the order of insertion in both of the trees. In the AVL insertion was done in O(lg(n)), same was in the RB. RB was a little faster because the five invariants in it are more relaxed than the AVL only invariant: for any node the difference in height between the right child and left child is at most one, this invariant forces a lot of rotations in the AVL insertions, so it is naturally slower.

Second thing was deletion, deletion again is better in the RB although they are both are O(lg(n)) for the same reason above.

Third thing is searching, which is better in AVL surprisingly, you may wonder why. Well, the invariant that slowed down the insertion and deletion made the AVL easier to be searched, as its height will never exceed lg(n)+1, RB could reach 2\*lg(n), so for the same input AVL is better.

Batch insertion and deletion inherit the same characteristics of insertion and deletion of both trees, but it inherits it \*n times, so the order was O(lg(n)).

Both getHeight, getSize are optimized, they work in O(1) because we save these values in both trees.

Alhamdullilah.

That’s it, thank you.

1. Comparison

You can find the whole comparison [here](https://drive.google.com/drive/folders/177FNhFeePl4v7PgYbe6g2e0-g_LVyezh?usp=share_link).

The AVL beats the RB in searching as expected. Also, the RB beats the all in deletion as expected. But the surprise was in the insertion, AVL beat RB in it. Maybe the input we used caused that. We used 200 different words as input for this comparison. The max height for the AVL was 7, the RB was 12-14.