**The Longest Almost-Increasing Subsequence***Nitin Kaveriappa Udiyanda Muthanna*nudiyand@masonlive.gmu.edu

In computer science terms, the longest increasing subsequence problem is to find a subsequence of a given sequence in which the subsequence's elements are in sorted order, lowest to highest, and in which the subsequence is as long as possible. The longest increasing subsequence problem is closely related to the longest common subsequence problem, which has a quadratic time dynamic programming solution: the longest increasing subsequence of a sequence S is the longest common subsequence of S and T, where T is the result of sorting S.

For example, in below given sequence of numbers

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

a longest increasing subsequence is

0, 2, 6, 9, 11, 15.

This paper by Amr Elmasry on “The longest almost-increasing subsequence (LiaS)” talks about another version of this known as longest-almost increasing subsequence.

The improved LiaS algorithm pseudocode mentioned in the paper has been implemented in Java using AVL trees. As we know it the said algorithm of LiaS runs in O(n log k) and this is done using the balanced trees as the structure for the linked list. The various operations like Predecessor Xi, Successor Xi, Check (Xi+C), Delete, Insert and Balance all run in O(log k) using the AVL trees. The other minute sub operations can be emitted and considered as a constant. But overall this takes O(n log k) time.

Given the circumstances I have implemented the first half of the pseudocode and the latter half still had some issues during runtime for which I did not find the time necessary to resolve. Kindly consider my submitted code and report for grading even if it is incomplete and partial.

The code can still be run by using the “run.sh” bash script file. The code outputs the correct answer in most of the cases. Different outputs can be given by modifying the ‘inarr[]’ variable or the random number generator can be modified and used. The number of iterations can be controlled by modifying the ‘run.sh’ bash script file by changing the ‘$COUNTER’ value.

The output is in the following format and is available in the ‘output.txt’ file:

<================== Start =================>

<input\_array>

<default initial Pi array>

<Steps in the algorithm>

-----------Steps-------------

In order: <in\_order array>

<input\_array>

<final Pi array>

COUNT <value>

<================== End =================>

The “Run-time” since it is too small to measure, the number of iterations of every loop (including tree routines) has been calculated for array values in the range 1-1000 and 1000 runs were made. The results of which are tabulated below in the graph: