CSC 323 Algorithm Design and Analysis
Fall 2018
Instructor: Dr. Natarajan Meghanathan

Project 3: Algorithm to Determine the Longest Subsequence of Consecutive Integers in an Array using Hash tables, its Optimization and the Analysis of the Space-Time Tradeoff

Due by: Oct. 4th, 11.30 AM **Submission through Canvas (as instructed below)**

In this project, you will work on the algorithm (discussed in Module 1) to determine the length of the longest sub sequence of consecutive integers in an array.

You will implement the algorithm using Hash tables. You are provided with sample code (in C++ and Java) representing the linked list-based implementation of Hash tables (as an array of Linked Lists). You could go through the code to understand the implementation of a Hash table and the functions that can be called on it.

In the Module 1 slides, it was mentioned that the array of integers for the algorithm has to be unique. Actually, it is not a mandatory requirement. The algorithm can still work on a random array of integers with some integers repeated. However, for an arbitrary array (that could have repeated integers), the algorithm could waste time in determining multiple occurrences of the same sub sequence of consecutive integers. For example, if the array is: **7 2 1 3 1 1 3 4**, the algorithm would find three instances of the longest sub sequence **1 2 3 4**. On the other hand, if the array is: **7 2 1 3 4**, the algorithm would find only one instance of the longest sub sequence **1 2 3 4**.

In this project, you will implement two versions of the algorithm: The first version would be the implementation of the algorithm (as discussed in Module 1) without any optimization. The second version would be an implementation of the algorithm with an optimization strategy. In this regard, you are supposed to design a strategy to prevent the algorithm from searching for the same sub sequence more than once and implement an optimized version of the algorithm incorporating the strategy. You could use additional space as part of this optimization strategy, as large as O(n), where n is the number of elements in the input array. The expectation is that the additional space used would aid in reducing the run time.

You are given a startup code (in C++ and Java) titled: *HashTable_LinkedList_ContinuousSeqVersion* that has the Linked List-based implementation of a Hash table as well as the main function setup to run the algorithm for several trials and measure the average length of the longest sub sequence of consecutive integers and the average run time (referred to in the code as average detection time and is measured in milliseconds).

The number of trials is always 25. The maximum value for an element in the array is 25000 (i.e., the range of elements generated is 1...25000). Note that the programs are likely to run for a longer time, even if the implementation is correct. So, patiently wait!

The parameters that you will vary for a particular run of the code are as follows:
(N) number of elements to be generated: 10000, 100000
(S) size of the hash table: 11, 101, 1009

Report (Submission through Canvas): Include the following and submit as one single PDF file.
(1 - 25 pts) Provide a description of your optimization strategy and explain how it could reduce the run time at the cost of additional space, if any.
(2 - 45 pts) The implementation code of the algorithm without and with optimization.
(3 - 15 pts) Present a table (the values of N as rows and the values of S as columns) for each of the average largest sequence length and the average detection time (in milliseconds).
(4 - 15 pts) Discuss the impact of the size of the hash table on the average detection time.