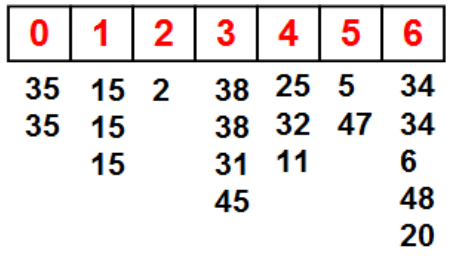
CSC 228 Data Structures and Algorithms, Fall 2018
Instructor: Dr. Natarajan Meghanathan

Project 6: Hash table Tradeoff Analysis: Average Insertion Time vs. Load Imbalance Index

Submission: Submit everything together as one PDF or Word file in Canvas.
Due: October 16th by 11.59 PM

You are given the code to construct a hash table of size 'm' for an array of 'n' integers. The hash table is an array of singly linked lists.

Your task in this project is to extend the code in the main function as well as the code in the Hashtable class to determine the following for different values of the hash table size (m):
(1) average time per insertion (measured in nanoseconds) in the hash table
(2) load imbalance index, a measure of the efficiency in using the memory allocated for the hash table.



Explanation of Load Imbalance Index: The "Load Imbalance Index" for a hash table is calculated as the ratio (Lmax-Lmin) / (Lmax + Lmin), where Lmax and Lmin are respectively the maximum and minimum lengths of the linked lists in the hash table. For example, consider the structure of a hash table of size 7 that indexes 20 elements overall (as shown in the figure).

The maximum size for any linked list in the hash table is 5 (the linked list at index 6) and the minimum size for any linked in the hash table is 1 (the linked list at index 2). Hence, the Load Imbalance Index for the hash table in the example is (5 - 1) / (5 + 1) = 4/6 = 0.67.

For your project, the number of integers (n) to be inserted is 100000 and the maximum value for an integer is 50000. The array of integers to be inserted is randomly generated, as given in the code.

As we discussed in class, the hash table size (m) is a prime integer.
The prime integers (m) to be used for this project are: 11, 29, 47, 71, 173, 281, 409, 541, 659, 809, 941, 1069, 1223 ,1373, 1511, 1657, 1811, 1987, 2129.

Tabulate the results for hash table size (m) vs. {average time per insertion (in nanoseconds), load imbalance index}.

Plot the results in Excel with the hash table size (m) in the X-axis and the average insertion time, load imbalance Index in the Y-axis. You could plot them separately (i.e., hash table size vs. average time per insertion and hash table size vs. load imbalance index in separate charts} or together with primary and secondary Y-axes (i.e., the average time per insertion as the primary Y-axis and the load balance index as the secondary Y-axis in the same Excel chart).

Analyze the results to infer about the impact of the hash table size on the average time per insertion and the load imbalance index and the tradeoff between the two metrics

Submission (as one PDF or Word file submitted through Canvas):
(1) The entire code with the main function and the hash table class enhanced to determine the average time per insertion and load imbalance index.
Your code accept the hash table size as input or could be automated to run for the given set of prime integers as the hash table sizes.
(2) Screenshots of your execution for any four of the prime integers from the above list.
(3) Table presenting the results for the hash table size (m) vs. average time per insertion (in nano seconds) and load imbalance index for all the given values of the hash table size.
(4) Excel chart(s) for the data in (3) and your inference about the impact of the hash table size (m) on the average time per insertion and load imbalance index. Is there any tradeoff between the average time per insertion and load imbalance index? Explain.