Data Type Graphs

Mac Decker

December 8, 2019

1 Introduction

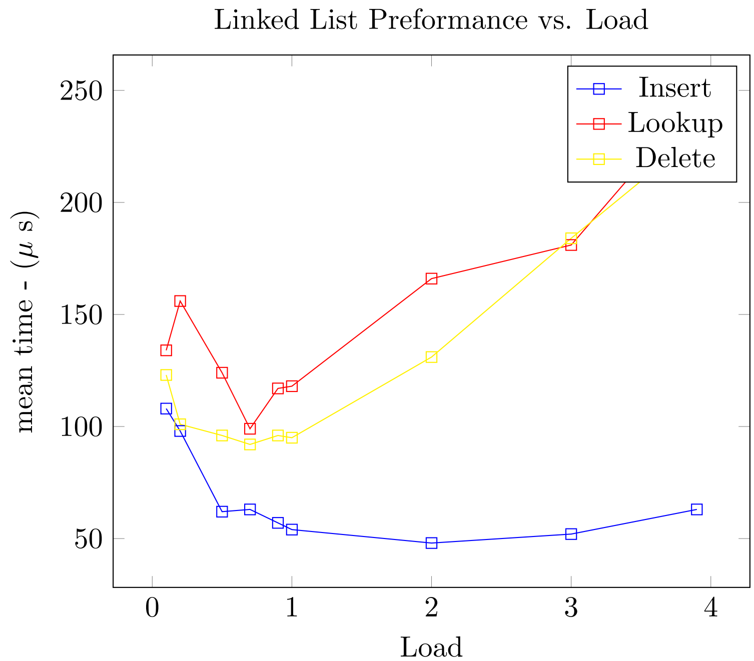
Given two different hashing algorithms and four types of collision resolution: linear probing, chaining with a linked list, chaining with a binary search tree and cuckoo hashing; our goal is to compare the time performance of the insertion, deletion and lookup of a group of 100 elements at various load factors.

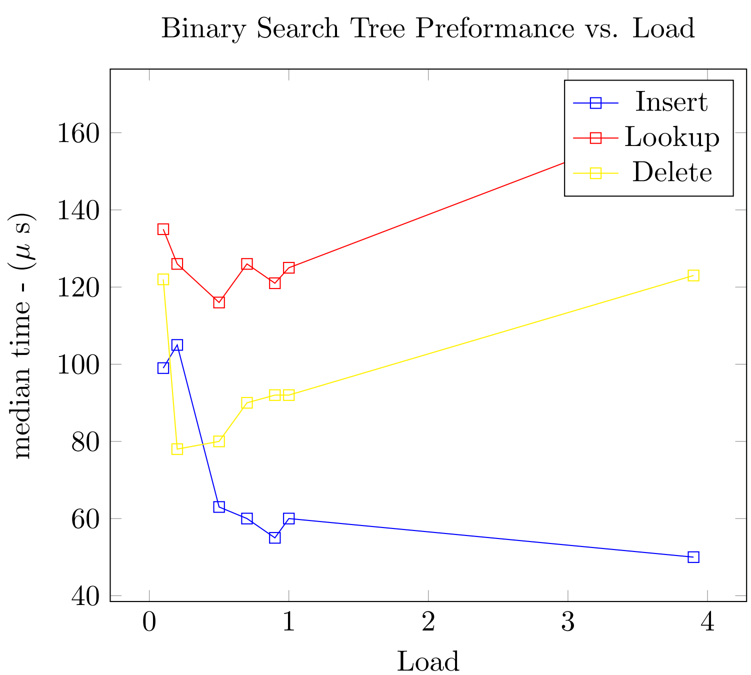
2 Procedure

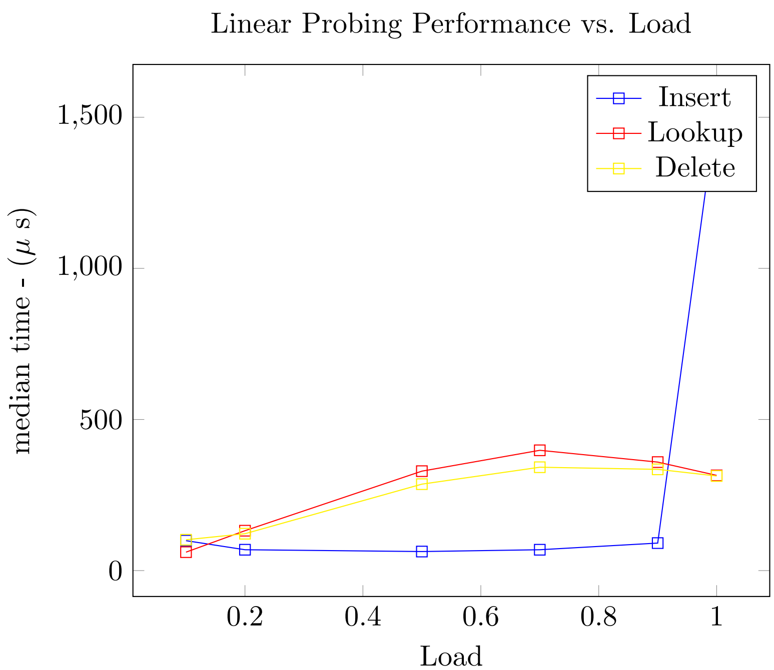
We will use an array of binary search trees with a left and right pointer to store our hashed data for each type of collision resolution.

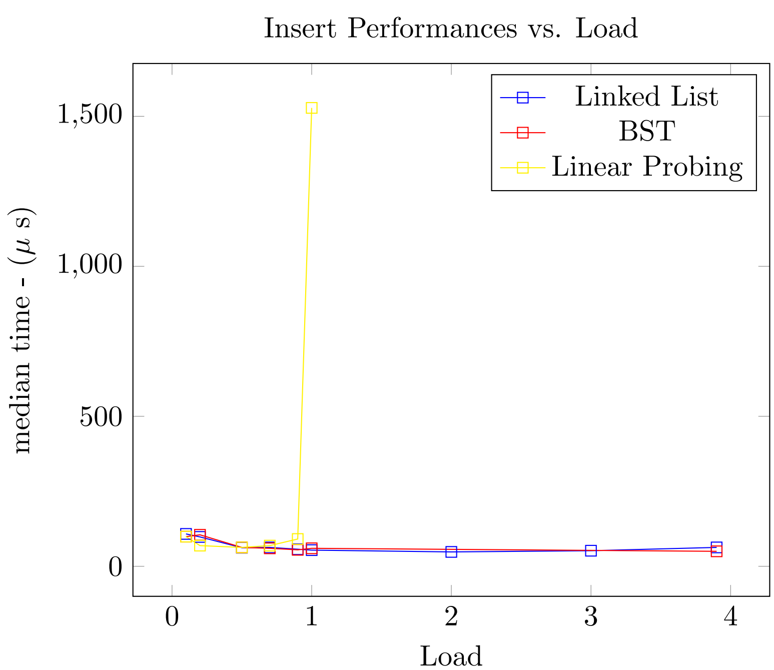
- The linear probing implementation only needs the base array to store our data.
- Utilizing only the left branch will give us the functionality of a linked list.
- Adding the right branch and forcing balancing will give us the functionality of our binary search tree.
- For the cuckoo hashing we first try to put the hashed value in our array and if there is a collision we try the second hash function, this so far only requires the base array. If there is a collision again at the second hash value, the new item overwrites the original collision, and the overwritten number is moved to the second hash value. This can cause a cycle if something would eventually move itself. Keeping track of what moved an item and what it moved in the left and right pointers of the binary search tree will allow us to know when we have a loop, if an item shows up twice on one branch, which will let us know when to resize the array.

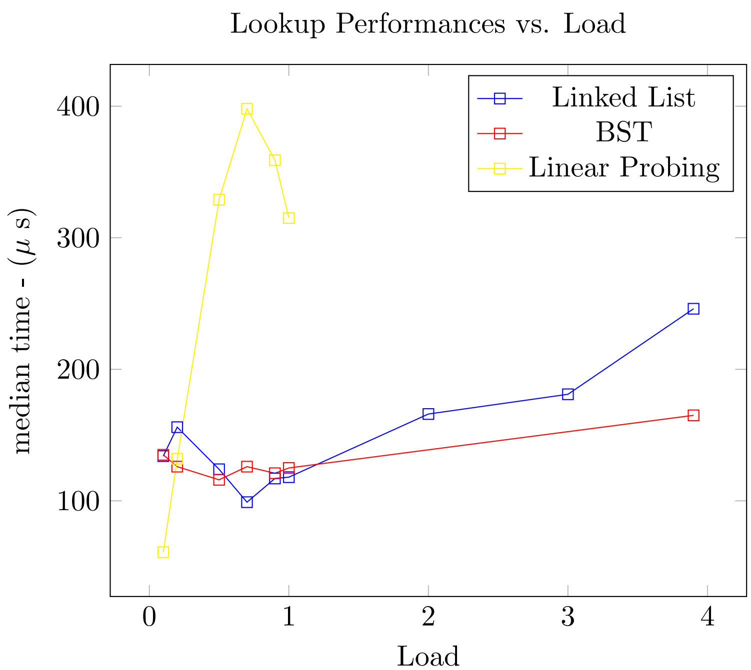
3 Graphs

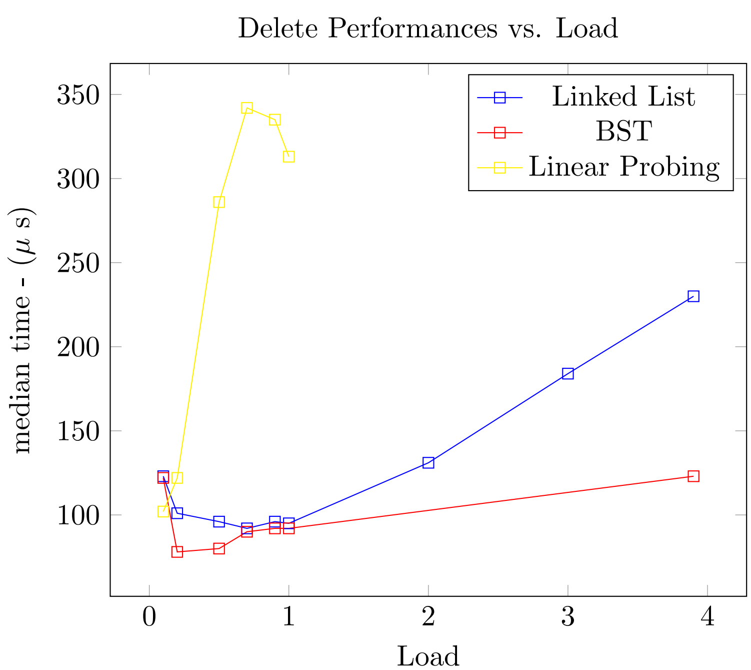

Linked List Preformance vs. Load
Insert
Lookup
Delete
mean time - (μ s)
250
200
150
100
50
0
1
2
3
4
Load


Binary Search Tree Preformance vs. Load
Insert
Lookup
Delete
median time - (μ s)
160
140
120
100
80
60
40
0
1
2
3
4
Load


Linear Probing Performance vs. Load
Insert
Lookup
Delete
median time - (μ s)
1,500
1,000
500
0
0.2
0.4
0.6
0.8
1
Load


Insert Performances vs. Load
Linked List
BST
Linear Probing
median time - (μ s)
1,500
1,000
500
0
0
1
2
3
4
Load





4 Results

A close analysis of the graphs shows a few things.

- Inserting a value is faster than deleting and searching, unless trying to insert into a full array.
- Linear probing scales the worst with load.
- Binary search tree outperforms linked list for collision resolution.

The decrease in linear probings performance after the load was around .8 is unexpected, as well as the increase in performance time at smaller loads for all of the resolution types. A larger data set, and better equipment, would help us see if the numbers are consistent.