Joel Turbi

Dr. Zavala

CS 312 –Analysis of Algorithms

Lab Assignment 9

11/06/2019

1. What does "perfect hashing" mean?

* Perfect hashing is a hash function that maps each different key to a distinct integer. Moreover, a hash table with “guaranteed” no collision involved.

1. Assuming perfect hashing, when should we prefer BST over Hash Tables?

* If we are assuming perfect hashing, BST should be preferred over Hash Tables if 1) all keys are in sorted order. 2) If we are concerned with finding a given value from a specific range (doing range queries). 3) Easier to implement when compared with a hash table.

1. What happens if there is no perfect hashing? What can be said about the running time of the operations above?

* If there is no “perfect hashing” due to collision, then all element in hash table must be rehashed. This insertion operation takes O(n).

1. What can be said regarding storage and the size of the input with both, BST's and Hash Table's?

* It can be said that as the size of the input grows it for a Hash Table there is likely a probability that collision could occur. As for BST’s it is guaranteed to have no collision when inserting. As for the space needed is the same size as the size of the input for both.
* Hash tables & BTS’s both use a dynamic array to store data.

1. In summary, which one is better, Hash Tables or BST's?

* There is no real way to determine whether a Hash tables or BST would be better over the other. This would be dependent on the number of inputs/requirements needed for a given application. Nevertheless, assuming “perfect hashing” and/or if we are not concerned with order of the values, then a Hash Table would be preferred over BST’s due to its time complexity O(1).