Trees vs Hash Table

This lesson highlights the differences between trees and hash tables.

We'll cover the following

Comparison Between Trees and Hash Tables#

Both of these data structures can be used for the same job, but their performance would vary based on the nature of your program. Let's take a look at some of the factors we need to keep in mind when deciding the appropriate data structure.

Basic Operations#

On average, hash tables can perform search, insertion, and deletion in constant time whereas trees usually work in *O(log n)*. However, in the worst case, the performance of hash tables can come down to *O(n)* where **n** is the total number of hash entries. An AVL tree would maintain *O(log n)* even in the worst case.

Hash Function#

An efficient hash table requires a smart hash function that would distribute the keys over all the space that is available to us. A tree is simpler to implement in this regard as it accesses extra space only when needed and no hash function is required to optimize its structure.

Order of Data#

If our application needs data to be ordered in a specific sequence, trees would prove more useful because a BST or an AVL tree maintains order. Hash tables are the smarter choice if your data can be stored randomly.

In the following lesson, we will discuss the difference between a dictionary and a set in python.

Back
A Quick Overview of Hash Tables

Next
Dictionary vs Set

Mark as Completed

Report an Issue