A BRIEF OVER LINEAR SEARCH AND BINARY SEARCH - PERFORMANCE

Let is take an example very practical with 1024 objects to be searched.

In linear search: the maximum search is 1024 (must go through all objects) and the minimum is 1 (finds the searched object first); on average, 512 searches.

In binary: maximum search of 10 (it is the logarithm in base 2 of the number of objects) and minimum of 1 (first find); on average, 5.5 searches. Binary search is faster than linear unless the number of objects is short. Of course, these numbers may vary in practice depending on the distribution of objects sought. Furthermore, in binary search the objects need to be sorted. However, this sorting is done once. If doubling the number of objects to 2048, the average changes to 6 in binary and to 1024 in linear. That is, the greater the number of objects, binary search becomes increasingly brutally advantageous to the point where linear search becomes prohibitive in terms of search time.

TREES AND BINARY SEARCH TREES (BST)

Tree is a widely used data structure that emulates a hierarchical tree structure with a set of linked nodes.

A node is a structure which may contain a value, a condition, or represent a separate data structure. Each node in a tree has zero or more child nodes, which are below it in the tree. Trees are drawn growing downwards. A node that has a child is called the child's parent node (or ancestor node, or superior). A node has at most one parent.

On the other hand, a binary tree is a tree in which each node has at most two children. More precisely, a node can have a left child or not, and have a right child or not. To make the properly analysis about its performance we need to considerate what is a full binary tree. A full binary tree is a binary tree in which each internal node has exactly two children. The performance is highly increased when the tree is full.

PERFORMANCE

The find, insert and delete algorithms start at the tree root and a follow path down to, at worst case, the leaf at the very lowest level. The total number of steps of these algorithms is, therefore, the largest level of the tree, which is called the depth of the tree.

The best running time occurs when the binary search tree is full – in which case the search processes used by the algorithms perform like a binary search. Let's verify this. A full binary tree is one in which nodes completely fill every level. Just one more concept to understand the worst and average cases, a full binary search tree is said to be balanced because all node's proper descendants are divided evenly between its left and right subtrees.

So, we have this example:

A full binary tree with 0, 1 and 2 depths. Writing down the example:

Level 0 – 1 node

Level 1 – 2 nodes

Level 2 – 4 nodes and so on.

This way it will have 2^d nodes at level d.

.: $n = 2^1 + 2^2 + 2^3 + \ldots\ldots + 2^d = 2^{(d + 1)} - 1$

Solving d:

$n + 1 = 2^{(d +1)}$

$\log (n+1) = \log (2) (d+1)$

$\log (n+1) = d + 1$

$d = \log (n+1) - 1$

$d = [\log(n)]$

Which means the depth of a binary search tree with n nodes can be no less than log(n) and so the running time of the find, insert and delete algorithms can be no less than log(n).

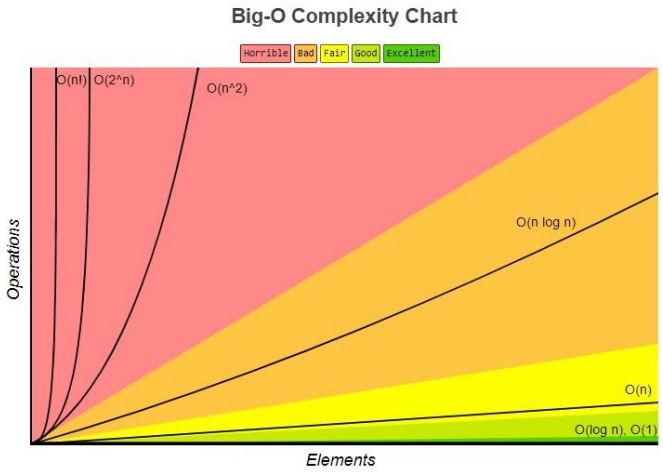

Fig.1 (Big-O Cheat Sheet-https://www.bigocheatsheet.com)

If the binary search tree becomes more and more unbalanced, the performance of the find, insert and delete algorithms degrades until reaching the worst case of O(n), where n is the number of nodes in the tree.

The average time of a binary search tree which is constructed through a random sequence of insertions is O(log n).

Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Fig.2 (Big-O Cheat Sheet-https://www.bigocheatsheet.com)

APLICATIONS LISTS

Binary search trees are used in many search applications where data is constantly coming in/out, such as “map” objects and “set” in multi-language libraries.

Binary space partitioning, which is a BST is used in almost all 3D video games to determine which objects need to be rendered. BSP trees are often used by 3D video games, particularly first-person shooters and those with indoor environments.

Binary Trees are used on almost all high-bandwidth routers to store router tables as a Binary Attempts.

Hash Trees which is another type of binary tree, it is used in p2p programs and specialized image signatures where a hash needs to be checked but the entire file is not available.

Heaps - Used in implementing efficient priority queues, which in turn are used to schedule processes in many operating systems, quality of service in routers, and A* (path location algorithm used in AI applications including robotics and video games). Also used in heap-sort.

Huffman coding is used in compression algorithms such as those used in .jpeg and .mp3 file formats.

The GGM tree which is used in cryptographic applications to generate a tree of pseudo-random numbers.Syntax tree - Built by compilers and (implicitly) calculators to parse expressions.

Treap which is randomized binary search tree is used in wireless networks and memory allocation.