Preorder: c->l->r, Inorder: l->c->r, Postorder: l->r->c

Binary Search Tree (BST): left < parent, right > parent, no duplicate nodes

Avl tree (self-balancing BST): difference of left and right subtrees is max 1

Height of tree: floor($\log_2 n$), where n = num of nodes

Big-oh: upper bound

Big-theta: upper and lower bound

Big-omega: lower bound

In2post: lab 5 qn 1, use stack, infix->postfix, precedence(peek(s))>=precedence(c)

In2pre: lab 5 qn 3, use stack, reverse infix -> revInfix-> revPrefix -> reverse revPrefix -> prefix, precedence(peek(s))>precedence(c)

Calculate postfix: lab 5 qn, use stack, if operand push, if operator pop top 2, calculate and push

Calculate prefix: same as calculate postfix but start from the back

Sequential search: worst and avg O(n)

Binary search: worst and avg O($\log_2 n$)

Closed addressing: linked list, worst O(n), avg O(a), where a is load factor (avg num of elements in each linked list)

Open addressing:

- Linear probing: $k, i = k + i \bmod h$ where $i \in [0, h-1]$
- Quadratic probing: $k, i = k + c1i + c2i^2 \bmod h$ where $c1$ and $c2$ are constants, $c2 \neq 0$
- Double hashing: $k, i = k + iD(k) \bmod h$ where $i \in [0, h-1]$ and $D(k)$ is another hash function

BFS (one level at a time, level order traversal):

The worst-case time complexity for BFS is $\Theta(|V| + |E|)$ if graph is represented by an adjacency lists. $\Theta(|V|^2)$ if graph is represented by an adjacency matrix.

queue (normally) Breadth First Search Tutorials & Notes | Algorithms | HackerEarth

no recursion

use bfs to find shortest path

DFS (goes as deep as possible before backing up, preorder traversal):

The worst-case time complexity for DFS is $\Theta(|V| + |E|)$ if graph is represented by an adjacency lists. $\Theta(|V|^2)$ if graph is represented by an adjacency matrix

stack (lab 8 qn 1), recursion (lab 8 qn 2)

Depth First Search Tutorials & Notes | Algorithms | HackerEarth

might not find shortest path, but finds a path

adjMatrix: 2d array of |V| rows and |V| cols, |V| is num of vertices in graph, each edge default is 0 (no edge), if 1 means have edge

adjList: array of adjacency lists, each entry is each vertex, each adj List represents what other vertices are connected to this vertex