DATASTRUCTURE & ALGORITHMS

**TOPICS TO BE COVERED**

***DATASTRUCTURES:***

1. Linked Lists
2. Doubly Linked Lists
3. Stacks & Queues
4. Binary Search Trees
5. Hash Tables
6. Graphs

***ALGORITHMS:***

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Merge Sort
5. Quick Sort
6. Tree Traversal

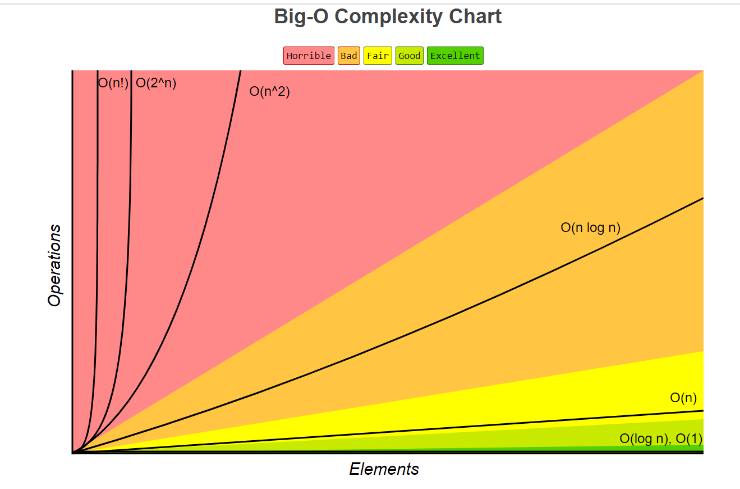
* Breadth First Search
* Depth First Search - Pre Order
* Depth First Search - Post Order
* Depth First Search - In Order

***COURSE TAKEN FROM UDEMY:***

[Java Data Structures & Algorithms + LEETCODE Exercises (udemy.com)](https://dxc.udemy.com/course/data-structures-and-algorithms-java/learn/lecture/27815396#overview)

**BIG O NOTATION**

* Time Complexity – Calculated Based on how long (time) it takes to execute the code
* Space Complexity – Calculated Based on how much space is consumed when code is executed

***CATERGORIES OF BIG – O:***

* *LINEAR TIME – O(n)*
* *LINEAR TIME- DROP CONSTANTS.*
* *QUADRATIC TIME – O(n^2)*
* *DROP NON-DOMINANTS*

***LINEAR TIME – O(n)***

*Examples:*

* *Looping through an element in an array*
* *Searching through a Linked List*

package com.big.o;  
  
public class LinearTime {  
//O(n) - LinearTime Example  
 public static void printItems(int n){  
 for(int i =0; i< n ;i++){  
 System.*out*.println(i);  
 }  
 }  
 public static void main(String[] args){  
 *printItems*(10);  
 }  
}

***LINEAR TIME- DROP CONSTANTS***

*O(Xn), where X may be 1,2,3…etc.*

*n + n = 2n , where we drop constants and it becomes O(n)*

package com.big.o;  
  
public class LinearTime\_dropConstants {  
//O(2n) or O(3n)... O(Xn) = Drop Constants and its O(n) - Linear Time Example  
 public static void printItems(int n){  
 for(int i =0; i< n ;i++){  
 System.*out*.println(i);  
 }  
 for(int j =0; j< n ;j++){  
 System.*out*.println(j);  
 }  
 }  
 public static void main(String[] args){  
  
 *printItems*(10);  
 }  
}

***QUADRATIC TIME – O(n^2)***

***n \* n = n2 🡺 O(n2)***

*EXAMPLES :*

* *Insertion Sort*
* *Selection Sort*
* *Bubble Sort*
* package com.big.o;  
    
  public class QuadraticTime {  
    
   //O(n Exponent of 2) - n \* n = n exponent 2  
    
   public static void printItems(int n){  
   for(int i =0; i< n ;i++){  
   for(int j =0; j< n ;j++){  
   System.*out*.println(i + " " +j);  
   }  
   }  
    
   }  
   public static void main(String[] args){  
    
   *printItems*(10);  
   }  
  }

***DROP NON-DOMINANTS***

***O(n2+ n) = O(n2) ,where + n is dropped***

*If n =100, (n2) = 1000 and n = 100*

*Here* ***(n2) is Dominant*** *and* ***n is Non – Dominant***

package com.big.o;  
  
public class QuadraticTime\_NonDominant\_Drop {  
 /\*O(n2+ n ) = O(n2) ,where + n is dropped  
 If n =100, (n2) = 1000 and n = 100  
 Here (n2) is Dominant and n is Non – Dominant\*/  
 public static void printItems(int n){  
 for(int i =0; i< n ;i++){  
 for(int j =0; j< n ;j++){  
 System.*out*.println(i + " " +j);  
 }  
 }  
 for( int k = 0; k < n ; k++){  
 System.*out*.println(k);  
 }  
 }  
 public static void main(String[] args){  
 *printItems*(10);  
 }  
}