Data Structures: Sorting and Searching

simplilearn

A Day in the Life of a Full Stack Developer

All the desired results are achieved in the sprint. The team has developed various components through pair programming and continuous delivery. The sprint is marked as completed. The team has collected various issues from the Test Engineers team. One of the issues raised is about the performance of the application. Test Engineers argued that the program logic could be changed or improved by using appropriate sorting or search techniques. This issue is considered during the sprint retrospective.

Since Joe developed most of the features of the application, he is asked to fix the bug raised by the Test Engineers. Joe has to decide the best searching and sorting algorithms and data structures so that the performance of the application can be enhanced.

In this lesson, we will learn how to solve this real-world scenario to help Joe complete his task effectively and quickly.

Learning Objectives

By the end of this lesson, you will be able to:

- Implement the suitable searching algorithms
- Implement the suitable sorting algorithms
- Implement the concepts of graphs in your project
- Predict the complexity of your program using asymptotic notations

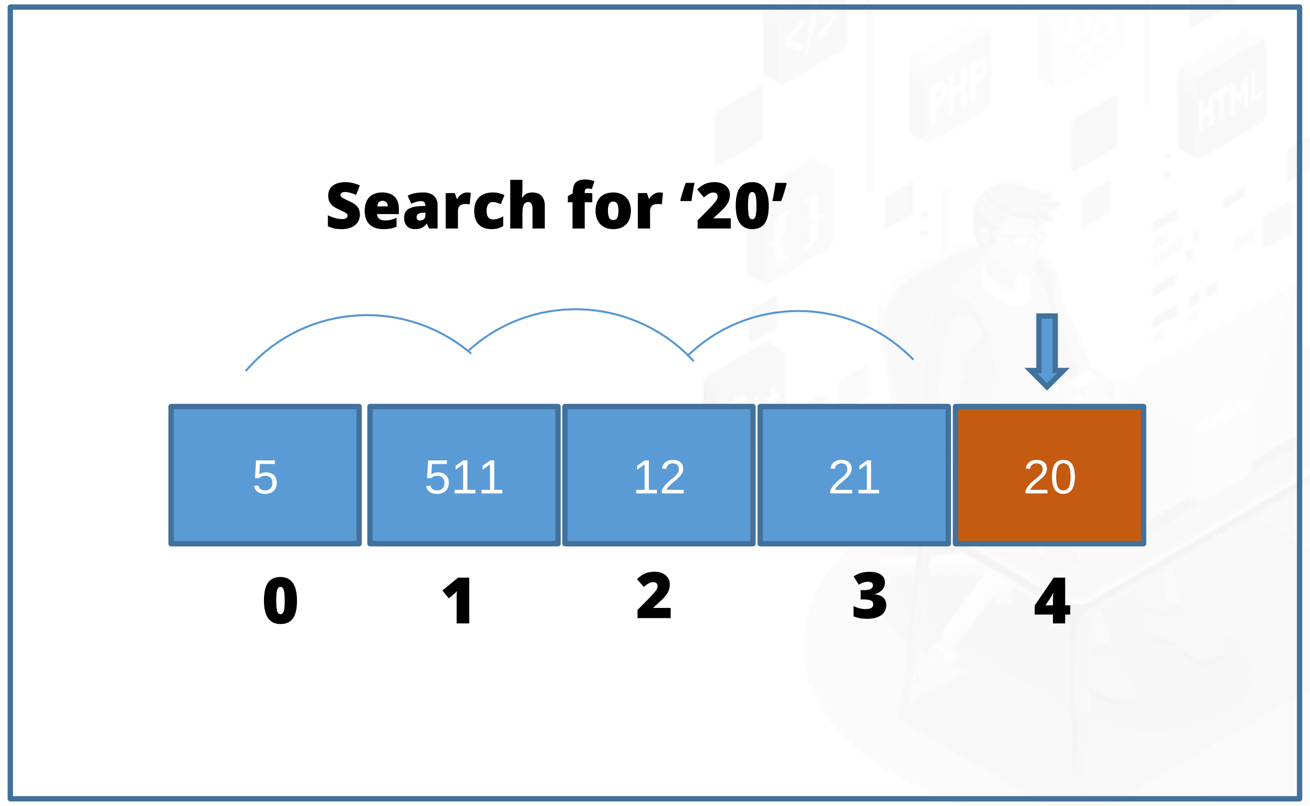
Linear Search

Linear Search

Linear search is a simple algorithm. It can be used to search an element or a value in a given array by going through it from the beginning till the element is found.

Approach to linear searching:

- Start from the leftmost element of an array.
- Compare the element, say X, with each element of the array.
- If the X matches with an element, return the index.
- If the X does not match with any of the elements, return -1.



Linear Search

Duration: 15 min.

Problem Statement:

You are given a project to demonstrate the workflow of a linear search algorithm.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate the workflow of a linear search algorithm:

1. Create a Java project in your IDE
2. Write a program in Java to demonstrate the workflow of a linear search algorithm.
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

FULL STACK

Binary Search

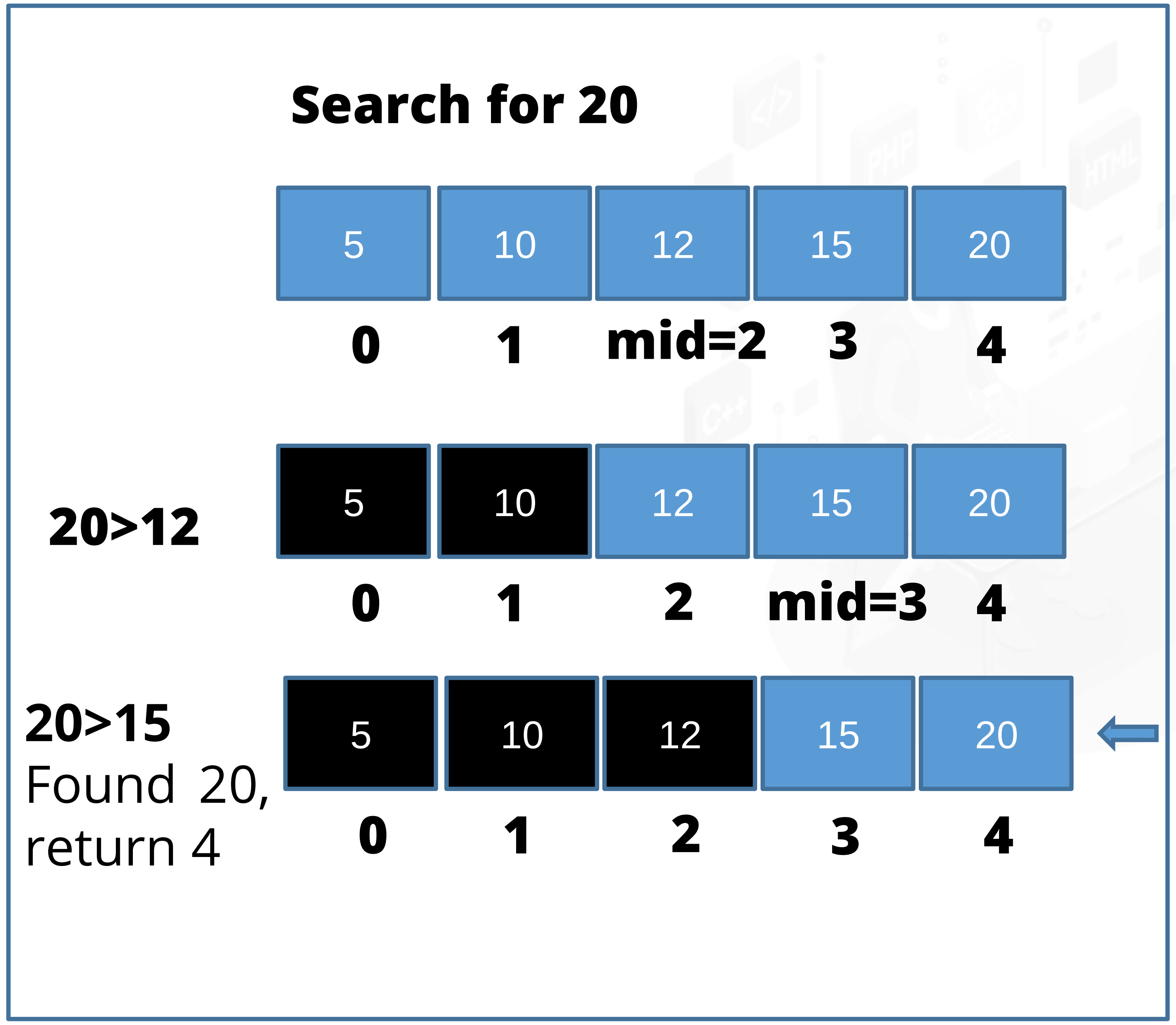
simplilearn

Binary Search

Binary search is also referred to as the logarithmic search. It finds the position of a particular value within a sorted array.

Approach to binary searching:

- Compare an element, say X, with the middle element.
- If X matches the middle element, return the *mid* index.
- If X is greater than the *mid* element, it can lie only in the right half of the subarray after the *mid* element.
- Otherwise, X can recur for the left half of the array, assuming the value is smaller than the *mid* element.



Binary Search

Duration: 15 min.

Problem Statement:
You are given a project to demonstrate the workflow of a binary search algorithm.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate the workflow of a binary search algorithm:

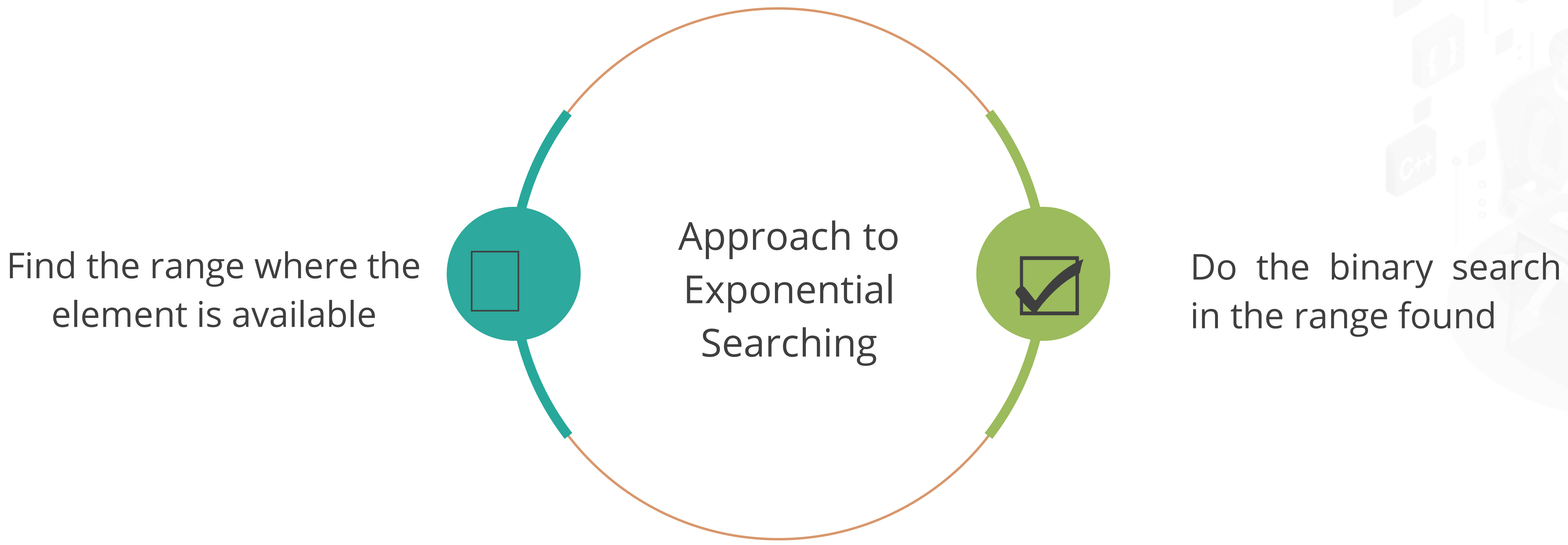
1. Create a Java project in your IDE
2. Write a program in Java to demonstrate the workflow of a binary search algorithm
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

FULL STACK

Exponential Search

Exponential Search

- Exponential search is also called the doubling search. It helps in searching through a sorted and unbounded array list for a specified input value.
- The first step determines a range in which the search key will reside if it is on the list. The second step performs the binary search on the array.


Find the range where the element is available
Approach to Exponential Searching
Do the binary search in the range found

Exponential Search

Duration: 15 min.

Problem Statement:

You are given a project to demonstrate the workflow of an exponential search algorithm.

Assisted Practice: Guidelines

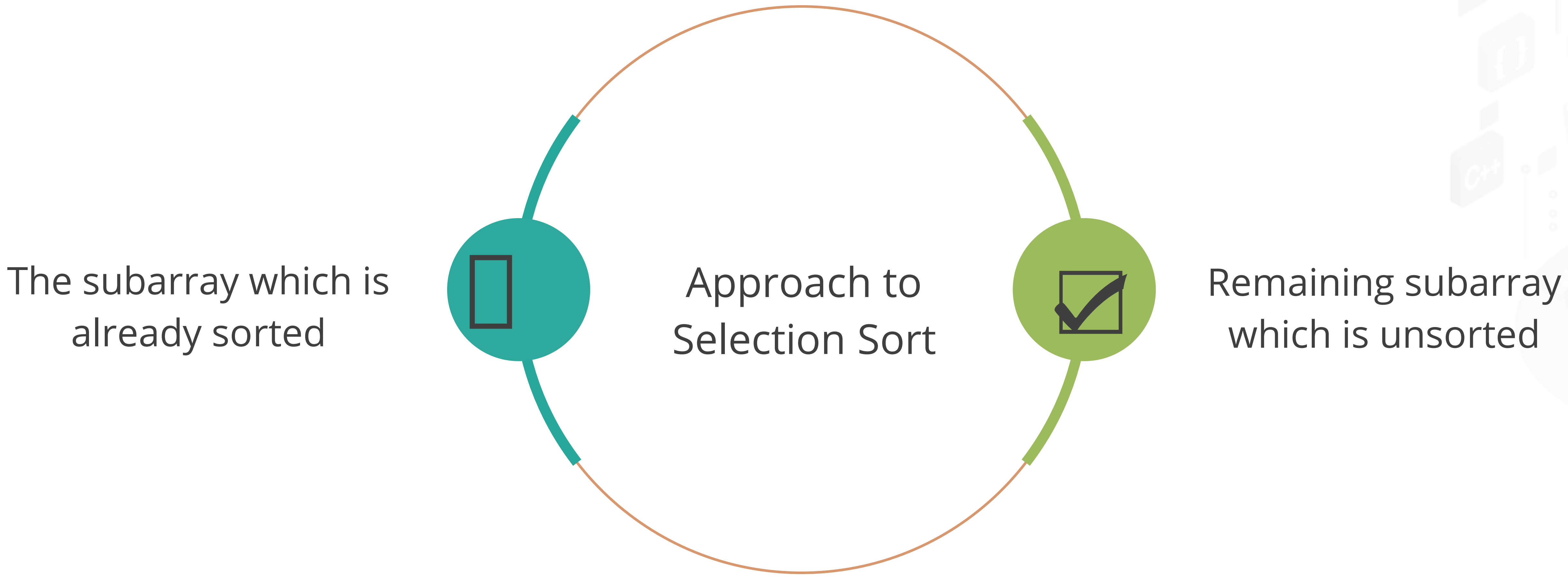
Steps to demonstrate the workflow of an exponential search algorithm:

1. Create a Java project in your IDE
2. Write a program in Java to demonstrate the workflow of an exponential search algorithm
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

Selection Sort

Selection Sort

Selection sort algorithm sorts an array by repeatedly finding the smallest number from the unsorted part and appending it at the beginning.


The subarray which is already sorted
Approach to Selection Sort
Remaining subarray which is unsorted

Selection Sort

Duration: 15 min.

Problem Statement:

You are given a project to demonstrate the workflow of a selection sort algorithm.

Assisted Practice: Guidelines

Steps to demonstrate selection sort algorithm:

1. Create a Java project in your IDE
2. Write a program in Java to demonstrate the workflow of a selection sort algorithm
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

Bubble Sort

Bubble Sort

Bubble sort or sinking sort, is the simplest sorting algorithm. It works by repeatedly swapping the adjacent elements if they are in an incorrect order.

```
procedure bubbleSort(A : list of sortable items )
n = length(A)
repeat
swapped = false
for i = 1 to n-1 inclusive do
/* if this pair is out of order */
if A[i] > A[i+1] then
/* them and remember something changed */
 (A[i], A[i+1] )
swapped = true
end if
end for
until not swapped
end procedure
```

Bubble Sort

Duration: 15 min.

Problem Statement:

You are given a project to demonstrate the workflow of a bubble sort algorithm.

Assisted Practice: Guidelines

Steps to demonstrate bubble sort algorithm:

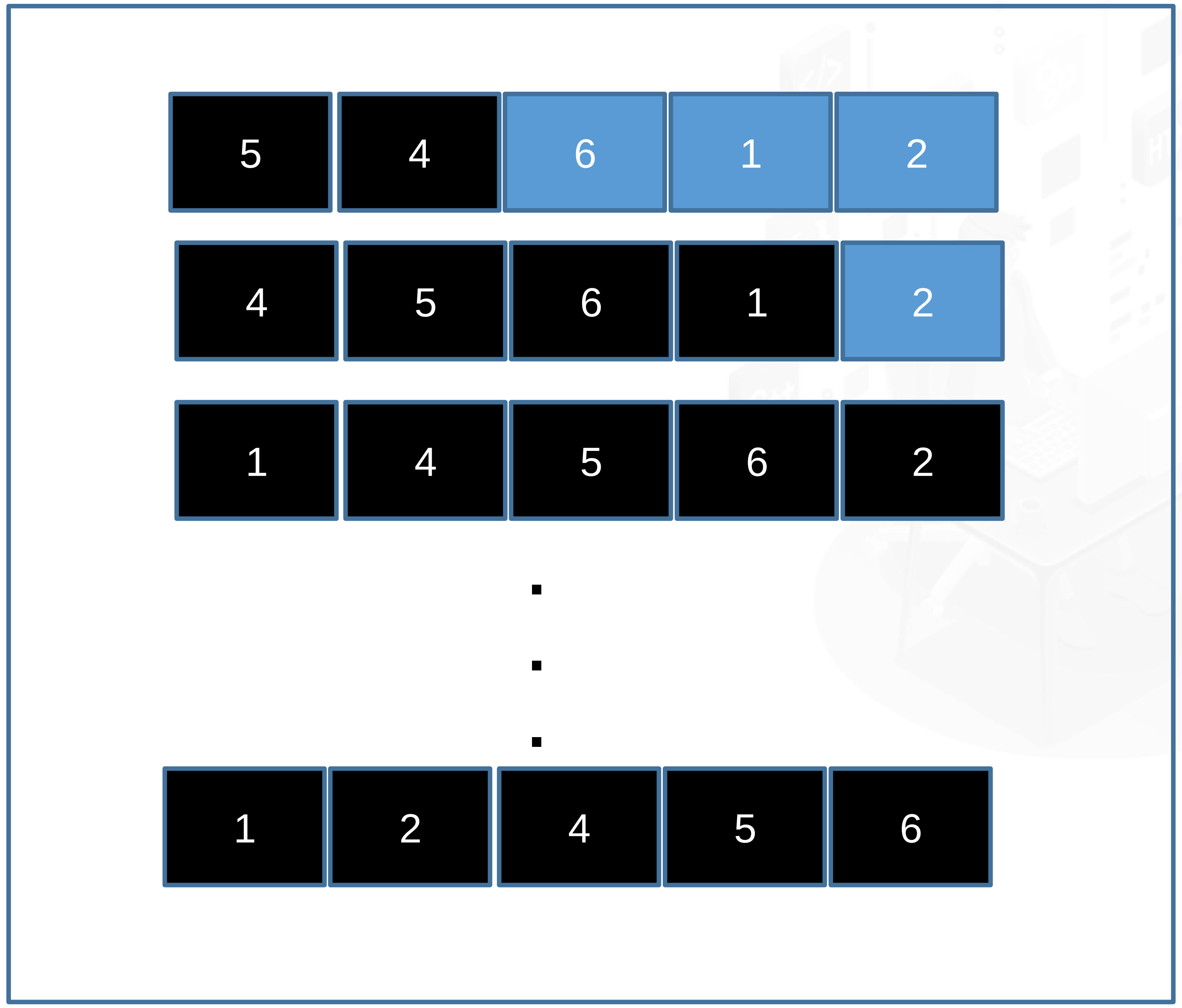
1. Create a Java project in your IDE
2. Write a program in Java to demonstrate the workflow of a bubble sort algorithm
3. Initialize the .git file
4. Add and commit the program files
5. Push the code the your GitHub repositories

Insertion Sort

Insertion Sort

Insertion sort builds the final sorted array by moving the elements to a sorted sublist. It is less efficient for a large set of values.

```
i ← 1
while i < length(A)
    j ← i
    while j > 0 and A[j-1] > A[j]
        swap A[j] and A[j-1]
        j ← j - 1
    end while
    i ← i + 1
end while
```



Insertion Sort

Duration: 15 min.

Problem Statement:
You are given a project to demonstrate the workflow of an insertion sort algorithm.

Assisted Practice: Guidelines

Steps to demonstrate insertion sort algorithm:

1. Create a Java project in your IDE
2. Write a program in Java to demonstrate the workflow of an insertion sort algorithm
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

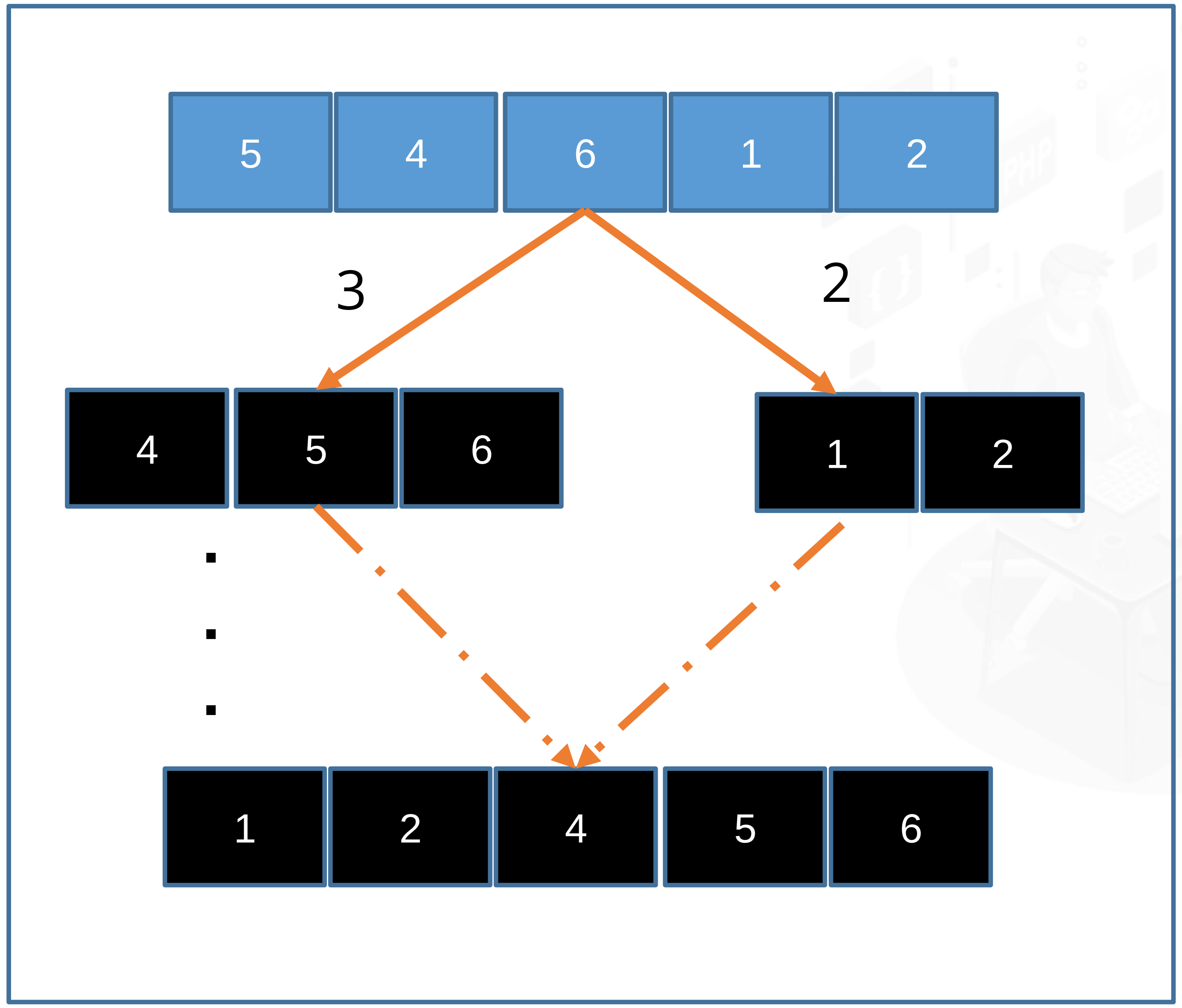
FULL STACK

Merge Sort

Merge Sort

Merge sort is a divide and conquer algorithm. It divides an input array into two halves, calls itself for the two halves, and merges the two sorted values until the given entire array is sorted.

```
MergeSort(arr[], l, r)
If r > l
1.Find the middle point to divide the array
into two halves:
        middle mid = (l+r)/2
2. Call mergeSort for first half: Call
        mergeSort(arr, l, mid)
3. Call mergeSort for second half: Call
        mergeSort(arr, mid+1, r)
4. Merge the two halves sorted in step 2 and
3: Call merge(arr, l, mid, r)
```



Merge Sort

Duration: 15 min.

Problem Statement:
You are given a project to demonstrate the workflow of a merge sort algorithm.

Assisted Practice: Guidelines

Steps to demonstrate the workflow of a merge sort algorithm:

1. Create a Java project in your IDE
2. Write a program in Java to demonstrate the workflow of a merge sort algorithm
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

Quick Sort

Quick Sort

Quick sort is a divide and conquer algorithm. It chooses an element as a pivot and divides the given array around the chosen pivot.

Choosing the pivot:

- Always pick the first element as pivot.
- Always pick last element as pivot.
- Pick a random element as pivot.
- Pick median as pivot.

```
quickSort(arr[], low, high)
{
        if (low < high)
        {
/* pi is partitioning index, arr[pi] is now at
right place */
pi = partition(arr, low, high);
quickSort(arr, low, pi - 1); // Before pi
quickSort(arr, pi + 1, high); // After pi
        }
}
```

Quick Sort

Duration: 15 min.

Problem Statement:
You are given a project to demonstrate the workflow of a quick sort algorithm.

Assisted Practice: Guidelines

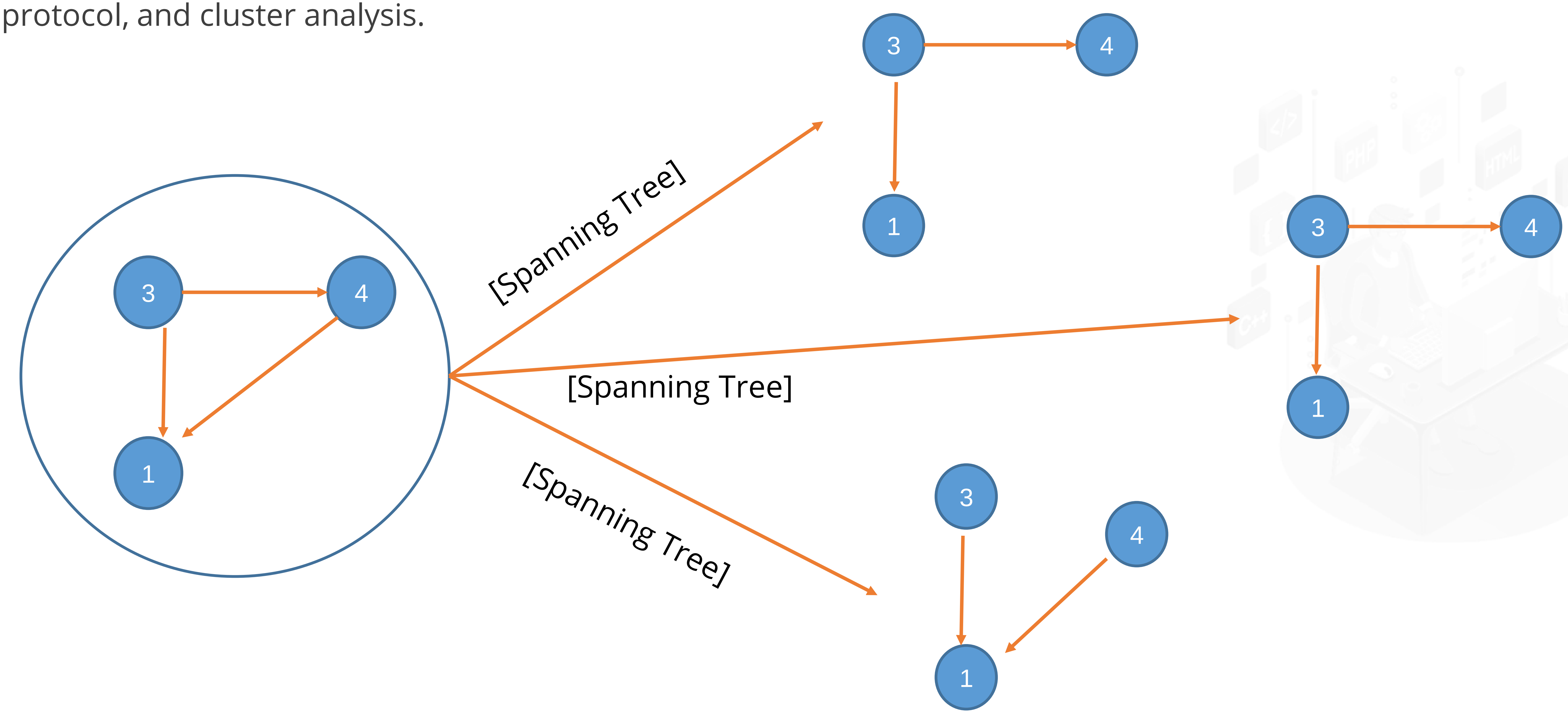
Steps to demonstrate the workflow of a quick sort algorithm:

1. Create a Java project in your IDE
2. Write a program in Java to demonstrate the workflow of a quick sort algorithm
3. Initialize the .git file
4. Add and commit the program files
5. Push the code to your GitHub repositories

Graphs

Minimum Spanning Tree

A spanning tree is a subset of a graph that covers all the vertices with minimum edges. It cannot have cycles and is disconnected. It is used in civil network planning, computer network routing protocol, and cluster analysis.



Index Mapping

To search any element, say x, in an array:

- If x is a nonnegative value, check if hash[x][0] is 1. If hash[x][0] is one, then the number is present.
- If x is a negative value, take the value of x and check if hash[x][1] is 1. If hash[x][1] is one, then the number is present.

Assign all the values of the hash matrix as 0. Traverse the given array:

If the element *ele* is nonnegative, assign hash[*ele*][0] as 1.

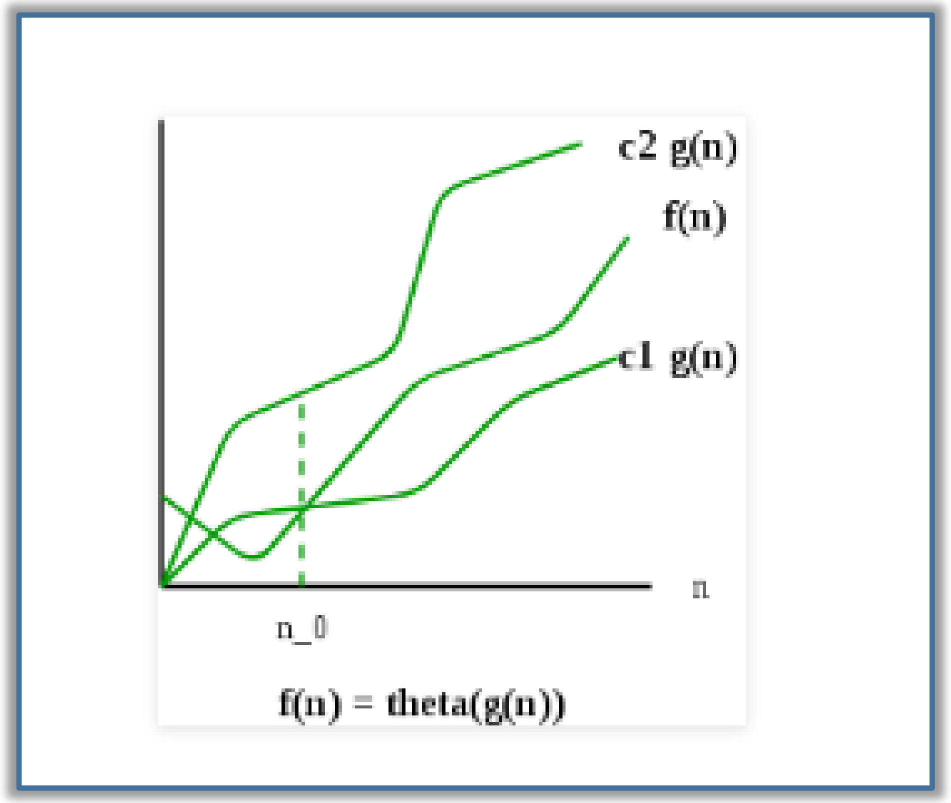
If not, take the absolute value of *ele* and assign hash[*ele*][1] as 1.

Performance and Complexity

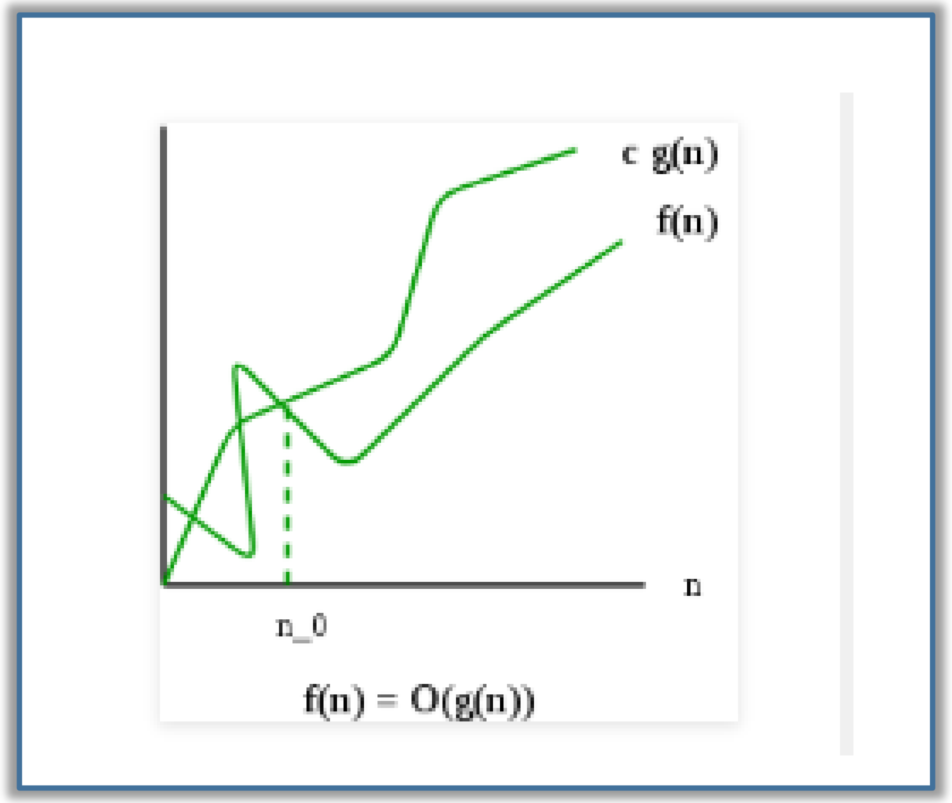
Asymptotic Notations

Asymptotic notations are standard notations used to analyze the complexity of any algorithm in terms of time and space. The following notations are predominantly used:

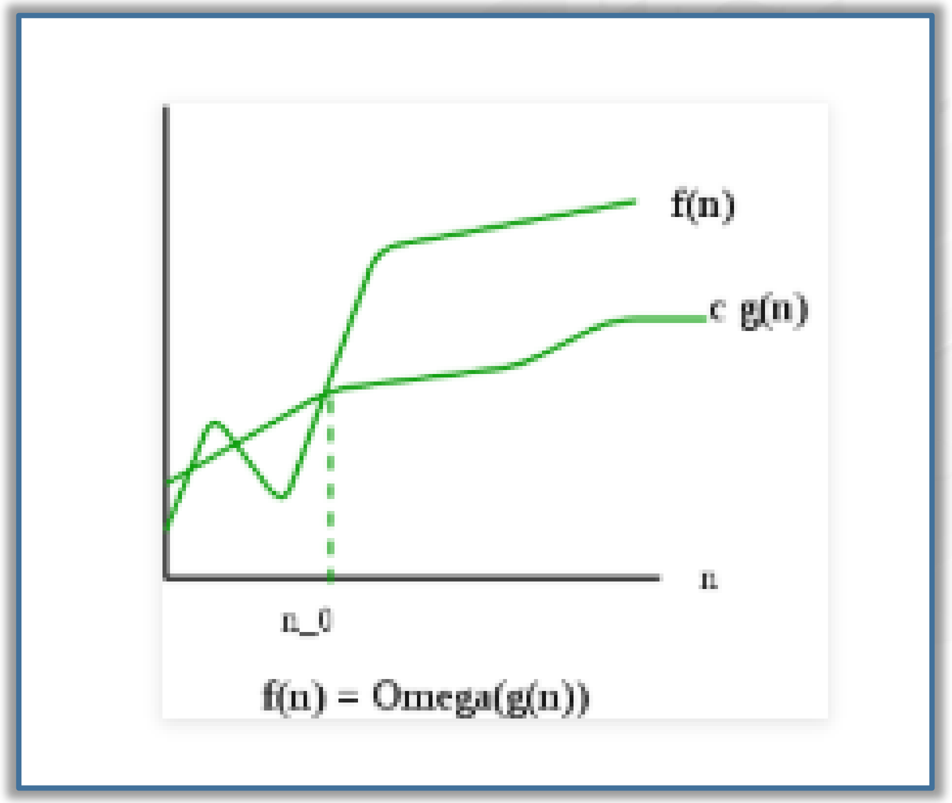
***Θ* Notation**



***Big O* Notation**



***Ω* Notation**



Source: https://www.geeksforgeeks.org/analysis-of-algorithms-set-3asymptotic-notations/

Θ Notation

The time for linear search is C1.n+C2.
C1 The sum of the times of the computations in one loop iteration.
C2 Extra overhead time for setting up the for-loop.

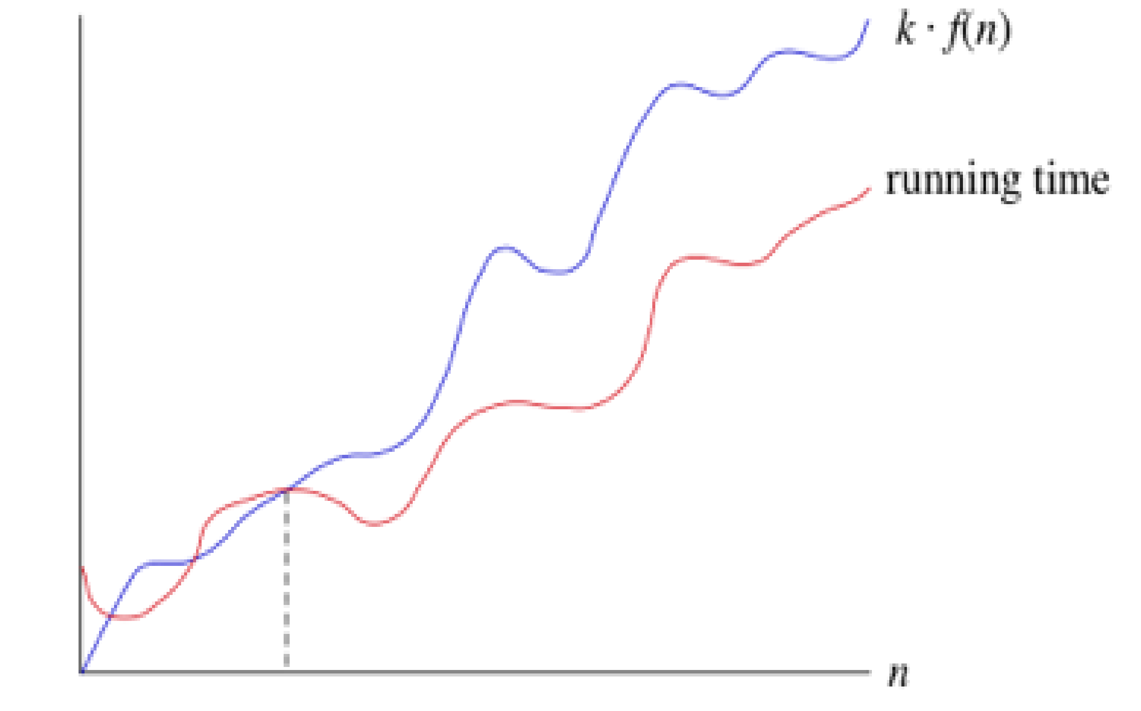
Θ(n) represents that:
If n gets larger, the running time is at least j1.n and at most j2.n for constants j1 and j2.

```
public void linear(array,targetValue){
        int n = array.length;
for(int i=0; i<n; i++)
        {
        i = i++;
        return i;
        }
}
```

Big O Notation

The *Big O* notation defines an upper bound of an algorithm; it binds a function only from the levels above.

If a running time is O(f(n)) for a function f(n), then for the larger n, the running time is at most k.f(n) for constant k value.



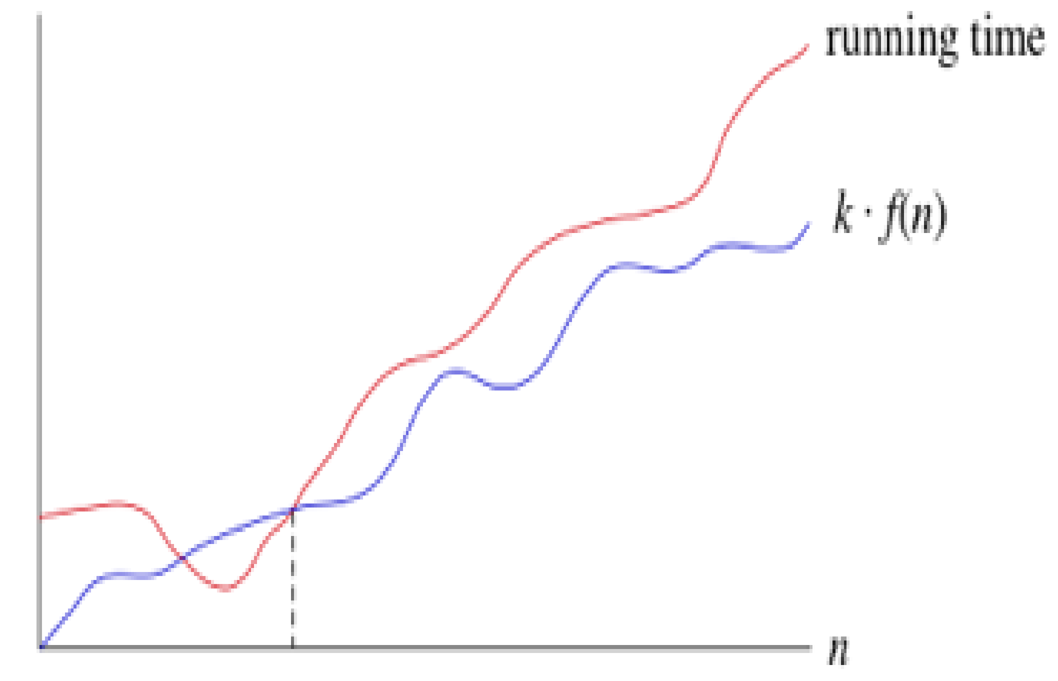
Example: k.f(n)

Source: https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation

Ω Notation

The big Ω notation defines a lower bound of an algorithm; it binds a function only from levels below.

If a running time is Ω(f(n)) for a function f(n), then for the larger n, the running time is at most k.f(n) for constant k value.



Example: k.f(n)

Source: https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation

Key Takeaways

- Searching algorithms are used to check for an element or retrieve an element from a data structure.
- We can use a linear search, binary search, or an exponential search to search the element from the respective data structure.
- Sorting algorithms are used to rearrange the array or a list of elements.
- To analyze the complexity of any algorithm in terms of time and space, asymptotic notations are used. Example: ʘ notation, *Big O* notation, and big Ω notation.

Fix Bugs of the Application

Duration: 30 min.

Problem Statement:
As a developer, you are asked to fix the bugs of the application using appropriate algorithm.

LESSON-END PROJECT

Before the Next Class

You should know:

- Agile and scrum core concepts
- Fundamental concepts of Git and GitHub
- Basics of Java programming
- Implementation of OOPs and Collections
- System defined classes
- Data structures

Virtual Key for Your Repositories

Duration: 240 min.

Project Objective:

As a Full Stack Developer, complete the features of the application by planning the development in terms of sprints and then push the source code to the GitHub repository. As this is a prototyped application, the user interaction will be via a command line.

PHASE-END PROJECT

Background of the Project Statement

Company Lockers Pvt. Ltd. hired you as a Full Stack Developer. They aim to digitize their products and chose LockedMe.com as their first project to start with. You're asked to develop a prototype of the application. The prototype of the application will then be presented to the relevant stakeholders for budget approval.

You Are Asked to Do

- Specification document - Product’s capabilities, appearance, and user interactions
- Number and duration of sprints required
- Setting up Git and GitHub account to store and track your enhancements of the prototype
- Java concepts being used in the project
- Data Structures where sorting and searching techniques are used
- Generic features and three operations:
  - Retrieving the file names in an ascending order
  - Business-level operations:
    - Option to add a user-specified file to the application
    - Option to delete a user-specified file from the application
    - Option to search a user-specified file from the application
    - Navigation option to close the current execution context and return to the main context
  - Option to close the application

The goal of the company is to deliver a high-end quality product as early as possible.

You must use the following

IDE: Eclipse or IntelliJ

Programming language: Java

Git and GitHub

Scrum

Searching and Sorting algorithms

Specification document