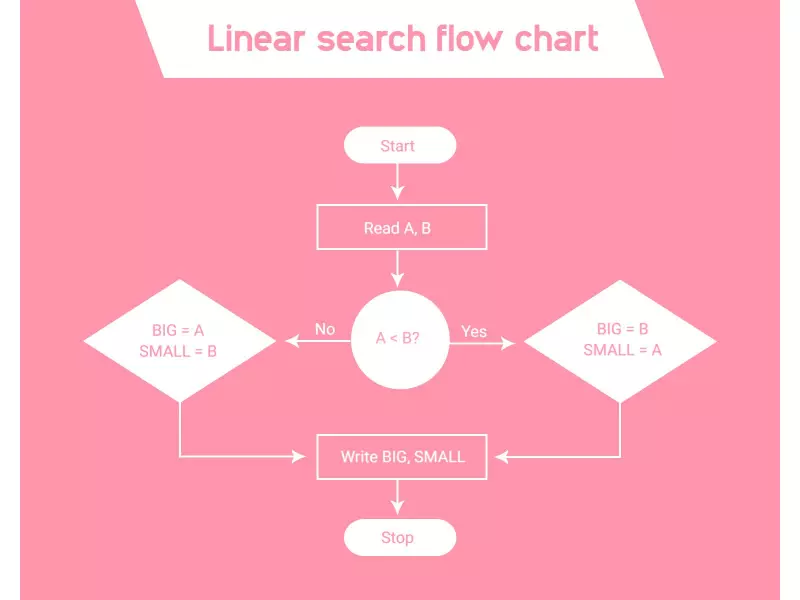
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
|  | Data Structures And Algorithms  Fatimaa Khan |  |  |  |

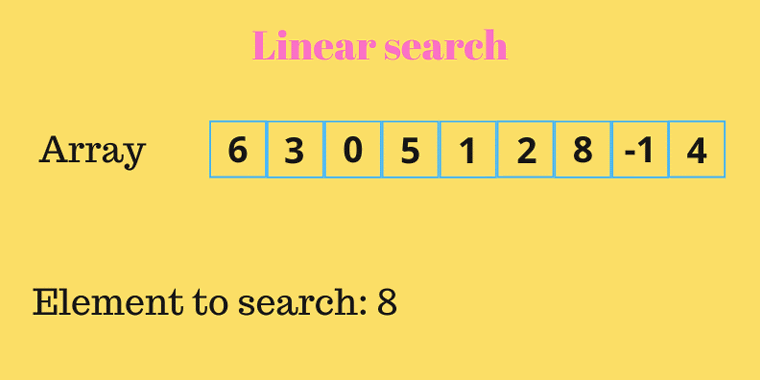
|  |  |  |
| --- | --- | --- |
|  | | |
|  | Data StructuresData = data is a collection of discrete or continuous values that convey informationStructure = The representation of data is known as structureData Structure = A data structure is a specialized format for organizing, processing, storing and retrieving data. |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Data Structures & Algorithms for humans | by Mohit Chawla | The Bit Theories |  | Difference Between Linear And Non-Linear Data Structures // Unstop |  |
|  |  |  |  |  |
|  | Primitive Data Structures Primitive data structures are the basic data types that are built into the programming language. They include types such as integers, floats, characters, and booleans. Non-Primitive Data Structures Non-Primitive data structure is a data structure that allows you to store multiple data type values. There are two types of non-primitive data structures:   1. Linear 2. Non-Linear |  | Linear Vs Non-Linear Data Structures Linear Data Structures: Data structure where data elements are arranged sequentially or linearly where each and every element is attached to its previous and next adjacent is called a linear data structure. Examples include Array, Stack, Queue Linked Lists.  Non-Linear Data Structures: Data structures where data elements are not arranged sequentially or linearly are called non-linear data structures. In a non-linear data structure, single level is not involved. Therefore, we can’t traverse all the elements in single run only. Examples include trees and graphs. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| A | An algorithm is a step-by-step procedure or set of rules to solve a problem efficiently. |  |  |

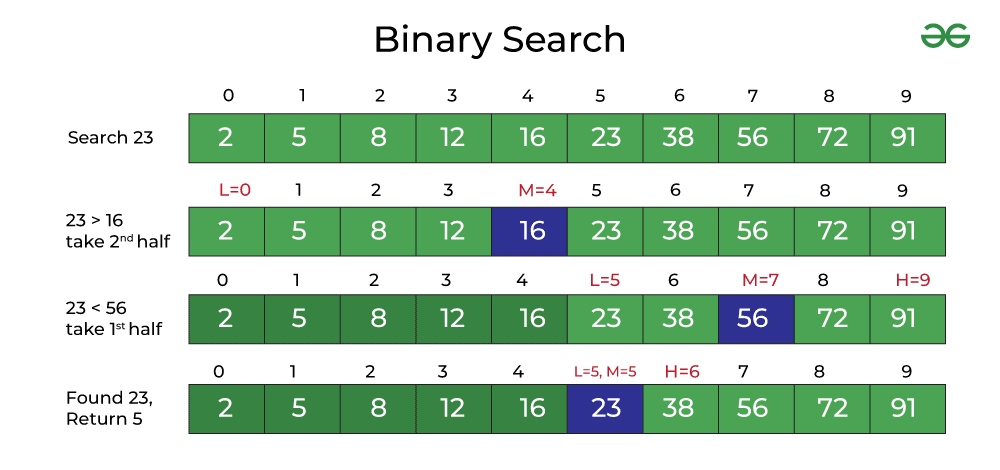
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Big O Notation Big-O notation is the language we use for talking about how long an algorithm takes to run (time complexity) or how much memory is used by an algorithm (space complexity).  All you need to know about “Big O Notation” to crack your next coding interview | | | | | | | | | | | | | |  |
|  |  |  |  |  | |  | |  | |  |  | |  |  |  |  |
|  | Linear Search O(n)  A linear search examines each element until it finds a match, starting at the beginning of the data set, until the end.  (Worse case scenario is if we check the whole array and target number isn’t found hence, number of searches becomes size of array!!) | | | |  | | Bubble Sort O(n²)  Bubble sort is a simple sorting algorithm that repeatedly steps through the input list element by element, comparing the current element with the one after it, swapping their values if needed. | | | | | Binary Search O(log n)  Binary Search Algorithm is a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. | | | |  |
|  | Linear Search Algorithm   * Start: Begin at the first element. * Compare: Compare the current element with the desired element. * Found: If the current element is equal to the desired element, return true or index to the current element. * Move: Otherwise, move to the next element in the collection. * Repeat: Repeat steps 2-4 until we have reached the end of collection. * Not found: If the end of the collection is reached without finding the desired element, return that the desired element is not in the array. | | | |  | |  | |



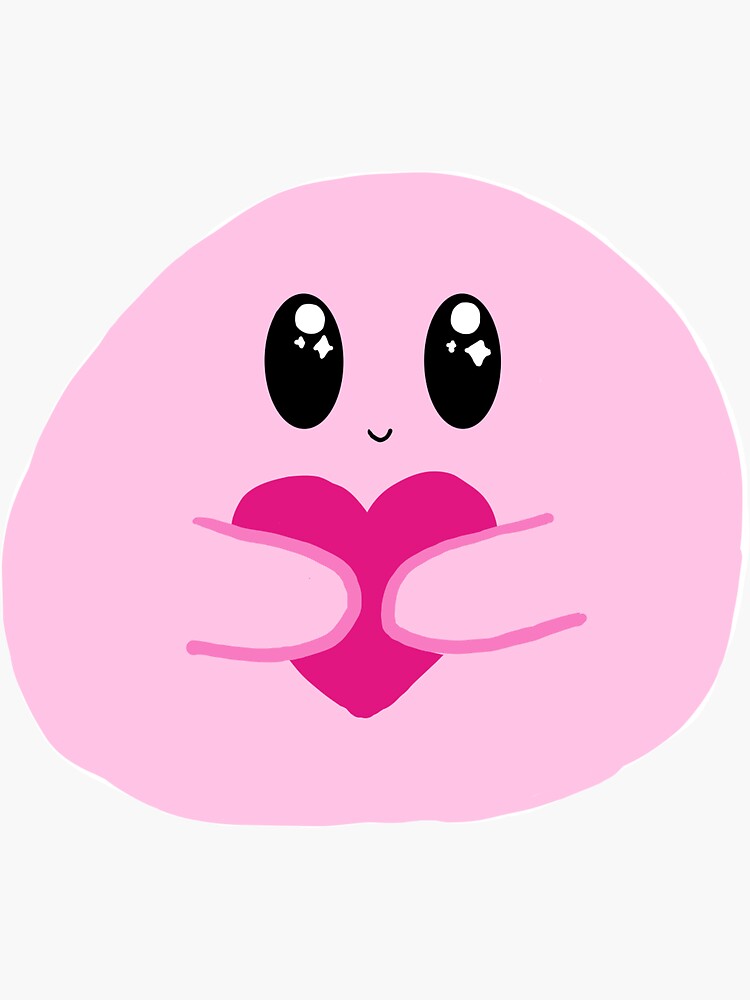


Binary Search Algorithm

* Divide the search space into two halves by finding the middle index.
* Compare the middle element of the search space with the **key**.
* If the **key**is found at middle element, the process is terminated.
* If the **key**is not found at middle element, choose which half will be used as the next search space.
  + If the **key**is smaller than the middle element, then the **left**side is used for next search.
  + If the **key**is larger than the middle element, then the **right**side is used for next search.
* This process is continued until the **key**is found or the total search space is exhausted.



Data Structure Operations:



Fatimaa Khan

CT-23024

1. Insertion: Adding data in the data structure.
2. Deletion: removing data from a data structure.
3. Sorting: Arrange data in increasing or decreasing order.
4. Searching: Finding the location of data in a data structure.
5. Merging: Combining the data of two different sorted files into a single sorted file.
6. Traversing: Accessing each data exactly once in the data structure so that each data item is traversed or visited.