Terms, Concepts, and Examples

- An **algorithm** is a step-by-step process, defined by a set of instructions to be executed sequentially to achieve a specified task, producing a determined output.

  Examples of common discrete math algorithms include:

  - Searching Algorithms to search for an item in a data set like a list or data structure like a tree.
  - Sorting Algorithms to sort items in a specific way.
  - Insertion and Deletion Algorithms to insert or delete an item in a tree or list.
  - Division Algorithms such as a procedure for dividing two integers or the Euclidean Algorithm to determine the greatest common divisor between two integers.
  - Optimization algorithms such as finding the line of best fit of set of points, or finding the nearest neighbor in a set of points to a given point.

  Many of the functions we saw and created in the Python module are examples of algorithms. One thing to note is that an algorithm should always end, or be finite. Also, all variables should be defined and used in the algorithm.

- **Searching Algorithms** - The problem of locating an element in an ordered list occurs in many contexts. For instance, a program that checks the spelling of words searches for them in a dictionary, which is just an ordered list of words. Problems of this kind are called searching problems.

  - The **Linear Search algorithm**, or sequential search, begins by comparing x and the first element in the list. If it find the element it wants there, then the algorithm returns 0 as the location of the element in the list. If not, compare x with a[1]. If you find the element you want, then the algorithm returns 1 as the location of the element. Continue this process, comparing x successively with each term of the list until a match is found. The return value will be the location of that element, unless no match occurs. If the entire list has been searched without locating x, the return is -1.

    ```
    def Linear_Search(a, x):
        for i in range(0, len(a)):
            if a[i] == x:
                return i
        return -1
    ```

    Linear Search Dance - A visual example of performing a linear search.
    Linear Search with Python Tutor

  - The second searching algorithm is called the **Binary Search algorithm**. A binary search can only be used when the list has terms occurring in order of increasing size. It proceeds by comparing the element t to be located to the "middle term" of

the list. The list is then split into two smaller sublists of the same size, or where one of these smaller lists has one fewer term than the other. The search continues by restricting the search to the appropriate sublist based on the comparison of the element t to be located and the middle term. Just as in a linear search the algorithm returns the location of the element (if it is found) and -1 if the element is not in the list.

```
def BinarySearch(a, t):
    L = 0
    R = len(a) - 1
    while L <= R:
        m = floor((L + R) / 2)
        if a[m] < t:
            L = m + 1
        elif a[m] > t:
            R = m - 1
        else:
            return m
    return -1
```

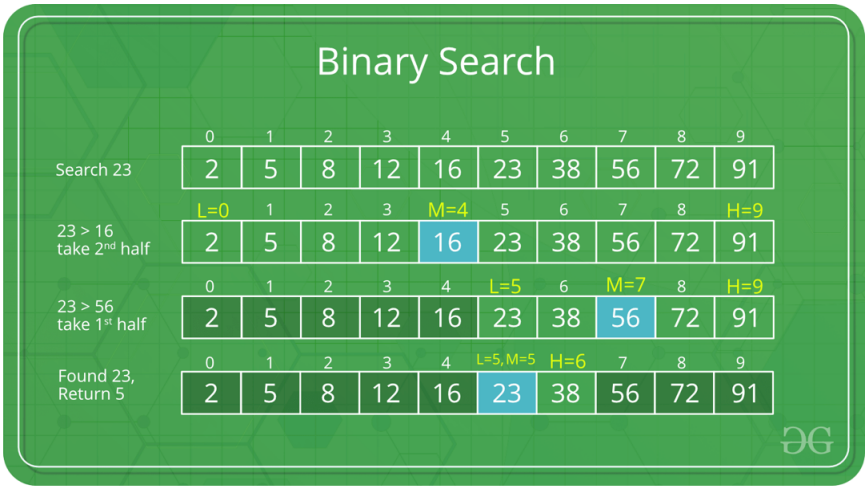
Binary Search Dance - A visual example of performing a binary search.

Binary Search on Python Tutor

- **"Tracing"** an algorithm is a way of listing all the steps taken when the algorithm gets applied. We can do this in a variety of ways. Sometimes we might list what happens at each step, other times we may give the state of a list at each particular step.

  Example: Use a Binary Search to find the position of 23 in the list [2, 5, 8, 12, 16, 23, 38, 56, 72, 91].

  Image from Geeks for Geeks



  Notice the important pieces of each line.

  – At the beginning it is checking a condition.

  – It gives the values of L, M and H at the current step.

  – It darkens a portion of the list that is no longer under consideration.

  – The return value is stated.

Video Example of Tracing an Algorithm

Practice Problems

1. Use Linear Search and "trace the algorithm" on these lists to search for the integer 5.
   (a) $[1, 6, 9, 23, 5, 3]$
   (b) $[2, 5, 8, 100, 10, 28, 99]$
2. Use Binary search and "trace the algorithm" on these lists searching for the integer 5
   (a) $[1, 2, 3, 4, 5, 6, 7, 8]$
   (b) $[1, 5, 8, 9, 10, 12, 14, 15, 18, 25, 31]$
   (c) $[1, 2, 4, 5]$