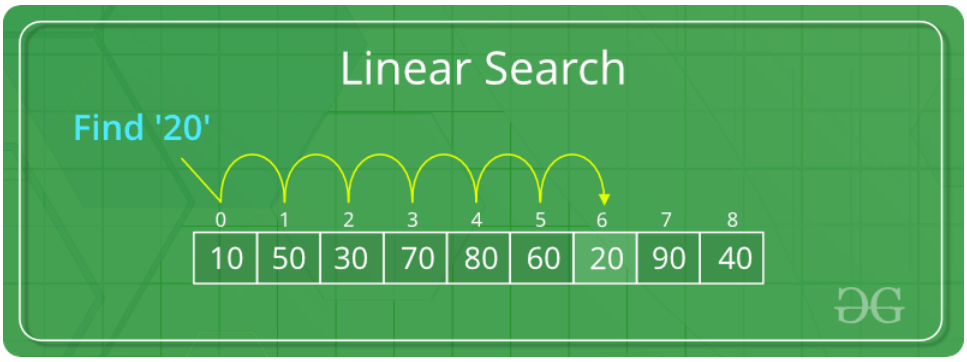
Searching Algorithms

Friday, 8 September 2023 10:24 AM

Linear Search



Binary Search

https://www.programiz.com/dsa/binary-search

Binary search (*or half interval search*) algorithm is a searching method used only in sorted arrays. It relies *on divide and conquer strategy* to accomplish its purpose. In every search iteration, half of the elements of the array are eliminated as possible solutions. Binary search is very efficient for large arrays. Its performance makes it ideal when resorting is not required.

In each iteration, the algorithm

1. Compares the search value with the value of the middle element of the array.
    a. If the values match, then the value was found .
    b. If the search value is less than the middle element of the array,
        - then the algorithm repeats its action on the sub-array to the left of the middle element.
    c. if the search value is greater than the middle element of the array,
        - then the algorithm repeats its action on the sub-array to the right of the middle element.
2. If the remaining array to be searched is empty, then the value was not found.

Programming Example 20: Binary search

```
//==== Binary Search =====
VALUES = [11,12,15,16,112,118,123,145] //sorted array elements
TARGET = 15 //search value
MIN = 0
HIGH = 7 // Number of array elements - 1
FOUND = false
ANSWER = 0
MID =0

loop while FOUND = true AND MIN <= HIGH
  MID = ((MIN + HIGH) div 2)
  if VALUES[MID] = TARGET then
      FOUND = true
      ANSWER = MID
  else if TARGET > VALUES[MID] then
     MIN = MID + 1
  else
     HIGH = MID - 1
  end if
end while
if FOUND = true then
   output TARGET , "FOUND AT ARRAY INDEX" , ANSWER
else
   output TARGET , " was not found"
end if
```

Output: 15 FOUND AT ARRAY INDEX 2

Comparison table of linear search and binary search

Binary search	**Linear search**
Works only on sorted elements	Works on sorted as well as unsorted items.
Generally number of comparisons are less	Efficient for few elements Efficient if the element to be found is located in the beginning of the array or list Generally more number of comparisons are required if the element to be found is not present in the beginning of the array or list
Time complexity: O(log n)	**Time complexity: O(n)**