DATA STRUCTURE - INTERPOLATION SEARCH

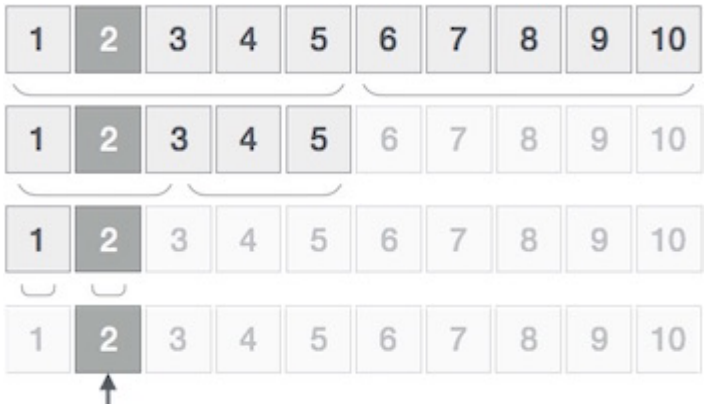
Interpolation search is an improved variant of binary search. This search algorithm works on the probing position of required value. For this algorithm to work properly the data collection should be in sorted form and equally distributed.

Binary search has huge advantage of time complexity over linear search. Linear search has worst-case complexity of O*n* whereas binary search has O*logn*.

There are cases where the location of target data may be known in advance. For example, in case of telephone directory, if we want to search telephone number of Morphius. Here, linear search and even binary search will seem slow as we can directly jump to memory space where names start from 'M' are stored.

Positioning in Binary Search

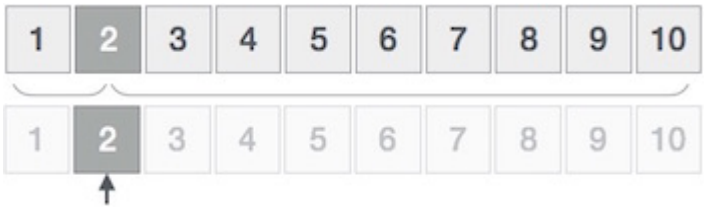
In binary search, if the desired data is not found then the rest of the list is divided in two parts, lower and higher. Then then search is done in either of them.



Even when the data is sorted, binary search does not take advantage of that to probe the position of desired data.

Position Probing in Interpolation Search

Interpolation search search a particular item by computing the probe position. Initially probe position is the position of the middle most item of the collection.



If match occurs then index of item is returned. To split the list into two parts we use the following method −

```
mid = Lo + ((Hi - Lo) / (A[Hi] - A[Lo])) * (X - A[Lo])

where −
   A    = list
   Lo   = Lowest index of the list
   Hi   = Highest index of the list
   A[n] = Value stored at index n in the list
```

If middle item is greater than item then probe position is again calculated in the sub-array to the right of the middle item other wise item is search in sub-array to the left of the middle item. This

process continues on sub-array as well until the size of subarray reduces to zero.

Runtime complexity of interpolation search algorithm is $O(log(logn))$ as compared to $O(logn)$ of BST in favourable situations.

Algorithm

As its an improvisation of existing BST algorithm, we are mentioning steps to search 'target' data value index, using position probing −

```
Step 1  − Start searching data from middle of the list.
Step 2  − If it is a match, return the index of the item, and exit.
Step 3  − If it is not a match, probe position.
Step 4  − Divide the list using probing formula and find the new midle.
Step 5  − If data is greater than middle, search in higher sub-list.
Step 6  − If data is smaller than middle, search in lower sub-list.
Step 7  − Repeat until match.
```

Pseudocode

```
A → Array list
N → Size of A
X → Target Value

Procedure Interpolation_Search()

   Set Lo  →  0
   Set Mid → -1
   Set Hi  →  N-1

   While X does not match

      if Lo equals to Hi OR A[Lo] equals to A[Hi]
         EXIT: Failure, Target not found
      end if

      Set Mid = Lo + ((Hi - Lo) / (A[Hi] - A[Lo])) * (X - A[Lo])

      if A[Mid] = X
         EXIT: Success, Target found at Mid
      else
         if A[Mid] < X
            Set Lo to Mid+1
         else if A[Mid] > X
            Set Hi to Mid-1
         end if
      end if

   End While

End Procedure
```

To see implementation of interpolation search in c programming language, click here.