Data Structure-Search

Kaustuv Bhattacharjee
University of Engineering & Management, Kolkata

Search-Introduction

- Searching means to find whether a particular value is present in a collection of data or not
- If the value is present in the collection, then searching is said to be successful and the searching process gives the location of that value in the array
- If the value is not present in the collection, the searching process displays an appropriate message and in this case searching is said to be unsuccessful

Search-Methods

- Linear Search
- Binary Search
- Interpolation Search

Linear Search

- (Also known as sequential search)
- Compare the value to be searched with every element of the array one by one in a sequence until a match is found
- Mostly used in an unordered list (data elements are not sorted) of elements

Linear Search-Algorithm

```
LINEAR_SEARCH(A, N, VAL)
Step 1: [INITIALIZE] SET POS = -1
Step 2: [INITIALIZE] SET I = 1
Step 3:    Repeat Step 4 while I<=N
Step 4:               IF A[I] = VAL
                           SET POS = I
                           PRINT POS
                           Go to Step 6
                      [END OF IF]
                      SET I = I + 1
           [END OF LOOP]
Step 5: IF POS = –1
PRINT “VALUE IS NOT PRESENT IN THE ARRAY”
STEP 6: EXIT
```

Linear Search-Time Complexity

- Linear search executes in O(n) time where n is the number of elements in the array.
- The best case of linear search is when VAL is equal to the first element of the array. In this case, only one comparison will be made.
- Likewise, the worst case will happen when either VAL is not present in the array or it is equal to the last element of the array. In both the cases, n comparisons will have to be made.

Binary Search

- A searching algorithm that works efficiently with a sorted list
- Repeatedly divide the search interval in half
- If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half.
- Otherwise narrow it to the upper half.
- Repeatedly check until the value is found or the interval is empty.

Binary Search-Example

- Consider an array A[] that is declared and initialized as

int A[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

and the value to be searched is VAL = 9.

- The algorithm will proceed in the following manner:

BEG = 0, END = 10, MID = (0 + 10)/2 = 5

- Now, VAL = 9 and A[MID] = A[5] = 5. A[5] is less than VAL, therefore, search for the value in the second half of the array. So, change the values of BEG and MID.
- Now, BEG = MID + 1 = 6, END = 10, MID = (6 + 10)/2 =16/2 = 8

VAL = 9 and A[MID] = A[8] = 8

A[8] is less than VAL, therefore, search for the value in the second half of the segment.

So, again change the values of BEG and MID.

Now, BEG = MID + 1 = 9, END = 10, MID = (9 + 10)/2 = 9

Now, VAL = 9 and A[MID] = 9.

So the element is found at position MID, i.e. 9.

Binary Search-Algorithm

```
BINARY_SEARCH(A, lower_bound, upper_bound, VAL)
Step 1: [INITIALIZE] SET BEG = lower_bound
            END = upper_bound, POS = - 1
Step 2:     Repeat Steps 3 and 4 while BEG <= END
Step 3:     SET MID = (BEG + END)/2
Step 4:      IF A[MID] = VAL
                        SET POS = MID
                        PRINT POS
                        Go to Step 6
            ELSE IF A[MID] > VAL
                        SET END = MID - 1
            ELSE
                        SET BEG = MID + 1
            [END OF IF]
    [END OF LOOP]
Step 5: IF POS = -1
    PRINT “VALUE IS NOT PRESENT IN THE ARRAY”
[END OF IF]
Step 6: EXIT
```

Binary Search-Time Complexity

- The complexity of the binary search algorithm can be expressed as f(n), where n is the number of elements in the array.
- The complexity of the algorithm is calculated depending on the number of comparisons that are made.
- In the binary search algorithm, we see that with each comparison, the size of the segment where search has to be made is reduced to half.
- Thus, we can say that, in order to locate a particular value in the array, the total number of comparisons that will be made is given as $2^{f(n)} = n$ or $\log_2 n$

Interpolation Search

- (Also known as Extrapolation search)
- A searching algorithm that works efficiently with a sorted list
- In each step of interpolation search, the remaining search space for the value to be found is calculated.
- The calculation is done based on the values at the bounds of the search space and the value to be searched.
- The value found at this estimated position is then compared with the value being searched for.
- If the two values are equal, then the search is complete.
- In case the values are not equal then depending on the comparison, the remaining search space is reduced to the part before or after the estimated position.

Interpolation Search-Example

- Given a list of numbers a[] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21}. Search for value 19 using interpolation search technique.
- Solution

  Low = 0, High = 10, VAL = 19, a[Low] = 1, a[High] = 21

  Middle = Low + (High – Low)×((VAL – a[Low]) /(a[High] – a[Low]))

  = 0 +(10 – 0) × ((19 – 1) / (21 – 1))

  = 0 + 10 × 0.9 = 9

  a[middle] = a[9] = 19 which is equal to value to be searched.

Interpolation Search-Algorithm

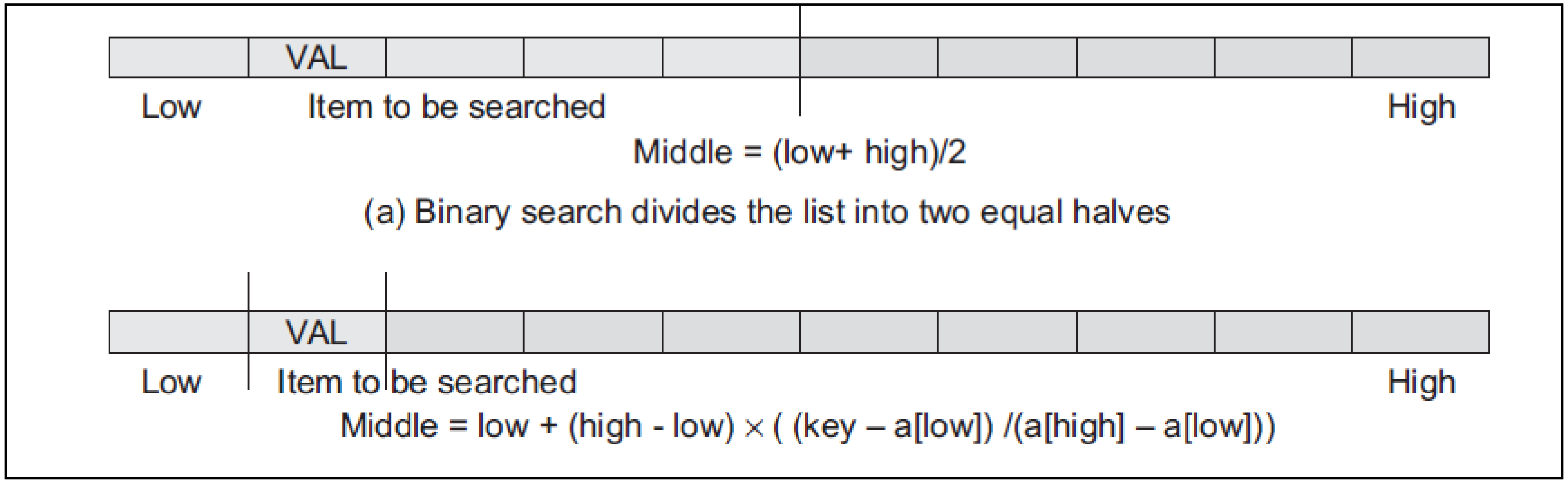
```
INTERPOLATION_SEARCH (A, lower_bound, upper_bound, VAL)
Step 1: [INITIALIZE] SET LOW = lower_bound, HIGH = upper_bound, POS = –1
Step 2: Repeat Steps 3 to 4 while LOW <= HIGH
Step 3:         SET MID = LOW + (HIGH – LOW) × ((VAL – A[LOW]) / (A[HIGH] – A[LOW]))
Step 4:         IF VAL = A[MID]
                    POS = MID
                    PRINT POS
                    Go to Step 6
                ELSE IF VAL < A[MID]
                    SET HIGH = MID – 1
                ELSE
                    SET LOW = MID + 1
                [END OF IF]
        [END OF LOOP]
Step 5: IF POS = –1
            PRINT "VALUE IS NOT PRESENT IN THE ARRAY"
        [END OF IF]
Step 6: EXIT
```

Interpolation Search-Time Complexity

- When n elements of a list to be sorted are uniformly distributed (average case), interpolation search makes about log(log n) comparisons.
- In the worst case, that is when the elements increase exponentially, the algorithm can make up to O(n) comparisons.

Binary Search & Interpolation Search-Comparison

- Binary search always selects the middle value of the remaining search space and discards half of the values based on the comparison between the value found at the estimated position and the value to be searched.
- In interpolation search, interpolation is used to find an item near the one being searched for, and then linear search is used to find the exact item.



VAL
Low
Item to be searched
High
Middle = (low+ high)/2
(a) Binary search divides the list into two equal halves
VAL
Low
Item to be searched
High
Middle = low + (high - low) × ((key – a[low]) /(a[high] – a[low]))

Assignments

- Write a program to search an element in an array using linear search.
- Write a program to search an element in an array using binary search.
- Write a program to search an element in an array using interpolation search.