Ernesto G. Alvarado

Professor Yuen Yuen

Co. Sci. 942

This Binary Search Algorithm can be used when the list is sorted in order of increasing size. For example, if the elements are numbers, they are listed from smallest to largest and if they are words, they are listed in alphabetical order. It begins by comparing the element to be located to the middle term of the list. The list is then split into two smaller sub-lists 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 sub-list based on the comparison of the element to be located and the middle term. The binary search algorithm is much more efficient than the linear search algorithm, because it splits the list into sub-list and restricts the search to the sub-list that contain the element. Whereas the linear search algorithm at worst case would have to search the entire list.

For example, to search for 6 in the list

1 2 3 4 5 6 7 8 9 10,

first split this list, which has 10 terms, into two smaller sub-lists with five terms each,

1 2 3 4 5 6 7 8 9 10.

Then it compares 6 and the largest term in the first list. Because 6 > 5, the search for 6 can be restricted to the list containing the 6th through the 10th terms of the original list. Next, split this list, which has five terms, into the two smaller lists of two or three terms each,

6 7 8 9 10.

Because 6 < 7 (comparing 6 with the largest term of the first list) the search is restricted to the first of these lists, which contains the 6th and 7th terms of the original list. The list 6 7 is split into two lists,

6 7.

Next, this list of two terms is split into two lists of one term each: 6 and 7. Because 6 < 7, the search is restricted to the first list: the list containing the 6th term of the list, which is 6. Now that the search has been narrowed down to one term, a comparison is made, and 6 is located as the 6th term in the original list.

**Pseudocode for the Binary Search Algorithm.**

**procedure** *binary search* (x: integer, a1, a2, . . . , an: increasing integers)

i := 1{i is left endpoint of search interval}

j := n {j is right endpoint of search interval}

**while** i < j

m := ⌊(i + j)/2⌋

**if** x > am **then** i :=m+1

**else** j := m

**if** x = ai **then** *location* := i

**else** *location* := 0

**return** *location*{*location* is the subscript i of the term ai equal to x, or 0 if x is not found}