Binary Search

Look at middle element of sorted array in range [first, last) until the target is found or run out of elements. Return index of target or last.

Analysis (same parameters as Sequential Search)

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>Kind</th>
<th>Movement</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>array of values</td>
<td>int[]</td>
<td>varying</td>
<td>received</td>
<td>arr</td>
</tr>
<tr>
<td>index of lower bound</td>
<td>int</td>
<td>varying</td>
<td>received</td>
<td>first</td>
</tr>
<tr>
<td>index of upper bound</td>
<td>int</td>
<td>varying</td>
<td>received</td>
<td>last</td>
</tr>
<tr>
<td>search target</td>
<td>int</td>
<td>varying</td>
<td>received</td>
<td>target</td>
</tr>
<tr>
<td>index of target</td>
<td>int</td>
<td>varying</td>
<td>returned</td>
<td>——</td>
</tr>
<tr>
<td>index of midpoint</td>
<td>int</td>
<td>varying</td>
<td>local</td>
<td>mid</td>
</tr>
<tr>
<td>value at midpoint</td>
<td>int</td>
<td>varying</td>
<td>local</td>
<td>midValue</td>
</tr>
<tr>
<td>original last index</td>
<td>int</td>
<td>varying</td>
<td>local</td>
<td>origLast</td>
</tr>
</tbody>
</table>

Algorithm

1. Initialize origLast to last

2. While first < last do

   2.1 Compute mid = (first+last)/2

   2.2 Set midValue to arr[mid]

   2.3 If target equals midValue then (found a match)

       2.3.1 Return mid

   2.4 Else if target less than midValue then (in lower half)

       2.4.1 Set last to mid

   2.5 Else (in upper half)

       2.5.1 Set first to mid+1

3. Return origLast (target not found)
int BinSearch (const int arr[], int first, int last, int target)
{
    int mid, // index of midpoint
        midValue, // value at midpoint
        origLast = last; // save original value of last

    // while the sublist is not empty
    while (first < last)
    {
        mid = (first+last)/2;
        midValue = arr[mid];
        if (target==midValue) // found a match
            return mid;
        else if (target < midValue) // search lower half
            last = mid;
        else // search upper half
            first = mid+1;
    } // end while

    return origLast; // target not found
} // end BinSearch